Modeling of the Unlv-Aro light-weight robot using Lagrange's assumed mode method, symbolic manipulation, and numerical simulation

Edward Yuk-Kam Lam
University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation
https://digitalscholarship.unlv.edu/rtds/135

This Thesis is brought to you for free and open access by Digital Scholarship@UNLV. It has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI
University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700  800/521-0600
Modeling of the UNLV-ARO light-weight robot using Lagrange’s assumed mode method, symbolic manipulation, and numerical simulation

Lam, Edward Yuk-Kam, M.S.
University of Nevada, Las Vegas, 1991
MODELING OF THE UNLV-ARO LIGHT-WEIGHT ROBOT USING LAGRANGE'S ASSUMED MODE METHOD, SYMBOLIC MANIPULATION, AND NUMERICAL SIMULATION

by
Edward Y. Lam

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

Master of Science

in
Mechanical Engineering

Department of Mechanical Engineering
University of Nevada, Las Vegas
May, 1991

This Research was Supported by the U.S. Army Research Office Under Grant No. DAAL03-87-G-0004
The thesis of Mr. Edward Y. Lam for the degree of Master of Science in Mechanical Engineering is approved.

Chairperson, Professor Mohamed B.E. Trabia, Ph.D.

Examiner Committee Member, Professor Douglas D. Regnolds, Ph.D.

Examiner Committee Member, Professor Woosoon Yim, Ph.D.

Graduate Faculty Representative, Professor Samaan G. Ladkany, Ph.D.

Graduate Dean, Professor Ronald W. Smith, Ph.D.

University of Nevada, Las Vegas
May, 1991
This thesis modeled the dynamics of the UNLV-ARO serially connected light-weight robot. The two outer-most links of the robot exhibited flexible behavior in the in-plane and the out-of-plane directions. In the latter, the torsional vibrations of the system was neglected. The robot was actuated by hydraulic cylinders. Lagrangian mechanics were used to model this system. Assumed modes method was used to approximate the vibration mode shape of the links. Different mode shapes were used to model the vibration modes. Using these analytical techniques in conjunction with the inverse dynamics method, a set of equations of motion was formulated to describe the vibrations of the robot. For most rigid multi-link robots, the equations of motion are fairly complicated. The complexity of these equations were increased dramatically with the inclusion of flexible links. To illustrate the principle behind this modeling scheme, two simpler cases were considered before the modeling of the UNLV-ARO robot. The equations of motion were derived using MACSYMA, a symbolic software. MACSYMA programs modules were written for the kinematics and dynamics of the robot. From MACSYMA, the equations of motion were translated into FORTRAN statements. In order to produce analytical results, FORTRAN programs were written using these equations. Since the equations of motion constituted a set of second order differential equations, subroutine `ivpag` of IMSL software package was used to solve these equations numerically. The scheme discussed in this thesis provided a step towards modeling the dynamics of a light-weight robot accurately. The results obtained demonstrate the vibration behavior of such a robot.
ACKNOWLEDGEMENTS

I would like to dedicate my completion of my master education to my sister Amy Choy and brother-in-law Johnson Choy, who have opened the world of opportunities for me.

I am very fortunate to be able to study under the guidance of some of the finest teachers, and I would like to express my gratitude to the professors in the Civil and Mechanical Engineering Department of UNLV. I am very thankful to Dr. Mohammed B. Trabia, my academic advisor, for his effort and his supervision during my master study. To Dr. Douglas D. Reynolds for assisting me into the graduate school. To Dr. Richard V. Wyman for his assistance and his support as the department chairman. Also, I am thankful to the Army Research Office (ARO) for funding this research project and to the research members for their assistance.

I am grateful to my sister, Lai-Ha Lam, for her support during our tough times together. To my friends Douglas S. Couch, James J. Ventresca and their families for their kindness and for treating me as part of their family. I am indebted to Mrs. Phyllis M. Silvaggio, Mr. Danny S. Molner, and Mrs. Mary Molner for assisting me throughout the years without reservation. And to Mr. and Mrs. Edwin L. Davenport for their patience and continuous support.

To Andreas O. Ranz, may you rest in peace.

Edward Y. Lam
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>iv</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>x</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>xv</td>
</tr>
<tr>
<td>NOMENCLATURE</td>
<td>xvi</td>
</tr>
<tr>
<td>CHAPTER 1  INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>CHAPTER 2  OVERVIEW</td>
<td>6</td>
</tr>
<tr>
<td>2.1 Fundamental Assumptions</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Flexible Robot's Generalized Coordinates</td>
<td>7</td>
</tr>
<tr>
<td>2.3 Lagrangian Mechanics</td>
<td>11</td>
</tr>
<tr>
<td>2.4 Symbolic Derivation of the Equations of Motion</td>
<td>15</td>
</tr>
<tr>
<td>2.5 Inverse Dynamics</td>
<td>16</td>
</tr>
<tr>
<td>2.5.1 Time History of the Joints Motion</td>
<td>17</td>
</tr>
<tr>
<td>2.5.2 Use of IMSL for Solving the Equations of Motion of Flexible Robots</td>
<td>26</td>
</tr>
<tr>
<td>2.6 Case Studies</td>
<td>28</td>
</tr>
</tbody>
</table>
CHAPTER 3 MODELING OF A TWO-LINK ROBOT WITH ONE FLEXIBLE OUTER-MOST LINK ................................................. 29

3.1 Kinematic Analysis of a Two-link Robot with One Flexible Outer-most Link .................................................................................................................................................. 31

3.1.1 Kinematic Analysis of Link 1 ........................................................................ 31

3.1.1.1 Displacement Vector of Link 1 ............................................... 31

3.1.1.2 Velocity Vector of Link 1 ......................................................... 32

3.1.2 Kinematic Analysis of Link 2 ........................................................................ 33

3.1.2.1 Displacement Vectors of Link 2 ............................................. 33

3.1.2.2 Velocity Vectors of Link 2 ...................................................... 35

3.2 Dynamic Analysis of a Two-link Robot with One Flexible Outer-most Link .................................................................................................................................................. 36

3.2.1 Kinetic and Potential Energy Equations of Link 1 .................................. 36

3.2.2 Kinetic and Potential Energy Equations of Link 2 .................................. 37

3.2.3 Lagrangian Equations ...................................................................................... 39

3.3 Initial Conditions .............................................................................................................. 41

3.4 MACSYMA Programs ................................................................................................. 41

3.4.1 MACSYMA Program Modules for the Derivation of the Equations of Motion for a Two-link Robot with One Flexible Outer-most Link .................................................................................................................................................. 42

3.4.2 Symbolic Derivation ....................................................................................... 45

3.5 Illustrative Examples ...................................................................................................... 48

CHAPTER 4 MODELING OF A THREE-LINK ROBOT WITH TWO FLEXIBLE OUTER-MOST LINKS ............................................. 60

4.1 Kinematic Analysis of a Three-link Robot With Two Flexible Outer-most Links .................................................................................................................................................. 62

4.1.1 Kinematic Analysis of Link 1 ........................................................................ 62

vi
5.1.2.2 Velocity Vectors of Link 2 ..................................................... 105
5.1.3 Kinematic Analysis of Link 3 ......................................................... 106
  5.1.3.1 Displacement Vector of Link 3 .............................................106
  5.1.3.2 Velocity Vector of Link 3 .......................................................107
5.1.4 Kinematic Analysis of Link C ....................................................... 108
  5.1.4.1 Displacement Vector of Link C .............................................108
  5.1.4.2 Velocity Vector of Link C ......................................................109
5.1.5 Kinematic Analysis of Link H3 ..................................................... 109
  5.1.5.1 Displacement Vectors of Link H3 ......................................... 112
  5.1.5.2 Velocity Vectors of Link H3 ..................................................117
5.2 Dynamic Analysis of the UNLV-ARO Light-weight Robot .................. 118
  5.2.1 Kinetic and Potential Energy Equations of Link 1 ..................... 118
  5.2.2 Kinetic and Potential Energy Equations of Link 2 ..................... 118
  5.2.3 Kinetic and Potential Energy Equations of Link 3 ..................... 119
  5.2.4 Kinetic and Potential Energy Equations of Link C ..................... 120
  5.2.5 Kinetic and Potential Energy Equations of Link H3 ................... 120
  5.2.6 Lagrangian Equations .............................................................. 122
5.3 Initial Conditions .............................................................................. 123
5.4 Analysis of the Force in the Hydraulic Cylinders ............................... 129
  5.4.1 Piston Force Experienced by Hydraulic Cylinder 1 .................... 129
  5.4.2 Piston Force Experienced by Hydraulic Cylinder 2 .................... 129
  5.4.3 Piston Force Experienced by Hydraulic Cylinder 3 .................... 131
5.5 MACSYMA Programs ...................................................................... 132
  5.5.1 MACSYMA Program Modules for the Derivation of the Equations of
        Motion for the UNLV-ARO Light-weight Robot .......................... 135
  5.5.2 Symbolic Derivation .................................................................. 137
5.6 Illustrative Example ........................................................................... 140
CHAPTER 6 CONCLUSIONS AND RECOMMENDATIONS .................. 169

6.1 Conclusions ................................................................................................................. 169
6.2 Recommendations ........................................................................................................ 173

APPENDIX A KINEMATIC REPRESENTATION OF ROBOT WITH SOME FLEXIBLE LINKS USING TRANSFORMATION MATRICES ................................................................................................................ 175

APPENDIX B ASSUMED MODES METHOD ................................................................. 184

APPENDIX C ADMISSIBLE FUNCTIONS ................................................................. 186
C.1 Case 1. Flexible Beam With Shearing Mass .......................................................... 186
C.2 Case 2. Flexible Beam With Shearing Mass and Bending Moment ............. 195
C.3 Case 3. Flexible Beam With Joint End ............................................................... 202
C.4 Case 4. Flexible Beam With Constrained End ................................................... 205

APPENDIX D MACSYMA PROGRAM LISTINGS OF THE TWO–LINK ROBOT WITH ONE FLEXIBLE OUTER–MOST LINK . 210

APPENDIX E MACSYMA PROGRAM LISTINGS OF THE THREE–LINK ROBOT WITH TWO FLEXIBLE OUTER–MOST LINK 225

APPENDIX F MACSYMA PROGRAM LISTINGS OF THE UNLV–ARO LIGHT–WEIGHT ROBOT ......................................................... 256

APPENDIX G FORTRAN PROGRAM LISTINGS OF THE TWO–LINK ROBOT WITH ONE FLEXIBLE OUTER–MOST LINK . 335

APPENDIX H FORTRAN PROGRAM LISTINGS OF THE THREE–LINK ROBOT WITH TWO FLEXIBLE OUTER–MOST LINK 362

APPENDIX I FORTRAN PROGRAM LISTINGS OF THE UNLV–ARO LIGHT–WEIGHT ROBOT ......................................................... 401

REFERENCES .................................................................................................................. 472
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Fig.</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>UNLV—ARO Hydraulically Activated Robot</td>
<td>5</td>
</tr>
<tr>
<td>2.1a</td>
<td>Projection of a Flexible Link in X—Y Plane (In—plane Deflection)</td>
<td>9</td>
</tr>
<tr>
<td>2.1b</td>
<td>Projection of a Flexible Link in X—Z Plane (Out—of—plane Deflection)</td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>Sample Acceleration Profiles</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>Sample Velocity Profiles</td>
<td>19</td>
</tr>
<tr>
<td>2.4</td>
<td>Sample Displacement Profiles</td>
<td>20</td>
</tr>
<tr>
<td>3.1</td>
<td>Two—link Robot with One Flexible Outer—most Link</td>
<td>30</td>
</tr>
<tr>
<td>3.2</td>
<td>Flow Chart of the Macsyma Program Modules</td>
<td>43</td>
</tr>
<tr>
<td>3.3</td>
<td>Acceleration Profiles</td>
<td>52</td>
</tr>
<tr>
<td>3.4</td>
<td>Velocity Profiles</td>
<td>53</td>
</tr>
<tr>
<td>3.5</td>
<td>Displacement Profiles</td>
<td>54</td>
</tr>
<tr>
<td>3.6</td>
<td>End—effector Deflections Due to the Bang—bang and the Polynomial Acceler...</td>
<td>55</td>
</tr>
<tr>
<td>3.7</td>
<td>Torques at Joint 2 Due to the Bang—bang and the Polynomial Acceler...</td>
<td>56</td>
</tr>
<tr>
<td>3.8</td>
<td>End—effector Deflections Due to the Bang—bang and the Polynomial Acceler...</td>
<td>57</td>
</tr>
<tr>
<td>3.9</td>
<td>Torques at Joint 2 Due to the Bang—bang and the Polynomial Acceler...</td>
<td>58</td>
</tr>
<tr>
<td>4.1</td>
<td>Three—link Robot with Two Flexible Outer—most Links</td>
<td>61</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Fig. 4.2</td>
<td>Flow Chart of the Macsyma Program Modules</td>
<td>73</td>
</tr>
<tr>
<td>Fig. 4.3</td>
<td>Acceleration Profiles for Link 1</td>
<td>79</td>
</tr>
<tr>
<td>Fig. 4.4</td>
<td>Acceleration Profiles for Link 2</td>
<td>80</td>
</tr>
<tr>
<td>Fig. 4.5</td>
<td>Acceleration Profiles for Link 3</td>
<td>81</td>
</tr>
<tr>
<td>Fig. 4.6</td>
<td>Velocity Profiles for Link 1</td>
<td>82</td>
</tr>
<tr>
<td>Fig. 4.7</td>
<td>Velocity Profiles for Link 2</td>
<td>83</td>
</tr>
<tr>
<td>Fig. 4.8</td>
<td>Velocity Profiles for Link 3</td>
<td>84</td>
</tr>
<tr>
<td>Fig. 4.9</td>
<td>Displacement Profiles for Link 1</td>
<td>85</td>
</tr>
<tr>
<td>Fig. 4.10</td>
<td>Displacement Profiles for Link 2</td>
<td>86</td>
</tr>
<tr>
<td>Fig. 4.11</td>
<td>Displacement Profiles for Link 3</td>
<td>87</td>
</tr>
<tr>
<td>Fig. 4.12</td>
<td>Deflections at the End of Link 2 Due to the Bang—bang and the Polynomial Acceleration Profiles</td>
<td>88</td>
</tr>
<tr>
<td>Fig. 4.13</td>
<td>End-effector Deflections Due to the Bang—bang and the Polynomial Acceleration Profiles</td>
<td>89</td>
</tr>
<tr>
<td>Fig. 4.14</td>
<td>Torques at Joint 1 Due to the Bang—bang and the Polynomial Acceleration Profiles</td>
<td>90</td>
</tr>
<tr>
<td>Fig. 4.15</td>
<td>Torques at Joint 2 Due to the Bang—bang and the Polynomial Acceleration Profiles</td>
<td>91</td>
</tr>
<tr>
<td>Fig. 4.16</td>
<td>Torques at Joint 3 Due to the Bang—bang and the Polynomial Acceleration Profiles</td>
<td>92</td>
</tr>
<tr>
<td>Fig. 4.17</td>
<td>Trajectories of the End-effector Due to the Bang—Bang Acceleration Profiles with the Trajectory Projections on the X—Y, Y—Z, and Z—X Planes</td>
<td>93</td>
</tr>
<tr>
<td>Fig. 4.18</td>
<td>Trajectories of the End-effector Due to the Polynomial Acceleration Profiles with the trajectory projections on the X—Y, Y—Z, and Z—X Planes</td>
<td>94</td>
</tr>
<tr>
<td>Fig. 4.19</td>
<td>Errors on the X-axis</td>
<td>95</td>
</tr>
</tbody>
</table>
Fig. 5.15 Out-of-plane Deflections of Link H3 Due to the Bang-bang and the Polynomial Acceleration Profiles ......................................................... 150

Fig. 5.16 In-plane Deflections of Link 2 and Link 3. (Deflection Generated by the Bang-Bang Acceleration Profile) ..................................... 151

Fig. 5.17 Out-of-plane Deflections of Link 2 and Link 3. (Deflection Generated by the Bang-Bang Acceleration Profile) ........................ 152

Fig. 5.18 Out-of-plane Deflections of Link 2 and Link H3. (Deflection Generated by the Bang-Bang Acceleration Profile) .................... 153

Fig. 5.19 Torques at Joint 1 Due to the Bang-bang and the Polynomial Acceleration Profiles ................................................................. 154

Fig. 5.20 Torques at Joint 2 Due to the Bang-bang and the Polynomial Acceleration Profiles ................................................................. 155

Fig. 5.21 Torques at Joint 3 Due to the Bang-bang and the Polynomial Acceleration Profiles ................................................................. 156

Fig. 5.22 Forces at Joint 1 Due to the Bang-bang and the Polynomial Acceleration Profiles ................................................................. 157

Fig. 5.23 Forces at Joint 2 Due to the Bang-bang and the Polynomial Acceleration Profiles ................................................................. 158

Fig. 5.24 Forces at Joint 3 Due to the Bang-bang and the Polynomial Acceleration Profiles ................................................................. 159

Fig. 5.25 Trajectories of the End-effector Due to the Bang-Bang Acceleration Profiles with the Trajectory Projections on the X-Y, Y-Z, and Z-X Planes ................................................................. 160

Fig. 5.26 Steady-state Trajectory of the End-effector (Flexible) Using the Bang-bang Acceleration Profile with the Trajectory Projections on the X-Y, Y-Z, AND Z-X Planes ......................................................... 161
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.27</td>
<td>Trajectories of the End-effector Due to the Polynomial Acceleration Profiles with the trajectory projections on the X-Y, Y-Z, and Z-X Planes</td>
</tr>
<tr>
<td>5.28</td>
<td>Steady-state Trajectory of the End-effector (Flexible) Using the Polynomial Acceleration Profile with the Trajectory Projections on the X-Y, Y-Z, AND Z-X Planes</td>
</tr>
<tr>
<td>5.29</td>
<td>Errors on the X-axis</td>
</tr>
<tr>
<td>5.30</td>
<td>Errors on the Y-axis</td>
</tr>
<tr>
<td>5.31</td>
<td>Errors on the Z-axis</td>
</tr>
<tr>
<td>A.1</td>
<td>Simple Joints and Linkage of a Robotic System</td>
</tr>
<tr>
<td>C.1</td>
<td>Flexible Beam with Shearing Mass</td>
</tr>
<tr>
<td>C.2</td>
<td>Flexible Beam with Shearing Mass and Bending Moment</td>
</tr>
<tr>
<td>C.3</td>
<td>Flexible Beam with Joint End</td>
</tr>
<tr>
<td>C.4</td>
<td>Flexible Beam with Constrained End</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>Table 3.1</td>
<td>Denavit–Hartenberg Parameters of a Two Links Robot</td>
</tr>
<tr>
<td>Table 3.2</td>
<td>System Specification of the First Robot</td>
</tr>
<tr>
<td>Table 3.3</td>
<td>System Specification of the Second Robot</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Denavit–Hartenberg Parameters of a Three Links Robot</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>System Specification</td>
</tr>
<tr>
<td>Table 5.1</td>
<td>Denavit–Hartenberg Parameters of the UNLV–ARO Robot</td>
</tr>
<tr>
<td>Table 5.2</td>
<td>System Specification</td>
</tr>
</tbody>
</table>
NOMENCLATURE

\( \mathbf{i}_i \) \hspace{1cm} Unit vector of link i along the \( x \)-axis.

\( \mathbf{j}_i \) \hspace{1cm} Unit vector of link i along the \( y \)-axis.

\( \mathbf{k}_i \) \hspace{1cm} Unit vector of link i along the \( z \)-axis.

\( \mathbf{u}_i \) \hspace{1cm} Displacement of link i along the \( x_i \)-axis.

\( \mathbf{v}_i \) \hspace{1cm} Displacement of link i along the \( y_i \)-axis.

\( \mathbf{w}_i \) \hspace{1cm} Displacement of link i along the \( z_i \)-axis.

\( \mathbf{u}_i' \) \hspace{1cm} Slope of the displacement of link i along the \( x_i \)-axis.

\( \mathbf{v}_i' \) \hspace{1cm} Slope of the displacement of link i along the \( y_i \)-axis.

\( \mathbf{w}_i' \) \hspace{1cm} Slope of the displacement of link i along the \( z_i \)-axis.

\( \dot{\mathbf{u}}_i \) \hspace{1cm} Velocity of link i along the \( x_i \)-axis.

\( \dot{\mathbf{v}}_i \) \hspace{1cm} Velocity of link i along the \( y_i \)-axis.

\( \dot{\mathbf{w}}_i \) \hspace{1cm} Velocity of link i along the \( z_i \)-axis.

\( \gamma_{\mathbf{v}_i} \) \hspace{1cm} In-plane (\( y_i \)-direction) admissible function of link i.

\( \gamma_{\mathbf{w}_i} \) \hspace{1cm} Out-of-plane (\( z_i \)-direction) admissible function of link i.

\( q_{\mathbf{v}_i} \) \hspace{1cm} In-plane (\( y_i \)-direction) time-dependent generalized coordinate of link i.

\( q_{\mathbf{w}_i} \) \hspace{1cm} Out-of-plane (\( z_i \)-direction) time-dependent generalized coordinate of link i.

\( d_i \) \hspace{1cm} Link distance between link \( i-1 \) and link i.

\( a_i \) \hspace{1cm} Link length of link i.
a_{ir} \quad \text{Link length of the rigid portion of link } i.

a_{ie} \quad \text{Link length of the flexible portion of link } i.

\theta_i(t) \quad \text{Joint angle of link } i.

\rho_i \quad \text{Density per unit length of link } i.

\rho_{ir} \quad \text{Density per unit length of the rigid portion of link } i.

\rho_{ie} \quad \text{Density per unit length of the flexible portion of link } i.

J_i \quad \text{Mass moment of inertia of link } i.

T_{i\rightarrow j} \quad \text{Transformation matrix of frame } i \text{ to frame } j.

\mathbf{V}_i \quad \text{Displacement vector of link } i.

\mathbf{V}_{ir} \quad \text{Displacement vector of the rigid portion of link } i.

\mathbf{V}_{ie} \quad \text{Displacement vector of the flexible portion of link } i.

\mathbf{V}_{i\rightarrow j} \quad \text{Displacement vector of link } i \text{ in frame } j.

\mathbf{V}_{ir\rightarrow j} \quad \text{Displacement vector of the rigid portion of link } i \text{ in frame } j.

\mathbf{V}_{ie\rightarrow j} \quad \text{Displacement vector of the flexible portion of link } i \text{ in frame } j.

\Omega_i \quad \text{Angular velocity vector of link } i.

\Omega_{i\rightarrow j} \quad \text{Angular velocity vector of link } i \text{ in frame } j.

\dot{\mathbf{V}}_i \quad \text{Velocity vector of link } i.

\dot{\mathbf{V}}_{ir} \quad \text{Velocity vector of the rigid portion of link } i.

\dot{\mathbf{V}}_{ie} \quad \text{Velocity vector of the flexible portion of link } i.

\dot{\mathbf{V}}_{i\rightarrow j} \quad \text{Velocity vector of link } i \text{ in frame } j.

\dot{\mathbf{V}}_{ir\rightarrow j} \quad \text{Velocity vector of the rigid portion of link } i \text{ in frame } j.

\dot{\mathbf{V}}_{ie\rightarrow j} \quad \text{Velocity vector of the flexible portion of link } i \text{ in frame } j.
E_i \quad \text{Modular of elasticity of link } i.
I_{v_i} \quad \text{Area moment of inertia of link } i \text{ about the } y-\text{axis.}
I_{w_i} \quad \text{Area moment of inertia of link } i \text{ about the } z-\text{axis.}
Im_i \quad \text{Lumped mass of link } i.
sm_i \quad \text{Shearing mass of link } i.
Rot \quad \text{Rotation matrix.}
Trans \quad \text{Translation matrix.}
KE_i \quad \text{Kinetic energy of link } i.
PE_i \quad \text{Potential energy of link } i.
L \quad \text{Lagrangian operator.}
CHAPTER ONE

INTRODUCTION

The words *robot* and *robotics* originate from the Czech and Russian word *robota* (Shahinpoor, 1987). In the 1920s, a Czech playwright name Karel Čapek wrote a play entitled *Rossum's Universal Robots*. In this play, small, artificial creatures called *robotnic* are characterized as creations which strictly obey their master's orders. The word *robota* means "drudgery" and "hard work" in Czech and Russian. From then on, the words *robot* and *robotics* were popularized by media such as the motion picture industry and the scientific novels of Isaac Asimov.

The kinematics and dynamics of robotics have been studied extensively in the past decades. From these studies, numerous works have been directed towards the development and the formulation of the dynamic equations of motion for robot arms, so that solutions may be obtained to control and manipulate these robot arms to perform useful work. Traditionally, in order to obtain accurate end-effector trajectory, it is essential that the arms of the robot be free from structural deformations. Also, the inertia of the robot arms and the structural deformations together will induce vibrations to the system, which further increases the inaccuracy. To eliminate such complications, today’s robot arms are massive in order to increase the stiffness. However, massive, rigid robot arms also created other problems as well. From the maneuverability point of view, big actuators are required to initiate, maintain, and terminate the motion of the heavy robot arms and the payload; if fast response is necessary, much bigger actuators are required to
satisfy such performance. With stiffened robot arms, the cost of material increases; with bigger actuators, the power consumption increases; therefore, the overall cost of the system rises accordingly.

Since rigid robot arms exhibit such undesirable characteristics, recently, more researches are focusing on the development of flexible robot linkages. That is, structural deformations are being considered and are recognized as part of the system. Flexible robot arms exhibit characteristics that are not favorable to rigid robot arms, but it is the undesirable characteristics of the rigid robot arms that make the flexible robot arms attractive. For example, flexible arms require less structural material, which reduces the system's weight and the material cost. With less weight, sizable actuators are sufficient to satisfy both payload requirement and high speed maneuverability. Relatively smaller actuators reduce power consumption, thus, minimize the overall cost. Also, smaller components improve the robot portability.

In 1987, the United States Army Research Office (ARO) granted a research fund to the University of Nevada, Las Vegas. The research aim was to build a three-link hydraulically actuated robot with a 2.5 meters reach and a payload of 25 kilograms. This lead to a multi-discipline research effort with the objective of modeling and controlling a light-weight flexible robot. Numerous M.S. theses had been produced under this funding. Structural analysis of the flexible robot was presented by Bannoura (1988). Marceau (1989) studied the dynamic response of a flexible three-link robot using strain gages, Lagrange polynomials, Fourier series, and the finite element analysis. A three dimensional computerized model of a flexible robot arm was introduced by Krueger (1989). Das (1989) worked on the de-coupling, adaptive control and stabilization of a two-link flexible robot arm. Hu (1990) concentrated on the optimal joint trajectory of a flexible robot arm.
As discussed previously, flexible robot arms are not without drawbacks; which brings back the original problem: the trajectory accuracy and controllability. An accurate dynamics model is needed to predict and suppress the vibrations of the robot arms. Many works were produced on the dynamic modeling and manipulator control for such a system. Their approaches were mostly based on the basic dynamics principles such as the Newtonian mechanics, the Lagrangian mechanics, and the Hamiltonian principles. From these principles, a set of differential equations were formulated. Thus far, numerous works focused on formulating such equations of motion for robot with one flexible link (Hastings and Book, 1986, and Wang and Wei, 1987); some works also put their attention toward two flexible links robot (Usoro et al., 1986, and Nicosia et al., 1986). Even for the case of a one flexible link robot, it may not be possible to solve the equations of motion exactly, therefore, other works had been concentrated on numerical solution method. For example, Book (1979) described flexible mechanical system using lumping approximation and assuming that the first mode of vibration was predominated. Truckenbrodt (1979,1982) used the Ritz—Kantorovich method to analyze the deformation of a series of flexible links. Judd and Falkenburg (1985) developed a scheme to model the kinematic and dynamic motion of flexible linkages. Usoro et al. (1986) used the Lagrangian approach with finite element method to mathematically model lightweight flexible manipulators. Nicosia et al. (1986,1989) also used the Ritz—Kantorovitch method based on a function series expansion with the help of MACSYMA. In their 1989 paper, the Hamiltonian approach and the assumed modes method were used to obtain an approximate finite dimension model of the robot dynamic. Wang and Wei (1987) used Galerkin's approximation method to analyze the vibrations of a one-link flexible robot with a prismatic joint. Low (1989) used separation of variables method and the Galerkin's approach for the boundary—value problem with time—dependent boundary conditions.
The objective of this research is to study the vibration behavior of the UNLV—ARO hydraulically activated light-weight robot. The system is a robot with five links (figure 1.1). The first link (the trunk) is rigid, and is serially connected to the second link. This second link is flexible and is also serially connected to the third link. The third link is composed of two sections. The first section (section 3a) is assumed to be rigid while the second section (section 3b) is flexible. The fourth link is the hydraulic actuator for the second link. This hydraulic actuator is assumed to be rigid. The fifth link is the hydraulic actuator for the third link. The shell of this hydraulic actuator is considered rigid while the piston rod of this actuator is flexible. The modeling of this robot will include the in—plane and the out—of—plane vibrations. Lagrangian mechanics along with the assumed modes method are used for this research. MACSYMA and IMSL software packages are used as the tools for solving this problem. MACSYMA is a symbolic software package which helps to derive the equations of motion of the robot. The equations obtained are a set of non—linear differential equations. Since MACSYMA does not have the means for solving such equations, the IMSL package is used. The equations derived from MACSYMA are incorporated into FORTRAN programs using IMSL subroutines to solve the equations of motion.

The scope of this research is to provide a basis for the modeling of a light—weight robot. Physical characteristics such as the damping and the torsional effects of this multi—link light—weight robot are not included. Therefore, this work is not considered as a complete modeling of a light—weight robot. The damping of this light—weight robot may be included only if the damping coefficients of the robot are identified experimentally. Torsional vibrations can be readily observed during the motion of the actual model in the laboratory. The existence and the contribution of such vibrations are recognized. However, for simplicity, these two characteristics are not considered in this research.
Figure 1.1 UNLV—ARO Hydraulically Activated Robot
CHAPTER TWO

OVERVIEW

This chapter contains information that are necessary to model the dynamic motion of flexible robots. Discussion of the fundamental assumptions, the generalized coordinates of the robot, the Lagrangian mechanics, the symbolic derivation of the equations of motion, and the inverse dynamics problem are discussed in the following sections.

2.1 Fundamental Assumptions

The modeling of the flexible robot is based upon the following assumptions:

1) Gravity field is present throughout the system.
2) All flexible links are assumed to follow the Euler—Bernoulli's beam. That is, the flexible links are allowed to deflect under load but will not elongate. This also implies that small deflection theory is applied.
3) Torsional stiffness of the links is infinite, therefore, torsion on the links are neglected. For a multi-link robotics system, torsional vibrations play an importance role, and this assumption may be unrealistic; but for simplicity, this is a necessary step.
4) There are two sources of damping inherent in the system.
a) Structural damping in the links and in the hydraulic cylinders of the robot. These damping are usually neglected for most robotics applications.

b) Joint damping. This is the energy disputed at the joint clearance. Although joint damping have significance contribution towards the damping of the system, it is difficult to model without experimental verification. Therefore, this damping is also neglected in this work.

5) Any form of friction that may exist within the robot system is neglected and energy is conserved.

More assumptions will be made where appropriate.

2.2 Flexible Robot's Generalized Coordinates

In robotics, the Denavit–Hartenberg representation (Appendix A) is usually used to describe the position and the orientation of a robot with rigid links. From the Denavit–Hartenberg representation, the generalized coordinates describing the motion of link i preceded by a revolute joint is $\theta_i(t)$. The generalized coordinates describing the motion of link i preceded by a prismatic joint is $d_i$. For the robots studied here, the links are connected with revolute joints. Hence, for a rigid robot with n links, the generalized coordinates are:

$$\{\theta_1(t), \theta_2(t), ..., \theta_n(t)\}^T$$ (2.1)
However, when the links of the robot exhibit flexible behavior, additional coordinates are needed to describe the robot's motion. These coordinates are $v(u,t)$ and $w(u,t)$ for each flexible link (figure 2.1a, 2.1b). $v(u,t)$ represents the deflection of the link in the $y-$direction and $w(u,t)$ represents the deflection in the $z-$direction.

The generalized coordinates of the robot become:

$$\{\theta_1(t), \theta_2(t), ..., \theta_n(t), v_i(u_i,t), w_i(u_i,t), ..., v_j(u_j,t), w_j(u_j,t)\}^T$$  (2.2)

where $i$ through $j$ subscripts represent the flexible links of the robot.

$u_i$ is the displacement of link $i$ along the $x_i-$axis.

$u_j$ is the displacement of link $i$ along the $x_j-$axis.

It can be shown that using these additional coordinates will yield a set of second order, non-linear partial differential equations (D'Souza et al, 1984). Therefore, the assumed modes method (Appendix B) is used to substitute into these flexible motion coordinates. The resulting equations of motion become a set of second order non-linear ordinary differential equations. From the assumed modes method, $v_k(u_k,t)$ and $w_k(u_k,t)$ of link $k$ may be expressed as:

$$v_k(u_k,t) = \sum_{m=1}^{\infty} \gamma_{v_{km}}(u_k) q_{v_{km}}(t)$$  (2.3)

$$w_k(u_k,t) = \sum_{m=1}^{\infty} \gamma_{w_{km}}(u_k) q_{w_{km}}(t)$$  (2.4)

where $\gamma_{v_{km}}(u_k)$ is the admissible function in the $y_k-$direction.
Figure 2.1a  Projection of a Flexible Link in X—Y Plane (In—plane Deflection)

Figure 2.1b  Projection of a Flexible Link in X—Z Plane (Out—of—plane Deflection)
\( \gamma_{w_{k_{m}}} (u_{k}) \) is the admissible function in the \( z_{k} \)-direction.

\( q_{v_{k_{m}}} (t) \) is the time-dependent generalized coordinate in the \( y_{k} \)-direction.

\( q_{w_{k_{m}}} (t) \) is the time-dependent generalized coordinate in the \( z_{k} \)-direction.

\( y_{k} \) and \( z_{k} \) are local coordinates for link \( k \).

Book (1979) and Hu (1990) found that the first mode of vibration of a robot arm is the only mode shape that is of any significance. This is because the first mode is much easier to excite than any other mode shapes. Also, including two or more mode shapes in the modeling will dramatically increase the order of complexity of the system, and analyzing the system becomes more difficult. Furthermore, the magnitude of vibrations of the higher order mode shapes are much smaller relative to that of the first mode shape. Therefore, only the first mode of vibration is being considered in this research, and \( v_{k}(u_{k},t) \) and \( w_{k}(u_{k},t) \) of link \( k \) are rewritten as:

\[
\begin{align*}
  v_{k}(u_{k},t) &= \gamma_{v_{k_{1}}} (u_{k}) q_{v_{k_{1}}} (t) \quad (2.3a) \\
  w_{k}(u_{k},t) &= \gamma_{w_{k_{1}}} (u_{k}) q_{w_{k_{1}}} (t) \quad (2.4a)
\end{align*}
\]

or simply,

\[
\begin{align*}
  v_{k}(u_{k},t) &= \gamma_{v_{k}} (u_{k}) q_{v_{k}} (t) \quad (2.3b) \\
  w_{k}(u_{k},t) &= \gamma_{w_{k}} (u_{k}) q_{w_{k}} (t) \quad (2.4b)
\end{align*}
\]
The admissible functions are determined using cantilever beams as model. A discussion of the admissible functions is included in Appendix C. Finally, the system's generalized coordinates of (2.2) become:

\[ \{P_1, ..., P_I\}^T = \{\theta_1(t), ..., \theta_n(t), q_{w_1}(t), q_{w_2}(t), ..., q_{v_1}(t), q_{v_2}(t)\}^T \] (2.5)

where \( P_i \) is the \( i \)th time-dependent generalized coordinate of the system.

\( I \) is the total number of time-dependent generalized coordinates.

According to Trabia and Yim (1989), it is advantageous to choose the intermediate link's frame of a \( n \)-link robot as the reference frame. That is, for a three-link robot, frame 2 should be chosen as the reference frame, and all other frames should be transformed into frame 2. This approach will help to reduce the size of the displacement vectors and the velocity vectors. This is important because, as revealed later, the size of the equations of motion become enormous if this step is not taken. Therefore, frame 2 will be chosen as the reference frame throughout this text.

2.3 Lagrangian Mechanics

Lagrangian mechanics is the basis of this research and is used to derive the equations of motion for the flexible robot. To analyze a mechanical system using Lagrangian mechanics, the first step is to set up the kinematics relationships of that system, then use these kinematics relationships to obtain the dynamics of the system. In other words, it is necessary to obtain the displacement and velocity equations of the system first; then use these equations to formulate the system's
kinetic energy and potential energy equations. By substituting these energy equations into the Lagrangian equation, the equations of motion of the system may be obtained accordingly.

For a rigid link \( i \) preceded with a revolute joint, the velocity vector is represented as:

\[
\dot{\bar{V}}_i = \dot{V}_{i-1} + \left[ \bar{\Omega}_i \times \bar{V}_i \right]
\]

(2.6)

where

\( \bar{V}_i \) is the displacement vector of link \( i \).

\( \dot{\bar{V}}_i \) is the velocity vector of link \( i \).

\( \dot{V}_{i-1} \) is the velocity at the origin of link \( i \).

\( \bar{\Omega}_i \) is the angular velocity of link \( i \).

For a flexible link \( i \) preceded with a revolute joint, the velocity vector is represented as:

\[
\dot{\bar{V}}_i = \dot{V}_{i-1} + \left[ \bar{\Omega}_i \times \bar{V}_i \right] + \dot{V}_i(u_i,t) \left[ a_{1i} \ddot{i}_i + b_{1i} \ddot{j}_i + c_{1i} \ddot{k}_i \right] + \dot{w}_i(u_i,t) \left[ a_{2i} \dddot{i}_i + b_{2i} \dddot{j}_i + c_{2i} \dddot{k}_i \right]
\]

(2.7)

where

\( \dot{V}_i(u_i,t) \) is the velocity of the deflection in the \( y_i \)-direction.

\( \dot{w}_i(u_i,t) \) is the velocity of the deflection in the \( z_i \)-direction.

\( a_{1i}, b_{1i}, \) and \( c_{1i} \) are the projection of \( \dot{V}_i(u_i,t) \) onto \( i_i, j_i, \) and \( k_i \).

\( a_{2i}, b_{2i}, \) and \( c_{2i} \) are the projection of \( \dot{w}_i(u_i,t) \) onto \( i_i, j_i, \) and \( k_i \).
The kinetic energy of link i is represented as:

$$ KE_i = KE_{1i} + KE_{2i} + KE_{3i} $$

where $KE_i$ is the total kinetic energy of link i.
$KE_{1i}$ is the kinetic energy of the rigid portion of link i.
$KE_{2i}$ is the kinetic energy of the flexible portion of link i.
$KE_{3i}$ is the kinetic energy of the lumped mass at the end of link i.

Similarly, the potential energy of link i is represented as:

$$ PE_i = PE_{1i} + PE_{2i} + PE_{3i} + PE_{4i} $$

where $PE_i$ is the total potential energy of link i.
$PE_{1i}$ is the strain energy of the flexible portion of link i.
$PE_{2i}$ is the potential energy of the rigid portion of link i.
$PE_{3i}$ is the potential energy of the flexible portion of link i.
$PE_{4i}$ is the potential energy of the lumped mass at the end of link i.

Finally, let:

$$ L_i = KE_i - PE_i $$

$$ L = \sum_{i=1}^{n} L_i $$

$$ \frac{d}{dt} \left( \frac{\partial L}{\partial P_j} \right) - \frac{\partial L}{\partial P_j} = Q_j(t) $$
where \( L_i \) is the Lagrange operator of link \( i \).

\( L \) is the Lagrange operator of the system.

\( n \) is the total number of links of the robotic system.

\( P_j \) is the \( j \)th time—dependent generalized coordinate of the system.

\( Q_j(t) \) is the \( j \)th time—dependent non—conservative generalized forces.

Substituting the generalized coordinates of (2.5) into the right hand side of (2.12), the time—dependent non—conservative generalized force of (2.12) become:

\[
Q_m(t) = \text{Torque}_m
\]  
(2.13)

or

\[
Q_m(t) = \text{Force}_m
\]  
(2.14)

where

\[
m = 1, ..., n
\]  
(2.15)

For the remaining equations of (2.12):

\[
Q_m(t) = 0
\]  
(2.16)

where

\[
m = n + 1, ..., l
\]  
(2.17)

Hence, substituting the generalized coordinates into the Lagrangian equation, (2.12) yields:

\[
\begin{bmatrix}
\ddot{\theta}_1 \\
\vdots \\
\ddot{\theta}_n \\
\dddot{\theta}_v_i \\
\vdots \\
\dddot{\theta}_w_j
\end{bmatrix}
+ \begin{bmatrix}
F_1 \\
\vdots \\
F_n \\
F_v \\
\vdots \\
F_w
\end{bmatrix}
= \begin{bmatrix}
\text{Torque}_1 \\
\vdots \\
\text{Torque}_n \\
0 \\
\vdots \\
0
\end{bmatrix}
\]  
(2.18)
where \( C \) is a \( l \times l \) square matrix containing the coefficients of the second time derivative of the generalized coordinates.

\( F_k \) contains the remaining terms of the left hand side of the \( k \)th equation. This equation is a function of \( P \)'s and \( \dot{P} \)'s.

(2.18) is a second order differential equation. This means that two initial conditions per equation are needed to solve the system.

2.4 Symbolic Derivation of the Equations of Motion

For a simple flexible robot arm, the equations of motion may be easily derived manually. However, for more complex flexible robots, the equations of motion are usually very lengthy and difficult to derive. Therefore, the equations of motion of the UNLV—ARO light—weight robot are derived symbolically using MACSYMA. MACSYMA is a symbolic computing software. It was originally developed by the Mathlab Group of the Massachusetts Institution of Technology Laboratory for Computer Science. The word MACSYMA stands for Project MAC's SYmbolic MANipulation System (Symbolics, 1985). MACSYMA's main objective is to perform symbolic and numeric mathematical manipulations. That is, given a mathematical equation, the user may perform any kind of mathematical operation to obtain a desired result. It allows users to differentiate or integrate equations, the result of these operations will be in symbolic form, similar to an engineer or a mathematician carrying out such operations on paper. MACSYMA also allows the users to perform matrix operations, taking limits, factoring polynomials, performing algebraic manipulation and Taylor series expansion, plotting curves, translating symbolic equations into FORTRAN codes, etc... just to name a few.
2.5 Inverse Dynamics

There are two kinds of dynamic problems in robotics. They are the direct dynamic problem and the inverse dynamic problem. In direct dynamic problem, known forces or torques are input into the system's equations in order to solve for the motion of the system. In inverse dynamic problem, it is required to input the joint motion parameters (time history of the angular displacement, velocity, and acceleration of the joints) into the system's equations in order to obtain a solution.

In this research, the inverse dynamic solution of flexible robots is considered. For rigid robots, the time history of the joint motion is input into the equations of motion. The results obtained are the torques needed at the joints to sustain the motion of the rigid robot. However, for flexible robots, substituting the time history of the joint motion into the torque equations (the top \( n \) equations of (2.18)) is not sufficient to solve the problem. This is due to the fact that these \( n \) equations contain terms that are also functions of the motion of the flexible links (\( q_v \)'s and \( q_w \)'s). Hence, it is essential to solve the lower \( l - n \) equations of (2.18) before calculating the joint torques. Solving these \( l - n \) equations simultaneously yield the solution for the time-dependent generalized coordinates \( q_v \)'s and \( q_w \)'s. The time history of the joint motion along with \( q_v \)'s and \( q_w \)'s are then input into the torque equations, which yields the torques needed at the joints to sustain the motion of the flexible robot.

From the discussion above, the lower \( l - n \) equations of (2.18) becomes:

\[
\begin{bmatrix}
\ddot{q}_{v_1} \\
\vdots \\
\ddot{q}_{v_l} \\
\ddot{q}_{w_1} \\
\vdots \\
\ddot{q}_{w_j}
\end{bmatrix} +
\begin{bmatrix}
F_{n+1} \\
\vdots \\
F_l
\end{bmatrix} =
\begin{bmatrix}
0 \\
\vdots \\
0
\end{bmatrix}
\] 

(2.19)
where \( D \) is a \( l \times n \) square matrix containing the coefficients of the second time derivative of the generalized coordinates \( q' \)'s. It is the lower right corner of \( [C] \) of (2.18).

\( F_k \) contains the remaining terms of the left hand side of the \( k \)th equation. This equation is a function of \( P \)'s and \( \dot{P} \)'s.

It is assumed that the studied motion always starts from a stationary position. This implies that the initial joint angles are known and the joint angular velocities are equal to zero. This assumption also implies that the initial link deflections may be calculated using standard static deflection equations. It also implies that the time rate of change of the link deflections, \( \dot{\nu} \) and \( \dot{\nu} \), are equal to zero.

2.5.1 Time History of the Joint Motion

From kinematics, the displacement and the velocity profiles of the system are greatly depended on the choice of the acceleration profile. Profiles such as the bang—bang acceleration and the cycloidal motion may yield the same initial and final displacement of the end—effector, but the trajectory of the end—effector and the torques at the joints may have significant differences. In this research, two acceleration profiles are used. The two profiles are the bang—bang acceleration profile and the polynomial acceleration profile.

Figure 2.2, 2.3, 2.4 depict the angular accelerations, the angular velocities, and the angular displacements of the bang—bang profile and the polynomial profile. The equations of the profiles are listed below.
Figure 2.2 Sample Acceleration Profiles
Figure 2.3 Sample Velocity Profiles
Figure 2.4 Sample Displacement Profiles
For the bang–bang acceleration profile. For \(0 \leq t < \frac{1}{2}t_m\):

\[
\ddot{\theta} = (\theta_f - \theta_i) \left[\frac{t_m}{2}\right]^{-2}
\]

(2.20)

\[
\dot{\theta} = \ddot{\theta} t
\]

(2.21)

\[
\theta(t) = \theta_i + \left[\frac{1}{2} \ddot{\theta} t \right]^2
\]

(2.22)

where

- \(t\) is the independent variable.
- \(t_m\) is the motion time of the link.
- \(\theta_i\) is the initial angular displacement of the joint.
- \(\theta_f\) is the final angular displacement of the joint.
- \(\theta(t)\) is the angular displacement of the joint.
- \(\dot{\theta}\) is the angular velocity of the joint.
- \(\ddot{\theta}\) is the angular acceleration of the joint.

For \(\frac{1}{2}t_m \leq t < t_m\):

\[
\ddot{\theta} = - (\theta_f - \theta_i) \left[\frac{t_m}{2}\right]^{-2}
\]

(2.23)

\[
\dot{\theta} = \ddot{\theta} (t_m - t)
\]

(2.24)

\[
\theta(t) = \theta_f - \left[\frac{1}{2} \ddot{\theta} (t_m - t) \right]^2
\]

(2.25)
For $t \geq t_m$:

\[ \ddot{\theta} = 0 \quad (2.26) \]

\[ \dot{\theta} = 0 \quad (2.27) \]

\[ \theta(t) = \theta_f \quad (2.28) \]

The polynomial acceleration profile included here is based on the research of Trabia and Yim (1990). The profile used here is slightly modified from that of Trabia and Yim. In their paper, the switching point of the acceleration profile is numerically approximated for best performance. However, in this research, the switching point of the acceleration profile is at the half way point of the total motion time.

The displacement equations of a polynomial profile has the following forms.

\[ \theta(t) = \sum_{i=0}^{n} d_i t^i \quad 0 \leq t < -\frac{1}{2}t_m \quad (2.29) \]

\[ \theta(t) = \sum_{i=0}^{n} e_i t^i \quad -\frac{1}{2}t_m < t < t_m \quad (2.30) \]

To solve for the coefficients of the profile, the following conditions are enforced.

For $t = 0$:

\[ \theta(0) = \theta_i \quad (2.31) \]
\[
\dot{\theta}(0) = 0 \quad (2.32)
\]
\[
\ddot{\theta}(0) = 0 \quad (2.33)
\]
\[
\dddot{\theta}(0) = 0 \quad (2.34)
\]

For \( t = \frac{t_m}{4} \):
\[
\dddot{\theta} \left[ \frac{t_m}{4} \right] = 0 \quad (2.35)
\]

For \( t = \frac{t_m}{2} \):
\[
\theta \left[ \frac{t_m}{2} \right] = \theta_i + \frac{\theta_f - \theta_i}{2} \quad (2.36)
\]
\[
\dddot{\theta} \left[ \frac{t_m}{2} \right] = 0 \quad (2.37)
\]
\[
\dddot{\theta} \left[ \frac{t_m}{2} \right] = 0 \quad (2.38)
\]
\[
\theta^V \left[ \frac{t_m}{2} \right] = 0 \quad (2.39)
\]

For \( t = t_m \):
\[
\theta(t_m) = \theta_f \quad (2.40)
\]

Solving the equations of the polynomial profile using the conditions listed above, the equations for the acceleration, the velocity and the displacement profile
for $0 \leq t < \frac{1}{2} t_m$ are:

$$
\ddot{\vartheta} = \sum_{i=4}^{8} (i) (i - 1) d_i t^{(i - 2)} \quad (2.41)
$$

$$
\dot{\vartheta} = \sum_{i=4}^{8} (i - 1) d_i t \quad (2.42)
$$

$$
\vartheta(t) = \vartheta_i + \sum_{i=4}^{8} d_i t^i \quad (2.43)
$$

For $\frac{1}{2} t_m \leq t < t_m$:

$$
\ddot{\vartheta} = \sum_{i=4}^{8} (i) (i - 1) e_i (t_m - t)^{(i - 2)} \quad (2.44)
$$

$$
\dot{\vartheta} = -\sum_{i=4}^{8} (i) e_i (t_m - t)^{(i - 1)} \quad (2.45)
$$

$$
\vartheta(t) = \vartheta_i + \sum_{i=4}^{8} e_i (t_m - t)^i \quad (2.46)
$$

For $t \geq t_m$:

$$
\ddot{\vartheta} = 0 \quad (2.47)
$$

$$
\dot{\vartheta} = 0 \quad (2.48)
$$

$$
\vartheta(t) = \vartheta_f \quad (2.49)
$$
Since the switching time of the profile is at the half way point of the motion time, this profile exhibits an inverted mirror image. Hence, the matrix for \( d_1 \) and \( e_1 \) are identical and only one \( 5 \times 5 \) matrix is needed. The solution for \( d_1 \) and \( e_1 \) are:

\[
\begin{bmatrix}
  d_4 \\
  d_5 \\
  d_6 \\
  d_7 \\
  d_8 \\
\end{bmatrix} =
\begin{bmatrix}
  \tau_1^4 & \tau_1^5 & \tau_1^6 & \tau_1^7 & \tau_1^8 \\
  12 \tau_1^2 & 20 \tau_1^3 & 30 \tau_1^4 & 42 \tau_1^5 & 56 \tau_1^6 \\
  24 \tau_2 & 60 \tau_2^2 & 120 \tau_2^3 & 210 \tau_2^4 & 336 \tau_2^5 \\
  24 \tau_1 & 60 \tau_1^2 & 120 \tau_1^3 & 210 \tau_1^4 & 336 \tau_1^5 \\
  24 & 120 \tau_1 & 360 \tau_1^2 & 840 \tau_1^3 & 1680 \tau_1^4 \\
\end{bmatrix}^{-1}
\begin{bmatrix}
  \Delta \theta \\
  0 \\
  0 \\
  0 \\
  0 \\
\end{bmatrix}
\]

(2.50)

\[
\begin{bmatrix}
  e_4 \\
  e_5 \\
  e_6 \\
  e_7 \\
  e_8 \\
\end{bmatrix} =
\begin{bmatrix}
  \tau_1^4 & \tau_1^5 & \tau_1^6 & \tau_1^7 & \tau_1^8 \\
  12 \tau_1^2 & 20 \tau_1^3 & 30 \tau_1^4 & 42 \tau_1^5 & 56 \tau_1^6 \\
  24 \tau_2 & 60 \tau_2^2 & 120 \tau_2^3 & 210 \tau_2^4 & 336 \tau_2^5 \\
  24 \tau_1 & 60 \tau_1^2 & 120 \tau_1^3 & 210 \tau_1^4 & 336 \tau_1^5 \\
  24 & 120 \tau_1 & 360 \tau_1^2 & 840 \tau_1^3 & 1680 \tau_1^4 \\
\end{bmatrix}^{-1}
\begin{bmatrix}
  -\Delta \theta \\
  0 \\
  0 \\
  0 \\
  0 \\
\end{bmatrix}
\]

(2.51)

where

\[
\tau_1 = \frac{t_m}{2}
\]

(2.52)

\[
\tau_2 = \frac{t_m}{4}
\]

(2.53)

\[
\Delta \theta = \frac{\theta_f - \theta_i}{2}
\]

(2.54)

The values of the coefficients \( d_4, \ldots, d_8 \), \( e_4, \ldots \), and \( e_8 \) depend on the initial and final angular displacement of the joints. The values of these coefficients may easily be obtained by solving the matrices.
2.5.2 Use of IMSL for Solving the Equations of Motion of Flexible Robots

As mentioned previously, the equations of motion are second order non-linear ordinary differential equations. Therefore, to solve for the time-dependent generalized coordinates $q_v$'s and $q_w$'s in (2.19), the IMSL subroutine *ivpag* is used. Subroutine *ivpag* is used to solve first order differential equations. For example:

$$\ddot{y} = f(x, \bar{y}) \tag{2.55}$$

Hence, second order non-linear ordinary differential equations have to be reduced to first order by using the concept of state variables from control theory. Consider a simple equation of motion:

$$M \ddot{x} + K x(t) = 0 \tag{2.56}$$

let

$$y = \dot{x} \tag{2.57}$$

Substitute (2.57) into (2.56) yield:

$$M \dot{y} + K x(t) = 0 \tag{2.58}$$

The system of (2.56) becomes:

$$\frac{d}{dt} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} y \\ -\frac{K}{M} x(t) \end{bmatrix} \tag{2.59}$$
To use subroutine *ivpag*, several important aspects need to be mentioned.

1) Subroutine *ivpag* provides the user with three solution methods. However, the *Dgear* method is used because it seemed most appropriate for the applications in this research.

2) The user is required to supply two subroutines for subroutine *ivpag*. The subroutines are *fcn* and *fenj*. Subroutine *fcn* contains the differential equations of the robotic system. Subroutine *fenj* contains the Jacobian of the robotic system. Subroutine *ivpag* calls these subroutines to evaluate the equations of the system. However, for this research, the Jacobian of the flexible robots is very complicated, and the solution method chosen does not require the Jacobian calculation. Hence, subroutine *fenj* is left blank.

3) Subroutine *ivpag* requires initial conditions. For the robotic systems studied here, the initial conditions are:
   a) The initial angular accelerations of the joints.
   b) The initial angular velocities of the joints.
   c) The initial angular displacements of the joints.
   d) The initial deflections of the flexible links.
   e) The initial slope of the deflections of the flexible links.

4) Subroutine *ivpag* performs its function in incremental time steps. At each time step, subroutine *ivpag* is called to approximate the best possible values for \(q_v\)'s and \(q_w\)'s.

5) The following is a list of variables used in subroutine *ivpag*.
   a) The variable \(x\) is used to represent the independent variables.
   b) The array variables \(y()\) and \(yprime()\) are used to represent the dependent variables and their time derivatives respectively.
2.6 Case Studies

In this research, three modeling case studies are considered. The first case study (chapter three) is the modeling of a two links robot with one flexible link. Although this case study is well presented in the literatures, it is included to provide the insight needed for more complex cases. The second case study (chapter four) is the modeling of a robot with three links actuated by rotary actuators. The first link of the robot is rigid while the two outer-most links are flexible. In this case study, the motion of the two flexible links are limited to the in-plane motion. Finally, for the third and final case study (chapter five), the modeling of the UNLV-ARO flexible robot is developed based on the experience gained from the previous case studies. The in-plane and the out-of-plane motions of the system are considered for this case.
CHAPTER THREE

MODELING OF A TWO-LINK ROBOT WITH 
ONE FLEXIBLE OUTER-MOST LINK

For this research, it is beneficial to begin the study of a flexible robot with one flexible link. Figure 3.1 shows a sketch of such a robot. This chapter will provide a simple guideline of modeling the vibrations of a robot with flexible links. It also acts as an example so that one may understand the fundamentals of this research. Furthermore, this chapter will help the reader to comprehend more complicated systems in the chapters that follow.

From figure 3.1, link 1 is the trunk of the robot and link 2 is the flexible outer-most link. For simplicity, joint 1 is fixed and joint 2 is allowed to rotate. The motion of link 2 is actuated by a rotary actuator located along the joint axis ($z_2$-axis) of link 2. Since this is a simple case study, the equations of motion are derived manually and by MACSYMA. The results from both methods are presented in this chapter.
Figure 3.1 Two-link Robot With One Flexible Outer-most Link
3.1 Kinematic Analysis of a Two-link Robot With One Flexible Outer-most Link

From figure 3.1, the Denavit-Hartenberg parameters (Appendix A) of this robot are:

<table>
<thead>
<tr>
<th>Link</th>
<th>$\alpha_i$</th>
<th>$a_i$</th>
<th>$\theta_i$</th>
<th>$d_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>1</td>
<td>$\pi/2$</td>
<td>0</td>
<td>0</td>
<td>$d_1$</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>$a_2$</td>
<td>$\theta_2(t)$</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.1 Denavit-Hartenberg Parameters of a Two Links Robot

3.1.1 Kinematic Analysis of link 1

3.1.1.1 Displacement Vector of Link 1

Since link 1 is assumed to be rigid, therefore, there is no deformation of any kind in any direction. The displacement vector of link 1 is:

$$
\mathbf{V}_1 = 0 \mathbf{i} + 0 \mathbf{j} - w_1 \mathbf{k} = (0 \quad 0 \quad -w_1)^T
$$

(3.1)

where $w_1$ is the displacement of link 1 along the $z_1$-axis.

As discussed in section 2.2, it is necessary to express $\mathbf{V}_1$ in terms of frame 2. So by substituting the Denavit-Hartenberg parameters of table 3.1 to (A.10), the orientation portion of (A.10) becomes:
\[ T_{1 \rightarrow 2} = \begin{bmatrix} \cos(\theta_2(t)) & 0 & \sin(\theta_2(t)) \\ -\sin(\theta_2(t)) & 0 & \cos(\theta_2(t)) \\ 0 & -1 & 0 \end{bmatrix} \] (3.2)

where \( T_{1 \rightarrow 2} \) is the transformation matrix of frame 1 to frame 2.

Therefore, the displacement vector of link 1 becomes:

\[ \bar{V}_{1 \rightarrow 2} = T_{1 \rightarrow 2} \cdot \bar{V}_1 = \begin{bmatrix} -w_1 \sin(\theta_2(t)) \\ -w_1 \cos(\theta_2(t)) \\ 0 \end{bmatrix} \] (3.3)

where \( \bar{V}_{1 \rightarrow 2} \) is the displacement vector of link 1 in frame 2.

3.1.1.2 Velocity Vectors of Link 1

Since there is no rotation about the \( z_1 \)-axis, the angular velocity of link 1 is:

\[ \bar{\Omega}_1 = 0 \] (3.4)

\[ \Rightarrow \quad \bar{\Omega}_{1 \rightarrow 2} = 0 \] (3.5)

where \( \bar{\Omega}_{1 \rightarrow 2} \) is the angular velocity vector of link 1 in frame 2.

From (2.6), the velocity vector of link 1 becomes:
where \( V_g \) is the velocity at the origin of link 1.

Since the base frame is fixed,

\[
\dot{V}_0 = 0
\]  

\[
\Rightarrow \quad \dot{V}_{1+2} = 0
\]

3.1.2 Kinematic Analysis of Link 2

3.1.2.1 Displacement Vectors of Link 2

The general displacement vector that describe each point on link 2 is:

\[
\vec{V}_2 = \vec{u}_2 \hat{i}_2 + v_2(u_2,t) \hat{j}_2 + 0 \hat{k}_2
\]

where \( u_2 \) is the displacement of link 2 along the \( x_2 \)-axis.
\( v_2(u_2,t) \) is the deflection of link 2 in the \( y_2 \)-direction.

From the assumed modes method (Appendix B) and the discussions of the generalized coordinates in section 2.2, \( v_2 \) may be expressed as:

\[
v_2(u_2,t) = \gamma v_2(u_2) q v_2(t)
\]
where $\gamma_{v_2}(u_2)$ is the admissible function of link 2 in the $y_2$-direction.

$q_{v_2}(t)$ is the time-dependent generalized coordinate of link 2 in the $y_2$-direction.

Since link 2 is composed of a rigid portion and a flexible portion, it is necessary to formulate two vectors to describe the two portions. For $0 \leq u_2 \leq a_{2r}$:

$$V_{2r} = u_2 i_2 + 0 j_2 + 0 k_2 = (u_2 \ 0 \ 0) \quad (3.11)$$

For $a_{2r} < u_2 \leq a_2$:

$$V_{2e} = u_2 i_2 + v_2(u_2, t) j_2 + 0 k_2 = \begin{bmatrix} u_2 \\ v_2(u_2, t) \\ 0 \end{bmatrix} \quad (3.12)$$

where $V_{2r}$ is the displacement vector for the rigid portion of link 2.

$V_{2e}$ is the displacement vector for the flexible portion of link 2.

$a_{2r}$ is the length of the rigid portion of link 2.

$a_2$ is the total length of link 2.

Link 2 may be modeled as a cantilever beam with shearing mass at the end of the beam. Using this assumption, case 1 of Appendix C is used to formulate an expression for the admissible function $\gamma_{v_2}$. From case 1 of Appendix C, (C.6) becomes:
where \( \text{l}_m^2 \) is the lumped mass at the end of link 2.

Substituting \( (3.13) \) into \( (C.6) \) of Appendix C will yield the proper admissible function of link 2.

### 3.1.2.2 Velocity Vectors of Link 2

The angular velocity of link 2 is:

\[
\dot{\Omega}_2 = 0 \dot{i}_2 + 0 \dot{j}_2 + \dot{\theta}_2 k_2 = (0 \quad 0 \quad \dot{\theta}_2)^T
\]  

(3.14)

The cross products of \( (2.6) \) and \( (2.7) \) become:

\[
\begin{align*}
\dot{\Omega}_2 \times \dot{V}_{2r} &= (0 \quad u_2 \dot{\theta}_2 \quad 0)^T \\
\dot{\Omega}_2 \times \dot{V}_{2e} &= \begin{bmatrix} -v_2(u_2, t) \dot{\theta}_2 \\ u_2 \dot{\theta}_2 \\ 0 \end{bmatrix}
\end{align*}
\]  

(3.15)

(3.16)

Substituting the equations above into \( (2.6) \) and \( (2.7) \) yield:

\[
\dot{V}_{2r} = (0 \quad u_2 \dot{\theta}_2 \quad 0)^T
\]  

(3.17)
\[
V_{2e} = \begin{bmatrix}
-v_2(u_2, t) \\
u_2 \dot{\theta}_2 + v_2(u_2, t) \\
0
\end{bmatrix}
\]  
(3.18)

3.2 Dynamic Analysis of a Two–link Robot With One Flexible Outer–most Link

3.2.1 Kinetic and Potential Energy Equations of Link 1

Since link 1 is fixed, the kinetic energy of this link is zero.

\[
KE_1 = 0
\]  
(3.19)

The potential energy of link 1 becomes:

\[
PE_1 = - \rho_1 \int_{0}^{-d_1} \left[ \underline{G_{1 \rightarrow 2}} \cdot \underline{V_{1 \rightarrow 2}} \right] dw_1
\]  
(3.20)

where \( \rho_1 \) is the mass per unit length of link 1.

\( \underline{G_{1 \rightarrow 2}} \) is the vector for the gravitational field with respect to frame 2.

Generally, the gravitational field is represented as the following:

\[
\underline{G} = \begin{bmatrix} i_1 \\ j_1 + (-g) k_1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -g \end{bmatrix}^T
\]  
(3.21)
However, since frame 2 is chosen as the reference frame, $\mathbf{G}$ needs to be transformed into frame 2.

$$
G_{1 \rightarrow 2} = T_{1 \rightarrow 2} \cdot \mathbf{G} = \begin{bmatrix}
-g \sin(\theta_2(t)) \\
-g \cos(\theta_2(t)) \\
0
\end{bmatrix}
$$

(3.22)

Solving the potential energy equation yields:

$$
PE_1 = -\frac{1}{2} \rho_1 g \frac{d_1}{2}
$$

(3.23)

3.2.2 Kinetic and Potential Energy Equations of Link 2

Using (3.9) through (3.18), the kinetic energy equation (2.8) of link 2 becomes:

$$
KE_2 = \frac{1}{2} \rho_2 r \int_0^{a_2r} \left[ \frac{\dot{V}_{2r}}{V_{2r}} \cdot \frac{\dot{V}_{2r}}{V_{2r}} \right] du_2 + \frac{1}{2} \rho_{2e} \int_{a_2r}^{a_2} \left[ \frac{\dot{V}_{2e}}{V_{2e}} \cdot \frac{\dot{V}_{2e}}{V_{2e}} \right] du_2
$$

$$
+ \frac{1}{2} \text{sm}_2 \left[ \frac{\dot{V}_{2e}(a_2)}{V_{2e}(a_2)} \cdot \frac{\dot{V}_{2e}(a_2)}{V_{2e}(a_2)} \right]
$$

(3.24)

$$
\text{sm}_2 = \text{lm}_2
$$

(3.25)

where

- $\rho_{2r}$ is the mass per unit length of the rigid portion of link 2.
- $\rho_{2e}$ is the mass per unit length of the flexible portion of link 2.
- $\text{sm}_2$ is the total shearing mass at the end of link 2.
Similarly, the potential energy equation (2.9) of link 2 becomes:

\[
PE_2 = \frac{1}{2} E_2 I_{v_2} \int_{a_{2r}}^{a_2} \left[ v_2 \left( u_{2,t} \right) \right]^2 du_2 - \rho_{2r} \int_{0}^{a_{2r}} \left[ \frac{G_{1+2}}{r} \cdot \bar{V}_{2r} \right] du_2 \\
- \rho_{2e} \int_{a_{2r}}^{a_2} \left[ \frac{G_{1+2}}{r} \cdot \bar{V}_{2e} \right] du_2 - sm_2 \left[ \frac{G_{1+2}}{r} \cdot \bar{V}_{2e}(a_2) \right]
\]

(3.26)

where \( E_2 \) is the modulus of elasticity of link 2.

\( I_{v_2} \) is the area moment of inertia of link 2 in the \( y_2 \)-direction.

Solving (3.24), the kinetic energy equation of link 2 becomes:

\[
KE_2 = \frac{1}{2} \rho_{2r} \int_{0}^{a_{2r}} \left[ u_2 \dot{\theta}_2 \right]^2 du_2 \\
+ \frac{1}{2} \rho_{2e} \int_{a_{2r}}^{a_2} \left[ \gamma_{v_2}(u_2) q_{v_2}(t) \dot{\theta}_2 \right]^2 du_2 \\
+ \frac{1}{2} sm_2 \left[ \gamma_{v_2}(a_2) q_{v_2}(t) \dot{\theta}_2 \right]^2 du_2
\]

(3.27)
Solving (3.26), the potential energy equation of link 2 becomes:

\[
PE_2 = \frac{1}{2} E_2 I_{v2} \int_{a_2 r}^{a_2} \left[ \gamma_{v2}''(u_2) q_{v2}(t) \right]^2 \, du_2
\]

\[
- \rho_2 \int_0^{a_2r} \left[ -g u_2 \sin(\theta_2(t)) \right] \, du_2
\]

\[
- \rho e \int_{a_2}^{a_2r} \left\{ -g \left[ u_2 \sin(\theta_2(t)) + \gamma_{v2}(u_2) q_{v2}(t) \cos(\theta_2(t)) \right] \right\} \, du_2
\]

\[
- \frac{1}{2} \rho_2 \gamma_{x2} \left\{ -g \left[ a_2 \sin(\theta_2(t)) + \gamma_{v2}(a_2) q_{v2}(t) \cos(\theta_2(t)) \right] \right\}
\]

(3.28)

3.2.3 Lagrangian Equations

For this flexible robot, there are two time—dependent generalized coordinates, they are \( q_{v2}(t) \) and \( \theta_2(t) \). Therefore, substituting these coordinates into (2.12) yield:

\[
\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_{v2}} - \frac{\partial L}{\partial q_{v2}(t)} = \text{Torque}_2
\]

(3.29)

\[
\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}_2} - \frac{\partial L}{\partial \theta_2(t)} = 0
\]

(3.30)

\[
L = KE_1 + KE_2 - PE_1 - PE_2
\]

(3.31)

Substitute (3.19), (3.23), (3.27), and (3.28) into (3.31), (3.29) becomes:
\[ \rho_{2e} \int_{a_2}^{a_2} \left\{ \gamma_{v_2}^{2}(u_2) q_{v_2}(t) \ddot{\theta}_2 + 2 \gamma_{v_2}^{2}(u_2) q_{v_2}(t) \dot{\theta}_2 \dot{q}_{v_2} + u_2 \left[ \gamma_{v_2}^{2}(u_2) q_{v_2} + u_2 \ddot{\theta}_2 \right] \right\} \, du_2 \\
+ \text{sm}_2 \left\{ \gamma_{v_2}^{2}(a_2) q_{v_2}(t) \ddot{\theta}_2 + 2 \gamma_{v_2}^{2}(a_2) q_{v_2}(t) \dot{\theta}_2 \dot{q}_{v_2} + a_2 \left[ \gamma_{v_2}^{2}(a_2) q_{v_2} + a_2 \ddot{\theta}_2 \right] \right\} + \frac{1}{3} \rho_{2r} a_{2r}^3 \ddot{\theta}_2 \\
+ \rho_{2e} g \int_{a_2}^{a_2} \left[ u_2 \cos(\theta_2(t)) - \gamma_{v_2}^{2}(u_2) q_{v_2}(t) \sin(\theta_2(t)) \right] \, du_2 \\
+ \text{sm}_2 g \left[ a_2 \cos(\theta_2(t)) - \gamma_{v_2}^{2}(a_2) q_{v_2}(t) \sin(\theta_2(t)) \right] \\
+ \frac{1}{2} \rho_{2r} g a_{2r}^2 \cos(\theta_2(t)) = \text{Torque}_2 \\
(3.32) \]

Substitute (3.19), (3.23), (3.27), and (3.28) into (3.31), (3.30) becomes:

\[ \rho_{2e} \int_{a_2}^{a_2} \gamma_{v_2}^{2}(u_2) \left[ u_2 \ddot{\theta}_2 + \gamma_{v_2}^{2}(u_2) \ddot{q}_{v_2} \right] \, du_2 \\
+ \text{sm}_2 \gamma_{v_2}^{2}(a_2) \left[ a_2 \ddot{\theta}_2 + \gamma_{v_2}^{2}(a_2) \ddot{q}_{v_2} \right] \\
+ E_2 I_{v_2} q_{v_2}(t) \int_{a_2}^{a_2} \left[ \gamma_{v_2}^{2}(u_2) \right] \, du_2 \\
- \rho_{2e} q_{v_2}(t) \dot{\theta}_2 \int_{a_2}^{a_2} \gamma_{v_2}^{2}(u_2) \, du_2 + \rho_{2e} g \cos(\theta_2(t)) \int_{a_2}^{a_2} \gamma_{v_2}^{2}(u_2) \, du_2 \\
+ \text{sm}_2 g \cos(\theta_2(t)) \gamma_{v_2}^{2}(a_2) = 0 \\
(3.33) \]
3.3 Initial Conditions

In order to solve the equations of motion, it is necessary to determine the initial conditions of the robot. From section 2.5.1, the initial values for $\theta_2(0)$, $\dot{\theta}_2(0)$, and $\ddot{\theta}_2(0)$ are pre-determined. The initial values for $q_{v2}(0)$ may be obtained from the initial static deflection of link 2. Since,

$$v_2(a_2,0) = \left[ -\frac{\rho_{2e} g a_2}{8 E_2 I_{v_2}} \right] + \left[ -\frac{\sin_2 g a_2}{3 E_2 I_{v_2}} \right] \cos(\theta_2(0)) \quad (3.34)$$

$$=> \quad q_{v2}(0) = \frac{v_2(a_2,0)}{a_{v2}(a_2)} \quad (3.35)$$

Initially, the link 2 of the robot is stationary, therefore:

$$\dot{q}_{v2}(0) = 0 \quad (3.36)$$

3.4 MACSYMA Programs

The same equations of motion derived in section 3.1 and section 3.2 may be derived using MACSYMA. To accomplish this, the procedure for deriving the equations of motion is translated into instructions written in MACSYMA commands. These instructions are stored in program modules. By executing these program modules in the MACSYMA environment, the symbolic equations of motion of this case study may be obtained.
In order to organize the symbolic derivation of the equations of motion in an orderly fashion, the derivation is separated into main program modules and sub-modules. Figure 3.2 shows the flow charts of the symbolic derivation programs of this case study. The listings of these MACSYMA program modules are included in Appendix D. A brief description of each program modules is discussed in section 3.4.1. The procedure of deriving the equations of motion symbolically is explained in section 3.4.2.

3.4.1 MACSYMA Program Modules for the Derivation of the Equations of Motion for a Two-link Robot With One Flexible Outer-most Link

The following gives a brief description of each derivation program module.

`2links.bat` This is the UNIX executable file that invokes the main derivation program modules.

`Link1.com` This is the main module for link 1. Other modules are called from this main program to setup the equations of motion of link 1.

`Link2.com` This is the main module for link 2. Other modules are called from this main program to setup the equations of motion of link 2.

`Init.com` This module helps to initialize variables for all links.

`Varsetup1.com` This module helps to setup the displacement and velocity vectors of link 1.

`Varsetup2.com` This module helps to setup the displacement and velocity vectors of link 2.
Figure 3.2 Flow Chart of the MACSYMA Program Modules
This module helps to setup the energy equations and the Lagrange operator of link 1.

Energy2.com

This module helps to setup the energy equations and the Lagrange operator of link 2.

Lagrange.com

This module is called by either energy1.com or energy2.com. This module uses the Lagrange operator to produce the equations of motion.

Coeff1.com

This module calls other modules to manipulate the equations of motion of link 1 in order to obtain the coefficients of the equations.

Coeff2.com

This module calls other modules to manipulate the equations of motion of link 2 in order to obtain the coefficients of the equations.

Killvar.com

This module helps to remove the content of the time derivatives of the generalized coordinate variables from memory.

Simplify2.com

This module along with module subsimp2.com perform simple expression substitutions for each equation of link 2. The expressions substituted are the time derivatives of the generalized coordinates.

Subsimp2.com

This module is called by simplify2.com. It performs simple expression substitutions for the current equation.

Coeff.com

This module helps to extract the coefficients from the current equation of motion.

Save1.com

This module helps to create output listings of link 1 and saves the results into data file.
Save2.com This module helps to create output listings of link 2 and saves the results into data file.

Listing.com This module is called from save1.com or save2.com. It creates the output listings of the system's equations for either link 1 or link 2.

Pickapart.com This module helps to break each equation into smaller entities.

Reduce.com This is the main module that calls other modules to substitute repeated expressions into variable names and to translate the equations into FORTRAN statements.

Reduce1.com This module helps to substitute repeated expressions of link 1 into variable names. It also translates the equations into FORTRAN statements.

Reduce2.com This module helps to substitute repeated expressions of link 2 into variable names. It also translates the equations into FORTRAN statements.

Subreduce2.com This module is called by reduce2.com. It helps to substitute repeated expressions into variable names for the current equation.

Sumup.com This module creates the equations that summarized the equations of motion of link 1 and link 2.

3.4.2 Symbolic Derivation

The derivation of the equations of motion of link 1 and link 2 in this case study are very similar. However, the derivation of link 2 provides a more thorough explanation. So the discussion of the symbolic derivation of this case study begins
with \textit{link2.com}. \textit{Link2.com} is the main program module for the symbolic derivation of link 2. This program is merely an organizer that calls other modules to perform useful work. The first module called by \textit{link2.com} is \textit{init.com}. \textit{Init.com} helps to define the cross product, the gravity field and the frame transformation process that are needed in the remaining modules. Next, \textit{link2.com} calls module \textit{varsetup1.com} and \textit{varsetup2.com} to setup the parameters of link 1 and link 2. The parameters of link 1 are the displacement vector and the velocity vectors (the velocity vectors are zero in this case). The parameters of link 2 are the generalized coordinate variables, the displacement vectors, the cross products, and the velocity vectors. After these kinematics variables are set, module \textit{energy2.com} is called to formulate the Lagrange operator of link 2 using the kinematics variables. From the Lagrange operator, the equations of motion are obtained by executing module \textit{lagrange.com}.

In order to rearrange the equations of motion into the form of (2.19), module \textit{coeff2.com} is called. This module extracts the coefficients of \(\ddot{q}\)'s from the equations of motion. In the case, (2.19) becomes:

\begin{equation}
D_{11} \ddot{q}_{v_2} + F_2 = 0
\end{equation}

The extraction of \(F_2\) from the (3.37) is accomplished by setting \(\ddot{q}_{v_2}\) to zero and then evaluate (3.37). \(D_{11}\) of (3.37) may be obtained by setting \(\ddot{q}_{v_2}\) to one, then set \(g, E_2, I_{v_2}, \dot{q}_{v_2}, \dot{\theta}_2,\) and \(\ddot{\theta}_2\) to zero, and evaluate (3.37).

The last module called by \textit{link2.com} is \textit{save2.com}. \textit{Save2.com} helps to create the output listing of the equations of motion of link 2. \textit{Save2.com} also saves the symbolic equations of motion of link 2.
The equations of motion contain many integrals, therefore, numerical integration of these integrals is necessary. Before integration, these integrals need to be extracted from the equations. To separate an integral from an equation, the content of the integral is extracted from the equation using the MACSYMA's \texttt{part()} command. Each integrand is assigned a new name. Each integral in the original equation is replaced with another variable name. The following demonstrate the extraction process of the integral.

Let

\[
\text{Torque2} = A + B \int C \, dt
\]  

(3.38)

By executing module \texttt{pickapart.com}, the original equation is reassigned as in (3.39). Each partition of (3.38) is represented by a new variable as in (3.40) and (3.41).

\[
\text{Torque2} = e_1 + e_2
\]  

(3.39)

\[
e_1 = A
\]  

(3.40)

\[
e_2 = B \int C \, dt
\]  

(3.41)

Using the \texttt{part()} command, the integrand \(C\) of (3.41) is separated from the integral. The value of the integral in (3.41) is obtained by numerically integrating \(C\) using FORTRAN subroutine. The result of the integration is stored in the variable \texttt{integral}, or

\[
e_2 = B \texttt{integral}
\]  

(3.42)

Finally, the result of (3.38) is obtained by evaluating (3.40), (3.42), and (3.39).
The next step is to translate the equations into FORTRAN statements. However, some of the variables or expressions need to be renamed first in order to be compatible with the FORTRAN language. Expressions such as \( \theta(t) \), \( q(t) \) and \( q(t) \) are either a function of time or a function of space. If these variables are translated into FORTRAN in their current form, they will be interpreted as array variables. Also, expressions such as \( \sin(\theta(t)) \), \( \cos(\theta(t)) \), \( q(t) \) are found repeatedly in the equations of motion. If these variables are translated into FORTRAN in their current form, the execution time of the FORTRAN programs will be prolonged due to the repeated calculations of these expressions. Therefore, each of these expressions is replaced by a single variable before the translation. For example, \( \sin(\theta(t)) \) is replaced by \( \sin\theta \) and \( q(t) \) is replaced by \( q(t) \). After the expressions are translated into FORTRAN, these new variables are calculated before the execution of the equations of motion. Hence, these expressions need to be evaluated only once in the FORTRAN programs. The module that helps to breakdown down the equations of motion into manageable entities is pickapart.com. The main module that performs the extraction of integrals, expression substitutions and translation to FORTRAN statements is reduce2.com.

3.5 Illustrative Examples

To demonstrate the result of this case study, two robots with difference link dimension are considered. The specifications of the two robots are listed in table 3.2 and table 3.3 respectively. For both cases, the starting and the final joint angular displacements are identical. Also, for each case, output data are collected for the Bang—bang acceleration profile and the polynomial acceleration profile.
***** System Parameters *****

Units are in SI unit : kg, meter and second

Link 1 Information

Total length of link 1 (m) = 0.64770

Link 2 Information

Total length of link 2 (m) = 1.65095

For the rigid portion of link 2

Length (m) = 0.40000
Cross sectional dimension
   i) Inside width x height (m) = 0.02540 X 0.02540
   ii) Outside width x height (m) = 0.06350 X 0.06350
Cross section area (m**2) = 0.33871e-02
Density (kg/m**3) = 7860.0000
Mass per unit length (kg/m) = 26.6253
Mass (kg) = 10.64901

For the elastic portion of link 2

Modulus of elasticity (Pa) = 0.20000e+12
Length (m) = 1.25095
Cross sectional dimension
   i) Inside width x height (m) = 0.02540 X 0.02540
   ii) Outside width x height (m) = 0.03810 X 0.03810
Cross section area (m**2) = 0.80645e-03
Density (kg/m**3) = 7860.0000
Mass per unit length (kg/m) = 6.33870
Mass (kg) = 7.92939
Area moment of inertia (m**4) = 0.14091e-06
Lump mass at the end of link 2 (kg) = 25.00000

General Information

Initial position of link 2 (rad) = 0.00000
Initial position of link 2 (deg) = 0.00000
Final position of link 2 (rad) = 1.57080
Final position of link 2 (deg) = 90.00000
Duration of the experiment (s) = 0.80000
Duration of motion (s) = 0.40000
Number of steps within the experiment time frame = 800.00000
Sampling interval (s) = 0.00100
The initial value of the step size H for IMSL = 0.50000e-05
The maximum steps for IMSL = 20000.00000
The tolerance (TOL) for IMSL = 0.10000e-05

Table 3.2 System Specifications of the First Robot
***** System Parameters *****

Units are in SI unit: kg, meter and second

Link 1 Information

Total length of link 1 (m) = 0.50000

Link 2 Information

Total length of link 2 (m) = 1.25000

For the rigid portion of link 2

Length (m) = 0.25000
Cross sectional dimension
  i) Inside width x height (m) = 0.00000 X 0.00000
  ii) Outside width x height (m) = 0.05000 X 0.05000
Cross section area (m**2) = 0.25000e-02
Density (kg/m**3) = 7860.00000
Mass per unit length (kg/m) = 19.65000
Mass (kg) = 4.91250

For the elastic portion of link 2

Modulus of elasticity (Pa) = 0.20000e+12
Length (m) = 1.00000
Cross sectional dimension
  i) Inside width x height (m) = 0.00000 X 0.00000
  ii) Outside width x height (m) = 0.02540 X 0.02540
Cross section area (m**2) = 0.64516e-03
Density (kg/m**3) = 7860.00000
Mass per unit length (kg/m) = 5.07096
Mass (kg) = 5.07096
Area moment of inertia (m**4) = 0.34686e-07
Lump mass at the end of link 2 (kg) = 5.00000

General Information

Initial position of link 2 (rad) = 0.00000
Initial position of link 2 (deg) = 0.00000
Final position of link 2 (rad) = 1.57080
Final position of link 2 (deg) = 90.00000
Duration of the experiment (s) = 0.80000
Duration of motion (s) = 0.40000
Number of steps within the experiment time frame = 800.00000
Sampling interval (s) = 0.00100
The initial value of the step size H for IMSL = 0.50000e-05
The maximum steps for IMSL = 20000.00000
The tolerance (TOL) for IMSL = 0.10000e-05

Table 3.3 System Specifications of the Second Robot
In the following pages, figure 3.3 through figure 3.5 display the input acceleration, velocity and displacement profiles respectively for both set of data. Figure 3.6 shows the end-effector deflections of the link of the first robot, and figure 3.7 displays the torques at the joint necessary to sustain this motion of this robot. Similarly, figure 3.8 and figure 3.9 display the deflections and the torques for the second robot respectively.
Figure 3.3 Acceleration Profiles
Figure 3.4 Velocity Profiles
Figure 3.5 Displacement Profiles
Figure 3.6 End-effector Deflections of the First Robot Due to the Bang-bang and the Polynomial Acceleration Profiles
Figure 3.7 Torques of the First Robot at Joint 2 Due to the Bang–bang and the Polynomial Acceleration Profiles
Figure 3.8 End-effector Deflections of the Second Robot Due to the Bang-bang and the Polynomial Acceleration Profiles
Figure 3.9 Torques of the Second Robot at Joint 2 Due to the Bang—bang and the Polynomial Acceleration Profiles
The following observations were noted from evaluating the output data.

1) Comparing the end-effector deflections of the two robots (figure 3.6 and figure 3.8), it is observed that, given the same angular displacement, different input acceleration profiles yield different link deflections. During the robot motion time frame (0 to 0.4 second), the amount of deflection produced by the polynomial acceleration profiles are much larger than those produced by the bang-bang acceleration profiles. The large deflection produced by the polynomial acceleration profiles during this time frame are mostly due to the higher amount of acceleration (figure 3.3).

2) From figure 3.6, the steady-state deflections produced by the bang-bang acceleration profile is about two times as large as the one produced by the polynomial acceleration profile. However, by keeping the same input profiles but with a different robot dimension, the bang-bang profile and the polynomial profile yield almost the same magnitude of steady-state deflections (figure 3.8). From these results, it may be concluded that the deflection of a flexible robot is very sensitive to the input parameters.

3) It is observed that the steady-state deflections of the link depends on the condition at the point where the motion of the robot ends (0.4 second of figure 3.6 and figure 3.8). At this point, it governs the magnitude of the steady-state deflections.

4) The curves of the torques (figure 3.7 and figure 3.9) and the curves of the deflections (figure 3.6 and figure 3.8) are mirror images along the zero axis. This means that a positive torque is needed to counter the negative deflection of the link and vice verse.
CHAPTER FOUR

MODELING OF A THREE-LINK ROBOT WITH TWO FLEXIBLE OUTER-MOST LINKS

The purpose of this chapter is to describe and model a three-link robot. Figure 4.1 depicts such a system. The first link of the system is rigid, while the two outer-most, serially connected links are flexible. Rotary actuators are used to move the linkage. In this study, only the in-plane vibration motions of link 2 and link 3 are being considered. That is, the deflection of link 2 in the $y_2$-direction, $v_2$, and the deflection of link 3 in the $y_3$-direction, $v_3$, are studied.

Due to the complexity of the robot considered in this chapter, the equations of motion are derived by MACSYMA. No manual derivation is provided.
Figure 4.1 Three-link Robot With Two Flexible Outer-most Links
4.1 Kinematic Analysis of a Three–Link Robot With Two Flexible Outer–most Links

From figure 4.1, the Denavit–Hartenberg parameters (Appendix A) of this robot are:

<table>
<thead>
<tr>
<th>Link i</th>
<th>( \alpha_i )</th>
<th>( a_i )</th>
<th>( \theta_i )</th>
<th>( d_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( \theta_1(t) )</td>
<td>( d_1 )</td>
</tr>
<tr>
<td>1</td>
<td>( \frac{\pi}{2} )</td>
<td>0</td>
<td>( \theta_2(t) )</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>( a_2 )</td>
<td>( \theta_3(t) )</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>( a_3 )</td>
<td>( \theta_3(t) )</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.1 Denavit–Hartenberg Parameters of a Three Links Robot

4.1.1 Kinematic Analysis of Link 1

4.1.1.1 Displacement Vector of Link 1

Since link 1 is rigid as in chapter 3, the displacement vector of link 1 is:

\[
\vec{V}_1 = 0 \hat{i}_1 + 0 \hat{j}_1 - w_1 \hat{k}_1 = (0 \quad 0 \quad -w_1)^T
\]  (4.1)

Transform the displacement vector of link 1 into frame 2 yield:

\[
\vec{V}_{1+2} = T_{1+2} \cdot \vec{V}_1
\]  (4.2)

Refer to section 3.1.1.1 for the discussion of \( T_{1+2} \).
4.1.1.2 Velocity Vectors of Link 1

The angular velocity of link 1 is:

\[
\underline{\Omega}_1 = 0 \hat{i}_1 + 0 \hat{j}_1 + \dot{\theta}_1 \hat{k}_1 = (0 \quad 0 \quad \dot{\theta}_1)^T
\]  
(4.3)

or

\[
\underline{\Omega}_{1 \rightarrow 2} = T_{1 \rightarrow 2} \cdot \underline{\Omega}_1
\]  
(4.4)

where \(\underline{\Omega}_{1 \rightarrow 2}\) is the angular velocity vector of link 1 in frame 2.

From (2.6), the velocity vector of link 1 becomes:

\[
\dot{\underline{V}}_{1 \rightarrow 2} = \dot{\underline{V}}_0 + \left[ \underline{\Omega}_{1 \rightarrow 2} \times \dot{\underline{V}}_{1 \rightarrow 2} \right]
\]  
(4.5)

Again, since the base frame is fixed,

\[
\dot{\underline{V}}_0 = 0
\]  
(4.6)

=>

\[
\dot{\underline{V}}_{1 \rightarrow 2} = \left[ \underline{\Omega}_{1 \rightarrow 2} \times \dot{\underline{V}}_{1 \rightarrow 2} \right]
\]  
(4.7)

4.1.2 Kinematic Analysis of Link 2

4.1.2.1 Displacement Vectors of Link 2

As in chapter 3, link 2 is composed of two portions. The displacement vectors of link 2 are:
\[
\bar{V}_{2r} = u_2 \hat{i}_2 + 0 \hat{j}_2 + 0 \hat{k}_2 = (u_2 \ 0 \ 0)^T \tag{4.8}
\]

\[
\bar{V}_{2e} = u_2 \hat{i}_2 + v_2(u_2,t) \hat{j}_2 + 0 \hat{k}_2 = \begin{bmatrix} u_2 \\ v_2(u_2,t) \\ 0 \end{bmatrix} \tag{4.9}
\]

\[
v_2(u_2,t) = \gamma_{v_2}(u_2) q_{v_2}(t) \tag{4.10}
\]

For this case study, it is assumed that link 3 and lumped mass 3 behave similar to lumped masses at the end of link 2. Also, since the joint 3 axis is normal to the in-plane, therefore, no bending moment exists at the end of link 2. From these observations, the admissible function \( \gamma_{v_2} \) is represented by case 1 of Appendix C. From case 1 of Appendix C, (C.6) becomes:

\[
sm = \text{lm}_2 + \rho_3 a_3 + \text{lm}_3 \tag{4.11}
\]

where \( \text{lm}_2 \) is the lumped mass at the end of link 2.
\( \rho_3 \) is the mass per unit length of link 3.
\( a_3 \) is the total length of link 3.
\( \text{lm}_3 \) is the lumped mass at the end of link 3.

### 4.1.2.2 Velocity Vectors of Link 2

The angular velocity vector of link 2 is represented as:

\[
\bar{\Omega}_2 = \bar{\Omega}_{1 \rightarrow 2} + (0 \ 0 \ \dot{\theta}_2)^T \tag{4.12}
\]
From (2.6) and (2.7), the velocity vectors of link 2 becomes:

\[
\dot{V}_{2r} = \dot{V}_{1+2}(0) + \left[ \Omega_2 \times \bar{V}_{2r} \right] \\
\dot{V}_{2e} = \dot{V}_{1+2}(0) + \left[ \Omega_2 \times \bar{V}_{2e} \right] + \dot{v}_2(u_2,t) (0 \ 1 \ 0)^T
\] (4.13)

(4.14)

4.1.3 Kinematic Analysis of Link 3

The kinematic of link 3 is very similar to link 2. It is assumed that link 3 is entirely flexible.

4.1.3.1 Displacement Vector of Link 3

The displacement vector of link 3 is:

\[
\bar{V}_3 = u_3 \bar{i}_3 + v_3(u_3,t) \bar{j}_3 + 0 \bar{k}_3 = \begin{bmatrix} u_3 \\
v_3(u_3,t) \\
0 \end{bmatrix}
\] (4.15)

where \( u_3 \) is the displacement of link 3 along the \( x_3 \)-axis.

\( v_3(u_2,t) \) is the deflection of link 3 in the \( y_3 \)-direction.

From the assumed mode method (Appendix B) and the discussions of the generalized coordinates in section 2.2, \( v_3 \) may be expressed as:

\[
v_3(u_3,t) = \gamma_{v_3} (u_3) q_{v_3}(t)
\] (4.16)
where \( \gamma_{v_3}(u_2) \) is the admissible function of link 3 in the \( y_3 \)-direction.

\( q_{v_3}(t) \) is the time-dependent generalized coordinate of link 3 in the \( y_3 \)-direction.

Link 3 may be modeled as a cantilever beam with shearing mass at the end of the beam. Using this assumption, case 1 of Appendix C is used to formulate an expression for the admissible function \( \gamma_{v_3} \). From case 1 of Appendix C, (C.6) becomes:

\[
sm = lm_3 \tag{4.17}
\]

Substituting (4.17) into (C.6) of Appendix C will yield the proper admissible function of link 3.

It is necessary to transform the displacement vector of link 3 into frame 2. Substituting the Denavit–Hartenberg parameters of table 4.1 into the orientation portion of (A.6) yields:

\[
T_{3\rightarrow2} = \begin{bmatrix}
\cos(\psi_3) & -\sin(\psi_3) & 0 \\
\sin(\psi_3) & \cos(\psi_3) & 0 \\
0 & 0 & 1
\end{bmatrix}
\tag{4.18}
\]

\[
\psi_3 = \theta_3(t) + v_2'(a_2,t) \tag{4.19}
\]

where \( T_{3\rightarrow2} \) is the transformation matrix of frame 3 to frame 2.

The displacement vector of link 3 becomes:
\[ \vec{V}_{3 \rightarrow 2} = T_{3 \rightarrow 2} \cdot \begin{bmatrix} u_3 \\ v_3(u_3,t) \\ 0 \end{bmatrix} \]  

where \( \vec{V}_{3 \rightarrow 2} \) is the displacement vector of link 3 in frame 2.

### 4.1.3.2 Velocity Vectors of Link 3

The angular velocity vector of link 3 becomes:

\[ \vec{\Omega}_{3 \rightarrow 2} = \vec{\Omega}_{2}(\vec{a}_2) + T_{3 \rightarrow 2} \cdot (0 \quad 0 \quad \dot{\theta}_3) \quad T \]  

From (2.7), the velocity vector of link 3 becomes:

\[ \vec{\dot{V}}_{3 \rightarrow 2} = \vec{V}_2(\vec{a}_2) + [\vec{\Omega}_{3 \rightarrow 2} \times \vec{V}_{3 \rightarrow 2}] + \vec{v}_3(u_3,t) \cdot \vec{p}_{v_3} \]  

\[ \vec{p}_{v_3} = T_{3 \rightarrow 2} \cdot (0 \quad 1 \quad 0) \quad T \]  

where \( \vec{p}_{v_3} \) is the projection of an unit vector onto the \( y_2 \)-axis.
4.2 Dynamic Analysis of a Three-link Robot With Two Flexible Outer-most Links

4.2.1 Kinetic and Potential Energy Equations of Link 1

The kinetic energy equation of link 1 is:

$$KE_1 = \frac{1}{2} J_1 \dot{\theta}_1$$  \hspace{1cm} (4.24)

where $J_1$ is the mass moment of inertia of link 1.

The potential energy equation of link 1 is:

$$PE_1 = -\rho_1 \int_0^{d_1} \left[ \frac{-1}{V_{1\rightarrow 2}} \cdot \frac{-1}{V_{1\rightarrow 2}} \right] dw_1$$  \hspace{1cm} (4.25)

4.2.2 Kinetic and Potential Energy Equations of Link 2

The kinetic energy equation of link 2 is:

$$KE_2 = \frac{1}{2} \rho_{2r} \int_{a_2r}^{a_2r} \left[ \frac{-\dot{V}_{2r}}{V_{2r}} \cdot \frac{-\dot{V}_{2r}}{V_{2r}} \right] du_2$$

$$+ \rho_{2e} \int_{a_2r}^{a_2r} \left[ \frac{-\dot{V}_{2e}}{V_{2e}} \cdot \frac{-\dot{V}_{2e}}{V_{2e}} \right] du_2 + \frac{1}{2} sm_2 \left[ \frac{-\dot{V}_{2e(a_2)}}{V_{2e(a_2)}} \cdot \frac{-\dot{V}_{2e(a_2)}}{V_{2e(a_2)}} \right]$$  \hspace{1cm} (4.26)

$$sm_2 = l_2m_2 + \rho_3 a_3 + l_3m_3$$  \hspace{1cm} (4.27)

where $sm_2$ is the total shearing mass at the end of link 2.
The potential energy equation of link 2 is:

\[ \text{PE}_2 = \frac{1}{2} E_2 I_v \int_{a_{2r}}^{a_2} \left[ v_2''(u_2,t) \right]^2 du_2 \]

\[ - \rho_{2r} \int_0^{a_{2r}} \left[ \frac{G_{1+2}}{V_{2r}} \right] du_2 - \rho_{2e} \int_{a_{2r}}^{a_2} \left[ \frac{G_{1+2}}{V_{2e}} \right] du_2 \]

\[ - \text{sg}_2 \left[ \frac{G_{1+2}}{V_{2e}(a_2)} \right] \]

\[ \text{(4.28)} \]

### 4.2.3 Kinetic and Potential Energy Equations of Link 3

The kinetic energy equation of link 3 is:

\[ \text{KE}_3 = \frac{1}{2} \rho_3 \int_0^{a_3} \left[ \frac{\dot{V}_{3+2}}{V_{3+2}} \cdot \frac{\dot{V}_{3+2}}{V_{3+2}} \right] du_3 \]

\[ + \frac{1}{2} \text{sg}_3 \left[ \frac{\dot{V}_{3+2}(a_3)}{V_{3+2}(a_3)} \cdot \frac{\dot{V}_{3+2}(a_3)}{V_{3+2}(a_3)} \right] \]

\[ \text{(4.29)} \]

\[ \text{sm}_3 = \text{lm}_3 \]

\[ \text{(4.30)} \]

where \( \rho_3 \) is the mass per unit length of link 3.

\( \text{sm}_3 \) is the total shearing mass at the end of link 3.
The potential energy equation of link 3 is:

\[
\begin{align*}
PE_3 &= \frac{1}{2} E_3 I_{v3} \int_0^{a_3} \left[ v_3''(u_3, t) \right]^2 du_3 \\
&- \rho_3 \int_0^{a_3} \left[ \bar{G}_{1\rightarrow 2} \cdot \bar{v}_{3\rightarrow 2} \right] du_3 - \sigma_{m3} \left[ \bar{G}_{1\rightarrow 2} \cdot \bar{v}_{3\rightarrow 2}(a_3) \right] \\
&\hspace{1cm} (4.31)
\end{align*}
\]

where \( E_3 \) is the modulus of elasticity of link 3.

\( I_{v3} \) is the area moment of inertia of link 3 in the \( y_3 \)-direction.

### 4.2.4 Lagrangian Equations

From the kinematics of this system, there are five time-dependent generalized coordinates. They are \( q_{v2}(t), q_{v3}(t), \theta_1(t), \theta_2(t), \) and \( \theta_3(t) \). Substitute these coordinates into (2.12) yield:

\[
\begin{align*}
\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\theta}_1} \right) - \frac{\partial L}{\partial \theta_1(t)} &= \text{Torque}_1 \\
\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\theta}_2} \right) - \frac{\partial L}{\partial \theta_2(t)} &= \text{Torque}_2 \\
\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\theta}_3} \right) - \frac{\partial L}{\partial \theta_3(t)} &= \text{Torque}_3 \\
\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_{v2}} \right) - \frac{\partial L}{\partial q_{v2}(t)} &= 0
\end{align*}
\]
\[
\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_{v_3}} \right) - \frac{\partial L}{\partial q_{v_3}(t)} = 0 \quad (4.36)
\]

where
\[
L = KE_1 + KE_2 + KE_3 - PE_1 - PE_2 - PE_3 \quad (4.37)
\]

Substituting the kinetic and potential energy equations into (4.37) yields the Lagrange operator of the robotic system. Applying the Lagrange operator into (4.32) through (4.36) yield the equations of motion of the system.

### 4.3 Initial Conditions

The discussion of the initial conditions of this case study is similar to the one in section 3.4. The initial values for \(q_{v_2}(0)\) may be determined from the static deflection equation:

\[
v_{2}(a_2,0) = \left[ -\rho_{2e} g \frac{a_{2e}}{8 E_2 I_{v_2}} \right] + \left[ -\frac{sm_2 g a_{2e}}{3 E_2 I_{v_2}} \right] \cos(\theta_2(0)) \quad (4.38)
\]

for
\[
a_{2e} = a_2 - a_{2r} \quad (4.39)
\]

\[
q_{v_2}(0) = \frac{v_{2}(a_2,0)}{\gamma_{v_2}(a_2)} \quad (4.40)
\]

As a reminder, \(sm_2\) in (4.38) is not the same as the one in (3.34). The static deflection equation for link 3 is:
\[ v_3(a_3,0) = \left[ -\frac{\rho_3 e g a_3}{8 E_3 I_{v_3}^4} \right] + \left[ -\frac{\text{sm}_3 g a_3}{3 E_3 I_{v_3}^3} \right] \cos(\chi) \quad (4.41) \]

for
\[ \chi = \theta_2(0) + v_2'(a_2,0) + \theta_3(0) \quad (4.42) \]

\[ => q_{v_3}(0) = \frac{v_3(a_3,0)}{\gamma_{v_3}(a_3)} \quad (4.43) \]

Initially, link 2 and link 3 are stationary, therefore:

\[ \dot{q}_{v_2}(0) = \dot{q}_{v_3}(0) = 0 \quad (4.44) \]

### 4.4 MACSYMA Programs

A number of MACSYMA program modules in the previous chapter may also be used in this case study. Some of these program modules are slightly modified to fit the specification of the robot of this case study. For example, the content of link1.com, energy1.com, lagrange.com, coeff2.com, coeff.com, save2.com, ... are modified. Along with the changes, new program modules are created for link 3. Figure 4.2 displays the flow chart of the MACSYMA program modules for this case study. The listings of these MACSYMA program modules are included in Appendix E. A brief description of the new program modules are discussed in section 4.4.1. The symbolic derivation of the equations of motion for this case study is briefly explained in section 4.4.2.
Figure 4.2 Flow Chart of the MACSYMA Program Modules
4.4.1 MACSYMA Program Modules for the Derivation of the Equations of Motion for a Three–link Robot With Two Flexible Outer–most Links

The following gives a brief description of the new program modules. Other modules are listed in section 3.4.1.

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Slinks.bat</code></td>
<td>This is the UNIX executable file that invokes the main derivation program modules.</td>
</tr>
<tr>
<td><code>Link3.com</code></td>
<td>This is the main module for link 3. Other modules are called from this main program to setup the equations of motion of link 3.</td>
</tr>
<tr>
<td><code>Varsetup3.com</code></td>
<td>This module helps to setup the displacement and velocity vectors of link 3.</td>
</tr>
<tr>
<td><code>Energy3.com</code></td>
<td>This module helps to setup the energy equations and the Lagrange operator of link 3.</td>
</tr>
<tr>
<td><code>Coeff3.com</code></td>
<td>This module calls other modules to manipulate the equations of motion of link 3 in order to obtain the coefficients of the equations.</td>
</tr>
<tr>
<td><code>Simplify3.com</code></td>
<td>This module along with module <code>subsimp3.com</code> perform simple expression substitutions for each equation of link 3. The expressions substituted are the time derivatives of the generalized coordinates.</td>
</tr>
<tr>
<td><code>Subsimp3.com</code></td>
<td>This module is called by <code>simplify3.com</code>. It performs simple expression substitutions for the current equation.</td>
</tr>
<tr>
<td><code>Save3.com</code></td>
<td>This module helps to create output listings of link 3 and saves the results into data file.</td>
</tr>
</tbody>
</table>
Reduce.com This module helps to substitute repeated expressions of link 3 into variable names. It also translates the equations into FORTRAN statements.

Subreduce3.com This module is called by reduce3.com. It helps to substitute repeated expressions into variable names for the current equation.

Error.com This module helps to set up the equations for the error analysis.

4.4.2 Symbolic Derivation

The symbolic derivation of the equations of motion of this case study is very similar to the one discussed in the previous chapter. Therefore, the discussion of the derivation will not be repeated. However, with the addition of link 3, (2.19) becomes:

\[
\begin{bmatrix}
D_{11} & D_{12} \\
D_{21} & D_{22}
\end{bmatrix}
\begin{bmatrix}
\ddot{q}_v^2 \\
\ddot{q}_v^3
\end{bmatrix}
+ 
\begin{bmatrix}
F_4 \\
F_5
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
0
\end{bmatrix}
\]

(4.45)

where

\(D_{11}\) is the coefficient of \(\ddot{q}_v^2\) of (4.35).

\(D_{12}\) is the coefficient of \(\ddot{q}_v^3\) of (4.35).

\(D_{21}\) is the coefficient of \(\ddot{q}_v^2\) of (4.36).

\(D_{22}\) is the coefficient of \(\ddot{q}_v^3\) of (4.36).
In order to extract $F_4$ and $F_5$ from (4.45), $\ddot{q}_{v_2}$ and $\ddot{q}_{v_3}$ are first set to zero, and then (4.45) is evaluated. The expressions for $D_{11}, D_{12}, D_{21},$ and $D_{22}$ may be obtained by first setting $g, E_2, E_3, I_{v_2}, I_{v_3}, \dot{q}_{v_2}, \dot{q}_{v_3}, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3,$ and $\ddot{\theta}_3$ to zero. For $D_{11}$ and $D_{21}$, $\ddot{q}_{v_2}$ is set to one, $\ddot{q}_{v_3}$ is set to zero and then evaluate (4.45). Similarly, $D_{12}$ and $D_{22}$ may be obtained by setting $\ddot{q}_{v_2}$ to zero, $\ddot{q}_{v_3}$ to one and then evaluate (4.45).

A new program module, \texttt{error.com}, is included in this case study. \texttt{Error.com} is created to formulate two sets of end-effector trajectory coordinate equations which are measured in the base frame. The first set of trajectory coordinate equations is that of the end-effector when all the robot links are rigid. The second set of trajectory coordinate equations is that of the end-effector when some of the robot links are flexible. The trajectory coordinate equations for a robot are formulated using the Denavit–Hartenberg parameters and the transformation matrices discussed in Appendix A.

4.5 Illustrative Example

In this case study, only one set of data is taken. The system specifications of the robot are listed in table 4.2. The input profiles of each joint are shown in figure 4.3 through figure 4.11. The deflections of link 2 and link 3 are shown in figure 4.12 and figure 4.13 respectively. The torques at joints 1, 2 and 3 are shown in figure 4.14 through figure 4.16 respectively. The end-effector trajectories are shown in figure 4.17 (Bang–bang profile) and figure 4.18 (Polynomial profile). Finally, the end-effector errors are shown in figure 4.19 through figure 4.21.
***** System Parameters *****

Units are in SI unit: kg, meter and second

Link 1 Information

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length (m)</td>
<td>0.64770</td>
</tr>
<tr>
<td>Inside diameter (m)</td>
<td>0.15000</td>
</tr>
<tr>
<td>Outside diameter (m)</td>
<td>0.22640</td>
</tr>
<tr>
<td>Cross section area (m**2)</td>
<td>0.22586e-01</td>
</tr>
<tr>
<td>Density (kg/m**3)</td>
<td>7860.00000</td>
</tr>
<tr>
<td>Mass per unit length (kg/m)</td>
<td>177.52332</td>
</tr>
<tr>
<td>Mass (kg)</td>
<td>114.98186</td>
</tr>
<tr>
<td>Mass moment of inertia (kg*m**2)</td>
<td>0.41332e+00</td>
</tr>
</tbody>
</table>

Link 2 Information

Total length of link 2 (m) = 1.65095

For the rigid portion of link 2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length (m)</td>
<td>0.40000</td>
</tr>
<tr>
<td>Cross sectional dimension</td>
<td></td>
</tr>
<tr>
<td>i) Inside width x height (m)</td>
<td>0.02540 X 0.02540</td>
</tr>
<tr>
<td>ii) Outside width x height (m)</td>
<td>0.06350 X 0.06350</td>
</tr>
<tr>
<td>Cross section area (m**2)</td>
<td>0.33871e-02</td>
</tr>
<tr>
<td>Density (kg/m**3)</td>
<td>7860.00000</td>
</tr>
<tr>
<td>Mass per unit length (kg/m)</td>
<td>26.62253</td>
</tr>
<tr>
<td>Mass (kg)</td>
<td>10.64901</td>
</tr>
</tbody>
</table>

For the elastic portion of link 2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modulus of elasticity (Pa)</td>
<td>0.20000e+12</td>
</tr>
<tr>
<td>Length (m)</td>
<td>1.25095</td>
</tr>
<tr>
<td>Cross sectional dimension</td>
<td></td>
</tr>
<tr>
<td>i) Inside width x height (m)</td>
<td>0.02540 X 0.02540</td>
</tr>
<tr>
<td>ii) Outside width x height (m)</td>
<td>0.03810 X 0.03810</td>
</tr>
<tr>
<td>Cross section area (m**2)</td>
<td>0.80645e-03</td>
</tr>
<tr>
<td>Density (kg/m**3)</td>
<td>7860.00000</td>
</tr>
<tr>
<td>Mass per unit length (kg/m)</td>
<td>6.33870</td>
</tr>
<tr>
<td>Mass (kg)</td>
<td>7.92939</td>
</tr>
<tr>
<td>In-plane area moment of inertia (m**4)</td>
<td>0.14091e-06</td>
</tr>
<tr>
<td>Lump mass at the end of link 2 (kg)</td>
<td>1.00000</td>
</tr>
</tbody>
</table>

Table 4.2 System Specifications
Link 3 Information

Modulus of elasticity (Pa) = 0.20000e+12
Length (m) = 0.93345
Cross sectional dimension
  i) Inside width x height (m) = 0.02540 X 0.02540
  ii) Outside width x height (m) = 0.03810 X 0.03810
Cross section area (m**2) = 0.80645e-03
Density (kg/m**3) = 7860.00000
Mass per unit length (kg/m) = 6.33870
Mass (kg) = 5.91686
In-plane area moment of inertia (m**4) = 0.14091e-06
Lump mass at the end of link 3 (kg) = 25.00000

General Information

Initial position of link 1 (rad) = -1.81514
Initial position of link 1 (deg) = -104.00000
Final position of link 1 (rad) = 1.81514
Final position of link 1 (deg) = 104.00000
Initial position of link 2 (rad) = -0.39095
Initial position of link 2 (deg) = -22.40000
Final position of link 2 (rad) = 0.45204
Final position of link 2 (deg) = 25.90000
Initial position of link 3 (rad) = -1.60745
Initial position of link 3 (deg) = -92.10000
Final position of link 3 (rad) = 0.13788
Final position of link 3 (deg) = 7.90000
Duration of the experiment (s) = 2.00000
Duration of motion (s) = 1.00000
Number of steps within the experiment time frame = 1000.00000
Sampling interval (s) = 0.00200
The initial value of the step size H for IMSL = 0.10000e-04
The maximum steps for IMSL = 10000.00000
The tolerance (TOL) for IMSL = 0.10000e-05

Table 4.2 System Specifications Continues
Figure 4.3 Acceleration Profiles of Link 1
Figure 4.4 Acceleration Profiles of Link 2
Figure 4.5 Acceleration Profiles of Link 3
Figure 4.6 Velocity Profiles of Link 1
Figure 4.7 Velocity Profiles of Link 2
Figure 4.8 Velocity Profiles of Link 3
Figure 4.9 Displacement Profiles of Link 1
Angular Displacements (rad)

Figure 4.10 Displacement Profiles of Link 2
Angular Displacements (rad)

Bang-Bang

Polynomial

Time (s)

Figure 4.11 Displacement Profiles of Link 3
Figure 4.12 Deflections at the End of Link 2 Due to the Bang–bang and the Polynomial Acceleration Profiles
Figure 4.13 End-effector Deflections Due to the Bang-bang and the Polynomial Acceleration Profiles
Figure 4.14 Torques at Joint 1 Due to the Bang–bang and Polynomial Acceleration Profiles
Figure 4.15 Torques at Joint 2 Due to the Bang–bang and Polynomial Acceleration Profiles
Figure 4.16 Torques at Joint 3 Due to the Bang–bang and Polynomial Acceleration Profiles
Figure 4.17 Trajectories of the End-effector Using the Bang-bang Acceleration Profiles with the Trajectories Projections on the X-Y, Y-Z, and Z-X Planes
Figure 4.18 Trajectories of the End-effector Using the Polynomial Acceleration Profiles with the Trajectory Projections on the X–Y, Y–Z, and Z–X Planes
Figure 4.19 Errors on the X-axis
Figure 4.20 Errors on the Y-axis
Figure 4.21 Errors on the Z-axis
Along with the evaluations of the output data discussed in chapter 3, the following new observations were noted from evaluating the output data of this case study.

1) By comparing the deflection curves of chapter 3 with the deflection curves of figure 4.12 and figure 4.13, the deflection curves of this case study are not as smooth as those shown in chapter 3. The roughness of these curves demonstrate the interaction of link 2 and link 3.

2) From figure 4.13, the deflection frequency of link 3 is a combination of two sources. The first one is the deflection frequency of link 2 and the second one is the natural deflection frequency of link 3. The natural deflection frequency of link 3 is superimposed onto the deflection frequency of link 2.

3) The natural frequency of link 2 is lower than the natural frequency of link 3. This is because link 2 experience a larger payload than link 3. From dynamics, when the payload experience by the beam increases, the natural frequency decreases and the amount of deflection increases.
CHAPTER FIVE

MODELING OF THE UNLV-ARO LIGHT-WEIGHT ROBOT

In this chapter, the development of the kinematics and the dynamics of the UNLV-ARO light-weight robot will include the out-of-plane motions of the linkage. In the previous chapters, the discussions thus far have been concentrated on the in-plane motions of the linkage. The analysis concerning the in-plane motions will be similar to those discussed in the previous chapters, while the analysis of the out-of-plane motions requires more detailed explanations.

In this chapter, links 1, 2, and 3 are actuated by hydraulic actuators instead of rotary actuators for a more realistic simulation (figure 5.1). The first and the second hydraulic cylinder are assumed to be rigid. The first hydraulic cylinder is attached to the robot base. It drives a rack and pinion mechanism and there is no load to bend this actuator. The load that acts upon the second hydraulic cylinder is mostly axial, which does not cause any deflection. However, the piston rod of the third hydraulic cylinder experiences non-axial loads from its cylinder shell as well as the vibrations from the bending of link 2. Therefore, this hydraulic cylinder is considered flexible.

Due to the complexity of this robotic system, the equations of motion are derived by MACSYMA only, no manual derivation is presented.
Figure 5.1 UNLV—ARO Light-weight Robot.
Three-link Robot with Two Flexible Outer-most Links
and Hydraulic Actuators
5.1 Kinematic Analysis of the UNLV—ARO Light-weight Robot

From figure 5.1, the Denavit—Hartenberg parameters (Appendix A) of the UNLV—ARO light-weight robot are:

\[
\begin{array}{cccc}
\text{Link } i & a_i & a_{i-1} & \theta_i & d_i \\
0 & 0 & 0 & - & - \\
1 & \frac{\pi}{2} & 0 & \theta_1(t) & d_1 \\
2 & 0 & a_2 & \theta_2(t) & 0 \\
3 & 0 & a_3 & \theta_3(t) & 0 \\
c & 0 & a_c & \theta_c(t) & 0 \\
h3 & 0 & a_{h3}(t) & \theta_{h3}(t) & 0 \\
\end{array}
\]

Table 5.1 Denavit—Hartenberg Parameters of the UNLV—ARO Light-weight Robot

5.1.1 Kinematic Analysis of link 1

Since there is no change in the configuration of link 1, the kinematics of this link remains the same as discussed in chapter 4.

5.1.2 Kinematic Analysis of Link 2

5.1.2.1 Displacement Vectors of Link 2

Since the analysis of this case study includes the out-of-plane motions of the robot, the displacement vectors of link 2 become:
\( \overset{\text{V}_2\Gamma}{\text{V}_2\Gamma} = u_2^\text{i}_2 + 0^\text{j}_2 + 0^\text{k}_2 = (u_2 \ 0 \ 0)^{\text{T}} \) \hspace{1cm} (5.1)

\( \overset{\text{V}_2\varepsilon}{\text{V}_2\varepsilon} = u_2^\text{i}_2 + v_2(u_2,t)^\text{j}_2 + w_2(u_2,t)^\text{k}_2 = \begin{bmatrix} u_2 \\ v_2(u_2,t) \\ w_2(u_2,t) \end{bmatrix} \) \hspace{1cm} (5.2)

From the assumed mode method (Appendix B) and the discussions of the generalized coordinates in section 2.2, \( v_2 \) and \( w_2 \) may be expressed as:

\[ v_2(u_2,t) = \gamma_{v_2}^i(u_2) q_{v_2}(t) \] \hspace{1cm} (5.3)

\[ w_2(u_2,t) = \gamma_{w_2}^j(u_2) q_{w_2}(t) \] \hspace{1cm} (5.4)

where \( w_2(u_2,t) \) is the deflection of link 2 in the \( z_2 \)-direction.

\( \gamma_{w_2}^j(u_2) \) is admissible function of link 2 in the \( z_2 \)-direction.

\( q_{w_2}(t) \) is the time-dependent generalized coordinate in the \( z_2 \)-direction.

For the in-plane admissible function of link 2, the discussion is identical to the one described in section 4.1.2.1. The in-plane admissible function may be represented by case 1 of Appendix C. Hence, (C.6) for the in-plane motion becomes:

\[ \text{sm} = \text{lm}_2 + \rho_3 \text{a}_3 + \text{lm}_3 + \rho_c \text{a}_c \] \hspace{1cm} (5.5)

where \( \rho_c \) is the mass per unit length of link c.

\( \text{a}_c \) is the total length of link c.
Replacing (C.6) in case 1 of Appendix C by (5.5) will yield the proper in–plane admissible function of link 2. Note that the mass of the third hydraulic cylinder is not included in (5.5). This is because the center of gravity of the third hydraulic cylinder is located away from the end of link 2, so the shearing effect of this hydraulic cylinder is assumed to be minimal and is not considered.

In order to find an expression for the out–of–plane admissible function of link 2, it is essential to observe and understand the physical aspects of the out–of–plane motions of this link. First of all, link 2 still experiences the shearing effect due to the masses at the end of this link. However, unlike $\gamma_{v_2}$, there exists a bending moment at the end of this link. The moment is created by link 3 and lumped mass 3 during the rotation of the robot about the $z_1$–axis. Also, the amount of moment asserted onto link 2 depends greatly on the orientation of link 3 (figure 5.2a and 5.2b). Finally, one might argue that link c also contribute some bending moment onto link 2. However, the bending effect of link c is neglected because of its relatively small mass and short length. So from these observations and by redistributing the mass of link 3 and lumped mass 3, $\gamma_{w_2}$ may be determined using the discussions stated in case 2 of Appendix C. From figure 5.2a and 5.2b, (C.51) of Appendix C becomes:

\[
M = -m_{3cg} \ddot{a}_{3cg} a_3 \cos(\theta_3(t)) \quad (5.6)
\]

for

\[
m_{3cg} = \rho_3 a_3 + \text{lm}_3 \quad (5.7)
\]

\[
a_{3cg} = \frac{1}{m_{3cg}} \left[ \frac{1}{2} \rho_3 a_3 - \text{lm}_3 a_3 \right] \quad (5.8)
\]

\[
\dddot{a}_{3cg} = \ddot{w}_2(a_2, t) + a_3 \cos(\theta_3(t)) \dddot{w}_2(a_2, t) \quad (5.9)
\]
Figure 5.2a Center of Gravity of Link 3 in the X–Y Plane.

Figure 5.2b Bending Moment at the End of Link 2 Due to Link 3
where $m_{3_{cg}}$ is the mass of the bending moment.

$acc_{3_{cg}}$ is the acceleration of $m_{3_{cg}}$.

$a_{3_{cg}}$ is the length from the end of link 2 to the center of mass of the bending moment.

Replacing (C.51) and (C.54) of case 2 with (5.6) and (5.5) respectively yields:

$$
\tau_1 = -\frac{m_{3_{cg}} a_{3_{cg}}^2 \cos(\theta_3(t)) \beta_{w_2}}{\rho_{2e} (a_2 - a_{2r})^3 \beta_{w_2}^4}
$$

(5.10)

$$
\tau_2 = -\frac{m_{3_{cg}} a_{3_{cg}} \cos(\theta_3(t)) \beta_{w_2}}{\rho_{2e} (a_2 - a_{2r})^2 \beta_{w_2}^4}
$$

(5.11)

where $\tau_1$ and $\tau_2$ are the coefficients of the bending moment boundary condition of link 2.

Using (5.10) and (5.11) yields the proper out-of-plane admissible function for link 2.

5.1.2.2 Velocity Vectors of Link 2

From (2.6) and (2.7), the velocity vectors of link 2 becomes:
\[ \dot{V}_{2r} = \dot{V}_{1+2}(0) + \left[ \Omega_2 \times V_{2r} \right] \] (5.12)

\[ \dot{V}_{2e} = \dot{V}_{1+2}(0) + \left[ \Omega_2 \times V_{2e} \right] + T \begin{pmatrix} \dot{v}_2(u_2, t) \end{pmatrix} (0 \ 1 \ 0) + T \begin{pmatrix} \dot{w}_2(u_2, t) \end{pmatrix} (0 \ 0 \ 1) \] (5.13)

Refer to chapter 4 for the discussion of \( \Omega_2 \).

5.1.3 Kinematic Analysis of Link 3

5.1.3.1 Displacement Vector of Link 3

The displacement vector of link 3 is:

\[ V_3 = u_3 i_3 + v_3(u_3, t) j_3 + w_3(u_3, t) k_3 = \begin{bmatrix} u_3 \\ v_3(u_3, t) \\ w_3(u_3, t) \end{bmatrix} \] (5.14)

where \( w_3(u_3, t) \) is the deflection of link 3 in the \( z_3 \)-direction.

From the assumed mode method (Appendix B), \( v_3 \) and \( w_3 \) may be expressed as:

\[ v_3(u_3, t) = \gamma v_3(u_3, t) q_v(t) \] (5.15)

\[ w_3(u_3, t) = \gamma w_3(u_3, t) q_w(t) \] (5.16)
where \( \gamma_{w_3}(u_3) \) is the admissible function of link 3 in the \( z_3 \)-direction.

\( q_{w_3}(t) \) is the time-dependent generalized coordinate of link 3 in the \( z_3 \)-direction.

The in-plane and out-of-plane admissible functions of link 3 may be obtained using the discussions described in case 1 of Appendix C. Hence, (C.6) of Appendix C becomes:

\[
sm = lm_3
\]

Substituting (5.18) to (C.6) in Appendix C will yield the proper admissible functions for link 3. The displacement vector of link 3 in frame 2 becomes:

\[
\bar{V}_{3 \rightarrow 2} = T_{3 \rightarrow 2} \cdot \begin{bmatrix}
  u_3 \\
  v_3(u_3,t) \\
  w_3(u_3,t)
\end{bmatrix}
\]

Refer to chapter 4 for the discussion of \( T_{3 \rightarrow 2} \).

### 5.1.3.2 Velocity Vectors of Link 3

From (2.7), the velocity vector of link 3 becomes:

\[
\dot{\bar{V}}_{3 \rightarrow 2} = \dot{V}_2(a_2) + [\Omega_{3 \rightarrow 2} \times \bar{V}_{3 \rightarrow 2}] + \dot{v}_3(u_3,t) \bar{p}_v + \dot{w}_3(u_3,t) \bar{p}_w
\]
\[ \bar{p}_{v_3} = T_{3\to2} \cdot (0 \ 1 \ 0)^T \]  
\[ \bar{p}_{w_3} = T_{3\to2} \cdot (0 \ 0 \ 1)^T \]  

where \( \bar{p}_{v_3} \) is the projection vector onto the \( y_2 \)-axis. 

\( \bar{p}_{w_3} \) is the projection vector onto the \( z_2 \)-axis.

Refer to chapter 4 for the discussion of \( \Omega_{3\to2} \).

5.1.4 Kinematic Analysis of Link C

5.1.4.1 Displacement Vector of Link C

The displacement vector of link \( c \) is:

\[ V_c = u_c \ i_c + 0 \ j_c + 0 \ k_c = (u_c \ 0 \ 0)^T \]  

where \( u_c \) is the displacement of link \( c \) along the \( x_c \)-axis.

Substituting the Denavit–Hartenberg parameters of table 5.1 into the orientation portion of (A.6) yields:

\[ T_{c\to2} = \begin{bmatrix} \cos(\kappa_c) & -\sin(\kappa_c) & 0 \\ \sin(\kappa_c) & \cos(\kappa_c) & 0 \\ 0 & 0 & 1 \end{bmatrix} \]  

(5.23)
for
\[ \kappa_c = \theta_c(t) + v_2'(a_2,t) \]  
(5.24)

where \( T_{c+2} \) is the transformation matrix of frame c to frame 2.

The displacement vector of link c in frame 2 becomes:
\[ \bar{V}_{c+2} = T_{c+2} \cdot (u_c \quad 0 \quad T) \]  
(5.25)

5.1.4.2 Velocity Vectors of Link C

The angular velocity vector of link c is:
\[ \bar{\Omega}_{c+2} = \bar{\Omega}_{3+2} \]  
(5.26)

From (2.7), the velocity vectors of link c become:
\[ \dot{\bar{V}}_{c+2} = \dot{V}_2(a_2) + \left[ \bar{\Omega}_{c+2} \times \bar{V}_{c+2} \right] \]  
(5.27)

5.1.5 Kinematic Analysis of Link H3

Before any attempt to discuss the kinematics of link h3, the following assumptions and observations are needed.

1) Since this hydraulic cylinder actuates link 3, its total length, \( a_{h3} \), is time dependent, or
For the purpose of simplicity, link h3 is assumed to be composed of three major parts. The hydraulic cylinder shell, a piston inside the shell, and a piston rod (figure 5.3).

3) The shell includes the outer layer of the hydraulic cylinder, the piston and part of the piston rod. The hydraulic cylinder shell is assumed to be rigid.

4) The piston rod of the hydraulic cylinder is assumed to be flexible.

5) The weight of the hydraulic cylinder fluid is neglected.

6) The hydraulic cylinder is supported by two revolute joints whose axes are normal to the robot plane (figure 5.1), so the deflection in the $y_{h3}$-direction of this link is largely due to the mass of the cylinder shell and the deflection motion of link \(2\).

7) The joints that connect link 2 with link c and link h3 with link c are fork-end joints. The axes of these two joints are parallel to each other and are normal to the plane of motion. Because link c is rigid, the deflection of link h3 is closely related with the deflection of link 2. Furthermore, the slope of the deflection at the end of link h3 is identical to the slope of the deflection at the end of link 2 (figure 5.4).
Figure 5.3 Cross Section of Hydraulic Cylinder 3

Figure 5.4 Out-of-plane Relationship of Link 2 and Link h3
5.1.5.1 Displacement Vectors of Link H3

In order to study the dynamics of the hydraulic cylinder, its translation and rotational velocities \( \dot{a}_{h3} \) and \( \dot{\theta}_{h3} \) are needed. This means that \( a_{h3} \) and \( \theta_{h3} \) should be calculated first. The following states the inter-relationship between them. From figure 5.5:

\[
a_{h3}(t) = \left[ \frac{h_{3y}^2 + h_{3x1}^2}{2} \right]^{1/2}
\]
(5.29)

\[
\theta_{h3}(t) = (\xi_1 - \xi_2) + \theta_2(t)
\]
(5.30)

for
\[
\lambda = \pi - (\theta_c(t) + v_2'(a_2,t))
\]
(5.31)

\[
h_{3y1} = a_c \sin(\lambda)
\]
(5.32)

\[
h_{3x2} = a_c \cos(\lambda)
\]
(5.33)

\[
h_{3y} = h_{3y1} + v_2(a_2,t) = a_c \sin(\lambda) + v_2(a_2,t)
\]
(5.34)

\[
h_{3x1} = a_2 - h_{3x2} = a_2 - a_c \cos(\lambda)
\]
(5.35)

\[
\xi_1 = \tan^{-1} \left[ \frac{h_{3y}}{h_{3x1}} \right] = \tan^{-1} \left[ \frac{a_c \sin(\lambda) + v_2(a_2,t)}{a_2 - a_c \cos(\lambda)} \right]
\]
(5.36)

\[
\xi_2 = \tan^{-1} \left[ \frac{v_{h3}(a_{h3}(t))}{a_{h3}(t)} \right] \equiv \left[ \frac{v_{h3}(a_{h3}(t))}{a_{h3}(t)} \right]
\]
(5.37)

Substituting (5.31) through (5.37) into (5.29) and (5.30) will yield the proper expressions for \( a_{h3}(t) \) and \( \theta_{h3}(t) \). From (5.31) through (5.37), it is observed that:
Figure 5.5 Geometry of Link 2, Link C, and Link H3
\( a_{h3} = \{a_{h3}(t), q_{v_{h3}}(t)\} \)  \( (5.38) \)

\[
\theta_{h3} = \{\theta_2(t), q_{v_2}(t), \theta_3(t), q_{v_{h3}}(t)\}
\]  \( (5.39) \)

The general displacement vector for link \( h3 \) is:

\[
\vec{V}_{h3} = u_{h3}\hat{i}_{h3} + v_{h3}(u_{h3}; t)\hat{j}_{h3} + w_{h3}(u_{h3}; t)\hat{k}_{h3}
\]  \( (5.40) \)

where \( u_{h3} \) is the displacement of link \( h3 \) along the \( x_{h3} \)-axis.

\( v_{h3}(u_{h3}; t) \) is the deflection of link \( h3 \) in the \( y_{h3} \)-direction.

\( w_{h3}(u_{h3}; t) \) is the deflection of link \( h3 \) in the \( z_{h3} \)-direction.

From the assumed modes method (Appendix B) and the discussions of the generalized coordinates in section 2.2, \( v_{h3} \) and \( w_{h3} \) may be expressed as:

\[
v_{h3}(u_{h3}, t) = \gamma_{v_{h3}}(u_{h3}) q_{v_{h3}}(t)
\]  \( (5.41) \)

\[
w_{h3}(u_{h3}, t) = \gamma_{w_{h3}}(u_{h3}) q_{w_{h3}}(t)
\]  \( (5.42) \)

where \( \gamma_{v_{h3}}(u_{h3}) \) is the admissible function of link \( h3 \) in the \( y_{h3} \)-direction.

\( q_{v_{h3}}(t) \) is the time-dependent generalized coordinate of link \( h3 \) in the \( y_{h3} \)-direction.

\( \gamma_{w_{h3}}(u_{h3}) \) is the admissible function of link \( h3 \) in the \( z_{h3} \)-direction.

\( q_{w_{h3}}(t) \) is the time-dependent generalized coordinate of link \( h3 \) in the \( z_{h3} \)-direction.
Since link h3 is composed of a rigid portion (the cylinder shell) and a flexible portion (the piston rod), it is necessary to formulate two vectors to describe the two portions. For \( 0 \leq u_{h3} \leq a_{h3s} \):

\[
V_{h3r} = u_{h3} \mathbf{i}_{h3} + 0 \mathbf{j}_{h3} + 0 \mathbf{k}_{h3} = (u_{h3}, 0, 0)^T \tag{5.43}
\]

For \( a_{h3s} \leq u_{h3} \leq a_{h3}(t) \):

\[
V_{h3e} = u_{h3} \mathbf{i}_{h3} + v_{h3}(u_{h3}(t)) \mathbf{j}_{h3} + w_{h3}(u_{h3}(t)) \mathbf{k}_{h3} = \begin{bmatrix} u_{h3} \\ v_{h3}(u_{h3}(t)) \\ w_{h3}(u_{h3}(t)) \end{bmatrix} \tag{5.44}
\]

where

\( a_{h3s} \) is the length of the cylinder shell.

\( V_{h3i} \) is the displacement vector of the rigid portion of link h3.

\( V_{h3e} \) is the displacement vector of the flexible portion of link h3.

For the in-plane admissible function of the hydraulic cylinder rod, case 3 of Appendix C is used. And because of observation 7 stated in section 5.1.5, case 4 of Appendix C is used to define an expression for \( \gamma_{w_{h3}} \). From figure 5.4:

\[
w_{h3}'(a_{h3}(t), t) = w_2'(a_2, t) \tag{5.45}
\]

\[
=> \quad q_{w_{h3}}(t) = \frac{w_2'(a_2, t)}{\gamma_{w_{h3}}'(a_{h3}(t))} = \frac{\gamma_{w_2}'(a_2, t) q_{w_2}(t)}{\gamma_{w_{h3}}'(a_{h3}(t))} \tag{5.46}
\]
(5.46) shows that $q_{w_{h3}}(t)$ is in terms of $q_{w_2}(t)$. Therefore, there is one less time-dependent generalized coordinate in the system. From figure 5.5, $\Delta$ and $\Lambda$ of (C.95) and (C.96) may be expressed as:

$$\Delta = w_2'(a_2, t)$$  \hspace{1cm} (5.47)

$$\Lambda = w_2(a_2) - a_c \cos(\lambda) \sin(-w_2'(a_2, t))$$  \hspace{1cm} (5.48)

Substituting (5.47) and (5.48) into (C.95) and (C.96) will yield the proper out-of-plane admissible function for link $h_3$.

It is necessary to transform the displacement vectors of link $h_3$ into frame 2. Substituting the Denavit–Hartenberg parameters of table 5.1 into the orientation portion of (A.6) yields:

$$T_{h3 \rightarrow 2} = \begin{bmatrix} \cos(\eta_{h3}) & -\sin(\eta_{h3}) & 0 \\ \sin(\eta_{h3}) & \cos(\eta_{h3}) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$  \hspace{1cm} (5.49)

$$\eta_{h3} = \theta_{h3}(t) - \theta_2(t)$$  \hspace{1cm} (5.50)

where $T_{h3 \rightarrow 2}$ is the transformation matrix of frame $h_3$ to frame 2.

The displacement vectors of link $h_3$ become:

$$V_{h3 \rightarrow 2} = T_{h3 \rightarrow 2} \cdot (u_{h3} \hspace{1cm} 0 \hspace{1cm} 0)^T$$  \hspace{1cm} (5.51)
\[
\begin{align*}
\overline{V}_{h3e\rightarrow2} &= T_{h3\rightarrow2} \cdot \begin{bmatrix}
u_{h3} \\ v_{h3}(u_{h3},t) \\ v_{h3}(u_{h3},t)
\end{bmatrix} \\
(5.52)
\end{align*}
\]

where \(\overline{V}_{h3r\rightarrow2}\) is the displacement vector of the rigid portion of link \(h3\) in frame 2.

\(\overline{V}_{h3e\rightarrow2}\) is the displacement vector of the flexible portion of link \(h3\) in frame 2.

### 5.1.5.2 Velocity Vectors of Link H3

The angular velocity of link \(h3\) is:

\[
\overline{\Omega}_{h3\rightarrow2} = \overline{\Omega} + T_{h3\rightarrow2} \cdot (0 \ 0 \ \dot{\theta}_{h3})^T \\
(5.53)
\]

The velocity vectors of link \(h3\) have the following form:

\[
\begin{align*}
\overline{\dot{V}}_{h3r\rightarrow2} &= \overline{\dot{V}}_{1\rightarrow2} + \left[ \overline{\Omega}_{h3\rightarrow2} \times \overline{V}_{h3r\rightarrow2} \right] \\
(5.54)
\end{align*}
\]

\[
\begin{align*}
\overline{\dot{V}}_{h3e\rightarrow2} &= \overline{\dot{V}}_{1\rightarrow2} + \left[ \overline{\Omega}_{h3\rightarrow2} \times \overline{V}_{h3e\rightarrow2} \right] \\
&+ \dot{u}_{h3} \overline{p}_{u_{h3}} + \dot{v}_{h3}(u_{h3},t) \overline{p}_{v_{h3}} + \dot{w}_{h3}(u_{h3},t) \overline{p}_{w_{h3}} \\
&+ \ddot{u}_{h3} \overline{p}_{u_{h3}} + \ddot{v}_{h3}(u_{h3},t) \overline{p}_{v_{h3}} + \ddot{w}_{h3}(u_{h3},t) \overline{p}_{w_{h3}} \\
(5.55)
\end{align*}
\]

\[
\begin{align*}
\overline{p}_{u_{h3}} &= T_{h3\rightarrow2} \cdot (1 \ 0 \ 0)^T \\
(5.56)
\end{align*}
\]

\[
\begin{align*}
\overline{p}_{v_{h3}} &= T_{h3\rightarrow2} \cdot (0 \ 1 \ 0)^T \\
(5.57)
\end{align*}
\]
\[ \bar{p}_{w_{h3}} = T_{h3 \rightarrow 2} \cdot (0 \quad 0 \quad 1)^T \] (5.58)

where
\( \bar{p}_{u_{h3}} \) is the projection vector onto the \( x_2 \)-axis.
\( \bar{p}_{v_{h3}} \) is the projection vector onto the \( y_2 \)-axis.
\( \bar{p}_{w_{h3}} \) is the projection vector onto the \( z_2 \)-axis.

5.2 Dynamic Analysis of the UNLV-ARO Light-weight Robot

5.2.1 Kinetic And Potential Energy Equations of Link 1

Since there is no change in the configuration of link 1, the kinetic and potential energy equations remain the same as discussed in chapter 4.

5.2.2 Kinetic And Potential Energy Equations of Link 2

The kinetic energy equation of link 2 is:

\[
KE_2 = \frac{1}{2} \rho_{2r} \int_{0}^{a_{2r}} \left[ \dot{V}_{2r} \cdot \dot{V}_{2r} \right] du_2 + \frac{1}{2} \rho_{2e} \int_{a_{2r}}^{a_{2}} \left[ \dot{V}_{2e} \cdot \dot{V}_{2e} \right] du_2 + \frac{1}{2} \text{sm}_2 \left[ \dot{V}_{2e}(a_2) \cdot \dot{V}_{2e}(a_2) \right] \quad (5.59)
\]

\[
\text{sm}_2 = \text{lm}_2 + \rho_3 a_3 + \text{lm}_3 + \rho_c a_c \quad (5.60)
\]

The potential energy equation of link 2 is:
5.2.3 Kinetic And Potential Energy Equations of Link 3

The kinetic energy equation of link 3 is:

\[
KE_3 = \frac{1}{2} \rho_3 \int_0^{a_3} \left[ \ddot{V}_{3+2} \cdot \dot{V}_{3+2} \right] du_3 + \frac{1}{2} \left[ \ddot{V}_{3+2}(a_3) \cdot \dot{V}_{3+2}(a_3) \right] \quad (5.62)
\]

The potential energy equation of link 3 is:

\[
PE_3 = -E_3 I_v \int_0^{a_3} \left[ \ddot{v}_{3+2} \cdot (u_{3,t}) \right]^2 du_3 + \frac{1}{2} E_3 I_w \int_0^{a_3} \left[ \ddot{w}_{3+2} \cdot (u_{3,t}) \right]^2 du_3 - \rho_3 \int_0^{a_3} \left[ G_{1+2} \cdot \dot{V}_{3+2} \right] du_3 - \sm_3 \left[ G_{1+2} \cdot \dot{V}_{3+2}(a_3) \right] \quad (5.64)
\]
5.2.4 Kinetic And Potential Energy Equations of Link C

The kinetic energy equation of link c is:

\[
KE_c = \frac{1}{2} \rho_c \int_0^{a_c} \left[ \frac{1}{V_{c+2}} \cdot \frac{1}{V_{c+2}} \right] du_c
\]  

(5.65)

The potential energy equation of link c is:

\[
PE_c = -\rho_c \int_0^{a_c} \left[ G_{1+2} \cdot \frac{1}{V_{c+2}} \right] du_c
\]  

(5.66)

5.2.5 Kinetic And Potential Energy Equations of Link H3

The kinetic energy equation of link h3 is:

\[
KE_{h3} = \frac{1}{2} \rho_{h3s} \int_0^{a_{h3s}} \left[ \frac{1}{V_{h3r+2}} \cdot \frac{1}{V_{h3r+2}} \right] du_{h3s}
\] 

\[
+ \frac{1}{2} \rho_{h3p} \int_{a_{h3pend}}^{a_{h3pstart}} \left[ \frac{1}{V_{h3r+2}} \cdot \frac{1}{V_{h3r+2}} \right] du_{h3p}
\] 

\[
+ \frac{1}{2} \rho_{h3r} \int_{a_{h3pend}}^{a_{h3s}} \left[ \frac{1}{V_{h3r+2}} \cdot \frac{1}{V_{h3r+2}} \right] du_{h3r}
\] 

\[
+ \frac{1}{2} \rho_{h3r} \int_{a_{h3s}}^{a_{h3e}} \left[ \frac{1}{V_{h3e+2}} \cdot \frac{1}{V_{h3e+2}} \right] du_{h3}
\]  

(5.67)
where $\rho_{h3s}$ is the mass per unit length of the hydraulic cylinder shell.

$\rho_{h3p}$ is the mass per unit length of the piston.

$\rho_{h3r}$ is the mass per unit length of the piston rod.

$a_{h3p\text{start}}$ marks the beginning location of the piston.

$a_{h3p\text{end}}$ marks the ending location of the piston.

$a_{h3p\text{start}}$ and $a_{h3p\text{end}}$ are determined by the following formulation.

$$a_{h3p\text{end}} = a_{h3}(t) - a_{h3r}$$ (5.68)

$$a_{h3p\text{start}} = a_{h3}(t) - a_{h3r} - a_{h3p}$$ (5.69)

where $a_{h3r}$ is the length of the piston rod.

$a_{h3p}$ is the length of the piston.

The potential energy equation of link $h3$ is:

$$PE_{h3} = \frac{1}{2} E_{h3} I_{h3} \int_{a_{h3s}}^{a_{h3p}} \left[ v_{h3\to2} \cdot (u_{h3}, t) \right]^2 + \left[ w_{h3\to2} \cdot (u_{h3}, t) \right]^2 \, du_{h3}$$

$$- \rho_{h3s} \int_{0}^{a_{h3s}} \left[ - G_{1\to2} \cdot \overrightarrow{V_{h3r\to2}} \right] \, du_{h3}$$

$$- \rho_{h3p} \int_{a_{h3p\text{start}}}^{a_{h3p\text{end}}} \left[ - G_{1\to2} \cdot \overrightarrow{V_{h3r\to2}} \right] \, du_{h3}$$

$$- \rho_{h3r} \int_{a_{h3p\text{end}}}^{a_{h3s}} \left[ - G_{1\to2} \cdot \overrightarrow{V_{h3r\to2}} \right] \, du_{h3}$$

$$- \rho_{h3r} \int_{a_{h3s}}^{a_{h3p}} \left[ - G_{1\to2} \cdot \overrightarrow{V_{h3e\to2}} \right] \, du_{h3}$$ (5.70)
where $E_{h3}$ is the modulus of elasticity of link $h3$.

$I_{h3}$ is the area moment of inertia of link $h3$ along the $x_{h3}$-axis.

### 5.2.6 Lagrangian Equations

From the kinematic of this system, there are eight time-dependent generalized coordinates. They are $q_{v2}(t)$, $q_{w2}(t)$, $q_{v3}(t)$, $q_{w3}(t)$, $q_{v_{h3}}(t)$, $\theta_1(t)$, $\theta_2(t)$, and $\theta_3(t)$. Substituting these coordinate into (2.12) yields:

\[
\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\theta}_1} \right) - \frac{\partial L}{\partial \theta_1(t)} = \text{Torque}_1 \tag{5.71}
\]

\[
\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\theta}_2} \right) - \frac{\partial L}{\partial \theta_2(t)} = \text{Torque}_2 \tag{5.72}
\]

\[
\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\theta}_3} \right) - \frac{\partial L}{\partial \theta_3(t)} = \text{Torque}_3 \tag{5.73}
\]

\[
\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_{v2}} \right) - \frac{\partial L}{\partial q_{v2}(t)} = 0 \tag{5.74}
\]

\[
\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_{w2}} \right) - \frac{\partial L}{\partial q_{w2}(t)} = 0 \tag{5.75}
\]

\[
\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_{v3}} \right) - \frac{\partial L}{\partial q_{v3}(t)} = 0 \tag{5.76}
\]
\[
\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_{w_3}} \right) - \frac{\partial L}{\partial q_{w_3}(t)} = 0 \tag{5.77}
\]

\[
\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_{v_{h3}}} \right) - \frac{\partial L}{\partial q_{v_{h3}}(t)} = 0 \tag{5.78}
\]

\[
L = KE_1 + KE_2 + KE_3 + KE_c + KE_{h3} - PE_1 - PE_2 - PE_3 - PE_c - PE_{h3} \tag{5.79}
\]

Substituting the kinetic and potential energy equations into (5.79) yields the Lagrange operator of this robot. Applying the Lagrange operator into (5.71) through (5.78) yields the equations of motion of the system.

5.3 Initial Conditions

From section 2.5.1, the initial values of \( \theta_1(0) \), \( \dot{\theta}_1(0) \), \( \ddot{\theta}_1(0) \), \( \theta_2(0) \), \( \dot{\theta}_2(0) \), \( \ddot{\theta}_2(0) \), \( \theta_3(0) \), \( \dot{\theta}_3(0) \), and \( \ddot{\theta}_3(0) \) are pre-determined. The initial values of \( q_{v_2}(0) \), \( q_{w_2}(0) \), \( q_{v_3}(0) \), \( q_{w_3}(0) \), and \( q_{v_{h3}}(0) \) may be obtained from the initial deflections of the links. However, the initial deflection of link \( h_3 \) may not be obtained using standard static deflection equations, therefore, it is necessary to determinate the reactions at the end link \( h_3 \). From figure 5.6a and 5.6b:

\[
\sum F_{ABx} = R_{Ax} + R_{Bx} = 0 \tag{5.80}
\]

\[
\sum F_{ABy} = R_{Ay} + R_{By} - R_{h3s} - R_{h3p} - R_{h3r} = 0 \tag{5.81}
\]
Figure 5.6a Free-body Diagram of the Reactions of Link H3

Figure 5.6b Free-body Diagram of the Reactions at the End of Link 2
\[ \sum M_A = -R_{Bx} \, a_{h3} \sin(\theta_{h3}) + R_{By} \, a_{h3} \cos(\theta_{h3}) \]
\[ -R_{h3s} \, \frac{a_{h3}}{2} \cos(\theta_{h3}) - R_{h3p} \left( a_{h3} - a_{h3r} - \frac{a_{h3p}}{2} \right) \cos(\theta_{h3}) \]
\[ -R_{h3r} \left( a_{h3} - \frac{a_{h3r}}{2} \right) \cos(\theta_{h3}) = 0 \]  
(5.82)

\[ \sum F_{BDx} = -R_{Bx} + R_{Dx} = 0 \]  
(5.83)

\[ \sum F_{BDy} = -R_{By} + R_{Dy} - R_{lm_2} - R_c - R_3 - R_{lm_3} = 0 \]  
(5.84)

\[ \sum M_D = R_{Bx} \, a_c \sin(\pi - (\theta_2 + \theta_3 + \xi)) + R_{By} \, a_c \cos(\pi - (\theta_2 + \theta_3 + \xi)) \]
\[ -R_3 \, \frac{a_3}{2} \cos(\theta_2 + \theta_3) - R_{lm_3} \, a_3 \cos(\theta_2 + \theta_3) \]
\[ + R_c \, \frac{a_c}{2} \cos(\pi - (\theta_2 + \theta_3 + \xi)) = 0 \]  
(5.85)

for
\[ R_{lm_2} = lm_2 \, g \]  
(5.86)
\[ R_3 = \rho_3 \, a_3 \, g \]  
(5.87)
\[ R_{lm_3} = lm_3 \, g \]  
(5.88)
\[ R_c = \rho_c \, a_c \, g \]  
(5.89)
\[ R_{h3s} = \rho_{h3s} \, a_{h3s} \, g \]  
(5.90)
\[ R_{h3p} = \rho_{h3p} \, a_{h3p} \, g \]  
(5.91)
\[ R_{h3r} = \rho_{h3r} \, a_{h3r} \, g \]  
(5.92)

where
\[ F_{ABx} \] is the force in the x—direction of figure 5.6a.
\[ F_{ABy} \] is the force in the y—direction of figure 5.6a.
\[ F_{BDx} \] is the force in the x—direction of figure 5.6b.
$F_{BDy}$ is the force in the $y$-direction of figure 5.6b.

$M_A$ is the moment about point A.

$M_D$ is the moment about point D.

$R_{Ax}$ is the reaction at point A in the $x$-direction.

$R_{Ay}$ is the reaction at point A in the $y$-direction.

$R_{Bx}$ is the reaction at point B in the $x$-direction.

$R_{By}$ is the reaction at point B in the $y$-direction.

$R_{Dx}$ is the reaction at point D in the $x$-direction.

$R_{Dy}$ is the reaction at point D in the $y$-direction.

$R_{1m_2}$ is the reaction of lumped mass 2.

$R_3$ is the reaction of link 3.

$R_{1m_3}$ is the reaction of lumped mass 3.

$R_c$ is the reaction of link c.

$R_{h3s}$ is the reaction of the hydraulic cylinder shell.

$R_{h3p}$ is the reaction of the piston.

$R_{h3r}$ is the reaction of the piston rod.

Rearrange (5.80) through (5.85) into matrix form yield:

\[
\begin{bmatrix}
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & -A_{3,3} & A_{3,4} & 0 & 0 \\
0 & 0 & -1 & 0 & 1 & 0 \\
0 & 0 & 0 & -1 & 0 & 1 \\
0 & 0 & A_{6,3} & A_{6,4} & 0 & 0
\end{bmatrix}
\begin{bmatrix}
R_{Ax} \\
R_{Ay} \\
R_{Bx} \\
R_{By} \\
R_{Dx} \\
R_{Dy}
\end{bmatrix}
= 
\begin{bmatrix}
B_1 \\
B_2 \\
B_3 \\
B_4 \\
B_5 \\
B_6
\end{bmatrix}
\]  

(5.93)

for

\[
A_{3,3} = a_{h3} \sin(\theta_{h3})
\]  

(5.94)
\[ A_{3,4} = a_{h3} \cos(\theta_{h3}) \] (5.95)

\[ A_{6,3} = a_c \sin(\pi - (\theta_2 + \theta_3 + \xi)) \] (5.96)

\[ A_{6,4} = a_c \cos(\pi - (\theta_2 + \theta_3 + \xi)) \] (5.97)

\[ B_1 = 0 \] (5.98)

\[ B_2 = R_{h3s} + R_{h3p} + R_{h3r} \] (5.99)

\[ B_3 = R_{h3s} \frac{a_{h3}}{2} \cos(\theta_{h3}) + R_{h3p} (a_{h3} - a_{h3r} - \frac{a_{h3p}}{2}) \cos(\theta_{h3}) \]

\[ + R_{h3r} (a_{h3} - \frac{a_{h3r}}{2}) \cos(\theta_{h3}) \] (5.100)

\[ B_4 = 0 \] (5.101)

\[ B_5 = R_{lm_2} + R_c + R_3 + R_{lm_3} \] (5.102)

\[ B_6 = R_3 \frac{a_3}{2} \cos(\theta_2 + \theta_3) + R_{lm_3} a_3 \cos(\theta_2 + \theta_3) \]

\[ - R_c \frac{a_c}{2} \cos(\pi - (\theta_2 + \theta_3 + \xi)) \] (5.103)

Solving the simultaneous equations of (5.93) yields the values for \( R_{Bx} \), \( R_{By} \), \( R_{Dx} \), and \( R_{Dy} \). Using these values, the initial deflection of link 2 and link h3 may be obtained by using the standard static deflection equations.

\[ v_2(a_2,0) = \left[ -\frac{\rho_{2e} g a_{2e}}{8 E_2 I_{v2}} \right] \cos(\theta_2(0)) + \left[ -\frac{R_D a_{2e}}{3 E_2 I_{v2}} \right] \] (5.104)
\[ v_{h3}(a_{h3},0) \neq \left[ -\frac{\rho_{h3r} a_{h3e}}{8 E_{h3} I_{h3}} \right]^4 \cos(\theta_{h3}(0)) + \left[ \frac{R_B a_{h3e}}{3 E_{h3} I_{h3}} \right]^3 \]  

for 
\[ R_B = -R_{Bx} \sin(\theta_{h3}(0)) + R_{By} \cos(\theta_{h3}(0)) \]  
\[ R_D = -R_{Dx} \sin(\theta_2(0)) + R_{Dy} \cos(\theta_2(0)) \]  
\[ a_{2e} = a_2 - a_{2r} \]  
\[ a_{h3e} = a_{h3} - a_{h3s} \]  

From the initial in-plane deflection of link 2 and link h3, the initial value of \( q_{v2}(0) \) and \( q_{v h3}(0) \) may be obtained by equating:

\[ q_{v2}(0) = \frac{v_{v2}(a_{2},0)}{v_{v2}(a_{2})} \]  
\[ q_{v h3}(0) = \frac{v_{h3}(a_{h3},0)}{v_{h3}(a_{h3})} \]  

The initial value for \( q_{v3}(0) \) remains the same as in chapter 4. Since there is no initial out-of-plane deflection, the initial values for \( q_{w2}(0) \) and \( q_{w3}(0) \) are zero, or

\[ q_{w2}(0) = q_{w3}(0) = 0 \]  

Initially, all the links of the robot are stationary, therefore:

\[ \dot{q}_{v2}(0) = \dot{q}_{w2}(0) = \dot{q}_{v3}(0) = \dot{q}_{w3}(0) = \dot{q}_{v h3}(0) = 0 \]
5.4 Analysis of the Force in the Hydraulic Cylinders

Since all the links of the UNLV—ARO robot are actuated by hydraulic cylinders, it is more appropriate to present the results in terms of piston forces. The following discuss the conversion from joint torques into piston forces.

5.4.1 Piston Force Experienced by Hydraulic Cylinder 1

Since hydraulic cylinder 1 moves link 1 through a rack and pinion arrangement, therefore, they maintain the same normal distance throughout the motion of the robot. From this observation:

\[
\text{Force}_1 = \frac{\text{Torque}_{1}}{l_{h1}} \quad (5.114)
\]

where \( \text{Force}_1 \) is the force experienced by the piston of hydraulic cylinder 1.

\( l_{h1} \) is the normal distance between the center of link 1 and the center of the piston.

5.4.2 Piston Force Experienced by Hydraulic Cylinder 2

Figure 5.7a displays the free-body diagram of the relationship between the torque at joint 2 and the force on the hydraulic cylinder 2. From this figure:

\[
P_2 = \frac{\text{Torque}_{2}}{l_{h2b}} \quad (5.115)
\]
Figure 5.7a Free–body Diagram of the Relationship of Torque 2 and Force 2

Figure 5.7b Free–body Diagram of the Relationship of Torque 3 and Force 3
\[ \text{Force}_2 = \frac{P_2}{\cos\left(\frac{\pi}{2} - \nu_2\right)} \] (5.116)
since
\[ \nu_3 = \cos \left\{ \frac{a_h^2 + a_c^2 - a_2^2}{2 a_h a_c} \right\} \]

therefore

\[ \text{Force}_3 = \frac{\text{Torque}_3}{a_c} \left[ \frac{1}{\cos(\nu_3 - \frac{\pi}{2})} \right] \]  

where Force\textsubscript{3} is the force experience by the piston of hydraulic cylinder 3.

5.5 MACSYMA Programs

Similar to the discussions in section 4.4, the content of several program modules are slightly modified to comprise with the configuration of the UNLV—ARO robot. New program modules are also created for link c and link h3. Figure 5.8a and 5.8b display the flow charts of the MACSYMA program for this case study. The listing of these MACSYMA programs are included in Appendix F. Brief description of the new program modules are discussed in section 5.5.1. The symbolic derivation of the equations of motion for this case study is briefly explained in section 5.5.2.
Figure 5.8a Flow Chart of the MACSYMA Program Modules
Figure 5.8b Flow Chart of the MACSYMA Program Modules
5.5.1 MACSYMA Program Modules for the Derivation of the Equations of Motion for the UNLV--ARO Light--weight Robot

The following gives a brief description of the new program modules. Other modules are listed in section 3.4.1. and 4.4.1.

**Aro.bat**
This is the UNIX executable file that invokes the main derivation program modules of the UNLV--ARO light--weight robot.

**Linkc.com**
This is the main module for link c. Other modules are called from this main program to setup the equations of motion of link c.

**Linkh3.com**
This is the main module for link h3. Other modules are called from this main program to setup the equations of motion of link h3.

**Varsetupc.com**
This module helps to setup the displacement and velocity vectors of link c.

**Varsetuph3.com**
This module helps to setup the displacement and velocity vectors of link h3.

**Energyc.com**
This module helps to setup the energy equations and the Lagrange operator of link c.

**Energyh3.com**
This module helps to setup the energy equations and the Lagrange operator of link h3.

**Coeffc.com**
This module calls other modules to manipulate the equations of motion of link c in order to obtain the coefficients of the equations.
This module calls other modules to manipulate the equations of motion of link h3 in order to obtain the coefficients of the equations.

This module along with module `subsimpc.com` perform simple expression substitutions for the equations of link c. The expression substituted are the time derivatives of the generalized coordinates.

This module along with module `subsimph3.com` perform simple expression substitutions for the equations of link h3. The expression substituted are the time derivatives of the generalized coordinates.

This module is called by `simplifyc.com`. It performs simple expression substitutions for the current equation.

This module is called by `simplifyh3.com`. It performs simple expression substitutions for the current equation.

This module helps to create output listings of link c and saves the results into data file.

This module helps to create output listings of link h3 and saves the results into data file.

This module helps to substitute repeated expressions of link c into variable names. It also translates the equations into FORTRAN statements.

This module helps to substitute repeated expressions of link h3 into variable names. It also translates the equations into FORTRAN statements.
This module is called by reducec.com. It helps to substitute repeated expressions into variable names for the current equation.

This module is called by reduceh3.com. It helps to substitute repeated expressions into variable names for the current equation.

This is the main program module that helps to define the variables of link h3.

This module helps to set up the variables of link h3.

This module along with h3subsimp.com simplifies the variables of link h3.

This module is called by h3simp.com to simplify the current equation.

This module creates a listing of FORTRAN codes for the variables of link h3.

5.5.2 Symbolic Derivation

The symbolic derivation of the equations of motion of the UNLV—ARO robot is different from the preceding case of chapter 4. One difference is the inclusion of the out-of-plane motions of the links. The derivation of the kinematics and dynamics of these motions are included in the variable setup program modules. The other difference is the existence of the third hydraulic cylinder. In order to derive the equations, mathematical expressions for $a_{h3}$ and $\theta_{h3}$ are needed. The expressions for these variables are quite long. The derivatives of these two variables are even more complicated. If these expressions are substituted into the kinetic and potential energy equations, the resulting equations of motion will be enormous.
Therefore, in order to reduce the complexity of the equations of motion for this link, the MACSYMA’s \texttt{depends()} function is used. \texttt{depends()} defines variables in terms of other variables. Using this feature, $a_{h3}$ and $\theta_{h3}$ are defined without the lengthy expressions. The first derivatives of these two variables become:

\begin{equation}
\dot{a}_{h3} = q_2 \frac{\partial a_{h3}}{\partial q_2} + \dot{\theta}_3 \frac{\partial a_{h3}}{\partial \theta_3}
\end{equation}

\begin{equation}
\dot{\theta}_{h3} = \dot{\theta}_2 \left( \frac{\partial \theta_{h3}}{\partial \theta_2} \right) + q_2 \frac{\partial \theta_{h3}}{\partial q_2} + \dot{\theta}_3 \left( \frac{\partial \theta_{h3}}{\partial \theta_3} \right) + q_{vh3} \left( \frac{\partial \theta_{h3}}{\partial q_{vh3}} \right)
\end{equation}

Since $a_{h3}$ and $\theta_{h3}$ are implicitly defined in terms of other variables. The full expressions of $a_{h3}$, $\theta_{h3}$, and their derivatives are needed in FORTRAN programs. For this case study, several new program modules are created. These modules are \texttt{h3var.com}, \texttt{h3init.com}, \texttt{h3simp.com}, \texttt{h3subsimp.com}, and \texttt{h3list.com}. \texttt{H3var.com} is the main program module that controls the other modules to complete this task. \texttt{H3init.com} helps to set up the expressions for $a_{h3}$, $\theta_{h3}$, and their derivatives. After the variables are initiated, \texttt{h3simp.com} along with \texttt{h3subsimp.com} are called to reduce the expressions into simpler form by replacing repeated expressions into variables. Finally, \texttt{h3list.com} is called to translate the variables of link h3 into FORTRAN statements.

With the addition of the out-of-plane motion, link c, and link h3, (2.19) becomes:
\[
\begin{bmatrix}
D_{11} & D_{12} & D_{13} & D_{14} & D_{15} \\
D_{21} & D_{22} & D_{23} & D_{24} & D_{25} \\
D_{31} & D_{32} & D_{33} & D_{34} & D_{35} \\
D_{41} & D_{42} & D_{43} & D_{44} & D_{45} \\
D_{51} & D_{52} & D_{53} & D_{54} & D_{55}
\end{bmatrix}
\begin{bmatrix}
\ddot{q}_{v_2} \\
\ddot{q}_{w_2} \\
\ddot{q}_{v_3} \\
\ddot{q}_{w_2} \\
\ddot{q}_{v_{h3}}
\end{bmatrix}
+ \begin{bmatrix}
F_1 \\
F_2 \\
F_3 \\
F_4 \\
F_5
\end{bmatrix} = \begin{bmatrix} 0 
\end{bmatrix}
\] (5.126)

where

\[D_{11}, D_{21}, D_{31}, D_{41}, \text{ and } D_{51} \text{ are the coefficient of } \ddot{q}_{v_2} \text{ of } (5.74).\]

\[D_{12}, D_{22}, D_{32}, D_{42}, \text{ and } D_{52} \text{ are the coefficient of } \ddot{q}_{w_2} \text{ of } (5.75).\]

\[D_{13}, D_{23}, D_{33}, D_{43}, \text{ and } D_{53} \text{ are the coefficient of } \ddot{q}_{v_3} \text{ of } (5.76).\]

\[D_{14}, D_{24}, D_{34}, D_{44}, \text{ and } D_{54} \text{ are the coefficient of } \ddot{q}_{w_3} \text{ of } (5.77).\]

\[D_{15}, D_{25}, D_{35}, D_{45}, \text{ and } D_{55} \text{ are the coefficient of } \ddot{q}_{v_{h3}} \text{ of } (5.78).\]

In order to extract \(F_1, F_2, F_3, F_4, \text{ and } F_5 \) from (5.126), \(\ddot{q}_{v_2}, \ddot{q}_{w_2}, \ddot{q}_{v_3}, \ddot{q}_{w_3}, \) and \(\ddot{q}_{v_{h3}}\) are first set to zero, and then (5.126) is evaluated. The expressions for \(D_{11}, D_{12}, \ldots, D_{45}, \text{ and } D_{55}\) may be obtained by first setting \(g, E_2, E_3, E_{h3}, I_{v_2}, I_{w_2}, I_{v_3}, I_{w_3}, I_{h3}, q_{v_2}, q_{w_2}, q_{v_3}, q_{w_3}, q_{v_{h3}}, \theta_1, \bar{\theta}_1, \theta_2, \bar{\theta}_2, \theta_3, \) and \(\bar{\theta}_3\) to zero. For \(D_1 \) through \(D_{51}, \ddot{q}_{v_2}\) is set to one, \(\ddot{q}_{w_2}, \ddot{q}_{v_3}, \ddot{q}_{w_3}, \) and \(\ddot{q}_{v_{h3}}\) are set to zero and then evaluate (5.126). Similarly, \(D_{12} \) through \(D_{52}\) may be obtained by setting \(\ddot{q}_{w_2}\) to one, \(\ddot{q}_{v_2}, \ddot{q}_{v_3}, \ddot{q}_{w_3}, \) and \(\ddot{q}_{v_{h3}}\) to zero and then evaluate (5.126). \(D_{13} \) through \(D_{53}, D_{14} \) through \(D_{54}, \) and \(D_{15} \) through \(D_{55} \) may be obtained by following the same pattern.
5.6 Illustrative Example

In this case study, only one set of data is taken. The system specifications of the robot are listed in table 5.2. The input profiles of each joint are the same as in chapter 4 (figure 4.3 through figure 4.11). The in–plane and out–of–plane deflections of link 2 are shown in figure 5.9 and figure 5.10 respectively. The deflections of link 3 are shown in figure 5.11 and figure 5.12. The deflections of link h3 are shown in figure 5.13 through figure 5.15. The torques at joints 1, 2 and 3 are shown in figure 5.16 through figure 5.18 respectively. The forces at the hydraulic cylinder pistons are shown in figure 5.19 through figure 5.21. The end–effector trajectories are shown in figure 5.22 (Bang–bang profile) and figure 5.23 (Polynomial profile). Finally, the end–effector errors are shown in figure 5.24 through figure 5.26.
****** System Parameters ******

Units are in SI unit: kg, meter and second

**Link 1 Information**

Length (m) = 0.64770
Inside diameter (m) = 0.17780
Outside diameter (m) = 0.22860
Cross section area (m**2) = 0.16215e-01
Density (kg/m**3) = 7860.00000
Mass per unit length (kg/m) = 127.44706
Mass (kg) = 82.54746
Mass moment of inertia (kg*m**2) = 0.21303e+00

**Link 2 Information**

Total length of link 2 (m) = 1.68275

For the rigid portion of link 2

Length (m) = 0.43180
Cross sectional dimension
  i) Inside width x height (m) = 0.02540 X 0.02540
  ii) Outside width x height (m) = 0.06350 X 0.06350
Cross section area (m**2) = 0.33871e-02
Density (kg/m**3) = 7860.00000
Mass per unit length (kg/m) = 26.62253
Mass (kg) = 11.49561

For the Flexible portion of Link 2

Modulus of elasticity (Pa) = 0.20000e+12
Length (m) = 1.25095
Cross sectional dimension
  i) Inside width x height (m) = 0.02540 X 0.02540
  ii) Outside width x height (m) = 0.03810 X 0.03810
Cross section area (m**2) = 0.80645e-03
Density (kg/m**3) = 7860.00000
Mass per unit length (kg/m) = 6.33870
Mass (kg) = 7.92939
In-plane area moment of inertia (m**4) = 0.14091e-06
Out-of-plane area moment of inertia (m**4) = 0.14091e-06
Lump mass at the end of link 2 (kg) = 1.00000

Table 5.2 System Specifications
Link 3 Information

Modulus of elasticity (Pa) = 0.20000e+12
Length (m) = 0.93345
Cross sectional dimension
  i) Inside width x height (m) = 0.02540 X 0.02540
  ii) Outside width x height (m) = 0.03810 X 0.03810
Cross section area (m**2) = 0.80645e-03
Density (kg/m**3) = 7860.0000
Mass per unit length (kg/m) = 6.33870
Mass (kg) = 5.91686
In-plane area moment of inertia (m**4) = 0.14091e-06
Out-of-plane area moment of inertia (m**4) = 0.14091e-06
Lump mass at the end of link 3 (kg) = 25.0000

Connecting Link (Link C) Information

Length (m) = 0.33020
Cross sectional dimension
  i) width x height (m) = 0.02540 X 0.05080
Cross section area (m**2) = 0.12903e-02
Density (kg/m**3) = 7860.0000
Mass per unit length (kg/m) = 10.14191
Mass (kg) = 3.34886
Zeta. Fix angle between link c and link 3 (rad) = 2.44346
Zeta. Fix angle between link c and link 3 (deg) = 140.0000

Hydraulic Cylinder 3 (Link H3) Information

For the Cylinder Shell of link H3

Length (m) = 0.91440
Insider diameter (m) = 0.07489
Outsider diameter (m) = 0.08489
Cross section area (m**2) = 0.12549e-02
Density (kg/m**3) = 7860.0000
Mass per unit length (kg/m) = 9.86358
Mass (kg) = 9.01926

For the Piston of Link H3

Length (m) = 0.05000
Insider diameter (m) = 0.00000
Outsider diameter (m) = 0.07000
Cross section area (m**2) = 0.38485e-02
Density (kg/m**3) = 7860.0000
Mass per unit length (kg/m) = 30.24883
Mass (kg) = 1.51244

Table 5.2 System Specifications Continues
For the Flexible Piston Rod of Link H3

Length (m) = 1.15000
Insider diameter (m) = 0.00000
Outsider diameter (m) = 0.03234
Cross section area (m**2) = 0.82143e-03
Density (kg/m**3) = 7860.00000
Mass per unit length (kg/m) = 6.45643
Mass (kg) = 7.42490
Area moment of inertia (m**4) = 0.53695e-07

General Information

Distance from the center of link 1 to the center of the piston of hydraulic cylinder 1 (m) = 0.13335
Distance from joint 2 to the joint located at the bottom of hydraulic cylinder 2 (m) = 0.53340
Distance from joint 2 to the joint located at the top of hydraulic cylinder 2 (m) = 0.23495
Initial position of link 1 (rad) = -1.81514
Initial position of link 1 (deg) = -104.00000
Final position of link 1 (rad) = 1.81514
Final position of link 1 (deg) = 104.00000
Initial position of link 2 (rad) = -0.39095
Initial position of link 2 (deg) = -22.40000
Final position of link 2 (rad) = 0.45204
Final position of link 2 (deg) = 25.90000
Initial position of link 3 (rad) = -1.60745
Initial position of link 3 (deg) = -92.10000
Final position of link 3 (rad) = 0.13788
Final position of link 3 (deg) = 7.90000
Duration of the experiment (s) = 2.00000
Duration of motion (s) = 1.00000
Number of steps within the experiment time frame = 1000.00000
Sampling interval (s) = 0.00200
The initial value of the step size H for IMSL = 0.20000e-04
The maximum steps for IMSL = 0.10000e+06
The tolerance (TOL) for IMSL = 0.10000e-04

Table 5.2 System Specifications Continues
Figure 5.9  In–plane Deflections at the End of Link 2 Due to the Bang–bang and the Polynomial Acceleration Profiles
Figure 5.10 Out—of—plane Deflections at the End of Link 2 Due to the Bang—bang and the Polynomial Acceleration Profiles
Figure 5.11 In-plane End-effector Deflections Due to the Bang-bang and the Polynomial Acceleration Profiles
Figure 5.12 Out-of-plane End-effector Deflections Due to the Bang-bang and the Polynomial Acceleration Profiles
Figure 5.13 In-plane Deflections of Link H3 Due to the Bang-bang Acceleration Profiles
Figure 5.14 In-plane Deflections of Link H3 Due to the Polynomial Acceleration Profiles
Figure 5.15 Out-of-plane Deflections of Link H3 Due to the Bang-bang and the Polynomial Acceleration Profiles
Figure 5.16 In-plane Deflections of Link 2 and Link 3.

(Deflections Generated by the Bang-bang Acceleration Profile)
Figure 5.17 Out-of-plane Deflections of Link 2 and Link 3.
(Deflections Generated by the Bang-bang Acceleration Profile)
Figure 5.18 Out-of-plane Deflections of Link 2 and Link H3.
(Deflections Generated by the Bang-bang Acceleration Profile)
Figure 5.19 Torques at Joint 1 Due to the Bang—bang and Polynomial Acceleration Profiles
Figure 5.20 Torques at Joint 2 Due to the Bang-bang and Polynomial Acceleration Profiles
<table>
<thead>
<tr>
<th>Torques (KNm)</th>
<th>Bang-Bang</th>
<th>Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>156</td>
<td>1.5</td>
<td>0.5</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>1.5</td>
</tr>
<tr>
<td>3.5</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>2.5</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 5.21 Torques at Joint 3 Due to the Bang-bang and Polynomial Acceleration Profiles
Figure 5.22 Forces at Joint 1 Due to the Bang-bang and Polynomial Acceleration Profiles
Figure 5.23 Forces at Joint 2 Due to the Bang—bang and Polynomial Acceleration Profiles
Figure 5.24 Forces at Joint 3 Due to the Bang-bang and Polynomial Acceleration Profiles
Figure 5.25 Trajectories of the End-effector Using the Bang-bang Acceleration Profiles with the Trajectory Projections on the X-Y, Y-Z, and Z-X Planes
Figure 5.26  Steady-state Trajectory of the End-effector (Flexible) Using the Bang-bang Acceleration Profiles with the Trajectory Projections on the X-Y, Y-Z, and Z-X Planes
Figure 5.27 Trajectories of the End-effector Using the Polynomial Acceleration Profiles with the Trajectory Projections on the X-Y, Y-Z, and Z-X Planes
Figure 5.28 Steady-state Trajectory of the End-effector (Flexible) Using the Polynomial Acceleration Profiles with the Trajectory Projections on the X–Y, Y–Z, and Z–X Planes.
Figure 5.29 Errors on the X-axis
Figure 5.30 Errors on the Y-axis
Figure 5.31 Errors on the Z-axis
Along with the evaluations discussed in previous chapters, the following new observations were noted for this case study.

1) From figure 5.10, the out-of-plane steady state deflections of link 2 are in the range of ± 0.06 meters. It seems quite high at first glance. However, for this set of data, the robot arms have to rotate 204° about the z_1-axis in just one second, the acceleration is quite high, therefore, it is reasonable to have such a large steady state deflections.

2) The in-plane deflection frequency of the flexible piston rod (figure 5.13) is relatively high when compared with the other links. This is due to the small moment of inertia of this rod.

3) The in-plane deflections of the flexible piston rod should be diminishing during the motion time frame of the robot. As the hydraulic cylinder retracted, the length of the flexible piston rod decreased. With a shorter rod, the magnitude of the deflection should be reduced. However, figure 5.13 did not presented such phenomena. This was because the length of the rod was not included as a boundary condition in the approximation of the mode shape of this link (Appendix C, case 3).

4) The out-of-plane deflection frequencies of link 3 also follow the out-of-plane deflection frequencies of link 2 (figure 5.17).

5) Since the out-of-plane deflections of link h3 is geometrically governed by the out-of-plane deflections of link 2, the deflections of these two links are very similar. In the beginning of the motion of the robot, link h3 has a larger deflection. This is because link h3 extends a bit further out than link 2. However, the out-of-plane deflections of link h3 diminishes as the hydraulic cylinder three retracts. This
observation may be shown by comparing the out-of-plane deflections of link 2 and the out-of-plane deflections of link h3 (figure 5.18).
CHAPTER SIX

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

The discussions in the previous chapters demonstrated a scheme to model the dynamics of the UNLV—ARO light—weight robot with flexible links. The robot was composed of a rigid trunk, two serially connected outer—most flexible links, and three hydraulic cylinders. The piston rod of the hydraulic cylinder which actuate the outer—most flexible link was assumed to be flexible as well. The modeling of this robot included the in—plane and the out—of—plane vibrations of the robot links. Lagrangian mechanics and the assumed mode method constituted the backbone of this research. The inverse dynamic method was used where the angular displacements, velocities, and accelerations of the joints were the input of the system. Because of the complexity of the problem, the symbolic software package, MACSYMA, was used to derive the equations of motion of this robot. The kinematics and the dynamics of the UNLV—ARO robot were included in MACSYMA program modules. The dynamics of this robot were solved by executing these program modules under MACSYMA environment. The result was a set of second order non—linear differential equations which described the motion of the robot links. From MACSYMA, this set of equations were translated into FORTRAN statements. In order to obtain analytical results, FORTRAN programs were written using these equations of motion, and subroutines from the numerical
solution package IMSL were used to solve the differential equations. Output data were then obtained by executing the FORTRAN programs.

Two simpler robot models were studied before the modeling of the UNLV—ARO robot. The first one (chapter three) was a two-link robot with the first link being rigid and the outer—most link being flexible. The motion of this robot is limited to the in—plane movement of the flexible link. Similarly, the second robot studied (chapter four) was a three-link robot but with the two outer—most links being flexible. The movements of the robot was also limited to the in—plane movement of the two outer—most flexible links. These two models provided the insight needed for the development of the modeling of the UNLV—ARO robot.

From the case studies discussed in the previous chapters, the following observations were noted.

1) The first case study (chapter three) presented the fundamental structure of the modeling scheme used in this thesis. From the output of this case study, it was found that:
   a) During the robot motion time frame (figure 3.6 and figure 3.8), the deflections produced by the polynomial acceleration profiles were larger than the Bang—bang acceleration profiles. This phenomena was due to the higher amount of acceleration of the polynomial profile.
   b) From the first example of this case study (figure 3.6), the magnitude of the steady—state deflections produced by the Bang—bang acceleration profile were about twice as much as the one produced by the polynomial acceleration profile. However, from the second example (figure 3.8) where only the dimension of the robot link was changed, the magnitude of the
steady-state deflections produced by the same two acceleration profiles were almost the same. From these observations, it was concluded that the steady-state deflections were very sensitive to the input acceleration profiles and the robot dimension.

c) The magnitude of the steady-state deflections were governed by the conditions at the point where the motion of the robot link ended. This was because at this point, it served as the input initial conditions for the steady-state deflections.

d) The curves of the torques (figure 3.7 and figure 3.9) and the curves of the deflections (figure 3.6 and figure 3.8) are mirror images along the zero axis. This was because a positive torque was needed to counter the negative deflection of the link and vice versa.

2) From the second case study (chapter four), the interaction of the vibrations between two serially connected links were presented. From the output of this case study, it was found that:

a) The deflection curves of this case study were not as smooth as those created in the previous case study. This was because the deflection frequencies of the flexible links were superimposed onto each other.

b) The deflection frequency of link 2 (the intermediate link) was lower than the deflection frequency of link 3 (the outer-most link). This was because link 2 experience a larger payload than link 2.

3) From the third and final case study, the solution for the UNLV-ARO robot was very sensitive to the mode shapes assumed for the robot links. The solution was especially sensitive to the mode shapes
assumed for the hydraulic cylinder piston rod. Numerous attempts of approximating the proper mode shapes had been tried before the mode shapes stated in Appendix C were found. From the output data, results similar to the previous case studies were obtained. In addition, it was observed that:

a) The in–plane deflection frequency of the flexible piston rod (figure 5.13) was relatively high when compared with the other links. This was due to the small moment of inertia of this rod.

b) The in–plane deflections of the flexible piston rod should be diminishing during the motion time frame of the robot. As the hydraulic cylinder retracted, the length of the flexible piston rod decreased. With a shorter rod, the magnitude of the deflection should be reduced. However, figure 5.13 did not present such phenomena. This was because the length of the rod was not included as a boundary condition in the approximation of the mode shape of this link (Appendix C, case 3).

c) The out–of–plane deflection frequencies of link 2 and link 3 were superimposed onto each other as in the case of the in–plane deflection frequencies.

d) The out–of–plane deflections of link 2 and the flexible piston rod shared similar deflection behavior. In the beginning of the robot motion, the flexible piston rod had a larger deflection. This was because the piston rod extended a bit further out than link 2. As the motion of the robot continued, the deflection of the piston rod diminished due to the retraction of the hydraulic cylinder. This phenomena obtained from the
output data agreed with the geometry that governed the out-of-plane deflections of link 2 and the flexible piston rod.

6.2 Recommendations

Several areas in this thesis may be enhanced by further research.

1) Experimental data are needed from actual flexible robot manipulators to verify the validity of the assumed mode shapes used in this research. This is needed especially for a more reasonable approximation of the mode shapes for the hydraulic cylinder piston rod. Also, a comparison of the experimental data with the data collect in this work will give a better understanding of how accurate this modeling scheme is.

2) The damping coefficients of the UNLV—ARO robot may be determined from the experimental data also. Thus, a complete modeling of the robot may be accomplished by including the torsional and the damping effects into this thesis.

3) The UNLV—ARO light—weight robot is a highly non—linear system. Therefore, another enhancement is to linearize the system using Taylor series expansion, so that comparison may be made between a full model and a linearized model.

4) From the results shown in the previous chapter, the magnitude of the steady—state deflections of a link was governed by the conditions where the motion of the link ended. This steady—state deflections of the link may be minimized by performing research on manipulating the governing conditions. This enhancement may be accomplished by
choosing an optimum input acceleration profile and by predicting the deflection at the end point of the robot link motion.

5) The symbolic software, MACSYMA, had played an importance role in this thesis. From the experience gained during the development of this work, it was becoming apparent that, a dedicated system program package may be created using the language of MACSYMA to fully integrate the development of any multi-link system. Therefore, a user only needs to specify the configuration of the system, and MACSYMA does the rest to produce the equations of motion for that system. Such software package may be constructed by modifying the MACSYMA program modules created for this thesis.
APPENDIX A

KINEMATIC REPRESENTATION OF ROBOT WITH SOME FLEXIBLE LINKS USING TRANSFORMATION MATRICES

The Denavit–Hartenberg representation (Craig, 1986) provides a scheme to describe the positions and orientations for the joints and links of rigid robot. It was developed by J. Denavit and R. S. Hartenberg in 1955. It is also known as the D–H representation in robotics. This scheme is based on the notion of Cartesian coordinate frame and the right hand rule. Figure A.1 depicts a simple joints and linkage of a robot. The steps of the scheme are as follow:

1) For a n–link robot, denote the joints from 1 to n.
2) The \( z_i \) axis is coincident with the joint i axis.
3) The \( x_i \) axis is the common normal of \( z_i \) and \( z_{i+1} \) axes and is pointing away from the \( z_i \) axis.
4) The \( y_i \) axis is chosen using the right hand rule with respect to the \( x_i \) axis and the \( z_i \) axis.
5) The angle between \( x_{i-1} \) and \( x_i \) axes measured about the \( z_i \) axis is denoted as \( \theta_i \). \( \theta_i \) is referred as the vector of the joint angle. \( \theta_i \) is positive according to the right hand rule about the \( z_i \) axis moving from \( x_{i-1} \) to \( x_i \).
Figure A.1 Simple Joints and Linkage of a Robotic System
6) The distance from \( x_{i-1} \) axis to \( x_i \) axis measured along the \( z_i \) axis is denoted as \( d_i \). This is the link distance between two links.

7) The distance from \( z_i \) axis to \( z_{i+1} \) axis measured along the \( x_i \) axis is denoted as \( a_i \). This is the link length of link \( i \).

8) The angle between \( z_i \) and \( z_{i+1} \) axes measured about the \( x_i \) axis is denoted as \( \alpha_i \). \( \alpha_i \) is the vector of the twist angle. \( \alpha_i \) is positive according to the right hand rule about the \( x_i \) axis moving from \( z_i \) to \( z_{i+1} \).

9) The origin of frame \( i \) is located where \( a_i \) perpendicular intersects the joint \( i \) axis.

A table of the Denavit–Hartenberg representation of a robot may be constructed by following the instructions stated above. Using these parameters, a transformation matrix from frame \( i \) to frame \( i-1 \) of a robot with rigid links may be obtained. The transformation is as followed:

\[
T_{i,i-1} = \text{Rot}(x,\alpha_{i-1}) \cdot \text{Trans}(x,a_{i-1}) \cdot \text{Rot}(z,\theta_i) \cdot \text{Trans}(z,d_i) \quad (A.1)
\]

\[
\text{Rot}(x,\alpha_{i-1}) = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & \cos(\alpha_{i-1}) & \sin(\alpha_{i-1}) & 0 & 0 \\
0 & \sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & 0 & 0 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix} \quad (A.2)
\]

\[
\text{Trans}(x,a_{i-1}) = \begin{bmatrix}
1 & 0 & 0 & a_{i-1} \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \quad (A.3)
\]
\[
R_{\text{ot}(z, \theta_i)} = \begin{bmatrix}
\cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\
\sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(A.4)

\[
T_{\text{rans}(z, d_i)} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & d_i \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(A.5)

where \( T_{i\rightarrow i-1} \) is the transformation matrix for frame \( i \) to frame \( i-1 \).

\( R_{\text{ot}(x, \alpha_{i-1})} \) is the rotation matrix about the x-axis.

\( T_{\text{rans}(x, a_{i-1})} \) is the translation matrix along the x-axis.

\( R_{\text{ot}(z, \theta_i)} \) is the rotation matrix about the z-axis.

\( T_{\text{rans}(z, d_i)} \) is the translation matrix along the z-axis.

Substituting (A.2) through (A.5) into (A.1) yields:

\[
T_{i\rightarrow i-1} = \begin{bmatrix}
c\theta_i & -s\theta_i & 0 & a_{i-1} \\
s\theta_i \ c\alpha_{i-1} & c\theta_i \ c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} \ d_i \\
s\theta_i \ s\alpha_{i-1} & c\theta_i \ s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} \ d_i \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(A.6)

\[
c\theta_i = \cos(\theta_i), \quad s\theta_i = \sin(\theta_i)
\]

(A.7)

\[
c\alpha_{i-1} = \cos(\alpha_{i-1}), \quad s\alpha_{i-1} = \sin(\alpha_{i-1})
\]

(A.8)

The 3 \times 3 matrix at the upper left hand corner of (A.6) is referred as the orientation portion of the transformation. The 3 \times 1 matrix at the upper right hand corner of (A.6) is referred as the translation portion. If only the orientation of the
frame transformation is needed, the orientation portion of (A.6) is sufficient to perform the transformation.

Since the coordinate frame of the intermediate link is chosen as the reference frame, therefore, transformation from frame $i - 1$ to frame $i$ is necessary. To transform from frame $i - 1$ to frame $i$ of a robot with rigid links, the following transformation is used:

\[
T_{i-1\rightarrow i} = \left[ T_{i\rightarrow i-1} \right]^{-1} 
\]

or

\[
T_{i-1\rightarrow i} = \begin{bmatrix}
  c\theta_i & s\theta_i & c\alpha_{i-1} & s\alpha_{i-1} & -c\theta_i & a_{i-1} \\
  -s\theta_i & c\theta_i & c\alpha_{i-1} & c\alpha_{i-1} & s\theta_i & a_{i-1} \\
  0 & -s\alpha_{i-1} & c\alpha_{i-1} & d_i \\
  0 & 0 & 0 & 1 
\end{bmatrix} \tag{A.9}
\]

If link $i$ of a robot is flexible, the transformations stated above are no longer valid. The following discussion concerns the case when:

1) Joint $i$ is a revolute joint.
2) Link $i$ is situated along the $x_i$-axis.
3) Link $i$ exhibits flexible deflections in the $x_i-y_i$ and the $x_i-z_i$ planes.

A flexible link with the specifications stated above has the following transformation.

\[
T_{i\rightarrow i-1} = \text{Rot}(x,\alpha_{i-1}) \cdot \text{Trans}(x,a_{i-1}) \cdot \text{Trans}(y,v_{i-1}(a_{i-1})) \cdot \text{Trans}(z,w_{i-1}(a_{i-1})) \cdot \text{Rot}(y,w_{i-1}(a_{i-1})) \cdot \text{Rot}(z,\theta_{i-1}+v_{i-1}(a_{i-1})) 
\]  \tag{A.11}
\[ \text{Trans}(y, \mathbf{v}_{i-1}(\mathbf{a}_{i-1})) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \mathbf{v}_{i-1}(\mathbf{a}_{i-1}) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]  \hspace{2cm} (A.12) \]

\[ \text{Trans}(z, \mathbf{w}_{i-1}(\mathbf{a}_{i-1})) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \mathbf{w}_{i-1}(\mathbf{a}_{i-1}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \]  \hspace{2cm} (A.13) \]

\[ \text{Rot}(y, \mathbf{w}_{i-1}(\mathbf{a}_{i-1})) = \begin{bmatrix} c\xi_{i-1} & 0 & s\xi_{i-1} & 0 \\ 0 & 1 & 0 & 0 \\ -s\xi_{i-1} & 0 & c\xi_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]  \hspace{2cm} (A.14) \]

\[ \text{Rot}(z, \theta_{i}+\mathbf{v}_{i-1}(\mathbf{a}_{i-1})) = \begin{bmatrix} c\zeta_{i} & -s\zeta_{i} & 0 & 0 \\ s\zeta_{i} & c\zeta_{i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]  \hspace{2cm} (A.15) \]

\[ c\xi_{i-1} = \cos(w_{i-1}(\mathbf{a}_{i-1})) \]  \hspace{2cm} (A.16) \]

\[ s\xi_{i-1} = \sin(w_{i-1}(\mathbf{a}_{i-1})) \]  \hspace{2cm} (A.17) \]

\[ c\zeta_{i} = \cos(\theta_{i}+\mathbf{v}_{i-1}(\mathbf{a}_{i-1})) \]  \hspace{2cm} (A.18) \]

\[ s\zeta_{i} = \sin(\theta_{i}+\mathbf{v}_{i-1}(\mathbf{a}_{i-1})) \]  \hspace{2cm} (A.19) \]

where \( \text{Rot}(y, \mathbf{w}_{i-1}(\mathbf{a}_{i-1})) \) is the rotation matrix about the \( y \)-axis.

\( \text{Trans}(y, \mathbf{v}_{i-1}(\mathbf{a}_{i-1})) \) is the translation matrix along the \( y \)-axis.
Substituting (A.2), (A.4), and (A.12) through (A.19) into (A.11) yield:

\[
T_{i \rightarrow i-1} = \begin{bmatrix}
    n_x & o_x & a_x & p_x \\
    n_y & o_y & a_y & p_y \\
    n_z & o_z & a_z & p_z \\
    0 & 0 & 0 & 1
\end{bmatrix}
\]  

\[n_x = c\zeta_i c\xi_{i-1}\]  

\[n_y = s\alpha_{i-1} c\zeta_i s\xi_{i-1} + c\alpha_{i-1} s\zeta_i\]  

\[n_z = s\alpha_{i-1} s\zeta_i - c\alpha_{i-1} c\zeta_i s\xi_{i-1}\]  

\[o_x = -s\zeta_i c\xi_{i-1}\]  

\[o_y = c\alpha_{i-1} c\zeta_i - s\alpha_{i-1} r\zeta_i s\xi_{i-1}\]  

\[o_z = c\alpha_{i-1} s\zeta_i s\xi_{i-1} + s\alpha_{i-1} c\zeta_i\]  

\[a_x = s\xi_{i-1}\]  

\[a_y = -s\alpha_{i-1} c\xi_{i-1}\]  

\[a_z = c\alpha_{i-1} c\xi_{i-1}\]  

\[p_x = a_{i-1}\]  

\[p_y = c\alpha_{i-1} v_{i-1}(a_{i-1}) - s\alpha_{i-1} w_{i-1}(a_{i-1})\]  

\[p_z = c\alpha_{i-1} w_{i-1}(a_{i-1}) + s\alpha_{i-1} v_{i-1}(a_{i-1})\]  

To transform from frame \(i - 1\) to frame \(i\) of a robot with flexible links, the following transformation is used:

\[
T_{i-1 \rightarrow i} = \left[T_{i \rightarrow i-1}\right]^{-1}
\]  

\[\text{(A.22)}\]
Using the transformation matrices stated above, the coordinate frame of link $k$ may be transformed to link $j$ (where $j < k$) by the following matrix multiplication:

$$T_{k\rightarrow j} = T_{j+1\rightarrow j} \cdot T_{j+2\rightarrow j+1} \cdot \cdots \cdot T_{k-1\rightarrow k-2} \cdot T_{k\rightarrow k-1} \quad (A.25)$$

Similarly, the coordinate frame of link $j$ may be transformed to link $k$ by:

$$T_{j\rightarrow k} = T_{k\rightarrow j}^{-1} \quad (A.26)$$

or by the following matrix multiplication:

$$T_{j\rightarrow k} = T_{k-1\rightarrow k} \cdot T_{k-2\rightarrow k-1} \cdot \cdots \cdot T_{j+1\rightarrow j+2} \cdot T_{j\rightarrow j+1} \quad (A.27)$$

Hence, to determine the transformation from the end-effector coordinate frame to the base frame, the following matrix multiplication is needed:

$$T_{e\rightarrow 0} = T_{1\rightarrow 0} \cdot T_{2\rightarrow 1} \cdot \cdots \cdot T_{i\rightarrow i-1} \cdot \cdots \cdot T_{n\rightarrow n-1} \cdot T_{e\rightarrow n} \quad (A.28)$$
So, the end-effector position with respect to the base frame becomes:

\[
\text{end-effector position} = T_{e-0} \cdot \begin{bmatrix}
0 \\
0 \\
0 \\
1
\end{bmatrix}
\]

(A.29)
APPENDIX B

ASSUMED MODES METHOD

Assumed modes method is an approximation technique for solving free vibration problem. A solution for the free vibration problem is assumed in the form of a series. Each element of the series is composed of an admissible function \( \gamma \) in spatial coordinate and a time-dependent generalize coordinate \( q \). For a continuous system, the solution may be written as:

\[
F(u,t) = \sum_{i=1}^{\infty} \gamma_i(u) q_i(t) \quad (B.1)
\]

For an \( n \)-degree-of-freedom system, the solution may be written as:

\[
F_{nn}(u,t) = \sum_{i=1}^{n} \gamma_i(u) q_i(t) \quad (B.2)
\]
The kinetic energy and potential energy for a continuous system may then be expressed as:

\[ KE = \frac{1}{2} \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} m_{ij} \dot{q}_i(t) \dot{q}_j(t) \]  
(B.3)

\[ PE = \frac{1}{2} \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} k_{ij} q_i(t) q_j(t) \]  
(B.4)

The kinetic energy and potential energy for an \( n \)-degree-of-freedom system may then be expressed as:

\[ KE = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} m_{ij} \dot{q}_i(t) \dot{q}_j(t) \]  
(B.5)

\[ PE = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} k_{ij} q_i(t) q_j(t) \]  
(B.6)

where \( m_{ij} \) are the mass coefficients of the system

\( k_{ij} \) are the stiffness coefficients of the system

The mass coefficients are depend on the distribution of mass of the system and the admissible functions that are chosen; while the stiffness coefficients depend on the stiffness properties of the system, the admissible functions that are chosen and the derivatives of the admissible functions.
APPENDIX C

ADMISSIBLE FUNCTIONS

In order to model the deflection of the flexible links, it is necessary to obtain some kind of representation for the admissible functions $\gamma_i$. To obtain a valid admissible function, the function chosen needs to satisfy the geometrical boundary conditions of the system. For the robot arms in this research, the arms may be modeled as cantilever beams with lumped masses at the end of the beams. Using this approach, the natural boundary conditions as well as the geometrical boundary conditions may be used to obtain the admissible function. By utilizing the natural boundary conditions together with the geometrical boundary conditions of the beam, the result obtained yield a better approximation of the admissible function.

C.1 Case 1. Flexible Beam with Shearing Mass

Figure C.1 depicts a cantilever beam. The beam is composed of a rigid portion and an flexible portion. By applying basic beam assumptions, the equation of motion of the cantilever beam is:

$$\begin{align*}
E I \frac{\partial}{\partial u} v(u,t) + \rho_e \frac{\partial^2}{\partial t^2} v(u,t) &= 0
\end{align*}$$

(C.1)
Figure C.1 Flexible Beam with Shearing Mass
The deflection at the beginning of the flexible portion of the beam is:

\[ v(a_r, t) = 0 \quad \text{(C.2)} \]

The slope of the deflection at the beginning of the flexible portion of the beam is:

\[ \frac{\partial}{\partial u} v(a_r, t) = v'(a_r, t) = 0 \quad \text{(C.3)} \]

The bending moment at the end of the beam is:

\[ \frac{2}{E I} v(a, t) = v''(a, t) = 0 \quad \text{(C.4)} \]

The shear at the end of the beam is:

\[ \frac{3}{E I} v(a, t) = \frac{d}{3} \frac{2}{dt} v(a, t) \quad \text{(C.5)} \]

or

\[ v''''(a, t) = \left[ \frac{1}{E I} \right] \smash{\bar{v}}(a, t) \quad \text{(C.5a)} \]

where \( E \) is the modulus of elasticity of the beam.

\( I \) is the area moment of inertia of the beam in the y direction.

\( v \) is the component of the displacement vector in the y direction.

\( u \) is the component of the displacement vector in the x direction.
\( \rho_e \) is the mass per unit length of the flexible portion of the beam.
\( a_r \) is the length of the rigid portion of the beam.
\( a_e \) is the length of the flexible portion of the beam.
\( a \) is the total length of the beam.
\( s_m \) is the shearing mass at the end of the beam.
\( l_m \) is the lumped mass at the end of the beam.

For this case, the shearing mass for the boundary condition is equal to the lumped mass at the end of the beam. That is:

\[ s_m = l_m \]  \hspace{1cm} (C.6)

For the case where there are several lumped masses attached to the end of the beam, the shearing mass will be the sum of all the lumped masses, or:

\[ s_m = l_m_1 + l_m_2 + ... + l_m_n \]  \hspace{1cm} (C.7)

Since the flexible portion of the beam is the major concern, it is convenience to represent \( v \) by using a non-dimensional variable \( \xi \) for the spatial coordinate,

\[ \xi = \frac{u - a_r}{a - a_r} \]  \hspace{1cm} (C.8)

and assume a periodic time function for the time-dependent generalize coordinate:

\[ q(t) = e^{i \omega t} \]  \hspace{1cm} (C.9)
therefore
\[ v(\xi,t) = \gamma(\xi) e^{i\omega t} \quad (C.10) \]

Now it is clear that, in order to obtain a valid admissible function for \( v \), it is necessary to solve the equation of motion of the cantilever beam (C.1). The first step of solving this equation is to obtain the partial derivatives of \( v \), then substitute these partial derivatives back into the equation of motion. That is,

\[
\frac{\partial \xi}{\partial u} = \frac{1}{a - a_r} \quad (C.11)
\]

then
\[
\frac{\partial}{\partial u} v(\xi,t) = \frac{\partial}{\partial u} \left[ \gamma(\xi) e^{i\omega t} \right] = \frac{\partial}{\partial \xi} \frac{\partial \xi}{\partial u} \left[ \gamma(\xi) e^{i\omega t} \right] = \left[ \frac{1}{a - a_r} \right] \gamma'(\xi) e^{i\omega t} \quad (C.12)
\]

similarly
\[
\frac{\partial^2}{\partial u^2} v(\xi,t) = \left[ \frac{1}{a - a_r} \right]^2 \gamma''(\xi) e^{i\omega t} \quad (C.13)
\]

\[
\frac{\partial^3}{\partial u^3} v(\xi,t) = \left[ \frac{1}{a - a_r} \right]^3 \gamma'''(\xi) e^{i\omega t} \quad (C.14)
\]

\[
\frac{\partial^4}{\partial u^4} v(\xi,t) = \left[ \frac{1}{a - a_r} \right]^4 \gamma^{IV}(\xi) e^{i\omega t} \quad (C.15)
\]

also
\[
\frac{\partial}{\partial t} v(\xi,t) = i\omega \gamma(\xi) e^{i\omega t} \quad (C.16)
\]
\[ \frac{2}{\partial t} v(\xi, t) = -\omega^2 \gamma(\xi) e^{i \omega t} \]  

(C.17)

Therefore, the equation of motion of the beam (C.1) may be rewritten as:

\[ \left[ \frac{1}{\frac{E I}{4}} \right] \gamma^{IV}(\xi) - \rho_e \omega^2 \gamma(\xi) = 0 \]  

(C.18)

or

\[ \gamma^{IV}(\xi) - \beta^4 \gamma(\xi) = 0 \]  

(C.19)

where

\[ \beta = \left[ \frac{1}{E I} \right] \rho_e \omega \left( \frac{a - a_r}{4} \right) \]  

(C.20)

As one might notice, (C.19) is a fourth ordered partial differential equation in the simplest form. From mathematics hand book, (C.19) has the following solution:

\[ \gamma(\xi) = c_1 \sin(\beta \xi) + c_2 \cos(\beta \xi) + c_3 \sinh(\beta \xi) + c_4 \cosh(\beta \xi) \]  

(C.21)

where \( c_1, c_2, c_3, \) and \( c_4 \) are coefficients.

(C.21) is a general solution for \( \gamma \). To obtain a particular solution for the cantilever beam, it is necessary to solve for the coefficients of (C.21) by applying the boundary conditions.

let

\[ \mu = \beta \xi \]  

(C.22)
\[ \gamma(\xi) = c_1 \sin(\mu) + c_2 \cos(\mu) + c_3 \sinh(\mu) + c_4 \cosh(\mu) \]  
(C.23)

\[ \gamma'(\xi) = \beta \{c_1 \cos(\mu) - c_2 \sin(\mu) + c_3 \cosh(\mu) + c_4 \sinh(\mu)\} \]  
(C.24)

\[ \gamma''(\xi) = \beta^2 \{-c_1 \sin(\mu) - c_2 \cos(\mu) + c_3 \sinh(\mu) + c_4 \cosh(\mu)\} \]  
(C.25)

\[ \gamma'''(\xi) = \beta^3 \{-c_1 \cos(\mu) + c_2 \sin(\mu) + c_3 \cosh(\mu) + c_4 \sinh(\mu)\} \]  
(C.26)

At \( u = a_r \) or \( \xi = 0 \), the deflection of the beam yield:

\[ \gamma(0) e^{i \omega t} = 0 \]  
(C.27)

\[ \Rightarrow \quad c_2 + c_4 = 0 \]  
(C.28)

At \( u = a_r \) or \( \xi = 0 \), the slope of the deflection of the beam yield:

\[ \left[ \frac{1}{a - a_r} \right] \gamma'(0) e^{i \omega t} = 0 \]  
(C.29)

\[ \Rightarrow \quad c_1 + c_3 = 0 \]  
(C.30)

At \( u = a \) or \( \xi = 1 \), the bending moment of the beam yield:

\[ \left[ \frac{1}{a - a_r} \right]^2 \gamma''(1) e^{i \omega t} = 0 \]  
(C.31)

\[ \Rightarrow \quad -c_1 \sin(\beta) - c_2 \cos(\beta) + c_3 \sinh(\beta) + c_4 \cosh(\beta) = 0 \]  
(C.32)
At $u = a$ or $\xi = 1$, the shear of the beam yield:

\[
\left[ \frac{1}{a - a_r} \right] \gamma'''(1) e^{i \omega t} = \left( -\omega^2 \right) \left[ \frac{1}{E I} \right] \sin \gamma(1) e^{i \omega t}
\]  
(C.33)

or

\[
\gamma'''(1) + \tau \gamma(1) = 0
\]  
(C.34)

where

\[
\tau = \frac{(a - a_r) \sin \omega}{E I} = \frac{\sin \beta}{\rho_e (a - a_r)}
\]  
(C.35)

\[
=> \begin{bmatrix}
-\beta \cos(\beta) + \tau \sin(\beta) \\
\beta \sin(\beta) + \tau \cos(\beta) \\
\beta \cosh(\beta) + \tau \sinh(\beta) \\
\beta \sinh(\beta) + \tau \cosh(\beta)
\end{bmatrix}
\begin{bmatrix}
c_1 \\
c_2 \\
c_3 \\
c_4
\end{bmatrix} = 0
\]  
(C.36)

Rearrange the results of the boundary conditions in matrix form yield:

\[
\begin{bmatrix}
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 \\
-\sin(\beta) & -\cos(\beta) & \sinh(\beta) & \cosh(\beta) \\
\zeta_1 & \zeta_2 & \zeta_3 & \zeta_4
\end{bmatrix}
\begin{bmatrix}
c_1 \\
c_2 \\
c_3 \\
c_4
\end{bmatrix} = 0
\]  
(C.37)

where

\[
\begin{align*}
\zeta_1 &= -\beta \cos(\beta) + \tau \sin(\beta) \\
\zeta_2 &= \beta \sin(\beta) + \tau \cos(\beta) \\
\zeta_3 &= \beta \cosh(\beta) + \tau \sinh(\beta) \\
\zeta_4 &= \beta \sinh(\beta) + \tau \cosh(\beta)
\end{align*}
\]  
(C.38a-d)
The values for $\beta$ may be obtained by solving the determinant of the matrix and equating the determinant to zero. Since the cantilever beam natural frequencies $\omega$ is related to $\beta$, rearrange (C.20) yield:

$$\omega = \left[ \frac{\beta}{a - a_r} \right]^2 \frac{E I}{\rho e} \quad (C.39)$$

The particular solution of $\gamma$ may now be obtained by rearranging (C.28), (C.30), and (C.32).

from (C.28) \[c_4 = -c_2 \quad (C.40)\]

from (C.30) \[c_3 = -c_1 \quad (C.41)\]

Then the admissible function becomes:

$$\gamma(\xi) = c_1 \{\sin(\mu) - \sinh(\mu)\} + c_2 \{\cos(\mu) - \cosh(\mu)\} \quad (C.42)$$

Substitute (C.40) and (C.41) into (C.32) yield:

$$-c_1 \{\sin(\beta) + \sinh(\beta)\} - c_2 \{\cos(\beta) + \cosh(\beta)\} = 0 \quad (C.43)$$

or

$$c_2 = c_1 \delta \quad (C.44)$$

where

$$\delta = -\frac{\sin(\beta) + \sinh(\beta)}{\cos(\beta) + \cosh(\beta)} \quad (C.45)$$

Since $c_1$ is an arbitrary constant, it is convenience to let $c_1$ equal to 1.

So finally, the admissible function for case 1 becomes:

$$\gamma(\xi) = \{\sin(\mu) - \sinh(\mu)\} + \delta \{\cos(\mu) - \cosh(\mu)\} \quad (C.46)$$
The following listed some of the important notes concerning the out-of-plane admissible function which apply throughout this appendix.

1) The displacement vector component $v$ may be replaced by the displacement vector component $w$ for the out-of-plane motion. The derivation will be identical.

2) According to the right hand rule, when the out-of-plane deflection is positive, the slope of the deflection is negative.

C.2 Case 2. Flexible Beam with Shearing Mass and Bending Moment

Figure C.2 depicted a cantilever beam experiencing a shearing affect and a bending moment at the end of the beam. The equation of motion of the beam remains unchanged as in case 1 and is expressed as:

$$\frac{4}{E} \frac{\partial}{\partial u} v(u,t) + \rho_e \frac{\partial^2}{\partial t^2} v(u,t) = 0 \quad (C.47)$$

The deflection at the beginning of the flexible portion of the beam is:

$$v(a_r,t) = 0 \quad (C.48)$$

The slope of the deflection at the beginning of the flexible portion of the beam is:

$$\frac{\partial}{\partial u} v(a_r,t) = v'(a_r,t) = 0 \quad (C.49)$$
Figure C.2 Flexible Beam with Shearing Mass and Bending Moment
The bending moment at the end of the beam is:

$$E I \frac{\partial^2 v(a,t)}{\partial u^2} = M$$  \hspace{1cm} (C.50)

or

$$v''(a,t) = \frac{M}{E I}$$  \hspace{1cm} (C.50a)

where \( M \) is the bending moment at the end of the beam.

From figure C.2,

$$M = m_{cg} acc_{cg} l_{cg}$$  \hspace{1cm} (C.51)

$$acc_{cg} = \frac{2}{\partial t} \left[ v(a,t) + l_{cg} \sin(v'(a,t)) \right]$$  \hspace{1cm} (C.52)

Applying the small deflection theory, (C.52) becomes:

$$acc_{cg} = \ddot{v}(a,t) + \dot{v}'(a,t) l_{cg}$$  \hspace{1cm} (C.52a)

The shear at the end of the beam is:

$$E I \frac{\partial^3 v(a,t)}{\partial u^3} = \frac{2}{\partial t} \left[ \frac{1}{E I} \right] sm \dddot{v}(a,t)$$  \hspace{1cm} (C.53)

or

$$v'''(a,t) = \left[ \frac{1}{E I} \right] sm \dddot{v}(a,t)$$  \hspace{1cm} (C.53a)

\[ sm = lm + m_{cg} \]  \hspace{1cm} (C.54)
Solving the equation of motion of the beam using the boundary conditions stated above, (C.47) has the following solution:

\[ \gamma(\xi) = c_1 \sin(\mu) + c_2 \cos(\mu) + c_3 \sinh(\mu) + c_4 \cosh(\mu) \]  
(C.55)

where \( c_1, c_2, c_3, \) and \( c_4 \) are coefficients.

To solve for the particular solution of this cantilever beam, the following boundary conditions are applied.

At \( u = a_r \) or \( \xi = 0 \), the deflection of the beam yield:

\[ \gamma(0) e^{i \omega t} = 0 \]  
(C.56)

\[ \Rightarrow c_2 + c_4 = 0 \]  
(C.57)

At \( u = a_r \) or \( \xi = 0 \), the slope of the deflection of the beam yield:

\[ \left[ \frac{1}{a - a_r} \right] \gamma'(0) e^{i \omega t} = 0 \]  
(C.58)

\[ \Rightarrow c_1 + c_3 = 0 \]  
(C.59)

At \( u = a \) or \( \xi = 1 \), the bending moment of the beam yield:

\[ \left[ \frac{1}{a - a_r} \right]^2 \gamma''(1) e^{i \omega t} = \frac{m_{cg} \gamma}{E I} \frac{1}{a - a_r} \left[ \gamma(1) + \frac{\gamma'(1)}{c_{cg}} \right] \]  
(C.60)

or

\[ \gamma''(1) + \tau_1 \gamma'(1) + \tau_2 \gamma(1) = 0 \]  
(C.61)
where
\[
\tau_1 = \frac{(a - a_r) m_{cg} (l_{cg} \omega)^2}{E I} = \frac{m_{cg} l_{cg}^2}{\rho_e (a - a_r)^3} \beta^3 \tag{C.62}
\]
\[
\tau_2 = \frac{(a - a_r) m_{cg} l_{cg}^2}{E I} = \frac{m_{cg} l_{cg}^2}{\rho_e (a - a_r)^2} \beta^2 \tag{C.63}
\]

Substitute (C.23), (C.24), and (C.25) into (C.61) yield:
\[
c_1 \left[ - \beta^2 \sin(\beta) + \tau_1 \beta \cos(\beta) + \tau_2 \sin(\beta) \right] \\
+ c_2 \left[ - (1 - a - a_r) \cos(\beta) - \tau_1 \beta \sin(\beta) + \tau_2 \cos(\beta) \right] \\
+ c_3 \left[ \beta^2 \sinh(\beta) + \tau_1 \beta \cosh(\beta) + \tau_2 \sinh(\beta) \right] \\
+ c_4 \left[ \beta \cosh(\beta) + \tau_1 \beta \sinh(\beta) + \tau_2 \cosh(\beta) \right] = 0 \tag{C.64}
\]

At \( u = a \) or \( \xi = 1 \), the shear of the beam yield:
\[
\left[ \frac{1}{a - a_r} \right]^3 \nu'''(1,t) = \left[ \frac{1}{E I} \right] \sm \nu(1,t) \tag{C.65}
\]
\[
\left[ \frac{1}{a - a_r} \right]^3 \gamma'''(1) e^{i \omega t} = (- \omega^2) \left[ \frac{1}{E I} \right] \sm \gamma(1) e^{i \omega t} \tag{C.66}
\]
or
\[
\gamma'''(1) + \tau_3 \gamma(1) = 0 \tag{C.67}
\]

where
\[
\tau_3 = \frac{(a - a_r)^3 \sm}{E I} = \frac{\sm}{\rho_e (a - a_r)} \beta^4 \tag{C.68}
\]
Substituting equation (C.23) and (C.26) into (C.67) yield:

\[
\begin{align*}
&c_1 \left[ -\beta \cos(\beta) + \tau_3 \sin(\beta) \right] \\
&\quad + c_2 \left[ \beta \sin(\beta) + \tau_3 \cos(\beta) \right] \\
&\quad + c_3 \left[ 3 \cosh(\beta) + \tau_3 \sinh(\beta) \right] \\
&\quad + c_4 \left[ \beta \sinh(\beta) + \tau_3 \cosh(\beta) \right] = 0
\end{align*}
\] (C.69)

Rearrange the results of the boundary condition into matrix form yield:

\[
\begin{bmatrix}
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 \\
\xi_1 & \xi_2 & \xi_3 & \xi_4 \\
\zeta_1 & \zeta_2 & \zeta_3 & \zeta_4
\end{bmatrix}
\begin{bmatrix}
c_1 \\
c_2 \\
c_3 \\
c_4
\end{bmatrix} = 0
\] (C.70)

where

\[
\begin{align*}
\xi_1 &= - 2 \beta \sin(\beta) + \tau_1 \beta \cos(\beta) + \tau_2 \sin(\beta) \\
\xi_2 &= - 2 \beta \cos(\beta) - \tau_1 \beta \sin(\beta) + \tau_2 \cos(\beta) \\
\xi_3 &= 2 \beta \sinh(\beta) + \tau_1 \beta \cosh(\beta) + \tau_2 \sinh(\beta) \\
\xi_4 &= 2 \beta \cosh(\beta) + \tau_1 \beta \sinh(\beta) + \tau_2 \cosh(\beta)
\end{align*}
\] (C.71)

\[
\begin{align*}
\zeta_1 &= - 3 \beta \cos(\beta) + \tau_3 \sin(\beta) \\
\zeta_2 &= 3 \beta \sin(\beta) + \tau_3 \cos(\beta) \\
\zeta_3 &= 3 \beta \cosh(\beta) + \tau_3 \sinh(\beta) \\
\zeta_4 &= 3 \beta \sinh(\beta) + \tau_3 \cosh(\beta)
\end{align*}
\] (C.72)
The particular solution for this cantilever beam may now be obtained by
rearranging equation (C.57), (C.59), and (C.69).

from (C.57) \[ c_4 = -c_2 \] (C.73)
from (C.59) \[ c_3 = -c_1 \] (C.74)

Then the admissible function becomes:

\[ \gamma(\xi) = c_1 \{\sin(\mu) - \sinh(\mu)\} + c_2 \{\cos(\mu) - \cosh(\mu)\} \] (C.75)

Substitute (C.73) and (C.74) into (C.69) yield:

\[
c_1 \left\{ \left[ -\beta^3 \cos(\beta) + \tau_3 \sin(\beta) \right] - \left[ \beta^3 \cosh(\beta) + \tau_3 \sinh(\beta) \right] \right\} + c_2 \left\{ \left[ \beta^3 \sin(\beta) + \tau_3 \cos(\beta) \right] - \left[ \beta^3 \sinh(\beta) + \tau_3 \cosh(\beta) \right] \right\} = 0
\] (C.76)

or

\[ c_2 = c_1 \delta \] (C.77)

\[
\delta = -\frac{\left[ -\beta^3 \cos(\beta) + \tau_3 \sin(\beta) \right] - \left[ \beta^3 \cosh(\beta) + \tau_3 \sinh(\beta) \right]}{\left[ \beta^3 \sin(\beta) + \tau_3 \cos(\beta) \right] - \left[ \beta^3 \sinh(\beta) + \tau_3 \cosh(\beta) \right]}
\] (C.78)

Since \( c_1 \) is an arbitrary constant, it is convenience to let \( c_1 \) equal to 1.

So finally, the admissible function for case 2 becomes:

\[
\gamma(\xi) = \{\sin(\mu) - \sinh(\mu)\} + \delta \{\cos(\mu) - \cosh(\mu)\}
\] (C.79)
C.3 Case 3. Flexible Beam with Joint End

Figure C.3 depicts a cantilever beam with a joint end. The deflection at the beginning of the flexible portion of the beam is:

\[ v(a_r, t) = 0 \]  \hspace{1cm} (C.80)

The slope of the deflection at the beginning of the flexible portion of the beam is:

\[ \frac{\partial}{\partial u} v(a_r, t) = v'(a_r, t) = 0 \]  \hspace{1cm} (C.81)

The bending moment at the end of the beam is:

\[ \frac{2}{E I} \frac{\partial^2}{\partial u^2} v(a, t) = v''(a, t) = 0 \]  \hspace{1cm} (C.82)

It is assumed that the admissible function \( \gamma \) has the following general solution:

\[ \gamma(\xi) = c_0 + c_1 \xi + c_2 \xi^2 + c_3 \xi^3 \]  \hspace{1cm} (C.83)

where \( c_0, c_1, c_2, \) and \( c_3 \) are coefficients.
Figure C.3 Flexible Beam with Joint End
To solve for the particular solution of this cantilever beam, the following boundary conditions are applied.

At \( u = a_r \) or \( \xi = 0 \), the deflection of the beam yield:

\[
\gamma(0) e^{i \omega t} = 0
\]

\[
\Rightarrow c_0 = 0
\]

At \( u = a_r \) or \( \xi = 0 \), the slope of the deflection of the beam yield:

\[
\begin{bmatrix}
1 \\
\frac{1}{a - a_r}
\end{bmatrix}
\gamma'(0) e^{i \omega t} = 0
\]

\[
\Rightarrow c_1 = 0
\]

At \( u = a \) or \( \xi = 1 \), the bending moment of the beam yield:

\[
\begin{bmatrix}
1 \\
\frac{1}{a - a_r}
\end{bmatrix}^2
\gamma''(1) e^{i \omega t} = 0
\]

\[
\Rightarrow 2 c_2 + 6 c_3 = 0
\]

Rearranging (C.89) yields:

\[
c_3 = -\frac{1}{3} c_2
\]
From (C.85), (C.87), and (C.90), the admissible function becomes:

\[ \gamma(\xi) = c_2 \begin{bmatrix} 2 & 1 & 3 \\ \xi & -\frac{1}{3} \xi & \end{bmatrix} \]  

(C.91)

Since \( c_2 \) is an arbitrary constant, it is convenient to let \( c_2 \) equal to 1.

So finally, the admissible function for case 3 becomes:

\[ \gamma(\xi) = \frac{2}{3} - \frac{1}{3} \xi \]  

(C.92)

**C.4 Case 4. Flexible Beam with Constrained End**

Figure C.4 depicts a cantilever beam with the deflection and the slope of the deflection at the end being constrained. The deflection at the beginning of the flexible portion of the beam is:

\[ v(a_r, t) = 0 \]  

(C.93)

The slope of the deflection at the beginning of the flexible portion of the beam is:

\[ \frac{\partial}{\partial u} v(a_r, t) = v'(a_r, t) = 0 \]  

(C.94)

The deflection at the end of the beam is:

\[ v(a, t) = \Delta \]  

(C.95)

where \( \Delta \) is the deflection at the end of the beam.
Figure C.4 Flexible Beam with Constrained End
The slope of the deflection at the end of the beam is:

\[
\frac{\partial}{\partial t} v(a, t) = v'(a, t) = \Lambda
\]  \hspace{1cm} (C.96)

where \( \Lambda \) is the slope of deflection at the end of the beam.

It is assumed that the admissible function \( \gamma \) has the following general solution:

\[
\gamma(\xi) = c_0 + c_1 \xi + c_2 \xi^2 + c_3 \xi^3
\]  \hspace{1cm} (C.97)

where \( c_0, c_1, c_2, \) and \( c_3 \) are coefficients.

To solve for the particular solution of this cantilever beam, the following boundary conditions are applied.

At \( u = a_r \) or \( \xi = 0 \), the deflection of the beam yield:

\[
\gamma(0) e^{i \omega t} = 0
\]  \hspace{1cm} (C.98)

\[
\Rightarrow \quad c_0 = 0 \hspace{1cm} (C.99)
\]

At \( u = a_r \) or \( \xi = 0 \), the slope of the deflection of the beam yield:

\[
\left[ \frac{1}{a - a_r} \right] \gamma'(0) e^{i \omega t} = 0
\]  \hspace{1cm} (C.100)

\[
\Rightarrow \quad c_1 = 0 \hspace{1cm} (C.101)
\]
At \( u = a \) or \( \xi = 1 \), the deflection of the beam yield:

\[
\gamma(1) \, q(t) = \Delta 
\]
\[\text{(C.102)}\]

\( \Rightarrow \)

\[
(c_2 + c_3) \, q(t) = \Delta 
\]
\[\text{(C.103)}\]

At \( u = a \) or \( \xi = 1 \), the slope of the deflection of the beam yield:

\[
\left[ \frac{1}{a - a_r} \right] \gamma'(1) \, q(t) = \Lambda
\]
\[\text{(C.104)}\]

\( \Rightarrow \)

\[
\left[ \frac{1}{a - a_r} \right] (2 \, c_2 + 3 \, c_3) \, q(t) = \Lambda
\]
\[\text{(C.105)}\]

Rearranging (C.90) and (C.92) yields:

\[
\left[ \frac{1}{a - a_r} \right] \frac{(2 \, c_2 + 3 \, c_3)}{(c_2 + c_3)} = \frac{\Lambda}{\Delta}
\]
\[\text{(C.106)}\]

or

\[
c_3 = - c_2 \left[ \frac{(2 - \tau)}{(3 - \tau)} \right]
\]
\[\text{(C.107)}\]

where

\[
\tau = (a - a_r) \left[ \frac{\Lambda}{\Delta} \right]
\]
\[\text{(C.108)}\]

From (C.99), (C.101), and (C.107), the admissible function becomes:

\[
\gamma(\xi) = c_2 \left\{ \xi - \left[ \frac{(2 - \tau)}{(3 - \tau)} \right] \xi^3 \right\}
\]
\[\text{(C.96)}\]
Since $c_2$ is an arbitrary constant, it is convenience to let $c_2$ equal to 1.

So finally, the admissible function for case 3 becomes:

$$\gamma(\xi) = \xi - \left[ \frac{(2 - \tau)}{(3 - \tau)} \right]^3 \xi$$

(C.97)
APPENDIX D

MACSYMA PROGRAM LISTINGS OF THE TWO–LINK ROBOT WITH ONE FLEXIBLE OUTER–MOST LINK
/* Program name: Coeff.com

This command procedure routine helps to determine the coefficients of the differential equations.

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

Extract the 'F' coefficient.
/*
qv2dd : 0$
coefff : ev ( line2 )$
/*

Set variables to zero for the coefficients of [ D ].
/*
g : 0$
eiv2 : 0$
th2d : 0$
th2dd : 0$
qv2d : 0$

Extract the 'A' coefficient. That is, 'A' = qv2.
/*
qv2dd : 1$
coeffa : ev ( line2 )$
/*
Done, see you later !!!
*/

/* Program name: Coeff1.com

This subroutine calls other subroutines to extract the coefficients from the equations of motion for link 1.

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

Remove the content of the time-dependent generalized coordinate.
/*
batchload ( "killvar.com" )$
/*
Call subroutine to extract the coefficients of link 1.
/*
batchload ( "coeff.com" )$
/*
That's it folks...
*/

/* Program name: Coeff2.com
This subroutine calls other subroutines to extract the coefficients from the equations of motion for link 2.

*******************************************************************
Remove the content of the time-dependent generalized coordinate.
*/
batchload ( "killvar.com" )$
/*  ***************************************************************
Replace the derivatives of the time-dependent generalized coordinate with their variable names.
*/
batchload ( "simplify2.com" )$
/*  ***************************************************************
Call subroutine to extract the coefficients of link 2.
*/
batchload ( "coeff.com" )$
/*  ***************************************************************
That's it folks...
*/

/*/ Program name : Energy1.com
This routine helps to set up the kinetic energy equation, the potential energy equation, and the Lagrange's equation for link 1.
*******************************************************************
Find out the kinetic and potential energy equations for link 1.
*/
lagrange : integrate ( rho1 * ( vector_1dtsq / 2 + gdotvector_1 ), w1, 0, -d1 )$
/*  ***************************************************************
Find out the Lagrange's equations from the given energy equation.
*/
batchload ( "lagrange.com" )$
/*  ***************************************************************
That's it folks...
*/

/*/ Program name : Energy2.com
This routine helps to set up the kinetic energy equations, the potential energy equations, and the Lagrange operator for link 2.
*******************************************************************
Find out the kinetic and potential energy equations for link 2.
*/
lagrange : integrate ( rho2r * vector_2rdsq / 2, u2, 0, a2r ) +
integrate ( rho2r * gdotvector_2r, u2, 0, a2r ) +
integrate ( rho2e * vector_2edsq / 2, u2, a2r, a2 ) +
integrate ( rho2e * gdotvector_2e, u2, a2r, a2 ) +
integrate ( -eiv2 * v2pp*2 / 2, u2, a2r, a2 ) +
integrate ( -eiw2 * w2pp*2 / 2, u2, a2r, a2 ) +
sm2 * ( vector_2e_at_a2dsq / 2 + gdotvector_2e_at_a2 )$

J it

Find out the Lagrangian equations from the given energy equations.

*/
batchload ( "lagrange.com" )$

/*
That's it folks...
*/

/* Program name : Init.com

This command procedure routine helps to initialize the system variables for later use.

*******************************************************************************/
cross_product(i, j,k) := [ unib * unitz - unite * unity,
unite * unitx - unita * unitz,
unita * unity - unitb * unitx ]$

*******************************************************************************/
transform_1_to_2 : matrix ( [ cos(th2(t)), 0, sin(th2(t)) ],
[ -sin(th2(t)), 0 , cos(th2(t)) ],
[ 0, -1 , 0 ] )$
gravity_at_1 : matrix ( [ 0 ], [ 0 ], [ -g ] )$
gravity_at_2 : transpose ( transform_1_to_2 . gravity_at_1 )$

J it

That's it for this routine, bye...

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

/* Program name : Killvar.com

This routine helps to remove the content of the time derivatives of the generalized coordinates.

*******************************************************************************/
kill ( th2d )$
kill ( qv2d )$
kill ( qv2dd )$

*******************************************************************************/
Mission accomplished, see you...

/* Program name: Lagrange.com
   This routine helps to formulate the Lagrangian equations.
   *****************************************************************************/

line1 : diff ( diff ( lagrange, th2d ), t ) - diff ( lagrange, th2(t) )
line2 : diff ( diff ( lagrange, qv2d ), t ) - diff ( lagrange, qv2(t) )

That's it for now, later...

/* Program name: Link1.com
   This is the procedure routine that helps to derive the equations of motion for the first link of the robot.
   *****************************************************************************/

nolabels : true$

/* Setup Macsyma parameter. */

batchload ( "init.com" )$

/* Set up variables and parameters for link 1. */

batchload ( "varsetup1.com" )$

/* Set up the energy equations and the lagrange equations for link 1. */

batchload ( "energy1.com" )$

/* Call subroutine to extract the coefficients of link 1. */

batchload ( "coeff1.com" )$

/* Create output listing files and save the results in a file for link 1. */
batchload ("save1.com")$
/*  *******************************************************************
   Mission accomplished, later.
*/
quit ()$

(run)
" Program name : Link2.com
   This is the procedure routine that helps to derive the equations
of motion for the second link of the robot.
   *******************************************************************
   Setup Macsyma parameter.
*/
notlabels : trues
/*  *******************************************************************
   Call subroutine to initialize variables.
*/
batchload ("init.com")$
/*  *******************************************************************
   Set up variables and parameters for link 1 and link 2.
*/
batchload ("varsetup1.com")$
batchload ("varsetup2.com")$
/*  *******************************************************************
   Set up the energy equations and the lagrange equations for link 2.
*/
batchload ("energy2.com")$
/*  *******************************************************************
   Call subroutine to extract the coefficients of link 2.
*/
batchload ("coeff2.com")$
/*  *******************************************************************
   Create output listing files and save the results in a file for
link 2.
*/
batchload ("save2.com")$
/*  *******************************************************************
   Mission accomplished, later.
*/
quit ()$

/* Program name: Listing.com
   This command procedure routine helps to create output listings for this project.
   *******************************************************************
   Print the equations into a listing file.
*/
writefile (filename1)$
line1;
line2;
closefile ()$
/*  *******************************************************************
   Print all the coefficients into a listing file.
*/
writefile (filename2)$
coeffa;
coefff;
closefile ()$
/*  *******************************************************************
   Done!!!
*/

(run)
/* Program name: Pickapart.com
   This subroutine helps to breakdown the equations of the system into smaller elements.
   *******************************************************************
   Load the data file of link 1 and breakdown the equations.
*/
loadfile ("link1.fil")$
print_torque2_1 : torque2l1 = pickapart (linel, 1);
print_coeffa_1 : coeffall = pickapart (coeffa, 1);
print_coefff_1 : coeffF1 = pickapart (coefff, 1);
save ("pickapart1.fil",
      print_torque2_1, print_coeffa_1, print_coefff_1, labels )$
/*  *******************************************************************
   Load the data file of link 2 and breakdown the equations.
*/
loadfile ("link2.fil")$
print_torque2_2 : torque2l2 = pickapart (linel, 1);
print_coeffa_2 : coeffall2 = pickapart (coeffa, 1);
print_coefff_2 : coeffF12 = pickapart (coefff, 1);
217

save ( "pickapart2.fil", print_torque2_2, print_coeffa_2, print_coefff_2, labels );

Mission accomplished, exit program.

quit ();

(run)

Program name: Reduce.com

This command procedure file helps to simplify and reduce the size of the equations of the system.

Call subroutines to perform the simplifications.

batchload ( "reduce1.com" );

batchload ( "reduce2.com" );

Setup the elements of the solution matrix and setup the final torque equations.

batchload ( "sumup.com" );

Mission accomplished, see ya...

quit ();

/* Program name: Reduce1.com

This subroutine helps to simplify all the equations of link 1.

Clear the memory.

kill ( all );

Load file into memory.

loadfile ( "pickapart1.fil" );

Translate the equations into Fortran codes.

writefile ( "fortran1.lst" );
fortran ( print_torque2_1 );
fortran ( print_coeffa_1 );
fortran ( print_coefff_1 );
closefile ();

/* *******************************************************************
   Mission accomplished, later...
*/

/* Program name : Reduce2.com
   This subroutine helps to simplify all the equations of link 2.
   *******************************************************************
   Clear the memory.
*/
k il l ( all );

/* *******************************************************************
   Load file into memory.
*/
loadfile ( "pickapart2.fil" );

/* *******************************************************************
   Simplify the torque equations.
*/
temp : part ( e7, 1, 2, 1 )$
k il l ( e7 )$
b at chload ( " subreduce2.com" )$
print_t_element1_2 : element ( 1, i ) = temp$
print_t_e7 : e7 = rho2e * integral ( 1 ) / 2$
temp : part ( e8, 3, 1 )$
k il l ( e8 )$
b at chload ( " subreduce2.com" )$
print_t_element2_2 : element ( 2, i ) = temp$
print_t_e8 : e8 = g * rho2e * integral ( 2 )$
temp : e9$
k il l ( e9 )$
b at chload ( " subreduce2.com" )$
print_t_e9 : e9 = temp$
temp : e10$
k il l ( e10 )$
b at chload ( " subreduce2.com" )$
print_t_e10 : e10 = temp$
temp : e11$
k il l ( e11 )$
b at chload ( " subreduce2.com" )$
print_t_e11 : e11 = temp$
temp : e12$
k il l ( e12 )$
b at chload ( " subreduce2.com" )$
print_t_e12 : e12 = temp$

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

Simplify the coefficient equations.
*

temp : part ( e13, 2, 1 )
kill ( e13 )
batchload ( "subreduce2.com" )
print_c_element1_2 : element ( 1, i ) = temp
print_c_e13 : e13 = rho2e * integral ( 1 )

temp : e14$
kill ( e14 )$
batchload ( "subreduce2.com" )$
print_c_e14 : e14 = temp$

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


temp : part ( e15, 3, 1 )$
kill ( e15 )$
batchload ( "subreduce2.com" )$
print_c_element2_2 : element ( 2, i ) = temp$
print_c_e15 : e15 = evl2 * qv2 * integral ( 2 )$

temp : part ( e16, 1, 4, 1 )$
kill ( e16 )$
batchload ( "subreduce2.com" )$
print_c_element3_2 : element ( 3, i ) = temp$
print_c_e16 : e16 = - rho2e * qv2 * th2dsq * integral ( 3 )$

temp : part ( e17, 3, 1 )$
kill ( e17 )$
batchload ( "subreduce2.com" )$
print_c_element4_2 : element ( 4, i ) = temp$
print_c_e17 : e17 = rho2e * th2dd * integral ( 4 )$

temp : part ( e18, 4, 1 )$
kill ( e18 )$
batchload ( "subreduce2.com" )$
print_c_element5_2 : element ( 5, i ) = temp$
print_c_e18 : e18 = g * rho2e * costh2 * integral ( 5 )$

temp : e19$
kill ( e19 )$
batchload ( "subreduce2.com" )$
print_c_e19 : e19 = temp$

temp : e20$
kill ( e20 )$
batchload ( "subreduce2.com" )$
print_c_e20 : e20 = temp$

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

Translate the equations into Fortran codes.
*

writefile ( "fortran2.lst" );

fortran ( print_t_element1_2 );
fortran ( print_t_element2_2 );
fortran ( print_t_e7 );
fortran ( print_t_e8 );
fortran ( print_t_e9 );
fortran ( print_t_e10 );
fortran ( print_t_e11 );
fortran ( print_t_e12 );
fortran ( print_torque2_2 );
fortran ( print_c_element1_2 );
fortran ( print_c_element2_2 );
fortran ( print_c_element3_2 );
fortran ( print_c_element4_2 );
fortran ( print_c_element5_2 );
fortran ( print_c_e13 );
fortran ( print_c_e14 );
fortran ( print_c_e15 );
fortran ( print_c_e16 );
fortran ( print_c_e17 );
fortran ( print_c_e18 );
fortran ( print_c_e19 );
fortran ( print_c_e20 );
fortran ( print_coeffa2 );
fortran ( print_coefff2 );
closefile();

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

Mission accomplished, later...
*/

/* Program name : Save1.com

This command procedure file helps to create output listings and saves the results of link 1.

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

filename1 : "link1.lst"
filename2 : "coeff1.lst"
batch ( "listing.com" )$
save ( "link1.fil", coeffa, coefff, line1, line2 )$

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

That's it folks...
*/

/* Program name : Save2.com

This command procedure file helps to create output listings and saves the results of link 2.

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

filename1 : "link2.lst"
filename2 : "coeff2.lst"
batch ( "listing.com" )$
save ( "link2.fil", coeffa, coefff, line1, line2 )$

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

That's it folks...
*/
/* Program name: Simplify2.com

This subroutine is targeted at the equations of link 2. It helps to replace the derivatives of the time-dependent generalized coordinate with their variable names.

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

temp : line1$
batchload ( "subsimp2.com" )$
line1 : temp$

temp : line2$
batchload ( "subsimp2.com" )$
line2 : temp$

/* Mission accomplished, see you... */

/* Program name: Subreduce2.com

This is the command procedure routine that helps to reduce and simplify the current link 2's equation in equation 'temp'.

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

temp : subst ( a2sq, a2^2, temp )$
temp : subst ( a2rsq, a2r^2, temp )$
temp : subst ( a2rcu, a2r^3, temp )$
temp : subst ( costh2, cos ( th2(t) ), temp )$
temp : subst ( costh2sq, costh2^2, temp )$
temp : subst ( sinth2, sin ( th2(t) ), temp )$
temp : subst ( th2dsq, th2d^2, temp )$
temp : subst ( qv2, qv2(t), temp )$
temp : subst ( qv2sq, qv2^2, temp )$
temp : subst ( qv2dsq, qv2d^2, temp )$
temp : subst ( g2psaq(i), ( diff ( g2(u2), u2, 2 ) )^2, temp )$
temp : subst ( g2(i), g2(u2), temp )$
temp : subst ( g2aq(i), g2a(i)*2, temp )$
temp : subst ( g2a2sq, g2a2^2, temp )$
temp : subst ( g2a2psq, g2a2p^2, temp )$
temp : subst ( u2(i), u2, temp )$

/* Mission accomplished, see you... */

/* Program name: Subsimp2.com

This subroutine helps to replace the time-dependent generalized coordinates with their variable names. The replacement is for the current link 2's equation stored in 'temp'.

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

temp : subst ( th2d, diff ( th2(t), t ), temp )$
temp : subst ( th2dd, diff ( th2(t), t, 2 ), temp )$
temp : subst ( qv2d, diff ( qv2(t), t ), temp )$
temp : subst ( qv2dd, diff ( qv2(t), t, 2 ), temp )$
Mission accomplished, see you...

Program name: Sumup.com

This subroutine helps to setup the elements of the solution matrix and sum up the torque generated from each link.

Sum up the torque of link 2.

\[
\text{torque}_2 : \text{torque}(2) = \text{torque}_2^{l1} + \text{torque}_2^{l2}
\]

Sum up the coefficients of each link.

\[
\text{coeff}_a : \text{coeffa} = \text{coeff}_{a1} + \text{coeff}_{a2}
\]
\[
\text{coeff}_f : \text{coefff} = -(\text{coeff}_{f1} + \text{coeff}_{f2})
\]

Translate the equations into Fortran codes.

\[
\text{writefile ("fortran.lst");}
\]
\[
\text{fortran (torque}_2\text{);}
\]
\[
\text{fortran (coeff}_a\text{);}
\]
\[
\text{fortran (coeff}_f\text{);}
\]
\[
\text{closefile();}
\]

Done. Bye...

Program name: Varsetup1.com

This routine helps to setup all the variables and parameters for link 1.

Setup the general variables for link 1.

\[
\text{omega}_1\_at\_1 : \text{matrix}([0],[0],[0])
\]
\[
\text{omega}_1\_1 : \text{transpose} (\text{transform}_1\_to\_2 . \text{omega}_1\_at\_1)
\]
\[
\text{velocity}_\text{at}_0(i,j,k) := [0,0,0]
\]

Set up the displacement vector and the velocity vector for link 1.
vector_1_at_1 : matrix ( [ 0 ], [ 0 ], [ -w1 ] )$
vector_1 : transpose ( transform_1_to_2 . vector_1_at_1 )$
gdotvector_1 : trigreduce ( gravity_at_2 . vector_1 )$

omega_1_x_vector_1 : cross_product(i,j,k)$
omega_1_x_vector_1 : subst ( part(omega_1(1,1), unita, omega_1_x_vector_1 )$
omega_1_x_vector_1 : subst ( part(omega_1,1,2), unitb, omega_1_x_vector_1 )$
omega_1_x_vector_1 : subst ( part(omega_1,1,3), unite, omega_1_x_vector_1 )$
omega_1_x_vector_1 : subst ( part(vector_1,1,1), unitx, omega_1_x_vector_1 )$
omega_1_x_vector_1 : subst ( part(vector_1,1,2), unity, omega_1_x_vector_1 )$
omega_1_x_vector_1 : subst ( part(vector_1,1,3), unitz, omega_1_x_vector_1 )$

vector_1d : velocity_at_0 (i,j,k ) +  omega_1_x_vector_1

vector_1dsq : vector_1d . vector_1d$

That's it folks...
Set up the displacement vector and the velocity vector for the elastic portion of link 2.

vector_2e : matrix ( [ u2, v2, 0 ] )$

vector_2e_at_a2 : matrix ( [ a2, v2_at_a2, 0 ] )$

gdotvector_2e : factor ( gravity_at_2 . vector_2e )$

gdotvector_2e_at_a2 : factor ( gravity_at_2 . vector_2e_at_a2 )$

omega_2_x_vector_2e : cross_product(i, j, k)$

omega_2_x_vector_2e : subst ( part(omega_2,1,1), unita, omega_2_x_vector_2e )$

omega_2_x_vector_2e : subst ( part(omega_2,1,2), unitb, omega_2_x_vector_2e )$

omega_2_x_vector_2e : subst ( part(omega_2,1,3), unitc, omega_2_x_vector_2e )$

omega_2_x_vector_2e : subst ( part(vector_2e,1,1), unitx, omega_2_x_vector_2e )$

omega_2_x_vector_2e : subst ( part(vector_2e,1,2), unity, omega_2_x_vector_2e )$

omega_2_x_vector_2e : subst ( part(vector_2e,1,3), unitz, omega_2_x_vector_2e )$

projection_v2(i,j,k) := [ 0, 1, 0 ]$

vector_2ed : velocity_at_1 + omega_2_x_vector_2e +
v_2d * projection_v2(i,j,k)$

vector_2edsq : vector_2ed . vector_2ed$

vector_2e_at_a2d : velocity_at_1 + omega_2_x_vector_2e_at_a2 +
v_2_at_a2d * projection_v2(i,j,k)$

vector_2e_at_a2dsq : vector_2e_at_a2d . vector_2e_at_a2d$

That's it folks...
APPENDIX E

MACSYMA PROGRAM LISTINGS OF THE THREE-LINK ROBOT WITH
TWO FLEXIBLE OUTER-MOST LINKS
/* Program name: Coeff.com
This command procedure routine helps to determine the coefficients
of the differential equations.
***************************************************************************/
/*
Extract the 'F' coefficient.
*/
qv2dd : 0$
qv3dd : 0$
coefff1 : ev ( line4 )$
coefff2 : ev ( line5 )$
/* *************************************************************************/
/*
Extract the 'A' coefficient. That is, 'A' * qv2.
*/
qv2dd : 1$
coeffa1 : ev ( line4 )$
coeffa2 : ev ( line5 )$
/* *************************************************************************/
/*
Extract the 'b' coefficient. That is, 'b' * qv3.
*/
qv2dd : 0$
qv3dd : 1$
coeffb1 : ev ( line4 )$
coeffb2 : ev ( line5 )$
/* *************************************************************************/
Done, see you later !!! */

/* Program name: Coeff1.com
This subroutine calls other subroutines to extract the coefficients
from the equations of motion for link 1.
***************************************************************************/
/*
Remove the content of the time-dependent generalized coordinate.
*/
batchload ("killvar.com")$

Replace the derivatives of the time-dependent generalized coordinate with their variable names.

batchload ("simplify1.com")$

Call subroutine to extract the coefficients of link 1.

batchload ("coeff.com")$

That's it folks...

Program name: Coeff2.com

This subroutine calls other subroutines to extract the coefficients from the equations of motion for link 2.

Remove the content of the time-dependent generalized coordinate.

batchload ("killvar.com")$

Replace the derivatives of the time-dependent generalized coordinate with their variable names.

batchload ("simplify2.com")$

Call subroutine to extract the coefficients of link 2.

batchload ("coeff.com")$

That's it folks...

Program name: Coeff3.com

This subroutine calls other subroutines to extract the coefficients from the equations of motion for link 3.

Remove the content of the time-dependent generalized coordinate.
batchload ( "killvar.com" )$
/*
   **************************************************************************************
   Replace the derivatives of the time-dependent generalized
   coordinate with their variable names.
   */
batchload ( "simplify3.com" )$
/*
   **************************************************************************************
   Call subroutine to extract the coefficients of link 3.
   */
batchload ( "coeff.com" )$
/*
   **************************************************************************************
   That's it folks...
   */

/* Program name : Energy1.com
This routine helps to set up the kinetic energy equations, the
potential energy equations, and the Lagrange operator for link 1.
**************************************************************************************

Find out the kinetic and potential energy equations for link 1.
*/
lagrange : ( j1 * th1d2 ) / 2 +
   integrate ( rho1 * ( vector_1dsq / 2 + gdotvector_1 ),
   w1, 0, -d1 )$
/*
   **************************************************************************************
   Find out the Lagrangian equations from the given energy equations.
   */
batchload ( "lagrange.com" )$
/*
   **************************************************************************************
   That's it folks...
   */

/* Program name : Energy2.com
This routine helps to set up the kinetic energy equations, the
potential energy equations, and the Lagrange operator for link 2.
**************************************************************************************

Find out the kinetic and potential energy equations for link 2.
*/
lagrange : integrate ( rho2r * vector_2rdsq / 2, u2, 0, a2r ) +
   integrate ( rho2r * gdotvector_2r, u2, 0, a2r ) +
   integrate ( rho2e * vector_2edsq / 2, u2, a2r, a2 ) +
   integrate ( rho2e * gdotvector_2e, u2, a2r, a2 ) +
   integrate ( eiv2 * v2pp2 / 2, u2, a2r, a2 ) +
   sm2 * ( vector_2e_at_a2dsq / 2 + gdotvector_2e_at_a2 )$
/*  **************************************************************************
Find out the Lagrangian equations from the given energy equations.
*/
batchload ( "lagrange.com" )$
/*  **************************************************************************
That's it folks...
*/

/* Program name : Energy3.com

This routine helps to set up the kinetic energy equations, the
potential energy equations, and the Lagrange operator for link 3.
**************************************************************************
Find out the kinetic and potential energy equations for link 3.
*/
lagrange : integrate ( rho3 * vector_3dsq / 2, u3, 0, a3 ) +
integrate ( rho3 * gdotvector_3, u3, 0, a3 ) +
integrate ( -eiv3 * v3pp*2 / 2, u3, 0, a3 ) +
sm3 * ( vector_3_at_a3dsq / 2 + gdotvector_3_at_a3 )$
/*  **************************************************************************
Find out the Lagrangian equations from the given energy equations.
*/
batchload ( "lagrange.com" )$
/*  **************************************************************************
That's it folks...
*/

(run)
/* Program name : Error.com

This command procedure routine helps to determine the trajectories
of the rigid and flexible robots.
**************************************************************************
Set up the rotation and translation variables.
*/
d1 : d1;
rotx : matrix ( [1,0,0,0], [0,cos(angle),-sin (angle),0],
[0,sin(angle),cos(angle),0], [0,0,0,1] );
roty : matrix ( [cos(angle),0,sin(angle),0], [0,1,0,0],
[-sin(angle),0,cos(angle),0], [0,0,0,1] );
rotz : matrix ( [cos(angle),-sin(angle),0,0],
[sin(angle),cos(angle),0,0],
[0,0,1,0], [0,0,0,1] );
transx : matrix ( [1,0,0,xpos], [0,1,0,0], [0,0,1,0], [0,0,0,1] );
transy : matrix ( [1,0,0,0], [0,1,0,x ypos]. [0,0,1,0], [0,0,0,1] );
transz : matrix ( [1,0,0,0], [0,1,0,0], [0,0,1,xpos], [0,0,0,1] );
endcoord : matrix ( [0], [0], [0], [1] );

/* ******************************************
  Set up the matrix dot product of each link.
*/

rot_theta : subst ( th1, angle, rotz );
trans_d : subst ( d1, zpos, transz );
matrix_a1 : rot_theta . trans_d;

rot_alpha : subst ( %pi/2, angle, rotx );
rot_theta : subst ( th2, angle, rotz );
matrix_a2 : rot_alpha . rot_theta;

rot_theta : subst ( th3 + v2a2p, angle, rotz );
trans_a : subst ( a2, xpos, transx );
trans : subst ( v2a2, ypos, transy );
matrix_a3 : trans_a . trans . rot_theta;

trans_a : subst ( a3, xpos, transx );
trans : subst ( v3a3, ypos, transy );
matrix_a4 : trans_a . trans;

/* ******************************************
  Determine the trajectories.
*/

transmatrix : matrix_a1 . matrix_a2 . matrix_a3 . matrix_a4;

trajectory : transmatrix . endcoord;

elasticx : elasticx = part ( trajectory, 1, 1 );
elasticy : elasticy = part ( trajectory, 2, 1 );
elasticz : elasticz = part ( trajectory, 3, 1 );
v2a2 : 0;
v2a2p : 0;
v3a3 : 0;

trajectory : ev ( trajectory );

rigidx : rigidx = part ( trajectory, 1, 1 );
rigidy : rigidy = part ( trajectory, 2, 1 );
rigidz : rigidz = part ( trajectory, 3, 1 );

/* ******************************************
  Print the output to a listing file.
*/

writefile ( "error.lst" );

fortran ( elasticx );
fortran ( elasticy );
fortran ( elasticz );
fortran ( rigidx );
fortran ( rigidy );
fortran ( rigidz );
closefile ();

/* ******************************************
  Mission accomplished.
*/

quit();
/* Program name: Init.com

This command procedure routine helps to initialize the system variables for later use.

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

Setup the cross product.

\[
\text{cross\_product}(i,j,k) := [ \text{unitb} \times \text{unitz} - \text{unite} \times \text{unity}, \\
\text{unite} \times \text{unitx} - \text{unita} \times \text{unitz}, \\
\text{unita} \times \text{unity} - \text{unitb} \times \text{unitx} ]$
\]

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

Setup the gravity field vector.

\[
\text{transform\_1\_to\_2} := \begin{bmatrix}
\cos(\text{th2}(t)), & 0, & \sin(\text{th2}(t)) \\
-\sin(\text{th2}(t)), & 0, & \cos(\text{th2}(t)) \\
0, & -1, & 0
\end{bmatrix}
\]

\[
\text{gravity\_at\_1} := \begin{bmatrix}
0, & 0, & -g
\end{bmatrix}
\]

\[
\text{gravity\_at\_2} := \text{transpose}(\text{transform\_1\_to\_2} \cdot \text{gravity\_at\_1})
\]

That's it for this routine, bye...

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

/* Program name: Killvar.com

This routine helps to remove the content of the time derivatives of the generalized coordinates.

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

\[
\text{kill}\ (\text{th1d})$
\]

\[
\text{kill}\ (\text{th2d})$
\]

\[
\text{kill}\ (\text{qv2d})$
\]

\[
\text{kill}\ (\text{qv2dd})$
\]

\[
\text{kill}\ (\text{th3d})$
\]

\[
\text{kill}\ (\text{qv3d})$
\]

\[
\text{kill}\ (\text{qv3dd})$
\]

Mission accomplished, see you...

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

/* Program name: Lagrange.com

This routine helps to formulate the Lagrangian equations.

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

\[
\text{line1} := \text{diff}\ (\text{diff}\ (\text{lagrange},\text{th1d}),t) - \text{diff}\ (\text{lagrange},\text{th1}(t))$
\]

\[
\text{line2} := \text{diff}\ (\text{diff}\ (\text{lagrange},\text{th2d}),t) - \text{diff}\ (\text{lagrange},\text{th2}(t))$
\]
line3 : diff ( diff ( lagrange, th3d ), t ) - diff ( lagrange, th3(t) )$
line4 : diff ( diff ( lagrange, qv2d ), t ) - diff ( lagrange, qv2(t) )$
line5 : diff ( diff ( lagrange, qv3d ), t ) - diff ( lagrange, qv3(t) )$

/ *  *********************************************************************************/
   That's it for now, later...
*/

(run)
/*  Program name : Link1.com
   This is the procedure routine that helps to derive the equations of motion for the first link of the robot.
   *********************************************************************************/

Setup Macsyma parameter.
*/

nolabels : true$
/*  *********************************************************************************/

Call subroutine to initialize variables.
*/

batchload ( "init.com" )$
/*  *********************************************************************************/

Set up variables and parameters for link 1.
*/

batchload ( "varsetup1.com" )$
/*  *********************************************************************************/

Set up the energy equations and the Lagrangian equations for link 1.
*/

batchload ( "energy1.com" )$
/*  *********************************************************************************/

Call subroutine to extract the coefficients of link 1.
*/

batchload ( "coeff1.com" )$
/*  *********************************************************************************/

Create output listing files and save the results in a file for link 1.
*/

batchload ( "savel.com" )$
/*  *********************************************************************************/

Mission accomplished, later.
*/

quit ()$
Program name: Link2.com

This is the procedure routine that helps to derive the equations of motion for the second link of the robot.

Setup Macsyma parameter.

Call subroutine to initialize variables.

batchload ("init.com")

Set up variables and parameters for link 1 and link 2.

batchload ("varsetup1.com")
batchload ("varsetup2.com")

Set up the energy equations and the Lagrangian equations for link 2.

batchload ("energy2.com")

Call subroutine to extract the coefficients of link 2.

batchload ("coeff2.com")

Create output listing files and save the results in a file for link 2.

batchload ("save2.com")

Mission accomplished, later.

quit ()

Program name: Link3.com

This is the procedure routine that helps to derive the equations of motion for the outer-most link of the three links robot.
*/ Setup Macsyma parameter.
/*
nolabels : true$

Call subroutine to initialize variables.

batchload ( "init.com" )$
/*

Set up variables and parameters for link 1, link 2, and link 3.

batchload ( "varsetup1.com" )$
batchload ( "varsetup2.com" )$
batchload ( "varsetup3.com" )$
/*

Set up the energy equations and the lagrangian equations for
link 3.

batchload ( "energy3.com" )$
/*

Call subroutine to extract the coefficients of link 3.

batchload ( "coeff3.com" )$
/*

Create output listing files and save the results in a file for
link 3.

batchload ( "save3.com" )$
/*

Mission accomplished, later.

quit ()$

Program name : Listing.com

This command procedure routine helps to create output listings for
this project.

Print the equations into a listing file.

writefile ( filename1 )$
line1;
line2;
line3;
Print all the coefficients into a listing file.

((run)
  Program name : Pickapart.com

  This subroutine helps to breakdown the equations of the system into smaller elements.

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

  Load the data file of link 1 and breakdown the equations.
  ******************************************************************************

  loadfile ("link1.fil")$

  print_torque1_1 : torque1l1 = pickapart (line1, 1);
  print_torque2_1 : torque2l1 = pickapart (line2, 1);
  print_torque3_1 : torque3l1 = pickapart (line3, 1);

  print_coeffa1_1 : coeffa1l1 = pickapart (coeffa1, 1);
  print_coeffb1_1 : coeffb1l1 = pickapart (coeffb1, 1);
  print_coefff1_1 : coefffl1 = pickapart (coefffl, 1);

  print_coeffa2_1 : coeffa2l1 = pickapart (coeffa2, 1);
  print_coeffb2_1 : coeffb2l1 = pickapart (coeffb2, 1);
  print_coefff2_1 : coefff2l1 = pickapart (coefff2, 1);

  save ("pickapartl.fit",
      print_torque1_1, print_torque2_1, print_torque3_1,
      print_coeffa1_1, print_coeffb1_1, print_coefff1_1,
      print_coeffa2_1, print_coeffb2_1, print_coefff2_1,
      labels )$

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

  Load the data file of link 2 and breakdown the equations.
  ******************************************************************************

  loadfile ("link2.fil")$

  print_torque1_2 : torque1l2 = pickapart (line1, 1);
  print_torque2_2 : torque2l2 = pickapart (line2, 1);
  print_torque3_2 : torque3l2 = pickapart (line3, 1);
print_coeffa1_2 : coeffa1l2 = pickapart ( coeffa1, 1 );
print_coeffb1_2 : coeffb1l2 = pickapart ( coeffb1, 1 );
print_coefff1_2 : coefffl12 = pickapart ( coefffl, 1 );

print_coeffa2_2 : coeffa2l2 = pickapart ( coeffa2, 1 );
print_coeffb2_2 : coeffb2l2 = pickapart ( coeffb2, 1 );
print_coefff2_2 : coefff2l2 = pickapart ( coefff2, 1 );

save ( "pickapart2.fi",
    print_torque1_2, print_torque2_2, print_torque3_2,
    print_coeffa1_2, print_coeffb1_2, print_coefffl2,
    print_coeffa2_2, print_coeffb2_2, print_coefff2_2,
    labels );$

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

Load the data file of link 3 and breakdown the equations.

loadfile ( "link3.fi" );$

print_torque1_3 : torque1l3 = pickapart ( line1, 1 );
print_torque2_3 : torque2l3 = pickapart ( line2, 1 );
print_torque3_3 : torque3l3 = pickapart ( line3, 1 );

print_coeffa1_3 : coeffa1l3 = pickapart ( coeffa1, 1 );
print_coeffb1_3 : coeffb1l3 = pickapart ( coeffb1, 1 );
print_coefff1_3 : coefffl13 = pickapart ( coefffl, 1 );

print_coeffa2_3 : coeffa2l3 = pickapart ( coeffa2, 1 );
print_coeffb2_3 : coeffb2l3 = pickapart ( coeffb2, 1 );
print_coefff2_3 : coefff2l3 = pickapart ( coefff2, 1 );

save ( "pickapart3.fi",
    print_torque1_3, print_torque2_3, print_torque3_3,
    print_coeffa1_3, print_coeffb1_3, print_coefff1_3,
    print_coeffa2_3, print_coeffb2_3, print_coefff2_3,
    labels );$

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

Mission accomplished, exit program.

quit ();$

(run)
/* Program name : Reduce.com

This command procedure file helps to simplify and reduce the size of the equations of the system.

Call subroutines to perform the simplifications.

*/
batchload ( "reduce1.com" );$
batchload ( "reduce2.com" );$
batchload ( "reduce3.com" );$

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

Setup the elements of the solution matrix and setup the final torque equations.
*/
batchload ("sumup.com")$ 
/** ***************************************************************************/ 
        Mission accomplished, see ya... */ quit ()$

/* Program name : Reduce1.com 
   This subroutine helps to simplify all the equations of link 1. 
   *************************************************************************/ 
          Clear the memory. */ kill (all);

    /*  *************************************************************************/ 
    Load file into memory. */ loadfile ("pickapart1.fil"); 

    /*  *************************************************************************/ 
    Translate the equations into Fortran codes. */ 
writefile ("fortran1.lst"); fortran (print_torque1_1); fortran (print_torque2_1); fortran (print_torque3_1); fortran (print_coeffa1_1); fortran (print_coeffb1_1); fortran (print_coeffc1_1); fortran (print_coeffa2_1); fortran (print_coeffb2_1); fortran (print_coeffc2_1); closefile (); 

    /*  *************************************************************************/ 
    Mission accomplished, later... */ 

/* Program name : Reduce2.com 
   This subroutine helps to simplify all the equations of link 2. 
   *************************************************************************/ 
          Clear the memory. */ 
kill (all);
/* ************************************************************
   * Load file into memory.
   */
loadfile ("pickapart2.fil");
/* ************************************************************
   * Simplify the torque equations.
   */
temp : part ( e13, 2, 1 )$ kill ( e13 )$
batchload ("subreduce2.com")$
print_t_element1_2 : element ( 1, i ) = temp$
print_t_e13 : e13 = rho2e * integral ( 1 )$

temp : e14$ kill ( e14 )$
batchload ("subreduce2.com")$
print_t_e14 : e14 = temp$

temp : e15$ kill ( e15 )$
batchload ("subreduce2.com")$
print_t_e15 : e15 = temp$

temp : e16$ kill ( e16 )$
batchload ("subreduce2.com")$
print_t_e16 : e16 = temp$

temp : e17$ kill ( e17 )$
batchload ("subreduce2.com")$
print_t_e17 : e17 = temp$

temp : part ( e18, 1, 2, 1 )$ kill ( e18 )$
batchload ("subreduce2.com")$
print_t_element2_2 : element ( 2, i ) = temp$
print_t_e18 : e18 = rho2e * integral ( 2 ) / 2.0$

temp : part ( e19, 1, 2, 1 )$ kill ( e19 )$
batchload ("subreduce2.com")$
print_t_element3_2 : element ( 3, i ) = temp$
print_t_e19 : e19 = - rho2e * integral ( 3 )$

temp : part ( e20, 3, 1 )$ kill ( e20 )$
batchload ("subreduce2.com")$
print_t_element4_2 : element ( 4, i ) = temp$
print_t_e20 : e20 = g * rho2e * integral ( 4 )$

temp : e21$ kill ( e21 )$
batchload ("subreduce2.com")$
print_t_e21 : e21 = temp$

temp : e22$ kill ( e22 )$
batchload ("subreduce2.com")$
print_t_e22 : e22 = temp$

temp : e23$ kill ( e23 )$
batchload ("subreduce2.com")$
print_t_e23 : e23 = temp$
temp : e24$
kill ( e24 )$
batchload ( "subreduce2.com" )$
print_t_e24 : e24 = temp$

temp : e25$
kill ( e25 )$
batchload ( "subreduce2.com" )$
print_t_e25 : e25 = temp$
/*
 *  ************************************************************
 Simplify the coefficient equations.
 */

temp : part ( e26, 2, 1 )$
kill ( e26 )$
batchload ( "subreduce2.com" )$
print_c_element1_2 : element ( 1, i ) = temp$
print_c_e26 : e26 = rho2e * integral ( 1 )$

temp : e27$
kill ( e27 )$
batchload ( "subreduce2.com" )$
print_c_e27 : e27 = temp$
/*
 *  ****************************************************************
 temp : part ( e28, 3, 1 )$
kill ( e28 )$
batchload ( "subreduce2.com" )$
print_c_element2_2 : element ( 2, i ) = temp$
print_c_e28 : e28 = eiv2 * qv2 * integral ( 2 )$

temp : part ( e29, 1, 2, 1 )$
kill ( e29 )$
batchload ( "subreduce2.com" )$
print_c_element3_2 : element ( 3, i ) = temp$
print_c_e29 : e29 = - rho2e * integral ( 3 ) / 2.0$

temp : part ( e30, 3, 1 )$
kill ( e30 )$
batchload ( "subreduce2.com" )$
print_c_element4_2 : element ( 4, i ) = temp$
print_c_e30 : e30 = rho2e * th2dd * integral ( 4 )$

temp : part ( e31, 4, 1 )$
kill ( e31 )$
batchload ( "subreduce2.com" )$
print_c_element5_2 : element ( 5, i ) = temp$
print_c_e31 : e31 = g * rho2e * costh2 * integral ( 5 )$

temp : e32$
kill ( e32 )$
batchload ( "subreduce2.com" )$
print_c_e32 : e32 = temp$

temp : e33$
kill ( e33 )$
batchload ( "subreduce2.com" )$
print_c_e33 : e33 = temp$
/*
 *  ****************************************************************
 Translate the equations into Fortran codes.
 */

writefile ( "fortran2.lst" );
fortran ( print_t_element1_2 );
fortran ( print_t_element2_2 );
fortran ( print_t_element3_2 );
fortran ( print_t_element4_2 );
fortran ( print_t_e13 );
fortran ( print_t_e14 );
fortran ( print_t_e15 );
fortran ( print_t_e16 );
fortran ( print_t_e17 );
fortran ( print_t_e18 );
fortran ( print_t_e19 );
fortran ( print_t_e20 );
fortran ( print_t_e21 );
fortran ( print_t_e22 );
fortran ( print_t_e23 );
fortran ( print_t_e24 );
fortran ( print_t_e25 );
fortran ( print_torque1_2 );
fortran ( print_torque2_2 );
fortran ( print_torque3_2 );
fortran ( print_c_element1_2 );
fortran ( print_c_element2_2 );
fortran ( print_c_element3_2 );
fortran ( print_c_element4_2 );
fortran ( print_c_element5_2 );
fortran ( print_c_e26 );
fortran ( print_c_e27 );
fortran ( print_c_e28 );
fortran ( print_c_e29 );
fortran ( print_c_e30 );
fortran ( print_c_e31 );
fortran ( print_c_e32 );
fortran ( print_c_e33 );
fortran ( print_coeff_a1_2 );
fortran ( print_coeff_b1_2 );
fortran ( print_coeff_f1_2 );
fortran ( print_coeff_f2_2 );
fortran ( print_coeff_a2_2 );
fortran ( print_coeff_b2_2 );
closefile ( );
/
 Mission accomplished, later...
/

/*  Program name : Reduce3.com
   This subroutine helps to simplify all the equations of link 3.
   ***************************************************************************/
   Clear the memory.
   */
   kill ( all );
   /*  **************************************************************************/
   Load file into memory.
   */
   loadfile ( "pickapart3.fil" );
   /*  **************************************************************************/
   Simplify the torque equations.
   */
/*

temp : part ( e38, 2, 1 )$ 
kill ( e38 )$
batchload ( "subreduce2.com" )$
batchload ( "subreduce3.com" )$
print_t_element1_3 : element ( 1, i ) = temp$
print_t_e38 : e38 = rho3 * integral ( 1 )$

 temp : e39$
kill ( e39 )$
batchload ( "subreduce2.com" )$
batchload ( "subreduce3.com" )$
print_t_e39 : e39 = temp$

 temp : e40$
kill ( e40 )$
batchload ( "subreduce2.com" )$
batchload ( "subreduce3.com" )$
print_t_e40 : e40 = temp$

*/
print_t_element6_3 : element ( 6, i ) = temp$
print_t_element6_4 : e47 = - rho3 * integral ( 6 ) / 2.0$

temp : part ( e48, 3, 1 )$
kill ( e48 )$
batchload ( "subreduce2.com" )$
batchload ( "subreduce3.com" )$
print_t_element7_3 : element ( 7, i ) = temp$
print_t_element7_4 : e48 = g * rho3 * integral ( 7 )$

temp : e49$
kll ( e49 )$
batchload ( "subreduce2.com" )$
batchload ( "subreduce3.com" )$
print_t_element7_4 : e49 = temp$

/*  *************************************************************/
Simplify the coefficient equations.
*/

temp : part ( e50, 1, 2, 1 )$
kll ( e50 )$
batchload ( "subreduce2.com" )$
batchload ( "subreduce3.com" )$
print_c_element1_3 : element ( 1, i ) = temp$
print_c_element1_4 : e51 = rho3 * integral ( 1 ) / 2.0$

temp : e52$
kll ( e52 )$
batchload ( "subreduce2.com" )$
batchload ( "subreduce3.com" )$
print_c_element1_4 : e52 = temp$

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

temp : part ( e53, 1, 2, 1 )$
kll ( e53 )$
batchload ( "subreduce2.com" )$
batchload ( "subreduce3.com" )$
print_c_element2_3 : element ( 2, i ) = temp$
print_c_element2_4 : e53 = rho3 * integral ( 2 ) / 2.0$

temp : e54$
kll ( e54 )$
batchload ( "subreduce2.com" )$
batchload ( "subreduce3.com" )$
print_c_element2_4 : e54 = temp$

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

temp : part ( e55, 1, 2, 1 )$
kll ( e55 )$
batchload ( "subreduce2.com" )$
batchload ( "subreduce3.com" )$
print_c_element3_3 : element ( 3, i ) = temp$
print_c_element3_4 : e55 = rho3 * integral ( 3 ) / 2.0$

temp : part ( e56, 1, 2, 1 )$
kll ( e56 )$
batchload ( "subreduce2.com" )$
batchload ( "subreduce3.com" )$
print_c_element4_3 : element ( 4, i ) = temp
print_c_e56 : e56 = - rho3 * integral ( 4 ) / 2.0$

.temp : part ( e57, 3, 1 )$
.kill ( e57 )$
.batchload ( "subreduce2.com" )$
.batchload ( "subreduce3.com" )$
.print_c_element5_3 : element ( 5, i ) = temp$
.print_c_e57 : e57 = g * rho3 * integral ( 5 )$

.temp : e58$
.kill ( e58 )$
.batchload ( "subreduce2.com" )$
.batchload ( "subreduce3.com" )$
.print_c_element5_3 : element ( 5, i ) = temp$
.print_c_e58 : e58 = e57$

/* ................................................................................
*/

.temp : part ( e60, 1, 2, 1 )$
.kill ( e60 )$
.batchload ( "subreduce2.com" )$
.batchload ( "subreduce3.com" )$
.print_c_element6_3 : element ( 6, i ) = temp$
.print_c_e60 : e60 = rho3 * integral ( 4 ) / 2.0$

.temp : e61$
.kill ( e61 )$
.batchload ( "subreduce2.com" )$
.batchload ( "subreduce3.com" )$
.print_c_element6_3 : element ( 6, i ) = temp$
.print_c_e61 : e61 = e60$

/* ................................................................................
*/
batchload ( "subreduce3.com" )
print_c_e66 : e66 = temp

temp : e67
kill ( e67 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_e67 : e67 = temp

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

Translate the equations into Fortran codes.
*/

writefile ( "fortran3.lst" );

fortran ( print_t_element1_3 );
fortran ( print_t_element2_3 );
fortran ( print_t_element3_3 );
fortran ( print_t_element4_3 );
fortran ( print_t_element5_3 );
fortran ( print_t_element6_3 );
fortran ( print_t_element7_3 );
fortran ( print_t_element8_3 );
fortran ( print_t_element9_3 );
fortran ( print_t_element10_3 );
fortran ( print_t_element11_3 );
fortran ( print_t_element12_3 );
fortran ( print_t_element13_3 );
fortran ( print_t_element14_3 );
fortran ( print_t_element15_3 );
fortran ( print_t_element16_3 );
fortran ( print_t_element17_3 );
fortran ( print_t_element18_3 );
fortran ( print_t_element19_3 );
fortran ( print_t_element20_3 );
fortran ( print_t_element21_3 );
fortran ( print_t_element22_3 );
fortran ( print_t_element23_3 );
fortran ( print_t_element24_3 );
fortran ( print_t_element25_3 );
fortran ( print_t_element26_3 );
fortran ( print_t_element27_3 );
fortran ( print_t_element28_3 );
fortran ( print_t_element29_3 );
fortran ( print_t_element30_3 );
fortran ( print_t_element31_3 );
fortran ( print_t_element32_3 );
fortran ( print_t_element33_3 );
fortran ( print_t_element34_3 );
fortran ( print_t_element35_3 );
fortran ( print_t_element36_3 );
fortran ( print_t_element37_3 );
fortran ( print_t_element38_3 );
fortran ( print_t_element39_3 );
fortran ( print_t_element40_3 );
fortran ( print_t_element41_3 );
fortran ( print_t_element42_3 );
fortran ( print_t_element43_3 );
fortran ( print_t_element44_3 );
fortran ( print_t_element45_3 );
fortran ( print_t_element46_3 );
fortran ( print_t_element47_3 );
fortran ( print_t_element48_3 );
fortran ( print_t_element49_3 );
fortran ( print_t_element50_3 );
fortran ( print_t_element51_3 );
fortran ( print_t_element52_3 );
fortran ( print_t_element53_3 );
fortran ( print_t_element54_3 );
fortran ( print_t_element55_3 );
fortran ( print_t_element56_3 );
fortran ( print_t_element57_3 );
fortran ( print_t_element58_3 );
fortran ( print_t_element59_3 );
fortran ( print_t_element60_3 );
fortran ( print_t_element61_3 );
fortran ( print_t_element62_3 );
fortran ( print_t_element63_3 );
fortran ( print_t_element64_3 );
fortran ( print_t_element65_3 );
fortran ( print_t_element66_3 );
fortran ( print_t_element67_3 );
fortran ( print_coeffa1_3 );
fortran ( print_coeffa1_3 );
fortran ( print_coeffb1_3 );
fortran ( print_coeffb1_3 );
fortran ( print_coeffb2_3 );
fortran ( print_coefff2_3 );
closefile ();
/* ***************************************************************
   Mission accomplished, later...
*/

/* Program name : Save1.com
   This command procedure file helps to create output listings and
   saves the results of link 1.
   ***************************************************************
*/
filename1 : "link1.lst"
filename2 : "coeff1.lst"
batch ( "listing.com" )
save ( "link1.fil",
      coeffa1, coeffb1, coefff1, coeffa2, coeffb2, coefff2,
      line1, line2, line3, line4, line5 )
/* ***************************************************************
   That's it folks...
*/

/* Program name : Save2.com
   This command procedure file helps to create output listings and
   saves the results of link 2.
   ***************************************************************
*/
filename1 : "link2.lst"
filename2 : "coeff2.lst"
batch ( "listing.com" )
save ( "link2.fil",
      coeffa1, coeffb1, coefff1, coeffa2, coeffb2, coefff2,
      line1, line2, line3, line4, line5 )
/* ***************************************************************
   That's it folks...
*/

/* Program name : Save3.com
   This command procedure file helps to create output listings and
   saves the results of link 3.
   ***************************************************************
*/
filename1 : "link3.lst"
filename2 : "coeff3.lst"

batch ( "listing.com" )

save ( "link3.fil",
coeffa1, coeffb1, coefff1, coeffa2, coeffb2, coefff2,
line1, line2, line3, line4, line5 )

/*/ *******************************************************************
 That's it folks...
 */

/*/ Program name : Simplify1.com

This subroutine is targeted at the equations of link 1. It helps to replace the derivatives of the time-dependent generalized coordinate with their variable names.

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

Mission accomplished, see you...
 */

/*/ Program name : Simplify2.com

This subroutine is targeted at the equations of link 2. It helps to replace the derivatives of the time-dependent generalized coordinate with their variable names.

*****************************************************************************
Program name: Simplify3.com

This subroutine is targeted at the equations of link 3. It helps to replace the derivatives of the time-dependent generalized coordinate with their variable names.

Mission accomplished, see you...

*/
/* Program name : Subreduce2.com

This is the command procedure routine that helps to reduce and simplify the current link 2's equation in equation 'temp'.

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

thldsq, th1dA 2, temp )$
a2sq, a2^2, temp )$
a2rsq, a2r^2, temp )$
a2rcu, a2r^3, temp )$
costh2, costh2(t), temp )$
costh2sq, costh2A 2, temp )$
sinth2, sin ( th2(t) ), temp )$
sinth2sq, sinth2A 2, temp )$
cos2, cos ( th2(t) ), temp )$
cos2sq, cos2A 2, temp )$
sin2, sin ( th2(t) ), temp )$
sin2sq, sin2A 2, temp )$
costh2dsq, costh2dA 2, temp )$
qv2, qv2(t), temp )$
qv2sq, qv2A 2, temp )$
qv2dsq, qv2dA 2, temp )$
qv2ppsq(i), ( diff ( gv2(u2), u2, 2 ) )^2, temp )$
qv2sq(i), qv2(i)A 2, temp )$
qv2a2sq, qv2a2A 2, temp )$
qv2a2psq, qv2a2pA 2, temp )$
u2(i), u2, temp )$
thd2, thd2^2, temp )$
qv3, qv3(t), temp )$
qv3sq, qv3^2, temp )$
qv3dsq, qv3d^2, temp )$
qv3pqsq(i), ( diff ( gv3(u3), u3, 2 ) )^2, temp )$
qv3sq(i), qv3(i)A 2, temp )$
qv3a3sq, qv3a3A 2, temp )$
u3(i), u3, temp )$
link3sub(1), ( cospsi * qv3 * gv3(i) + sinpsi * u3(i) ), temp )$
link3sub(2), ( cospsi * u3(i) - sinpsi * qv3 * gv3(i) ), temp )

Mission accomplished, see you...

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

/* Program name : Subreduce3.com

This is the command procedure routine that helps to reduce and simplify the current link 3's equation in equation 'temp'.

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

cospsi, cos ( th3(t) + gv2a2p * qv2 ), temp )$
欢喜, cospsiA 2, temp )$
sinpsi, sin ( th3(t) + gv2a2p * qv2 ), temp )$
sinpsiA 2, temp )$
chi, th3CtJ + th2(t) + gv2a2p * qv2, temp )$
chid, th3d + th2d + gv2a2p * qv2d, temp )$
chidd, th3dd + th2dd + gv2a2p * qv2dd, temp )$
qv3, qv3(t), temp )$
qv3sq, qv3^2, temp )$
qv3dsq, qv3d^2, temp )$
qv3pqsq(i), ( diff ( gv3(u3), u3, 2 ) )^2, temp )$
qv3sq(i), qv3(i)A 2, temp )$
qv3a3sq, qv3a3A 2, temp )$
u3(i), u3, temp )$
link3sub(1), ( cospsi * qv3 * gv3(i) + sinpsi * u3(i) ), temp )$
link3sub(2), ( cospsi * u3(i) - sinpsi * qv3 * gv3(i) ), temp )$

Mission accomplished, see you...
temp : subst ( 'link3sub(3)',
    ( -psid * sinpsi * u3(i) + sinpsi * gv3(i) * qv3d + psid * cospsi * gv3(i) * qv3 ), temp )$

temp : subst ( 'link3sub(4)',
    ( psid * cospsi * u3(i) + cospsi * gv3(i) * qv3d + psid * sinpsi * gv3(i) * qv3 ), temp )$

temp : subst ( 'link3sub(5)',
    ( -psid * cospsi * u3(i) - cospsi * gv3(i) * qv3d + psid * sinpsi * gv3(i) * qv3 ), temp )$

temp : subst ( 'link3sub(6)',
    ( coschi * u3(i) - sinchi * qv3 * gv3(i) ), temp )$

temp : subst ( 'link3sub(7)',
    ( -coschi * u3(i) + gv2a2 * qv2 * sinth2 + gv3(i) * qv3 * sinchi - a2 * costh2 ), temp )$

temp : subst ( 'link3sub(1)',
    ( -cospsi * qv3 * gv3a3 + sinpsi * a3 ), temp )$

temp : subst ( 'link3sub(2)',
    ( -cospsi * a3 - sinpsi * qv3 * gv3a3 ), temp )$

temp : subst ( 'link3sub(3)',
    ( psid * sinpsi * a3 + sinpsi * gv3a3 * qv3d + psid * cospsi * gv3a3 * qv3 ), temp )$

temp : subst ( 'link3sub(4)',
    ( -psid * cospsi * a3 - cospsi * gv3a3 * qv3d + psid * sinpsi * gv3a3 * qv3 ), temp )$

temp : subst ( 'link3sub(5)',
    ( -coschi * a3 - sinchi * qv3 * gv3a3 ), temp )$

temp : subst ( 'link3sub(6)',
    ( cos2a2p * link3sub(2) + gv2a2 ), temp )$

temp : subst ( 'link3sub(7)',
    ( -coschi * a3 - gv2a2 * qv2 * sinth2 - gv3a3 * qv3 * sinchi - a2 * costh2 ), temp )$

temp : subst ( 'link3sub(8)',
    ( -cospsi * u3(i) + sinpsi * gv3(i) * qv3 ), temp )$

temp : subst ( 'link3sub(9)',
    ( -psid * sinpsi * u3(i) - sinpsi * gv3(i) * qv3d + psid * cospsi * gv3(i) * qv3 ), temp )$

temp : subst ( 'link3sub(10)',
    ( -psid * cospsi * u3(i) + cospsi * gv3(i) * qv3d + psid * sinpsi * gv3(i) * qv3 ), temp )$

temp : subst ( 'link3sub(11)',
    ( coschi * u3(i) - sinchi * qv3 * gv3(i) ), temp )$

temp : subst ( 'link3sub(12)',
    ( -coschi * u3(i) + sinchi * qv3 * gv3(i) ), temp )$

temp : subst ( 'link3sub(13)',
    ( -coschi * a3 + sinchi * qv3 * gv3a3 ), temp )$

temp : subst ( 'link3sub(14)',
    ( coschi * a3 - sinchi * qv3 * gv3a3 ), temp )$

temp : subst ( 'link3sub(15)',
    ( cos2a2p * link3sub(2) + gv2a2 ), temp )$

temp : subst ( 'link3sub(16)',
    ( coschi * a3 - gv2a2 * qv2 * sinth2 - gv3a3 * qv3 * sinchi - a2 * costh2 ), temp )$

temp : subst ( 'link3sub(17)',
    ( -coschi * a3 + gv2a2 * qv2 * sinth2 - gv3a3 * qv3 * sinchi + a2 * costh2 ), temp )$

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

Mission accomplished, see you...
*/

/* Program name: Subsimp1.com

This subroutine helps to replace the time-dependent generalized coordinate with their variable names. The replacement is for the current link 1's equation stored in 'temp'. */
temp : subst ( th1d, diff ( th1(t), t ), temp )$
temp : subst ( th1dd, diff ( th1(t), t, 2 ), temp )$

Mission accomplished, see you...

/* Program name : Subsimp2.com
This subroutine helps to replace the time-dependent generalized coordinate with their variable names. The replacement is for the current link 2‘s equation stored in ‘temp’.

***************************************************************************
*/
temp : subst ( th2d, diff ( th2(t), t ), temp )$
temp : subst ( th2dd, diff ( th2(t), t, 2 ), temp )$
temp : subst ( qv2d, diff ( qv2(t), t ), temp )$
temp : subst ( qv2dd, diff ( qv2(t), t, 2 ), temp )$

Mission accomplished, see you...

/* Program name : Subsimp3.com
This subroutine helps to replace the time-dependent generalized coordinate with their variable names. The replacement is for the current link 3‘s equation stored in ‘temp’.

***************************************************************************
*/
temp : subst ( th3d, diff ( th3(t), t ), temp )$
temp : subst ( th3dd, diff ( th3(t), t, 2 ), temp )$
temp : subst ( qv3d, diff ( qv3(t), t ), temp )$
temp : subst ( qv3dd, diff ( qv3(t), t, 2 ), temp )$

Mission accomplished, see you...

/* Program name : Sumup.com
This subroutine helps to setup the elements of the solution matrix and sum up the torque generated from each link.

***************************************************************************
Sum up the torques.
*/
torque_1 : torque ( 1 ) = torque1l1 + torque1l2 + torque1l3$
torque_2 : torque ( 2 ) = torque2l1 + torque2l2 + torque2l3$
torque_3 : torque ( 3 ) = torque3l1 + torque3l2 + torque3l3$

/* Sum up the coefficients of each link. */
coeff_a1 : a ( 1, 1 ) = coeffa111 + coeffa1l2 + coeffa1l3$
coeff_b1 : a ( 1, 2 ) = coeffb111 + coeffb1l2 + coeffb1l3$
coeff_a2 : a ( 2, 1 ) = coeffa211 + coeffa2l2 + coeffa2l3$
coeff_b2 : a ( 2, 2 ) = coeffb211 + coeffb2l2 + coeffb2l3$
coeff_f1 : b ( 1 ) = -( coefff111 + coefff1l2 + coefff1l3 )$
coeff_f2 : b ( 2 ) = -( coefff211 + coefff2l2 + coefff2l3 )$

/* Translate the equations into Fortran codes. */
writefile ( "fortran.lst" );
fortran ( torque_1 );
fortran ( torque_2 );
fortran ( torque_3 );
fortran ( coeff_a1 );
fortran ( coeff_b1 );
fortran ( coeff_a2 );
fortran ( coeff_b2 );
fortran ( coeff_f1 );
fortran ( coeff_f2 );
closefile ( );

Done. Bye...

/* Program name : Varsetup1.com
This routine helps to setup all the variables and parameters for
link 1.

Setup the general variables for link 1. */

th1d : diff ( th1(t), t )$
omega_1_at_1 : matrix ( [ 0 ], [ 0 ], [ th1d ] )$
omega_1 : transpose ( transform_1_to_2 . omega_1_at_1 )$
velocity_at_0(i,j,k) := [ 0, 0, 0 ]$

/* Set up the displacement vector and the velocity vector for link 1. */
vector_1_at_1 : matrix ( [ 0 ], [ 0 ], [ -w ] )$
vector_1 : transpose ( transform_1_to_2 . vector_1_at_1 )$
gdotvector_1 : trigreduce ( gravity_at_2 . vector_1 )$

omega_1_x_vector_1 : cross_product(i,j,k)$
omega_1_x_vector_1 : subst ( part(omega_1,1,1), unita, omega_1_x_vector_1 )$
omega_1_x_vector_1 : subst ( part(omega_1,1,2), unitb, omega_1_x_vector_1 )$
omega_1_x_vector_1 : subst ( part(omega_1,1,3), unitc, omega_1_x_vector_1 )$
omega_1_x_vector_1 : subst ( part(vector_1,1,1), unitx, omega_1_x_vector_1 )$
omega_1_x_vector_1 : subst ( part(vector_1,1,2), unity, omega_1_x_vector_1 )$
omega_1_x_vector_1 : subst ( part(vector_1,1,3), unitz, omega_1_x_vector_1 )$

vector_1d : velocity_at_0 ( i,j,k )  +  omega_1_x_vector_1$
vector_1dsq : vector_1d ,~vector_1d$

That's it folks...

*/

/* Program name : Varsetup2.com

This routine helps to set up all the variables and parameters for link 2.
******************************************************************************
Set up the general variables for link 2.
******************************************************************************
*/

th2d : diff ( th2(t), t )$
v2 : gv2(u2) * qv2(t)$
v2d : diff ( v2, t )$
v2p : diff ( v2, u2 )$
v2pp : diff ( v2p, u2 )$
v2_at_a2 : g2a2 * qv2(t)$
v2_at_a2d : diff ( v2_at_a2, t )$
v2_at_a2p : g2a2p * qv2(t)$
v2_at_a2pd : diff ( v2_at_a2p, t )$
qv2d : diff ( qv2(t), t )$

omega_2_at_2 : matrix ( [ 0, 0, th2d ] )$
omega_2 : omega_1 + omega_2_at_2$
velocity_at_1 : subst ( d1, w1, vector_1d )$

Set up the displacement vector and the velocity vector for the rigid portion for link 2.
******************************************************************************

vector_2r : matrix ( [ u2, 0, 0 ] )$
gdotvector_2r : gravity_at_2 . vector_2r$

omega_2_x_vector_2r : cross_product(i,j,k)$
omega_2_x_vector_2r : subst ( part(omega_2,1,1), unita, omega_2_x_vector_2r )$
omega_2_x_vector_2r : subst ( part(omega_2,1,2), unitb, omega_2_x_vector_2r )$
omega_2_x_vector_2r : subst ( part(omega_2,1,3), unitc, omega_2_x_vector_2r )$
omega_2_x_vector_2r : subst ( part(vector_2r,1,1), unitx, omega_2_x_vector_2r )
omega_2_x_vector_2r : subst ( part(vector_2r,1,2), unity, omega_2_x_vector_2r )
omega_2_x_vector_2r : subst ( part(vector_2r,1,3), unitz, omega_2_x_vector_2r )

vector_2rd : velocity_at_1 + omega_2_x_vector_2r
vector_2rdsq : factor ( vector_2rd . vector_2rd )

/* ************** Set up the displacement vector and the velocity vector for the elastic portion for link 2. */

vector_2e : matrix ( [ u2, v2, 0 ] )
vector_2e_at_a2 : matrix ( [ a2, v2_at_a2, 0 ] )
gdotvector_2e : factor ( gravity_at_2 . vector_2e )
gdotvector_2e_at_a2 : factor ( gravity_at_2 . vector_2e_at_a2 )

omega_2_x_vector_2e : cross_product(i,j,k)
omega_2_x_vector_2e : subst ( part(omega_2,1,1), unita, omega_2_x_vector_2e )
omega_2_x_vector_2e : subst ( part(omega_2,1,2), unitb, omega_2_x_vector_2e )
omega_2_x_vector_2e : subst ( part(omega_2,1,3), unitc, omega_2_x_vector_2e )

projection_v2(i,j,k) := [ 0 , 1 , 0 ]

vector_2ed : velocity_at_1 + omega_2_x_vector_2e + v2d * projection_v2(i,j,k)
vector_2edsq : vector_2ed . vector_2ed

vector_2e_at_a2d : velocity_at_1 + omega_2_x_vector_2e_at_a2 + v2_at_a2d * projection_v2(i,j,k)
vector_2e_at_a2dsq : vector_2e_at_a2d . vector_2e_at_a2d

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

That's it folks...
program name: Varsetup3.com

This routine helps to set up all the variables and parameters for link 3.

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

Set up the general variables for link 3.

th3d = diff (th3(t), t)
psi = th3(t) + v2_at_a2p
v3 = gv3(u3) * qv3(t)
v3d = diff (v3, t)
v3p = diff (v3, u3)
v3pp = diff (v3p, u3)
v3_at_a3 = gv3a3 * qv3(t)
v3_at_a3d = diff (v3_at_a3, t)
qv3d = diff (qv3(t), t)
transform_3_to_2 = matrix ( [cos(psi), -sin(psi), 0],
                        [sin(psi), cos(psi), 0],
                        [0, 0, 1])
omega_2_at_a2 = omega_2 + matrix ( [0, 0, v2_at_a2p] )
omega_3_at_3 = matrix ( [0], [0], [th3d])
omega_3 = omega_2_at_a2 + transpose(transform_3_to_2 . omega_3_at_3)
velocity_at_2 = vector_2e_at_a2d

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

Set up the displacement vector and the velocity vector for link 3.

vector_3_at_3 = matrix ( [u3], [v3], [0])
vector_3 = transpose(transform_3_to_2 . vector_3_at_3)
vector_3_at_a3_at_3 = matrix ( [a3], [v3_at_a3], [0])
vector_3_at_a3 = transpose(transform_3_to_2 . vector_3_at_a3_at_3)
gdotvector_3 = gravity_at_2 . (vector_2e_at_a2 + vector_3)
gdotvector_3 = factor (trigreduce (expand(gdotvector_3)))
gdotvector_3_at_a3 = gravity_at_2 . (vector_2e_at_a2 + vector_3_at_a3)
gdotvector_3_at_a3 = factor (trigreduce (expand(gdotvector_3_at_a3)))

omega_3_x_vector_3 = cross_product(i, j, k)
omega_3_x_vector_3 = subst (part(omega_3, 1, 1), unita,
                        omega_3_x_vector_3)
omega_3_x_vector_3 = subst (part(omega_3, 1, 2), unitb,
                        omega_3_x_vector_3)
omega_3_x_vector_3 = subst (part(omega_3, 1, 3), unitc,
                        omega_3_x_vector_3)
omega_3_x_vector_3 = subst (part(omega_3, 1, 1), unitx,
                        omega_3_x_vector_3)
omega_3_x_vector_3 = subst (part(omega_3, 1, 2), unity,
                        omega_3_x_vector_3)
omega_3_x_vector_3 = subst (part(omega_3, 1, 3), unitz,
                        omega_3_x_vector_3)
part1 = part (omega_3_x_vector_3, 1)
part2 = part (omega_3_x_vector_3, 2)
part3 = part (omega_3_x_vector_3, 3)
part3 = factor (trigreduce (part3))
omega_3_x_vector_3 : [ part1, part2, part3 ]$

omega_3_x_vector_3_at_a3 : cross_product(i,j,k)$

omega_3_x_vector_3_at_a3 : subst (part(omega_3,1,1), unita, omega_3_x_vector_3_at_a3)$

omega_3_x_vector_3_at_a3 : subst (part(omega_3,1,2), unitb, omega_3_x_vector_3_at_a3)$

omega_3_x_vector_3_at_a3 : subst (part(omega_3,1,3), unitc, omega_3_x_vector_3_at_a3)$

omega_3_x_vector_3_at_a3 : subst (part(vector_3_at_a3,1,1), unitx, omega_3_x_vector_3_at_a3)$

omega_3_x_vector_3_at_a3 : subst (part(vector_3_at_a3,1,2), unity, omega_3_x_vector_3_at_a3)$

omega_3_x_vector_3_at_a3 : subst (part(vector_3_at_a3,1,3), unitz, omega_3_x_vector_3_at_a3)$

part1 : part (omega_3_x_vector_3_at_a3, 1)$

part2 : part (omega_3_x_vector_3_at_a3, 2)$

part3 : part (omega_3_x_vector_3_at_a3, 3)$

part3 : factor (trigreduce (part3))$

omega_3_x_vector_3_at_a3 : [ part1, part2, part3 ]$

temp : transform_3_to_2 . matrix ([0], [1], [0])$

temp : transpose (temp)$

part1 : part (temp, 1)$

part2 : part (temp, 2)$

part3 : part (temp, 3)$

projection_v3 : cross_product(i,j,k)$

projection_v3 : [ part1, part2, part3 ]$

vector_3d : velocity_at_2 + omega_3_x_vector_3 + v3d * projection_v3$

vector_3dsq : vector_3d . vector_3d$

vector_3_at_a3d : velocity_at_2 + omega_3_x_vector_3_at_a3 + v3_at_a3d * projection_v3$

vector_3_at_a3dsq : vector_3_at_a3d . vector_3_at_a3d$

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

That's it folks...

*/
APPENDIX F

MACSYMA PROGRAM LISTINGS OF THE UNLV–ARO LIGHT–WEIGHT ROBOT
Program name: Coeff.com

This command procedure routine helps to determine the coefficients of the differential equations.

******************************************************************************
Extract the 'F' coefficient.
******************************************************************************
qv2dd : 0$
qw2dd : 0$
qv3dd : 0$
qw3dd : 0$
qvh3dd : 0$
coefffl : ev ( line4 )$
coefff2 : ev ( line5 )$
coefff3 : ev ( line5 )$
coefff4 : ev ( line7 )$
coefff5 : ev ( line8 )$
******************************************************************************
Set variables to zero for coefficients of [ D ].
******************************************************************************

g : 0$
th1d : 0$

th1dd : 0$
eiv2 : 0$
eiw2 : 0$

th2d : 0$

th2dd : 0$
qv2d : 0$
qw2d : 0$
eiv3 : 0$
eiw3 : 0$

th3d : 0$

th3dd : 0$
qv3d : 0$
qw3d : 0$
eih3 : 0$

qvh3d : 0$
******************************************************************************
Extract the 'A' coefficient. That is, 'A' * qv2.
******************************************************************************
qv2dd : 1$
coeffa1 : ev ( line4 )$
coeffa2 : ev ( line5 )$
coeffa3 : ev ( line5 )$
coeffa4 : ev ( line7 )$
coeffa5 : ev ( line8 )$
******************************************************************************
Extract the 'B' coefficient. That is, 'B' * qw2.
******************************************************************************
qv2dd : 0$
qw2dd : 1$
coeffb1 : ev ( line4 )$
coeffb2 : ev ( line5 )$
coeffb3 : ev ( line5 )$
coeffb4 : ev ( line7 )
coeffb5 : ev ( line8 )$

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

  Extract the 'C' coefficient. That is, 'C' * qv3.

*/
qv2dd : 0$
qv3dd : 1$
coeffc1 : ev ( line4 )$
coeffc2 : ev ( line5 )$
coeffc3 : ev ( line6 )$
coeffc4 : ev ( line7 )$
coeffc5 : ev ( line8 )$

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

  Extract the 'D' coefficient. That is, 'D' * qw3.

*/
qv3dd : 0$
qv4dd : 1$
coeffd1 : ev ( line4 )$
coeffd2 : ev ( line5 )$
coeffd3 : ev ( line6 )$
coeffd4 : ev ( line7 )$
coeffd5 : ev ( line8 )$

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

  Extract the 'E' coefficient. That is, 'E' * qvh3.

*/
qv3dd : 0$
qv4dd : 1$
coeffe1 : ev ( line4 )$
coeffe2 : ev ( line5 )$
coeffe3 : ev ( line6 )$
coeffe4 : ev ( line7 )$
coeffe5 : ev ( line8 )$

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

  Done, see you later !!!

*/

/*  Program name : Coeff1.com

  This subroutine calls other subroutines to extract the coefficients from the equations of motion for link 1.

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

  Remove the content of the time-dependent generalized coordinate.

*/
batchload ( "killvar.com" )$

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

  Replace the derivatives of the time-dependent generalized coordinate with their variable names.

*/
batchload ("simplify1.com")$

Call subroutine to extract the coefficients of link 1.

batchload ("coeff.com")$

That's it folks...

Program name: Coeff2.com

This procedure file calls other procedure files to extract the
coefficient of the system's equations for link 2.

Remove the content of the time-dependent generalize coordinate.

batchload ("killvar.com")$

Replace the derivatives of the time-dependent generalize coordinate
with their variable names.

batchload ("simplify2.com")$

Call subroutine to extract the coefficients of link 2.

batchload ("coeff.com")$

That's it folks...

Program name: Coeff3.com

This procedure file calls other procedure files to extract the
coefficient of the system's equations for link 3.

Remove the content of the time-dependent generalize coordinate.

batchload ("killvar.com")$

Replace the derivatives of the time-dependent generalize coordinate
with their variable names.
batchload ( "simplify3.com" )$

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

Call subroutine to extract the coefficients of link 3.

*/

batchload ( "coeff.com" )$

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

That's it folks...

*/

/* Program name : Coeffc.com

This procedure file calls other procedure files to extract the coefficient of the system's equations for the connecting link.

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

Remove the content of the time-dependent generalize coordinate.

*/

batchload ( "killvar.com" )$

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

Replace the derivatives of the time-dependent generalize coordinate with their variable names.

*/

batchload ( "simplifyc.com" )$

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

Call subroutine to extract the coefficients of link c.

*/

batchload ( "coeff.com" )$

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

That's it folks...

*/

/* Program name : Coeffh3.com

This procedure file calls other procedure files to extract the coefficient of the system's equations for the hydraulic cylinder.

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

Remove the content of the time-dependent generalize coordinate.

*/

batchload ( "killvar.com" )$

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

Replace the derivatives of the time-dependent generalize coordinate with their variable names.

*/
Call subroutine to extract the coefficients of link h3.

That's it folks...

Program name: Error.com

This command procedure routine helps to determine the trajectories of the rigid and flexible robots.

Set up the rotation and translation variables.

Set up the matrix dot product of each link.

batchload ("simplifyh3.com")$
/* 琎pageNumberaneous***
  Call subroutine to extract the coefficients of link h3.
*/
batchload ("coeff.com")$
/* 琎pageNumberaneous***
That's it folks...

(program)
/* Program name: Error.com

This command procedure routine helps to determine the trajectories of the rigid and flexible robots.

Set up the rotation and translation variables.

Set up the matrix dot product of each link.

rot_x : subst ( %pi/2, angle, rot_x);
rot_z : subst ( th2, angle, rot_z);
matrix_a2 : rot_x . rot_z;
trans_x : subst ( a2, xpos, trans_x);
trans_y : subst ( v2a2, ypos, trans_y);
trans_z : subst ( w2a2, zpos, trans_z);
rot_y : subst ( w2a2p, angle, rot_y);
rot_z : subst ( th3 + v2a2p, angle, rot_z);
matrix_a3 : trans_x . trans_y . trans_z . rot_y . rot_z;
trans_x : subst ( a3, xpos, trans_x);
trans_y : subst ( v5a3, ypos, trans_y);
trans_z : subst ( w5a3, zpos, trans_z);
matrix_a4 : trans_x . trans_y . trans_z;
Determine the trajectories.

transmatrix : matrix_a1 . matrix_a2 . matrix_a3 . matrix_a4;
trajectory : transmatrix . endcoord;
estaticx : elastix = part ( trajectory, 1, 1 );
estaticy : elasticy = part ( trajectory, 2, 1 );
estaticz : elastiz = part ( trajectory, 3, 1 );
v2a2 : 0;
v2a2p : 0;
w2a2 : 0;
w2a2p : 0;
v3a3 : 0;
w3a3 : 0;
trajectory : ev ( trajectory );

rigidx : rigidx = part ( trajectory, 1, 1 );
rigidy : rigidy = part ( trajectory, 2, 1 );
rigidz : rigidz = part ( trajectory, 3, 1 );

Print the output to a listing file.

writefile ( "erro.lst" );
fortran ( elastix );
fortran ( elasticy );
fortran ( elastiz );
fortran ( rigidx );
fortran ( rigidy );
fortran ( rigidz );
closefile ();

Mission accomplished.

Program name : Energy1.com
This routine helps to set up the kinetic energy equations, the potential energy equations, and the Lagrange operator for link 1.

Find out the kinetic and potential energy equations for link 1.

lagrange : ( j1 * th1d^2 ) / 2 +
integrate ( rho1 * ( vector_1dsq / 2 + gdotvector_1 ),
wt, 0, -d1 );

Find out the Lagrangian equations from the given energy equations.

batchload ( "lagrange.com" );
That's it folks...

Program name: Energy2.com

This routine helps to set up the kinetic energy equations, the potential energy equations, and the Lagrange operator for link 2.

Find out the kinetic and potential energy equations for link 2.

```c
lagrange : integrate ( rho2r * vector_2rdsq / 2, u2, 0, a2r ) +
          integrate ( rho2r * gdotvector_2r, u2, 0, a2r ) +
          integrate ( rho2e * vector_2edsq / 2, u2, a2r, a2 ) +
          integrate ( rho2e * gdotvector_2e, u2, a2r, a2 ) +
          integrate ( -eiv2 * v2ppA 2 / 2, u2, a2r, a2 ) +
          integrate ( -eiw2 * w2ppA 2 / 2, u2, a2r, a2 ) +
          sm2 * ( vector_2e_at_a2dsq / 2 + gdotvector_2e_at_a2 )$
```

Find out the Lagrangian equations from the given energy equations.

```c
batchload ( "lagrange.com" )$
```

That's it folks...

Program name: Energy3.com

This routine helps to set up the kinetic energy equations, the potential energy equations, and the Lagrange operator for link 3.

Find out the kinetic and potential energy equations for link 3.

```c
lagrange : integrate ( rho3 * vector_3dsq / 2, u3, 0, a3 ) +
          integrate ( rho3 * gdotvector_3, u3, 0, a3 ) +
          integrate ( -eiv3 * v3ppA 2 / 2, u3, 0, a3 ) +
          integrate ( -eiw3 * w3ppA 2 / 2, u3, 0, a3 ) +
          sm3 * ( vector_3_at_a3dsq / 2 + gdotvector_3_at_a3 )$
```

Find out the Lagrangian equations from the given energy equations.

```c
batchload ( "lagrange.com" )$
```

That's it folks...
/* Program name : EnergyC.com

This routine helps to set up the kinetic energy equations, the
potential energy equations, and the Lagrange operator for link c.
*******************************************************************
Find out the kinetic and potential energy equations for link c.
*/

ke : rhoc * trigreduce ( integrate ( vector_cdsq / 2, uc, 0, ac ) )$
pe : - integrate ( rhoc * gdotvector_c, uc, 0, ac )$
lagrange : ke - pe$

Find out the Lagrangian equations from the given energy equations.

batchload ("lagrange.com")$

That's it folks...

*/

/* Program name : EnergyH3.com

This routine helps to set up the kinetic energy equations, the
potential energy equations, and the Lagrange operator for link h3.
*******************************************************************
Find out the kinetic and potential energy equations for link h3.
*/

temp1 : integrate ( rhoh3s * vector_h3dsq/2, uh3, 0, ah3s )$
temp2 : integrate ( rhoh3p * vector_h3dsq/2, uh3, ah3pstart, ah3pend )$
temp3 : integrate ( rhoh3r * vector_h3dsq/2, uh3, ah3pend, ah3s )$
ke : factor ( temp1 + temp2 + temp3 )$
temp1 : integrate ( rhoh3s * gdotvector_h3r, uh3, 0, ah3s )$
temp2 : integrate ( rhoh3p * gdotvector_h3r, uh3, ah3pstart, ah3pend )$
temp3 : integrate ( rhoh3r * gdotvector_h3r, uh3, ah3pend, ah3s )$
pe : factor ( temp1 + temp2 + temp3 )$
lagrange : ke + pe +
        integrate ( rhoh3r * vector_h3edsq / 2, uh3, ah3s, ah3 ) +
        integrate ( rhoh3r * gdotvector_h3e, uh3, ah3s, ah3 ) +
        integrate ( -eih3 * (vh3pp^2+wh3pp^2) / 2, uh3, ah3s, ah3 )$
lagrange : subst ( diff ( ah3, t ), uh3d, lagrange )$
lagrange : subst ( diff ( thh3, t ), thh3d, lagrange )$

Find out the Lagrangian equations from the given energy equations.

batchload ("lagrange.com")$

That's it folks...
Program name: H3init.com

This command procedure file helps to initialize and setup variables for the third hydraulic cylinder.

Initialize variables.

\*

v2 : gv2(u2) * qv2(t)
v2_at_a2 : gv2a2 * qv2(t)
v2p : diff ( v2, u2 )
v2_at_a2p : gv2a2p * qv2(t)

vh3_at_ah3 : gvh3ah3 * qvh3(t)

\lambda : % pi - ( zeta + \theta3(t) + v2_at_a2p )
h3x2 : ac * cos ( \lambda )
h3x1 : a2 - h3x2
h3y1 : ac * sin ( \lambda )
h3y : h3y1 + v2_at_a2

ah3sq : h3y^2 + h3x1^2
ah3 : sqrt ( ah3sq )

\eta : atan ( h3y / h3x1 ) - atan ( vh3_at_ah3 / ah3 )

\theta3 : eta + \theta2(t)

dah3t : dah3t = diff ( ah3, t)
dah3tt : dah3tt = diff ( ah3, t, 2 )
dah3tqv2 : dah3tqv2 = diff ( diff ( ah3, t ), qv2(t) )
dah3tth3 : dah3tth3 = diff ( diff ( ah3, t ), th3(t) )
dah3qv2 : dah3qv2 = diff ( ah3, qv2(t) )
dah3th3 : dah3th3 = diff ( ah3, th3(t) )

dth3t : dth3t = diff ( th3, t )
dth3tt : dth3tt = diff ( th3, t, 2 )
dth3tqv2 : dth3tqv2 = diff ( diff ( th3, t ), qv2(t) )
dth3tth3 : dth3tth3 = diff ( diff ( th3, t ), th3(t) )
dth3tqvh3 : dth3tqvh3 = diff ( diff ( th3, t ), qvh3(t) )
dth3th2 : dth3th2 = diff ( thh3, th2(t) )
dth3qv2 : dth3qv2 = diff ( thh3, qv2(t) )
dth3th3 : dth3th3 = diff ( thh3, th3(t) )
dth3qvh3 : dth3qvh3 = diff ( thh3, qvh3(t) )

detat : detat = diff ( \eta, t )
detat : detatt = diff ( \eta, t, 2 )
detatqv2 : detatqv2 = diff ( diff ( \eta, t ), qv2(t) )
detatth3 : detatth3 = diff ( diff ( \eta, t ), th3(t) )
detatqvh3 : detatqvh3 = diff ( diff ( \eta, t ), qvh3(t) )
detath3 : detath3 = diff ( \eta, th3(t) )
detath3 : detaqv3 = diff ( \eta, qvh3(t) )

That's it for now, see you...

*/
Program name: H3list.com

This command procedure file helps to create a printout listing of the variables of link h3.

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

Open the output listing file.
*/
writefile ( "h3var.lst" );

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

List out the content of the variables.
*/
fortran { dah3t };
fortran { dah3tt };
fortran { dah3tqv2 };
fortran { dah3tth3 };
fortran { dah3qv2 };
fortran { dah3th3 };

fortran { dthh3t };
fortran { dthh3tt };
fortran { dthh3tqv2 };
fortran { dthh3tth3 };
fortran { dthh3qv2 };
fortran { dthh3th3 };

fortran { detat };
fortran { detatt };
fortran { detatqv2 };
fortran { detatth3 };
fortran { detatqvh3 };
fortran { detaqv2 };
fortran { detath3 };
fortran { detaqvh3 };

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

Close the output listing file.
*/
closefile ();

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

That's all she wrote...
*/

Program name: H3simp.com

This is the command procedure file that helps to simplify and reduce the variables into a shorter form.

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

Remove the content of some of the variables first.
*/
kill { lamda }$
kill { ah3 }$
kill ( ah3sq )$
kill ( h3x1 )$
kill ( h3x2 )$
kill ( h3y )$
kill ( h3y1 )$

/*********************************************************
Simplify and reduce each variable.
*/

   temp : dah3ti
batchload ( "h3subsimp.com" )$
dah3t : temp$

temp : dah3tt$
batchload ( "h3subsimp.com" )$
dah3tt : temp$

temp : dah3tcqv2$
batchload ( "h3subsimp.com" )$
dah3tcqv2 : temp$

temp : dah3tcv2$
batchload ( "h3subsimp.com" )$
dah3tcv2 : temp$

temp : dah3th3$
batchload ( "h3subsimp.com" )$
dah3th3 : temp$

temp : dtth3ti
batchload ( "h3subsimp.com" )$
dtth3t : temp$

}
temp : dthh3qvh3$
batchload ( "h3subsimp.com" )$
dthh3qvh3 : temp$

temp : detat$
batchload ( "h3subsimp.com" )$
detat : temp$

temp : detatt$
batchload ( "h3subsimp.com" )$
detatt : temp$

temp : detaqv2$
batchload ( "h3subsimp.com" )$
detaqv2 : temp$

temp : detath3$
batchload ( "h3subsimp.com" )$
detath3 : temp$

temp : detaqvh3$
batchload ( "h3subsimp.com" )$
detaqvh3 : temp$

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

See you later...

*/

/∗  Program name : H3subsimp.com ∗/

This command procedure file helps to simplify and reduce the
content of the variable currently stored in 'temp'.

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

/*
*/

temp : subst ( lambda, ( th3(t) + gv2a3p * qv2(t) + zeta ), temp )$
temp : subst ( coslambda, cos ( lambda ), temp )$
temp : subst ( coslambdaasq, coslambda^2, temp )$
temp : subst ( sinlambda, sin ( lambda ), temp )$
temp : subst ( sinlambdaasq, sinlambda^2, temp )$
temp : subst ( h3x2, ac * coslambda, temp )$
temp : subst ( h3x2asq, h3x2^2, temp )$
temp : subst ( h3x1, h3x2 + a2, temp )$
temp : subst ( h3x1asq, h3x1^2, temp )$
temp : subst ( h3x1cu, h3x1^3, temp )$
temp : subst ( h3y1, ac * sinlambda, temp )$
temp : subst ( h3y1asq, h3y1^2, temp )$
temp : subst ( h3y1cu, h3y1^3, temp )$
temp : subst ( h3y, ( h3y1 + v2_at_a2 ), temp )$
temp : subst ( h3ysq, h3y^2, temp )$
temp : subst ( ah3sq, ( h3ysq + h3x1asq ), temp )$
temp : subst ( ah3cu, ah3sq^2, temp )$
temp : subst ( ah3fo, ah3sq^2, temp )$
temp : subst ( ah3fi, ah3sq^3, temp )$
temp : subst ( ah3se, ah3sq^4, temp )$

temp : subst ( ah3, sqrt ( ah3sq ), temp )$

temp : subst ( acsq, acA^2, temp )$

temp : subst ( a^2, a^2A^2, temp )$

temp : subst ( thd, diff ( th^2(t), t ), temp )$

temp : subst ( th3dd, diff ( th^3(t), t, 2 ), temp )$

temp : subst ( qv2d, diff ( qv^2(t), t ), temp )$

temp : subst ( qv2dd, diff ( qv^2(t), t, 2 ), temp )$

temp : subst ( qv2, qv^2(t), temp )$

temp : subst ( gv2a2sq, gv2a2p^2, temp )$

temp : subst ( psid, th^3d + gv2a2p * qv^2d, temp )$

temp : subst ( psid2, th3dd + gv2a2p * qv2dd, temp )$

temp : subst ( qvh3d, diff ( qvh^3(t), t ), temp )$

temp : subst ( qvh3dd, diff ( qvh^3(t), t, 2 ), temp )$

temp : subst ( qvh3, qvh^3(t), temp )$

temp : subst ( qvh3sq, qvh^3s^2, temp )$

temp : subst ( qvh3cu, qvh^3c^2, temp )$

temp : subst ( h3var(1),
( 2 * ac * coslambda * h3y - 2 * ac * sinlambda * h3x1 ), temp )$

temp : subst ( h3var(2),
( ac * coslambda * gv2a2p + gv2a2 ), temp )$

temp : subst ( h3var(3),
( h3ysq / h3x1sq ) + 1 ), temp )$

temp : subst ( h3var(4), h3var(3)^2, temp )$

temp : subst ( h3var(5),
( gvh3h3sqq / ah3sq ) + 1 ), temp )$

temp : subst ( h3var(6), h3var(5)^2, temp )$

temp : subst ( h3var(7),
( gv2a2 * qv^2d + ac * coslambda * psid ), temp )$

temp : subst ( h3var(8), h3var(7)^2, temp )$

temp : subst ( h3var(9),
( ac * psid * sinlambda * h3y ), temp )$

temp : subst ( h3var(10),
( h3var(9) / h3x1sq + h3var(7) / h3x1 ), temp )$

temp : subst ( h3var(11),
( ac * sinlambda * h3y / h3x1sq ) + ( h3x2 / h3x1 ) ), temp )$

temp : subst ( h3var(12),
( 2 * h3var(2) * h3y - 2 * ac * gv2a2p * sinlambda * h3x1 ), temp )$

temp : subst ( h3var(13),
( 2 * h3var(7) * h3y - 2 * ac * psid * sinlambda * h3x1 ), temp )$

temp : subst ( h3var(14), h3var(13)^2, temp )$

temp : subst ( h3var(15),
( gvh3h3sq / qvh3d / ah3 - gvh3h3 * h3var(13) * qv^3 / h3x1cu ), temp )$

temp : subst ( h3var(16),
( 2 * ac * gv2a2p * sinlambda * h3ysq / h3x1cu + 2 * h3var(2) * h3y / h3x1sq ), temp )$

temp : subst ( h3var(17),
( 2 * ac * sinlambda * h3ysq / h3x1cu ) + ( 2 * ac * coslambda * h3y / h3x1sq ), temp )$

temp : subst ( h3var(18),
( 2 * ac * coslambda * h3y / h3x1sq ) + ( h3var(2) / h3x1 ) ), temp )$

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

That's it folks...

*/
/* Program name : H3var.com  
This is the main program that helps to set up a listing of definitions for the variables of the second hydraulic cylinder.  
*********************************************************************/

Setup the Maxima parameter.

nolabels: true

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

Call subprogram to initialize and setup variables.

batchload ("h3init.com")$  

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

Call subprogram to simplify the variables.

batchload ("h3simp.com")$  

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

Call subprogram to create an output listing.

batchload ("h3list.com")$  

/*  Mission accomplished, see you...  */

quit()$

/* Program name : Init.com  
This command procedure routine helps to initialize the system variables for later use.  
*********************************************************************/

Setup the cross product.

cross_product(i,j,k) := [ unitb * unitz - unite * unity, 
unite * unitx - unita * unitz, 
unita * unity - unitb * unitx ]$

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

Setup the gravity field vector.

transform_1_to_2 : matrix ( [ cos(th2(t)), 0, sin(th2(t)) ], 
[ -sin(th2(t)), 0, cos(th2(t)) ], 
[ 0, -1, 0 ] )$

gravity_at_1 : matrix ( [ 0 ], [ 0 ], [ -g ] )$

gravity_at_2 : transpose ( transform_1_to_2 . gravity_at_1 )$
That's it for this routine, bye...

/* Program name: Killvar.com

This routine helps to remove the content of the time derivatives of the generalized coordinates.

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

kill (th1d)
kill (th2d)
kill (qv2d)
kill (qv2dd)
kill (qv2d)
kill (qv2dd)
kill (th3d)
kill (qv3d)
kill (qv3dd)
kill (qv3d)
kill (qv3dd)
kill (qv3d)
kill (qv3dd)

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

Mission accomplished, see you...

/* Program name: Lagrange.com

This routine helps to formulate the Lagrangian equations.

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

l ine1 : diff (diff (lagrange, th1d), t ) - diff (lagrange, th1(t))
line2 : diff (diff (lagrange, th2d), t ) - diff (lagrange, th2(t))
line3 : diff (diff (lagrange, th3d), t ) - diff (lagrange, th3(t))
line4 : diff (diff (lagrange, qv2d), t ) - diff (lagrange, qv2(t))
line5 : diff (diff (lagrange, qv2d), t ) - diff (lagrange, qv2(t))
line6 : diff (diff (lagrange, qv3d), t ) - diff (lagrange, qv3(t))
line7 : diff (diff (lagrange, qv3d), t ) - diff (lagrange, qv3(t))
line8 : diff (diff (lagrange, qv3d), t ) - diff (lagrange, qv3(t))

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

That's it for now, later...

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

(run)
This is the procedure routine that helps to derive the equations of motion for the first link of the UNLV-ARO robot.

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

Setup Macsyma parameter.

nolabels : true$

Call subroutine to initialize variables.

batchload ( "init.com" )$

Set up variables and parameters for link 1.

batchload ( "varsetup1.com" )$

Set up the energy equations and the Lagrangian equations for link 1.

batchload ( "energy1.com" )$

Call subroutine to extract the coefficients of link 1.

batchload ( "coeff1.com" )$

Create output listing files and save the results in a file for link 1.

batchload ( "save1.com" )$

Mission accomplished, later.

quit ()$

(run)

Program name : Link2.com

This is the procedure routine that helps to derive the equations of motion for the second link of the UNLV-ARO robot.

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

Setup Macsyma parameter.

nolabels : true$

*******************************************************************
Call subroutine to initialize variables.

*/
batchload ( "init.com" )$
/*  **************************************************
Set up variables and parameters for link 1 and link 2.
*/
batchload ( "varsetup1.com" )$
batchload ( "varsetup2.com" )$
/*  **************************************************
Set up the energy equations and the Lagrangian equations for
link 2.
*/
batchload ( "energy2.com" )$
/*  **************************************************
Call subroutine to extract the coefficients of link 2.
*/
batchload ( "coeff2.com" )$
/*  **************************************************
Create output listing files and save the results in a file for
link 2.
*/
batchload ( "save2.com" )$
/*  **************************************************
Mission accomplished, later.
*/
quit ()$

(run)
/*  Program name : Link3.com

This is the procedure routine that helps to derive the equations
of motion for the outer-most link of the UNLV-ARD robot.

**************************************************
Setup Macsyma parameter.
*/
nolabels : true$
/*  **************************************************
Call subroutine to initialize variables.
*/
batchload ( "init.com" )$
/*  **************************************************
Set up variables and parameters for link 1, link 2, and link 3.
batchload ("varsetup1.com")$
batchload ("varsetup2.com")$
batchload ("varsetup3.com")$
/*  ***************************************************************/
  Set up the energy equations and the lagrangian equations for
  link 3.
/*
batchload ("energy3.com")$
/*  ***************************************************************/
  Call subroutine to extract the coefficients of link 3.
/*
batchload ("coeff3.com")$
/*  ***************************************************************/
  Create output listing files and save the results in a file for
  link 3.
/*
batchload ("save3.com")$
/*  ***************************************************************/
  Mission accomplished, later.
*/
quit ()$

(run)
/*  Program name : Linkc.com
   This is the procedure routine that helps to derive the equations
   of motion for the connecting link of the UNLV-ARO robot.
   ***************************************************************/
  Setup Macsyma parameter.
/*
nolabels : true$
/*  ***************************************************************/
  Call subroutine to initialize variables.
/*
batchload ("init.com")$
/*  ***************************************************************/
  Set up variables and parameters for link 1, link 2, link 3, and
  link c.
/*
batchload ("varsetup1.com")$
batchload ("varsetup2.com")$
batchload ("varsetup3.com")$

batchload ("varsetupc.com")$

/* ***********************************************
   Set up the energy equations and the lagrangian equations for
   link c.
*/
batchload ("energy.c")$

/* ***********************************************
   Call subroutine to extract the coefficients of link c.
*/
batchload ("coeffc.com")$

/* ***********************************************
   Create output listing files and save the results in a file for
   link c.
*/
batchload ("savec.com")$

/* ***********************************************
   Mission accomplished, later.
*/
quit ()$

/* Program name : Linkh3.com
   This is the procedure routine that helps to derive the equations
   of motion for hydraulic cylinder 3 of the UNLV-ARD robot.
   ***********************************************
   Setup Macsyma parameter.
*/
nolabels : true$

/* ***********************************************
   Call subroutine to initialize variables.
*/
batchload ("init.com")$

/* ***********************************************
   Set up variables and parameters for link 1, link 2, and link h3.
*/
batchload ("varsetup1.com")$
batchload ("varsetup2.com")$
batchload ("varsetup3.com")$

/* ***********************************************
   Set up the energy equations and the lagrangian equations for
link h3.

batchload ("energyh3.com")$

/*
   **************************************************************
   Call subroutine to extract the coefficients of link h3.
   */
batchload ("coeffh3.com")$

/*
   **************************************************************
   Create output listing files and save the results in a file for
   link h3.
   */
batchload ("saveh3.com")$

/*
   Mission accomplished, later.
   */
quit ()$

/* Program name: Listing.com
   This command procedure routine helps to create output listings for
   this project.
   **************************************************************
   Print the equations into a listing file.
   */
writefile (filename1)$
line1;
line2;
line3;
line4;
line5;
line6;
line7;
line8;
closefile ()$

/*
   **************************************************************
   Print all the coefficients into a listing file.
   */
writefile (filename2)$
coeffa1;
coeffb1;
coeffc1;
coeffd1;
coeffe1;
coefff1;
coeffg1;
coeffh2;
coeffi2;
coeffj2;
coeffk2;
coeffl2;
coeffm2;
Done!!!

(/run)

/* Program name: Pickapart.com

This subroutine helps to breakdown the equations of the system into smaller elements.

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

Load the data file of link 1 and breakdown the equations.

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

loadfile ("link1.fil")$

print_torque1_1 : torque1_1 = pickapart (line1, 1);
print_torque2_1 : torque2_1 = pickapart (line2, 1);
print_torque3_1 : torque3_1 = pickapart (line3, 1);
print_coeffa1_1 : coeffa1_1 = pickapart (coeffa1, 1);
print_coeffb1_1 : coeffb1_1 = pickapart (coeffb1, 1);
print_coeffc1_1 : coeffc1_1 = pickapart (coeffc1, 1);
print_coeffd1_1 : coeffd1_1 = pickapart (coeffd1, 1);
print_coeffe1_1 : coeffe1_1 = pickapart (coeffe1, 1);
print_coefff1_1 : coefff1_1 = pickapart (coefff1, 1);
print_coeffa2_1 : coeffa2_1 = pickapart (coeffa2, 1);
print_coeffb2_1 : coeffb2_1 = pickapart (coeffb2, 1);
print_coeffc2_1 : coeffc2_1 = pickapart (coeffc2, 1);
print_coeffd2_1 : coeffd2_1 = pickapart (coeffd2, 1);
print_coeffe2_1 : coeffe2_1 = pickapart (coeffe2, 1);
print_coefff2_1 : coefff2_1 = pickapart (coefff2, 1);
print_coeffa3_1 : coeffa3_1 = pickapart (coeffa3, 1);
print_coeffb3_1 : coeffb3_1 = pickapart (coeffb3, 1);
print_coeffc3_1 : coeffc3_1 = pickapart (coeffc3, 1);
print_coeffd3_1 : coeffd3_1 = pickapart (coeffd3, 1);
print_coeffe3_1 : coeffe3_1 = pickapart (coeffe3, 1);
print_coefff3_1 : coefff3_1 = pickapart (coefff3, 1);
print_coeffa4_1 : coeffa4_1 = pickapart (coeffa4, 1);
print_coeffb4_1 : coeffb4_1 = pickapart (coeffb4, 1);
print_coeffc4_1 : coeffc4_1 = pickapart (coeffc4, 1);
print_coeffd4_1 : coeffd4l1 = pickapart ( coeffd4 , 1 )
print_coeffe4_1 : coeffe4l1 = pickapart ( coeffe4 , 1 )
print_coefff4_1 : coefff4l1 = pickapart ( coefff4 , 1 )
print_coeffa5_1 : coeffa5l1 = pickapart ( coeffa5 , 1 )
print_coeffb5_1 : coeffb5l1 = pickapart ( coeffb5 , 1 )
print_coeffc5_1 : coeffc5l1 = pickapart ( coeffc5 , 1 )
print_coeffd5_1 : coeffd5l1 = pickapart ( coeffd5 , 1 )
print_coeffe5_1 : coeffe5l1 = pickapart ( coeffe5 , 1 )
print_coefff5_1 : coefff5l1 = pickapart ( coefff5 , 1 )

save ( "pickapartl.fil", print_torque1_1, print_torque2_1, print_torque3_1, print_coeffa1, print_coeffb1, print_coeffc1, print_coeffd1, print_coeffe1, print_coefff1, print_coeffa2, print_coeffb2, print_coeffc2, print_coeffd2, print_coeffe2, print_coefff2, print_coeffa3, print_coeffb3, print_coeffc3, print_coeffd3, print_coeffe3, print_coefff3, print_coeffa4, print_coeffb4, print_coeffc4, print_coeffd4, print_coeffe4, print_coefff4, print_coeffa5, print_coeffb5, print_coeffc5, print_coeffd5, print_coeffe5, print_coefff5, labels )$

kill ( all )$

/*
 * Load the data file of link 2 and breakdown the equations.
 */

loadfile ( "link2.fil" )$

print_torque1_2 : torque1l2 = pickapart ( line1 , 1 )
print_torque2_2 : torque2l2 = pickapart ( line2 , 1 )
print_torque3_2 : torque3l2 = pickapart ( line3 , 1 )
print_coeffa1_2 : coeffa1l2 = pickapart ( coeffa1 , 1 )
print_coeffb1_2 : coeffbl2 = pickapart ( coeffb1 , 1 )
print_coeffc1_2 : coeffcl2 = pickapart ( coeffc1 , 1 )
print_coeffd1_2 : coeffdl2 = pickapart ( coeffdl , 1 )
print_coeffe1_2 : coeffel2 = pickapart ( coeffel , 1 )
print_coefff1_2 : coefffl2 = pickapart ( coefff1 , 1 )
print_coeffa2_2 : coeffa2l2 = pickapart ( coeffa2 , 1 )
print_coeffb2_2 : coeffbl2 = pickapart ( coeffb2 , 1 )
print_coeffc2_2 : coeffcl2 = pickapart ( coeffc2 , 1 )
print_coeffd2_2 : coeffdl2 = pickapart ( coeffd2 , 1 )
print_coeffe2_2 : coeffel2 = pickapart ( coeffe2 , 1 )
print_coefff2_2 : coefffl2 = pickapart ( coefff2 , 1 )
print_coeffa3_2 : coeffa3l2 = pickapart ( coeffa3 , 1 )
print_coeffb3_2 : coeffbl2 = pickapart ( coeffb3 , 1 )
print_coeffc3_2 : coeffcl2 = pickapart ( coeffc3 , 1 )
print_coeffd3_2 : coeffdl2 = pickapart ( coeffd3 , 1 )
print_coeffe3_2 : coeffel2 = pickapart ( coeffe3 , 1 )
print_coefff3_2 : coefffl2 = pickapart ( coefff3 , 1 )
print_coeffa4_2 : coeffa4l2 = pickapart ( coeffa4 , 1 )
print_coeffb4_2 : coeffbl2 = pickapart ( coeffb4 , 1 )
print_coeffc4_2 : coeffcl2 = pickapart ( coeffc4 , 1 )
print_coeffd4_2 : coeffdl2 = pickapart ( coeffd4 , 1 )
print_coeffe4_2 : coeffel2 = pickapart ( coeffe4 , 1 )
print_coefff4_2 : coefffl2 = pickapart ( coefff4 , 1 )
print_coeffa5_2 : coeffa5l2 = pickapart ( coeffa5 , 1 )
print_coeffb5_2 : coeffbl2 = pickapart ( coeffb5 , 1 )
print_coeffc5_2 : coeffcl2 = pickapart ( coeffc5 , 1 )
print_coeffd5_2 : coeffdl2 = pickapart ( coeffd5 , 1 )
print_coeffe5_2 : coeffel2 = pickapart ( coeffe5 , 1 )
print_coefff5_2 : coefff5l2 = pickapart ( coefff5, 1 );

save ( "pickapart2.fil",
    print_torque1_2, print_torque2_2, print_torque3_2,
    print_coeffa1_2, print_coeffb1_2, print_coeffc1_2,
    print_coeffd1_2, print_coeffe1_2, print_coefff1_2,
    print_coeffa2_2, print_coeffb2_2, print_coeffc2_2,
    print_coeffd2_2, print_coeffe2_2, print_coefff2_2,
    print_coeffa3_2, print_coeffb3_2, print_coeffc3_2,
    print_coeffd3_2, print_coeffe3_2, print_coefff3_2,
    print_coeffa4_2, print_coeffb4_2, print_coeffc4_2,
    print_coeffd4_2, print_coeffe4_2, print_coefff4_2,
    print_coeffa5_2, print_coeffb5_2, print_coeffc5_2,
    print_coeffd5_2, print_coeffe5_2, print_coefff5_2,
    labels );

kill ( all )$

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

  Load the data file of link 3 and breakdown the equations.
/*

loadfile ( "link3.fil" )$

print_torque1_3 : torque1l3 = pickapart ( line1, 1 );
print_torque2_3 : torque2l3 = pickapart ( line2, 1 );
print_torque3_3 : torque3l3 = pickapart ( line3, 1 );

print_coeffa1_3 : coeffa1l3 = pickapart ( coeffa1, 1 );
print_coeffb1_3 : coeffb1l3 = pickapart ( coeffb1, 1 );
print_coeffc1_3 : coeffc1l3 = pickapart ( coeffc1, 1 );
print_coeffd1_3 : coeffd1l3 = pickapart ( coeffd1, 1 );
print_coeffe1_3 : coeffe1l3 = pickapart ( coeffe1, 1 );
print_coefff1_3 : coefff1l3 = pickapart ( coefff1, 1 );

print_coeffa2_3 : coeffa2l3 = pickapart ( coeffa2, 1 );
print_coeffb2_3 : coeffb2l3 = pickapart ( coeffb2, 1 );
print_coeffc2_3 : coeffc2l3 = pickapart ( coeffc2, 1 );
print_coeffd2_3 : coeffd2l3 = pickapart ( coeffd2, 1 );
print_coeffe2_3 : coeffe2l3 = pickapart ( coeffe2, 1 );
print_coefff2_3 : coefff2l3 = pickapart ( coefff2, 1 );

print_coeffa3_3 : coeffa3l3 = pickapart ( coeffa3, 1 );
print_coeffb3_3 : coeffb3l3 = pickapart ( coeffb3, 1 );
print_coeffc3_3 : coeffc3l3 = pickapart ( coeffc3, 1 );
print_coeffd3_3 : coeffd3l3 = pickapart ( coeffd3, 1 );
print_coeffe3_3 : coeffe3l3 = pickapart ( coeffe3, 1 );
print_coefff3_3 : coefff3l3 = pickapart ( coefff3, 1 );

print_coeffa4_3 : coeffa4l3 = pickapart ( coeffa4, 1 );
print_coeffb4_3 : coeffb4l3 = pickapart ( coeffb4, 1 );
print_coeffc4_3 : coeffc4l3 = pickapart ( coeffc4, 1 );
print_coeffd4_3 : coeffd4l3 = pickapart ( coeffd4, 1 );
print_coeffe4_3 : coeffe4l3 = pickapart ( coeffe4, 1 );
print_coefff4_3 : coefff4l3 = pickapart ( coefff4, 1 );

print_coeffa5_3 : coeffa5l3 = pickapart ( coeffa5, 1 );
print_coeffb5_3 : coeffb5l3 = pickapart ( coeffb5, 1 );
print_coeffc5_3 : coeffc5l3 = pickapart ( coeffc5, 1 );
print_coeffd5_3 : coeffd5l3 = pickapart ( coeffd5, 1 );
print_coeffe5_3 : coeffe5l3 = pickapart ( coeffe5, 1 );
print_coefff5_3 : coefff5l3 = pickapart ( coefff5, 1 );

save ( "pickapart3.fil",
    print_torque1_3, print_torque2_3, print_torque3_3,
    print_coeffa1_3, print_coeffb1_3, print_coeffc1_3,
    print_coeffd1_3, print_coeffe1_3, print_coefff1_3,
    print_coeffa2_3, print_coeffb2_3, print_coeffc2_3,
    print_coeffd2_3, print_coeffe2_3, print_coefff2_3,
    print_coeffa3_3, print_coeffb3_3, print_coeffc3_3,
Load the data file of link c and break down the equations.

```c
loadfile( "linkc.fil" )

print_torque1_c : torque1lc = pickapart( line1, 1)
pickapart( coeffa1, 1)
pickapart( coeffb1, 1)
pickapart( coeffc1, 1)
pickapart( coeffd1, 1)
pickapart( coeffe1, 1)
pickapart( coefff1, 1)
pickapart( coeffa2, 1)
pickapart( coeffb2, 1)
pickapart( coeffc2, 1)
pickapart( coeffd2, 1)
pickapart( coeffe2, 1)
pickapart( coefff2, 1)
pickapart( coeffa3, 1)
pickapart( coeffb3, 1)
pickapart( coeffc3, 1)
pickapart( coeffd3, 1)
pickapart( coeffe3, 1)
pickapart( coefff3, 1)
pickapart( coeffa4, 1)
pickapart( coeffb4, 1)
pickapart( coeffc4, 1)
pickapart( coeffd4, 1)
pickapart( coeffe4, 1)
pickapart( coefff4, 1)
pickapart( coeffa5, 1)
pickapart( coeffb5, 1)
pickapart( coeffc5, 1)
pickapart( coeffd5, 1)
pickapart( coeffe5, 1)
pickapart( coefff5, 1)
```

save( "pickapartc.fil",
      print_torque1_c, print_torque2_c, print_torque3_c,
pickapart( coeffa1, 1)
pickapart( coeffb1, 1)
pickapart( coeffc1, 1)
pickapart( coeffd1, 1)
pickapart( coeffe1, 1)
pickapart( coefff1, 1)
pickapart( coeffa2, 1)
pickapart( coeffb2, 1)
pickapart( coeffc2, 1)
pickapart( coeffd2, 1)
pickapart( coeffe2, 1)
pickapart( coefff2, 1)
pickapart( coeffa3, 1)
pickapart( coeffb3, 1)
pickapart( coeffc3, 1)
pickapart( coeffd3, 1)
pickapart( coeffe3, 1)
pickapart( coefff3, 1)
pickapart( coeffa4, 1)
pickapart( coeffb4, 1)
pickapart( coeffc4, 1)
pickapart( coeffd4, 1)
pickapart( coeffe4, 1)
pickapart( coefff4, 1)
pickapart( coeffa5, 1)
pickapart( coeffb5, 1)
pickapart( coeffc5, 1)
pickapart( coeffd5, 1)
pickapart( coeffe5, 1)
pickapart(coefff5, 1)
labels )
```
Load the data file of link h3 and breakdown the equations.

/*
 *  **********************************************************
 *  Load the data file of link h3 and breakdown the equations.
 *  
 *  loadfile ("linkh3.file")
 *  
 *  print_torque1_h3 : torque1lh3 = pickapart line1, 1
 *  print_torque2_h3 : torque2lh3 = pickapart line2, 1
 *  print_torque3_h3 : torque3lh3 = pickapart line3, 1
 *  
 *  print_coeffa1_h3 : coeffa1lh3 = pickapart coeffal, 1
 *  print_coeffb1_h3 : coeffb1lh3 = pickapart coeffbl, 1
 *  print_coeffc1_h3 : coeffc1lh3 = pickapart coeffcl, 1
 *  print_coeffd1_h3 : coeffd1lh3 = pickapart coeffdl, 1
 *  print_coeffe1_h3 : coeffe1lh3 = pickapart coeffel, 1
 *  print_coefff1_h3 : coeffflh3 = pickapart coefffl, 1
 *  
 *  print_coeffa2_h3 : coeffa2lh3 = pickapart coeffa2, 1
 *  print_coeffb2_h3 : coeffb2lh3 = pickapart coeffb2, 1
 *  print_coeffc2_h3 : coeffc2lh3 = pickapart coeffc2, 1
 *  print_coeffd2_h3 : coeffd2lh3 = pickapart coeffd2, 1
 *  print_coeffe2_h3 : coeffe2lh3 = pickapart coeffe2, 1
 *  print_coefff2h3 : coefffh3 = pickapart coefff2, 1
 *  
 *  print_coeffa3_h3 : coeffa3lh3 = pickapart coeffa3, 1
 *  print_coeffb3_h3 : coeffb3lh3 = pickapart coeffb3, 1
 *  print_coeffc3_h3 : coeffc3lh3 = pickapart coeffc3, 1
 *  print_coeffd3_h3 : coeffd3lh3 = pickapart coeffd3, 1
 *  print_coeffe3_h3 : coeffe3lh3 = pickapart coeffe3, 1
 *  print_coefff3h3 : coefffh3 = pickapart coefff3, 1
 *  
 *  print_coeffa4_h3 : coeffa4lh3 = pickapart coeffa4, 1
 *  print_coeffb4_h3 : coeffb4lh3 = pickapart coeffb4, 1
 *  print_coeffc4_h3 : coeffc4lh3 = pickapart coeffc4, 1
 *  print_coeffd4_h3 : coeffd4lh3 = pickapart coeffd4, 1
 *  print_coeffe4_h3 : coeffe4lh3 = pickapart coeffe4, 1
 *  print_coefff4h3 : coefffh3 = pickapart coefff4, 1
 *  
 *  print_coeffa5_h3 : coeffa5lh3 = pickapart coeffa5, 1
 *  print_coeffb5_h3 : coeffb5lh3 = pickapart coeffb5, 1
 *  print_coeffc5_h3 : coeffc5lh3 = pickapart coeffc5, 1
 *  print_coeffd5_h3 : coeffd5lh3 = pickapart coeffd5, 1
 *  print_coeffe5_h3 : coeffe5lh3 = pickapart coeffe5, 1
 *  print_coefff5h3 : coefffh3 = pickapart coefff5, 1
 *  
 *  save("pickaparth3.file",
 *  print_torque1_h3, print_torque2_h3, print_torque3_h3,
 *  print_coeffa1_h3, print_coeffb1_h3, print_coeffc1_h3,
 *  print_coeffd1_h3, print_coeffe1_h3, print_coefff1_h3,
 *  print_coeffa2_h3, print_coeffb2_h3, print_coeffc2_h3,
 *  print_coeffd2_h3, print_coeffe2_h3, print_coefff2_h3,
 *  print_coeffa3_h3, print_coeffb3_h3, print_coeffc3_h3,
 *  print_coeffd3_h3, print_coeffe3_h3, print_coefff3_h3,
 *  print_coeffa4_h3, print_coeffb4_h3, print_coeffc4_h3,
 *  print_coeffd4_h3, print_coeffe4_h3, print_coefff4_h3,
 *  print coeffa5_h3, print_coeffb5_h3, print_coeffc5_h3,
 *  print_coeffd5_h3, print_coeffe5_h3, print_coefff5_h3,
 *  labels)
 *  
 *  kill (all)
 */

Mission accomplished, exit program.
*/

quit()
/* Program name : Reduce.com

   This command procedure file helps to simplify and reduce the size of the equations of the system.

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

   Call subroutines to perform the simplications.
*/
batchload ( "reduce1.com" )$
batchload ( "reduce2.com" )$
batchload ( "reduce3.com" )$
batchload ( "reducec.com" )$
batchload ( "reduceh3.com" )$

batchload ( "sumup.com" )$

mission accomplished, see ya...
quit ()$
/* Program name: Reduce2.com

This subroutine helps to simplify all the equations of link 2.

****************************
Mission accomplished, later...

*/

kill (all);

loadfile ("pickapart2.fi" );

/* Simplify the torque equations. */

print_t_element1_2 : element (1, 1) = temp$
print_t_e2 : e2 = rho2e * integral (1) / 2.0$
temp : e3$
kill ( e3 )$
batchload ( "subreduce2.com" )$
print_t_e3 : e3 = temp$

/*
   **************************************************************************
   *                     Simplify the coefficient equations.                *
   **************************************************************************
*/

temp : e14$
kill ( e14 )$
batchload ( "subreduce2.com" )$
print_c_element1_2 : element ( 1, i ) = temp$
print_c_e14 : e14 = rho2e * integral ( 1 )$

temp : e15$
kill ( e15 )$
batchload ( "subreduce2.com" )$
print_c_e15 : e15 = temp$
/*
   .................................................................................*/

*/

temp : part ( e19, 3, 1 )$
kill ( e19 )$
batchload ( "subreduce2.com" )$
print_c_element2_2 : element ( 2, i ) = temp$
print_c_e19 : e19 = eiw2 * qv2 * integral ( 2 )$

temp : part ( e20, 1, 1, 2, 1 )$
kill ( e20 )$
batchload ( "subreduce2.com" )$
print_c_element3_2 : element ( 3, i ) = temp$
print_c_e20 : e20 = - rho2e * integral ( 3 ) / 2.0$

temp : part ( e21, 2, 1 )$
kill ( e21 )$
batchload ( "subreduce2.com" )$
print_c_element4_2 : element ( 4, i ) = temp$
print_c_e21 : e21 = rho2e * integral ( 4 )$

temp : e22$
kill ( e22 )$
batchload ( "subreduce2.com" )$
print_c_e22 : e22 = g * rho2e * costh2 * integral ( 5 )$

temp : e23$
kill ( e23 )$
batchload ( "subreduce2.com" )$
print_c_e23 : e23 = temp$
/*
   .................................................................................*/

temp : part ( e25, 2, 1 )$
kill ( e25 )$
batchload ( "subreduce2.com" )$
print_c_element6_2 : element ( 6, i ) = temp$
print_c_e25 : e25 = rho2e * integral ( 6 )$

temp : e26$
kill ( e26 )$
batchload ( "subreduce2.com" )$
print_c_e26 : e26 = temp$
/*
   .................................................................................*/

temp : part ( e29, 3, 1 )$
kill ( e29 )$
batchload ( "subreduce2.com" )$
print_c_element7_2 : element ( 7, i ) = temp$
print_c_e29 : e29 = eiw2 * qv2 * integral ( 7 )$

temp : part ( e30, 1, 1, 2, 1 )$
kill ( e30 )$
batchload ( "subreduce2.com" )$
print_c_element8_2 : element ( 8, i ) = temp$
\[ e_{30} = -\rho_{2e} \int e \, ds \]

\[ t_{\text{temp}} = \text{part}(e_{31}, 2, 1) \]
\[ \text{kill}(e_{31}) \]
\[ \text{batchload}(\text{"subreduce2.com"}) \]
\[ \text{print}_{\text{element9,2}} = \text{element}(9, i) = t_{\text{temp}} \]
\[ \text{print}_{\text{element31}} = \rho_{2e} \int e \, ds \]

\[ t_{\text{temp}} = e_{32} \]
\[ \text{kill}(e_{32}) \]
\[ \text{batchload}(\text{"subreduce2.com"}) \]
\[ \text{print}_{\text{element32}} = e_{32} = t_{\text{temp}} \]

\[ t_{\text{temp}} = e_{33} \]
\[ \text{kill}(e_{33}) \]
\[ \text{batchload}(\text{"subreduce2.com"}) \]
\[ \text{print}_{\text{element33}} = e_{33} = t_{\text{temp}} \]

/* *******************************************************************
*  Translate the equations into Fortran codes.                      
*/

\text{writefile}(\text{"fortran2.lst"})
\text{fortran}(\text{print}_t_{\text{element1,2}})
\text{fortran}(\text{print}_t_{\text{element2,2}})
\text{fortran}(\text{print}_t_{\text{element3,2}})
\text{fortran}(\text{print}_t_{\text{element4,2}})
\text{fortran}(\text{print}_t_{\text{element1,2}})
\text{fortran}(\text{print}_t_{\text{element2,2}})
\text{fortran}(\text{print}_t_{\text{element3,2}})
\text{fortran}(\text{print}_t_{\text{element4,2}})
\text{fortran}(\text{print}_t_{\text{element5,2}})
\text{fortran}(\text{print}_t_{\text{element6,2}})
\text{fortran}(\text{print}_t_{\text{element7,2}})
\text{fortran}(\text{print}_t_{\text{element8,2}})
\text{fortran}(\text{print}_t_{\text{element9,2}})
\text{fortran}(\text{print}_c_{e14})
\text{fortran}(\text{print}_c_{e15})
\text{fortran}(\text{print}_c_{e19})
\text{fortran}(\text{print}_c_{e20})
\text{fortran}(\text{print}_c_{e21})
\text{fortran}(\text{print}_c_{e22})
\text{fortran}(\text{print}_c_{e23})
\text{fortran}(\text{print}_c_{e24})
\text{fortran}(\text{print}_c_{e25})
\text{fortran}(\text{print}_c_{e26})
\text{fortran}(\text{print}_c_{e29})
\text{fortran}(\text{print}_c_{e30})
\text{fortran}(\text{print}_c_{e31})
\text{fortran}(\text{print}_c_{e32})
fortran ( print_c_e33 );
fortran ( print_coeffa1_2 );
fortran ( print_coeffb1_2 );
fortran ( print_coeffc1_2 );
fortran ( print_coeffd1_2 );
fortran ( print_coeffe1_2 );
fortran ( print_coefff1_2 );
fortran ( print_coeffa2_2 );
fortran ( print_coeffb2_2 );
fortran ( print_coeffc2_2 );
fortran ( print_coeffd2_2 );
fortran ( print_coeffe2_2 );
fortran ( print_coefff2_2 );
fortran ( print_coeffa3_2 );
fortran ( print_coeffb3_2 );
fortran ( print_coeffc3_2 );
fortran ( print_coeffd3_2 );
fortran ( print_coeffe3_2 );
fortran ( print_coefff3_2 );
fortran ( print_coeffa4_2 );
fortran ( print_coeffb4_2 );
fortran ( print_coeffc4_2 );
fortran ( print_coeffd4_2 );
fortran ( print_coeffe4_2 );
fortran ( print_coefff4_2 );
fortran ( print_coeffa5_2 );
fortran ( print_coeffb5_2 );
fortran ( print_coeffc5_2 );
fortran ( print_coeffd5_2 );
fortran ( print_coeffe5_2 );
fortran ( print_coefff5_2 );
closefile ();
/
MISSION accomplished, later...
/*/ 

/* Program name : Reduce3.com 
This subroutine helps to simplify all the equations of link 3.
*******************************************************************************
Clear the memory.
*/
kill ( all );
/
*******************************************************************************
Load file into memory.
*/
loadfile ( "pickapart3.fil" );
/
*******************************************************************************
Simplify the torque equations.
*/
temp : part ( e2, 1, 2, 1 );$
kill (e2)
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_t_element1_3 : element (1, 1) = temp$
print_t_e2 : e2 = rho3 * integral (1) / 2.0$

temp : e3$
kill (e3)$
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_t_e3 : e3 = temp$
/
/*
*-------------------------------------------------------------------*/
*/
temp : part (e4, 1, 2, 1)$
kill (e4)$
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_t_element2_3 : element (2, i) = temp$
print_t_e4 : e4 = rho3 * integral (2) / 2.0$
temp : part (e5, 1, 1, 2, 1)$
kill (e5)$
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_t_element3_3 : element (3, 1) = temp$
print_t_e5 : e5 = - rho3 * integral (3) / 2.0$
temp : part (e6, 3, 1)$
kill (e6)$
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_t_element4_3 : element (4, i) = temp$
print_t_e6 : e6 = g * rho3 * integral (4)$
temp : e7$
kill (e7)$
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_t_e7 : e7 = temp$
temp : e8$
kill (e8)$
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_t_e8 : e8 = temp$
/
/*
*-------------------------------------------------------------------*/
*/
temp : part (e9, 1, 2, 1)$
kill (e9)$
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_t_element5_3 : element (5, i) = temp$
print_t_e9 : e9 = rho3 * integral (5) / 2.0$
temp : part (e10, 1, 1, 2, 1)$
kill (e10)$
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_t_element6_3 : element (6, i) = temp$
print_t_e10 : e10 = - rho3 * integral (6) / 2.0$
temp : part (e11, 3, 1)$
kill (e11)$
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_t_element7_3 : element (7, i) = temp$
print_t_e11 : e11 = g * rho3 * integral (7)
temp : e12
kill ( e12 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_t_e12 : e12 = temp

temp : e13
kill ( e13 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_t_e13 : e13 = temp

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

Simplify the coefficient equations.
*/

temp : part ( e14, 1, 2, 1 )
kill ( e14 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_element1_3 : element ( 1, i ) = temp
print_c_e14 : e14 = rho3 * integral ( 1 ) / 2.0

temp : e15
kill ( e15 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_e15 : e15 = temp

/* ------------------------------------------------------------------------*/

temp : part ( e16, 1, 5, 1 )
kill ( e16 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_element2_3 : element ( 2, i ) = temp
print_c_e16 : e16 = -gw2a2p * gw3a2p * rho3 * qw3 * integral ( 2 )

temp : e17
kill ( e17 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_e17 : e17 = temp

/* ------------------------------------------------------------------------*/

temp : part ( e18, 1, 2, 1 )
kill ( e18 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_element3_3 : element ( 3, i ) = temp
print_c_e18 : e18 = rho3 * integral ( 3 ) / 2.0

temp : e19
kill ( e19 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_e19 : e19 = temp

/* ------------------------------------------------------------------------*/

temp : part ( e21, 1, 2, 1 )
kill ( e21 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_element4_3 : element ( 4, i ) = temp
print_c_e21 : e21 = rho3 * integral (4) / 2.0

temp : part (e22, 1, 1, 2, 1)
kill (e22)
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_c_element5_3 : element (5, i) = temp
print_c_e22 : e22 = - rho3 * integral (5) / 2.0

temp : part (e23, 3, 1)
kill (e23)
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_c_element6_3 : element (6, i) = temp
print_c_e23 : e23 = g * rho3 * integral (6)

temp : e24
kill (e24)
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_c_e24 : e24 = temp

temp : e25
kill (e25)
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_c_e25 : e25 = temp

/* ..................................................................................................*/

temp : part (e26, 1, 2, 1)
kill (e26)
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_c_element7_3 : element (7, i) = temp
print_c_e26 : e26 = rho3 * integral (7) / 2.0

/* ..................................................................................................*/

temp : part (e28, 1, 5, 1)
kill (e28)
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_c_element8_3 : element (8, i) = temp
print_c_e28 : e28 = gw2a2p * rho3 * qw3 * sinpsi * integral (8)

temp : e29
kill (e29)
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_c_e29 : e29 = temp

/* ..................................................................................................*/

temp : part (e30, 2, 1)
kill (e30)
batchload ("subreduce2.com")
batchload ("subreduce3.com")
print_c_element9_3 : element (9, i) = temp
print_c_e30 : e30 = rho3 * integral (9)

temp : e31
kill ( e31 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_e31 : e31 = temp$

/* .................................................................*
 temp : part ( e32, 1, 2, 1 )
 kill ( e32 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_element10_3 : element ( 10, i ) = temp$
print_c_e32 : e32 = rho3 * integral ( 10 ) / 2.0$

 temp : part ( e33, 1, 1, 2, 1 )
 kill ( e33 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_element11_3 : element ( 11, i ) = temp$
print_c_e33 : e33 = - rho3 * integral ( 11 ) / 2.0$

 temp : e34$
 kill ( e34 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_e34 : e34 = temp$

 temp : e35$
 kill ( e35 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_e35 : e35 = temp$

/* .................................................................*
 temp : part ( e37, 1, 2, 1 )
 kill ( e37 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_element12_3 : element ( 12, i ) = temp$
print_c_e37 : e37 = rho3 * integral ( 12 ) / 2.0$

 temp : e38$
 kill ( e38 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_e38 : e38 = temp$

/* .................................................................*
 temp : part ( e40, 3, 1 )
 kill ( e40 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_element13_3 : element ( 13, i ) = temp$
print_c_e40 : e40 = eiv3 * qv3 * integral ( 13 )$

 temp : part ( e41, 1, 2, 1 )
 kill ( e41 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_element14_3 : element ( 14, i ) = temp$
print_c_e41 : e41 = rho3 * integral ( 14 ) / 2.0$

 temp : part ( e42, 1, 1, 2, 1 )
 kill ( e42 )
batchload ( "subreduce2.com" )
batchload ( "subreduce3.com" )
print_c_element15_3 : element ( 15, i ) = temp 
print_c_e42 : e42 = - rho3 * integral ( 15 ) / 2.0 

temp : part ( e43, 4, 1 )$ 
kill ( e43 )$ 
batchload ( "subreduce2.com" )$
batchload ( "subreduce3.com" )$
print_c_element16_3 : element ( 16, i ) = temp 
print_c_e43 : e43 = g * rho3 * coschi * integral ( 16 )$

temp : e44$ 
kill ( e44 )$ 
batchload ( "subreduce2.com" )$
batchload ( "subreduce3.com" )$
print_c_e44 : e44 = temp 

print_c_e45 : e45 = temp 

print_c_e48 : e48 = rho3 * integral ( 18 )$

temp : part ( e50, 3, 1 )$

print_c_e50 : e50 = e1w3 * qw3 * integral ( 18 )$

temp : part ( e51, 1, 2, 1 )$

print_c_e51 : e51 = - rho3 * integral ( 19 ) / 2.0$ 

print_c_e52 : e52 = rho3 * integral ( 20 )$

print_c_e53 : e53 = temp 

print_c_e54$ 
kill ( e54 )$ 
batchload ( "subreduce2.com" )$
Translate the equations into Fortran codes.

writefile ("fortran3.lst");

fortran (print_t_element1_3);
fortran (print_t_element2_3);
fortran (print_t_element3_3);
fortran (print_t_element4_3);
fortran (print_t_element5_3);
fortran (print_t_element6_3);
fortran (print_t_element7_3);
fortran (print_t_e2);
fortran (print_t_e3);
fortran (print_t_e4);
fortran (print_t_e5);
fortran (print_t_e6);
fortran (print_t_e7);
fortran (print_t_e8);
fortran (print_t_e9);
fortran (print_t_e10);
fortran (print_t_e11);
fortran (print_t_e12);
fortran (print_t_e13);
fortran (print_torque1_3);
fortran (print_torque2_3);
fortran (print_torque3_3);
fortran (print_c_element1_3);
fortran (print_c_element2_3);
fortran (print_c_element3_3);
fortran (print_c_element4_3);
fortran (print_c_element5_3);
fortran (print_c_element6_3);
fortran (print_c_element7_3);
fortran (print_c_element8_3);
fortran (print_c_element9_3);
fortran (print_c_element10_3);
fortran (print_c_element11_3);
fortran (print_c_element12_3);
fortran (print_c_element13_3);
fortran (print_c_element14_3);
fortran (print_c_element15_3);
fortran (print_c_element16_3);
fortran (print_c_element17_3);
fortran (print_c_element18_3);
fortran (print_c_element19_3);
fortran (print_c_element20_3);
fortran (print_c_e14);
fortran (print_c_e15);
fortran (print_c_e16);
fortran (print_c_e17);
fortran (print_c_e18);
fortran (print_c_e19);
fortran (print_c_e21);
fortran (print_c_e22);
fortran (print_c_e23);
fortran (print_c_e24);
fortran (print_c_e25);
fortran (print_c_e26);
fortran (print_c_e27);
fortran (print_c_e28);
fortran (print_c_e29);
fortran ( print_c_e30 );
fortran ( print_c_e31 );
fortran ( print_c_e32 );
fortran ( print_c_e33 );
fortran ( print_c_e34 );
fortran ( print_c_e35 );
fortran ( print_c_e37 );
fortran ( print_c_e38 );
fortran ( print_c_e40 );
fortran ( print_c_e41 );
fortran ( print_c_e42 );
fortran ( print_c_e43 );
fortran ( print_c_e44 );
fortran ( print_c_e45 );
fortran ( print_c_e48 );
fortran ( print_c_e49 );
fortran ( print_c_e50 );
fortran ( print_c_e51 );
fortran ( print_c_e52 );
fortran ( print_c_e53 );
fortran ( print_c_e54 );
fortran ( print_coeffa1_3 );
fortran ( print_coeffb1_3 );
fortran ( print_coeffc1_3 );
fortran ( print_coeffd1_3 );
fortran ( print_coeffe1_3 );
fortran ( print_coefff1_3 );
fortran ( print_coeffa2_3 );
fortran ( print_coeffb2_3 );
fortran ( print_coeffc2_3 );
fortran ( print_coeffd2_3 );
fortran ( print_coeffe2_3 );
fortran ( print_coefff2_3 );
fortran ( print_coeffa3_3 );
fortran ( print_coeffb3_3 );
fortran ( print_coeffc3_3 );
fortran ( print_coeffd3_3 );
fortran ( print_coeffe3_3 );
fortran ( print_coefff3_3 );
fortran ( print_coeffa4_3 );
fortran ( print_coeffb4_3 );
fortran ( print_coeffc4_3 );
fortran ( print_coeffd4_3 );
fortran ( print_coeffe4_3 );
fortran ( print_coefff4_3 );
fortran ( print_coeffa5_3 );
fortran ( print_coeffb5_3 );
fortran ( print_coeffc5_3 );
fortran ( print_coeffd5_3 );
fortran ( print_coeffe5_3 );
fortran ( print_coefff5_3 );
closefile ();

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

        Mission accomplished, later...

*/

/* Program name : Reducec.com

This subroutine helps to simplify all the equations of link c.
Clear the memory.

\[
\text{kill ( all );}
\]

Load file into memory.

\[
\text{loadfile ( "pickapartc.fim" );}
\]

Simplify the torque equations.

\[
\text{temp : e2};
\text{kill ( e2 );}
\text{batchload ( "subreducec.com" );}
\text{batchload ( "subreduce2.com" );}
\text{print_t_e2 : e2 = temp;}
\]

Simplify the coefficient equations.

\[
\text{temp : e8};
\text{kill ( e8 );}
\text{batchload ( "subreducec.com" );}
\text{batchload ( "subreduce2.com" );}
\text{print_c_e8 : e8 = temp;}
\]
/*
 * temp : e12$
 * kill ( e12 )$
 * batchload ( "subreducec.com" )$
 * batchload ( "subreduce2.com" )$
 * print_c_e12 : e12 = temp$
 *
 * temp : e13$
 * kill ( e13 )$
 * batchload ( "subreducec.com" )$
 * batchload ( "subreduce2.com" )$
 * print_c_e13 : e13 = temp$
 *
 * temp : e14$
 * kill ( e14 )$
 * batchload ( "subreducec.com" )$
 * batchload ( "subreduce2.com" )$
 * print_c_e14 : e14 = temp$
 *
 * temp : e15$
 * kill ( e15 )$
 * batchload ( "subreducec.com" )$
 * batchload ( "subreduce2.com" )$
 * print_c_e15 : e15 = temp$
 *
 * temp : e19$
 * kill ( e19 )$
 * batchload ( "subreducec.com" )$
 * batchload ( "subreduce2.com" )$
 * print_c_e19 : e19 = temp$
 *
 * temp : e20$
 * kill ( e20 )$
 * batchload ( "subreducec.com" )$
 * batchload ( "subreduce2.com" )$
 * print_c_e20 : e20 = temp$
 */

Translate the equations into Fortran codes.
*/

writefile ( "fortranc.lst" );

fortran ( print_t_e2 );
fortran ( print_t_e3 );
fortran ( print_t_e4 );
fortran ( print_t_e5 );
fortran ( print_t_e6 );
fortran ( print_t_e7 );
fortran ( print_torque1_c );
fortran ( print_torque2_c );
fortran ( print_torque3_c );
fortran ( print_c_e8 );
fortran ( print_c_e12 );
fortran ( print_c_e13 );
fortran ( print_c_e14 );
fortran ( print_c_e15 );
fortran ( print_c_e19 );
fortran ( print_c_e20 );
fortran ( printCoeffa1_c );
fortran ( printCoeffb1_c );
fortran ( printCoeffc1_c );
fortran ( printCoeffd1_c );
fortran ( printCoeffe1_c );
fortran ( printCoefff1_c );

fortran ( printCoeffa2_c );
fortran ( printCoeffb2_c );
fortran ( printCoeffc2_c );
fortran ( printCoeffd2_c );
fortran ( printCoeffe2_c );
fortran ( printCoefff2_c );

fortran ( printCoeffa3_c );
fortran ( printCoeffb3_c );
fortran ( printCoeffc3_c );
fortran ( printCoeffd3_c );
fortran ( printCoeffe3_c );
fortran ( printCoefff3_c );

fortran ( printCoeffa4_c );
fortran ( printCoeffb4_c );
fortran ( printCoeffc4_c );
fortran ( printCoeffd4_c );
fortran ( printCoeffe4_c );
fortran ( printCoefff4_c );

fortran ( printCoeffa5_c );
fortran ( printCoeffb5_c );
fortran ( printCoeffc5_c );
fortran ( printCoeffd5_c );
fortran ( printCoeffe5_c );
fortran ( printCoefff5_c );

closefile ();

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

Mission accomplished, later...

*/

/* Program name : Reduceh3.com
This subroutine helps to simplify all the equations of link h3.

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

Clear the memory.

*/

kill ( all );

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

Set up the general variables for hydraulic cylinder 3.

*/

depends ( ah3pstart, [ qv2(t), th3(t) ] )$
depends ( ah3pend, [ qv2(t), th3(t) ] )$
depends ( ah3, [ qv2(t), th3(t) ] )$
depends ( eta, [ qv2(t), th3(t), qvh3(t) ] )$
depends ( thh3, [ th2(t), qv2(t), th3(t), qvh3(t) ], [ qv2(t), th3(t), ah3, eta ] )$

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

Load file into memory.
loadfile ("pickaparth3.fil");

/*
 *  ............................................................................................................................................................................
 * /

Simplify the torque equations.

/*

temp : part ( e2, 1, 2, 1, 1 )
batchload ("subreduceh3.com")
batchload ("subreduce2.com")
print_t_element1_h3 : element ( 1, i ) = temp$
temp : part ( e2, 1, 2, 2 )
batchload ("subreduceh3.com")
batchload ("subreduce2.com")
kil ( e2 )
print_t_e2 : e2 = rhoh3r * ( integral ( 1 ) + temp ) / 2.0$
temp : e3$
kil ( e3 )$
batchload ("subreduceh3.com")
print_t_e3 : e3 = temp$
temp : e4$
kil ( e4 )$
batchload ("subreduceh3.com")
print_t_e4 : e4 = temp$
temp : e5$
kil ( e5 )$
batchload ("subreduceh3.com")
print_t_e5 : e5 = temp$

/*
 *  ............................................................................................................................................................................
 * /

temp : part ( e6, 1, 2, 1, 1 )
batchload ("subreduceh3.com")
batchload ("subreduce2.com")
print_t_element2_h3 : element ( 2, i ) = temp$
temp : part ( e6, 1, 2, 2 )
batchload ("subreduceh3.com")
batchload ("subreduce2.com")
kil ( e6 )
print_t_e6 : e6 = rhoh3r * ( integral ( 2 ) + temp ) / 2.0$
temp : part ( e7, 1, 1, 2, 1 )
kil ( e7 )$
batchload ("subreduceh3.com")
batchload ("subreduce2.com")
print_t_element3_h3 : element ( 3, i ) = temp$
print_t_e7 : e7 = - rhoh3r * integral ( 3 ) / 2.0$
temp : part ( e8, 3, 1 )
kil ( e8 )$
batchload ("subreduceh3.com")
pri$$
kill (e11)
batchload ("subreduceh3.com")
print_t_e11 : e11 = temp

temp : e12
kill (e12)
batchload ("subreduceh3.com")
print_t_e12 : e12 = temp

temp : e13
kill (e13)
batchload ("subreduceh3.com")
print_t_e13 : e13 = temp

/*
 *-----------------------------------------
 */
temp : part (e14, 1, 1, 2, 1, 1)
batchload ("subreduceh3.com")
batchload ("subreduce2.com")
print_t_element5_h3 : element (5, i) = temp

temp : part (e14, 1, 1, 2, 2)
batchload ("subreduceh3.com")
batchload ("subreduce2.com")
kil (e14)
print_t_e14 : e14 = -rho3r * (integral (5) + temp) / 2.0

temp : part (e15, 3, 1, 1)
batchload ("subreduceh3.com")
batchload ("subreduce2.com")
print_t_element6_h3 : element (6, i) = temp

temp : part (e15, 3, 2)
batchload ("subreduceh3.com")
batchload ("subreduce2.com")
kil (e15)
print_t_e15 : e15 = g * rho3r * (integral (6) + temp)

temp : e16
kill (e16)
batchload ("subreduceh3.com")
print_t_e16 : e16 = temp

temp : e17
kill (e17)
batchload ("subreduceh3.com")
print_t_e17 : e17 = temp

temp : e18
kill (e18)
batchload ("subreduceh3.com")
print_t_e18 : e18 = temp

temp : e19
kill (e19)
print_t_e19 : e19 = temp

temp : e20
kill (e20)
batchload ("subreduceh3.com")
batchload ("subreduce2.com")
print_t_e20 : e20 = temp

/*
 *-----------------------------------------
 */

/* Simplify the coefficient equations.
*/
temp : part (e21, 1, 2, 1)
kil (e21)
batchload ("subreduceh3.com")
print_c_element1_h3 : element ( 1, i ) = temp$
print_c_e21 : e21 = rhoh3r * integral ( 1 ) / 2.0$

temp : e22$
kill ( e22 )$
batchload ( "subreduceh3.com" )$
print_c_e22 : e22 = temp$
/*
  ................................................................................................................................................
*/
temp : part ( e25, 1, 2, 1 )$
kill ( e25 )$
batchload ( "subreduceh3.com" )$
batchload ( "subreduce2.com" )$
print_c_element2_h3 : element ( 2, i ) = temp$
print_c_e25 : e25 = rhoh3r * integral ( 2 ) / 2.0$

temp : e26$
kill ( e26 )$
batchload ( "subreduceh3.com" )$
print_c_e26 : e26 = temp$
/*
  ................................................................................................................................................
*/
temp : part ( e27, 1, 2, 1, 1 )$
batchload ( "subreduceh3.com" )$
batchload ( "subreduce2.com" )$
print_c_element3_h3 : element ( 3, i ) = temp$
temp : part ( e27, 1, 2, 2 )$
batchload ( "subreduceh3.com" )$
batchload ( "subreduce2.com" )$
kil ( e27 )$
print_c_e27 : e27 = rhoh3r * ( integral ( 3 ) + temp ) / 2.0$

temp : part ( e28, 1, 1, 2, 1, 1 )$
batchload ( "subreduceh3.com" )$
batchload ( "subreduce2.com" )$
print_c_element4_h3 : element ( 4, i ) = temp$
temp : part ( e28, 1, 1, 2, 2 )$
batchload ( "subreduceh3.com" )$
batchload ( "subreduce2.com" )$
kil ( e28 )$
print_c_e28 : e28 = - rhoh3r * ( integral ( 4 ) + temp ) / 2$

temp : part ( e29, 3, 1, 1 )$
batchload ( "subreduceh3.com" )$
print_c_element5_h3 : element ( 5, i ) = temp$
temp : part ( e29, 3, 2 )$
batchload ( "subreduceh3.com" )$
kil ( e29 )$
print_c_e29 : e29 = g * rhoh3r * integral ( 5 ) + temp$

temp : e30$
kil ( e30 )$
batchload ( "subreduceh3.com" )$
print_c_e30 : e30 = temp$

temp : e31$
kil ( e31 )$
batchload ( "subreduceh3.com" )$
print_c_e31 : e31 = temp$

temp : e32$
kil ( e32 )$
batchload ( "subreduceh3.com" )$
print_c_e32 : e32 = temp$

temp : e33$
kil ( e33 )$
batchload ( "subreduceh3.com" )$
print_c_e33 : e33 = temp$

temp : e34$
kill ( e34 )$
batchload ( "subreduceh3.com" )$
print_c_e34 : e34 = temp$

temp : e35$
kill ( e35 )$
batchload ( "subreduceh3.com" )$
print_c_e35 : e35 = temp$

temp : e36$
kill ( e36 )$
batchload ( "subreduceh3.com" )$
print_c_e36 : e36 = temp$

temp : e37$
kill ( e37 )$
batchload ( "subreduceh3.com" )$
batchload ( "subreduce2.com" )$
print_c_e37 : e37 = temp$

/*  ............................................................................................................................................................................ *

 temp : part ( e38, 3, 1 )$
 kill ( e38 )$
batchload ( "subreduceh3.com" )$
 print_c_element6_h3 : element ( 6, i ) = temp$
 print_c_e38 : e38 = gw2a2psq * rhoh3r * integral ( 6 )$
 kill ( e39 )$
 print_c_e39 : e39 = gwh3ah3psq$

/*  ............................................................................................................................................................................ *

 temp : part ( e42, 1, 4, 1 )$
 kill ( e42 )$
batchload ( "subreduceh3.com" )$
 print_c_element7_h3 : element ( 7, i ) = temp$
 print_c_e42 : e42 = eih3 * gw2a2psq * qu2 * integral ( 7 ) / gwh3ah3psq$

 temp : part ( e43, 1, 1, 2, 1 )$
 kill ( e43 )$
batchload ( "subreduceh3.com" )$
batchload ( "subreduce2.com" )$
 print_c_element8_h3 : element ( 8, i ) = temp$
 print_c_e43 : e43 = - rhoh3r * integral ( 8 ) / 2$

 temp : part ( e44, 1, 3, 1, 1 )$
 batchload ( "subreduceh3.com" )$
 print_c_element9_h3 : element ( 9, i ) = temp$
 temp : part ( e44, 1, 3, 2 )$
 batchload ( "subreduceh3.com" )$
 kill ( e44 )$
 print_c_e44 : e44 = gw2a2p * rhoh3r * ( integral ( 9 ) + temp ) / gwh3ah3p$

/*  ............................................................................................................................................................................ *

 temp : part ( e60, 1, 2, 1 )$
 kill ( e60 )$
batchload ( "subreduceh3.com" )$
 print_c_element10_h3 : element ( 10, i ) = temp$
 print_c_e60 : e60 = rhoh3r * integral ( 10 ) / 2.0$

 temp : e61$
 kill ( e61 )$
Translate the equations into Fortran codes.

writefile ("fortranh3.lst");

fortran (print_t_element1_h3);
fortran (print_t_element2_h3);
fortran (print_t_element3_h3);
fortran (print_t_element4_h3);
fortran (print_c_e69);
fortran (print_c_e70);

fortran (print_coeffa1_h3);
fortran (print_coeffb1_h3);
fortran (print_coeffc1_h3);
fortran (print_coeffd1_h3);
fortran (print_coeffe1_h3);
fortran (print_coefff1_h3);

fortran (print_coeffa2_h3);
fortran (print_coeffb2_h3);
fortran (print_coeffc2_h3);
fortran (print_coeffd2_h3);
fortran (print_coeffe2_h3);
fortran (print_coefff2_h3);

fortran (print_coeffa3_h3);
fortran (print_coeffb3_h3);
fortran (print_coeffc3_h3);
fortran (print_coeffd3_h3);
fortran (print_coeffe3_h3);
fortran (print_coefff3_h3);

fortran (print_coeffa4_h3);
fortran (print_coeffb4_h3);
fortran (print_coeffc4_h3);
fortran (print_coeffd4_h3);
fortran (print_coeffe4_h3);
fortran (print_coefff4_h3);

fortran (print_coeffa5_h3);
fortran (print_coeffb5_h3);
fortran (print_coeffc5_h3);
fortran (print_coeffd5_h3);
fortran (print_coeffe5_h3);
fortran (print_coefff5_h3);

closefile();
/
Mission accomplished, later...
/

/* Program name: Save1.com

This command procedure file helps to create output listings and saves the results of link 1.

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

filename1: "$link1.lst$
filename2: "$coeff1.lst$
batch ("listing.com")
save ("link1.fil",
coeffa1, coeffb1, coeffc1, coeffd1, coeffe1, coefff1,
coeffa2, coeffb2, coeffc2, coeffd2, coeffe2, coefff2,
coeffa3, coeffb3, coeffc3, coeffd3, coeffe3, coefff3,
coeffa4, coeffb4, coeffc4, coeffd4, coeffe4, coefff4,
coeffa5, coeffb5, coeffc5, coeffd5, coeffe5, coefff5,
line1, line2, line3, line4, line5, line6, line7, line8)
/
*******************************************************************************/
That's it folks...

/* Program name : Save2.com

This command procedure file helps to create output listings and saves the results of link 2.

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

filename1 : "link2.lst$
filename2 : "coeff2.lst$

batch ( "listing.com")$
save ( "link2.fil",
coeffa1, coeffb1, coeffc1, coeffd1, coeffe1, coefff1,
coeffa2, coeffb2, coeffc2, coeffd2, coeffe2, coefff2,
coeffa3, coeffb3, coeffc3, coeffd3, coeffe3, coefff3,
coeffa4, coeffb4, coeffc4, coeffd4, coeffe4, coefff4,
coeffa5, coeffb5, coeffc5, coeffd5, coeffe5, coefff5,
line1, line2, line3, line4, line5, line6, line7, line8 )$

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

That's it folks...

*/

/* Program name : Save3.com

This command procedure file helps to create output listings and saves the results of link 3.

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

filename1 : "link3.lst$
filename2 : "coeff3.lst$

batch ( "listing.com")$
save ( "link3.fil",
coeffa1, coeffb1, coeffc1, coeffd1, coeffe1, coefff1,
coeffa2, coeffb2, coeffc2, coeffd2, coeffe2, coefff2,
coeffa3, coeffb3, coeffc3, coeffd3, coeffe3, coefff3,
coeffa4, coeffb4, coeffc4, coeffd4, coeffe4, coefff4,
coeffa5, coeffb5, coeffc5, coeffd5, coeffe5, coefff5,
line1, line2, line3, line4, line5, line6, line7, line8 )$

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

That's it folks...

*/

/* Program name : Savec.com

This command procedure file helps to create output listings and saves the results of link c.

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

кажется, что вы не хотите, чтобы я прочитал текст этой страницы. Но если вы хотите, я могу попытаться помочь вам. Но если вы хотите, я могу попытаться помочь вам. Вот текст в формате текстового файла:

```
/* Program name : Save2.com

This command procedure file helps to create output listings and
saves the results of link 2.

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

filename1 : "link2.lst$
filename2 : "coeff2.lst$

batch ( "listing.com")$
save ( "link2.fil",
coeffa1, coeffb1, coeffc1, coeffd1, coeffe1, coefff1,
coeffa2, coeffb2, coeffc2, coeffd2, coeffe2, coefff2,
coeffa3, coeffb3, coeffc3, coeffd3, coeffe3, coefff3,
coeffa4, coeffb4, coeffc4, coeffd4, coeffe4, coefff4,
coeffa5, coeffb5, coeffc5, coeffd5, coeffe5, coefff5,
line1, line2, line3, line4, line5, line6, line7, line8 )$

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

That's it folks...

*/

/* Program name : Save3.com

This command procedure file helps to create output listings and
saves the results of link 3.

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

filename1 : "link3.lst$
filename2 : "coeff3.lst$

batch ( "listing.com")$
save ( "link3.fil",
coeffa1, coeffb1, coeffc1, coeffd1, coeffe1, coefff1,
coeffa2, coeffb2, coeffc2, coeffd2, coeffe2, coefff2,
coeffa3, coeffb3, coeffc3, coeffd3, coeffe3, coefff3,
coeffa4, coeffb4, coeffc4, coeffd4, coeffe4, coefff4,
coeffa5, coeffb5, coeffc5, coeffd5, coeffe5, coefff5,
line1, line2, line3, line4, line5, line6, line7, line8 )$

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

That's it folks...

*/

/* Program name : Savec.com

This command procedure file helps to create output listings and
saves the results of link c.

*******************************************************************************/
```
/* Program name: Saveh3.com

This command procedure file helps to create output listings and
saves the results of link h3.

*******************************************************************************
*/
filename1: "linkh3.lst"
filename2: "coeffh3.lst"
batch ( "listing.com" )$
save ( "linkh3.fil",
    coeffa1, coeffb1, coeffc1, coeffd1, coeffe1, coefff1,
    coeffa2, coeffb2, coeffc2, coeffd2, coeffe2, coefff2,
    coeffa3, coeffb3, coeffc3, coeffd3, coeffe3, coefff3,
    coeffa4, coeffb4, coeffc4, coeffd4, coeffe4, coefff4,
    coeffa5, coeffb5, coeffc5, coeffd5, coeffe5, coefff5,
    line1, line2, line3, line4, line5, line6, line7, line8 )$

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

That's it folks...

*/

/* Program name: Simplify1.com

This subroutine is targeted at the equations of link 1. It helps
replace the derivatives of the time-dependent generalized
coordinate with their variable names.

*******************************************************************************
*/
temp : line1$
batchload ( "subsimp1.com" )$
line1 : temp$
temp : line2$
batchload ( "subsimp1.com" )$
line2 : temp$
temp : line3$
batchload ( "subsimp1.com" )$
line3 : temp$
temp : line4$
batchload ( "subsimp1.com" )$
line4 : temp$

temp : line5$
batchload ( "subsimp1.com" )$
line5 : temp$

temp : line6$
batchload ( "subsimp1.com" )$
line6 : temp$

temp : line7$
batchload ( "subsimp1.com" )$
line7 : temp$

temp : line8$
batchload ( "subsimp1.com" )$
line8 : temp$

/*
Mission accomplished, see you...
*/

/* Program name : Simplify2.com

This subroutine is targeted at the equations of link 2. It helps to replace the derivatives of the time-dependent generalize coordinate with their variable names.

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

*/

batchload ( "subsimp1.com" )$
batchload ( "subsimp2.com" )$
line1 : temp$

batchload ( "subsimp1.com" )$
batchload ( "subsimp2.com" )$
line2 : temp$

batchload ( "subsimp1.com" )$
batchload ( "subsimp2.com" )$
line3 : temp$

batchload ( "subsimp1.com" )$
batchload ( "subsimp2.com" )$
line4 : temp$

batchload ( "subsimp1.com" )$
batchload ( "subsimp2.com" )$
line5 : temp$

batchload ( "subsimp1.com" )$
batchload ( "subsimp2.com" )$
line6 : temp$

batchload ( "subsimp1.com" )$
batchload ( "subsimp2.com" )$
line7 : temp$

Mission accomplished, see you...

Program name: Simplify3.com

This subroutine is targeted at the equations of link 3. It helps to replace the derivatives of the time-dependent generalized coordinate with their variable names.

******************************************************************************
Mission accomplished, see you...

/* Program name : Simplifyc.com

This subroutine is targeted at the equations of link c. It helps to replace the derivatives of the time-dependent generalize coordinate with their variable names.

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

temp : line1$
batchload ( "subsimp1.com" )$
batchload ( "subsimp2.com" )$
batchload ( "subsimp3.com" )$
line1 : temp$

temp : line2$
batchload ( "subsimp1.com" )$
batchload ( "subsimp2.com" )$
batchload ( "subsimp3.com" )$
line2 : temp$

temp : line3$
batchload ( "subsimp1.com" )$
batchload ( "subsimp2.com" )$
batchload ( "subsimp3.com" )$
line3 : temp$

temp : line4$
batchload ( "subsimp1.com" )$
batchload ( "subsimp2.com" )$
batchload ( "subsimp3.com" )$
line4 : temp$

temp : line5$
batchload ( "subsimp1.com" )$
batchload ( "subsimp2.com" )$
batchload ( "subsimp3.com" )$
line5 : temp$

temp : line6$
batchload ( "subsimp1.com" )$
batchload ( "subsimp2.com" )$
batchload ( "subsimp3.com" )$
line6 : temp$

temp : line7$
batchload ( "subsimp1.com" )$
batchload ( "subsimp2.com" )$
batchload ( "subsimp3.com" )$
line7 : temp$

temp : line8$
batchload ( "subsimp1.com" )$
batchload ( "subsimp2.com" )$
batchload ( "subsimp3.com" )$
line8 : temp$

MISSION ACCOMPLISHED, SEE YOU...
/* Program name: Simplifyh3.com

This subroutine is targeted at the equations of link h3. It helps to replace the derivatives of the time-dependent generalized coordinate with their variable names.

*******************************************************************
*/
temp : line1
batchload ("subsimp1.com")
batchload ("subsimp2.com")
batchload ("subsimp3.com")
batchload ("subsimph3.com")
line1 : temp

temp : line2
batchload ("subsimp1.com")
batchload ("subsimp2.com")
batchload ("subsimp3.com")
line2 : temp

temp : line3
batchload ("subsimp1.com")
batchload ("subsimp2.com")
batchload ("subsimp3.com")
line3 : temp

temp : line4
batchload ("subsimp1.com")
batchload ("subsimp2.com")
batchload ("subsimp3.com")
line4 : temp

temp : line5
batchload ("subsimp1.com")
batchload ("subsimp2.com")
batchload ("subsimp3.com")
line5 : temp

temp : line6
batchload ("subsimp1.com")
batchload ("subsimp2.com")
batchload ("subsimp3.com")
line6 : temp

temp : line7
batchload ("subsimp1.com")
batchload ("subsimp2.com")
batchload ("subsimp3.com")
line7 : temp

temp : line8
batchload ("subsimp1.com")
batchload ("subsimp2.com")
batchload ("subsimp3.com")
line8 : temp

/* Mission accomplished, see you...*/
/* Program name : Subreduco2.com 

This is the command procedure routine that helps to reduce and simplify the current link 2's equation in equation 'temp'. 

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

*: subst ( thldsq, th1dA 2, temp > $ 
*: subst ( a2sq, a2rA 2, temp )$ 
*: subst ( a2rsq, a2rA 2, temp )$ 
*: subst <  a2rcu, a2rA 3, temp )$ 
*: subst ( costh2, cos ( th2(t) ), temp )$ 
*: subst ( costh2sq, costh2A 2, temp )$ 
*: subst ( cos2th2, cos ( 2 * th2(t) ), temp )$ 
*: subst ( sinth2, sin ( th2(t) ), temp )$ 
*: subst ( sin2th2, sin ( 2 * th2(t) ), temp )$ 
*: subst ( th2dsq, th2dA 2, temp )$ 
*: subst ( qv2, qv2(t), temp )$ 
*: subst ( qv2sq, qv2r2, temp )$ 
*: subst ( qv2dsq, qv2d2, temp )$ 
*: subst ( qw2, qw2(t), temp )$ 
*: subst ( qw2sq, qw2r2, temp )$ 
*: subst ( qw2dsq, qw2d2, temp )$ 
*: subst ( gw2ppsq(i), ( diff ( gw2(u2), u2, 2  ) )A 2, temp )S 
*: subst ( gw2(i), gw2(u2), temp )$ 
*: subst ( gw2sq(i), gw2(i)A 2, temp )$ 
*: subst ( gw2a2sq, gw2a2r2, temp )$ 
*: subst ( gw2(i), gw2(u2), temp )$ 
*: subst ( gw2sq(i), gw2(i)A 2, temp )$ 
*: subst ( gw2a2sq, gw2a2r2, temp )$ 
*: subst ( u2(i), u2, temp )$ 

*: subst ( l2sub(1), 
*: ( costh2 * u2(i) - sinh2 * qv2 * gv2(i) ), temp )$ 
*: subst ( -l2sub(1), 
*: ( sinh2 * qv2 * gv2(i) - costh2 * u2(i) ), temp )$ 
*: subst ( l2sub(2), 
*: ( costh2 * gw2(i) * qv2 * thld - gv2(i) * qv2 * th2d ), 
*: temp )$ 
*: subst ( l2sub(3), 
*: ( sinh2 * thld * u2(i) + costh2 * gv2(i) * qv2 * th1d ), 
*: temp )$ 
*: subst ( l2sub(4), 
*: ( -cos2th2 * th1d * u2(i) + gv2(i) * qv2 * sinh2 * th1d + 
*: gw2(i) * qv2d ), temp )$ 
*: subst ( l2sub(5), 
*: ( sinh2 * th2d * u2(i) + costh2 * gv2(i) * qv2 * th2d + 
*: gv2(i) * qv2d + sinh2 ), temp )$ 
*: subst ( l2sub(6), 
*: ( th2d * u2(i) - gw2(i) * qv2 * sinh2 * th1d + 
*: gv2(i) * qv2d ), temp )$ 

*: subst ( l2sub(1), 
*: ( costh2 * a2 - sinh2 * qv2 * gv2a2 ), temp )$ 
*: subst ( -l2sub(1), 
*: ( sinh2 * qv2 * gv2a2 - costh2 * a2 ), temp )$ 
*: subst ( l2sub(2), 
*: ( costh2 * gw2a2 * qv2 * th1d - gv2a2 * qv2 * th2d ), 
*: temp )$ 
*: subst ( l2sub(3), 
*: ( sinh2 * th1d * a2 + costh2 * gv2a2 * qv2 * th1d ), 
*: temp )$ */
temp : subst ( l2sub(4),
( -cosh^2 * th1d * a2 + gv2a2 * qv2 * sinh^2 * th1d +
gw2a2 * qw2d ), temp )$

temp : subst ( l2sub(5),
( sinh2 * th2d * a2 + cosh2 * gv2a2 * qv2 * th2d +
gv2a2 * qv2d * sinh2 ), temp )$

temp : subst ( l2sub(6),
( th2d * a2 - gw2a2 * qw2 * sinh2 * th1d +
gv2a2 * qv2d ), temp )$

/* **************************************************************************
Mission accomplished, see you...
*/

/* Program name : Subreduce3.com

This is the command procedure routine that helps to reduce and
simplify the current link 3's equation in equation 'temp'.

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

temp : subst ( psid, th3d +  gv2a2p * qv2d, temp )$

temp : subst ( cospsi, cos ( th3(t) +  gv2a2p * qv2 ), temp )$

temp : subst ( cospsisq, cospsi^2, temp )$

temp : subst ( sinpsi, sin ( th3(t) + gv2a2p * qv2 ), temp )$

temp : subst ( sinpsisq, sinpsi^2, temp )$

temp : subst ( chi, th3(t) + th2(t) +  gv2a2p * qv2, temp )$

temp : subst ( chid, th3d + th2d +  gv2a2p * qv2d, temp )$

temp : subst ( chidd, th3dd + th2dd +  gv2a2p * qv2dd, temp )$

temp : subst ( coschi, cos ( chi ), temp )$

temp : subst ( sinchi, sin ( chi ), temp )$

temp : subst ( th3dsq, th3d^2, temp )$

temp : subst ( th3ddth2dd, th3dd + th2dd, temp )$

temp : subst ( qv3, qv3(t), temp )$

temp : subst ( qv3sq, qv3^2, temp )$

temp : subst ( qv3dsq, qv3d^2, temp )$

temp : subst ( qv3ddsq, qv3d^2, temp )$

temp : subst ( gv3sq(i), ( diff ( gv3(u3), u3, 2 ) )^2, temp )$

temp : subst ( gv3(i), gv3(u3), temp )$

temp : subst ( gw3sq(i), ( diff ( gw3(u3), u3, 2 ) )^2, temp )$

temp : subst ( gw3(i), gw3(u3), temp )$

temp : subst ( u3(i), u3, temp )$

/* Program name : Subreduce3.com

This is the command procedure routine that helps to reduce and
simplify the current link 3's equation in equation 'temp'.

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

temp : subst ( l3sub(1),
( cospsi * qv3 + gv3(i) * sinpsi * u3(i) ), temp )$

temp : subst ( l3sub(2), l3sub(1)+2, temp )$

temp : subst ( l3sub(3),
( -cospisi + u3(i) - sinpsi * qv3 * gv3(i) ), temp )$

temp : subst ( l3sub(4),
( -cospsi * qv2a2p * u3(i) -
\( \text{temp} : \text{subst} \ ( -l3sub(4), \ ( -cospsi * gv2a2p * u3(i) + \\
\text{gv2a2p} * gv3(i) * qv3 * sinpsi ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(5), \ ( -cospsi * sinpsi * u3(i) + \\
\text{cospsi} * gv2a2p * qv3(i) * qv3 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(6), \ ( -cospsi * sinpsi * u3(i) - \\
\text{cospsi} * gv2a2p * qv3(i) * qv3 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(7), \ \text{psid} * \text{sinpsi} * u3(i) + \text{sinpsi} * qv3(i) * qv3d + \\
\text{psid} * \text{cospsi} * gv3(i) * qv3 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(8), \ ( -cospsi * u3(i) - \text{cospsi} * qv3(i) * qv3 - \\
\text{gv2a2p} * qv2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(9), \ ( -cospsi * u3(i) - qv3(i) * qv3 * sinpsi + a2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(10), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(11), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(12), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(13), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(14), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(15), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(16), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(17), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(18), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(19), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(20), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(21), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)

\[
\text{temp} : \text{subst} \ ( -l3sub(4), \ ( -cospsi * gv2a2p * u3(i) + \\
\text{gv2a2p} * gv3(i) * qv3 * sinpsi ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(5), \ ( -cospsi * sinpsi * u3(i) + \\
\text{cospsi} * gv2a2p * qv3(i) * qv3 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(6), \ ( -cospsi * sinpsi * u3(i) - \\
\text{cospsi} * gv2a2p * qv3(i) * qv3 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(7), \ \text{psid} * \text{sinpsi} * u3(i) + \text{sinpsi} * qv3(i) * qv3d + \\
\text{psid} * \text{cospsi} * gv3(i) * qv3 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(8), \ ( -cospsi * u3(i) - \text{cospsi} * qv3(i) * qv3 - \\
\text{gv2a2p} * qv2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(9), \ ( -cospsi * u3(i) - qv3(i) * qv3 * sinpsi + a2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(10), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(11), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(12), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(13), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(14), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(15), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(16), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(17), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(18), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(19), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(20), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
\( \text{temp} : \text{subst} \ ( l3sub(21), \ ( -cosh2 * gw3(i) * qw3 - \text{cosh2} * gw2a2p * qw2 ), \ temp ) \)
temp : subst ( l3sub(22),
( a2 * th2d - gw3(i) * qw3 * sinh2 * thld -
gw2a2 * qw2 * sinh2 * thld + cospsi * gw3(i) * qw3d +
gw2a2 * qw2d + l3sub(3) * chid ), temp )$

temp : subst ( l3sub(23),
( gw3(i) * sinh2 * thld +
gw3(i) * gw2a2p * qw2d * sinpsi ), temp )$

temp : subst ( l3sub(24),
( -coschi * u3(i) + gw2a2 * qw2 * sinh2 +
gw3(i) * qv3 * sinh2 - a2 * costh2 ), temp )$

temp : subst ( -l3sub(24),
( gw3(i) * sinchi * thld +
gw3(i) * gw2a2p * qw2d * sinpsi ), temp )$

temp : subst ( l3sub(25),
( -gw3(i) * qw3 * sinh2 * thld -
gw2a2 * qw2 * sinh2 * thld ), temp )$

temp : subst ( l3sub(26),
( gw3(i) * qv3 * costh2 * thld -
gw2a2 * qw2 * costh2 * thld ), temp )$

temp : subst ( l3sub(27),
( -sinpsi * gw3(i) * qw3d - l3sub(1) * chid ), temp )$

temp : subst ( l3sub(28),
( -cospsi * gw3(i) * qw3d - l3sub(3) * chid ), temp )$

temp : subst ( l3sub(29),
( -gw3(i) * qw3 * sinh2 * th2d -
gw2a2 * qw2 * sinh2 * th2d +
costh2 * gw3(i) * qw3d +
costh2 * gw2a2 * qw2d ), temp )$

temp : subst ( l3sub(30),
( -cospsi * gw3(i) * qw3 * th2d -
costh2 * gw2a2 * qw2 * th2d -
gw3(i) * qw3d * sinh2 -
gw2a2 * qw2d * sinh2 ), temp )$

temp : subst ( l3sub(1),
( cospsi * qw3 * gw3a3 * sinpsi * a3 ), temp )$

temp : subst ( -l3sub(1),
( -cospsi * qw3 * gw3a3 - sinpsi * a3 ), temp )$

temp : subst ( l3sub(2), l3sub(1) * 2, temp )$

temp : subst ( l3sub(3),
( cospsi * a3 - sinpsi * gw3 * gw3a3 ), temp )$

temp : subst ( l3sub(12),
( cospsi * gw2a2p * a3 -
gw2a2p * gw2a2 * gw3a3 * qw3 ), temp )$

temp : subst ( l3sub(4),
( cospsi * gw2a2p * a3 -
gw2a2p * gw2a2 * gw3a3 * qw3 * sinpsi ), temp )$

temp : subst ( -l3sub(5),
( gw2a2p * sinpsi * a3 +
cospsi * gw2a2p * gw3a3 * qw3 ), temp )$

temp : subst ( -l3sub(5),
( -gw2a2p * sinpsi * a3 -
cospsi * gw2a2p * gw3a3 * qw3 ), temp )$

temp : subst ( l3sub(6),
( psid * sinpsi * a3 + sinpsi * gw3a3 * qw3d +
psid * cospsi * gw3a3 * qw3 ), temp )$

temp : subst ( -l3sub(6),
( -psid * sinpsi * a3 - sinpsi * gw3a3 * qw3d -
psid * cospsi * gw3a3 * qw3 ), temp )$

temp : subst ( l3sub(7),
( psid * cospsi * a3 + cospsi * gw3a3 * qw3d -
psid * sinpsi * gw3a3 * qw3 ), temp )$

temp : subst ( -l3sub(7),
( -psid * cospsi * a3 - cospsi * gw3a3 * qw3d +
psid * sinpsi * gw3a3 * qw3 ), temp )$

temp : subst ( l3sub(8),
( sinpsi * a3 + cospsi * gw3a3 * qw3 + gw2a2 * qw2 ),
temp )$
temp : subst (-I3sub(8),
(-sinpsi * a3 - cospsi * gv3a3 * qv3 -
-gv2a2 * qv2), temp )$

temp : subst (I3sub(9),
(cospsi * a3 - gv3a3 * qv3 - sinpsi + a2 ), temp )$

temp : subst (-I3sub(9),
(-cospsi * a3 + gv3a3 * qv3 * sinpsi - a2 ),
temp )$

temp : subst (I3sub(10),
(costh2 * gw3a3 * qw3 + costh2 * gw2a2 * qw2 ), temp )$

temp : subst (-I3sub(10),
(-costh2 * gw3a3 * qw3 - costh2 * gw2a2 * qw2 ),
temp )$

temp : subst (I3sub(11),
(-sinth2 * gw3a3 * qw3 + sinth2 * gw2a2 * qw2 ), temp )$

temp : subst (-I3sub(11),
(-sinth2 * gw3a3 * qw3 - sinth2 * gw2a2 * qw2 ),
temp )$

temp : subst (I3sub(12),
(coschi * a3 - sinchi * qv3 * gv3a3 ), temp )$

temp : subst (-I3sub(12),
(-coschi * a3 + sinchi * qv3 * gv3a3 ), temp )$

temp : subst (I3sub(13), (I3sub(3) + I3sub(2)), temp )$

temp : subst (I3sub(14), (I3sub(13))^2, temp )$

temp : subst (-I3sub(15),
(-gv2a2 * th2d - cospsi * gw2a2p * gv3a3 * qv3d -
I3sub(4) * chid ), temp )$

temp : subst (I3sub(16),
(-gv2a2p * gv3a3 * qv3d * sinpsi -
I3sub(5) * chid ), temp )$

temp : subst (I3sub(17),
(coschi * gw2a2p * a3 -
gv2a2p * gv3a3 * qv3 * sinchi +
costh2 * gw2a2p ), temp )$

temp : subst (I3sub(18),
(-cospsi * gw2a2p * a3 +
gv3a3 * gw2a2p * qv3 * sinpsi +
gw2a2p ), temp )$

temp : subst (I3sub(19), (I3sub(18))^2, temp )$

temp : subst (I3sub(20),
(gw2a2p * psid * sinpsi * a3 +
gv3a3 * gw2a2p + qv3d * sinpsi +
cospsi * gv3a3 + gw2a2p * psid * qv3 ), temp )$

temp : subst (I3sub(21),
(-gv2a2 * qv2 + th2d + costh2 * gw2a2 + qv2 * thd -
gv3a3 * qv3 * sinpsi + omegasubl * gw3a3 * qw3 -
I3sub(1) * chid ), temp )$

temp : subst (I3sub(22),
(a2 * th2d - gw3a3 * qw3 * sinth2 * thd -
gw2a2 * qw2 * sinth2 * thd + cospsi * gv3a3 * qv3d +
gv2a2 * qv2d + I3sub(3) * chid ), temp )$

temp : subst (I3sub(23),
(gv3a3 * sinchi * th1d +
gv3a3 * gw2a2p * qw2d * sinpsi ), temp )$

temp : subst (-I3sub(24),
(-coschi * a3 + gw2a2 * qv2 * sinth2 +
gv3a3 * qv3 * sinchi - a2 * costh2 ), temp )$

temp : subst (-I3sub(24),
(coschi * a3 - gw2a2 * qv2 * sinth2 -
gv3a3 * qv3 * sinchi + a2 * costh2 ), temp )$

temp : subst (I3sub(25),
(-gw3a3 * qw3 * sinth2 * th1d -
gw2a2 * qv2 * sinth2 * th1d ), temp )$

temp : subst (I3sub(26),
(-gw3a3 * qv3 * costh2 * th1d -
gw2a2 * qw2 * costh2 * th1d ), temp )$

temp : subst (I3sub(27),
(-sinpsi * gw3a3 * qv3d - I3sub(1) * chid ), temp )$

temp : subst (I3sub(28),
(-cospsi * gv3a3 * qv3d - I3sub(3) * chid ), temp )$

temp : subst (I3sub(29),
( -gw3a3 * qw3 * sinh2 * th2d -
gw2a2 * qw2 * sinh2 * th2d +
costh2 * gw3a3 * qw3d +
costh2 * gw2a2 * qw2d ), temp )$

temp : subst ( l3sub(30),
( -costh2 * gw3a3 * qw3 * th2d -
costh2 * gw2a2 * qw2 * th2d -
gw3a3 * qw3d * sinh2 -
gw2a2 * qw2d * sinh2 ), temp )$

/* *****************************
Mission accomplished, see you...
*/

/* Program name : Subreducec.com
This is the command procedure routine that helps to reduce and
simplify the current link c's equation in equation 'temp'.
***************************************************************************/

temp : subst ( acsq, ac2, temp )$
temp : subst ( accu, ac3, temp )$
temp : subst ( psid, th3d + gv2a2p * qv2d, temp )$
temp : subst ( 2 * psid, ( 2 * th3d + 2 * gv2a2p * qv2d ), temp )$
temp : subst ( chid, th3d + th2d + gv2a2p * qv2d, temp )$
temp : subst ( 2 * chid,
( 2 * th3d + 2 * th2d + 2 * gv2a2p * qv2d ), temp )$
temp : subst ( l1sub(1),
( 2 * th3d + th2d + 2 * gv2a2p * qv2d ), temp )$
temp : subst ( l1sub(2),
( th3d + 2 * th2d + gv2a2p * qv2d ), temp )$
temp : subst ( l1sub(3),
( th3d - th2d + gv2a2p * qv2d ), temp )$
temp : subst ( l1sub(4),
( 2 * a2 * acsq * gw2a2p - 6 * acsq * gw2a2 ), temp )$
temp : subst ( l1sub(5),
( 2 * accu * gw2a2p - 12 * a2 * ac * gw2a2 ), temp )$
temp : subst ( l1sub(6),
( 3 * a2'2 * ac - 3 * ac * gw2a2'2 * qv2(t)'2 ), temp )$
temp : subst ( l1sub(7),
( 12 * ac * gw2a2'2 * qw(t) + qv2d +
6 * ac * gw2a2'2 * qv2(t) * qv2d ), temp )$
temp : subst ( l1sub(8),
( 6 * ac * gw2a2'2 * qv2(t)'2 +
3 * ac * gw2a2'2 * qv2(t)''2 + accu + 3 * a2'2 * ac ),
temp )$
temp : subst ( l1sub(9),
( -12 * a2 * ac * gw2a2 * qw(t) * sin(th2(t)) -
12 * ac * cos(th2(t)) * gw2a2 * gw2a2 * qw2(t) * qv2(t) ), temp )$
temp : subst ( l1sub(10),
( 12 * ac * gw2a2 * gw2a2 * qw2(t) * qw2d -
12 * ac * gw2a2 * gw2a2 * qv2d * qw2(t) ), temp )$
temp : subst ( l1sub(11),
( 12 * ac * gw2a2 * gw2a2 * qw2(t) * qw2dd -
12 * ac * gw2a2 * gw2a2 * qv2dd * qv2(t) ), temp )$
temp : subst ( l1sub(12),
( -6 * acsq * gw2a2 * qv2(t) * th3d -
6 * acsq * gw2a2 * qv2(t) * th2d -
3 * acsq * gw2a2 * gw2a2p * qv2(t) * qv2d -
6 * acsq * gw2a2p * gw2a2 * qv2d * qv2(t) ), temp )$
temp : subst ( l1sub(13),
( 12 * ac * gw2a2 * gw2a2 * qw(t) * qw2(t) * sin(th2(t)) * th2d -
12 * ac * gw2a2 * gw2a2 * qv2(t) * gw2(t) * sin(th2(t)) * th2d -
nth2 * th2d -
6 * gw2a2p * qv2(t) * gw2a2p * qv2d * qv2(t) ), temp )$

12 * a2 * ac * cos(th2(t)) * gw2a2 * qw2(t) * th2d -
12 * a2 * ac * gw2a2 * qw2d * sin(th2(t)) -
12 * ac * cos(th2(t)) * gw2a2 * gw2a2 * qw2(t) * qw2d -
12 * ac * cos(th2(t)) * gw2a2 * gw2a2 * qw2d *
qw2(t) ), temp )$

temp : subst ( lcsub(14),
( -6 * acsq * gw2a2 * qw2(t) * th3dd -
6 * acsq * gw2a2 * gw2d * th3d -
6 * acsq * gw2a2 * qw2(t) * th2dd -
6 * acsq * gw2a2 * gw2d * th2d -
3 * acsq * gw2a2p * gw2a2p * qw2(t) * qw2dd -
3 * acsq * gw2a2p * gw2a2p * qw2d * qw2d -
6 * acsq * gw2a2p * gw2a2 * qw2d * qw2d -
6 * acsq * gw2a2p * gw2a2 * qw2dd * qw2(t) ), temp )$

temp : subst ( lcsub(15),
sin ( 2 * zeta + 2 * th3(t) + 2 * th2(t) +
2 * gw2a2p * qw2(t) ), temp )$

temp : subst ( lcsub(16),
cos ( 2 * zeta + 2 * th3(t) + 2 * th2(t) +
2 * gw2a2p * qw2(t) ), temp )$

temp : subst ( lcsub(17),
sin ( zeta + th3(t) + 2 * th2(t) + gw2a2p * qw2(t) ),
temp )$

temp : subst ( lcsub(18),
cos ( zeta + th3(t) + 2 * th2(t) + gw2a2p * qw2(t) ),
temp )$

temp : subst ( lcsub(19),
sin ( zeta + th3(t) + th2(t) + gw2a2p * qw2(t) ),
temp )$

temp : subst ( lcsub(20),
cos ( zeta + th3(t) + th2(t) + gw2a2p * qw2(t) ),
temp )$

temp : subst ( lcsub(21),
sin ( zeta + th3(t) - th2(t) + gw2a2p * qw2(t) ),
temp )$

temp : subst ( lcsub(22),
cos ( zeta + th3(t) - th2(t) + gw2a2p * qw2(t) ),
temp )$

temp : subst ( lcsub(23),
sin ( zeta + th3(t) + gw2a2p * qw2(t) ), temp )$

temp : subst ( lcsub(24),
cos ( zeta + th3(t) + gw2a2p * qw2(t) ), temp )$

temp : subst ( lcsub(25),
sin ( 2 * zeta + 2 * th3(t) + th2(t) +
2 * gw2a2p * qw2(t) ), temp )$

temp : subst ( lcsub(26),
cos ( 2 * zeta + 2 * th3(t) + th2(t) +
2 * gw2a2p * qw2(t) ), temp )$

temp : subst ( lcsub(27),
sin ( 2 * zeta + 2 * th3(t) + 2 * gw2a2p * qw2(t) ),
temp )$

temp : subst ( lcsub(28),
cos ( 2 * zeta + 2 * th3(t) + 2 * gw2a2p * qw2(t) ),
temp )$

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

Mission accomplished, see you...
*/

/* Program name : Subreduceh3.com

This is the command procedure routine that helps to reduce and simplify the current link h3's equation in equation 'temp'.

*************************************************************/
temp : subst ( diff ( th1(t), t ), th1d, temp )$

$$
\text{temp : subst ( diff ( th1(t), t, 2 ), th1dd, temp )$

$$
\text{temp : subst ( diff ( th2(t), t ), th2d, temp )$

$$
\text{temp : subst ( diff ( th2(t), t, 2 ), th2dd, temp )$

$$
\text{temp : subst ( diff ( th3(t), t ), th3d, temp )$

$$
\text{temp : subst ( diff ( th3(t), t, 2 ), th3dd, temp )$

$$
\text{temp : subst ( diff ( qv2(t), t ), qv2d, temp )$

$$
\text{temp : subst ( diff ( qv2(t), t, 2 ), qv2dd, temp )$

$$
\text{temp : subst ( diff ( qv2(t), t ), qv2d, temp )$

$$
\text{temp : subst ( diff ( qw2(t), t ), qw2d, temp )$

$$
\text{temp : subst ( diff ( qw2(t), t, 2 ), qw2dd, temp )$

$$
\text{temp : subst ( diff ( qw2(t), t ), qw2d, temp )$

$$
\text{temp : subst ( diff ( qw2(t), t, 2 ), qw2dd, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t ), qv3d, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t, 2 ), qv3dd, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t ), qv3d, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t, 2 ), qv3dd, temp )$

$$
\text{temp : subst ( diff ( qw3(t), t ), qw3d, temp )$

$$
\text{temp : subst ( diff ( qw3(t), t, 2 ), qw3dd, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t ), qw3d, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t, 2 ), qw3dd, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t ), qv3d, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t, 2 ), qv3dd, temp )$

$$
\text{temp : subst ( diff ( qw3(t), t ), qw3d, temp )$

$$
\text{temp : subst ( diff ( qw3(t), t, 2 ), qw3dd, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t ), qw3d, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t, 2 ), qw3dd, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t ), qv3d, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t, 2 ), qv3dd, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t ), qv3d, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t, 2 ), qv3dd, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t ), qv3d, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t, 2 ), qv3dd, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t ), qv3d, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t, 2 ), qv3dd, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t ), qv3d, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t, 2 ), qv3dd, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t ), qv3d, temp )$

$$
\text{temp : subst ( diff ( qv3(t), t, 2 ), qv3dd, temp )$
temp : subst ( ah3ssq, ah3s^2, temp )$

temp : subst ( ah3scu, ah3s^3, temp )$

temp : subst ( ah3spstartsq, ah3spstart^2, temp )$

temp : subst ( ah3spstartcu, ah3spstart^3, temp )$

temp : subst ( ah3sendsq, ah3send^2, temp )$

temp : subst ( ah3sendcu, ah3send^3, temp )$

temp : subst ( ah3sq, ah3^2, temp )$

temp : subst ( costhh3, cos ( thh3 ), temp )$

temp : subst ( sinthh3, sin ( eta + th2(t) ), temp )$

temp : subst ( coseta, cos ( eta ), temp )$

temp : subst ( sineta, sin ( eta ), temp )$

temp : subst ( qvh3dd, diff ( qvh3(t), t, 2 ), temp )$

temp : subst ( qvh3d, diff ( qvh3(t), t ), temp )$

temp : subst ( qvh3ssq, qvh3(t)A 2, temp )$

temp : subst ( qvh3cu, qvh3(t)A 3, temp )$

temp : subst ( qvhh3, qvhh3(uh3), temp )$

temp : subst ( gvh3ppsq(i), ( diff ( gvh3(uh3), uh3, 2 ) )^2, temp )$

temp : subst ( gvh3(i), gvh3(uh3), temp )$

temp : subst ( gvh3ssq(i), gvh3(i)A 2, temp )$

temp : subst ( gvh3cu, gvh3(i)A 3, temp )$

temp : subst ( gvh3ah3ppsq, ( diff ( gvh3(ah3), ah3, 2 ) )^2, temp )$

temp : subst ( gvh3ah3, gvh3(ah3), temp )$

temp : subst ( gwh3ppsq(i), ( diff ( gwh3(uh3), uh3, 2 ) )^2, temp )$

temp : subst ( gwh3(i), gwh3(uh3), temp )$

temp : subst ( gwh3ssq(i), gwh3(i)A 2, temp )$

temp : subst ( gwh3ah3ppsq, ( diff ( gwh3(ah3), ah3, 2 ) )^2, temp )$

temp : subst ( gwh3ah3, gwh3(ah3), temp )$

temp : subst ( gwh3ah3psq, gwh3ah3pA 2, temp )$

temp : subst ( uh3(i), uh3, temp )$

temp : subst ( lh3sub(1),
( costhh3 * qvh3 * gvh3(i) + sinthh3 * uh3(i) ), temp )$

temp : subst ( -lh3sub(1),
( -costhh3 * qvh3 * gvh3(i) - sinthh3 * uh3(i) ), temp )$

temp : subst ( detaqv2 * lh3sub(1),
( detaqv2 * uh3(i) * sinthh3 +
- costhh3 * detaqv2 * gvh3(i) * qvh3 ), temp )$

temp : subst ( detath3 * lh3sub(1),
( detath3 * uh3(i) * sinthh3 +
- costhh3 * detath3 * gvh3(i) * qvh3 ), temp )$

temp : subst ( -dthh3qv2 * lh3sub(1),
( -dthh3qv2 * sinthh3 * uh3(i) -
- costhh3 * dthh3qv2 * gvh3(i) * qvh3 ), temp )$

temp : subst ( -dthh3th3h3 * lh3sub(1),
( -dthh3th3h3 * sinthh3 * uh3(i) -
- costhh3 * dthh3th3h3 * gvh3(i) * qvh3 ), temp )$

temp : subst ( lh3sub(2),
( costhh3 * uh3(i) - sinthh3 * qvh3 * gvh3(i) ), temp )$

temp : subst ( -lh3sub(2),
( sinthh3 * qvh3 * gvh3(i) - costhh3 * uh3(i) ), temp )$

temp : subst ( detaqv2 * lh3sub(2),
( detaqv2 * uh3(i) -
- detath3 * gvh3(i) * qvh3 * sinthh3 ), temp )$

temp : subst ( detath3 * lh3sub(2),
( detath3 * uh3(i) -
- detath3 * gvh3(i) * qvh3 * sinthh3 ), temp )$

temp : subst ( dthh3qv2 * lh3sub(2),
( dthh3qv2 * gvh3(i) * qvh3 * sinthh3 ), temp )$

temp : subst ( dthh3th3h3 * lh3sub(2),
( dthh3th3h3 * uh3(i) -
- costhh3 * gvh3(i) * qvh3 * sinthh3 ), temp )$

temp : subst ( lh3sub(3),
( coseta * qvh3 * gvh3(i) + sineta * uh3(i) ), temp )$
temp : subst ( -lh3sub(3),
( -coseta * qvh3 * gvh3(i) - sineta * uh3(i) ), temp )$

temp : subst ( -detaqv2 * lh3sub(3),
( -detaqv2 * uh3(i) - sineta * coseta * detaqv2 * gvh3(i) * gvh3 ), temp )$

temp : subst ( -detath3 * lh3sub(3),
( -detath3 * uh3(i) * sineta - coseta * detath3 * gvh3(i) * gvh3 ), temp )$

temp : subst ( dthh3qv2 * lh3sub(3),
( dthh3qv2 * sineta * uh3(i) + coseta * dthh3qv2 * gvh3(i) * gvh3 ), temp )$

temp : subst ( dthh3th3 * lh3sub(3),
( dthh3th3 * sineta * uh3(i) + coseta * dthh3th3 * gvh3(i) * gvh3 ), temp )$

temp : subst ( lh3sub(4),
( coseta * uh3(i) - sineta * qvh3 * gvh3(i) ), temp )$

temp : subst ( -lh3sub(4),
( sineta * qvh3 * gvh3(i) - coseta * uh3(i) ), temp )$

temp : subst ( detaqv2 * lh3sub(4),
( coseta * detaqv2 * uh3(i) - detaqv2 * gvh3(i) * qvh3 * sineta ), temp )$

temp : subst ( detath3 * lh3sub(4),
( coseta * detath3 * uh3(i) - detath3 * gvh3(i) * qvh3 * sineta ), temp )$

temp : subst ( dthh3qv2 * lh3sub(4),
( coseta * dthh3qv2 * uh3(i) - dthh3qv2 * gvh3(i) * qvh3 * sineta ), temp )$

temp : subst ( dthh3th3 * lh3sub(4),
( coseta * dthh3th3 * uh3(i) - dthh3th3 * gvh3(i) * qvh3 * sineta ), temp )$

temp : subst ( lh3sub(5),
( -gvh3(i) * qvh3d * sineta - detat * uh3(i) * sineta - coseta * detat * gvh3(i) * qvh3 ), temp )$

temp : subst ( lh3sub(6),
( -detat * gvh3(i) * qvh3 * sineta + coseta * gvh3(i) * qvh3d + coseta * detat * uh3(i) ), temp )$

temp : subst ( lh3sub(7),
( gvh3(i) * gvh3d * sinhthh3 + dthh3t * uh3(i) * sinhthh3 + costhh3 * dthh3t * gvh3(i) * qvh3 ), temp )$

temp : subst ( lh3sub(8),
( gw2a2p * qw2(t) * sin ( th2(t) ) + th1d * th2d * gvh3(i) / gwh3ah3p ), temp )$

temp : subst ( lh3sub(9),
( gw2a2p * qw2(t) * cos ( th2(t) ) + th1dd * gwh3(i) / gwh3ah3p ), temp )$

temp : subst ( lh3sub(10),
( gw2a2p * qw2d * cos ( th2(t) ) + th1d * gwh3(i) / gwh3ah3p ), temp )$

temp : subst ( lh3sub(11),
( gw2a2p * qw2(t) * cos ( th2(t) ) + th1d * th2d * gwh3(i) / gwh3ah3p ), temp )$

temp : subst ( lh3sub(12),
( gw2a2p * qw2(t) * sin ( th2(t) ) + th1dd * gwh3(i) / gwh3ah3p ), temp )$

temp : subst ( lh3sub(13),
( gw2a2p * qw2d * sin ( th2(t) ) + th1d * gwh3(i) / gwh3ah3p ), temp )$

temp : subst ( lh3sub(14),
( gw2a2p * qw2(t) * sin ( th2(t) ) + th1d * gwh3(i) / gwh3ah3p ), temp )$

temp : subst ( lh3sub(15),
( gw2a2p * qw2(t) * cos ( th2(t) ) + th1d * gwh3(i) / gwh3ah3p ), temp )$

temp : subst ( lh3sub(16),
( gw2a2p * qw2d * gwh3(i) / gwh3ah3p ), temp )$

temp : subst ( lh3sub(17),
( lh3sub(16) - lh3sub(2) * th1d ), temp )$

temp : subst ( lh3sub(18),
temp: subst ( lh3sub(19),
(dah3qv2 * sineta + lh3sub(4) * dthh3qv2 ), temp )$

temp: subst ( lh3sub(20),
(gvh3(i) * sineta - lh3sub(3) * dthh3qv3 ), temp )$

temp: subst ( lh3sub(21),
(coseta * gvh3(i) + lh3sub(4) * dthh3qv3 ), temp )$

temp: subst ( lh3sub(22),
(-gvh3(i) * qvh3d * sineta - lh3sub(3) * dthh3t +
coseta * dah3t + lh3sub(15) ), temp )$

temp: subst ( lh3sub(23),
(dah3t * sineta + coseta * gvh3(i) * qvh3d +
lh3sub(4) * dthh3t - lh3sub(14) ), temp )$

temp: subst ( lh3sub(24),
(-dah3qv2 * gvh3(i) * qvh3d * sineta + dah3t +
sineta + coseta * gvh3(i) * qvh3d + lh3sub(4) * dthh3t +
lh3sub(5) * dthh3t + coseta * dah3t +
{} \text{detat} - lh3sub(13) - lh3sub(12) - lh3sub(11) ), temp )$

temp: subst ( lh3sub(25),
(-dah3t * gvh3(i) * qvh3d + dah3t * sineta -
detat * gvh3(i) * qvh3d - lh3sub(3) * dthh3t +
lh3sub(6) * dthh3t +
coseta * dah3t + lh3sub(10) + lh3sub(9) -
lh3sub(8) ), temp )$

...
coseta * dah3t + lh3sub(15), temp )$

```
** Program name : Subsimp1.com  
This subroutine helps to replace the time-dependent generalized coordinate with their variable names. The replacement is for the current link 1's equation stored in 'temp'.
**************************************************************************
Mission accomplished, see you...
*/
```

/* Program name : Subsimp2.com

This subroutine helps to replace the time-dependent generalize
cordinate with their variable names. The replacement is for the
current link 2's equation stored in 'temp'.

***************************************************************************
*/
temp : subst ( th2d, diff ( th2(t), t ), temp )$
temp : subst ( th2dd, diff ( th2(t), t, 2 ), temp )$
temp : subst ( qv2d, diff ( qv2(t), t ), temp )$
temp : subst ( qv2dd, diff ( qv2(t), t, 2 ), temp )$
temp : subst ( qw2d, diff ( qw2(t), t ), temp )$
temp : subst ( qw2dd, diff ( qw2(t), t, 2 ), temp )$

Mission accomplished, see you...


/* Program name : Subsimp3.com

This subroutine helps to replace the time-dependent generalize
cordinate with their variable names. The replacement is for the
current link 3's equation stored in 'temp'.

***************************************************************************
*/
temp : subst ( th3d, diff ( th3(t), t ), temp )$
temp : subst ( th3dd, diff ( th3(t), t, 2 ), temp )$
temp : subst ( qv3d, diff ( qv3(t), t ), temp )$
temp : subst ( qv3dd, diff ( qv3(t), t, 2 ), temp )$
temp : subst ( qw3d, diff ( qw3(t), t ), temp )$
temp : subst ( qw3dd, diff ( qw3(t), t, 2 ), temp )$

Mission accomplished, see you...


/* Program name : Subsimp3h.com

This subroutine helps to replace the time-dependent generalize
cordinate with their variable names. The replacement is for the
current link h3's equation stored in 'temp'.

***************************************************************************
*/
temp : subst ( qvh3d, diff ( qvh3(t), t ), temp )$
temp : subst ( qvh3dd, diff ( qvh3(t), t, 2 ), temp )$

Mission accomplished, see you...


/* Program name : Sumup.com

***************************************************************************
This subroutine helps to setup the elements of the solution matrix and sum up the torque generated from each link.

Sum up the torques.

```
torque_1 : torque (1) = torque1l1 + torque1l2 + torque1l3 + torque1lc + torque1lh3

torque_2 : torque (2) = torque2l1 + torque2l2 + torque2l3 + torque2lc + torque2lh3

torque_3 : torque (3) = torque3l1 + torque3l2 + torque3l3 + torque3lc + torque3lh3
```

Sum up the coefficients of each link.

```
coeff_a1 : a (1, 1) = coeffa1l1 + coeffa1l2 + coeffa1l3 + coeffa1lc + coeffa1lh3
coeff_b1 : a (1, 2) = coeffb1l1 + coeffb1l2 + coeffb1l3 + coeffb1lc + coeffb1lh3
coeff_c1 : a (1, 3) = coeffc1l1 + coeffc1l2 + coeffc1l3 + coeffc1lc + coeffc1lh3
coeff_d1 : a (1, 4) = coeffd1l1 + coeffd1l2 + coeffd1l3 + coeffd1lc + coeffd1lh3
coeff_e1 : a (1, 5) = coeffe1l1 + coeffe1l2 + coeffe1l3 + coeffe1lc + coeffe1lh3

coeff_a2 : a (2, 1) = coeffa2l1 + coeffa2l2 + coeffa2l3 + coeffa2lc + coeffa2lh3
coeff_b2 : a (2, 2) = coeffb2l1 + coeffb2l2 + coeffb2l3 + coeffb2lc + coeffb2lh3
coeff_c2 : a (2, 3) = coeffc2l1 + coeffc2l2 + coeffc2l3 + coeffc2lc + coeffc2lh3
coeff_d2 : a (2, 4) = coeffd2l1 + coeffd2l2 + coeffd2l3 + coeffd2lc + coeffd2lh3
coeff_e2 : a (2, 5) = coeffe2l1 + coeffe2l2 + coeffe2l3 + coeffe2lc + coeffe2lh3

coeff_a3 : a (3, 1) = coeffa3l1 + coeffa3l2 + coeffa3l3 + coeffa3lc + coeffa3lh3
coeff_b3 : a (3, 2) = coeffb3l1 + coeffb3l2 + coeffb3l3 + coeffb3lc + coeffb3lh3
coeff_c3 : a (3, 3) = coeffc3l1 + coeffc3l2 + coeffc3l3 + coeffc3lc + coeffc3lh3
coeff_d3 : a (3, 4) = coeffd3l1 + coeffd3l2 + coeffd3l3 + coeffd3lc + coeffd3lh3
coeff_e3 : a (3, 5) = coeffe3l1 + coeffe3l2 + coeffe3l3 + coeffe3lc + coeffe3lh3

coeff_a4 : a (4, 1) = coeffa4l1 + coeffa4l2 + coeffa4l3 + coeffa4lc + coeffa4lh3
coeff_b4 : a (4, 2) = coeffb4l1 + coeffb4l2 + coeffb4l3 + coeffb4lc + coeffb4lh3
coeff_c4 : a (4, 3) = coeffc4l1 + coeffc4l2 + coeffc4l3 + coeffc4lc + coeffc4lh3
coeff_d4 : a (4, 4) = coeffd4l1 + coeffd4l2 + coeffd4l3 + coeffd4lc + coeffd4lh3
coeff_e4 : a (4, 5) = coeffe4l1 + coeffe4l2 + coeffe4l3 + coeffe4lc + coeffe4lh3

coeff_a5 : a (5, 1) = coeffa5l1 + coeffa5l2 + coeffa5l3 + coeffa5lc + coeffa5lh3
coeff_b5 : a (5, 2) = coeffb5l1 + coeffb5l2 + coeffb5l3 + coeffb5lc + coeffb5lh3
coeff_c5 : a (5, 3) = coeffc5l1 + coeffc5l2 + coeffc5l3 + coeffc5lc + coeffc5lh3
coeff_d5 : a (5, 4) = coeffd5l1 + coeffd5l2 + coeffd5l3 + coeffd5lc + coeffd5lh3
```
coefficient_l + coefficient_r
coefficient_e5 : a ( 5, 5 ) = coefficient_e5_l1 + coefficient_e5_l2 + coefficient_e5_l3 +
coefficient_e5_l4 + coefficient_e5_l5
coefficient_f1 : b ( 1 ) = -( coefficient_f1_l1 + coefficient_f1_l2 + coefficient_f1_l3 +
coefficient_f1_l4 + coefficient_f1_l5 )
coefficient_f2 : b ( 2 ) = -( coefficient_f2_l1 + coefficient_f2_l2 + coefficient_f2_l3 +
coefficient_f2_l4 + coefficient_f2_l5 )
coefficient_f3 : b ( 3 ) = -( coefficient_f3_l1 + coefficient_f3_l2 + coefficient_f3_l3 +
coefficient_f3_l4 + coefficient_f3_l5 )
coefficient_f4 : b ( 4 ) = -( coefficient_f4_l1 + coefficient_f4_l2 + coefficient_f4_l3 +
coefficient_f4_l4 + coefficient_f4_l5 )
coefficient_f5 : b ( 5 ) = -( coefficient_f5_l1 + coefficient_f5_l2 + coefficient_f5_l3 +
coefficient_f5_l4 + coefficient_f5_l5 )

/* **************************************************
   Translate the equations into Fortran codes.
*/
writefile ( "fortran.lst" );
fortran ( torque_1 );
fortran ( torque_2 );
fortran ( torque_3 );
fortran ( coefficient_a1 );
fortran ( coefficient_b1 );
fortran ( coefficient_c1 );
fortran ( coefficient_d1 );
fortran ( coefficient_e1 );
fortran ( coefficient_a2 );
fortran ( coefficient_b2 );
fortran ( coefficient_c2 );
fortran ( coefficient_d2 );
fortran ( coefficient_e2 );
fortran ( coefficient_a3 );
fortran ( coefficient_b3 );
fortran ( coefficient_c3 );
fortran ( coefficient_d3 );
fortran ( coefficient_e3 );
fortran ( coefficient_a4 );
fortran ( coefficient_b4 );
fortran ( coefficient_c4 );
fortran ( coefficient_d4 );
fortran ( coefficient_e4 );
fortran ( coefficient_a5 );
fortran ( coefficient_b5 );
fortran ( coefficient_c5 );
fortran ( coefficient_d5 );
fortran ( coefficient_e5 );
fortran ( coefficient_f1 );
fortran ( coefficient_f2 );
fortran ( coefficient_f3 );
fortran ( coefficient_f4 );
fortran ( coefficient_f5 );
closefile ();

/* **************************************************
   Done. Bye...
*/
Program name: V8rsetup1.com
This routine helps to set up all the variables and parameters for link 1.

Setup the general variables for link 1.

\[
\text{thld} : \text{diff}(\text{th1}(t), t) $
\]

\[
\omega_1 \_{\text{at}_1} : \text{matrix}([0],[0],[\text{thld}])$
\]

\[
\omega_1 = \text{transpose}(\text{transform} \_1 \_\to \_2 \cdot \omega_1 \_{\text{at}_1})$
\]

\[
\text{velocity}_\text{at} \_0(i,j,k) := [0,0,0]$
\]

Set up the displacement vector and the velocity vector for link 1.

\[
\text{vector}_1 \_{\text{at}_1} : \text{matrix}([0],[0],[-w1])$
\]

\[
\text{vector}_1 = \text{transpose}(\text{transform} \_1 \_\to \_2 \cdot \omega_1) $
\]

\[
\text{gdotvector}_1 = \text{trigreduce}(\text{gravity} \_2 \cdot \text{vector}_1)$
\]

\[
\omega_1 \times \text{vector}_1 = \text{cross_product}(i,j,k)$
\]

\[
\omega_1 = \text{subst}\left(\text{part}(\omega_1,1,1), \unita, \omega_1 \times \text{vector}_1\right) $
\]

\[
\omega_1 = \text{subst}\left(\text{part}(\omega_1,1,2), \unitb, \omega_1 \times \text{vector}_1\right) $
\]

\[
\omega_1 = \text{subst}\left(\text{part}(\omega_1,1,3), \unitc, \omega_1 \times \text{vector}_1\right) $
\]

\[
\omega_1 = \text{subst}\left(\text{part}(\text{vector}_1,1,T), \unitx, \omega_1 \times \text{vector}_1\right) $
\]

\[
\omega_1 = \text{subst}\left(\text{part}(\text{vector}_1,1,2), \unity, \omega_1 \times \text{vector}_1\right) $
\]

\[
\omega_1 = \text{subst}\left(\text{part}(\text{vector}_1,1,3), \unitz, \omega_1 \times \text{vector}_1\right) $
\]

\[
\text{vector}_1 = \text{velocity}_\text{at} \_0(i,j,k) + \omega_1 \times \text{vector}_1$
\]

That's it folks...

Program name: Varsetup2.com
This routine helps to set up all the variables and parameters for link 2.

Set up the general variables for link 2.

\[
\text{th2d} : \text{diff}(\text{th2}(t), t) $
\]

\[
\text{v2} : \text{gv2}(u2) * \text{qv2}(t) $
\]

\[
\text{v2d} : \text{diff}(\text{v2}, t) $
\]

\[
\text{v2p} : \text{diff}(\text{v2}, u2) $
\]

\[
\text{v2pp} : \text{diff}(\text{v2p}, u2) $
\]

\[
\text{v2} \_\text{at} \_a2 : \text{gv2a2} * \text{qv2}(t) $
\]

\[
\text{v2} \_\text{at} \_a2 \_d : \text{diff}(\text{v2} \_\text{at} \_a2, t) $
\]
Set up the displacement vector and the velocity vector for the rigid portion for link 2.

\[
\begin{align*}
v_{2\text{a}} & : g(v_{2\text{a}}) * q(v_{2\text{a}}(t)) \\
v_{2\text{a}p} & : \text{diff} (v_{2\text{a}}, t) \\
v_{2\text{a}pd} & : \text{diff} (v_{2\text{a}p}, t) \\
v_{2} & : g(u_{2}) * q(u_{2}(t)) \\
v_{2d} & : \text{diff} (v_{2}, t) \\
v_{2p} & : \text{diff} (v_{2}, u_{2}) \\
v_{2pp} & : \text{diff} (v_{2p}, u_{2}) \\
w_{2} & : g(w_{2}) * q(w_{2}(t)) \\
w_{2d} & : \text{diff} (w_{2}, t) \\
w_{2p} & : \text{diff} (w_{2}, u_{2}) \\
w_{2pp} & : \text{diff} (w_{2p}, u_{2}) \\
w_{2a} & : g(w_{2a}) * q(w_{2a}(t)) \\
w_{2ad} & : \text{diff} (w_{2a}, t) \\
w_{2ap} & : g(w_{2ap}) * q(w_{2ap}(t)) \\
w_{2apd} & : \text{diff} (w_{2ap}, t) \\
q_{v2d} & : \text{diff} (q(v_{2}(t)), t) \\
q_{w2d} & : \text{diff} (q(w_{2}(t)), t) \\
\omega_{2} & : \omega_{1} + \omega_{2} \\
\omega_{2a} & : \omega_{2} \\
\omega_{2d} & : \text{diff} (\omega_{2}, t) \\
\omega_{2p} & : \text{diff} (\omega_{2}, u_{2}) \\
w_{2a} & : g(u_{2}) * q(u_{2}(t)) \\
w_{2ad} & : \text{diff} (w_{2a}, t) \\
w_{2ap} & : g(w_{2ap}) * q(w_{2ap}(t)) \\
w_{2apd} & : \text{diff} (w_{2ap}, t) \\
\text{velocity}_at_1 & : \text{subst} (d_1, w_1, \text{vector}_1d) \\
\text{velocity}_2 & : \text{subst} (d_2, w_2, \text{vector}_2) \\
\text{velocity}_2d & : \text{diff} (\text{velocity}_2, t) \\
\text{velocity}_2d & : \text{diff} (\text{velocity}_2, t) \\
\omega_{2} & : \omega_{2} \\
\omega_{2a} & : \omega_{2} \\
\omega_{2d} & : \text{diff} (\omega_{2}, t) \\
\omega_{2p} & : \text{diff} (\omega_{2}, u_{2}) \\
\omega_{2ad} & : \text{diff} (\omega_{2a}, t) \\
\omega_{2ap} & : g(w_{2ap}) * q(w_{2ap}(t)) \\
\omega_{2apd} & : \text{diff} (w_{2ap}, t) \\
\omega_{2ad} & : \text{diff} (w_{2ad}, t) \\
\omega_{2ap} & : g(w_{2ap}) * q(w_{2ap}(t)) \\
\omega_{2apd} & : \text{diff} (w_{2ap}, t) \\
\text{omega}_2 & : \text{matrix}([0, 0, \text{th}_2d]) \\
\text{omega}_2 & : \text{omega}_2 \\
\text{velocity}_at_1 & : \text{subst} (d_1, w_1, \text{vector}_1d) \\
/* *******************************************************

Set up the displacement vector and the velocity vector for the rigid portion for link 2.

*/

\[
\begin{align*}
\text{vector}_2r & : \text{matrix}([u_2, 0, 0]) \\
\text{gdotvector}_2r & : \text{gravity}_2 . \text{vector}_2r \\
\omega_{2} & : \text{matrix}([0, 0, \text{th}_2]) \\
\omega_{2} & : \text{omega}_2 \\
\omega_{2} & : \text{omega}_2 \\
\text{velocity}_at_1 & : \text{subst} (d_1, w_1, \text{vector}_1d) \\
/* *******************************************************

Set up the displacement vector and the velocity vector for the elastic portion for link 2.

*/

\[
\begin{align*}
\text{vector}_2e & : \text{matrix}([u_2, v_2, w_2]) \\
\text{gdotvector}_2e & : \text{factor} (\text{velocity}_2, \text{vector}_2e) \\
\omega_{2} & : \text{matrix}([0, 0, \text{th}_2]) \\
\omega_{2} & : \text{omega}_2 \\
\omega_{2} & : \text{omega}_2 \\
\text{velocity}_at_1 & : \text{subst} (d_1, w_1, \text{vector}_1d) \\
/* *******************************************************
omega_2_x_vector_2e : subst ( part(vector_2e,1,3), unitz, omega_2_x_vector_2e )

omega_2_x_vector_2e_at_a2 : cross_product(i,j,k)

omega_2_x_vector_2e_at_a2 : subst ( part(omega_2,1,1), unita, omega_2_x_vector_2e_at_a2 )

omega_2_x_vector_2e_at_a2 : subst ( part(omega_2,1,2), unitb, omega_2_x_vector_2e_at_a2 )

omega_2_x_vector_2e_at_a2 : subst ( part(omega_2,1,3), unitc, omega_2_x_vector_2e_at_a2 )

omega_2_x_vector_2e_at_a2 : subst ( part(vector_2e_at_a2,1,1), unitx, omega_2_x_vector_2e_at_a2 )

omega_2_x_vector_2e_at_a2 : subst ( part(vector_2e_at_a2,1,2), unity, omega_2_x_vector_2e_at_a2 )

omega_2_x_vector_2e_at_a2 : subst ( part(vector_2e_at_a2,1,3), unitz, omega_2_x_vector_2e_at_a2 )

projection_v2(i,j,k) := [ 0, 1, 0 ]
projection_w2(i,j,k) := [ 0, 0, 1 ]

vector_2ed : velocity_at_1 + omega_2_x_vector_2e + v2d * projection_v2(i,j,k) + w2d * projection_w2(i,j,k)

vector_2edsq : vector_2ed . vector_2ed

vector_2e_at_a2d : velocity_at_1 + omega_2_x_vector_2e_at_a2 + v2_at_a2d * projection_v2(i,j,k) + w2_at_a2d * projection_w2(i,j,k)

vector_2e_at_a2dsq : vector_2e_at_a2d . vector_2e_at_a2d

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

That's it folks...

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

/* Program name : Varsetup3.com

This routine helps to set up all the variables and parameters for
link 3.

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

Set up the general variables for link 3.

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

th3d : diff ( th3(t), t )
psid : th3(t) + v2_at_a2p

v3 : gv3(u3) * qv3(t)
v3d : ddiff ( v3, t )
v3p : ddiff ( v3, u3 )
v3pp : ddiff ( v3p, u3 )
v3_at_a3 : gv3a3 * qv3(t)
v3_at_a3d : ddiff ( v3_at_a3, t )
w3 : gw3(u3) * qw3(t)
w3d : ddiff ( w3, t )
w3p : ddiff ( w3, u3 )
w3pp : ddiff ( w3p, u3 )
w3_at_a3 : gw3a3 * qw3(t)
w3_at_a3d : ddiff ( w3_at_a3, t )
qv3d : ddiff ( qv3(t), t )
qw3d : ddiff ( qw3(t), t )

transform_3_to_2 : matrix ( [ cos(psi), sin(psi), 0 ], 
\[
\begin{bmatrix}
\sin(\psi), \cos(\psi), 0 \\
0, 0, 1
\end{bmatrix}
\]

\[
\omega_{2 \text{ at } a_2} : \omega_2 + \text{matrix} \left( \begin{bmatrix}
0, w_{a2 \text{ at } a2d}, v_{a2 \text{ at } a2d}
\end{bmatrix} \right)
\]

\[
\omega_{3 \text{ at } 3} : \text{matrix} \left( \begin{bmatrix}
0, 0, t_{3 \text{d}}
\end{bmatrix} \right)
\]

\[
\omega_3 = \omega_{2 \text{ at } a_2} + \text{transpose} \left( \text{transform}_{3 \text{ to } 2} \cdot \omega_{3 \text{ at } 3} \right)
\]

\[
\text{velocity}_{a2} : \text{vector}_{2e \text{ at } a2d}
\]

/*
Set up the displacement vector and the velocity vector for link 3.
*/

\[
\text{vector}_{3 \text{ at } 3} : \text{matrix} \left( \begin{bmatrix}
u_3, v_3, w_3
\end{bmatrix} \right)
\]

\[
\text{vector}_3 = \text{transpose} \left( \text{transform}_{3 \text{ to } 2} \cdot \text{vector}_{3 \text{ at } 3} \right)
\]

\[
\text{vector}_{3 \text{ at } a_3 \text{ at } 3} : \text{matrix} \left( \begin{bmatrix}
a_3, v_{3 \text{ at } a_3}, w_{3 \text{ at } a_3}
\end{bmatrix} \right)
\]

\[
\text{vector}_{3 \text{ at } a_3} = \text{transpose} \left( \text{transform}_{3 \text{ to } 2} \cdot \text{vector}_{3 \text{ at } a_3 \text{ at } 3} \right)
\]

\[
g_{dot}\text{vector}_3 : \text{gravity}_{a2} \cdot \left( \text{vector}_{2e \text{ at } a2d} + \text{vector}_3 \right)
\]

\[
g_{dot}\text{vector}_3 = \text{factor} \left( \text{trigreduce} \left( \text{expand} \left( g_{dot}\text{vector}_3 \right) \right) \right)
\]

\[
g_{dot}\text{vector}_{a3} : \text{gravity}_{a2} \cdot \left( \text{vector}_{2e \text{ at } a2d} + \text{vector}_{3 \text{ at } a3} \right)
\]

\[
g_{dot}\text{vector}_{a3} = \text{factor} \left( \text{trigreduce} \left( \text{expand} \left( g_{dot}\text{vector}_{a3} \right) \right) \right)
\]

\[
\omega_3 \times \text{vector}_3 : \text{cross_product} (i, j, k)
\]

\[
\omega_3 \times \text{vector}_3 = \text{sub} \left( \text{part} (\omega_3, 1,1), \text{unita}, \omega_3 \times \text{vector}_3 \right)
\]

\[
\omega_3 \times \text{vector}_3 = \text{sub} \left( \text{part} (\omega_3, 1,2), \text{unitb}, \omega_3 \times \text{vector}_3 \right)
\]

\[
\omega_3 \times \text{vector}_3 = \text{sub} \left( \text{part} (\omega_3, 1,3), \text{unitc}, \omega_3 \times \text{vector}_3 \right)
\]

\[
\omega_3 \times \text{vector}_3 = \text{sub} \left( \text{part} (\vector_3, 1,1), \text{unixa}, \omega_3 \times \text{vector}_3 \right)
\]

\[
\omega_3 \times \text{vector}_3 = \text{sub} \left( \text{part} (\vector_3, 1,2), \text{unib}, \omega_3 \times \text{vector}_3 \right)
\]

\[
\omega_3 \times \text{vector}_3 = \text{sub} \left( \text{part} (\vector_3, 1,3), \text{unicz}, \omega_3 \times \text{vector}_3 \right)
\]

\[
\text{part}_1 : \text{part} \left( \omega_3 \times \text{vector}_3, 1 \right)
\]

\[
\text{part}_2 : \text{part} \left( \omega_3 \times \text{vector}_3, 2 \right)
\]

\[
\text{part}_3 : \text{part} \left( \omega_3 \times \text{vector}_3, 3 \right)
\]

\[
\text{part}_3 = \text{factor} \left( \text{trigreduce} \left( \text{part}_3 \right) \right)
\]

\[
\omega_3 \times \text{vector}_3 = \text{sub} \left( \text{part}_1, \text{part}_2, \text{part}_3 \right)
\]

\[
\omega_3 \times \text{vector}_3 \text{ at } a_3 : \text{cross_product} (i, j, k)
\]

\[
\omega_3 \times \text{vector}_3 \text{ at } a_3 = \text{sub} \left( \text{part} (\omega_3, 1,1), \text{unita}, \omega_3 \times \text{vector}_3 \text{ at } a_3 \right)
\]

\[
\omega_3 \times \text{vector}_3 \text{ at } a_3 = \text{sub} \left( \text{part} (\omega_3, 1,2), \text{unitb}, \omega_3 \times \text{vector}_3 \text{ at } a_3 \right)
\]

\[
\omega_3 \times \text{vector}_3 \text{ at } a_3 = \text{sub} \left( \text{part} (\omega_3, 1,3), \text{unitc}, \omega_3 \times \text{vector}_3 \text{ at } a_3 \right)
\]

\[
\omega_3 \times \text{vector}_3 \text{ at } a_3 = \text{sub} \left( \text{part} (\vector_3, 1,1), \text{unixa}, \omega_3 \times \text{vector}_3 \text{ at } a_3 \right)
\]

\[
\omega_3 \times \text{vector}_3 \text{ at } a_3 = \text{sub} \left( \text{part} (\vector_3, 1,2), \text{unib}, \omega_3 \times \text{vector}_3 \text{ at } a_3 \right)
\]

\[
\omega_3 \times \text{vector}_3 \text{ at } a_3 = \text{sub} \left( \text{part} (\vector_3, 1,3), \text{unicz}, \omega_3 \times \text{vector}_3 \text{ at } a_3 \right)
\]

\[
\text{part}_1 = \text{part} \left( \omega_3 \times \text{vector}_3 \text{ at } a_3, 1 \right)
\]

\[
\text{part}_2 = \text{part} \left( \omega_3 \times \text{vector}_3 \text{ at } a_3, 2 \right)
\]

\[
\text{part}_3 = \text{part} \left( \omega_3 \times \text{vector}_3 \text{ at } a_3, 3 \right)
\]

\[
\text{part}_3 = \text{factor} \left( \text{trigreduce} \left( \text{part}_3 \right) \right)
\]

\[
\omega_3 \times \text{vector}_3 \text{ at } a_3 = \text{sub} \left( \text{part}_1, \text{part}_2, \text{part}_3 \right)
\]

\[
\text{temp} = \text{transform}_{3 \text{ to } 2} \cdot \text{matrix} \left( \begin{bmatrix}
0, 1, 0
\end{bmatrix} \right)
\]
temp : transpose ( temp )$
part1 : part ( temp, 1, 1 )$
part2 : part ( temp, 1, 2 )$
part3 : part ( temp, 1, 3 )$

projection_v3 : cross_product(i,j,k)$
projection_v3 : [ part1, part2, part3 ]$

projection_w3 : cross_product(i,j,k)$
projection_w3 : [ part1, part2, part3 ]$

vector_3d : velocity_at_2 + omega_3_x_vector_3 +
      v3d * projection_v3 +
      w3d * projection_w3$

vector_3dsq : vector_3d . vector_3d$

That's it folks...

Program name : Varsetupc.com
This routine helps to set up all the variables and parameters for
the connecting link.

Set up the general variables for the connecting link.

kappa : psi + zeta$
transform_c_to_2 : matrix ( [ cos(kappa), -sin(kappa), 0 ],
                      [ sin(kappa), cos(kappa), 0 ],
                      [ 0, 0, 1 ] )$

omega_c : omega_3$

Set up the displacement vector and the velocity vector for the
connecting link.

vector_c_at_c : matrix ( [ uc ], [ 0 ], [ 0 ] )$
vector_c : transpose ( transform_c_to_2 . vector_c )$
gdotvector_c : trigreduce ( gravity_at_2 . vector_c )$

omega_c_x_vector_c : cross_product(i,j,k)$
omega_c_x_vector_c : subst ( part(omega_c,1,1), unita,
                      omega_c_x_vector_c )$
omega_c_x_vector_c : subst ( part(omega_c,1,2), unitb,
                      omega_c_x_vector_c )$
omega_c_x_vector_c : subst ( part(omega_c,1,3), unitc,
                      omega_c_x_vector_c )$
omega_c_x_vector_c : subst ( part(vector_c,1,1), unitx,
                      omega_c_x_vector_c )$
omega_c_x_vector_c : subst ( part(vector_c,1,2), unity,
                                      omega_c_x_vector_c )$
omega_c_x_vector_c : subst ( part(vector_c,1,3), unitz,
                                      omega_c_x_vector_c )$

part1 : part ( omega_c_x_vector_c, 1 )$
part2 : part ( omega_c_x_vector_c, 2 )$
part3 : part ( omega_c_x_vector_c, 3 )$
part3 : trigreduce (~part3 )$

omega_c_x_vector_c : [ part1, part2, part3 ]$

vector_cd : velocity_at_2 + omega_c_x_vector_c$
vector_cdsq : vector_cd . vector_cd$

That's it folks...

/* Program name : Varsetuph3.com

This routine helps to set up all the variables and parameters for
the hydraulic cylinder 3.

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

Set up the general variables for hydraulic cylinder 3.

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

depends ( ah3pstart, [ qv2(t), th3(t) ] )$
depends ( ah3pend, [ qv2(t), th3(t) ] )$
depends ( ah3, [ qv2(t), th3(t) ] )$
depends ( eta, [ qv2(t), th3(t), qvh3(t) ] )$
depends ( thh3, [ th2(t), qv2(t), th3(t), qvh3(t) ] )$

vh3 : gvh3(uh3) * qvh3(t)$
vh3d : diff ( vh3, t )$
vh3p : diff ( vh3, uh3 )$
vh3pp : diff ( vh3, uh3, 2 )$

wh3 : gwh3(uh3) * ( w2_at_a2p / gwh3ah3p )$
wh3d : diff ( wh3, t )$
wh3p : diff ( wh3, uh3 )$
wh3pp : diff ( wh3, uh3, 2 )$

qvh3d : diff ( qvh3(t), t )$

transform_h3_to_2 : matrix ( [ cos(eta), -sin(eta), 0 ],
                            [ sin(eta), cos(eta), 0 ],
                            [ 0, 0, 1 ] )$

omega_h3_at_h3 : matrix ( [ 0, 1, 0 ],
                         [ 0, 1, 0 ],
                         [ thh3d ] )$
omega_h3 : omega_1 +
           transpose ( transform_h3_to_2 . omega_h3_at_h3 )$

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

Set up the displacement vector and the velocity vector for the
rigid portion of hydraulic cylinder 2.

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

vector_h3r_at_h3 : matrix ( [ uh3 ], [ 0 ], [ 0 ] )$
vector_h3r : transpose ( transform_h3_to_2 . vector_h3r_at_h3 )$
gdotvector_h3r : trigreduce ( gravity_at_2 . vector_h3r )$

omega_h3_x_vector_h3r : cross_product(i,j,k)$
omega_h3_x_vector_h3r : subst( part(omega_h3,1,1), unita, omega_h3_x_vector_h3r)$
omega_h3_x_vector_h3r : subst( part(omega_h3,1,2), unitb, omega_h3_x_vector_h3r)$
omega_h3_x_vector_h3r : subst( part(omega_h3,1,3), unite, omega_h3_x_vector_h3r)$
omega_h3_x_vector_h3r : subst( part(omega_h3,1,4), unitc, omega_h3_x_vector_h3r)$
omega_h3_x_vector_h3r : subst( part(omega_h3,1,5), unitd, omega_h3_x_vector_h3r)$
omega_h3_x_vector_h3r : subst( part(omega_h3,1,6), unite, omega_h3_x_vector_h3r)$

part1 : part(omega_h3_x_vector_h3r, 1)$
part2 : part(omega_h3_x_vector_h3r, 2)$
part3 : part(omega_h3_x_vector_h3r, 3)$

part3 : factor( trigreduce( part3 ) )$

omega_h3_x_vector_h3r : [ part1, part2, part3 ]$

part1 : part(omega_h3_x_vector_h3r, 1)$
part2 : part(omega_h3_x_vector_h3r, 2)$
part3 : part(omega_h3_x_vector_h3r, 3)$

part3 : factor( trigreduce( part3 ) )$

omega_h3_x_vector_h3e : [ part1, part2, part3 ]$

/* Set up the displacement vector and the velocity vector for the elastic portion of hydraulic cylinder 2. */

vector_h3rd : velocity_at_1 + omega_h3_x_vector_h3r$
vector_h3rdsq : trigsimp( vector_h3rd . vector_h3rd )$

projection_uh3 : cross_product(i, j, k)$
projection_uh3 : [ part1, part2, part3 ]$

projection_vh3 : cross_product(i, j, k)$
projection_vh3 : [ part1, part2, part3 ]$
temp : transform_h3_to_2, matrix ([ 0 ], [ 0 ], [ 1 ])

temp : transpose ( temp )$

part1 : part ( temp, 1, 1 )$

part2 : part ( temp, 1, 2 )$

part3 : part ( temp, 1, 3 )$

projection_wh3 : cross_product(i, j, k)$

vector_h3ed : velocity_at_1 + omega_h3_x_vector_h3e + uh3d * projection_uh3 + vh3d * projection_vh3 + wh3d * projection_wh3$

vector_h3edsq : vector_h3ed . vector_h3ed$

/* ********************************************************************************
   That's it folks...
*/
APPENDIX G

FORTRAN PROGRAM LISTINGS OF THE TWO–LINK ROBOT WITH
ONE FLEXIBLE OUTER–MOST LINK
Program name : 2links.f
This is the main driver for the vibration analysis of a two-link
robot with one elastic links.

Call subroutine 'init' to get input from an input file and
initialize variables.

Call subroutine 'prtparam' to print out the system parameters.

Call subroutine 'solver' to begin calculation.

That's it, late...

stop
end

subroutine admissfn ( whatlink )
Program name : Admissfn.f
This is the subroutine that helps to estimate the values of the
admissible function gammas for link 2.

Declare variables and include the common blocks.
implicit real (a-z)
integer power, maxpower, sign1, sign2
character * 3 whatlink

Initialize variables.
power = 1
maxpower = 10
estimate = 1.0 / ( 10.0 ** power )
beta = estimate

First, calculate the determinate of the matrix with beta = 0.0

call betamat ( whatlink, beta, determinate1 )
if ( determinate1 .eq. 0.0 ) then
   sign1 = 0
else
   if ( determinate1 .lt. 0.0 ) then
      sign1 = -1
else
   end if
endif
else
    sign1 = 1
end if
end if

Using numerical approximation method to estimate beta.

beta = beta + estimate

call betamat (whatlink, beta, determinate2)

if (determinate2 .eq. 0.0) then
    sign2 = 0
else
    if (determinate2 .lt. 0.0) then
        sign2 = -1
    else
        sign2 = 1
    end if
end if

if (determinate1 .eq. determinate2) then
    if (power .eq. maxpower) then
        call calgamma (whatlink, beta)
        return
    else
        if (power .eq. maxpower) then
            call calgamma (whatlink, beta)
            return
        else
            power = power + 1
            beta = beta - estimate
            estimate = 1.0 / (10.0 ** power)
            goto 10
        end if
    end if
end if

if (sign2 .eq. 0) then
    call calgamma (whatlink, beta)
    return
else
    if (sign1 .ne. sign2) then
        if (power .eq. maxpower) then
            call calgamma (whatlink, beta)
            return
        else
            power = power + 1
            beta = beta - estimate
            estimate = 1.0 / (10.0 ** power)
            goto 10
        end if
    else
        determinate1 = determinate2
    end if
    goto 10
end if

c Mission accomplished, later...
end

subroutine betamat (whatlink, beta, determinate)

c Program name: Betamat.f

c This is a subroutine that helps to determine the values for the beta matrix for link 2.
Declare variables and include the common blocks.

```
implicit real (a-z)
integer numsegment
character * 3 whatlink
parameter ( numsegment = 100 )

dimension betamatrix (4,4)
dimension u2 (0:numsegment)
common /link2/ a2r, a2rsq, a2rcu, rho2r, mass2r,
*    a2e, rho2e, mass2e, sm2, lm2, eiv2,
*    u2, a2, a2sq, costh2, costh2sq, sinh2
```

Initialize variables.

```
betasq = beta ** 2.0
betacu = beta ** 3.0
betafo = beta ** 4.0
sine = sin ( beta )
sineh = sinh ( beta )
cosine = cos ( beta )
cosineh = cosh ( beta )
```

Perform calculations.

```
if ( whatlink .eq. ' v2' ) then
  tau = ( sm2 * betafo ) / ( rho2e * a2e )
  betamatrix (1,1) = 0.0
  betamatrix (1,2) = 1.0
  betamatrix (1,3) = 0.0
  betamatrix (1,4) = 1.0
  betamatrix (2,1) = 1.0
  betamatrix (2,2) = 0.0
  betamatrix (2,3) = 1.0
  betamatrix (2,4) = 0.0
  betamatrix (3,1) = -sine
  betamatrix (3,2) = -cosine
  betamatrix (3,3) = sineh
  betamatrix (3,4) = cosineh
  betamatrix (4,1) = -( betacu * cosine ) + ( tau * sine )
  betamatrix (4,2) = ( betacu * sine ) + ( tau * cosine )
  betamatrix (4,3) = ( betacu * cosineh ) + ( tau * sineh )
  betamatrix (4,4) = ( betacu * sineh ) + ( tau * cosineh )
end if
```

Call subroutine to find out the determinate of the matrix.

```
call det4x4 ( betamatrix, determinate )
```

Mission accomplished, see you...

```
return
end
```
subroutine calcoeff ( neq, yprime )

Program name: Calcoeff.f

This subroutine helps to calculate the coefficients of the system.

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

Declare variables.
implicit real (a-z)
integer neq
dimension yprime ( neq )
********************************************************************************

Calculate the coefficients for link 2.
call clink2
********************************************************************************

Calculate the coefficients for the whole system.
call coefftot ( neq, yprime )
********************************************************************************

Time to go, later...
return
end

subroutine calgamma ( whatlink, beta )

Program name: calgamma.f

This is the subroutine that helps to calculate the values of the admissible function gammas.

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

Declare variables and include the common blocks.
implicit real (a-z)
character * 3 whatlink
integer i, numsegment
parameter ( numsegment = 100 )
dimension u2 (0:numsegment)
dimension gv2 (0:numsegment), gv2sq (0:numsegment),
  * gv2p (0:numsegment), gv2p2sq (0:numsegment),
  * gv2pp (0:numsegment), gv2pppsq (0:numsegment)
common /constant/  g, pi, segment
common /link2/ a2r, a2rsq, a2rcu, rho2r, mass2r,
  * a2e, rho2e, mass2e, sm2, ln2, eiv2,
  * u2, a2, a2sq, costh2, costh2sq, sinh2
common /link2gs/ gv2, gv2sq, gv2p, gv2p2sq, gv2pp, gv2pppsq,
*              gv2a2, gv2a2sq, gv2a2p, gv2a2psq

** Initialize variable. **

    betasq = beta ** 2.0 
    betacu = beta ** 3.0 
    betafo = beta ** 4.0 
    sinbeta = sin ( beta ) 
    sinhbeta = sinh ( beta ) 
    cosbeta = cos ( beta ) 
    coshbeta = cosh ( beta ) 

** Calculate gammav2, gammav2' and gammav2'' for link 2. **

if ( whatlink .eq. ' v2' ) then 
    del = -( sinbeta + sinhbeta ) / ( cosbeta + coshbeta )
    do 10 i = 0, numsegment 
        temp = beta * ( float ( i ) / segment ) 
        sine = sin ( temp ) 
        sinh = sinh ( temp ) 
        cosine = cos ( temp ) 
        cosh = cosh ( temp ) 
        u2 ( i ) = a2r + ( ( float ( i ) / segment ) * a2e ) 
        * 
        gv2 ( i ) = ( ( sine - sinh ) + 
                     ( del * ( cosine - cosh ) ) ) 
        * 
        gv2p ( i ) = beta * ( ( cosine - cosh ) + 
                       ( del * ( -sine - sinh ) ) ) 
        * 
        gv2pp ( i ) = betasq * ( ( -sine - sinh ) + 
                           ( del * ( -cosine - cosh ) ) ) 
        if ( i .eq. numsegment ) then 
            test = gv2pp ( i ) 
            if ( test .lt. 0.0 ) then 
                test = -test 
            end if 
            if ( test .lt. 10.0**(-6.0) ) then 
                gv2pp(i) = 0.0 
            end if 
            end if 
        gv2sq ( i ) = gv2 ( i ) ** 2.0 
        gv2psq ( i ) = gv2p ( i ) ** 2.0 
        gv2ppsq ( i ) = gv2pp ( i ) ** 2.0 
    continue 
    gv2a2 = gv2 (numsegment) 
    gv2a2p = gv2p (numsegment) 
    gv2a2sq = gv2a2 ** 2.0 
    gv2a2psq = gv2a2p ** 2.0 
end if 

** That's it, later... **

return 
end
subroutine caltorqu
Program name: Caltorqu.f
This subroutine helps to calculate the torques of the system.
******************************************************************
Calculate the torques for link 2.
call tlink2
******************************************************************
Calculate the torques for the whole system.
call torqutot
******************************************************************
Time to go, later...
return
end

subroutine clink2
Program name: Clink2.f
This subroutine helps to setup and calculate the coefficients for
link 2.
******************************************************************
Declare variables and include the common blocks.
implicit real (a-z)
go to the previous page
element(3,i) = gv2sq(i)
element(4,i) = gv2(i)*u2(i)

20 continue

call integrat ( element, maxarray, numsegment, a2r, a2, integral )
c
******************************************************************
c Set variables.
e13 = integral(1)*rho2e
e14 = gv2a2sq*sm2
e15 = integral(2)*gv2*gv2

continue

call integrat ( element, maxarray, numsegment, a2r, a2, integral )
c
******************************************************************

coeffal2 = e14+e13
c
coefffl2 = e20+e19+e18+e17+e16+e15
c
******************************************************************

Mission complete, see you...
return
end

subroutine coeffO

Program name: Coeff0.f

This subroutine helps to setup the zero coefficients for all links.

Declare variables and include the common blocks.

implicit real (a-z)

common /coeff11/ coeffall, coefffl1
c
common /coeff12/ coeffal2, coefffl2
c

Set zero coefficient for link 1.

coeffal1 = 0.0
coefffl1 = 0.0
c

Set zero coefficient for link 2.

coeffal2 = 0.0
coefffl2 = 0.0
c

Mission complete, see you...
subroutine coefftot ( neq, yprime )
c  Program name : Coefftot.f
c  This subroutine helps to add the coefficients of all links
c  together and store the values into an array.
c *************************************************************************
c  Declare variables and include the common blocks.
  implicit real (a-z)
  integer neq
dimension yprime (neq)
common /link2qs/ qv2, qv2sq, qv2d, qv2dsq, qv2dd
common /coeff11/ coeffall, coefffll
common /coeffl2/ coeffal2, coefffl2
  *************************************************************************
c  Add the coefficients together.
coeffa = coeffal2+coeffal1
coefff = -coefff12-coefff11
  *************************************************************************
c  Calculate the value for qv2.
yprime (2) = coefff / coeffa
qv2dd = yprime (2)
  *************************************************************************
c  Time to go, bye...
return
end

subroutine det3x3 ( matrix, determinate )
c  Program name : Det3x3.f
c  This is the subroutine that helps to calculate the determinate of
c  a 3x3 matrix.
c  *************************************************************************
c  Declare variable.
  implicit real (a-z)
dimension matrix (3,3)
  *************************************************************************
c  Perform calculation.

determinate = ( matrix (1,1) * matrix (2,2) * matrix (3,3) ) + 
  ( matrix (1,2) * matrix (2,3) * matrix (3,1) ) + 
  ( matrix (1,3) * matrix (2,1) * matrix (3,2) ) - 
  ( matrix (1,1) * matrix (2,3) * matrix (3,2) ) - 
  ( matrix (1,2) * matrix (2,1) * matrix (3,3) ) 

******************************************************************
That's it, later...
return
end

subroutine det4x4 ( betamatrix, determinate )
  Program name : Det4x4.f
  This is the subroutine that helps to calculate the determinate of
  a 4x4 matrix.
  Declare variables.
  implicit real (a-z)
  integer i, k, l, m, sign
  dimension betamatrix (4,4), matrix (3,3)
  Perform calculations.
  determinate = 0.0
  do 10 i = 1, 4
    if ( i .eq. 1 ) then
      k = 2
      l = 3
      m = 4
    else if ( i .eq. 2 ) then
      k = 1
      l = 3
      m = 4
    else if ( i .eq. 3 ) then
      k = 1
      l = 2
      m = 4
    else if ( i .eq. 4 ) then
      k = 1
      l = 2
      m = 3
    end if
    matrix (1,1) = betamatrix (2,k)
    matrix (1,2) = betamatrix (2,l)
    matrix (1,3) = betamatrix (2,m)
    matrix (2,1) = betamatrix (3,k)
    matrix (2,2) = betamatrix (3,l)
    matrix (2,3) = betamatrix (3,m)
    matrix (3,1) = betamatrix (4,k)
    matrix (3,2) = betamatrix (4,l)
    matrix (3,3) = betamatrix (4,m)
  end do
call det3x3 ( matrix, determinate3x3 )

  sign = (-1) ** ( 1 + i )
  determinate = determinate +
                ( sign * betamatrix (1,i) * determinate3x3 )

10 continue

** Time to go, see you...

return
end

subroutine fcn ( neq, x, y, yprime )

  Program name : Fcn.f

  This is a subroutine that helps to set up the condition of theta 2
  and then calculate the corresponding q and q'.

  Generates the equations used by IMSL (lvpag).

  Called by 'lvpag' in 'solver'.

  Calls 'calco eff'.

  Declare variables and include the common blocks.
  implicit real (a-z)
  integer neq
  dimension y (neq), yprime (neq)

**

  Call subroutine to set variables.
  call setvar ( neq, x, y, yprime )

**

  Call subroutine to find out the values of the coefficients.
  call calcoeff ( neq, yprime )

**

  Time to go, later...

  return
end

subroutine fcnj ( neq, x, y, dypdy )

  Program name : Fcnj.f

  This is a dummy subroutine that do not do anything.

**
initialize variables.

implicit real (a-z)

integer neq
dimension y (neq), dypy (neq,neq), x

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

later...

return
end

subroutine init

program name : init.f

subroutine to initialize variables and get value from the user.

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

declare variables and include the common blocks.

implicit real (a-z)

integer numstep, numsegment

parameter ( numsegment = 100 )

character * 3 whatlink
character * 4 profiletype

dimension u2 (0:numsegment)
dimension polyd2 (5), polye2 (5)

common /constant/ g, pi, segment
common /general/ length1,
  * density2r, length2r, iwidth2r, iheight2r,
  * owidth2r, oheight2r, csarea2r,
  * density2e, elasticity2, areamoi2, length2e,
  * iwidth2e, iheight2e, owidth2e, oheight2e,
  * csarea2e, lumpedmass2,
  * theta2i, theta2f
common /link1/ d1
common /link2/ a2r, a2rsq, a2rcu, rho2r, mass2r,
  * a2e, rho2e, mass2e, sm2, ln2, eiv2,
  * u2, a2, a2sq, cosh2, cosh2sq, sinh2
common /theta2/ th2i, th2f, th2, th2o, th2dsq, th2dd
common /profile/ profiletype, total, motion, numstep, interval,
  * bang2, polyd2, polye2
common /imsl/ hinit, maxstep, toleranc

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

read data from the input file.

open ( unit = 10, file = 'input.fil', status = 'old' )

read (10,*) length1
read (10,*) density2r
read (10,*) length2r, iwidth2r, iheight2r, owidth2r, oheight2r
read (10,*) density2e, elasticity2, lumpedmass2
read (10,*) length2e, iwidth2e, iheight2e, owidth2e, oheight2e
read (10,*) theta2i
read (10,*) theta2f
read (10,*) profiletype
read (10,*) totalt, motiont, numstep
read (10,*) maxstep, toleranc

close ( unit = 10 )

******
Set variables for later use.
g = 9.80665
pi = 3.141592654
radian = pi / 180.0
segment = 100.0
******

Set variables for link 1.
d1 = length1

******
Set variables for the rigid portion of link 2.
a2r = length2r
a2rsq = a2r ** 2.0
a2rcu = a2r ** 3.0
csarea2r = ( owidth2r * oheight2r ) - ( iwidth2r * iheight2r )
rho2r = density2r * csarea2r
mass2r = rho2r * a2r
******

Set variables for the elastic portion of link 2.
a2 = length2r + length2e
a2sq = a2 ** 2.0
a2e = length2e
csarea2e = ( owidth2e * oheight2e ) -
* ( iwidth2e * iheight2e )
areamoi2 = ( ( owidth2e * ( oheight2e ** 3.0 ) ) -
* ( iwidth2e * ( iheight2e ** 3.0 ) ) ) / 12.0
rho2e = density2e * csarea2e
mass2e = rho2e * a2e
sm2 = lumpedmass2
lm2 = lumpedmass2
eiv2 = elasticity2 * areamoi2
******

Set variables for the initial and final displacement parameters
for the links.

th2i = theta2i * radian
th2f = theta2f * radian

******
Find out the acceleration profile and determine the necessary
values for the profile.
call profiles
******

Set up some of the IMSL parameters.
interval = totalt / float ( numstep )
hinit = 0.01 * interval / 2.0
******
Call subroutine to estimate the values for the admissible function gamma for link 2.
whatlink = ' v2'
call admisssfn ( whatlink )

Call subroutines to initialize all coefficients and torques to zero.
call coeff0
call torque0

That's it, later...
return
end

subroutine initdefl ( neq, y )

Program name : Initdefl.f

Subroutine that helps to find out the initial deflections of link 2.

Declare variables and include the common blocks.
implicit real (a-z)
integer neq, numsegment
parameter ( numsegment = 100 )
dimension y (neq)
dimension u2 (0:numsegment), gv2sq (0:numsegment),
*      gv2p (0:numsegment), gv2psq (0:numsegment),
*      gv2pp (0:numsegment), gv2ppsq (0:numsegment)
common /constant/ g, pi, segment
common /link2/ a2r, a2rsq, a2rsqu, rho2r, mas2r,
*     a2e, rho2e, mass2e, sm2, le2, eiv2,
*     u2, a2, a2sq, costh2, costh2sq, sinth2
common /link2gs/ gv2, gv2sq, gv2psq, gv2ppsq, gv2psq,
*      gv2a2, gv2e2sq, gv2a2p, gv2a2psq
common /theta2/ th2i, th2f, th2, th2d, th2dsq, th2dd

The initial deflection for link 3.
v2a2 = ( sm2 * g * a2e ** 3.0 ) / ( 3.0 * eiv2 ) +
*     ( rho2e * g * a2e ** 4.0 ) / ( 8.0 * eiv2 )
v2a2 = -v2a2 * cos ( th2i )

Setup the initial boundary conditions for y.
y (1) = v2a2 / gv2a2
subroutine integrat ( element, maxarray, numsegment, upperlimit, lowerlimit, integral )

Program name : Integrat.f

This is a subroutine that make use of the 'Trapezoid rule' to calculate integration.

Declare variables.

implicit real (a-z)
integer i, j, maxarray, numsegment
dimension element (maxarray,0:numsegment), integral (maxarray)
common /constant/ g, pi, segment

Perform calculations.

intervalwidth = (lowerlimit - upperlimit) / segment

do 10 i = 1, maxarray
    integral (i) = 0.0
    do 20 j = 1, numsegment
        integral (i) = integral (i) + element (i,j-1) + element (i,j)
    20 continue
    integral (i) = integral (i) * intervalwidth / 2.0
10 continue

That's it, later...

return
end

subroutine outputs ( x )

Program name : Outputs.f

This is the subroutine that helps to print the outputs into files.

Declare variables and include the common blocks.
implicit real (a-z)

integer numsegment

parameter ( numsegment = 100 )

dimension torque (2)
dimension g2 (0:numsegment), g2sq (0:numsegment),
* g2p (0:numsegment), g2psq (0:numsegment),
* g2pp (0:numsegment), g2ppsq (0:numsegment)

c  common /constant/ g, pi, segment
c  common /link2gs/ g2, g2sq, g2p, g2psq, g2pp, g2ppsq,
* g2a2, g2a2sq, g2a2p, g2a2psq

c  common /link2qs/ qv2, qv2sq, qv2d, qv2dsq, qv2dd
c  common /theta2/ th2i, th2f, th2, th2d, th2dsq, th2dd
c  common /torques/ torque
c  common /vs/ v2a2

c  ****************************************************************************
c  Set up the format statement.
10 format ( f6.4, a1, f8.5 )
20 format ( f6.4, a1, f8.4 )
30 format ( f6.4, a1, f8.5, a1, f8.5, a1, f8.3 )

c  ****************************************************************************
c  Find out the deflections and print them onto the output file.
write (10,10) x, v2a2

c  ****************************************************************************
c  Print the torques onto the output file.
write (11,20) x, torque(2) / 1000.0

c  ****************************************************************************
c  Print the displacement, velocity and acceleration profiles.
write (12,30) x, th2, th2d, th2dd

c  ****************************************************************************
c  Mission accomplished, see you...
return
end

subroutine profiles

c  Program name : Profiles.f

c  This subroutine helps to determine the values of the acceleration
profile.

c  ******************************************************************************
c  Declare variables and include the common blocks.

implicit real (a-z)

integer numstep
integer ipath, lda, n
parameter ( ipath = 1, lda = 5, n = 5 )

character * 4 profiletype

dimension polyd2 (5), polye2 (5)
dimension matrixa (n,n), matrixb (n)

common /theta2/ th2i, th2f, th2, th2d, th2dsq, th2dd
common /profile/ profiletype, totalt, motiont, numstep, interval,
*   bang2, polyd2, polye2

******************************************************************
Find out what kind of acceleration profile.

if ( profiletype .eq. 'bang' ) then
   bang2 = ( th2f - th2i ) / ( ( motiont / 2.0 ) ** 2.0 )
else
   delta2 = ( th2f - th2i ) / 2.0

   tau1 = motiont / 2.0
   tau1sq = tau1 ** 2.0
   tau1cu = tau1 ** 3.0
   tau1fo = tau1 ** 4.0
   tau1fi = tau1 ** 5.0
   tau1si = tau1 ** 6.0
   tau1se = tau1 ** 7.0
   tau1ei = tau1 ** 8.0

   tau2 = motiont / 4.0
   tau2sq = tau2 ** 2.0
   tau2cu = tau2 ** 3.0
   tau2fo = tau2 ** 4.0
   tau2fi = tau2 ** 5.0

   matrixa (1,1) = tau1fo
   matrixa (1,2) = tau1fi
   matrixa (1,3) = tau1si
   matrixa (1,4) = tau1se
   matrixa (1,5) = tau1ei
   matrixa (2,1) = 12.0 * tau1sq
   matrixa (2,2) = 20.0 * tau1cu
   matrixa (2,3) = 30.0 * tau1fo
   matrixa (2,4) = 42.0 * tau1fi
   matrixa (2,5) = 56.0 * tau1si
   matrixa (3,1) = 24.0 * tau2
   matrixa (3,2) = 60.0 * tau2sq
   matrixa (3,3) = 120.0 * tau2cu
   matrixa (3,4) = 210.0 * tau2fo
   matrixa (3,5) = 336.0 * tau2fi
   matrixa (4,1) = 24.0 * tau1
   matrixa (4,2) = 60.0 * tau1sq
   matrixa (4,3) = 120.0 * tau1cu
   matrixa (4,4) = 210.0 * tau1fo
   matrixa (4,5) = 336.0 * tau1fi
   matrixa (5,1) = 24.0
   matrixa (5,2) = 120.0 * tau1
   matrixa (5,3) = 360.0 * tau1sq
   matrixa (5,4) = 840.0 * tau1cu
   matrixa (5,5) = 1680.0 * tau1fo

   matrixb (2) = 0.0
   matrixb (3) = 0.0
   matrixb (4) = 0.0
   matrixb (5) = 0.0

   matrixb (1) = delta2
call Isarg ( n, matrixa, lda, matrixb, ipath, polyd2 )
matrixb (1) = -delta2
call Isarg ( n, matrixa, lda, matrixb, ipath, polye2 )
end if

**********************************************************************************************************
That's it, later...
return
end

subroutine prtparam

Program name: Prtparam.f

This is the subroutine that helps to print the parameters into
and output file.

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

Declare variables and include the common blocks.

implicit real (a-z)
integer numsegment, numstep
character * 4 profiletype
parameter ( numsegment = 100 )
dimension u2 (0:numsegment)
dimension polyd2 (5), polye2 (5)
common /general/ length1,
* density2r, length2r, iwidth2r, iheight2r,
* owidth2r, oheight2r, csarea2r,
* density2e, elasticity2, areamoi2, length2e,
* iwidth2e, oheight2e, owidth2e, oheight2e,
* csarea2e, lumpedmass2,
* theta2i, theta2f
common /link1/ d1
common /link2/ a2r, a2rsq, a2rcu, rho2r, mass2r,
* a2a, rho2e, mass2e, sm2, lam2, elv2,
* u2, a2, a2sq, costh2, costh2sq, sinth2
common /theta2/ th2i, th2f, th2, th2d, th2dsq, th2dd
common /profile/ profiletype, totalt, motiont, numstep, interval,
* bang2, polyd2, polye2
common /imsl/ hinit, maxstep, toleranc

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

Initialize variables.

10 format ( a50 )
20 format ( a50, 5x, e11.5 )
30 format ( a50, 5x, f11.5 )
40 format ( a50, 5x, f7.5, a3, f7.5 )

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

Put the general information into 'output.fil' file.

if ( profiletype .eq. 'bang' ) then

open (unit = 10, file = 'boutput.fil')
else
  open (unit = 10, file = 'poutput.fil')
end if

write (10,10) ! ***** System Parameters *****
write (10,) !
write (10,10) ! Units are in SI unit: kg, meter and second
write (10,*) !
write (10,10) ! Link 1 Information
write (10,*) !
write (10,30) ! Total length of link 1 (m) =
  !
write (10,*) !
write (10,30) ! Link 2 Information
write (10,*) !
write (10,30) ! Total length of link 2 (m) =
  !
write (10,*) !
write (10,30) ! For the rigid portion of link 2
write (10,10) !
write (10,10) ! Length (m) =
  !
write (10,30) ! Cross sectional dimension
write (10,10) !
write (10,40) ! i) Inside width x height (m) =
  !
write (10,30) ! ii) Outside width x height (m) =
  !
write (10,30) ! Cross section area (m**2) =
  !
write (10,30) ! Density (kg/m**3) =
  !
write (10,30) ! Mass per unit length (kg/m) =
  !
write (10,30) ! Mass (kg) =
  !
write (10,*) !
write (10,30) ! For the elastic portion of link 2
write (10,10) !
write (10,20) ! Modulus of elasticity (Pa) =
  !
write (10,30) ! Length (m) =
  !
write (10,30) ! Cross sectional dimension
write (10,10) !
write (10,40) ! i) Inside width x height (m) =
  !
write (10,30) ! ii) Outside width x height (m) =
  !
write (10,30) ! Cross section area (m**2) =
  !
write (10,30) ! Density (kg/m**3) =
  !
write (10,30) ! Mass per unit length (kg/m) =
  !
write (10,30) ! Mass (kg) =
  !
write (10,20) ! Area moment of inertia (m**4) =
  !
write (10,30) ! Lump mass at the end of link 2 (kg) =
  !
write (10,*) !
write (10,10) ! General Information
write (10,*) !
write (10,30) ! Initial position of link 2 (rad) =
  !
write (10,30) ! Initial position of link 2 (deg) =
write (10,30) 'Final position of link 2 (rad) = ', th2f
write (10,30) 'Final position of link 2 (deg) = ', theta2f
write (10,30) 'Duration of the experiment (s) = ', totalt
write (10,30) 'Duration of motion (s) = ', motion
write (10,30) 'Number of steps within the experiment time frame = ', numstep
write (10,30) 'Sampling interval (s) = ', interval
write (10,30) 'The initial value of the step size H for IMSL = ', hinit
write (10,30) 'The maximum steps for IMSL = ', maxstep
write (10,30) 'The tolerance (TOL) for IMSL = ', toleranc

C *****************************************************
C Close the output file.
close (unit = 10)
C *****************************************************
C That's it, time to go, see you...
return
eend

subroutine setvar ( neq, x, y, yprime )
C Program name : Setvar.f
C This subroutine helps to set up values for all links with respect to the time steps.
C *****************************************************
C Declare variables and include the common blocks.
imPLICIT REAL (A-Z)
integer neq, numsegment
parameter ( numsegment = 100 )
dimension y (neq), yprime (neq)
dimension u2 (0:numsegment), g2sq (0:numsegment),
* g2p (0:numsegment), g2psq (0:numsegment),
* g2pp (0:numsegment), g2ppsq (0:numsegment)
commOn /link2/ a2r, a2rsq, a2rcu, rho2r, mass2r,
* a2e, rho2e, mass2e, sn2, ln2, ein2,
* u2, a2, a2sq, costh2, costhsq, sinh2
C commOn /link2qs/ g2, g2sq, g2d, g2dsq, g2dd
C commOn /link2gs/ g2v, g2vsq, g2vp, g2vpsq, g2vpp, g2vppsq
* g2vds, g2vsdsq, g2vsd, g2vdsq, g2vdd
C commOn /theta2/ th2i, th2f, th2, th2d, th2dsq, th2dd
C commOn /vs/ v2a2
C *****************************************************
C Calculate the input angle time history.
call thistory ( x )

c Setup yprimes.

ypprime (1) = y (2)

c

subroutine solver

c Program name: Solver.f

c This is the subroutine that helps to solve our problem.

c Declare variables and include the common blocks.

external fcn, fcnj
implicit real (a-z)
integer i, numstep
integer neq, nparam, ido, inorm, meth, miter
character * 4 profiletype
parameter ( neq = 2, nparam = 50 )
dimension y (neq), ypprime (neq), param (nparam)
dimension polyd2 (5), polye2 (5)
common /profile/
profiletype, totalt, motiont, numstep, interval,
* bang2, polyd2, polye2
common /imsl/
hinit, maxstep, toleranc

c

c Set variable.

x = 0.0

c

c Calculate initial conditions for the ys (qs) from the given
boundary condition.

\[ y(1) = 0.0 \]
\[ y(2) = 0.0 \]
\[ y'(1) = 0.0 \]
\[ y'(2) = 0.0 \]

call setvar ( neq, x, y, yprime )
call initdefl ( neq, y )

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

Open the output files.

if ( filetype .eq. 'bang' ) then
    open ( unit = 10, file = 'bdeflections.fil' )
    open ( unit = 11, file = 'btorques.fil' )
    open ( unit = 12, file = 'bprofile.fil' )
else
    open ( unit = 10, file = 'pdeflections.fil' )
    open ( unit = 11, file = 'ptorques.fil' )
    open ( unit = 12, file = 'pprofile.fil' )
end if

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

Call subroutine 'fcn' to set up the variables for the coefficients and torques, call 'caltorqu' to calculate the result at time 0 and call 'error' to calculate the position of the end-effector at time zero.

call fcn ( neq, x, y, yprime )
call caltorqu ( x )
call outputs ( x )

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

Set values for the IMSL subroutine.

ido = 1
inorm = 1
meth = 2
miter = 2
tol = tolerance

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

Call the IMSL subroutines. Then call 'caltorqu' to calculate the corresponding torque for the three joints and call 'error' to calculate the position of the end-effector.

call ssset ( nparam, 0.0, param, 1 )

param (1) = hinit
param (4) = maxstep
param (10) = inorm
param (12) = meth
param (13) = miter

do 20 i = 1, numstep
    xend = float ( i ) * interval
    call ivpag ( ido, neq, fcn, fcnj, a, x, xend, tol, param, y )
call fcn ( neq, x, y, yprime )
call caltorqu ( x )
call outputs ( x )

20 continue
cont i nue

ido = 3

call ivpag (ido, neq, fcn, fcnj, a, x, xend, tol, param, y)

Close the output files.
close (unit = 10)
close (unit = 11)
close (unit = 12)

Mission accomplished, see you...
return
end

Subroutine history (x)
Program name: History.f
This subroutine helps to set up the time history of the acceleration profile, velocity profile and the displacement profile.

Declare variables and include the common blocks.
imPLICIT REAL (A-Z)
iNTEGER numstep
INTEGER i, j
CHARACTER *4 profiletype
dIMENSION polyd2 (5), polye2 (5)
COMMON /theta2/ th2i, th2f, th2, th2d, th2dsq, th2dd
COMMON /profile/ profiletype, totalt, motiont, numstep, interval,
                 * bang2, polyd2, polye2

Calculate the input angle time history.
if (profiletype .EQ. "bang") then
  if (x .LT. 0.5 * motiont) then
    th2   = th2i + ( 0.5 * bang2 * ( x ** 2.0 ) )
th2d   = bang2 * x
th2dsq = th2d ** 2.0
th2dd  = bang2
  else if (( x .GE. 0.5 * motiont) .AND.
               *( x .LT. motiont )) then
    th2   = th2f - ( 0.5 * bang2 * ((motiont-x) ** 2.0 ))
th2d   = bang2 * (motiont - x)
th2dsq = th2d ** 2.0
th2dd  = -bang2

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
else
    th2  = th2f
    th2d = 0.0
    th2dsq = 0.0
    th2dd = 0.0
end if
else
    if ( x .lt. 0.5 * motiont ) then
        th2  = th2i
        th2d = 0.0
        th2dd = 0.0
        do 10 i = 4, 8
            j = i - 3
            th2 = th2 + polyd2 (j) * ( x ** float (i) )
            th2d = th2d + float (i) * polyd2 (j) * 
                  ( x ** float (i-1) )
            th2dd = th2dd + float (i) * float (i-1) * polyd2 (j) * 
                   ( x ** float (i-2) )
        10 continue
        th2dsq = th2d ** 2.0
    else if ( ( x .ge. 0.5 * motiont ) .and. 
              ( x .lt. motiont ) ) then
        th2  = th2f
        th2d = 0.0
        th2dd = 0.0
        do 20 i = 4, 8
            j = i - 3
            th2 = th2 + polye2 (j) * 
                  ( ( motiont - x ) ** float (i) )
            th2d = th2d + float (i) * polye2 (j) * 
                   ( ( motiont - x ) ** float (i-1) )
            th2dd = th2dd + float (i) * float (i-1) * polye2 (j) * 
                    ( ( motiont - x ) ** float (i-2) )
        20 continue
        th2d = - th2d
        th2dsq = th2d ** 2.0
    else
        th2  = th2f
        th2d = 0.0
        th2dsq = 0.0
        th2dd = 0.0
    end if
end if

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
c Time to go, later...
return
end
subroutine tlink2

Program name: Tlink2.f

This subroutine helps to setup and calculate the torques for link 2.

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

Declare variables and include the common blocks.

implicit real (a-z)

integer i, maxarray, numsegment

parameter ( maxarray = 2, numsegment = 100 )

dimension u2 (0:numsegment)

dimension gv2 (0:numsegment), gv2sq (0:numsegment),
  * gv2p (0:numsegment), gv2psq (0:numsegment),
  * gv2pp (0:numsegment), gv2ppsq (0:numsegment)

dimension element (maxarray,0:numsegment), integral (maxarray)

common /constant/ g, pi, segment

common /link2/ a2r, a2rsq, a2rcu, rho2r, mass2r,
  * a2e, rho2e, mass2e, sm2, ln2, elv2,
  * u2, a2, a2sq, costh2, costh2sq, sinh2

common /link2gs/ gv2, gv2sq, gv2p, gv2psq, gv2pp, gv2ppsq,
  * gv2a2, gv2a2sq, gv2a2p, gv2a2psq

common /link2qs/ qv2, qv2sq, qv2d, qv2dsq, qv2dd

common /theta2/ th2i, th2f, th2, th2d, th2dsq, th2dd

common /torque2l2/ torque2l2

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

Calculate the values of the integrals of link 2.

do 10 i = 0, numsegment

  element(1,i) = 2*u2(i)*(u2(i)*th2dd+gv2(i)*qv2dd)+2*gv2sq(i)*qv2sq
  *th2dd+4*gv2sq(i)*qv2*qv2d*th2d

  element(2,i) = costh2*u2(i)-gv2(i)*qv2*sinth2

continue

10 continue

call integrat ( element, maxarray, numsegment, a2r, a2, integral )

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

Set variables.

e7 = integral(1)*rho2e/2.0
e8 = integral(2)*g*rho2e
e9 = sm2*(2*a2*th2dd+gv2a2*qv2dd)+2*gv2a2sq*qv2sq*th2dd+4*gv2a
  *2sq*qv2*gv2d*th2d/2.0
e10 = a2rcu*rho2r*th2dd/3.0
e11 = g*(a2*costh2-gv2a2*qv2*sinth2)*sm2

 e12 = a2rsq*costh2*ghrho2r/2.0

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

Calculate the torques with the variables just calculated.

torque2l2 = e9+e8+e7+e12+e11+e10

******************************************
subroutine torque0

Program name : Torque0.f

This subroutine helps to set the torque of the linkage to zero.

Declare variables and include the common blocks.

implicit real (a-z)

common /torque1/ torque2l1
common /torque2/ torque2l2

Zero out variables.

torque2l1 = 0.0
torque2l2 = 0.0

Mission complete, see you...

return

end

subroutine torquotot

Program name : torquotot.f

This subroutine helps to add the torques of all links together and store the values into an array.

Declare variables and include the common blocks.

implicit real (a-z)

dimension torque (2)

common /torque1/ torque2l1
common /torque2/ torque2l2
common /torques/ torque

Add the torques together.

torque(2) = torque2l2+torque2l1

Time to go, bye...

return
end
APPENDIX H

FORTRAN PROGRAM LISTINGS OF THE THREE-LINK ROBOT WITH TWO FLEXIBLE OUTER-MOST LINKS
Program name: 31inks.f

This is the main driver for the vibration analysis of a three links robot with two flexible outer-most links.

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

Call subroutine 'init' to get input from an input file and initialize variables.
call init

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

Call subroutine 'prtparam' to print out the system parameters.
call prtparam

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

Call subroutine 'solver' to begin calculation.
call solver

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

That's it, late...
stop
end

subroutine admissfn ( whatlink )

Program name: Admissfn.f

This is the subroutine that helps to estimate the values of the admissible function gamma for link 2.

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

Declare variables and include the common blocks.
implicit real (a-z)
integer power, maxpower, sign1, sign2
character * 3 whatlink

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

Initialize variables.
power = 1
maxpower = 10
estimate = 1.0 / ( 10.0 ** power )
beta = estimate

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

First, calculate the determinant of the matrix with beta = 0.0
call betamat ( whatlink, beta, determinate1 )
if ( determinate1 .eq. 0.0 ) then
   sign1 = 0
else
   if ( determinate1 .lt. 0.0 ) then
      sign1 = -1
   end if
end if

else
    sign1 = 1
end if
end if

Using numerical approximation method to estimate beta.

beta = beta + estimate

call betamat ( whatlink, beta, determinate2 )

if ( determinate2 .eq. 0.0 ) then
    sign2 = 0
else
    if ( determinate2 .lt. 0.0 ) then
        sign2 = -1
    else
        sign2 = 1
    end if
end if

if ( determinate1 .eq. determinate2 ) then
    if ( power .eq. maxpower ) then
        call calgamma ( whatlink, beta )
        return
    else
        power = power + 1
        beta = beta - estimate
        estimate = 1.0 / ( 10.0 ** power )
        goto 10
    end if
end if

if ( sign2 .eq. 0 ) then
    call calgamma ( whatlink, beta )
    return
else
    if ( sign1 .ne. sign2 ) then
        if ( power .eq. maxpower ) then
            call calgamma ( whatlink, beta )
            return
        else
            power = power + 1
            beta = beta - estimate
            estimate = 1.0 / ( 10.0 ** power )
        end if
    else
        determinate1 = determinate2
        sign1 = sign2
    end if

goto 10

Mission accomplished, later...

end

subroutine betamat ( whatlink, beta, determinate )

Program name : Betamat.f

This is a subroutine that helps to determine the values for the beta matrix for link 2.
Declare variables and include the common blocks.

implicit real (a-z)
integer numsegment
character * 3 whatlink
parameter (numsegment = 100)
dimension betamatrix (4,4)
dimension u2 (0:numsegment), u3 (0:numsegment)

common /lin1/ a2r, a2rsq, a2rcu, rho2r, mass2r,
* a2e, rho2e, mass2e, sm2, lm2, eiv2,
* u2, a2, a2sq, costh2, costh2sq, sinh2
common /lin2/ u2, a2, a2sq, costh2, costh2sq, sinh2
* psid, cospsi, cospsisq, sinpsi, sinpsisq,
* ch1, chid, chidd, coschi, sinchi, th3dth2dd

Initialize variables.

betasq = beta ** 2.0
betacu = beta ** 3.0
betafo = beta ** 4.0
sine = sin (beta)
sineh = sinh (beta)
cosine = cos (beta)
cosineh = cosh (beta)

Perform calculations.

if (whatlink.eq.'v2') then
  tau = (sm2 * betafo) / (rho2e * a2e)
else if (whatlink.eq.'v3') then
  tau = (sm3 * betafo) / (rho3 * a3)
end if

betamatrix (1,1) = 0.0
betamatrix (1,2) = 1.0
betamatrix (1,3) = 0.0
betamatrix (1,4) = 1.0
betamatrix (2,1) = 1.0
betamatrix (2,2) = 0.0
betamatrix (2,3) = 1.0
betamatrix (2,4) = 0.0
betamatrix (3,1) = -sine
betamatrix (3,2) = -cosine
betamatrix (3,3) = sineh
betamatrix (3,4) = cosineh
betamatrix (4,1) = -(betacu * cosine) + (tau * sine)
betamatrix (4,2) = (betacu * sine) + (tau * cosine)
betamatrix (4,3) = (betacu * cosineh) + (tau * sinh)
betamatrix (4,4) = (betacu * sinh) + (tau * cosineh)

Call subroutine to find out the determinate of the matrix.
call det4x4 (betamatrix, determinate)
 Mission accomplished, see you...

return
end

subroutine calcoeff ( neq, yprime )
Program name : Calcoeff.f
This subroutine helps to calculate the coefficients of the system.
Declare variables.
implicit real (a-z)
integer neq
dimension yprime ( neq )
Calculate the coefficients for link 2.
call clink2

Calculate the coefficients for link 3.
call clink3

Calculate the coefficients for the whole system.
call coefftot ( neq, yprime )
Time to go, later...
return
end

subroutine calgamma ( whatlink, beta )
Program name : Calgamma.f
This is the subroutine that helps to calculate the values of the admissible function gammas
Declare variables and include the common blocks.
implicit real (a-z)
character * 3 whatlink
integer i, numsegment
parameter ( numsegment = 100 )

dimension u2 (0: numsegment), u3 (0: numsegment)
dimension gv2 (0: numsegment), gv2sq (0: numsegment),
*   gv2p (0: numsegment), gv2ppsq (0: numsegment)
dimension gv3 (0: numsegment), gv3sq (0: numsegment),
*   gv3p (0: numsegment), gv3psq (0: numsegment),
*   gv3pp (0: numsegment), gv3ppsq (0: numsegment)

common /constant/ g, pi, segment
common /link2/   a2r, a2rsq, a2rcu, rho2r, mass2r,
*   a2e, rho2e, mass2e, sm2, lm2, eiv2,
*   u2, a2, a2sq, cost2, costh2sq, sinth2
common /link3/   u3, a3, rho3, mass3, sm3, lm3, eiv3,
*   psid, cospsi, cospsisq, sinpsi, sinpsisq,
*   ch1, chid, chidd, coschi, sinchi, th3ddth2dd
common /link2gs/ gv2, gv2sq, gv2p, gv2psq, gv2pp, gv2ppsq,
*   gv2a2, gv2a2sq, gv2a2p, gv2a2psq
common /link3gs/ gv3, gv3sq, gv3p, gv3psq, gv3pp, gv3ppsq,
*   gv3a3, gv3a3sq

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

c Initialize variable.

betasq = beta ** 2.0
betacu = beta ** 3.0
betafo = beta ** 4.0
sinbeta = sin ( beta )
sinhbeta = sinh ( beta )
cosbeta = cos ( beta )
coshbeta = cosh ( beta )

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

C Calculate gammav2, gammav2' and gammav2'' for link 2.

if ( whatlink .eq. 'v2' ) then
  
  del = -( sinbeta + sinhbeta ) / ( cosbeta + coshbeta )
  do 10 i = 0, numsegment
  temp = beta * ( float ( i ) / segment )
  sine = sin ( temp )
  sinh = sinh ( temp )
  cosine = cos ( temp )
  cosinh = cosh ( temp )
  u2 ( i ) = a2r + ( ( float ( i ) / segment ) * a2e )
  gv2 ( i ) = ( ( sine - sinh ) +
   ( del * ( cosine - cosinh ) ) )
  gv2p ( i ) = beta * ( ( cosine - cosinh ) +
   ( del * ( -sine - sinh ) ) )
  gv2pp ( i ) = betasq * (( -sine - sinh ) +
   ( del * ( -cosine - cosinh ) ) )
  if ( i .eq. numsegment ) then
    test = gv2pp ( i )
    if ( test .lt. 0.0 ) then
      test = -test
    end if
    if ( test .lt. 10.0**(-6.0) ) then
      gv2pp ( i ) = 0.0
    end if
  end if
  u2sq ( i ) = u2 ( i ) ** 2.0
  end if

******************************************************************************
gv2psq (i) = gv2p (i) ** 2.0
gv2ppsq (i) = gv2pp (i) ** 2.0

continue

gv2a2 = gv2 (numsegment)
gv2a2p = gv2p (numsegment)
gv2a2sq = gv2a2 ** 2.0

gv2a2psq = gv2a2p ** 2.0

C    Calculate gammav3, gammav3' and gammav3'' for link 3.
else if ( whatlink .eq. ' v3' ) then
    del = -( sinbeta + sinhbeta ) / ( cosbeta + coshbeta )
    do 20 i = 0, numsegment
        temp = beta * ( float (i) / segment )
        sine = sin ( temp )
        sinh = sinh ( temp )
        cosine = cos ( temp )
        cosh = cosh ( temp )
        u3 (i) = ( float (i) / segment ) * a3
          * gv3 (i) = (( sine - sinh ) +
                     ( del * ( cosine - cosh ) )
        gv3p (i) = beta * (( cosine - cosh ) +
                     ( del * ( -sine - sinh ) )
        gv3pp (i) = beta * (( -sine - sinh ) +
                        ( del * ( -cosine - cosh ) )
          if ( i .eq. numsegment ) then
            test = gv3pp (i)
            if ( test .lt. 0.0 ) then
                test = -test
            end if
            if ( test .lt. 10.0**(-6.0) ) then
                gv3pp (i) = 0.0
            end if
        end if
        gv3sq (i) = gv3 (i) ** 2.0
        gv3psq (i) = gv3p (i) ** 2.0
        gv3ppsq (i) = gv3pp (i) ** 2.0

continue

gv3a3 = gv3 (numsegment)
gv3a3sq = gv3a3 ** 2.0
end if

That's it, later...
return
end

subroutine caltorqu

Program name : Caltorqu.f

This subroutine helps to calculate the torques of the system.
Calculate the torques for link 1.
call tlink1

Calculate the torques for link 2.
call tlink2

Calculate the torques for link 3.
call tlink3

Calculate the torques for the whole system.
call torqueto

Time to go, later...
return
end

subroutine tlink2

Program name: Clink2.f

This subroutine helps to set up and calculate the coefficients for link 2.

Declare variables and include the common blocks.

implicit real (a-z)

integer i, maxarray, numsegment
parameter (maxarray = 5, numsegment = 100)

dimension u2 (0:numsegment)
dimension gv2 (0:numsegment), gv2sq (0:numsegment),
* gv2p (0:numsegment), gv2psq (0:numsegment),
* gv2pp (0:numsegment), gv2ppsq (0:numsegment)
dimension element (maxarray,0:numsegment), integral (maxarray)

common /constant/ g, pi, segment
common /link2/ a2r, a2rsq, a2rcu, rho2r, mass2r,
* a2e, rho2e, mass2e, sm2, lm2, eiv2,
* u2, a2, a2sq, cosh2, cosh2sq, sinh2
common /link2gs/
* gv2, gv2sq, gv2p, gv2psq, gv2pp, gv2ppsq,
* gv2q2, gv2sq2, gv2p2, gv2psq2, gv2pp2
common /link2qs/
* gv2q, gv2sq, gv2p, gv2psq, gv2pp
common /theta1/
* th1i, th1f, th1, th1d, th1dsq, th1dd
common /theta2/
* th2i, th2f, th2, th2d, th2dsq, th2dd
common /coeff12/
* coeff1l2, coeffb1l2, coefff1l2,
* coeff2l2, coeffb2l2, coefff2l2
Calculate the values of the integrals of link 2.

do 10 i = 0, numsegment

element(1,i) = gv2sq(i)
element(2,i) = gv2ppsq(i)
element(3,i) = 2*gv2sq(i)*qv2*th2dsq+2*gv2(i)*sinth2*th1d*(gv2(i)*
1  qv2*sinh2*th1d-costh2*u2(i)*th1d)
element(4,i) = gv2(i)*u2(i)
element(5,i) = gv2(i)

10 continue

call integrat ( element, maxarray, numsegment, a2r, a2, integral )

Set variables.
e26 = integral(1)*rhoe2
e27 = gv2a2sq*sm2

e28 = integral(2)*eiv2*qv2
e29 = -0.5*integral(5)*rhoe2
e30 = integral(4)*rhoe2*th2dd
e31 = integral(5)*cosh2*g*rhoe2

e32 = a2*gv2e2*sm2*th2dd

e33 = -sm2*((2*gv2a2sq*qv2*th2dsq+2*gv2a2*sinh2*th1d*(gv2a2*qv2+1
1  sinh2*th1d-a2*cosh2*th1d))/2.0-cosh2*g*gv2a2)

Calculate the coefficients with the variables just calculated.

coeffl12 = e27+e26
coeffl12 = e33+e32+e31+e30+e29+e28

Mission complete, see you...

return
end

subroutine clink3

Program name : Clink3.f

This subroutine helps to setup and calculate the coefficients for
link 3.

Declare variables and include the common blocks.

implicit real (a-z)

integer i, maxarray, numsegment

parameter ( maxarray = 10, numsegment = 100 )
dimension u2 (0:numsegment), u3 (0:numsegment)
dimension gv2 (0:numsegment), gv2sq (0:numsegment),
*   gv2p (0:numsegment), gv2psq (0:numsegment),
*   gv2pp (0:numsegment), gv2ppsq (0:numsegment)
dimension gv3 (0:numsegment), gv3sq (0:numsegment),
*   gv3p (0:numsegment), gv3psq (0:numsegment),
*   gv3pp (0:numsegment), gv3ppsq (0:numsegment)
Calculate the values of the integrals of link 3.

do 10 i = 0 , numsegment

\[
\text{link3sub}(1) = u_3(i) \cdot \sin \psi + \cos \psi \cdot g v_3(i) \cdot q v_3
\]

\[
\text{link3sub}(2) = \cos \psi \cdot u_3(i) - g v_3(i) \cdot g v_3(i) \cdot q v_3 \cdot \sin \psi
\]

\[
\text{link3sub}(3) = g v_3(i) \cdot g v_3(i) \cdot \sin \psi + u_3(i) \cdot \cos \psi \cdot g v_3(i) \cdot q v_3
\]

\[
\text{link3sub}(4) = -g v_3(i) \cdot q v_3 \cdot \sin \psi \cdot \sin \psi - g v_3(i) \cdot q v_3 \cdot \cos \psi \cdot \cos \psi
\]

\[
\text{link3sub}(5) = \cos \psi \cdot g v_3(i) \cdot q v_3 \cdot \sin \psi + u_3(i) \cdot g v_3(i) \cdot q v_3 \cdot \cos \psi
\]

\[
\text{link3sub}(6) = 2 \cdot u_3(i) \cdot \sin \psi + 2 \cdot \cos \psi \cdot g v_3(i) \cdot q v_3
\]

\[
\text{link3sub}(7) = 2 \cdot u_3(i) \cdot \sin \psi + 2 \cdot \cos \psi \cdot g v_3(i) \cdot q v_3
\]

\[
\text{link3sub}(8) = 2 \cdot \cos \psi \cdot g v_3(i) \cdot q v_3
\]

\[
\text{link3sub}(9) = 2 \cdot \cos \psi \cdot g v_3(i) \cdot q v_3
\]

\[
\text{link3sub}(10) = 2 \cdot g v_3(i)
\]
continue

call integrat ( element, maxarray, numsegment, 0, 'a3', integral )
c
******************************************************************************
c
Set variables.

e51 = 0.5*integral(1)*rho3
e52 = (2*link3sub(1)**2*gv2a2psq+2*link3sub(6)**2)*sm3/2.0
e53 = 0.5*integral(2)*rho3
e54 = (2*link3sub(1)*gv2a2p*gv3a3*sinpsip+2*link3sub(6)*cospsip*gv3a3
  3)*sm3/2.0
e55 = 0.5*integral(3)*rho3
e56 = 0.5*integral(4)*rho3
e57 = integral(5)*rho3

e58 = sm3*(2*link3sub(6)*((link3sub(2)*th3ddth2dd+a2*th2dd-gv3a3*ps))
  id*gv3d*sinpsii-link3sub(3)*chidi-2*link3sub(1)*gv2a2p*(-link3su
  bc1(th3ddth2dd-gv2a2p*gv2*th2dd-gv2a2*gv2d*th2dd-cospsip*gv3a3*ps)
  id*gv3d3-(link3sub(1)*chidi)-2*link3sub(4)*gv2a2p*(-gv2a2p*gv2*th2dd
  -gv3a3*gv3d*sinpsip-link3sub(1)*chidi)-2*link3sub(3)*gv2a2p*(a2*t
  h2d*cospsip*gv3a3*gv3d*gv2a2p*gv2d*link3sub(2)*chidi))/2.0

e59 = -sm3*((a3*cospsip*gv2a2p-gv2a2*gv3a3*gv3d*qv1
  3*sinpsip-cospsip*gv2a2p*gv3a3*gv3d*qv1*(gv2a2p*gv2*th2dd-gv3a3*gv3d
  3-2*link3sub(1)*chidi)*2*chidi*(-a3*gv2a2p*sinpsip-cospsip*gv2
  3*gv2a33*gv34*gv2a3p*gv2d*link3sub(3)*chidi)-2*link3sub(3)*chidi
  *2*link3sub(3)*chidi)*2*link3sub(3)*chidi)*2*link3sub(3)*chidi)*2*link3sub(3)*chidi)
  /2.0*

e60 = 0.5*integral(6)*rho3

e61 = (2*gv3a3sq*sinpsjq+2*cospsjq*gv3a3sq)*sm3/2.0

e62 = integral(7)*eiv3*qv3

e63 = 0.5*integral(8)*rho3

e64 = -0.5*integral(9)*rho3

e65 = integral(10)*cochilg*rho3

e66 = sm3*(2*cospsip*gv3a3*(link3sub(2)*th3ddth2dd+a2*th2dd-gv3a3*p)
  id*gv3d*sinpsip-link3sub(3)*chidi)-2*gv3a3*sinpsip*(-link3sub(1)*
  th3ddth2dd-gv2a2p*gv2*th2dd-gv2a2*gv2d*th2dd-cospsip*gv3a3*p
  3d-link3sub(4)*chidi)-2*cospsip*gv3a3*psid*(-gv2a2p*gv2*th2dd-gv3a3
  3d*link3sub(1)*chidi)-2*gv3a3*psid*sinpsip((a2*th2dd*cos
  3psigv3a3*gv3d*gv2a2p*gv2d*link3sub(2)*chidi))/2.0

e67 = -sm3*((a2*chidi*cospsip*gv3a37*(-gv2a2p*gv2*th2dd-gv3a3*gv3d*sinp
  3d*link3sub(1)*chidi)-2*chidi*gv3a3*sinpsip((a2*th2dd*cospsip*gv3a3*
  3d*link3sub(2)*chidi)+2*gv3a3*sinchidi*(a2*th2dd*cospsip*gv3a3*
  3*link3sub(2)*chidi)+2*gv3a3*sinchidi*(a2*th2dd*cospsip*gv3a3*
  3*link3sub(2)*chidi))/2.0-
  *coschilg*gv

******************************************************************************
c
Calculate the coefficients with the variables just calculated.

coeffa113 = e52+e51
coeffb113 = e54+e53
coefff113 = e59+e58+e57+e56+e55
coeffa213 = e54+e53
coeffb213 = e61+e60
coefff213 = e67+e66+e65+e64+e62

c******************************************************************************
c
Mission complete, see you...

return

end

subroutine coeff0
Program name: Coeff0.f

This subroutine helps to setup the zero coefficients for all links.

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

Declare variables and include the common blocks.

implicit real (a-z)

common /coeff11/ coeffa1l1, coeffb1l1, coefff1l1,
* coeffa2l1, coeffb2l1, coefff2l1
common /coeff12/ coeffa1l2, coeffb1l2, coefff1l2,
* coeffa2l2, coeffb2l2, coefff2l2
common /coeff13/ coeffa1l3, coeffb1l3, coefff1l3,
* coeffa2l3, coeffb2l3, coefff2l3

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

Set zero coefficient for link 1.

coeffa1l1 = 0.0
coeffb1l1 = 0.0
coefff1l1 = 0.0
coeffa2l1 = 0.0
coeffb2l1 = 0.0
coefff2l1 = 0.0

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

Set zero coefficient for link 2.

coeffa1l2 = 0.0
coeffb1l2 = 0.0
coefff1l2 = 0.0
coeffa2l2 = 0.0
coeffb2l2 = 0.0
coefff2l2 = 0.0

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

Set zero coefficient for link 3.

coeffa1l3 = 0.0
coeffb1l3 = 0.0
coefff1l3 = 0.0
coeffa2l3 = 0.0
coeffb2l3 = 0.0
coefff2l3 = 0.0

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

Mission complete, see you...

return
end

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

subroutine coefftot ( neq, yprime )

Program name: Coefftot.f

This subroutine helps to add the coefficients of all links together and store the values into an array.

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

Declare variables and include the common blocks.

******************************************************************************
implicit real (a-z)

integer neq, integer ipath, lda, n
parameter ( ipath = 1, lda = 2, n = 2 )

dimension yprime (neq)
dimension a (n,n), b (n), x (n)

common /link2qs/ qv2, qv2sq, qv2d, qv2dsq, qv2dd
common /link3qs/ qv3, qv3sq, qv3d, qv3dsq, qv3dd
common /coeff11/ coeffa1l1, coeffb1l1, coefff1l1,
* coeffa2l1, coeffb2l1, coefff2l1
common /coeff12/ coeffa1l2, coeffb1l2, coefff1l2,
* coeffa2l2, coeffb2l2, coefff2l2
common /coeff13/ coeffa1l3, coeffb1l3, coefff1l3,
* coeffa2l3, coeffb2l3, coefff2l3

*** Add the coefficients together.

a(1,1) = coeffa1l3+coeffa1l2+coeffa1l1
a(1,2) = coeffb1l3+coeffb1l2+coeffb1l1
a(2,1) = coeffa2l3+coeffa2l2+coeffa2l1
a(2,2) = coeffb2l3+coeffb2l2+coeffb2l1
b(1) = -coefff1l3-coefff1l2-coefff1l1
b(2) = -coefff2l3-coefff2l2-coefff2l1

*** Call IMSL subroutine to solve the matrix.

call lsarg ( n, a, lda, b, ipath, x )

yprime (2) = x (1)
yprime (4) = x (2)
qv2dd = yprime (2)
qv3dd = yprime (4)

*** Time to go, bye...

return
end

subroutine det3x3 ( matrix, determinate )

Program name : Det3x3.f

This is the subroutine that helps to calculate the determinate of a 3x3 matrix.

Declare variable.

implicit real (a-z)
dimension matrix (3,3)

perform calculation.
determinate = ( matrix \( (1,1) \) * matrix (2,2) * matrix (3,3) ) + 
* ( matrix (1,2) * matrix (2,3) * matrix (3,1) ) + 
* ( matrix (1,3) * matrix (2,1) * matrix (3,2) ) - 
* ( matrix (1,1) * matrix (2,3) * matrix (3,2) ) - 
* ( matrix (1,2) * matrix (2,1) * matrix (3,3) ) - 
* ( matrix (1,3) * matrix (2,2) * matrix (3,1) )

That's it, later...

return

end

subroutine det4x4 ( betamatrix, determinate )

Program name : Det4x4.f

This is the subroutine that helps to calculate the determinate of a 4x4 matrix.

Declare variables.

implicit real (a-z)

integer i, k, l, m, sign

dimension betamatrix (4,4), matrix (3,3)

Perform calculations.

determinate = 0.0

do 10 i = 1, 4

  if ( i .eq. 1 ) then
    k = 2
    l = 3
    m = 4
  else if ( i .eq. 2 ) then
    k = 1
    l = 3
    m = 4
  else if ( i .eq. 3 ) then
    k = 1
    l = 2
    m = 4
  else if ( i .eq. 4 ) then
    k = 1
    l = 2
    m = 3
  end if

  matrix (1,1) = betamatrix (2,k)
  matrix (1,2) = betamatrix (2,l)
  matrix (1,3) = betamatrix (2,m)
  matrix (2,1) = betamatrix (3,k)
  matrix (2,2) = betamatrix (3,l)
  matrix (2,3) = betamatrix (3,m)
  matrix (3,1) = betamatrix (4,k)
  matrix (3,2) = betamatrix (4,l)
  matrix (3,3) = betamatrix (4,m)

10 continue
call det3x3 ( matrix, determinate3x3 )

sign = (-1) ** (1 + i)
determinate = determinate +
        ( sign * betamatrix (1,i) * determinate3x3 )

10 continue

continue

subroutine error

Program name : Error.f

This subroutine helps to calculate the position of the
end-effector and the amount of errors.

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

Declare variables and include the common blocks.

implicit real (a-z)

integer numsegment

parameter ( numsegment = 100 )
dimension u2 (0:numsegment), u3 (0:numsegment)

common /link1/   d1, rho1, mass1, j1
common /link2/   a2r, a2rsq, a2rcu, rho2r, mass2r,
*   a2e, rho2e, mass2e, sm2, lm2, eiv2,
*   u2, a2, a2sq, costh2, cosh2sq, sinh2
common /link3/   u3, a3, rho3, mass3, sm3, lm3, eiv3,
*   psid, cospsi, cospsisq, sinpsi, sinsqpsi,
*   chi, chid, chidd, coschi, sinchi, th3ddth2dd
common /theta1/  th1i, th1f, th1, th1d, th1dsq, th1dd
common /theta2/  th2i, th2f, th2, th2d, th2dsq, th2dd
common /theta3/  th3i, th3f, th3, th3d, th3dsq, th3dd
common /v2/     v2a2, v2a2p, v3a3
common /errors/  errorx, errory, errorz
common /trajectory/  rigidx, rigidy, rigidz,
*   elasticx, elasticy, elasticz

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

Calculate the position of the end-effector and the errors.

elasticx = cos(th1)*(cos(th2)*(-sin(v2a2p+th3)*v3a3+a3*cos(v2a2p+t
1  h3)*a2)+sin(th2)*(-cos(v2a2p+th3)*v3a3+a3*sin(v2a2p+th3)+v2a2))
elasticy = sin(th1)*(cos(th2)*(-sin(v2a2p+th3)*v3a3+a3*cos(v2a2p+t
1  h3)*a2)+sin(th2)*(-cos(v2a2p+th3)*v3a3+a3*sin(v2a2p+th3)+v2a2))
estilzc = sin(th2)*(-sin(v2a2p+th3)*v3a3+a3*cos(v2a2p+th3)+v2a2)+c0
rigidx = cos(th1)*cos(th2)*(a5*cos(th3)+a2)+a3*sin(th3)
rigidz = sin(th1)*cos(th2)*(a3*cos(th3)+a2)-a3*sin(th2)*sin(th3)
errorx = elasticx - rigidx
errory = elasticy - rigidy
errorz = elasticz - rigidz

*****************************************************************************
c Time to go, later...
return
end

subroutine fcn ( neq, x, y, yprime )
c Program name : Fcn.f

This is a subroutine that helps to set up the condition of theta 2
and then calculate the corresponding q and q.
Generates the equations used by IMSL (ivpag).
Called by 'ivpag' in 'solver'.
Calls 'calcoeff'.
******************************************************************************

Declare variables and include the common blocks.
implicit real (a-z)
integer neq
dimension y (neq), yprime (neq)
******************************************************************************

Call subroutine to set variables.
call setvar ( neq, x, y, yprime )
******************************************************************************

Call subroutine to find out the values of the coefficients.
call calcoeff ( neq, yprime )
******************************************************************************

Time to go, later...
return
end

subroutine fcnj ( neq, x, y, dyddy )
c Program name : Fcnj.f

This is a dummy subroutine that do not do anything.
******************************************************************************

Initialize variables.
implicit real (a-z)
integer neq
dimension y (neq), dyddy (neq, neq), x
******************************************************************************
Later...

return
end

subroutine init

Program name: Init.f

Subroutine to initialize variables and get value from the user.

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

Declare variables and include the common blocks.

implicit real (a-z)

integer numstep, numsegment

parameter (numsegment = 100)

character * 3 whatlink
character * 6 profiletype

dimension u2 (0:numsegment), u3 (0:numsegment)

dimension polyd1 (5), polye1 (5), polyd2 (5), polye2 (5),
* polyd3 (5), polye3 (5)

common /constant/ g, pi, segment
common /general/ density1, massmo1, length1,
* idiameter1, odiame1r1, csarea1,
* density2r, length2r, iwidth2r, iheight2r,
* owidth2r, oheight2r, csarea2r,
* density2e, elasticity2, areamoiv2, length2e,
* iwidth2e, iheight2e, owidth2e, oheight2e,
* csarea2e, lumpedmass2,
* density3, elasticity3, areamoiv3, length3,
* iwidth3, iheight3, owidth3, oheight3, csarea3,
* lumpedmass3,
* theta1i, theta1f, theta2i, theta2f,
* theta3i, theta3f

common /link1/ d1, rho1, mass1, j1
common /link2/ a2r, a2rsq, a2rcu, rho2r, mass2r,
* a2e, rho2e, mass2e, sm2, ln2, ei2,
* u2, a2, a2sq, costh2, costh2sq, sinh2
common /link3/ u3, a3, rho3, mass3, sm3, ln3, ei3,
* psid, cospsi, cospisq, sinpsi, sinpsisq,
* chi, chid, chidd, cschi, sinh1, th3dth2dd
common /theta1/ th1i, th1f, th1, th1d, th1dsq, th1dd
common /theta2/ th2i, th2f, th2, th2d, th2dsq, th2dd
common /theta3/ th3i, th3f, th3, th3d, th3dsq, th3dd
common /profile/ profiletype, totalt, motion, numstep, interval,
* bang1, bang2, bang3,
* polyd1, polye1, polyd2, polye2, polyd3, polye3
common /insl/ hinit, maxstep, toleranc

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

Read data from the input file.

open (unit = 10, file = 'input.fil', status = 'old')

read (10,*) density1
read (10,*) length1, idiameter1, odiame1r1
read (10,*) density2r
read (10,*) length2r, iwidth2r, iheight2r, owidth2r, oheight2r
read (10,*) density2e, elasticity2, lumpedmass2
read (10,*) length2e, iwidth2e, iheight2e, owidth2e, oheight2e
read (10,*) density3, elasticity3, lumpedmass3
read (10,*) length3, iwidth3, iheight3, owidth3, oheight3
read (10,*) theta1i, theta2i, theta3i
read (10,*) theta1f, theta2f, theta3f
read (10,*) proffiletype
read (10,*) totalt, motiont, numstep
read (10,*) maxstep, toleranc

close (unit = 10)

c

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

c Set variables for later use.

g   = 9.80665
pi   = 3.141592654
radian = pi / 180.0
segment = 100.0

c

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

c Set variables for link 3.

a3 = length3
csarea3 = ( owidth3 * oheight3 ) - ( iwidth3 * iheight3 )
areamoiv3 = ( ( owidth3 * ( oheight3 ** 3.0 ) ) - *  ( iwidth3 * ( iheight3 ** 3.0 ) ) ) / 12.0
rho3 = density3 * csarea3
mass3 = rho3 * a3
sm3 = lumpedmass3
lm3 = lumpedmass3
eiv3 = elasticity3 * areamoiv3

c

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

c Set variables for the rigid portion of link 2.

a2r = length2r
a2rsq = a2r ** 2.0
a2rcu = a2r ** 3.0
csarea2r = ( owidth2r * oheight2r ) - ( iwidth2r * iheight2r )
rho2r = density2r * csarea2r
mass2r = rho2r * a2r

c

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

c Set variables for the elastic portion of link 2.

a2  = length2r + length2e
a2sq = a2 ** 2.0
a2e = length2e
csarea2e = ( owidth2e * oheight2e ) - *
( iwidth2e * iheight2e )
areamoiv2 = ( ( owidth2e * ( oheight2e ** 3.0 ) ) - *
( iwidth2e * ( iheight2e ** 3.0 ) ) ) / 12.0
rho2e = density2e * csarea2e
mass2e = rho2e * a2e
sm2 = lumpedmass2 + mass3 + lumpedmass3
lm2 = lumpedmass2
eiv2 = elasticity2 * areamoiv2

c

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

c Set variables for link 1.

d1 = length1
csarea1 = pi * ( odiameyer1 ** 2.0 - idiameyer1 ** 2.0 ) / 4.0
rho1 = density1 * csarea1
mass1 = rho1 * d1
massmo1 = mass1 * ( odiameyer1 ** 2.0 - idiameyer1 ** 2.0 ) / 8.0
\[ j_1 = \text{massmoi1} \]

Set variables for the initial and final displacement parameters for the links.

\[ \text{th1i} = \text{theta1i} \times \text{radian} \]
\[ \text{th1f} = \text{theta1f} \times \text{radian} \]
\[ \text{th2i} = \text{theta2i} \times \text{radian} \]
\[ \text{th2f} = \text{theta2f} \times \text{radian} \]
\[ \text{th3i} = \text{theta3i} \times \text{radian} \]
\[ \text{th3f} = \text{theta3f} \times \text{radian} \]

Find out the acceleration profile and determine the necessary values for the profile.

Call profiles

Set up some of the IMSL parameters.

\[ \text{interval} = \frac{\text{total} \times \text{numstep}}{\text{float} \times \text{interval}} \]
\[ \text{hinit} = 0.01 \times \text{interval} / 2.0 \]

Call subroutine to estimate the values for the admissible function gamma for link 2.

\[ \text{whatlink} = ' v2' \]

Call admisfn (whatlink)

Call subroutine to estimate the values for the admissible function gamma for link 3.

\[ \text{whatlink} = ' v3' \]

Call admisfn (whatlink)

Call subroutines to initialize all coefficients and torques to zero.

Call coeff0

Call torque0

That's it, later...

return

end

subroutine initdefl ( neq, y )

Program name : Initdefl.f

Subroutine that helps to find out the initial deflections of link 2.
Declare variables and include the common blocks.

implicit real (a-z)

integer neq, numsegment

parameter (numsegment = 100)

dimension y(neq)
dimension u2(0:numsegment), u3(0:numsegment)
dimension gv2(0:numsegment), gv2sq(0:numsegment),
  * gv2p(0:numsegment), gv2psq(0:numsegment),
  * gv2pp(0:numsegment), gv2ppsq(0:numsegment)
dimension gv3(0:numsegment), gv3sq(0:numsegment),
  * gv3p(0:numsegment), gv3psq(0:numsegment),
  * gv3pp(0:numsegment), gv3ppsq(0:numsegment)

common /constant/ g, pi, segment
common /link2/ a2r, a2rsq, a2rcu, rho2r, mass2r,
  * a2e, rho2e, mass2e, sm2, ln2, eiv2,
  * u2, a2, a2sq, costh2, costh2sq, sinh2
common /link3/ u3, a3, rho3, mass3, sm3, ln3, eiv3,
  * psid, cospsid, cospsidq, sinpsid, sinpsidq,
  * ch, chid, chidd, cosch, sinch, th3ddth2dd
common /link2gs/ gv2, gv2sq, gv2p, gv2psq, gv2pp, gv2ppsq,
  * gv2a2, gv2a2sq, gv2a2p, gv2a2psq
common /link3gs/ gv3, gv3q, gv3p, gv3psq, gv3pp, gv3ppsq,
  * gv3a3, gv3a3sq
common /thetare2/ th2i, th2f, th2, th2d, th2dsq, th2dd
common /thetare3/ th3i, th3f, th3, th3d, th3dsq, th3dd

The initial deflection for link 2.

\[ v2a2 = \frac{(sm2 \times g \times a2e^3)}{(3.0 \times eiv2)} + \frac{(rho2e \times g \times a2e^4)}{(8.0 \times eiv2)} \]

\[ v2a2 = -v2a2 \times \cos(\theta2i) \]

\[ y(1) = \frac{v2a2}{gv2a2} \]

\[ v2a2p = y(1) \times gv2a2p \]

The initial deflection for link 3.

\[ v3a3 = \frac{(sm3 \times g \times a3^3)}{(3.0 \times eiv3)} + \frac{(rho3 \times g \times a3^4)}{(8.0 \times eiv3)} \]

\[ v3a3 = -v3a3 \times \cos(\theta3i + v2a2p + th3i) \]

\[ y(3) = \frac{v3a3}{gv3a3} \]

That's it, later...

return

end

subroutine integrat (element, maxarray, numsegment, 
  * upperlimit, lowerlimit, integral)

Program name: integrat.f
This is a subroutine that make use of the 'Trapezoid rule' to calculate integration.

Declare variables.

implicit real (a-z)
integer i, j, maxarray, numsegment
dimension element (maxarray,0:numsegment), integral (maxarray)
common /constant/ g, pi, segment

Perform calculations.

intervalwidth = (lowerlimit - upperlimit) / segment
do 10 i = 1, maxarray
   integral (i) = 0.0
   do 20 j = 1, numsegment
      integral (i) = integral (i) + element (i, j-1) +
         element (i, j)
   20 continue
   integral (i) = integral (i) * intervalwidth / 2.0
10 continue

That's it, later...
return
end

subroutine outputs (x)

Program name : Outputs.f

This is the subroutine that helps to print the outputs into files.

Declare variables and include the common blocks.

implicit real (a-z)
dimension torque (3)

common /theta1/ th1i, th1f, th1, th1d, th1dsq, th1dd
common /theta2/ th2i, th2f, th2, th2d, th2dsq, th2dd
common /theta3/ th3i, th3f, th3, th3d, th3dsq, th3dd
common /torques/ torque
common /vs/ v2a2, v2a2p, v3a3
common /errors/ errorx, errory, errorz
common /trajectory/ rigidx, rigidy, rigidz,
   elastix, elasticy, elasticz
c Set up the format statement.
10 format ( f6.4, 2 ( a1, f8.5 ) )
20 format ( f6.4, 3 ( a1, f8.4 ) )
30 format ( f6.4, 3 ( a1, f8.5, a1, f8.5, a1, f8.3 ) )
40 format ( f6.4, 3 ( a1, f8.6 ) )
50 format ( f6.4, 2 ( a1, f8.5, a1, f8.5, a1, f8.5 ) )
c *****************************************************
c Find out the deflections and print them onto the output file.
write (10,10) x, ', v2a2, ', v3a3

c *********************************************************
c Print the torques onto the output file.
write (11,20) x, ' ', torque(1) / 1000.0,
* ' ', torque(2) / 1000.0, ' ', torque(3) / 1000.0

c *********************************************************
c Print the displacement, velocity and acceleration profiles.
write (12,30) x, ' ', th1, ' ', th1d,
* ' ', th2, ' ', th2d,
* ' ', th3, ' ', th3d

c *********************************************************
c Print the errors.
write (13,40) x, ' ', errorx, ' ', errory, ' ', errorz

c *********************************************************
c Print the trajectories.
write (14,50) x, ' ', rigidx, ' ', rigidy, ' ', rigidz,
* ' ', elasticx, ' ', elasticy, ' ', elasticz

c ************************************************************
c Mission accomplished, see you...
return
end

subroutine profiles

c Program name: Profiles.f

c This subroutine helps to determine the values of the acceleration profile.

c *****************************************************
c Declare variables and include the common blocks.
implicit real (a-z)
integer i, numstep
integer ipath, lda, n
parameter ( ipath = 1, lda = 5, n = 5 )
character * 4 profiletype
Find out what kind of acceleration profile.

if (profiletype .eq. 'bang') then

bang1 = (th1f - th1i) / ((motiont / 2.0)**2.0)
bang2 = (th2f - th2i) / ((motiont / 2.0)**2.0)
bang3 = (th3f - th3i) / ((motiont / 2.0)**2.0)
else

delta1 = (th1f - th1i) / 2.0
delta2 = (th2f - th2i) / 2.0
delta3 = (th3f - th3i) / 2.0

tau1 = motiont / 2.0
tau1sq = tau1**2.0
tau1cu = tau1**3.0
tau1fo = tau1**4.0
tau1fi = tau1**5.0
tau1se = tau1**7.0
tau1ei = tau1**8.0

tau2 = motiont / 4.0
tau2sq = tau2**2.0
tau2cu = tau2**3.0
tau2fo = tau2**4.0
tau2fi = tau2**5.0

matrixa (1,1) = tau1fo
matrixa (1,2) = tau1fi
matrixa (1,3) = tau1si
matrixa (1,4) = tau1se
matrixa (1,5) = tau1ei
matrixa (2,1) = 12.0 * tau1sq
matrixa (2,2) = 20.0 * tau1cu
matrixa (2,3) = 30.0 * tau1fo
matrixa (2,4) = 42.0 * tau1fi
matrixa (2,5) = 56.0 * tau1si
matrixa (3,1) = 24.0 * tau2
matrixa (3,2) = 60.0 * tau2sq
matrixa (3,3) = 120.0 * tau2cu
matrixa (3,4) = 210.0 * tau2fo
matrixa (3,5) = 336.0 * tau2fi
matrixa (4,1) = 24.0 * tau1
matrixa (4,2) = 60.0 * tau1sq
matrixa (4,3) = 120.0 * tau1cu
matrixa (4,4) = 210.0 * tau1fo
matrixa (4,5) = 336.0 * tau1fi
matrixa (5,1) = 24.0
matrixa (5,2) = 120.0 * tau1
matrixa (5,3) = 360.0 * tau1sq
matrixa (5,4) = 840.0 * tau1cu
matrixa (5,5) = 1680.0 * tau1fo

matrixb (2) = 0.0
matrixb (3) = 0.0
matrixb (4) = 0.0
matrixb (5) = 0.0

matrixb (1) = delta1
    call lsarg ( n, matrixa, lda, matrixb, ipath, polyd1 )

matrixb (1) = -delta1
    call lsarg ( n, matrixa, lda, matrixb, ipath, polye1 )

matrixb (1) = delta2
    call lsarg ( n, matrixa, lda, matrixb, ipath, polyd2 )

matrixb (1) = -delta2
    call lsarg ( n, matrixa, lda, matrixb, ipath, polye2 )

matrixb (1) = delta3
    call lsarg ( n, matrixa, lda, matrixb, ipath, polyd3 )

matrixb (1) = -delta3
    call lsarg ( n, matrixa, lda, matrixb, ipath, polye3 )
end if

That's it, later...
return
end

subroutine prtparam

Program name: Prtparam.f

This is the subroutine that helps to print the parameters into
and output file.

Declare variables and include the common blocks.

implicit real (a-z)
integer numstep
integer numsegment
character * 4 profiletype

parameter ( numsegment = 100 )

dimension u2 (0:numsegment), u3 (0:numsegment)
dimension polyd1 (5), polye1 (5), polyd2 (5), polye2 (5),
* polyd3 (5), polye3 (5)

common /general/ density1, massmol1, length1,
* idiameter1, odiameter1, csarea1,
* density2r, length2r, iwidth2r, iheight2r,
* owidth2r, oheight2r, csarea2r,
* density2e, elasticity2, areamol2, length2e,
* iwidth2e, iheight2e, owidth2e, oheight2e,
* csarea2e, lumpedmass2,
* density3, elasticity3, areamoiv3, length3,
* iwidth3, iheight3, owidth3, oheight3, csarea3,
* lumpedmass3,
* theta1i, theta1f, theta2i, theta2f,
* theta3i, theta3f
common /link1/
  d1, rho1, mass1, j1
common /link2/
  a2r, a2rsq, a2rcu, rho2r, mass2r,
  a2e, rho2e, mass2e, sm2, lm2, ev2,
  u2, a2, a2sq, costh2, costh2sq, sinh2
common /link3/
  u3, a3, rho3, mass3, sm3, lm3, ev3,
  psid, cospsi, cospissq, sinpsi, sinsisq,
  chi, chid, chidd, coschi, sinchi, th3dth2dd
common /theta1/ th1i, th1f, th1, th1d, th1sq, th1dd
common /theta2/ th2i, th2f, th2, th2d, th2sq, th2dd
common /theta3/ th3i, th3f, th3, th3d, th3sq, th3dd
common /profile/ proftyle, totalt, motiont, numstep, interval,
  bang1, bang2, bang3,
  polyd1, polye1, polyd2, polye2, polyd3, polye3
common /ims1/ hinit, maxstep, toleranc

c **************************************************************
c
Initialize variables.
10 format ( a50 )
20 format ( a50, 5x, e11.5 )
30 format ( a50, 1x, f11.5 )
40 format ( a50, 5x, f7.5, a3, f7.5 )
c
**************************************************************
c
Put the general information into ‘output.fil’ file.

if ( proftyle .eq. ‘bang’ ) then
  open ( unit = 10, file = ‘boutput.fil’ )
else
  open ( unit = 10, file = ‘poutput.fil’ )
end if

write (10,10),       ***** System Parameters *****
write (10,*)
write (10,10) Units are in SI unit : kg, meter and second
write (10,*)
write (10,10) Link 1 Information
write (10,*)
write (10,30) 'Length (m) =
  length1
write (10,30) 'Inside diameter (m) =
  idiameter1
write (10,30) 'Outside diameter (m) =
  odiameter1
write (10,20) 'Cross section area (m**2) =
  csarea1
write (10,30) 'Density (kg/m**3) =
  density1
write (10,30) 'Mass per unit length (kg/m) =
  rho1
write (10,30) 'Mass (kg) =
  mass1
write (10,20) 'Mass moment of inertia (kg*m**2) =
  massmoi1
write (10,*)
write (10,*)
write (10,10) Link 2 Information
write (10,*)
write (10,30) 'Total length of link 2 (m) =
  length2r + length2e
write (10,"*") / / ' For the rigid portion of link 2 
write (10,10) / / ' Length (m) = 
write (10,30) / / *
write (10,10) / / ' Cross sectional dimension 
write (10,40) / / i) Inside width x height (m) = 
* 
write (10,40) / / ii) Outside width x height (m) = 
* 
write (10,20) / / ' Cross section area (m**2) = 
* 
write (10,30) / / ' Density (kg/m**3) = 
* 
write (10,30) / / ' Mass per unit length (kg/m) = 
* 
write (10,30) / / ' Mass (kg) = 
* 
write (10,"*) / / ' For the elastic portion of link 2 
write (10,10) / / ' Length (m) = 
write (10,30) / / *
write (10,10) / / ' Cross sectional dimension 
write (10,40) / / i) Inside width x height (m) = 
* 
write (10,40) / / ii) Outside width x height (m) = 
* 
write (10,20) / / ' Cross section area (m**2) = 
* 
write (10,30) / / ' Density (kg/m**3) = 
* 
write (10,30) / / ' Mass per unit length (kg/m) = 
* 
write (10,30) / / ' Mass (kg) = 
* 
write (10,20) / / ' In-plane area moment of inertia (m**4) = 
* 
write (10,30) / / ' Lump mass at the end of link 2 (kg) = 
* 
write (10,"*) / / ' Link 3 Information 
write (10,10) / / 
write (10,20) / / ' Modulus of elasticity (Pa) = 
* 
write (10,30) / / ' Length (m) = 
write (10,30) / / *
write (10,10) / / ' Cross sectional dimension 
write (10,40) / / i) Inside width x height (m) = 
* 
write (10,40) / / ii) Outside width x height (m) = 
* 
write (10,20) / / ' Cross section area (m**2) = 
* 
write (10,30) / / ' Density (kg/m**3) = 
* 
write (10,30) / / ' Mass per unit length (kg/m) = 
* 
write (10,30) / / ' Mass (kg) = 
* 
write (10,20) / / ' In-plane area moment of inertia (m**4) = 
* 
write (10,30) / / ' Lump mass at the end of link 3 (kg) = 
* 
write (10,"*) / / ' General Information 
write (10,10) / / 
write (10,"*) / / ' Initial position of link 1 (rad) = 
* 
write (10,30) / / ' th1
write (10,30) 'Initial position of link 1 (deg) = ', theta1i
write (10,30) 'Final position of link 1 (rad) = ', th1f
write (10,30) 'Initial position of link 1 (deg) = ', theta1f
write (10,30) 'Initial position of link 2 (rad) = ', th2i
write (10,30) 'Final position of link 2 (rad) = ', th2f
write (10,30) 'Final position of link 2 (deg) = ', theta2f
write (10,30) 'Initial position of link 2 (deg) = ', theta2i
write (10,30) 'Final position of link 3 (rad) = ', th3i
write (10,30) 'Initial position of link 3 (deg) = ', theta3i
write (10,30) 'Final position of link 3 (deg) = ', theta3f
write (10,30) 'Final position of link 3 (rad) = ', th3f
write (10,30) 'Duration of the experiment (s) = ', totaIt
write (10,30) 'Duration of motion (s) = ', motiont
write (10,30) 'Number of steps within the experiment time frame = ', floaT (numstep)
write (10,30) 'Sampling interval (s) = ', interval
write (10,20) 'The initial value of the step size \( H \) for IML = ', hinit
write (10,30) 'The maximum steps for IML = ', floaT (maxstep)
write (10,20) 'The tolerance (TOL) for IML = ', toleranc
c **************************************************************************
c Close the output file.
c close ( unit = 10 )
c **************************************************************************
c That's it, time to go, see you...
c return
c end

c subroutine setvar ( neq, x, y, yprime )
c Program name : Setvar.f
c This subroutine helps to set up values for all links with respect
to the time steps.
c **************************************************************************
c Declare variables and include the common blocks.
implicit real (a-z)
integer neq, numsegment
parameter ( numsegment = 100 )
dimension y (neq), yprime (neq)
Calculate the input angle time history.
call thistory (x)

c
Setup yprimes.
yprime (1) = y (2)
yprime (3) = y (4)

c
Set variable for link 2.
costh2 = cos (th2)
costh2sq = costh2 ** 2.0
sinth2 = sin (th2)

qv2 = y (1)
qv2sq = qv2 ** 2.0
qv2d = y (2)

v2a2 = vqv2 + qv2 * v2a2
v2a2p = vqv2 + qv2 * v2a2p

c
Set variable for link 3.

psid = th3d + qv2 + qv2 * qv2
cospisi = cos (th3 + qv2 + qv2 * qv2)
cospisiq = cospisi ** 2.0
sinpisi = sin (th3 + qv2 + qv2 * qv2)
sinpisiq = sinpisi ** 2.0

chid = th3d + th2d + qv2 + qv2d
chidd = th3dd + th2dd + qv2 + qv2dd

th3ddth2dd = th3dd + th2dd
qv3 = y (3)
qv3sq = qv3 ** 2.0
qv3d = y (4)
qv3dsq = qv3d ** 2.0
qv3dd = yprime (4)

v3a3 = qv3 * qv3a3

c ************************************************************
c
Time to go, later...

return
c
end

subroutine solver
c
Program name : Solver.f
c
This is the subroutine that helps to solve our problem.
c
*************************************************************c
c
Declare variables and include the common blocks.

external fcn, fenj

implicit real (a-z)

integer i, numstep
integer neq, nparam, ido, inorm, meth, miter

character * 4 profile type

parameter ( neq = 4, nparam = 50 )

dimension y (neq), yprime (neq), param (nparam)
dimension polyd1 (5), polye1 (5), polyd2 (5), polye2 (5),
* polyd3 (5), polye3 (5)

common /profile/ profile type, totalt, motiont, numstep, interval,
* bang1, bang2, bang3,
* polyd1, polye1, polyd2, polye2, polyd3, polye3

common /imsl/ hinit, maxstep, toleranc

c *************************************************************c

Set variable.

x = 0.0

c *************************************************************c

Calculate initial conditions for the ys (qs) from the given boundary condition.

y (1) = 0.0
y (2) = 0.0
y (3) = 0.0
y (4) = 0.0
yprime (1) = 0.0
yprime (2) = 0.0
yprime (3) = 0.0
yprime (4) = 0.0

call setvar ( neq, x, y, yprime )
call initald ( neq, y )
**Open the output files.**

```plaintext
if ( filetype .eq. 'bang' ) then
    open ( unit = 10, file = 'bdeflections.fil' )
    open ( unit = 11, file = 'btorques.fil' )
    open ( unit = 12, file = 'bprofile.fil' )
    open ( unit = 13, file = 'berrors.fil' )
    open ( unit = 14, file = 'btrajectory.fil' )
else
    open ( unit = 10, file = 'pdeflections.fil' )
    open ( unit = 11, file = 'ptorques.fil' )
    open ( unit = 12, file = 'pprofile.fil' )
    open ( unit = 13, file = 'perrors.fil' )
    open ( unit = 14, file = 'ptrajectory.fil' )
end if
```

**Call subroutine 'fe n' to set up the variables for the coefficients and torques, call 'caltorq' to calculate the result at time 0 and call 'error' to calculate the position of the end-effector at time zero.**

```plaintext
call fcn ( neq, x, y, yprime )
call caltorq ( x )
call error
call outputs ( x )
```

**Set values for the IMSL subroutine.**

```plaintext
ido = 1
inorm = 1
meth = 2
miter = 2
tol = tolerance
```

**Call the IMSL subroutines. Then call 'caltorq' to calculate the corresponding torque for the three joins and call 'error' to calculate the position of the end-effector.**

```plaintext
call sset ( nparam, 0.0, param, 1 )
param (1) = hinit
param (4) = maxstep
param (10) = inorm
param (12) = meth
param (13) = miter
do 20 i = 1, numstep
    xend = float (i) * interval
    call ivpag ( ido, neq, fcn, fnj, a, x, xend, tol, param, y )
call fcn ( neq, x, y, yprime )
call caltorq ( x )
call error
call outputs ( x )
20 continue
ido = 3
```
call ivpag (ido, neq, fen, fenj, a, x, xend, tol, param, y)

Close the output files.

close (unit = 10)
close (unit = 11)
close (unit = 12)
close (unit = 13)
close (unit = 14)

Mission accomplished, see you...

return
end

subroutine history (x)

Program name: History.f

This subroutine helps to set up the time history of the acceleration profile, velocity profile and the displacement profile.

Declare variables and include the common blocks.

implicit real (a-z)

integer numstep
integer i, j

character * 4 proffilename

dimension polydl (5), polye1 (5), polyd2 (5), polye2 (5),
* polyd3 (5), polye3 (5)

common /theta1/ th1, th1f, th1, th1d, th1dsq, th1dd
common /theta2/ th2, th2f, th2, th2d, th2dsq, th2dd
common /theta3/ th3, th3f, th3, th3d, th3dsq, th3dd

common /profile/ proffilename, totaltime, motiontime, numstep, interval,
* bang1, bang2, bang3,
* polydl, polye1, polyd2, polye2, polyd3, polye3

Calculate the input angle time history.

if (proffilename .eq. "bang") then

if (x .lt. 0.5 * motiontime) then

th1 = th1 + (0.5 * bang1 * (x ** 2.0))
th1d = bang1 * x
th1dsq = th1d ** 2.0
th1dd = bang1
th2 = th2 + (0.5 * bang2 * (x ** 2.0))
th2d = bang2 * x
th2dsq = th2d ** 2.0
th2dd = bang2
th3 = th3 + (0.5 * bang3 * (x ** 2.0))
th3d = bang3 * x
th3dsq = th3d ** 2.0
th3dd = bang3

else if ( ( x .ge. 0.5 * motiont ) .and. 
* ( x .lt. motiont ) ) then

th1 = th1f - ( 0.5 * bang1 * ( ( motiont-x ) ** 2.0 ) )
th1d = bang1 * ( motiont - x )

th1dsq = th1d ** 2.0

th1dd = -bang1

th2 = th2f - ( 0.5 * bang2 * ( ( motiont-x ) ** 2.0 ) )
th2d = bang2 * ( motiont - x )

th2dsq = th2d ** 2.0

th2dd = -bang2

th3 = th3f - ( 0.5 * bang3 * ( ( motiont-x ) ** 2.0 ) )
th3d = bang3 * ( motiont - x )

th3dsq = th3d ** 2.0

th3dd = -bang3

else

th1 = th1f

th1d = 0.0

th1dsq = 0.0

th1dd = 0.0

th2 = th2f

th2d = 0.0

th2dsq = 0.0

th2dd = 0.0

th3 = th3f

th3d = 0.0

th3dsq = 0.0

th3dd = 0.0

end if

else

if ( x .lt. 0.5 * motiont ) then

th1 = th1i

th1d = 0.0

th1dd = 0.0

th2 = th2i

th2d = 0.0

th3 = th3i

th3d = 0.0


do 10 i = 4, 8

j = i - 3

th1 = th1 + polyd1 (j + 1) * ( x ** float (i) )

th1d = th1d + float (i) * polyd1 (j) *

* ( x ** float (i-1) )

th1dd = th1dd + float (i) * float (i-1) * polyd1 (j) *

* ( x ** float (i-2) )

th2 = th2 + polyd2 (j) * ( x ** float (i) )

th2d = th2d + float (i) * polyd2 (j) *

* ( x ** float (i-1) )

th2dd = th2dd + float (i) * float (i-1) * polyd2 (j) *

* ( x ** float (i-2) )

th3 = th3 + polyd3 (j) * ( x ** float (i) )

th3d = th3d + float (i) * polyd3 (j) *

* ( x ** float (i-1) )

th3dd = th3dd + float (i) * float (i-1) * polyd3 (j) *

* ( x ** float (i-2) )

continue

10 continue
th1dsq = th1d ** 2.0
th2dsq = th2d ** 2.0
th3dsq = th3d ** 2.0

else if ((x .ge. 0.5 * motiont) .and. (x .lt. motiont)) then

  th1 = th1f
  th1d = 0.0
  th1dd = 0.0
  th2 = th2f
  th2d = 0.0
  th3 = th3f
  th3d = 0.0
  th3dd = 0.0

  do 20 i = 4, 8

    j = i - 3

    th1 = th1 + polye1 (j) * ((motiont - x) ** float (i))
    th1d = th1d + float (i) * polye1 (j) * ((motiont - x) ** float (i-1))
    th1dd = th1dd + float (i) * float (i-1) * polye1 (j) * ((motiont - x) ** float (i-2))
    th2 = th2 + polye2 (j) * ((motiont - x) ** float (i))
    th2d = th2d + float (i) * polye2 (j) * ((motiont - x) ** float (i-1))
    th2dd = th2dd + float (i) * float (i-1) * polye2 (j) *

  continue

  th1d = - th1d
  th1dsq = th1d ** 2.0
  th2d = - th2d
  th2dsq = th2d ** 2.0
  th3d = - th3d
  th3dsq = th3d ** 2.0

else

  th1 = th1f
  th1d = 0.0
  th1dsq = 0.0
  th1dd = 0.0
  th2 = th2f
  th2d = 0.0
  th2dsq = 0.0
  th2dd = 0.0
  th3 = th3f
  th3d = 0.0
  th3dsq = 0.0
  th3dd = 0.0

end if

end if

c

* Time to go, later...*
subroutine tlink1

Program name: Tlink1.f

This subroutine helps to setup and calculate the torques for link 1.

Declare variables and include the common blocks.

implicit real (a-z)

common /link1/  d1, rho1, mass1, j1
common /theta1/  th1i, th1f, th1, th1d, th1dsq, th1dd
common /torque1/ torque1l1, torque2l1, torque3l1

Set variables.

torque1l1 = j1*th1dd

Mission complete, see you...

return
end

subroutine tlink2

Program name: Tlink2.f

This subroutine helps to setup and calculate the torques for link 2.

Declare variables and include the common blocks.

implicit real (a-z)

integer i, maxarray, numsegment

parameter ( maxarray = 4, numsegment = 100 )

dimension u2 (0:numsegment)
dimension gv2 (0:numsegment), gv2sq (0:numsegment),
*  gv2p (0:numsegment), gv2psq (0:numsegment),
*  gv2pp (0:numsegment), gv2ppsq (0:numsegment)
dimension element (maxarray,0:numsegment), integral (maxarray)

common /constant/  g, pi, segment
common /link2/  a2r, a2rsq, a2rcu, rho2r, mass2r,
*  a2e, rho2e, mass2e, sm2, lm2, ei2,
*  u2, a2, a2sq, costh2, costh2sq, sinh2
common /link2gs/  gv2, gv2sq, gv2p, gv2psq, gv2pp, gv2ppsq,
*  gv2a2, gv2a2a, gv2a2p, gv2a2psq
common /link2qs/  qv2, qv2sq, qv2d, qv2dsq, qv2dd
Calculate the values of the integrals of link 2.

do 10 i = 0, numsegment

element(1,i) = (gv2(i)*qv2*sinth2-costh2*u2(i))*(u2(i)*sinth1-th1d)
  *th2d+cosh2*gv2(i)*qv2*th1d-th2d+gv2(i)*qv2*sinth2*th1dd+cosh2
  2*u2(i)*th1d+gv2(i)*qv2d*sinth2+th1d+gv2(i)*qv2*sinth2*th1d-th2d
  3+cosh2*th1d)*u2(i)*sinth2-th2d+cosh2*gv2(i)*qv2*th2d-gv2(i)*qv2d*sinth2

element(2,i) = 2*u2(i)*th2dd+gv2(i)*qv2dd+2*gv2sq(i)*qv2sq
  *th2dd+4*gv2sq(i)*qv2d+th2d

element(3,i) = (u2(i)*sinth2*th1d+costh2*gv2(i)*qv2*th1d)*(gv2(i)*
  1 qv2*sinth2*th1d-costh2*u2(i)*th1d)

element(4,i) = costh2*u2(i)-gv2(i)*qv2*sinth2

continue

call integrat ( element, maxarray, numsegment, a2r, a2, integral )

Set variables.

e13 = integral(1)*rho2e

e14 = (gv2a2*gv2*sinth2-a2*cosh2)*sm2*(a2*sinth2*th1d-th2d+cosh2
  1*gv2a2*gv2*sinth2*th1d-th2d-gv2a2*gv2*sinth2*th1dd-a2*cosh2*th1dd-gv2
  2+a2*sinth2*th1d)

e15 = sm2*(gv2a2*gv2*sinth2*th1d-a2*cosh2*th1d)*(a2*sinth2*th2d+c
  1*costh2*gv2a2*gv2d*th2d-gv2a2*gv2*d*sinth2)

e16 = (-2.0)*a2rcu*cosh2*rho2r*sinth2*th1d-th2d/3.0

e17 = a2rcu*cosh2sq*rho2r*th1d/3.0

e18 = 0.5*integral(2)*rho2e

e19 = -integral(3)*rho2e

e20 = integral(4)*g*rho2e

e21 = sm2*(2*a2*(a2*th2dd+gv2a2*gv2dd)+2*gv2a2sq*gv2sq*th2dd+4*gv2
  1 a5sq*gv2gv2d+th2d)/2.0

e22 = a2rcu*rho2r*th2dd/3.0

e23 = a2rcu*cosh2*rho2r*sm2*th1d+cosh2

e24 = sm2*(a2*sinth2*th1d+costh2*gv2a2*gv2d*(gv2a2*gv2d)*
  1 th1d-a2*cosh2*th1d-g*(a2*cosh2-gv2a2*gv2*sinth2))

e25 = a2rsq*cosh2g*rho2r/2.0

c Calculate the torques with the variables just calculated.

torque1l2 = e17+e16+e15+e14+e13

torque2l2 = e25+e24+e23+e22+e21+e20+e19+e18

c Mission complete, see you...

return

end

subroutine tlink3

c Program name : Tlink3.f

c This subroutine helps to setup and calculate the torques for
2 link 3.
Declare variables and include the common blocks.

implicit real (a-z)

integer i, maxarray, numsegment

parameter ( maxarray = 7, numsegment = 100 )

dimension u2 (0:numsegment), u3 (0:numsegment)

dimension gv2 (0:numsegment), gv2sq (0:numsegment),
  * gv2p (0:numsegment), gv2psq (0:numsegment),

dimension gv3 (0:numsegment), gv3sq (0:numsegment),
  * gv3p (0:numsegment), gv3psq (0:numsegment),
  gv3pp (0:numsegment), gv3ppsq (0:numsegment)

dimension element (maxarray,0:numsegment), integral (maxarray)

common /constant/ g, pi, segment

common /link2/ a2r, a2sq, a2rcu, rho2r, mass2r,
  * a2e, rho2e, mass2e, sm2, lm2, eiv2,
  * u2, a2, a2sq, costh2, costh2sq, sinh2

common /link3/ u3, a3, rho3, mass3, sm3, lm3, eiv3,
  * psid, cosp3, cosp3sq, sinpsi, sinpsiq,
  * chi, chid, chidd, coschi, sinhchi, th3ddth2dd

common /link3qs/ qv2, qv2sq, qv2d, qv2dsq, qv2dd

common /link3gs/ gv2, gv2sq, gv2p, gv2psq, gv2pp, gv2ppsq,
  * gv2a2, gv2asq, gv2ap, gv2apsq

common /link3qs/ qv3, qv3sq, qv3d, qv3dsq, qv3dd

common /link3gs/ gv3, gv3sq, gv3p, gv3psq, gv3pp, gv3ppsq,
  * gv3a3, gv3asq

common /thetap3/ th1i, th1f, th1, th1d, th1dsq, th1dd

common /thetap2/ th2i, th2f, th2, th2d, th2dsq, th2dd

common /thetap3/ th3i, th3f, th3, th3d, th3dsq, th3dd

common /torque/ torque1l3, torque2l3, torque3l3

c Calculate the values of the integrals of link 3.

do 10 i = 0, numsegment

link3sub(1) = u3(i)*sinpsi+cosp3*gv3(i)*qv3

link3sub(2) = cosp3*u3(i)-gv3(i)*qv3*psid

link3sub(3) = gv3(i)*qv3*psid*gv2a2+gv2asq+gv2ap

link3sub(4) = -gv3(i)*qv3*psid*gv2a2+gv2asq+gv2ap

link3sub(5) = cosp3*gv3(i)*psid+gv2a2+gv2asq+gv2ap

link3sub(6) = -gv3(i)*psid*gv2a2+gv2asq+gv2ap+gv2ap

link3sub(7) = gv2a2+gv2asq+gv2ap+gv2ap

* 2*(u3(i)*sinpsi+cosp3*gv3(i)*qv3)

10 continue
\[
\begin{align*}
8 & \quad 2^*\text{chid} \\
\text{element}(3, j) &= (a^2\sin\theta_2\text{th1d}(u^3(i)\sin\chi_i + \cos\chi_i g^3(i)\text{qv3}\text{th}) \\
 & \quad \text{dcos}\text{th2}\text{gv2a2}\text{qv2}\text{th1d}(\text{qgv2a2}\text{qv2}\text{sin}\text{th2}\text{th1d} - a^2\text{cos}\text{th2}\text{th1d}) \\
 & \quad - e^3\text{linksub}(5)^*\text{th1d} \\
\text{element}(4, j) &= -e^3\text{linksub}(7) \\
\text{element}(5, j) &= -2^*\text{linksub}(1)^*(\text{gv2a2}\text{qv2}\text{th2dd}\text{gv2a2}\text{qv2}\text{th2dd} - \text{gv2a2}\text{qv2}) \\
 & \quad 3(i)\text{qv3}\text{d}\text{sin}\psi_i\text{cos}\psi_i g^3(i)\text{psid}\text{qv3}\text{d}\text{linksub}(1)^*\text{chid}\text{d} - \text{lin} \\
 & \quad \text{ksub(4)^*chid(2)^*a^2\text{th2dd}\text{g}^3(1)^*\text{psid}\text{qv3}\text{d}\text{sin}\psi_i + \text{cos}\psi_i g^3(i)^*\text{qv3}\text{dd}\text{gv2a2}\text{qv2}\text{dd}\text{linksub}(2)^*\text{chid}\text{d}\text{linksub}(3)^*c} \\
 & \quad 4(i)\text{th2dd}\text{gv3(1)^*qv3}\text{d}\text{sin}\psi_i\text{linksub}(1)^* - e^3\text{linksub}(1)^*\text{chid}\text{d} \\
 & \quad 5(1)^*\text{chid}\text{d} - e^3\text{linksub}(3)^*\text{a}^2\text{th2dd}\text{cos}\psi_i g^3(i)^*\text{qv3}\text{d}\text{gv2a2}\text{qv2}\text{dd} + \text{linksub(2)^*chid} \\
\text{element}(6, j) &= 2^*(-\text{cos}\psi_i g^3(i)^*\text{qv3}\text{d}\text{linksub}(2)^*\text{chid} d^*\text{gv2a2}\text{qv2}) \\
 & \quad 2^*\text{th2dd}\text{gv3(1)^*qv3}\text{d}\text{sin}\psi_i\text{linksub}(1)^*\text{chid}\text{d} + 2^*(-\text{g}^3(1)^*\text{qv3}\text{sin}\psi_i \\
 & \quad \text{psi}\text{1}\text{linksub}(1)^*\text{chid}\text{d}^*\text{a}^2\text{th2dd}\text{cos}\psi_i g^3(i)^*\text{qv3}\text{d}\text{gv2a2}\text{qv2}\text{dd} + \text{linksub}(2)^*\text{chid} \\
 & \quad \text{ninksub(2)^*chid(2)^*u^3(i)\sin\chi_i + \cos\chi_i g^3(1)^*qv3(1)^*d}) - \text{d}^*\text{g}^2(1)^*\text{qv2}\text{th1d}^*\text{chid}\text{d}^* \text{gv2a2} \\
 & \quad \text{qv2}\text{sin}\text{th2}\text{th1d}^*\text{a}^2\text{c}^2\text{th2}\text{th1d}^*\text{d}^*\text{linksub}(5)^*\text{th1d} \\
\text{element}(7, j) &= \text{linksub}(5) \\
\text{continue} \\
\text{call integrat}(\text{element}, \text{maxarray}, \text{numsegment}, 0, a^3, \text{integral}) \\
\text{c} \\
& \quad \text{Set variables.} \\
\text{c} \\
\text{e38} &= \text{integral}(1)^*\rho^3 \text{e39} &= \text{linksub}(7)^*\text{s}\text{a}^3\text{a}^2\text{sin}\text{th2}\text{th1d}^*\text{th2dd}\text{c}^2\text{th2}\text{th1d}^*\text{g}^2(1)^*\text{qv2}\text{th1d}^*d \\
 & \quad 2^*\text{gv2a2}\text{qv2}\text{sin}\text{th2}\text{th1d}^*\text{d}\text{gv2}\text{qv2}\text{th1d}^*\text{d}\text{linksub}(5)^*\text{th1d}^*\text{gv} \\
 & \quad \text{chi}\text{1}\text{g}^3(1)^*\text{qv3}\text{d}\text{sin}\psi_i + \text{a}^3\text{chid}\text{d}\text{sin}\chi_i + \text{chid}\text{d}\text{cos}\chi_i \\
\text{e40} &= \text{a}^3\text{gv2a2}\text{qv2}\text{sin}\text{th2}\text{th1d}^*\text{d}\text{a}^2\text{c}^2\text{th2}\text{th1d}^*\text{d}\text{linksub}(5)^*\text{th1d}^* \text{e41} &= 0.5^*\text{integral}(2)^*\rho^3 \\
\text{e42} &= -\text{integral}(3)^*\rho^3 \text{e43} &= \text{integral}(4)^*\rho^3 \text{e44} &= \text{a}^3\text{gv3}\text{d}\text{a}^2\text{sin}\phi\text{sinpsi}\text{a}^3\text{qv3}\text{dd}\text{gv2a2}\text{qv2}\text{th2dd} \\
 & \quad -\text{gv2a2}\text{qv2}\text{dd}\text{linksub}(5)^*\text{chid}\text{d}\text{a}^2\text{c}^2\text{th2}\text{th1d}^* - \text{cos}\phi\text{gv3}\text{sin}\phi\text{a}^3\text{qv3}\text{dd}\text{a}^2\text{sin}\phi \\
\text{e45} &= -\text{sn}\text{ga}^3\text{qv3}\text{d}\text{a}^2\text{cos}\phi\text{cos}\phi\text{a}^3\text{qv3}\text{dd}\text{linksub}(3)^*\text{a}^2\text{th2dd} \\
\text{e46} &= 0.5^*\text{integral}(5)^*\rho^3 \text{e47} &= -0.5^*\text{integral}(6)^*\rho^3 \text{e48} &= \text{integral}(7)^*\rho^3 \text{e49} &= \text{sn}\text{a}^3\text{a}^2\text{linksub}(1)^*\text{chid}\text{d}\text{linksub}(4)^*\text{chid}\text{d}^*2^*\text{gv3}\text{d}\text{qv3}\text{dd}\text{sinpsi}\text{a}^3\text{cos} \\
\text{e50} &= -\text{sn}\text{a}^3\text{a}^2\text{linksub}(2)^*\text{chid}\text{d}\text{linksub}(3)^*\text{chid}\text{d}^*2^*\text{gv3}\text{d}\text{qv3}\text{dd}\text{sinpsi}\text{a}^3\text{cos} \\
\text{c} \\
\text{Calculate the torques with the variables just calculated.} \\
\text{torque1l3} &= \text{e40+e39+e38} \\
\text{torque2l3} &= \text{e45+e44+e43+e42+e41} \\
\text{c} \\
\text{***************}
torque3l3 = e50+e49+e48+e47+e46

Mission complete, see you...
return
eend

subroutine torque0

Program name: Torque0.f

This subroutine helps to set the torque of the linkage to zero.

Declare variables and include the common blocks.

implicit real (a-z)

common /torque11/ torque11, torque211, torque311
common /torque12/ torque112, torque212, torque312
common /torque13/ torque113, torque213, torque313

Set zero torque for link 1.

torque11 = 0.0
torque211 = 0.0
torque311 = 0.0

Set zero torque for link 2.

torque112 = 0.0
torque212 = 0.0
torque312 = 0.0

Set zero torque for link 3.

torque113 = 0.0
torque213 = 0.0
torque313 = 0.0

Mission complete, see you...
return
eend

subroutine torquentot

Program name: torquentot.f

This subroutine helps to add the torques of all links together and store the values into an array.

*****************************************************************************
Declare variables and include the common blocks.

implicit real (a-z)

dimension torque (3)

common /torque11/ torque11, torque21, torque311
common /torque12/ torque12, torque212, torque312
common /torque13/ torque13, torque213, torque313
common /torques/ torque

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

c Add the torques together.

torque(1) = torque11+torque12+torque111
 torque(2) = torque213+torque212+torque211
 torque(3) = torque313+torque312+torque311

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

c Time to go, bye...

return
end
APPENDIX I

FORTRAN PROGRAM LISTINGS OF THE UNLV-ARO
LIGHT-WEIGHT ROBOT
subroutine admisssfn ( whatlink )

Program name : Amdissfn.f

This is the subroutine that helps to estimate the values of the
admissible function gammas.

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

Declare variables and include the common blocks.

implicit real (a-z)

integer power, maxpower, sign1, sign2

character * 3 whatlink

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

If the admissible function of 'wh3' is needed, call another
subroutine.

if ( whatlink .eq. 'vh3' .or. whatlink .eq. 'wh3' ) then
  call calgamma ( whatlink, 0.0 )
  return
end if

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

Initialize variables.

power = 1
maxpower = 10
estimate = 1.0 / ( 10.0 ** power )
beta = estimate

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

First, calculate the determinate of the matrix with beta = 0.0

call betamat ( whatlink, beta, determinate1 )

if ( determinate1 .eq. 0.0 ) then
  sign1 = 0
else
  if ( determinate1 .lt. 0.0 ) then
    sign1 = -1
  else
    sign1 = 1
  end if
end if

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

Using numerical approximation method to estimate beta.

10 beta = beta + estimate

call betamat ( whatlink, beta, determinate2 )

if ( determinate2 .eq. 0.0 ) then
  sign2 = 0
else
  if ( determinate2 .lt. 0.0 ) then
    sign2 = -1
  else
    sign2 = 1
  end if
end if

if ( determinate1 .eq. determinate2 ) then
if ( power .eq. maxpower ) then
    call calgamma ( whatlink, beta )
    return
else
    power = power + 1
    beta = beta - estimate
    estimate = 1.0 / ( 10.0 ** power )
goto 10
end if
end if

if ( sign2 .eq. 0 ) then
    call calgamma ( whatlink, beta )
    return
else
    if ( sign1 .ne. sign2 ) then
        call calgamma ( whatlink, beta )
        return
    else
        power = power + 1
        beta = beta - estimate
        estimate = 1.0 / ( 10.0 ** power )
    end if
    else
        determinate1 = determinate2
        sign1 = sign2
    end if
end if

 goto 10

cMISSION ACCOMPLISHED, LATER . . .
c
end

Program name : Aro.f
This is the main driver for the vibration analysis of the UNLV-ARO robot.
c
CALL SUBROUTINE 'INIT' TO GET INPUT FROM AN INPUT FILE AND INITIALIZE VARIABLES.
call init

cCALL SUBROUTINE 'PRTPARAM' TO PRINT OUT THE SYSTEM PARAMETERS.
call prtparam

cCALL SUBROUTINE 'SOLVER' TO BEGIN CALCULATION.
call solver

cTHAT'S IT, LATE...
stop
end
subroutine betamat ( whatlink, beta, determine )

Program name : Betamat.f

This is a subroutine that helps to determine the values for the beta matrix for link 2, link 3 and hydraulic cylinder 3.

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

Declare variables and include the common blocks.

implicit real (a-z)
character * 3 whatlink
integer numsegment
parameter ( numsegment = 100 )
dimension betamatrix (4,4)
dimension u2 (0:numsegment), u3 (0:numsegment), uh3 (0:numsegment)

common /constant/ a2r, a2rsq, a2rcu, rho2r, mass2r,
* a2e, rho2e, mass2e, sm2, lm2, eif2, eif2,
* u2, a2, a2sq, costh2, costh2sq, cos2th2,
* sinh2, sin2th2
common /link2/ a3, a3eg, rho3, mass3, sm3, lm3, m3cg, eif3,
* eif3, psid, psidsq, psidd, cospsi, cospsiq,
* sindsi, sindisq,
* chi, chid, chidd, coschi, sinchi, th3dth2dd,
* omegasub1, omegasub2, omegasub3
common /link3/ ac, acsq, accu, zeta, rho3, mass3,
* ah3s, ah3sq, ah3cu, rho3h, mass3h,
* ah3p, ah3psart, ah3psartsc, ah3psartscu,
* ah3pend, ah3pendsq, ah3pendcu, rho3h3p, mass3h3p,
* ah3r, rho3h3r, mass3h3r, sm3h3, eih3,
* uh3, ah3, ah3sq, ah3cu, ah3fo, ah3fi, ah3se,
* eta, coseta, sineta, thh3, costh3, costh3sq,
* sinh3h3, sinh3h3sq, lamda, coslamda, coslamdasq,
* sinlamda, sinlamdasq
common /theta3/ th3i, th3f, th3l, th3d, th3dsq, th3dd

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

Initialize variables.

betamatrix (1,1) = 0.0
betamatrix (1,2) = 1.0
betamatrix (1,3) = 0.0
betamatrix (1,4) = 1.0
betamatrix (2,1) = 1.0
betamatrix (2,2) = 0.0
betamatrix (2,3) = 1.0
betamatrix (2,4) = 0.0

betasq = beta ** 2.0
betacu = beta ** 3.0
betafo = beta ** 4.0
sine = sin ( beta )
sineh = sinh ( beta )
cosine = cos ( beta )
cosineh = cosh ( beta )

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

Perform calculations.
if ( whatlink .eq. 'v2' ) then

    case = 1
    tau = ( sm2 * betafo ) / ( rho2e * a2e )

else if ( whatlink .eq. 'w2' ) then

    case = 2
    tau1 = - ( m3cg * ( a3cg * cos ( th3 ) ) ** 2.0 * betafo ) / 
          ( rho2e * ( a2e ** 3.0 ) )
    tau2 = - ( m3cg * a3cg * cos ( th3 ) * betafo ) / 
          ( rho2e * ( a2e ** 2.0 ) )
    tau3 = ( sm2 * betafo ) / ( rho2e * a2e )

else if ( whatlink .eq. 'v3' ) then

    case = 1
    tau = ( sm3 * betafo ) / ( rho3 * a3 )

else if ( whatlink .eq. 'w3' ) then

    case = 1
    tau = ( sm3 * betafo ) / ( rho3 * a3 )

end if

c
******************************************************************************

C Set up the matrix.
if ( case .eq. 1 ) then

    betamatrix (3,1) = - sine
    betamatrix (3,2) = - cosine
    betamatrix (3,3) = sinh
    betamatrix (3,4) = cosh
    betamatrix (4,1) = -( betacu * cosine ) + ( tau * sine )
    betamatrix (4,2) = ( betacu * sine ) + ( tau * cosine )
    betamatrix (4,3) = ( betacu * cosh ) + ( tau * sinh )
    betamatrix (4,4) = ( betacu * sinh ) + ( tau * cosh )

else

    betamatrix (3,1) = -( betasq * sine ) + 
                     ( tau1 * beta * cosine ) + ( tau2 * sine )
    betamatrix (3,2) = -( betasq * cosine ) - 
                     ( tau1 * beta * sine ) + ( tau2 * cosine )
    betamatrix (3,3) = ( betasq * sinh ) + 
                     ( tau1 * beta * cosh ) + 
                     ( tau2 * sinh )
    betamatrix (3,4) = ( betasq * cosh ) + 
                     ( tau1 * beta * sinh ) + 
                     ( tau2 * cosh )
    betamatrix (4,1) = -( betacu * cosine ) + ( tau3 * sine )
    betamatrix (4,2) = ( betacu * sine ) + ( tau3 * cosine )
    betamatrix (4,3) = ( betacu * cosh ) + ( tau3 * sinh )
    betamatrix (4,4) = ( betacu * sinh ) + ( tau3 * cosh )

end if

C
******************************************************************************

C Call subroutine to find out the determinate of the matrix.
call det4x4 ( betamatrix, determinate )

C
******************************************************************************

C Mission accomplished, see you...
subroutine calcCoeff ( neq, yprime )
  c Program name : CalcCoeff.f
  c This subroutine helps to calculate the coefficients of the system.
  ********************************************
  c Declare variables.
  implicit real (a-z)
  integer neq
  dimension yprime ( neq )
  ********************************************
  c Calculate the coefficients for link 2.
  call clink2
  ********************************************
  c Calculate the coefficients for link 3.
  call clink3
  ********************************************
  c Calculate the coefficients for link c.
  call clinkc
  ********************************************
  c Calculate the coefficients for link h3.
  call clinkh3
  ********************************************
  c Calculate the coefficients for the whole system.
  call coefftot ( neq, yprime )
  ********************************************
  c Time to go, later...
  return
end

subroutine calforce
  c Program name : Calforce.f
  c This subroutine helps to calculate the piston force of the
  c hydraulic cylinders needed to sustain the motion of the robot.
  ********************************************
Declare variables and include the common blocks.

implicit real (a-z)

integer numsegment

parameter ( numsegment = 100 )

dimension u2 (0:numsegment), uh3 (0:numsegment)
dimension torque (3), force (3)

common /constant/ g, pi, segment
common /link2/ a2r, a2rsq, a2rcu, rho2r, mass2r,
  a2e, rho2e, mass2e, sm2, ln2, e1v2, e1w2,
  u2, e2, a2sq, costh2, cosht2sq, cos2th2,
  sinh2, sinh2sq
common /linkh3/ ah3s, ah3sq, ah3cru, rho3s, mass3s,
  ah3p, ah3pstart, ah3pstartsq, ah3pstartcu,
  ah3pend, ah3pendsq, ah3pendcu, rho3p, mass3p,
  ah3r, rho3r, mass3r, sm3, ei3,
  uh3, ah3, ah3sq, ah3cu, ah3f, ah3s, e3,
  sith3, sith3sq, lamda, coslamda, coslamdasq,
  sinlamda, sinlamdasq
common /linkc/ ac, acsq, accu, zeta, rhoc, massc
common /hydraulic/ lh1, lh2a, lh2b
common /theta2/ th2i, th2f, th2, th2d, th2dsq, th2dd
common /theta3/ th3i, th3f, th3, th3d, th3dsq, th3dd
common /torques/ torque
common /forces/ force

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

c Calculate the force produced by the first hydraulic cylinder.

force(1) = torque(1) / lh1

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

c Calculate the force produced by the second hydraulic cylinder.

ah2 = sqrt ( lh2a ** 2.0 + lh2b ** 2.0 -
  2.0 * lh2a * lh2b * cos ( pi / 2.0 + th2 ) )
num2 = asin ( lh2a * ( sin ( pi / 2.0 + th2 ) / ah2 ) )
force(2) = ( torque(2) / lh2b ) * ( 1.0 / cos ( pi / 2.0 - num2 ) )

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

c Calculate the force produced by the third hydraulic cylinder.

num3 = acos ( ( ah3sq + acsq * a2sq ) / ( 2.0 * ah3 * ac ) )
force(3) = ( torque(3) / ac ) * ( 1.0 / cos ( num3 - pi / 2.0 ) )

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

c Mission complete, see you...

return
end

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

subroutine calgamma ( whatlink, beta )

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

Program name : calgamma.f

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

This is the subroutine that helps to calculate the values of the
admissible function gammas

******************************************************************************
Declare variables and include the common blocks.

implicit real (a-z)

character * 3 whatlink

integer i, numsegment

parameter (numsegment = 100)

dimension u2 (0:numsegment), u3 (0:numsegment), uh3 (0:numsegment)
dimension gv2 (0:numsegment), gv2aq (0:numsegment),
  * gv2p (0:numsegment), gv2pq (0:numsegment),
  * gv2pp (0:numsegment), gv2pqp (0:numsegment)
dimension gw2 (0:numsegment), gw2aq (0:numsegment),
  * gw2p (0:numsegment), gw2pq (0:numsegment),
  * gw2pp (0:numsegment), gw2pqp (0:numsegment)
dimension gv3 (0:numsegment), gv3aq (0:numsegment),
  * gv3p (0:numsegment), gv3pq (0:numsegment),
  * gv3pp (0:numsegment), gv3pqp (0:numsegment)
dimension gw3 (0:numsegment), gw3aq (0:numsegment),
  * gw3p (0:numsegment), gw3pq (0:numsegment),
  * gw3pp (0:numsegment), gw3pqp (0:numsegment)
dimension gvh3 (0:numsegment), gvh3aq (0:numsegment),
  * gvh3p (0:numsegment), gvh3pq (0:numsegment),
  * gvh3pp (0:numsegment), gvh3pqp (0:numsegment)
dimension gwh3 (0:numsegment), gwh3aq (0:numsegment),
  * gwh3p (0:numsegment), gwh3pq (0:numsegment),
  * gwh3pp (0:numsegment), gwh3pqp (0:numsegment)

common /constant/ g, pi, segment

common /link2/ a2r, a2rq, a2rcu, rho2r, mass2r,
  * a2e, rho2e, mass2e, sm2, lam2, eiv2, eiv2,
  * u2, a2, a2aq, costh2, costh2sq, cos2th2,
  * sinh2, sinh2th2
common /link3/ u3, a3, a3cq, rho3, mass3, sm3, lm3, m3cg, eiv3,
  * eiv3, psid, psidaq, psidd, cospsi, cospsiq,
  * sinpsi, sinpsiq,
  * ch1, chid, ch1dd, cosch1, cosch1i, th3ddth2dd,
  * omegasub1, omegasub2, omegasub3
common /linkh3/ eh3s, ah3c, rhoh3s, msh3s,
  * ah3p, ah3start, ah3startc, ah3startcu,
  * ah3pend, ah3pendc, ah3pendcu, rhoh3p, massh3p,
  * ah3r, rhoh3r, massh3r, smh3, eih3,
  * uh3, ah3a, ah3c, ah3cu, ah3fo, ah3fi, ah3se,
  * eta, coseta, sineta, thh3, costhh3, costhsh3,
  * sinthh3, sinthsh3, lamda, conslmda, conslmdasq,
  * sinalmda, sinalmdasq
common /link2gs/ gv2, gv2aq, gv2p, gv2pq, gv2pp, gv2pqp,
  * gv2a2, gv2aq2, gv2ap2, gv2ap2aq,
  * gv2ap2p, gv2ap2pqp,
  * gw2, gw2aq, gw2p, gw2pq, gw2pp, gw2pqp,
  * gw2a2, gw2aq2, gw2ap2, gw2ap2aq,
  * gw2ap2p, gw2ap2pqp
common /link3gs/ gv3, gv3aq, gv3p, gv3pq, gv3pp, gv3pqp,
  * gv3a3, gv3aq3, gv3ap3,
  * gv3, gv3aq, gv3p, gv3pq, gv3pp, gv3pqp,
  * gva3, gva3aq, gva3p, gva3pq, gva3pp, gva3pqp
common /linkh3gs/ gvh3, gvh3aq, gvh3p, gvh3pq, gvh3pp, gvh3pqp,
  * gvh3ah3, gvh3ah3aq, gvh3ah3p, gvh3ah3pq,
  * gvh3ah3pp, gvh3ah3pqp,
  * gwh3, gwh3aq, gwh3p, gwh3pq, gwh3pp, gwh3pqp,
  * gwh3ah3, gwh3ah3aq, gwh3ah3p, gwh3ah3pq,
  * gwh3ah3pp, gwh3ah3pqp
common /theta1/ th1i, th1f, th1, th1d, th1dsq, th1dd
common /theta2/ th2i, th2f, th2, th2d, th2dsq, th2dd
common /theta3/ th3i, th3f, th3, th3d, th3dsq, th3dd
common /vs/ v2a2, v2a2p, v2a2p, v2a2p, v2a2p,
  * v3a3, v3a3p, v3a3p, v3a3p, v3a3p
  * v3a3, v3a3p, v3a3p, v3a3p, v3a3p
  * v3a3, v3a3p, v3a3p, v3a3p, v3a3p
betasq = beta ** 2.0
betacu = beta ** 3.0
betafo = beta ** 4.0
sinbeta = sin ( beta )
sinhbeta = sinh ( beta )
cosbeta = cos ( beta )
coshbeta = cosh ( beta )

C Calculates gammav2, gammav2' and gammav2'' for link 2.
if ( whatlink .eq. ' v2' ) then
  del = - ( sinbeta + sinhbeta ) / ( cosbeta + coshbeta )
do 10 i = 0 , numsegment
  floati = float ( i ) / segment
  temp = beta * floati
  sine = sin ( temp )
sinhe = sinh ( temp )
cosine = cos ( temp )
cosineh = cosh ( temp )
  u2 ( i ) = a2r + ( floati * a2e )
  gv2 ( i ) = ( ( sine - sinhe ) +
              ( del * ( cosine - cosineh ) ) )
  gv2p ( i ) = beta * ( ( cosine - cosineh ) +
              ( del * ( -sine - sinhe ) ) )
  gv2pp ( i ) = betasq * ( ( -sine - sinhe ) +
               ( del * ( -cosine - cosineh ) ) )
  gv2sq ( i ) = gv2 ( i ) ** 2.0
  gv2psq ( i ) = gv2p ( i ) ** 2.0
  gv2ppsq ( i ) = gv2pp ( i ) ** 2.0
10 continue
  gv2a2 = gv2 ( numsegment )
  gv2a2sq = gv2sq ( numsegment )
  gv2a2p = gv2p ( numsegment )
  gv2a2psq = gv2psq ( numsegment )
  gv2a2pp = gv2pp ( numsegment )
  gv2a2ppsq = gv2ppsq ( numsegment )

C Calculates gammaw2, gammaw2' and gammaw2'' for link 2.
else if ( whatlink .eq. ' w2' ) then
  tau3 = ( sm2 * betafo ) / ( rho2e * a2e )
  del = - ( -betacu * cosbeta + tau3 * sinbeta ) -
            ( -betacu * coshbeta + tau3 * sinhbeta ) ) /
            ( ( -betacu * sinbeta + tau3 * cosbeta ) -
            ( -betacu * sinhbeta + tau3 * coshbeta ) )
do 20 i = 0 , numsegment
  floati = float ( i ) / segment
  temp = beta * floati
  sine = sin ( temp )
sinhe = sinh ( temp )
cosine = cos ( temp )
cosineh = cosh ( temp )
gw2 (i) = (( sine - sinh ) + 
( del * ( cosine - cosinh ) ))

gw2p (i) = -beta * (( cosine - cosinh ) + 
( del * (-sine - sinh ) ))

gw2pp (i) = -beta sq * (( -sine - sinh ) + 
( del * ( -cosine - cosinh ) ))

gw2sq (i) = gw2 (i) ** 2.0
gw2psq (i) = gw2p (i) ** 2.0
gw2ppsq (i) = gw2pp (i) ** 2.0

continue

gw2a2 = gw2 (numsegment)
gw2a2sq = gw2sq (numsegment)
gw2a2p = gw2p (numsegment)
gw2a2psq = gw2psq (numsegment)
gw2a2pp = gw2pp (numsegment)
gw2a2ppsq = gw2ppsq (numsegment)

Calculate gammav3, gammav3' and gammav3'' for link 3.

else if ( what I ink .eq. ' v3' ) then

del = -( sinhbeta + sinhbeta ) / ( cosbeta + coshbeta )

continue

gv3a3 = gv3 (numsegment)
gv3a3sq = gv3sq (numsegment)
gv3a3p = gv3p (numsegment)

c Calculate gammaw3, gammaw3' and gammaw3'' for link 3.

else if ( what link .eq. ' w3' ) then

del = -( sinhbeta + sinhbeta ) / ( cosbeta + coshbeta )

continue

gw3 (i) = (( sine - sinh ) + 
( del * ( cosine - cosinh ) ))

gw3p (i) = beta * (( cosine - cosinh ) + 
( del * (-sine - sinh ) ))

gw3pp (i) = beta sq * (( -sine - sinh ) + 
( del * ( -cosine - cosinh ) ))

gw3sq (i) = gw3 (i) ** 2.0
gw3psq (i) = gw3p (i) ** 2.0
gw3ppsq (i) = gw3pp (i) ** 2.0

continue

gw3a3 = gw3 (numsegment)
gw3a3sq = gw3sq (numsegment)
gw3a3p = gw3p (numsegment)
* gw3p (i) = \( \text{del} \times (\cosine - \cosineh) \)
* gw3pp (i) = \( \text{beta} \times (\cosine - \sinh - \sinh) \)
* gw3sq (i) = gw3 (i) ** 2.0
* gw3psq (i) = gw3p (i) ** 2.0
* gw3ppsq (i) = gw3pp (i) ** 2.0

40 continue

gw3a3 = gw3 (numsegment)
gw3a3sq = gw3sq (numsegment)
gw3a3p = gw3p (numsegment)

c ***********************************************
c Calculate gammavh3, gammavh3' and gammavh3'' for link 3.
else if (what link .eq. 'vh3') then
do 50 i = 0, numsegment
   floati = float (i) / segment
   gvh3 (i) = floati ** 2.0 - (floati ** 3.0) / 3.0
   gvh3p (i) = 2.0 * floati - floati ** 2.0
   gvh3pp (i) = 2.0 - 2 * floati
   gvh3sq (i) = gvh3 (i) ** 2.0
   gvh3psq (i) = gvh3p (i) ** 2.0
   gvh3ppsq (i) = gvh3pp (i) ** 2.0
50 continue

gvh3ah3 = gvh3 (numsegment)
gvh3ah3sq = gvh3sq (numsegment)
gvh3ah3cu = gvh3ah3 ** 3.0
gh3ah3p = gvh3p (numsegment)
gvh3ah3pp = gvh3pp (numsegment)
gvh3ah3ppsq = gvh3ppsq (numsegment)

c ***********************************************
c Calculate gammawh3, gammawh3' and gammawh3'' for link 3.
else if (what link .eq. 'wh3') then
\[ \text{ah3} = \text{ah3} - \text{ah3s} \]
if (w2a2p .eq. 0.0) then
   \( \tau = 1.5 \)
else
   \( \tau = \text{abs} \left( \text{w2a2p} / \text{wh3ah3} \right) \times \text{ah3e} \)
end if
\[ \text{del} = \left( \tau - 2.0 \right) / \left( \tau - 3.0 \right) \]
do 60 i = 0, numsegment
   floati = float (i) / segment
   uh3 (i) = ah3s + (floati * ah3e)
   gwh3 (i) = floati ** 2.0 - (floati ** 3.0)
   gwh3p (i) = 2.0 * floati - floati ** 2.0
   gwh3pp (i) = 2.0 - 2 * floati
   gwh3sq (i) = gwh3 (i) ** 2.0
   gwh3psq (i) = gwh3p (i) ** 2.0
   gwh3ppsq (i) = gwh3pp (i) ** 2.0
60 continue
end if

That's it, later...

return
end

subroutine caltorqu

Program name : Caltorqu.f

This subroutine helps to calculate the torques of the system.

Calculate the torques for link 1.
call tlink1

Calculate the torques for link 2.
call tlink2

Calculate the torques for link 3.
call tlink3

Calculate the torques for link c.
call tlinkc

Calculate the torques for link h3.
call tlinkh3

Calculate the torques for the whole system.
call torquot

Time to go, later...

return
end

subroutine clink2
Program name: Clink2.f

This subroutine helps to setup and calculate the coefficients for link 2.

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

Declare variables and include the common blocks.

implicit real (a-z)

integer i, maxarray, numsegment

parameter (maxarray = 9, numsegment = 100)

dimension u2(0:numsegment), l2sub(6)
dimension gv2(0:numsegment), gw2sq(0:numsegment),
* gv2p(0:numsegment), gw2psq(0:numsegment),
* gw2pp(0:numsegment), gw2ppsq(0:numsegment)
dimension element(maxarray,0:numsegment), integral(maxarray)

common /constant/ g, pi, segment
common /link2/ a2r, a2rsq, a2rcu, rho2r, mass2r,
* a2e, rho2e, mass2e, sm2, lm2, eiv2, eiu2,
* u2, a2, a2sq, costh2, costh2sq, costh2p,
* sinh2, sinh2p
common /link2gs/ gv2, gv2sq, gv2p, gv2psq, gw2sq, gw2psq,
* gv2a2, gv2a2sq, gv2a2p, gv2a2psq,
* gw2a2sq, gw2a2psq, gw2a2p, gw2a2ppsq,
* gw2a2pp, gw2a2psq
common /link2gsq/ qv2, qv2sq, qv2d, qv2dsq, qv2dd,
* qw2, qw2sq, qw2d, qw2dsq, qw2dd
common /theeta1/ th1, th1f, th1d, th1dsq, th1dd
common /theeta2/ th2, th2f, th2d, th2dsq, th2dd
common /coeffl2/ coeffa1l2, coeffb1l2, coeffc1l2, coeffd1l2,
* coeffe1l2, coefff1l2, coeffg1l2, coeffh1l2,
* coeffa2l2, coeffb2l2, coeffc2l2, coeffd2l2,
* coeffe2l2, coefff2l2, coeffg2l2, coeffh2l2,
* coeffa3l2, coeffb3l2, coeffc3l2, coeffd3l2,
* coeffe3l2, coefff3l2, coeffg3l2, coeffh3l2,
* coeffa4l2, coeffb4l2, coeffc4l2, coeffd4l2,
* coeffe4l2, coefff4l2, coeffg4l2, coeffh4l2,
* coeffa5l2, coeffb5l2, coeffc5l2, coeffd5l2,
* coeffe5l2, coefff5l2

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

Calculate the values of the integrals of link 2.

do 10 i = 0, numsegment

l2sub(1) = costh2 * u2(i) - sinh2 * qv2 * gv2(i)
l2sub(2) = costh2 * gw2(0) * qv2 * th1d - gv2(i) * qv2 * th2d
l2sub(3) = sinh2 * th1d * u2(i) + costh2 * u2(i) * gv2(i) * qv2 * th1d
l2sub(4) = -costh2 * th1d * u2(i) - gv2(i) * qv2 * sinh2 * th1d +
* gw2(i) * qv2d
l2sub(5) = sinh2 * th2d + u2(i) + costh2 * gv2(i) * qv2 * th2d +
* gw2(i) * qv2d * sinh2l
l2sub(6) = th2d * u2(i) - gw2(i) * qv2 * sinh2 * th1d +
* gv2(i) * qv2d

element(1,i) = gv2a2sq(i)

element(2,i) = gv2a2psq(i)

element(3,i) = 2*l2sub(4)*gv2(i)*sinh2*th1d-2*l2sub(2)*gv2(i)*th2d

1 d

element(4,i) = gv2(i)*(u2(i)*th2dd-costh2*gw2(i)*qv2*th1d*th2d-gw2
1 (i)*qv2*sinh2*th1dd-gw2(i)*qv2d*sinh2*th1d)
element(5, i) = gv2(i)
element(6, i) = gw2sq(i)
element(7, i) = gw2ppsq(i)
element(8, i) = 2*l2sub(2)*costh2*gw2(i)*th1d-2*l2sub(6)*gw2(i)*sin th2*th1d
element(9, i) = gw2(i)*(u2(i)*sin th2*th1d*th2d+costh2*gv2(i)*qv2*th1d*th2d+gv2(i)*qv2*sin th2*th1d)-costh2*u2(i)*th1d+gv2(i)*qv2d*sin th2*th1d)

continue
call integrat (element, maxarray, numsegment, a2r, a2, integral)

Set variables.
e14 = integral(1)*rho2e
e15 = gv2a2sq*sm2
e19 = integral(2)*eiw2*qv2
e20 = -0.5*integral(3)*rho2e
e21 = integral(4)*rho2e
e22 = integral(5)*costh2*gw2a2*sm2
e23 = gw2a2sq*sm2*(a2*th1d*th2d-gw2a2*th1d*th2d-gw2a2*qw2*th1d*th2d-gw2a2*qw2*sin th2*th1d-gw2a2*qw2d*sin th2*th1d)
e24 = -sm2*((2*l2sub(4)*gw2a2*sm2*th1d*th2d+2*l2sub(2)*gw2a2*th1d))/2.0
1 = -costh2*gw2a2

e25 = integral(6)*rho2e
e26 = gw2a2sq*sm2
e29 = integral(7)*eiw2*qw2
e30 = -0.5*integral(8)*rho2e
e31 = integral(9)*rho2e
e32 = -sm2*(2*l2sub(2)*costh2*gw2a2*th1d*th2d-gw2a2*th1d*th2d+gw2a2*sm2*th1d)
e33 = gw2a2*sm2*(a2*sm2*th1d*th2d+gw2a2*qv2*th1d*th2d+gv2a2*qv2d*sm2*th1d)
e34 = gw2a2*sm2*(a2*sm2*th1d*th2d+gw2a2*qv2*th1d*th2d+gv2a2*qv2d*sm2*th1d)
e35 = gw2a2*sm2*(a2*sm2*th1d*th2d+gw2a2*qv2*th1d*th2d+gv2a2*qv2d*sm2*th1d)
e36 = gw2a2*sm2*(a2*sm2*th1d*th2d+gw2a2*qv2*th1d*th2d+gv2a2*qv2d*sm2*th1d)
e37 = gw2a2*sm2*(a2*sm2*th1d*th2d+gw2a2*qv2*th1d*th2d+gv2a2*qv2d*sm2*th1d)
e38 = gw2a2*sm2*(a2*sm2*th1d*th2d+gw2a2*qv2*th1d*th2d+gv2a2*qv2d*sm2*th1d)
e39 = gw2a2*sm2*(a2*sm2*th1d*th2d+gw2a2*qv2*th1d*th2d+gv2a2*qv2d*sm2*th1d)
e40 = gw2a2*sm2*(a2*sm2*th1d*th2d+gw2a2*qv2*th1d*th2d+gv2a2*qv2d*sm2*th1d)
e41 = gw2a2*sm2*(a2*sm2*th1d*th2d+gw2a2*qv2*th1d*th2d+gv2a2*qv2d*sm2*th1d)
e42 = gw2a2*sm2*(a2*sm2*th1d*th2d+gw2a2*qv2*th1d*th2d+gv2a2*qv2d*sm2*th1d)

Calculate the coefficients with the variables just calculated.

coefffl2 = e15+e14
coefffl2 = e26+e25+e22+e21+e20+e19
coefffl2 = e26+e25
coefffl2 = e26+e25
coefffl2 = e33+e32+e31+e30+e29

Mission complete, see you...
return
end

subroutine clink3

Program name : Clink3.f

This subroutine helps to setup and calculate the coefficients for
link 3.

Declare variables and include the common blocks.

implicit real (a-z)
integer i, maxarray, numsegment
parameter ( maxarray = 20, numsegment = 100 )
dimension u2 (0:numsegment), u3 (0:numsegment), l3sub (30)
dimension gv2 (0:numsegment), gv2sq (0:numsegment),
*  gv2p (0:numsegment), gv2psq (0:numsegment),
*  gv2pp (0:numsegment), gv2ppsq (0:numsegment)
dimension gw2 (0:numsegment), gw2sq (0:numsegment),
*  gw2p (0:numsegment), gw2psq (0:numsegment),
*  gw2pp (0:numsegment), gw2ppsq (0:numsegment)
dimension gv3 (0:numsegment), gv3sq (0:numsegment),
*  gv3p (0:numsegment), gv3psq (0:numsegment),
*  gv3pp (0:numsegment), gv3ppsq (0:numsegment)
dimension gw3 (0:numsegment), gw3sq (0:numsegment),
*  gw3p (0:numsegment), gw3psq (0:numsegment),
*  gw3pp (0:numsegment), gw3ppsq (0:numsegment)
dimension element (maxarray,0:numsegment), integral (maxarray)

common /constant/ g, pi, segment
common /link2/ a2r, a2rsq, a2rcu, rho2r, mass2r,
*  a2e, rho2e, mass2e, sm2, lm2, eiv2, eiv2,
*  u2, a2, a2sq, costh2, costh2sq, s2th2,
*  sin2th2, sin2th2
common /link3/ u3, a3, a3c, rho3, mass3, sm3, lm3, m3c, eiv3,
*  eiv3, psid, psidsq, psid1, cospsi, cospsi1q,
*  sinpsi, sinpsiq, sinpsi, sinpsiq, sinpsi, sinpsiq,
*  chi, chid, chidd, coschi, sinchi, th3ddth2dd,
*  omegasub1, omegasub2, omegasub3
common /link2gs/ gv2, gv2sq, gv2p, gv2psq, gv2pp, gv2ppsq,
*  gv2a2, gv2a2sq, gv2a2p, gv2a2psq,
*  gv2a2pp, gv2a2ppsq,
*  gw2, gw2sq, gw2p, gw2psq, gw2pp, gw2ppsq,
*  gw2a2, gw2a2sq, gw2a2p, gw2a2psq,
*  gw2a2pp, gw2a2ppsq
common /link3gs/ gv3, gv3sq, gv3p, gv3psq, gv3pp, gv3ppsq,
*  gv3a3, gv3a3sq, gv3a3p,
*  gw3, gw3sq, gw3p, gw3psq, gw3pp, gw3ppsq,
*  gw3a3, gw3a3sq, gw3a3p
common /link2qs/ qv2, qv2sq, qv2d, qv2dsq, qv2dd,
*  qv2, qv2sq, qv2d, qv2dsq, qv2dd
common /link3qs/ qv3, qv3sq, qv3d, qv3dsq, qv3dd,
*  qv3, qv3sq, qv3d, qv3dsq, qv3dd
common /theta1/ th11, th1f, th1, th1d, th1sq, th1dd
common /theta2/ th21, th2f, th2, th2d, th2sq, th2dd
common /theta3/ th31, th3f, th3, th3d, th3sq, th3dd
common /coefficient/ coeffa1l3, coeffb1l3, coeffc1l3, coeffd1l3,
*  coeffe1l3, coefff1l3, coeffa2l3, coeffb2l3,
*  coeffc2l3, coeffd2l3, coeffe2l3, coefff2l3,
*  coeffa3l3, coeffb3l3, coeffc3l3, coeffd3l3, coeffe3l3, coefff3l3,
* coeffa4l3, coeffb4l3, coeffc4l3, coeffd4l3, coeffe4l3, coefff4l3,
*  coeffa5l3, coeffb5l3, coeffc5l3, coeffd5l3, coeffe5l3, coefff5l3

c Calculate the values of the integrals of link 3.
do 10 i = 0, numsegment
l3sub(1) = cospsi * qv3 * gv3(i) + sinpsi * u3(i)
l3sub(2) = l3sub(1) ** 2.0
l3sub(3) = cospsi * u3(i) - sinpsi * qv3 * gv3(i)
l3sub(4) = cospsi * gv2a2p * u3(i) -
*  gv2a2p * gv3(i) * qv3 * sinpsi
l3sub(5) = gv2a2p * sinpsi * u3(i) +
*  cospsi * gv2a2p * gv3(i) * qv3
l3sub(6) = psid * sinpsi * u3(i) + sinpsi * gv3(i) * qv3d +
*  psid * cospsi * gv3(i) * qv3
l3sub(7) = psid * cospsi * u3(i) + cospsi * gv3(i) * qv3d -
*  psid * sinpsi * gv3(i) * qv3
l3sub(8) = sinpsi * u3(i) + cospsi * gv3(i) * qv3 + gv2a2 * qv2
\[ I_{3sub(9)} = \cos \psi_3 \cdot u_3(i) - g v_3(i) \cdot \sin \psi_3 + a_2 \]

\[ I_{3sub(10)} = \cos \theta_2 \cdot g v_3(i) + q v_3(i) \cdot \sin \theta_2 + g v_2a(i) \cdot q v_2(i) \]

\[ I_{3sub(11)} = \cos \theta_2 \cdot g v_3(i) + q v_3(i) \cdot \sin \theta_2 + g v_2a(i) \cdot q v_2(i) \]

\[ I_{3sub(12)} = \cos \theta_2 \cdot g v_3(i) + q v_3(i) \cdot \sin \theta_2 + g v_2a(i) \cdot q v_2(i) \]

\[ I_{3sub(13)} = g v_2a(i) \cdot I_{3sub(3)} + g v_2a(i) \]

\[ I_{3sub(14)} = (I_{3sub(3)})^2 \cdot 2.0 \]

\[ I_{3sub(15)} = g v_2a(i) \cdot t h_2d - \cos \psi_3 \cdot g v_2a(i) \cdot q v_3(i) \cdot \sin \theta_2 - (I_{3sub(4)})^2 \cdot \sin \theta_2 \]

\[ I_{3sub(16)} = -g v_2a(i) \cdot g v_3(i) \cdot q v_3(i) \cdot \sin \psi_3 - I_{3sub(5)} \cdot \sin \theta_2 \]

\[ I_{3sub(17)} = \cos \chi \cdot g v_2a(i) \cdot u_3(i) - g v_3(i) \cdot \sin \chi + \cos \theta_2 \cdot g v_2a(i) \]

\[ I_{3sub(18)} = -g v_2a(i) \cdot q v_2(i) \cdot \sin \theta_2 + g v_3(i) \cdot q v_3(i) \cdot \sin \psi_3 + \omega_{subl} \cdot g v_3(i) \cdot q w_3(i) - I_{3sub(6)} \cdot \sin \theta_2 \]

\[ I_{3sub(19)} = g v_2a(i) \cdot \sin \chi \cdot \sin \theta_2 + g v_3(i) \cdot q v_3(i) \cdot \cos \theta_2 + g v_2a(i) \cdot q v_2(i) \]

\[ I_{3sub(20)} = g v_2a(i) \cdot \sin \chi \cdot \sin \theta_2 + g v_3(i) \cdot q v_3(i) \cdot \cos \theta_2 + g v_2a(i) \cdot q v_2(i) \]

\[ I_{3sub(21)} = g v_2a(i) \cdot q v_2(i) \cdot \sin \psi_3 + g v_3(i) \cdot \sin \psi_3 + \omega_{subl} \cdot g v_3(i) \cdot q w_3(i) - I_{3sub(6)} \cdot \sin \theta_2 \]

\[ I_{3sub(22)} = a_2 \cdot \sin \theta_2 - g v_3(i) \cdot q v_3(i) \cdot \sin \psi_3 - I_{3sub(4)} \cdot \sin \chi \]

\[ I_{3sub(23)} = g v_3(i) \cdot \cos \chi \cdot \sin \chi \cdot \sin \theta_2 + g v_2a(i) \cdot q v_3(i) \cdot \cos \theta_2 + g v_2a(i) \cdot q v_2(i) \]

\[ I_{3sub(24)} = g v_3(i) \cdot \sin \chi \cdot \sin \psi_3 + g v_2a(i) \cdot q v_3(i) \cdot \cos \theta_2 + g v_2a(i) \cdot q v_2(i) \]

\[ I_{3sub(25)} = -g v_2a(i) \cdot q v_2(i) \cdot \sin \theta_2 + g v_3(i) \cdot q v_3(i) \cdot \sin \psi_3 + \omega_{subl} \cdot g v_3(i) \cdot q w_3(i) - I_{3sub(6)} \cdot \sin \theta_2 \]

\[ I_{3sub(26)} = g v_3(i) \cdot q v_3(i) \cdot \cos \theta_2 + g v_2a(i) \cdot q v_2(i) \cdot \sin \theta_2 + g v_2a(i) \cdot q v_2(i) \]

\[ I_{3sub(27)} = g v_3(i) \cdot q v_3(i) \cdot \cos \theta_2 + g v_2a(i) \cdot q v_2(i) \cdot \sin \theta_2 + g v_2a(i) \cdot q v_2(i) \]

\[ I_{3sub(28)} = g v_3(i) \cdot q v_3(i) \cdot \cos \theta_2 + g v_2a(i) \cdot q v_2(i) \cdot \sin \theta_2 + g v_2a(i) \cdot q v_2(i) \]

\[ I_{3sub(29)} = g v_3(i) \cdot q v_3(i) \cdot \cos \theta_2 + g v_2a(i) \cdot q v_2(i) \cdot \sin \theta_2 + g v_2a(i) \cdot q v_2(i) \]

\[ I_{3sub(30)} = g v_3(i) \cdot q v_3(i) \cdot \cos \theta_2 + g v_2a(i) \cdot q v_2(i) \cdot \sin \theta_2 + g v_2a(i) \cdot q v_2(i) \]

\[ I_{3sub(31)} = g v_3(i) \cdot q v_3(i) \cdot \cos \theta_2 + g v_2a(i) \cdot q v_2(i) \cdot \sin \theta_2 + g v_2a(i) \cdot q v_2(i) \]

\[ I_{3sub(32)} = g v_3(i) \cdot q v_3(i) \cdot \cos \theta_2 + g v_2a(i) \cdot q v_2(i) \cdot \sin \theta_2 + g v_2a(i) \cdot q v_2(i) \]

\[ I_{3sub(33)} = g v_3(i) \cdot q v_3(i) \cdot \cos \theta_2 + g v_2a(i) \cdot q v_2(i) \cdot \sin \theta_2 + g v_2a(i) \cdot q v_2(i) \]

\[ I_{3sub(34)} = g v_3(i) \cdot q v_3(i) \cdot \cos \theta_2 + g v_2a(i) \cdot q v_2(i) \cdot \sin \theta_2 + g v_2a(i) \cdot q v_2(i) \]

\[ I_{3sub(35)} = g v_3(i) \cdot q v_3(i) \cdot \cos \theta_2 + g v_2a(i) \cdot q v_2(i) \cdot \sin \theta_2 + g v_2a(i) \cdot q v_2(i) \]
\[ 8 \sin psi + 2 \sigma_3 (i) \sin psi + \cos psi \]
\[
e_{30} = \text{integral}(9) \cdot r_{30}
\]
\[
e_{31} = 0.5 \cdot \text{integral}(10) \cdot r_{30}
\]
\[
e_{32} = 0.5 \cdot \text{integral}(11) \cdot r_{30}
\]
\[
e_{34} = \text{sm}^3(2g_{2ax}^2 - q_{3ax}^3 g_{2ax}^3 (-l_{3sub(1)} + j \cdot d_{3dd}) - g_{2ax}^2 q_{2x}^2 d + c o s t h_{3dd}) - g_{2ax}^2 q_{2x}^2 t
\]
\[
e_{35} = -s m^3(2g_{2ax}^2 - q_{3ax}^3 g_{2ax}^3 (-l_{3sub(1)} + j \cdot d_{3dd}) - g_{2ax}^2 q_{2x}^2 d + c o s t h_{3dd}) - g_{2ax}^2 q_{2x}^2 t
\]
\[
e_{36} = 0.5 \cdot \text{integral}(12) \cdot r_{30}
\]
\[
e_{37} = 0.5 \cdot \text{integral}(13) \cdot e_{iv3} 
\]
\[
e_{38} = 0.5 \cdot \text{integral}(14) \cdot r_{30}
\]
\[
e_{39} = 0.5 \cdot \text{integral}(15) \cdot r_{30}
\]
\[
e_{40} = 0.5 \cdot \text{integral}(16) \cdot s c o s h \cdot g \cdot r_{30}
\]
\[
e_{41} = 0.5 \cdot \text{integral}(17) \cdot r_{30}
\]
\[
e_{42} = 0.5 \cdot \text{integral}(18) \cdot e_{iv3} 
\]
\[
e_{43} = 0.5 \cdot \text{integral}(19) \cdot r_{30}
\]
\[
e_{44} = 0.5 \cdot \text{integral}(20) \cdot r_{30}
\]
\[
e_{45} = 0.5 \cdot \text{integral}(21) \cdot e_{iv3} 
\]
\[
e_{46} = 0.5 \cdot \text{integral}(22) \cdot r_{30}
\]
\[
e_{47} = 0.5 \cdot \text{integral}(23) \cdot r_{30}
\]
\[
e_{48} = 0.5 \cdot \text{integral}(24) \cdot e_{iv3} 
\]
\[
e_{49} = 0.5 \cdot \text{integral}(25) \cdot r_{30}
\]
\[
e_{50} = 0.5 \cdot \text{integral}(26) \cdot r_{30}
\]
\[
e_{51} = 0.5 \cdot \text{integral}(27) \cdot r_{30}
\]
\[
e_{52} = 0.5 \cdot \text{integral}(28) \cdot r_{30}
\]
\[
e_{53} = 0.5 \cdot \text{integral}(29) \cdot r_{30}
\]
\[
e_{54} = 0.5 \cdot \text{integral}(30) \cdot r_{30}
\]
\[
e_{55} = 0.5 \cdot \text{integral}(31) \cdot r_{30}
\]
\[
e_{56} = 0.5 \cdot \text{integral}(32) \cdot r_{30}
\]
\[
e_{57} = 0.5 \cdot \text{integral}(33) \cdot r_{30}
\]
\[
e_{58} = 0.5 \cdot \text{integral}(34) \cdot r_{30}
\]
\[
e_{59} = 0.5 \cdot \text{integral}(35) \cdot r_{30}
\]
subroutine clinkc

Program name : Clinkc.f

This subroutine helps to setup and calculate the coefficients for link c.

Declare variables and include the common blocks.

implicit real (a-z)

integer numsegment

parameter ( numsegment = 100 )

dimension u2 (0: numsegment), lcsub (28)
dimension gv2 (0: numsegment), gv2sq (0: numsegment),
*   gv2p (0: numsegment), gv2psq (0: numsegment),
*   gv2pp (0: numsegment), gv2ppsq (0: numsegment)
dimension gw2 (0: numsegment), gw2sq (0: numsegment),
*   gw2p (0: numsegment), gw2psq (0: numsegment),
*   gw2pp (0: numsegment), gw2ppsq (0: numsegment)

c /constant/ g, pi, segment
c /link2/ a2r, a2rsc, a2rsc, rho2r, mass2r,
*   a2e, rho2e, mass2e, sm2, lm2, elv2, eiw2,
*   u2, a2, a2sq, costh2, costh2sq, cos2th2,
*   sinth2, sin2th2
c /linkc/ ac, acsq, accu, zeta, rhoc, massc
c /link2gs/ gv2, gv2sq, gv2p, gv2psq, gv2pp, gv2ppsq,
    *   gv2a2, gv2a2sq, gv2a2p, gv2a2psq,
    *   gv2a2pp, gv2a2ppsq,
    *   gw2, gw2sq, gw2p, gw2psq, gw2pp, gw2ppsq,
    *   gw2a2, gw2a2sq, gw2a2p, gw2a2psq,
    *   gw2a2pp, gw2a2ppsq,
c /link2qs/ gv2, gv2sq, gv2d, gv2dsq, gv2dd,
    *   gw2, gw2sq, gw2d, gw2dsq, gw2dd
c /theta1/ th1i, th1f, th1, th3, th1dsq, th1dd
c /theta2/ th2i, th2f, th2, th2d, th2dsq, th2dd
c /theta3/ th3i, th3f, th3, th3d, th3dsq, th3dd
c /coefflc/ coeffa1lc, coeffb1lc, coeffc1lc, coeffd1lc,
    *   coeffe1lc, coefff1lc, coeffg1lc, coeffh1lc,
    *   coeffc2lc, coeffd2lc, coeffe2lc, coefff2lc,
    *   coeffa3lc, coeffb3lc, coeffc3lc, coeffd3lc,
    *   coeffe3lc, coefff3lc, coeffg3lc, coeffh3lc,
    *   coeffc4lc, coeffd4lc, coeffe4lc, coefff4lc,
    *   coeffb5lc, coeffc5lc, coeffd5lc, coeffe5lc,
    *   coefff5lc, coeffg5lc

c
Set variables.

lcsub(1) = 2 * th3d + th2d + 2 * gv2a2p * gv2d
lcsub(2) = th3d + 2 * th2d + gv2a2p * gv2d
lcsub(3) = th3d - th2d + gv2a2p * gv2d
lcsub(4) = 3 * a2 * acsq * gw2a2p - 6 * acsq * gw2a2
lcsub(5) = 2 * accu * gw2a2p - 12 * a2 * ac * gw2a2
$lcsub(6) = 3 \cdot a2sq \cdot ac - 3 \cdot ac \cdot gv2a2sq \cdot gv2sq$

$lcsub(7) = 12 \cdot ac \cdot gw2a2sq \cdot qw2 \cdot qw2d + 6 \cdot ac \cdot gv2a2sq \cdot qv2 \cdot qv2d$

$lcsub(8) = 6 \cdot ac \cdot gw2a2sq \cdot gw2a2p \cdot 2.0 + 3 \cdot ac \cdot gv2a2sq \cdot ac + 3 \cdot a2sq \cdot ac$

$lcsub(9) = -12 \cdot a2 \cdot ac \cdot gw2a2 \cdot qw2 \cdot qv2 \cdot sinth2 - 12 \cdot ac \cdot costh2 \cdot gw2a2 \cdot gw2a2p \cdot qv2 \cdot qw2$

$lcsub(10) = 12 \cdot ac \cdot gw2a2 \cdot gw2a2p \cdot qw2 \cdot qw2d - 12 \cdot ac \cdot gw2a2 \cdot gw2a2p \cdot qv2 \cdot qv2d$

$lcsub(11) = 12 \cdot ac \cdot gw2a2 \cdot gw2a2p \cdot qv2 \cdot qw2d - 12 \cdot ac \cdot gw2a2 \cdot gw2a2p \cdot qv2 \cdot qv2d$

$lcsub(12) = 6 \cdot acsq \cdot gw2a2 \cdot qw2 \cdot th3d - 6 \cdot acsq \cdot gw2a2 \cdot qw2 \cdot th2d - 3 \cdot acsq \cdot gw2a2p \cdot qw2 \cdot gw2a2p \cdot gw2a2p \cdot th2d$

$lcsub(13) = 12 \cdot ac \cdot gw2a2 \cdot gw2a2p \cdot qw2 \cdot qw2d - 12 \cdot ac \cdot gw2a2 \cdot gw2a2p \cdot qw2 \cdot qv2 \cdot qv2d$

$lcsub(14) = -6 \cdot acsq \cdot gw2a2 \cdot qw2 \cdot th3d - 6 \cdot acsq \cdot gw2a2 \cdot qw2 \cdot th2d - 6 \cdot acsq \cdot gw2a2 \cdot qw2 \cdot th2d - 3 \cdot acsq \cdot gw2a2p \cdot gw2a2p \cdot qv2 \cdot th2d$

$lcsub(15) = \sin (2 \cdot zeta + 2 \cdot th3 + 2 \cdot th2 + 2 \cdot gw2a2p \cdot qv2 )$

$lcsub(16) = \cos (2 \cdot zeta + 2 \cdot th3 + 2 \cdot th2 + 2 \cdot gw2a2p \cdot qv2 )$

$lcsub(17) = \sin (zeta + th3 + 2 \cdot th2 + gw2a2p \cdot qv2 )$

$lcsub(18) = \cos (zeta + th3 + 2 \cdot th2 + gw2a2p \cdot qv2 )$

$lcsub(19) = \sin (zeta + th3 + th2 + gw2a2p \cdot qv2 )$

$lcsub(20) = \cos (zeta + th3 + th2 + gw2a2p \cdot qv2 )$

$lcsub(21) = \sin (zeta + th3 - th2 + gw2a2p \cdot qv2 )$

$lcsub(22) = \cos (zeta + th3 - th2 + gw2a2p \cdot qv2 )$

$lcsub(23) = \sin (zeta + th3 + th2 + gw2a2p \cdot qv2 )$

$lcsub(24) = \cos (zeta + th3 + th2 + gw2a2p \cdot qv2 )$

$lcsub(25) = \sin (2 \cdot zeta + 2 \cdot th3 + 2 \cdot th2 + 2 \cdot gw2a2p \cdot qv2 )$

$lcsub(26) = \cos (2 \cdot zeta + 2 \cdot th3 + 2 \cdot th2 + 2 \cdot gw2a2p \cdot qv2 )$

$e8 = (2 \cdot (6 \cdot ac \cdot gv2a2sq \cdot gw2a2p \cdot qv2) + 12 \cdot lcsb(24) \cdot acsq \cdot gv2a2 \cdot qv2) / 12.0$

$e12 = -\rho hoc \cdot lcsb(23) \cdot gv2a2p \cdot (6 \cdot acsq \cdot th2d + 6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e13 = \rho hoc \cdot lcsb(24) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e14 = \rho hoc \cdot lcsb(23) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e15 = \rho hoc \cdot lcsb(22) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e16 = \rho hoc \cdot lcsb(21) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e17 = \rho hoc \cdot lcsb(20) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e18 = \rho hoc \cdot lcsb(19) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e19 = \rho hoc \cdot lcsb(18) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e20 = \rho hoc \cdot lcsb(17) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e21 = \rho hoc \cdot lcsb(16) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e22 = \rho hoc \cdot lcsb(15) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e23 = \rho hoc \cdot lcsb(14) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e24 = \rho hoc \cdot lcsb(13) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e25 = \rho hoc \cdot lcsb(12) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e26 = \rho hoc \cdot lcsb(11) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e27 = \rho hoc \cdot lcsb(10) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e28 = \rho hoc \cdot lcsb(9) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e29 = \rho hoc \cdot lcsb(8) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e30 = \rho hoc \cdot lcsb(7) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$

$e31 = \rho hoc \cdot lcsb(6) \cdot gw2a2p \cdot (6 \cdot acsq \cdot gw2a2p \cdot qv2d )$
csub(23)*acsq*gv2a2*gv2a2p*qv2d*th2d+6*lcsub(24)*acsq*gv2a2*gv2a2
a2p*psid*qv2*th2d-12*ac*gv2a2*gw2a2*gw2a2qv2d*th1d-6*lcsub(19)
-6*Icsub(19)*acsq*gv2a2p*gw2a2*gw2a2qv2d*th1d-6*lcsub(20)*acsq*chid
gv2a2p*psid*qv2*th2d/12.0
e14 = lcsub(20)*acsq*gv2a2p*rhoc/2.0
e15 = (2*(6*ac*gw2a2sq+accu*gw2a2psq)+2*lcsub(28)*accu*gw2a2psq)
e19 = rhoc*(-lcsub(5)*sinth2*th1d+12*ac*costh2*gv2a2*gw2a2*qv1d)
e20 = -rhoc*(-lcsub(19)*(-6*acsq*gw2a2*th1d*th3d-6*acsq*gw2a2*th1d*
12*ac*costh2*gw2a2*qv2d*th1d)+(-12*a2*ac*gw2a2*sinth2*th1d-12*ac*gw2a2*gw2a2qv2d*
12*ac*costh2*gw2a2*qv2d*th1d))/12.0

Calculate the coefficients with the variables just calculated.

coeff1lc = e8/12.0
coeff1f1c = e14+e13+e12
coeff2lc = e15/12.0
coeff2f1c = e20+e19

Mission complete, see you...

return
derend

subroutine clinkh3

Program name: Clinkh3.f

This subroutine helps to setup and calculate the coefficients for
hydraulic cylinder 3.

Declare variables and include the common blocks.

implicit real (a-z)

integer i, maxarray, numsegment

parameter (maxarray = 14, numsegment = 100)
dimension u2 (0:numsegment), uh3 (0:numsegment), lh3sub (25)
dimension gv2 (0:numsegment), gv2sq (0:numsegment),
  *  gv2p (0:numsegment), gv2psq (0:numsegment),
  *  gv2pp (0:numsegment), gv2ppsq (0:numsegment),
dimension gw2 (0:numsegment), gw2sq (0:numsegment),
  *  gw2p (0:numsegment), gw2psq (0:numsegment),
  *  gw2pp (0:numsegment), gw2ppsq (0:numsegment),
dimension gh3 (0:numsegment), gh3sq (0:numsegment),
  *  gh3p (0:numsegment), gh3psq (0:numsegment),
  *  gh3pp (0:numsegment), gh3ppsq (0:numsegment),

dimension gwh3 (0:numsegment), gwh3sq (0:numsegment),
* gwh3p (0:numsegment), gwh3psq (0:numsegment),
* gwh3pp (0:numsegment), gwh3ppsq (0:numsegment)
dimension element (maxarray,0:numsegment), integral (maxarray)

common /constant/ g, pi, segment
common /link2/ a2r, a2rq, a2rcu, rho2r, mass2r,
* a2e, rho2e, mass2e, sm2, ln2, el2, elw2,
* u2, a2, a2q, costh2, costh2q, cos2th2,
* sinth2, sin2th2
common /linkh3/ ah3s, ah3sq, ah3cu, rho3s, mass3s,
* ah3p, ah3pstart, ah3pstartsq, ah3pstartcu,
* ah3pend, ah3pendsq, ah3pendcu, rho3p, mass3p, ah3r, rho3r, mass3r, smh3, eih3,
* uh3, ah3, ah3sq, ah3cu, ah3fo, ah3fi, uh3e,
* eta, coseta, sineta, thh3, costhh3, costhh3sq,
* sinh3h, sinh3h3sq, lama, coslame, coslamsq,
* sinlams, sinlamsasq
common /link2gs/ gv2, gv2q, gv2p, gv2pq, gv2pp, gv2ppsq,
* gv2a2, gv2aq, gv2ap, gv2apq, gv2app,
* gv2a2p, gv2a2pq,
* gw2, gw2q, gw2p, gw2pq, gw2pp, gw2ppsq,
* gw2a2, gw2aq, gw2ap, gw2apq, gw2app,
* gw2a2p, gw2a2pq
common /linkh3gs/ gvh3, gvh3s, gvh3p, gvh3sp, gvh3pp, gvh3ppp,
* gvh3ah3sq, gvh3ah3cu, gvh3ah3p,
* gvh3ah3pp, gvh3ah3p,
* gwh3, gwh3sq, gwh3p, gwh3pp, gwh3ppp,
* gwh3ah3, gwh3ah3sq, gwh3ah3cu, gwh3ah3p,
* gwh3ah3pp, gwh3ah3ppp,
* gwh3ah3sp
common /link2qs/ qv2, qv2q, qv2d, qv2dp, qv2dsq, qv2dds,
* qv2, qv2q, qv2d, qv2dsq, qv2dds
common /linkh3qs/ qvh3, qvh3s, qvh3p, qvh3sp, qvh3pp, qvh3ppp,
* qvh3ah3cu, qvh3ah3p, qvh3ah3pp,
* qvh3ah3sp
common /diffh3/ dah3t, dah3tt, dah3tq2v, dah3th2t, dah3tv2,
* dah3th3, detat, detatt, dotatq2v, dotattth3,
* dotatq3v2, dotatq3v3, dotatq3v4, dotatq3v5,
* dotatq3v6, dotatq3v7, dotatq3v8, dotatq3v9,
* dotatq3v10, dotatq3v11, dotatq3v12, dotatq3v13,
* dotatq3v14, dotatq3v15, dotatq3v16, dotatq3v17,
* dotatq3v18, dotatq3v19, dotatq3v20, dotatq3v21,
* dotatq3v22, dotatq3v23, dotatq3v24, dotatq3v25,
* dotatq3v26, dotatq3v27, dotatq3v28, dotatq3v29,
* dotatq3v30, dotatq3v31, dotatq3v32, dotatq3v33,
* dotatq3v34, dotatq3v35, dotatq3v36, dotatq3v37,
* dotatq3v38, dotatq3v39, dotatq3v40, dotatq3v41,
* dotatq3v42, dotatq3v43, dotatq3v44, dotatq3v45,
* dotatq3v46, dotatq3v47, dotatq3v48, dotatq3v49,
* dotatq3v50, dotatq3v51, dotatq3v52, dotatq3v53,
* dotatq3v54, dotatq3v55, dotatq3v56, dotatq3v57,
* dotatq3v58, dotatq3v59, dotatq3v60, dotatq3v61,
* dotatq3v62, dotatq3v63, dotatq3v64, dotatq3v65,
* dotatq3v66, dotatq3v67, dotatq3v68, dotatq3v69,
* dotatq3v70, dotatq3v71, dotatq3v72, dotatq3v73,
* dotatq3v74, dotatq3v75, dotatq3v76, dotatq3v77,
* dotatq3v78, dotatq3v79, dotatq3v80, dotatq3v81,
* dotatq3v82, dotatq3v83, dotatq3v84, dotatq3v85,
* dotatq3v86, dotatq3v87, dotatq3v88, dotatq3v89,
* dotatq3v90, dotatq3v91, dotatq3v92, dotatq3v93,
* dotatq3v94, dotatq3v95, dotatq3v96, dotatq3v97,
* dotatq3v98, dotatq3v99, dotatq3v100, dotatq3v101,
* dotatq3v102, dotatq3v103, dotatq3v104, dotatq3v105,
* dotatq3v106, dotatq3v107, dotatq3v108, dotatq3v109,
* dotatq3v110, dotatq3v111, dotatq3v112, dotatq3v113,
* dotatq3v114, dotatq3v115, dotatq3v116, dotatq3v117,
* dotatq3v118, dotatq3v119, dotatq3v120, dotatq3v121,
* dotatq3v122, dotatq3v123, dotatq3v124, dotatq3v125,
* dotatq3v126, dotatq3v127, dotatq3v128, dotatq3v129,
* dotatq3v130, dotatq3v131, dotatq3v132, dotatq3v133,
\[ \text{\text{lh3sub}(10) = gw2a2p * qw2dP * costh2 * th1d * gwh3(i) / gwh3ah3p} \]
\[ \text{\text{lh3sub}(11) = gw2a2p * qw2 * costh2 * th1d * gwh3(i) / gwh3ah3p} \]
\[ \text{\text{lh3sub}(12) = gw2a2p * qw2 * sinth2 * th1d * gwh3(i) / gwh3ah3p} \]
\[ \text{\text{lh3sub}(13) = gw2a2p * qw2d * sinth2 * th1d * gwh3(i) / gwh3ah3p} \]
\[ \text{\text{lh3sub}(14) = gw2a2p * qw2 * costh2 * th1d * gwh3(i) / gwh3ah3p} \]
\[ \text{\text{lh3sub}(15) = gw2a2p * qw2 * costh2 * th1d * gwh3(i) / gwh3ah3p} \]
\[ \text{\text{lh3sub}(16) = gw2a2p * qw2d * gwh3(i) / gwh3ah3p} \]
\[ \text{\text{lh3sub}(17) = \text{\text{lh3sub}(16)} - \text{\text{lh3sub}(2)} * \text{\text{th1d}} \]
\[ \text{\text{lh3sub}(18) = dah3qv2 * \text{\text{sin\theta h3}} + \text{\text{lh3sub}(4)} * \text{\text{dthh3qv2}} \]
\[ \text{\text{lh3sub}(19) = cos\theta h3 * dah3qv2 - \text{\text{lh3sub}(3)} * \text{\text{dthh3qv2}} \]
\[ \text{\text{lh3sub}(20) = -gvh3(i) * \text{\text{sin\eta}} - \text{\text{lh3sub}(3)} * \text{\text{dthh3qvh3}} \]
\[ \text{\text{lh3sub}(21) = cos\eta * gvh3(i) + \text{\text{lh3sub}(4)} * \text{\text{dthh3qvh3}} \]
\[ \text{\text{lh3sub}(22) = \text{\text{lh3sub}(2)} + \text{\text{lh3sub}(4)} * \text{\text{sin\eta}} - \text{\text{lh3sub}(3)} * \text{\text{dthh3t}} + \text{\text{lh3sub}(15)} \]
\[ \text{\text{lh3sub}(23) = \text{\text{dah3t}} * \text{\text{sin\eta}} + \text{\text{lh3sub}(1)} * \text{\text{dthh3t}} \]
\[ \text{\text{lh3sub}(24) = \text{\text{detaqvh3}} * \text{\text{uh3(i)}} * \text{\text{sin\theta h3}} + \text{\text{lh3sub}(4)} * \text{\text{dthh3tqvh3}} \]
\[ \text{\text{lh3sub}(25) = \text{\text{lh3sub}(2)} + \text{\text{lh3sub}(4)} * \text{\text{sin\eta}} - \text{\text{dah3t}} * \text{\text{dthh3tqvh3}} \]
\[ \text{\text{lh3sub}(26) = \text{\text{dah3t}} * \text{\text{sin\eta}} - \text{\text{detaqvh3}} * \text{\text{uh3(i)}} * \text{\text{sin\theta h3}} - \text{\text{lh3sub}(3)} * \text{\text{dthh3tqvh3}} \]
\[ \text{\text{lh3sub}(27) = \text{\text{detaqvh3}} * \text{\text{uh3(i)}} * \text{\text{sin\theta h3}} + \text{\text{lh3sub}(4)} * \text{\text{dthh3tqvh3}} \]
\[ \text{\text{element}(1,i) = 2*\text{\text{lh3sub}(19)}^{2} + 2*\text{\text{lh3sub}(18)}^{2} \]
\[ \text{\text{element}(2,i) = 2*\text{\text{lh3sub}(18)}^{2} + 2*\text{\text{lh3sub}(19)}^{2} + 2*\text{\text{lh3sub}(21)}^{2} + 2*\text{\text{lh3sub}(20)}^{2} \]
\[ \text{\text{element}(3,i) = 2*\text{\text{lh3sub}(18)}^{2} + 2*\text{\text{lh3sub}(21)}^{2} + 2*\text{\text{lh3sub}(20)}^{2} \]
\[ \text{\text{element}(4,i) = 2*\text{\text{lh3sub}(18)}^{2} + 2*\text{\text{lh3sub}(21)}^{2} + 2*\text{\text{lh3sub}(20)}^{2} \]
\[ \text{\text{element}(5,i) = \text{\text{detaqvh3}} * \text{\text{uh3(i)}} * \text{\text{sin\theta h3}} + \text{\text{lh3sub}(4)} * \text{\text{dthh3tqvh3}} \]
\[ \text{\text{element}(6,i) = gvh3ppsq(i) \]
\[ \text{\text{element}(7,i) = gvh3ppsq(i) \]
\[ \text{\text{element}(8,i) = 2*\text{\text{lh3sub}(22)}^{2} + 2*\text{\text{lh3sub}(21)}^{2} + 2*\text{\text{lh3sub}(20)}^{2} \]
\[ \text{\text{element}(9,i) = 2*\text{\text{lh3sub}(22)}^{2} + 2*\text{\text{lh3sub}(21)}^{2} + 2*\text{\text{lh3sub}(20)}^{2} \]
\[ \text{\text{element}(10,i) = 2*\text{\text{lh3sub}(22)}^{2} + 2*\text{\text{lh3sub}(21)}^{2} + 2*\text{\text{lh3sub}(20)}^{2} \]
\[ \text{\text{element}(11,i) = 2*\text{\text{lh3sub}(22)}^{2} + 2*\text{\text{lh3sub}(21)}^{2} + 2*\text{\text{lh3sub}(20)}^{2} \]
\[ \text{\text{element}(12,i) = 2*\text{\text{lh3sub}(22)}^{2} + 2*\text{\text{lh3sub}(21)}^{2} + 2*\text{\text{lh3sub}(20)}^{2} \]
\[ \text{\text{element}(13,i) = 2*\text{\text{lh3sub}(22)}^{2} + 2*\text{\text{lh3sub}(21)}^{2} + 2*\text{\text{lh3sub}(20)}^{2} \]
\[ \text{\text{element}(14,i) = -detaqvh3 * gvh3(i) * \text{\text{sin\theta h3}} + \text{\text{lh3sub}(3)} * \text{\text{dthh3tqvh3}} \]
\[ \text{\text{element}(15,i) = 3*\text{\text{cos\theta h3}} * \text{\text{gvh3(i)}} \]

\text{continue}
Set variables.

e21 = 0.5*integral(1)*rhoh3r

e22 = lh3sub(30)*dthh3qvh3**2/3.0

e25 = 0.5*integral(2)*rhoh3r

e26 = lh3sub(30)*dthh3qvh3**2/3.0

e27 = 0.5*((2*lh3sub(18)*lh3sub(23)+2*lh3sub(19)*lh3sub(22))*dah3t

+ integral(3))*rholh3r

1 = ((lh3sub(23)**2+lh3sub(22)**2+lh3sub(17)**2)*dah3q2+integral

1 al(-4))*rholh3r/2.0

e29 = integral(5)*g*rholh3r+lh3sub(1)*dah3q2

e30 = lh3sub(27)*dthh3q2/3.0

e31 = lh3sub(28)*dah3q2*(costh3q2*th1d3q2+dthh3t**2)/6.0

e32 = lh3sub(30)*(2*dthh3t*dthh3q2-2*costh3q2*detaq3v2*sinth33)*th

1 1d3q2/6.0

e33 = lh3sub(30)*dthh3t*dthh3q2/3.0

e34 = lh3sub(28)*dah3t*dthh3q2*dthh3t/3.0

e35 = g*(-2*ah3pend*dah3q2*rholh3r-2*ah3pstart*dah3q2*rholh3r+2*ah

+ 3pend*dah3q2*rholh3r)*sinth33/2.0

e36 = lh3sub(29)*costh33*detaq3v2*2/2.0

e37 = dah3q2*eih3*(gw2a2psq*gw3ah3psq+qw2sq/gw3ah3psq+gw3ah3psq+qvh33h3p

+ psq*qvh33q3)/2.0

e38 = integral(6)*gw2a2psq*rholh3r

e39 = gw3ah3psq

e42 = integral(7)*eih3*gw2a2psq*qw2/gw3ah3psq

e43 = -integral(8)*rholh3r/2.0

e44 = gw2a2p*(lh3sub(17)*dah3t*gwh3ah3+p*rholh3r)*rholh3r/gwh3ah3

1 p

e60 = 0.5*integral(10)*rholh3r

e61 = lh3sub(30)*dthh3qvh3**2/5.0

e62 = integral(11)*eih3*qvh3

e63 = 0.5*((2*lh3sub(21)*lh3sub(23)+2*lh3sub(20)*lh3sub(22))*dah3t

+ integral(12))*rholh3r

e64 = -0.5*integral(13)*rholh3r

e65 = integral(14)*g*rholh3r

e66 = lh3sub(27)*lh3sub(30)*dthh3qvh3/3.0

e67 = -lh3sub(30)*(2*dthh3t*dthh3qvh3-2*costh33*detaq3v3*sinth33)*

1 thd3q2/6.0

e68 = lh3sub(30)*dthh3t*dthh3qvh3/3.0

e69 = lh3sub(28)*dah3t*dthh3qvh3*dthh3t/3.0

e70 = lh3sub(29)*costh33*detaq3v3*g/2.0

Calculate the coefficients with the variables just calculated.

ccoeffa1lh3 = e22+e21

ccoeffa1lh3 = e26+e25
ccoefff1lh3 = e37+e36+e34+e34+e32+e31+e30+e29+e28+e27

ccoefff1lh3 = e38+e29
ccoefff1lh3 = e44+e43+e42
ccoefff1lh3 = e45+e40
ccoefff1lh3 = e46+e42
ccoefff1lh3 = e70+e69+e68+e67+e66+e65+e64+e63+e62

Mission complete, see you...

return
end

subroutine coeff0
Program name: Coeff0.f

This subroutine helps to set up the zero coefficients for all links.

Declare variables and include the common blocks.

```
imPLICIT REAL (A-Z)

COMMON /COEFF1/ COEFFA1L1, COEFFB1L1, COEFFC1L1, COEFFD1L1,
               * COEFFE1L1, COEFFF1L1, COEFFA2L1, COEFFB2L1,
               * COEFFC2L1, COEFFD2L1, COEFFE2L1, COEFFF2L1,
               * COEFFA3L1, COEFFB3L1, COEFFC3L1, COEFFD3L1,
               * COEFFE3L1, COEFFF3L1, COEFFA4L1, COEFFB4L1,
               * COEFFC4L1, COEFFD4L1, COEFFE4L1, COEFFF4L1,
               * COEFFA5L1, COEFFB5L1, COEFFC5L1, COEFFD5L1,
               * COEFFE5L1, COEFFF5L1

COMMON /COEFF2/ COEFFA2L2, COEFFB2L2, COEFFC2L2, COEFFD2L2,
               * COEFFE2L2, COEFFF2L2, COEFFA3L2, COEFFB3L2,
               * COEFFC3L2, COEFFD3L2, COEFFE3L2, COEFFF3L2,
               * COEFFA4L2, COEFFB4L2, COEFFC4L2, COEFFD4L2,
               * COEFFE4L2, COEFFF4L2, COEFFA5L2, COEFFB5L2,
               * COEFFC5L2, COEFFD5L2, COEFFE5L2, COEFFF5L2

COMMON /COEFF3/ COEFFA3L3, COEFFB3L3, COEFFC3L3, COEFFD3L3,
               * COEFFE3L3, COEFFF3L3, COEFFA4L3, COEFFB4L3,
               * COEFFC4L3, COEFFD4L3, COEFFE4L3, COEFFF4L3,
               * COEFFA5L3, COEFFB5L3, COEFFC5L3, COEFFD5L3,
               * COEFFE5L3, COEFFF5L3

COMMON /COEFF4/ COEFFA4L4, COEFFB4L4, COEFFC4L4, COEFFD4L4,
               * COEFFE4L4, COEFFF4L4, COEFFA5L4, COEFFB5L4,
               * COEFFC5L4, COEFFD5L4, COEFFE5L4, COEFFF5L4

COMMON /COEFF5/ COEFFA5L5, COEFFB5L5, COEFFC5L5, COEFFD5L5,
               * COEFFE5L5, COEFFF5L5
```

Set zero coefficient for link 1.

```
COEFFA1L1 = 0.0
COEFFB1L1 = 0.0
COEFFC1L1 = 0.0
COEFFD1L1 = 0.0
COEFFE1L1 = 0.0
COEFFF1L1 = 0.0
COEFFA2L1 = 0.0
COEFFB2L1 = 0.0
COEFFC2L1 = 0.0
COEFFD2L1 = 0.0
COEFFE2L1 = 0.0
COEFFF2L1 = 0.0
COEFFA3L1 = 0.0
COEFFB3L1 = 0.0
COEFFC3L1 = 0.0
COEFFD3L1 = 0.0
COEFFE3L1 = 0.0
COEFFF3L1 = 0.0
```
coeffd3l1 = 0.0
coeffe3l1 = 0.0
coefff3l1 = 0.0
coeffa4l1 = 0.0
coeffb4l1 = 0.0
coeffc4l1 = 0.0
coeffd4l1 = 0.0
coeffe4l1 = 0.0
coefff4l1 = 0.0
coeffa5l1 = 0.0
coeffb5l1 = 0.0
coeffc5l1 = 0.0
coeffd5l1 = 0.0
coeffe5l1 = 0.0
coefff5l1 = 0.0

c  ***********************************************
c  Set zero coefficient for link 2.
coeffa1l2 = 0.0
coeffb1l2 = 0.0
coeffc1l2 = 0.0
coeffd1l2 = 0.0
coeffe1l2 = 0.0
coefff1l2 = 0.0
coeffa2l2 = 0.0
coeffb2l2 = 0.0
coeffc2l2 = 0.0
coeffd2l2 = 0.0
coeffe2l2 = 0.0
coefff2l2 = 0.0
coeffa3l2 = 0.0
coeffb3l2 = 0.0
coeffc3l2 = 0.0
coeffd3l2 = 0.0
coeffe3l2 = 0.0
coefff3l2 = 0.0
coeffa4l2 = 0.0
coeffb4l2 = 0.0
coeffc4l2 = 0.0
coeffd4l2 = 0.0
coeffe4l2 = 0.0
coefff4l2 = 0.0
coeffa5l2 = 0.0
coeffb5l2 = 0.0
coeffc5l2 = 0.0
coeffd5l2 = 0.0
coeffe5l2 = 0.0
coefff5l2 = 0.0

c  ***********************************************
c  Set zero coefficient for link 3.
coeffa1l3 = 0.0
coeffb1l3 = 0.0
coeffc1l3 = 0.0
coeffd1l3 = 0.0
coeffe1l3 = 0.0
coefff1l3 = 0.0
coeffa2l3 = 0.0
coeffb2l3 = 0.0
coeffc2l3 = 0.0
coeffd2l3 = 0.0
coeffe2l3 = 0.0
coefff2l3 = 0.0
coeffa3l3 = 0.0
coeffb3l3 = 0.0
coeffc3l3 = 0.0
coeffd3l3 = 0.0
coeffa3l3 = 0.0
coeffb3l3 = 0.0
coeffc4l3 = 0.0
coeffd4l3 = 0.0
coeffe4l3 = 0.0
coeffa4l3 = 0.0
coeffb4l3 = 0.0
coeffc4l3 = 0.0
coeffd4l3 = 0.0
coeffe4l3 = 0.0
coefff4l3 = 0.0
coeffa5l3 = 0.0
coeffb5l3 = 0.0
coeffc5l3 = 0.0
coeffd5l3 = 0.0
coeffe5l3 = 0.0
coefff5l3 = 0.0

c
**************************************************************************************************************
c
Set zero coefficient for link c.

coeffa1lc = 0.0
coeffb1lc = 0.0
coeffc1lc = 0.0
coeffd1lc = 0.0
coeffe1lc = 0.0
coefff1lc = 0.0
coeffa2lc = 0.0
coeffb2lc = 0.0
coeffc2lc = 0.0
coeffd2lc = 0.0
coeffe2lc = 0.0
coefff2lc = 0.0
coeffa3lc = 0.0
coeffb3lc = 0.0
coeffc3lc = 0.0
coeffd3lc = 0.0
coeffe3lc = 0.0
coefff3lc = 0.0
coeffa4lc = 0.0
coeffb4lc = 0.0
coeffc4lc = 0.0
coeffd4lc = 0.0
coeffe4lc = 0.0
coefff4lc = 0.0
coeffa5lc = 0.0
coeffb5lc = 0.0
coeffc5lc = 0.0
coeffd5lc = 0.0
coeffe5lc = 0.0
coefff5lc = 0.0

c
**************************************************************************************************************
c
Set zero coefficient for link h3.

coeffa1lh3 = 0.0
coeffb1lh3 = 0.0
coeffc1lh3 = 0.0
coeffd1lh3 = 0.0
coeffe1lh3 = 0.0
coefff1lh3 = 0.0
coeffa2lh3 = 0.0
coeffb2lh3 = 0.0
coeffc2lh3 = 0.0
coeffd2lh3 = 0.0
coeffe2lh3 = 0.0
coefff2lh3 = 0.0
coeffa3lh3 = 0.0
coeffb3lh3 = 0.0
coeffc3lh3 = 0.0
coeffd3lh3 = 0.0
coeffe3lh3 = 0.0
coefff3lh3 = 0.0
coeffa4lh3 = 0.0
coeffb4lh3 = 0.0
coeffc4lh3 = 0.0
coeffd4lh3 = 0.0
coeffe4lh3 = 0.0
coefff4lh3 = 0.0
coeffa5lh3 = 0.0
coeffb5lh3 = 0.0
coeffc5lh3 = 0.0
coeffd5lh3 = 0.0
coeffe5lh3 = 0.0
coefff5lh3 = 0.0
**Mission complete, see you...**

return
end

```fortran
subroutine coefftot ( neq, yprime )
  c Program name: Coefftot.f
  c This subroutine helps to add the coefficients of all links
together and store the values into an array.
  c
  Declare variables and include the common blocks.
  implicit real (a-z)
  integer i, j
  integer neq, ipath, lda, n
  parameter ( ipath = 1, lda = 5, n = 5 )
  dimension yprime (neq), a (n,n), b (n), x (n)
  common /link2qs/ qv2, qv2sq, qv2d, qv2dsq, qv2dd,
*   qv2, qv2sq, qv2d, qv2dsq, qv2dd
  common /link3qs/ qv3, qv3sq, qv3d, qv3dsq, qv3dd,
*   qv3, qv3sq, qv3d, qv3dsq, qv3dd
  common /linkh3qs/ qvh3, qvh3sq, qvh3cu, qvh3d, qvh3dsq, qvh3dd
  common /coeff1/ coeffa11, coeffb11, coeffc11, coeffd11,
*   coeffe11, coefff11, coeffg11, coeffh11,
  * coeffa12, coeffb12, coeffc12, coeffd12,
*   coeffe12, coefff12, coeffg12, coeffh12,
  * coeffa13, coeffb13, coeffc13, coeffd13,
*   coeffe13, coefff13, coeffg13, coeffh13,
  * coeffa14, coeffb14, coeffc14, coeffd14,
*   coeffe14, coefff14, coeffg14, coeffh14,
  * coeffa15, coeffb15, coeffc15, coeffd15,
*   coeffe15, coefff15, coeffg15, coeffh15,
  * coeffa21, coeffb21, coeffc21, coeffd21,
*   coeffe21, coefff21, coeffg21, coeffh21,
  * coeffa22, coeffb22, coeffc22, coeffd22,
*   coeffe22, coefff22, coeffg22, coeffh22,
  * coeffa23, coeffb23, coeffc23, coeffd23,
*   coeffe23, coefff23, coeffg23, coeffh23,
  * coeffa24, coeffb24, coeffc24, coeffd24,
*   coeffe24, coefff24, coeffg24, coeffh24,
  * coeffa25, coeffb25, coeffc25, coeffd25,
*   coeffe25, coefff25, coeffg25, coeffh25,
  * coeffa31, coeffb31, coeffc31, coeffd31,
*   coeffe31, coefff31, coeffg31, coeffh31,
  * coeffa32, coeffb32, coeffc32, coeffd32,
*   coeffe32, coefff32, coeffg32, coeffh32,
  * coeffa33, coeffb33, coeffc33, coeffd33,
*   coeffe33, coefff33, coeffg33, coeffh33,
  * coeffa34, coeffb34, coeffc34, coeffd34,
*   coeffe34, coefff34, coeffg34, coeffh34,
  * coeffa35, coeffb35, coeffc35, coeffd35,
*   coeffe35, coefff35, coeffg35, coeffh35,
  * coeffa41, coeffb41, coeffc41, coeffd41,
*   coeffe41, coefff41, coeffg41, coeffh41,
  * coeffa42, coeffb42, coeffc42, coeffd42,
*   coeffe42, coefff42, coeffg42, coeffh42,
  * coeffa43, coeffb43, coeffc43, coeffd43,
*   coeffe43, coefff43, coeffg43, coeffh43,
  * coeffa44, coeffb44, coeffc44, coeffd44,
*   coeffe44, coefff44, coeffg44, coeffh44,
  * coeffa45, coeffb45, coeffc45, coeffd45,
*   coeffe45, coefff45, coeffg45, coeffh45,
  * coeffa51, coeffb51, coeffc51, coeffd51,
*   coeffe51, coefff51, coeffg51, coeffh51,
  * coeffa52, coeffb52, coeffc52, coeffd52,
*   coeffe52, coefff52, coeffg52, coeffh52,
  * coeffa53, coeffb53, coeffc53, coeffd53,
*   coeffe53, coefff53, coeffg53, coeffh53,
  * coeffa54, coeffb54, coeffc54, coeffd54,
*   coeffe54, coefff54, coeffg54, coeffh54,
  * coeffa55, coeffb55, coeffc55, coeffd55,
*   coeffe55, coefff55, coeffg55, coeffh55,
  * coeffa61, coeffb61, coeffc61, coeffd61,
*   coeffe61, coefff61, coeffg61, coeffh61,
  * coeffa62, coeffb62, coeffc62, coeffd62,
*   coeffe62, coefff62, coeffg62, coeffh62,
  * coeffa63, coeffb63, coeffc63, coeffd63,
*   coeffe63, coefff63, coeffg63, coeffh63,
  * coeffa64, coeffb64, coeffc64, coeffd64,
*   coeffe64, coefff64, coeffg64, coeffh64,
  * coeffa65, coeffb65, coeffc65, coeffd65,
*   coeffe65, coefff65, coeffg65, coeffh65,
  * coeffa71, coeffb71, coeffc71, coeffd71,
*   coeffe71, coefff71, coeffg71, coeffh71,
  * coeffa72, coeffb72, coeffc72, coeffd72,
*   coeffe72, coefff72, coeffg72, coeffh72,
  * coeffa73, coeffb73, coeffc73, coeffd73,
*   coeffe73, coefff73, coeffg73, coeffh73,
  * coeffa74, coeffb74, coeffc74, coeffd74,
*   coeffe74, coefff74, coeffg74, coeffh74,
  * coeffa75, coeffb75, coeffc75, coeffd75,
*   coeffe75, coefff75, coeffg75, coeffh75,
  * coeffa81, coeffb81, coeffc81, coeffd81,
*   coeffe81, coefff81, coeffg81, coeffh81,
  * coeffa82, coeffb82, coeffc82, coeffd82,
*   coeffe82, coefff82, coeffg82, coeffh82,
  * coeffa83, coeffb83, coeffc83, coeffd83,
*   coeffe83, coefff83, coeffg83, coeffh83,
  * coeffa84, coeffb84, coeffc84, coeffd84,
*   coeffe84, coefff84, coeffg84, coeffh84,
  * coeffa85, coeffb85, coeffc85, coeffd85,
*   coeffe85, coefff85, coeffg85, coeffh85,
  * coeffa91, coeffb91, coeffc91, coeffd91,
*   coeffe91, coefff91, coeffg91, coeffh91,
  * coeffa92, coeffb92, coeffc92, coeffd92,
*   coeffe92, coefff92, coeffg92, coeffh92,
  * coeffa93, coeffb93, coeffc93, coeffd93,
*   coeffe93, coefff93, coeffg93, coeffh93,
  * coeffa94, coeffb94, coeffc94, coeffd94,
*   coeffe94, coefff94, coeffg94, coeffh94,
  * coeffa95, coeffb95, coeffc95, coeffd95,
*   coeffe95, coefff95, coeffg95, coeffh95,
```
Add the coefficients together.

\[
a_{1,1} = \text{coeffa1l3} + \text{coeffb1l3} + \text{coeffc1l3} + \text{coeffd1l3} + \text{coeffe1l3} \\
a_{1,2} = \text{coeffa1l2} + \text{coeffb1l2} + \text{coeffc1l2} + \text{coeffd1l2} + \text{coeffe1l2} \\
a_{1,3} = \text{coeffa1l1} + \text{coeffb1l1} + \text{coeffc1l1} + \text{coeffd1l1} + \text{coeffe1l1} \\
a_{1,4} = \text{coeffa2l3} + \text{coeffb2l3} + \text{coeffc2l3} + \text{coeffd2l3} + \text{coeffe2l3} \\
a_{1,5} = \text{coeffa2l2} + \text{coeffb2l2} + \text{coeffc2l2} + \text{coeffd2l2} + \text{coeffe2l2} \\
a_{1,6} = \text{coeffa2l1} + \text{coeffb2l1} + \text{coeffc2l1} + \text{coeffd2l1} + \text{coeffe2l1} \\
\]

b(1) = \text{coeffff1l3} + \text{coeffff1l2} + \text{coeffff1l1} \\
b(2) = \text{coeffff2l3} + \text{coeffff2l2} + \text{coeffff2l1} \\
b(3) = \text{coeffff3l3} + \text{coeffff3l2} + \text{coeffff3l1} \\
b(4) = \text{coeffff4l3} + \text{coeffff4l2} + \text{coeffff4l1} \\
b(5) = \text{coeffff5l3} + \text{coeffff5l2} + \text{coeffff5l1}

Call IMSL subroutine to solve the matrix.

call lsarg (n, a, lda, b, ipath, x)

yprime(2) = x(1) 

yprime(4) = x(2) 

yprime(6) = x(3) 

yprime(8) = x(4) 

yprime(10) = x(5) 

call lsarg (n, a, lda, b, ipath, x) 

qv2dd = yprime(2) 

qw2dd = yprime(4) 

qv3dd = yprime(6) 

qw3dd = yprime(8)
qv3d = yprime (10)

Time to go, bye...

return

end

subroutine det3x3 ( matrix, determinate )

Program name: Det3x3.f

This is the subroutine that helps to calculate the determinant of a 3x3 matrix.

Declare variable.

implicit real (a-z)

dimension matrix (3,3)

Perform calculation.

determinate = ( matrix (1,1) * matrix (2,2) * matrix (3,3) ) +
* ( matrix (1,2) * matrix (2,3) * matrix (3,1) ) +
* ( matrix (1,3) * matrix (2,1) * matrix (3,2) ) -
* ( matrix (1,1) * matrix (2,3) * matrix (3,2) ) -
* ( matrix (1,2) * matrix (2,1) * matrix (3,3) ) -
* ( matrix (1,3) * matrix (2,2) * matrix (3,1) )

That's it, later...

return

end

subroutine det4x4 ( betamatrix, determinate )

Program name: Det4x4.f

This is the subroutine that helps to calculate the determinant of a 4x4 matrix.

Declare variables.

implicit real (a-z)

integer i, k, l, m, sign

dimension betamatrix (4,4), matrix (3,3)

Perform calculations.

determinate = 0.0
do 10 i = 1, 4
   if ( i .eq. 1 ) then
      k = 2
      l = 3
      m = 4
   else if ( i .eq. 2 ) then
      k = 1
      l = 3
      m = 4
   else if ( i .eq. 3 ) then
      k = 1
      l = 2
      m = 4
   else if ( i .eq. 4 ) then
      k = 1
      l = 2
      m = 3
   end if
   matrix (1,1) = beta_matrix (2,k)
   matrix (1,2) = beta_matrix (2,l)
   matrix (1,3) = beta_matrix (2,m)
   matrix (2,1) = beta_matrix (3,k)
   matrix (2,2) = beta_matrix (3,l)
   matrix (2,3) = beta_matrix (3,m)
   matrix (3,1) = beta_matrix (4,k)
   matrix (3,2) = beta_matrix (4,l)
   matrix (3,3) = beta_matrix (4,m)
   call det3x3 ( matrix, determinate3x3 )
   sign = ( -1 ) ** ( 1 + i )
   determinate = determinate +
      ( sign * beta_matrix (1,i) * determinate3x3 )
10 continue

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Time to go, see you... *
* return *
* end *

subroutine error
* Program name : Error.f *
* This subroutine helps to calculate the position of the *
* end-effector and the amount of errors. *
* ******************************************************************** *
* Declare variables and include the common blocks. *
* implicit real (a-z) *
* integer numsegment *
* parameter ( numsegment = 100 ) *
* dimension u2 (0:numsegment), u3 (0:numsegment) *
* common /link1/  d1, rho1, mass1, j1 *
* common /link2/  a2r, a2rsq, a2rcu, rho2r, mass2r,
*                a2e, rho2e, mass2e, sm2, lm2, ei2, eiw2,
*                u2, a2, a2eq, costh2, costh2sq, cos2th2,
Calculate the position of the end-effector and the errors.

\[
\text{elastic x} = \cos(\theta_1)\cos(\theta_2)\left(\sin(2a_2p) + a_3\cos(\theta_3) + a_2\cos(\theta_3) + a_3\sin(\theta_3) + a_2\sin(\theta_3)\right)
\]

\[
\text{elastic y} = \sin(\theta_1)\cos(\theta_2)\left(\sin(2a_2p) + a_3\cos(\theta_3) + a_2\cos(\theta_3) + a_3\sin(\theta_3) + a_2\sin(\theta_3)\right)
\]

\[
\text{elastic z} = \sin(\theta_2)\left(\sin(2a_2p) + a_3\cos(\theta_3) + a_2\cos(\theta_3) + a_3\sin(\theta_3) + a_2\sin(\theta_3)\right)
\]

\[
\text{rigid x} = \cos(\theta_1)\cos(\theta_2)\left(\sin(2a_2p) + a_3\cos(\theta_3) + a_2\cos(\theta_3) + a_3\sin(\theta_3) + a_2\sin(\theta_3)\right)
\]

\[
\text{rigid y} = \sin(\theta_1)\cos(\theta_2)\left(\sin(2a_2p) + a_3\cos(\theta_3) + a_2\cos(\theta_3) + a_3\sin(\theta_3) + a_2\sin(\theta_3)\right)
\]

\[
\text{rigid z} = \sin(\theta_2)\left(\sin(2a_2p) + a_3\cos(\theta_3) + a_2\cos(\theta_3) + a_3\sin(\theta_3) + a_2\sin(\theta_3)\right)
\]

\[
\text{error x} = \text{elastic x} - \text{rigid x}
\]

\[
\text{error y} = \text{elastic y} - \text{rigid y}
\]

\[
\text{error z} = \text{elastic z} - \text{rigid z}
\]

Time to go, later...

return

end

subroutine fcn ( neq, x, y, yprime )

Program name: Fcn.f

This is a subroutine that helps to set up the condition of theta 2

and then calculates the corresponding q and q.

Generates the equations used by IMSL (ivpag).

Called by 'ivpag' in 'solver'.

Calls 'calccoeff'.

Declare variables and include the common blocks.

implicit real (a-z)

integer neq
dimension y (neq), yprime (neq)
call subroutine to set variables.
call setvar ( neq, x, y, yprime )
call subroutines to find out the values of the coefficients.
call calccoeff ( neq, yprime )
That's it, later bro...
return
end

 subroutine fcnj ( neq, x, y, dypdy )
c Program name : Fcnj.f
c This is a dummy subroutine that do not do anything.
c Initialize variables.
imPLICIT REAL (a-z)
integer neq
dimension y (neq), dypdy (neq,neq), x
c Later...
return
end

 subroutine init
c Program name : Init.f
c Subroutine to initialize variables and get value from the user.
c Declare variables and include the common blocks.
imPLICIT REAL (a-z)
integer numstep, numsegment
parameter ( numsegment = 100 )
character * 3 whatlink
character * 4 profiletyle
dimension u2 (0:numsegment), u3 (0:numsegment), uh3 (0:numsegment)
dimension polyd1 (5), polye1 (5), polyd2 (5), polye2 (5),
*         polyd3 (5), polye3 (5)
common /constant/g, pi, segment
common /general/density1, length1, idiameter1, odiameter1,
*        csarea1,
*        density2r, length2r, iwidth2r, iheight2r,
*        owidth2r, oheight2r, csarea2r,
*        density2e, elasticity2, areamoi2, areamoiw2,
*        length2e, iwidth2e, iheight2e,
*        owidth2e, oheight2e, csarea2e, lumpedmass2,
*        density3, elasticity3, areamoi3, areamoiw3,
*        length3, iwidth3, iheight3, owidth3, oheight3,
*        csarea3, lumpedmass3,
*        densityc, lengthc, widthc, heightc, zetac,
*        csareac,
*        densityh3s, lengthh3s,
*        idiameterh3s, odiameterh3s, csareah3a,
*        densityh3p, lengthh3p, odiameterh3p, csareah3p,
*        densityh3r, elasticityh3, areamoih3, lengthh3r,
*        odiameterh3r, csareah3r,
*        theta1i, theta1f, theta2i, theta2f,
*        theta3i, theta3f
common /link1/d1, rho1, mass1, j1
common /link2/a2r, a2sq, a2cu, rho2r, mass2r,
*        a2e, rho2e, mass2e, sm2, lm2, ei1v, ei2v,
*        u2, a2v, costh2, costh2sq, costh2cu,
*        sinth2, sin2th2
common /link3/u3, a3, a3sq, rho3, mass3, sm3, lm3, m3sq, ei3v, 
*        ei2v, psd, psd2sq, psd3cu, cospsi, cospsiq,
*        csipsi, csipsiq,
*        ch1, chid, chidd, coschi, sinchi, the3dthh2dd,
*        omegasub1, omegasub2, omegasub3
common /linkc/ax, acsq, accu, zeta, rhoc, massc
common /linkh3/ah3s, ah3sq, rhoh3s, massh3s,
*        ah3p, ah3pstart, ah3pstartsq, ah3pstartcu,
*        ah3pend, ah3pendsq, ah3pendcu, rhoh3p, massh3p,
*        ah3r, rhoh3r, massh3r, smh3, eih3,
*        uh3, ah3, ah3cu, ah3sq, ah3v, ah3psi, ah3sq,
*        eta, coseta, sineta, thh3, costh3, costh3sq,
*        sinthh3, sinthh3sq, lama, coslama, coslamaq,
*        sinlamaq, sinlamaq
common /hydraulic/th1, th2a, th2b
common /theta1/th1i, th1f, th1, th1d, th1da, th1dd
common /theta2/th2i, th2f, th2, th2d, th2da, th2dd
common /theta3/th3i, th3f, th3, th3d, th3da, th3dd
common /profile/profiletype, totalt, motion, numstep, interval,
*        bang1, bang2, bang3,
*        polyd1, polye1, polyd2, polye2, polyd3, polye3
common /imsi/ 
*        hinit, maxstep, tolerance

Read data from the input file.

open ( unit = 10, file = 'input.fil', status = 'old' )
read (10,*) density1
read (10,*) length1, idiameter1, odiameter1
read (10,*) density2r
read (10,*) length2r, iwidth2r, iheight2r, owidth2r, oheight2r
read (10,*) density2e, elasticity2, lumpedmass2
read (10,*) length2e, iwidth2e, iheight2e, owidth2e, oheight2e
read (10,*) density3, elasticity3, lumpedmass3
read (10,*) length3, iwidth3, iheight3, owidth3, oheight3
read (10,*) densityc
read (10,*) lengthc, widthc, heightc, zetac
read (10,*) densityh3s
read (10,*) lengthh3s, idiameterh3s, odiameterh3s
read (10,*) densityh3p
read (10,*) lengthh3p, odiameterh3p
read (10,*) densityh3r, elasticityh3
read (10,*) lengthh3r, odiameterh3r
read (10,*) lh1, lh2a, lh2b
read (10,*) theta1i, theta2i, theta3i
read (10,*) theta1f, theta2f, theta3f
read (10,*) profiletype
read (10,*) totalt, motiont, numstep
read (10,*) maxstep, toleranc

close ( unit = 10 )

c
Set variables for later use.
g = 9.80665
pi = 3.141592654
radian = pi / 180.0
segment = 100.0

c
Set variables for the hydraulic cylinder shell.

ah3s = lengthh3s
ah3ssq = ah3s ** 2.0
ah3scu = ah3s ** 3.0

csareah3s = pi * (odiameterh3s ** 2.0 - idiameterh3s ** 2.0) / 4.0
rhoh3s = densityh3s * csareah3s
massh3s = rhoh3s * ah3s

c
Set variables for the hydraulic cylinder piston.

ah3p = lengthh3p
csareah3p = pi * (odiameterh3p ** 2.0) / 4.0
rhoh3p = densityh3p * csareah3p
massh3p = rhoh3p * ah3p

c
Set variables for the hydraulic cylinder rod.

ah3r = lengthh3r
csareah3r = pi * (odiameterh3r ** 2.0) / 4.0
areamoih3 = pi * (odiameterh3r ** 4.0) / 64.0
rhoh3r = densityh3r * csareah3r
massh3r = rhoh3r * ah3r
smh3 = massh3s
eih3 = elasticityh3 * areamoih3

c
Set variables for the connecting rod.

ac = lengthc
acsq = ac ** 2.0
accu = ac ** 3.0
zeta = zetac * radian
csarea = widthc * heightc
rhoc = densityc * csarea
massc = rhoc * lengthc

c
Set variables for link 3.
a3 = length3
csarea3 = (owidth3 * oheight3) - (iwidth3 * iheight3)
\[
\text{areamoiv3} = (\text{owidth3} \times (\text{oheight3} \times 3.0)) - \\
(\text{iwidth3} \times (\text{iheight3} \times 3.0)) / 12.0 \\
\text{areamoiw3} = (\text{oheight3} \times (\text{owidth3} \times 3.0)) - \\
(\text{iheight3} \times (\text{iwidth3} \times 3.0)) / 12.0 \\
\text{rho3} = \text{density3} \times \text{csarea3} \\
\text{mass3} = \text{rho3} \times \text{a3} \\
\text{sm3} = \text{lumpedmass3} \\
\text{lm3} = \text{lumpedmass3} \\
\text{eiv3} = \text{elasticity3} \times \text{areamoiv3} \\
\text{eiw3} = \text{elasticity3} \times \text{areamoiw3} \\
\text{m3cg} = \text{mass3} + \text{lm3} \\
\text{a3cg} = (\text{a3} \times ((\text{mass3} / 2.0) + \text{lm3})) / \text{m3cg}
\]

Set variables for the rigid portion of link 2.

\[
\text{a2r} = \text{length2r} \\
\text{a2rsq} = \text{a2r}^2 \times 2.0 \\
\text{a2cu} = \text{a2r}^3 \times 3.0 \\
\text{csarea2r} = (\text{owidth2r} \times \text{oheight2r}) - (\text{iwidth2r} \times \text{iheight2r}) \\
\text{rho2r} = \text{density2r} \times \text{csarea2r} \\
\text{mass2r} = \text{rho2r} \times \text{a2r}
\]

Set variables for the elastic portion of link 2.

\[
\text{a2} = \text{length2r} + \text{length2e} \\
\text{a2sq} = \text{a2}^2 \times 2.0 \\
\text{a2e} = \text{length2e} \\
\text{csarea2e} = (\text{owidth2e} \times \text{oheight2e}) - \\
(\text{iwidth2e} \times \text{iheight2e}) \\
\text{areamoiv2} = (\text{owidth2e} \times (\text{oheight2e} \times 3.0)) - \\
(\text{iwidth2e} \times (\text{iheight2e} \times 3.0)) / 12.0 \\
\text{areamoiw2} = (\text{oheight2e} \times (\text{owidth2e} \times 3.0)) - \\
(\text{iheight2e} \times (\text{iwidth2e} \times 3.0)) / 12.0 \\
\text{rho2e} = \text{density2e} \times \text{csarea2e} \\
\text{mass2e} = \text{rho2e} \times \text{a2e} \\
\text{sm2} = \text{lumpedmass2} + \text{mass3} + \text{lumpedmass3} + \text{massc} \\
\text{lm2} = \text{lumpedmass2} \\
\text{eiv2} = \text{elasticity2} \times \text{areamoiv2} \\
\text{eiw2} = \text{elasticity2} \times \text{areamoiw2}
\]

Set variables for link 1.

\[
\text{d1} = \text{length1} \\
\text{csarea1} = \pi \times ((\text{oiameter1} \times 2.0) - (\text{diameter1} \times 2.0)) / 4.0 \\
\text{rho1} = \text{density1} \times \text{csarea1} \\
\text{mass1} = \text{rho1} \times \text{d1} \\
\text{j1} = \text{mass1} \times ((\text{oiameter1} \times 2.0) - (\text{diameter1} \times 2.0)) / 8.0
\]

Set variables for the initial and final displacement parameters for the links.

\[
\text{th1i} = \text{theta1i} \times \text{radian} \\
\text{th1f} = \text{theta1f} \times \text{radian} \\
\text{th2i} = \text{theta2i} \times \text{radian} \\
\text{th2f} = \text{theta2f} \times \text{radian} \\
\text{th3i} = \text{theta3i} \times \text{radian} \\
\text{th3f} = \text{theta3f} \times \text{radian}
\]

Find out the acceleration profile and determine the necessary values for the profile.
call profiles

Set up some of the IMSL parameters.

interval = total / float (numstep)
hinit  = 0.01 * interval

Call subroutine to estimate the values for the admissible function
gamma for link 2.

whatlink = 'v2'
call admissfn (whatlink)

Call subroutine to estimate the values for the admissible
functions gammav3 and gammaw3 for link 3.

whatlink = 'v3'
call admissfn (whatlink)

whatlink = 'w3'
call admissfn (whatlink)

Call subroutine to estimate the values for the admissible function
gamma for link h3.

whatlink = 'vh3'
call admissfn (whatlink)

Call subroutines to initialize all coefficients and torques to
zero.

call coeff0
call torque0

That's it, later...

return

end

subroutine initdefl ( neq, y )

Program name : Initdefl.f

Subroutine that helps to find out the initial deflections of
link 2.

Declare variables and include the common blocks.

implicit real (a-z)

integer neq, numsegment, ipath, lda, n

parameter (numsegment = 100, ipath = 1, lda = 6, n = 6)
Initialize variables.

\[ \text{ah3e} = \text{ah3} - \text{ah3s} \]

\[ \text{Rlm2} = \text{Im2} \times g \]

\[ \text{R3} = \text{mass3} \times g \]

\[ \text{Rc} = \text{massc} \times g \]

\[ \text{Rh3s} = \text{massh3s} \times g \]
\[ Rh3p = massh3p \times g \]
\[ Rh3r = massh3r \times g \]

**Set up the matrix to solve for the reactions at joint 3 and the connecting joint.**

\[
\begin{align*}
a(1,1) &= 1.0 \\
a(1,2) &= 0.0 \\
a(1,3) &= 1.0 \\
a(1,4) &= 0.0 \\
a(1,5) &= 0.0 \\
a(1,6) &= 0.0 \\
a(2,1) &= 0.0 \\
a(2,2) &= 1.0 \\
a(2,3) &= 0.0 \\
a(2,4) &= 1.0 \\
a(2,5) &= 0.0 \\
a(2,6) &= 0.0 \\
a(3,1) &= 0.0 \\
a(3,2) &= 0.0 \\
a(3,3) &= -ah3 \times \sinh3 \\
a(3,4) &= ah3 \times \cosh3 \\
a(3,5) &= 0.0 \\
a(3,6) &= 0.0 \\
a(4,1) &= 0.0 \\
a(4,2) &= 0.0 \\
a(4,3) &= 1.0 \\
a(4,4) &= 0.0 \\
a(4,5) &= 1.0 \\
a(4,6) &= 0.0 \\
a(5,1) &= 0.0 \\
a(5,2) &= 0.0 \\
a(5,3) &= 1.0 \\
a(5,4) &= 0.0 \\
a(5,5) &= 0.0 \\
a(5,6) &= 1.0 \\
a(6,1) &= 0.0 \\
a(6,2) &= 0.0 \\
a(6,3) &= ac \times \sin \left( \pi - (\theta 2i + \theta 3i + \zeta) \right) \\
a(6,4) &= ac \times \cos \left( \pi - (\theta 2i + \theta 3i + \zeta) \right) \\
a(6,5) &= 0.0 \\
a(6,6) &= 0.0
\end{align*}
\]

**Call IMSL subroutine to solve the matrix.**

\[
call Isarg \left( n, a, lda, b, ipath, x \right)
\]

\[
\begin{align*}
reaction2 &= -x(5) \times \sinh2 + x(6) \times \cosh2 \\
reactionh3 &= -x(3) \times \sinh3 + x(4) \times \cosh3
\end{align*}
\]

**The initial deflection for link 2 and hydraulic cylinder 2.**

\[
\begin{align*}
disbload2 &= \left( -\rho h2e \times g \times a2e ** 4.0 \right) / \left( 8.0 \times eiv2 \right) \\
pointload2 &= \left( -reaction2 \times a2e ** 3.0 \right) / \left( 3.0 \times eiv2 \right)
\end{align*}
\]
v2a2 = ( disbload2 * csih2 ) + pointload2

disbloadh3 = ( -rho3r * g * ah3e ** 4.0 ) / ( 8.0 * eih3 )
pointloadh3 = ( reactionh3 * ah3e ** 3.0 ) / ( 3.0 * eih3 )
vh3ah3 = ( disbloadh3 * csih3 ) + pointloadh3

y (1) = v2a2 / gv2a2
y (9) = vh3ah3 / gvh3ah3

v2a2p = gv2a2p * y (1)

The initial deflection for link 3.

v3a3 = ( -Rlm3 * a3 ** 3.0 ) / ( 3.0 * eiv3 ) +
( -rho3 * g * a3 ** 4.0 ) / ( 8.0 * eiv3 )
v3a3 = v3a3 * cos ( th2i + v2a2p + th3i )
y (5) = v3a3 / gv3a3

That's it, later...

return
end

subroutine integrat ( element, maxarray, numsegment, 
upperlimit, lowerlimit, integral )

Program name : Integrat.f

This is a subroutine that make use of the 'Trapezoid rule' to calculate integration.

Declare variables.
implicit real (a-z)
integer i, j, maxarray, numsegment
dimension element (maxarray,0:numsegment), integral (maxarray)
common /constant/ g, pi, segment

Perform calculations.
intervalwidth = ( lowerlimit - upperlimit ) / segment
do 10 i = 1, maxarray
   integral (i) = 0.0
   do 20 j = 1, numsegment
      integral (i) = integral (i) + element (i,i-1) +
      element (i,i)
   20 continue
   integral (i) = integral (i) * intervalwidth / 2.0
10 continue
That's it, later...

return
end

subroutine outputs ( x )

Program name: Outputs.f

This is the subroutine that helps to print the outputs into files.

Declare variables and include the common blocks.

implicit real (a-z)
dimension torque (3), force (3)

common /theta1/ th1i, th1f, th1, th1d, th1dsq, th1dd
common /theta2/ th2i, th2f, th2, th2d, th2dsq, th2dd
common /theta3/ th3i, th3f, th3, th3d, th3dsq, th3dd
common /torques/ torque
common /forces/ force
common /vs/ v2a2, v2a2p, w2a2, w2a2p,
* v3a3, v3a3p, w3a3, w3a3p,
* vh3ah3, vh3ah3p, vh3ah3, vh3ah3p
common /errors/ errorx, errory, errorz
common /trajectory/ rigidx, rigidy, rigidz,
* elasticx, elastics, elastics

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

Set up the format statement.

10 format ( f6.4, 6 ( a1, f8.5 ) )
20 format ( f6.4, 3 ( a1, f8.5, a1, f8.5, a1, f8.5 ) )
30 format ( f6.4, 3 ( a1, f8.5, a1, f8.5, a1, f8.5, a1, f8.5 ) )
40 format ( f6.4, 2 ( a1, f8.5, a1, f8.5, a1, f8.5, a1, f8.5 ) )

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

Find out the deflections and print them onto the output file.

write (10,10) x, ' ', v2a2, ' ', w2a2, ' ', v3a3, ' ', w3a3,
* vh3ah3, ' ', vh3ah3

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

Print the torques onto the output file.

write (11,20) x, ' ', torque(1) / 1000.0,
* ' ', torque(2) / 1000.0, ' ', torque(3) / 1000.0

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

Print the forces onto the output file.

write (12,20) x, ' ', force(1) / 1000.0,
* ' ', force(2) / 1000.0, ' ', force(3) / 1000.0

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

Print the displacement, velocity and acceleration profiles.
write (13,30) x, '//', th1, '//', th1d, '//', th1dd,
  *  '//', th2, '//', th2d, '//', th2dd,
  *  '//', th3, '//', th3d, '//', th3dd

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

Print the errors.

write (14,40) x, '//', errx, '//', erry, '//', errz

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

Print the trajectories.

write (15,50) x, '//', rigidx, '//', rigidy, '//', rigidz,
  *  '//', elasticx, '//', elasctic, '//', elasticz

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

Mission accomplished, see you...

return
end

subroutine profiles

Program name: Profiles.f

This subroutine helps to determine the values of the acceleration profile.

Declare variables and include the common blocks.

implicit real (a-z)
integer i, numstep
integer ipath, lda, n
parameter (ipath = 1, lda = 5, n = 5)
character *4 profiletype
dimension polydl (5), polyel (5), polyd2 (5), polye2 (5),
  *  polyd3 (5), polye3 (5)
dimension matrixa (n,n), matrixb (n)
common /theta1/ th1, th1f, th1, th1d, th1sq, th1dd
common /theta2/ th2, th2f, th2, th2d, th2sq, th2dd
common /theta3/ th3, th3f, th3, th3d, th3sq, th3dd
common /profile/ profiletype, totalt, motiont, numstep, interval,
  *  bang1, bang2, bang3,
  *  polydl, polyel, polyd2, polye2, polyd3, polye3

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

Find out what kind of acceleration profile.

if (profiletype .eq. 'bang') then

bang1 = ( th1f - th1 ) / ( ( motiont / 2.0 ) ** 2.0 )
bang2 = ( th2f - th2 ) / ( ( motiont / 2.0 ) ** 2.0 )
bang3 = ( th3f - th3 ) / ( ( motiont / 2.0 ) ** 2.0 )

write(*,*) 'bang1 = ', bang1
write (*,*) 'bang2 = ', bang2
write (*,*) 'bang3 = ', bang3

else

delta1 = ( thlf - thli ) / 2.0
delta2 = ( th2f - th2i ) / 2.0
delta3 = ( th3f - th3i ) / 2.0

tau1 = motiont / 2.0
tau1sq = tau1 ** 2.0
tau1cu = tau1 ** 3.0
tau1fo = tau1 ** 4.0
tau1fi = tau1 ** 5.0
tau1si = tau1 ** 6.0
tau1se = tau1 ** 7.0
tau1ei = tau1 ** 8.0

tau2 = motiont / 4.0
tau2sq = tau2 ** 2.0
tau2cu = tau2 ** 3.0
tau2fo = tau2 ** 4.0
tau2fi = tau2 ** 5.0

matrixa (1,1) = tau1fo
matrixa (1,2) = tau1fi
matrixa (1,3) = tau1si
matrixa (1,4) = tau1se
matrixa (1,5) = tau1ei
matrixa (2,1) = 12.0 * tau1sq
matrixa (2,2) = 20.0 * tau1cu
matrixa (2,3) = 30.0 * tau1fo
matrixa (2,4) = 42.0 * tau1fi
matrixa (2,5) = 56.0 * tau1si
matrixa (3,1) = 24.0 * tau2
matrixa (3,2) = 60.0 * tau2sq
matrixa (3,3) = 120.0 * tau2cu
matrixa (3,4) = 210.0 * tau2fo
matrixa (3,5) = 336.0 * tau2fi
matrixa (4,1) = 24.0 * tau1
matrixa (4,2) = 60.0 * tau1sq
matrixa (4,3) = 120.0 * tau1cu
matrixa (4,4) = 210.0 * tau1fo
matrixa (4,5) = 336.0 * tau1fi
matrixa (5,1) = 24.0
matrixa (5,2) = 120.0 * tau1
matrixa (5,3) = 360.0 * tau1sq
matrixa (5,4) = 840.0 * tau1cu
matrixa (5,5) = 1680.0 * tau1fo

matrixb (2) = 0.0
matrixb (3) = 0.0
matrixb (4) = 0.0
matrixb (5) = 0.0

matrixb (1) = delta1

call lsarg ( n, matrixa, lda, matrixb, ipath, polyd1 )

matrixb (1) = -delta1

call lsarg ( n, matrixa, lda, matrixb, ipath, polye1 )

matrixb (1) = delta2

call lsarg ( n, matrixa, lda, matrixb, ipath, polyd2 )

matrixb (1) = -delta2

call lsarg ( n, matrixa, lda, matrixb, ipath, polye2 )
matrixb (1) = delta3

call lsarg ( n, matrixa, lda, matrixb, ipath, polyd3 )

matrixb (1) = -delta3

call lsarg ( n, matrixa, lda, matrixb, ipath, polye3 )

do 10 i = 1, 5
          
          write (*,*) polyd1(i), polye1(i), polyd2(i), polye2(i),
                  polyd3(i), polye3(i)
          
10       continue
end if

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

That's it, later...

return
end

subroutine prtparam

Program name : Prtparam.f

This is the subroutine that helps to print the parameters into
and output file.

Declare variables and include the common blocks.

implicit real (a-z)

integer numstep, numsegment

parameter ( numsegment = 100 )

character * 4 profiletype

dimension u2 (0:numsegment), u3 (0:numsegment), uh3 (0:numsegment)
dimension polyd1 (5), polye1 (5), polyd2 (5), polye2 (5),
           polyd3 (5), polye3 (5)

common /general/ density1, length1, idiameter1, odiameter1,
*     csarea1,
*     density2r, length2r, iwidth2r, iheight2r,
*     owidth2r, oheight2r, csarea2r,
*     density2e, elasticity2, areamoi2, areamio2,
*     length2e, iwidth2e, iheight2e,
*     owidth2e, oheight2e, csarea2e, lumpedmass2,
*     density3, elasticity3, areamoi3, areamio3,
*     length3, iwidth3, iheight3, owidth3, oheight3,
*     csarea3, lumpedmass3,
*     ou1, ou2, ou3, ou4, ou5,
*     i1, i2, i3, i4, i5,
*     o1, o2, o3, o4, o5,
*     densityh3s, lengthh3s,
*     idiameterh3s, odiameterh3s, csareaeh3s,
*     densityh3p, lengthh3p, odiameterh3p, csareaeh3p,
*     densityh5r, elasticityh3, areamoih3, lengthh3r,
*     odiameteh3r, csareaeh3r,
*     theta1f, theta1f, theta2f, theta3f,
*     theta3f, theta3f

common /link1/ d1, rho1, mass1, j1

common /link2/ e2r, e2rsq, e2rcu, rho2r, mass2r,
Initialize variables.

10 format (a50)
20 format (a50, 5x, e11.5)
30 format (a50, 1x, f11.5)
40 format (a50, 5x, f7.5, a3, f7.5)

c

Put the general information into 'output.fil' file.

if (profiletype .eq. 'bang') then
    open (unit = 10, file = 'output.fil')
else
    open (unit = 10, file = 'output.fil')
end if

write (10,10) '***** System Parameters *****'
write (10,*)
write (10,10) 'Units are in SI unit : kg, meter and second'
write (10,*)
write (10,10) 'Link 1 Information'
write (10,*)
write (10,30) 'Length (m) =
    length1
write (10,30) 'Inside diameter (m) =
    idiameter1
write (10,30) 'Outside diameter (m) =
    odiameter1
write (10,30) 'Cross section area (m**2) =
    csarea1
write (10,30) 'Density (kg/m**3) =
    density1
write (10,30) 'Mass per unit length (kg/m) =
    rhof1
write (10,30) 'Mass (kg) =
    mass1
write (10,30) 'Mass moment of inertia (kg*m**2) =
    j1
write (10,*) ;
write (10,*) ;
write (10,10) Link 2 Information ;
write (10,*) ;
write (10,30) 'Total length of link 2 (m) =
*  a2
write (10,*) ;
write (10,10) For the rigid portion of link 2
write (10,*) ;
write (10,30) 'Length (m) =
*  length2r
write (10,10) 'Cross sectional dimension
write (10,40) i) Inside width x height (m) =
*  iwidth2r, ' X ', iheight2r
write (10,40) ii) Outside width x height (m) =
*  owidth2r, ' X ', oheight2r
write (10,20) 'Cross section area (m**2) =
*  csarea2r
write (10,30) 'Density (kg/m**3) =
*  density2r
write (10,30) 'Mass per unit length (kg/m) =
*  rho2r
write (10,30) 'Mass (kg) =
*  mass2r
write (10,*) ;
write (10,10) For the Flexible portion of Link 2
write (10,*) ;
write (10,20) 'Modulus of elasticity (Pa) =
*  elasticity2
write (10,30) 'Length (m) =
*  length2e
write (10,10) 'Cross sectional dimension
write (10,40) i) Inside width x height (m) =
*  iwidth2e, ' X ', iheight2e
write (10,40) ii) Outside width x height (m) =
*  owidth2e, ' X ', oheight2e
write (10,20) 'Cross section area (m**2) =
*  csarea2e
write (10,30) 'Density (kg/m**3) =
*  density2e
write (10,30) 'Mass per unit length (kg/m) =
*  rho2e
write (10,30) 'Mass (kg) =
*  mass2e
write (10,20) 'In-plane area moment of inertia (m**4) =
*  areamoi2
write (10,20) 'Out-of-plane area moment of inertia (m**4) =
*  areamoiw2
write (10,30) 'Lump mass at the end of link 2 (kg) =
*  lumpedmass2
write (10,*) ;
write (10,10) Link 3 Information
write (10,*) ;
write (10,20) 'Modulus of elasticity (Pa) =
*  elasticity3
write (10,30) 'Length (m) =
*  length3
write (10,10) 'Cross sectional dimension
write (10,40) i) Inside width x height (m) =
*  iwidth3, ' X ', iheight3
write (10,40) ii) Outside width x height (m) =
*  owidth3, ' X ', oheight3
write (10,20) 'Cross section area (m**2) =
*  csarea3
write (10,30) 'Density (kg/m**3) =
*  density3
write (10,30) 'Mass per unit length (kg/m) =
*  rho3
write (10,30) 'Mass (kg) =
*  mass3
write (10,20) 'In-plane area moment of inertia (m**4) =
/ * write (10,20) 'Out-of-plane area moment of inertia (m**4) = ' / * areamoiw3 / * write (10,30) 'Lump mass at the end of link 3 (kg) = ' / * lumpedmass3 / * write (10,*); / / write (10,10) 'Connecting Link (Link C) Information / * write (10,*) / * write (10,30) 'Length (m) = ' / * lengthc / * write (10,10) 'Cross sectional dimension / * write (10,40) ' i) width x height (m) = ' / * widthc, ' X ', heightc / * write (10,20) 'Cross section area (m**2) = ' / * csareac / * write (10,30) 'Density (kg/m**3) = ' / * densityc / * write (10,30) 'Mass per unit length (kg/m) = ' / * rhoc / * write (10,30) 'Mass (kg) = ' / * massc / * write (10,30) 'Zeta. Fix angle between link c and link 3 (rad) = ' / * zeta / * write (10,30) 'Zeta. Fix angle between link c and link 3 (deg) = ' / * zetac / * write (10,*); / / write (10,10) 'Hydraulic Cylinder 3 (Link H3) Information / * write (10,*); / / write (10,*); / / For the Cylinder Shell of link H3 / * write (10,*); / / write (10,30) 'Length (m) = ' / * lengthh3s / * write (10,30) 'Insider diameter (m) = ' / * idiameterh3s / * write (10,30) 'Outsider diameter (m) = ' / * odiameterh3s / * write (10,20) 'Cross section area (m**2) = ' / * csareah3s / * write (10,30) 'Density (kg/m**3) = ' / * densityh3s / * write (10,30) 'Mass per unit length (kg/m) = ' / * rhoh3s / * write (10,30) 'Mass (kg) = ' / * massh3s / * write (10,*); / / write (10,10) 'For the Piston of Link H3 / * write (10,*); / / write (10,30) 'Length (m) = ' / * lengthh3p / * write (10,30) 'Insider diameter (m) = ' / * 0.0 / * write (10,30) 'Outsider diameter (m) = ' / * odiameterh3p / * write (10,20) 'Cross section area (m**2) = ' / * csareah3p / * write (10,30) 'Density (kg/m**3) = ' / * densityh3p / * write (10,30) 'Mass per unit length (kg/m) = ' / * rhoh3p / * write (10,30) 'Mass (kg) = ' / * massh3p / * write (10,*); / / write (10,10) 'For the Flexible Piston Rod of Link H3 / * write (10,*); / / write (10,30) 'Length (m) = ' / * lengthh3r / * write (10,30) 'Insider diameter (m) = ' / * 0.0 / * write (10,30) 'Outsider diameter (m) = ' / * odiameterh3r / * write (10,20) 'Cross section area (m**2) = ' /


```fortran
* 
write (10,30)'Density (kg/m**3) = ', density3r
write (10,30)'Mass per unit length (kg/m) = ', rhoh3r
write (10,30)'Mass (kg) = ', massh3r
write (10,30)'Area moment of inertia (m**4) = ', areamoi3h3

write (10,*), 'General Information
write (10,*), 'Distance from the center of link 1 to the center of the piston of hydraulic cylinder 1 (m) = ', rh1
write (10,30)'Distance from joint 2 to the joint located at the bottom of hydraulic cylinder 2 (m) = ', th2a
write (10,30)'Distance from joint 2 to the joint located at the top of hydraulic cylinder 2 (m) = ', th2b
write (10,30)'Initial position of link 1 (rad) = ', th1i
write (10,30)'Initial position of link 1 (deg) = ', th1i
write (10,30)'Final position of link 1 (rad) = ', th1f
write (10,30)'Final position of link 1 (deg) = ', th1f
write (10,30)'Initial position of link 2 (rad) = ', th2i
write (10,30)'Initial position of link 2 (deg) = ', th2i
write (10,30)'Final position of link 2 (rad) = ', th2f
write (10,30)'Final position of link 2 (deg) = ', th2f
write (10,30)'Initial position of link 3 (rad) = ', th3i
write (10,30)'Initial position of link 3 (deg) = ', th3i
write (10,30)'Final position of link 3 (rad) = ', th3f
write (10,30)'Final position of link 3 (deg) = ', th3f
write (10,30)'Duration of the experiment (s) = ', totalt
write (10,30)'Duration of motion (s) = ', motiont
write (10,30)'Number of steps within the experiment time frame = ', float (numstep)
write (10,30)'Sampling interval (s) = ', interval
write (10,20)'The initial value of the step size H for IMSL = ', hinit
write (10,20)'The maximum steps for IMSL = ', float (maxstep)
write (10,20)'The tolerance (TOL) for IMSL = ', toleranc

* *******************************************
c
Close the output file.
close ( unit = 10 )
c
* *******************************************
c
That's it, time to go, see you...
return
```
Program name: Setvar.f

This subroutine helps to set up values for all links with respect to the time steps.

Declare variables and include the common blocks.

implicit real (a-z)

integer numsegment

c Parameter ( numsegment = 100 )

character * 3 whatlink

dimension y (neq), yprime (neq), h3var (18)
dimension u2 (0:numsegment), u3 (0:numsegment), uh3 (0:numsegment)
dimension gv2 (0:numsegment), gv2sq (0:numsegment),
  * gv2p (0:numsegment), gv2psq (0:numsegment),
  * gv2pp (0:numsegment), gv2ppsq (0:numsegment),
  * gw2 (0:numsegment), gw2sq (0:numsegment),
  * gw2p (0:numsegment), gw2psq (0:numsegment),
  * gw2pp (0:numsegment), gw2ppsq (0:numsegment),
  * gw3 (0:numsegment), gw3sq (0:numsegment),
  * gw3p (0:numsegment), gw3psq (0:numsegment),
  * gw3pp (0:numsegment), gw3ppsq (0:numsegment),
  * gw3ppp (0:numsegment), gw3pppsq (0:numsegment),
  * gw3pppp (0:numsegment), gw3pppsqs (0:numsegment),
  * gwh3 (0:numsegment), gwh3sq (0:numsegment),
  * gw3ppp (0:numsegment), gw3pppsq (0:numsegment),
  * gw3pppp (0:numsegment), gw3pppsqs (0:numsegment),
  * gw3ppppp (0:numsegment), gw3pppsq (0:numsegment),
  * gw3pppppp (0:numsegment), gw3pppsqs (0:numsegment)

common /constant/ g, pi, segment
common /link2/ a2r, a2rsq, a2rcu, rho2r, mass2r,
  * a2e, rho2e, mass2e, sm2, lm2, eiv2, eiv22,
  * u2, a2, a2sq, costh2, costh2sq, cos2th2,
  * sinth2, sin2th2
common /link3/ u3, a3, a3sq, rho3, mass3, sm3, lm3, m3sq, eiv3,
  * eiv3, psd, psdsq, psdd, cospsi, cospsiq,
  * sinpsi, sinpsiq,
  * ch, chd, chqd, cosch, sinch, thh3dth2dd,
  * omegasub1, omegasub2, omegasub3
common /linkc/ a5c, a5csq, accu, zeta, rhoc, massc
common /linkh3/ ah3s, ah3ssq, ah3scu, rhoh3s, massh3s,
  * ah3p, ah3pstart, ah3pstartsq, ah3pstartcu,
  * ah3pend, ah3pendsq, ah3pendcu, rho3p, mass3p,
  * ah3r, rho3r, mass3r, sinh3, eih3,
  * uh3, ah3, ah3sq, ah3cu, ah3fo, ah3fe,
  * eh3, coset, sinet, thh3, costh3, costh3sq,
  * sinthh3, sinthh3sq, lambda, coslambda, coslandasq,
  * sinlanda, sinlandasq
common /link2g/ gv2, gv2sq, gv2p, gv2psq, gv2pp, gv2ppsq,
  * gv2z, gv2zsq, gv2zq, gv2zp, gv2zpsq,
  * gv2zpsq, gv2zpsqs,
  * gw2, gw2sq, gw2p, gw2psq, gw2pp, gw2ppsq,
  * gw2z, gw2zsq, gw2zq, gw2zp, gw2zpsq,
  * gw2zpsq, gw2zpsqs
Calculate the input angle time history.

call thistory (x)

Setup yprimes.

yrprime (1) = y (2)
yrprime (3) = y (4)
yrprime (5) = y (6)
yrprime (7) = y (8)
yrprime (9) = y (10)

Set variable for link 2.

whatlink = 'w2'
call admissfn (whatlink)

costh2 = cos (th2)
costh2sq = costh2 ** 2.0
cos2th2 = cos (2 * th2)
sinhth2 = sin (th2)
sin2th2 = sin (2 * th2)

qv2 = y (1)
qv2sq = qv2 ** 2.0
qv2d = y (2)
qv2dsq = qv2d ** 2.0
qv2dd = yrprime (2)
qv2 = y (3)
qv2sq = qv2 ** 2.0
qv2d = y (4)
qv2dsq = qv2d ** 2.0
qv2dd = yrprime (4)

w2a2 = gv2a2 * qv2
w2a2p = gv2a2p * qv2
w2a2 = gw2a2 * qv2
w2a2p = gw2a2p * qv2
Set variable for link 3.

\[
\begin{align*}
psid &= \text{th}3 + \text{gv}2a2p \times \text{qv}2d \\
psidsq &= (psid) ^ 2.0 \\
psidd &= \text{th}3dd + \text{gv}2a2p \times \text{qv}2dd \\
cospssi &= \cos(\text{th}3 + \text{gv}2a2p \times \text{qv}2) \\
cospssisq &= (cospssi) ^ 2.0 \\
sinpsi &= \sin(\text{th}3 + \text{gv}2a2p \times \text{qv}2) \\
sinpssisq &= (sinpsi) ^ 2.0 \\
chi &= \text{th}3 + \text{th}2 + \text{gv}2a2p \times \text{qv}2 \\
chid &= \text{th}3d + \text{th}2d + \text{gv}2a2p \times \text{qv}2d \\
chidd &= \text{th}3dd + \text{th}2dd + \text{gv}2a2p \times \text{qv}2dd \\
coschi &= \cos(\chi) \\
sinch &= \sin(\chi) \\
\text{omegaseub}1 &= \text{cosh}2 * \text{th}1d + \text{gv}2a2p \times \text{qv}2d \\
\text{omegaseub}2 &= \text{cosh}2 * \text{th}1dd - \sin th2 * \text{th}1d * \text{th}2d \\
\text{omegaseub}3 &= -\sin th2 * \text{th}1d * \text{th}2d + \text{cosh}2 * \text{th}1dd + \\
& \quad \text{gv}2a2p \times \text{qv}2dd \\
qv3 &= \text{y}(5) \\
qv3sq &= (qv3) ^ 2.0 \\
qv3d &= \text{y}(6) \\
qv3dd &= \text{yprme}(6) \\
qv3 &= \text{y}(7) \\
qv3sq &= (qv3) ^ 2.0 \\
qv3d &= \text{y}(8) \\
qv3 &= \text{yprme}(8) \\
v3a3 &= \text{gv}3a3 \times \text{qv}3 \\
v3a3p &= \text{gv}3a3p \times \text{qv}3 \\
w3a3 &= \text{gw}3a3 \times \text{qv}3 \\
w3a3p &= \text{gw}3a3p \times \text{qv}3 \\
\text{Set variable for hydraulic cylinder 3.} \\
lamda &= \text{th}3 + \text{gv}2a2p \times \text{qv}2 + \text{zeta} \\
coslamda &= \cos(\lamda) \\
coslamdasq &= (coslamda) ^ 2.0 \\
sinlamda &= \sin(\lamda) \\
sinlamdasq &= (sinlamda) ^ 2.0 \\
h3x2 &= \text{ac} \times \text{coslamda} \\
h3x2sq &= (h3x2) ^ 2.0 \\
h3x1 &= \text{h}3x2 + a2 \\
h3x1sq &= (h3x1) ^ 2.0 \\
h3x1cu &= (h3x1) ^ 3.0 \\
h3y1 &= \text{ac} \times \text{sinlamda} \\
h3y &= h3y1 + v2a2 \\
h3ysq &= (h3y) ^ 2.0 \\
ah3sq &= (h3ysq + h3x1sq) \\
ah3 &= \sqrt{ah3sq} \\
ah3cu &= (ah3) ^ 3.0 \\
ah3fo &= (ah3) ^ 4.0 \\
ah3fi &= (ah3) ^ 5.0 \\
ah3se &= (ah3) ^ 7.0 \\
ah3pstart &= ah3 - ah3r - ah3p \\
ah3pstartsq &= (ah3pstart) ^ 2.0 \\
ah3pstartcu &= (ah3pstart) ^ 3.0 \\
ah3pend &= ah3 - ah3r \\
ah3pendsq &= (ah3pend) ^ 2.0
ah3pendcu = ah3pend ** 3.0
wh3ah3 = w2a2 - ac * cos ( pi - lambda ) * sin ( -w2a2p )

whatlink = 'wh3'
call admisfn ( whatlink )
qvh3 = y (9)
qvh3sq = qvh3 ** 2.0
qvh3cu = qvh3 ** 3.0
qvh3d = y (10)
qvh3dsq = qvh3d ** 2.0
qvh3dd = yprime (10)

vh3ah3 = qvh3ah3 * qvh3
vh3ah3p = qvh3ah3p * qvh3
vh3ah3p = w2a2p
ta = atan2 ( h3y , h3x1 ) - ( vh3ah3 / ah3 )
coseta = cos ( eta )
sineta = sin ( eta )

thh3 = eta + th2
costh3 = cos ( thh3 )
costh3sq = costh3 ** 2.0
sinth3 = sin ( thh3 )
sinth3sq = sinh3 ** 2.0

h3var(1) = 2.0 * ac * coslambda * h3y -
* 2.0 * ac * sinlambda * h3x1
h3var(2) = ac * coslambda * qv2a2p + qv2a2
h3var(3) = ( h3y sq / h3x1sq ) + 1.0
h3var(4) = h3var(3) ** 2.0
h3var(5) = ( qvh3ah3sq * qvh3sq / ah3sq ) + 1.0
h3var(6) = h3var(5) ** 2.0
h3var(7) = gv2a2 * qv2d + ac * coslambda * psid
h3var(8) = h3var(7) ** 2.0
h3var(9) = ac * psid * sinlambda * h3y
h3var(10) = ( h3var(9) / h3x1sq ) + ( h3var(7) / h3x1 )
h3var(11) = ( ac * sinlambda * h3y / h3x1sq ) + ( h3x2 / h3x1 )
h3var(12) = 2.0 * h3var(2) / h3y -
* 2.0 * ac * qv2a2p * sinlambda * h3y
h3var(13) = 2.0 * h3var(7) / h3y -
* 2.0 * ac * psid * sinlambda * h3x1
h3var(14) = h3var(13) ** 2.0
h3var(15) = ( qvh3ah3 / qv3h3 / ah3 ) -
* ( qvh3ah3 / h3var(15) / qvh3 / ( 2.0 * ah3cu ) )
* ( h3var(16) )
* ( 2.0 * ac * qv2a2p * sinlambda * h3y sq / h3x1cu )
* ( 2.0 * h3var(2) / h3y / h3x1sq )
* ( 2.0 * ac * sinlambda * h3y sq / h3x1cu )
* ( 2.0 * ac * coslambda * h3y / h3x1sq )
* ( ac * qv2a2p * sinlambda * h3y / h3x1sq )
* ( h3var(2) / h3x1 )


dah3t = h3var(13)/ah3/2.0
dah3t = (2*acsq*psidq*sinlandasq+2*h3y*(-ac*psidq*sinlanda+gv2a
1 2*qv2dd*ac*coslanda*psisd)-2*ac*h3x1*psid*sinlanda-2*ac*psid
2 da*h3x1*psidq*2*h3var(8))/ah3/2.0-h3var(14)/ah3cu/4.0
dah3qv2 = (2*acsq*gv2a2p*psid*sinlandasq-2*ac*gv2a2p*h3y*psid*sin
1 landa-2*ac*coslanda*gv2a2p*x1*psid+2*h3var(2)*h3var(7))/ah3/2
2 .0-h3var(12)/h3var(13)/ah3cu/4.0
dah3th3 = (2*acsq*psid*sinlandasq-2*ac*h3y*psid*sinlanda-2*ac*cos
1 landa*3*hx1*psid*2*h3var(7)*ac*coslanda)/ah3/2.0-h3var(13)/ah3cu/4.0
dah3qv2 = h3var(12)/ah3/2.0
dah3th3 = h3var(1)/ah3/2.0

detet = h3var(10)/h3var(3)-h3var(15)/h3var(5)
detett = (-qvvh3ah3*gv3h3*(2*ac*psidq*sinlandasq+2*h3y*(-ac*psid
1 sq*sinlanda+gv2a2qv2dd*ac*coslanda*psisd)-2*ac*h3x1*psidq*sin
2 landa-2*ac*coslanda*h3x1*psidsq+2*h3var(8))/ah3cu/2.0+vh3ah3*gv3
3 h3dd/ah3-h3var(13)*gvh3ah3*qvh3d/ah3cu+3.0*h3var(14)*gvh3ah3*qv
4 h3/(4.0*ah3fi))/h3var(5)+(2*acsq*h3y*psidsq*sinlamdasq/h3x1cu*(
5 - ac*psidsq*inlamba+gv2a2*gv2dd+ac*coslamba*psidd)/h3x1+ac*h3y*
6 psid*inlamba/h3x1sq+2*h3var(7)*ac*psid*inlamba/h3x1sq+ac*cos
7 lamba*h3y*psidsq/h3x1sq)/h3var(3)-h3var(10)*(2*ac*h3y*cos*psid*si
8 nlamba/h3x1cu*2*h3var(7)*h3y/h3x1sq)+h3var(4)+h3var(15)*(2*gvh3
9 ah3sq*qv*h3d/ah3cu*h3var(15)*gvh3ah3sq*qv*h3sq/ah3fo)/h3var(
: 6)

detatqv2 = -(-gvh3ah3sq*qv*h3d/(2*acsq*gv2a2p*psid*inlamba*2-2*ac*gv2
1 a2p*h3y*psid*inlamba-2*ac*coslamba*gv2a2p*h3x1*psid+2*h3var(2)
2 *h3var(7))/ah3cu/2.0-h3var(12)*gvh3ah3*qv*h3d/ah3cu/2.0+3.0*h3va
3 r(12)*h3var(13)*gvh3ah3*qv/h3x1cu/(4.0*ah3fi))/h3var(5)*(2*acsq*gv2a
4 2p*h3y*psid*inlamba/h3x1cu*h3var(2)*ac*psid*inlamba/h3x1sq-
5 ac*gv2a2p*psid*inlamba/h3x1/h3var(7)*ac*gv2a2p*psid*inlamba/h3x1sq
+ac*coslamba*gv2a2p*psid*inlamba/h3x1cu*h3var(7)*ac*gv2a2p*psid*inlamba/h3x1sq
+ac*coslamba*gv2a2p*psid*inlamba/h3x1cu*h3var(12)*h3var(1)
7 5)*gvh3ah3sq*qv*h3sq/(h3var(6)*ah3fo)-h3var(10)*h3var(16)/h3var(
8 4)

detath3 = -ac*coslamba*gv2a2p*psid*inlamba*2-2*ac*coslamba*gv2
1 a2p*h3y*psid*inlamba-2*ac*coslamba*gv2a2p*h3x1*psid+2*h3var(2)
2 *h3var(7))/ah3cu/2.0-h3var(12)*gvh3ah3*qv*h3d/ah3cu/2.0+3.0*h3va
3 r(12)*h3var(13)*gvh3ah3*qv/h3x1cu/(4.0*ah3fi))/h3var(5)*(2*acsq*gv2a
4 2p*h3y*psid*inlamba/h3x1cu*h3var(2)*ac*psid*inlamba/h3x1sq-
5 ac*gv2a2p*psid*inlamba/h3x1/h3var(7)*ac*gv2a2p*psid*inlamba/h3x1sq
+ac*coslamba*gv2a2p*psid*inlamba/h3x1cu*h3var(7)*ac*gv2a2p*psid*inlamba/h3x1sq
+ac*coslamba*gv2a2p*psid*inlamba/h3x1cu*h3var(12)*h3var(1)
7 5)*gvh3ah3sq*qv*h3sq/(h3var(6)*ah3fo)-h3var(10)*h3var(16)/h3var(
8 4)

detatqv3 = 2*h3var(15)*gvh3ah3sq*qv*h3d/(h3var(6)*ah3sq)+h3var(13)*
1 gvh3ah3/(h3var(5)*ah3cu)/2.0
detatqv2 = h3var(12)*gvh3ah3*qv*h3d/(h3var(5)*ah3cu)/2.0+h3var(18)/h3
1 var(3)
detatqv3 = h3var(15)*gvh3ah3sq*qv*h3d/(h3var(5)*ah3cu)/2.0+h3var(11)/h3
1 var(3)
detatqvh3 = gvh3ah3/(h3var(5)*ah3cu)/2.0
detatqv3 = h3var(15)*gvh3ah3sq*qv*h3d/(h3var(6)*ah3sq)+h3var(13)*
1 gvh3ah3/(h3var(5)*ah3cu)/2.0
detatqv2 = h3var(12)*gvh3ah3*qv*h3d/(h3var(5)*ah3cu)/2.0+h3var(18)/h3
1 var(3)
detatqv3 = h3var(15)*gvh3ah3sq*qv*h3d/(h3var(5)*ah3cu)/2.0+h3var(11)/h3
1 var(3)
detatqvh3 = gvh3ah3/(h3var(5)*ah3cu)/2.0

c ***************************************************************
c
return
end
c

 subroutine solver
c
 Program name : Solver.f
c
 This is the subroutine that helps to solve our problem.
c
 ***************************************************************
c
 Declare variables and include the common blocks.
 external fcn, fcnj
 implicit real (a-z)
 integer i, numstep
t integer neq, nparam, ido, inorm, meth, miter
character * 4 profilename

dimension y (neq), yprime (neq), param (nparam)
* polyd1 (5), polye1 (5), polyd2 (5), polye2 (5),
  polyd3 (5), polye3 (5)

common /profile/ profilename, totalt, motion, numstep, interval,
  bang1, bang2, bang3,
  polyd1, polye1, polyd2, polye2, polyd3, polye3
common /imsl/ hinit, maxstep, tolerance

set variable.

x = 0.0

Calculate initial conditions for the ys (qs) from the given
boundary condition.

do 10 i = 1, 10
  y (i) = 0.0
  yprime (i) = 0.0
10 continue

if (profilename .eq. 'bang') then
  open (unit = 10, file = 'bdeflections.fil'
  open (unit = 11, file = 'btorques.fil'
  open (unit = 12, file = 'bforces.fil'
  open (unit = 13, file = 'bprofile.fil'
  open (unit = 14, file = 'berrors.fil'
  open (unit = 15, file = 'btrajectory.fil'
else
  open (unit = 10, file = 'pdeflections.fil'
  open (unit = 11, file = 'ptorques.fil'
  open (unit = 12, file = 'pforces.fil'
  open (unit = 13, file = 'pprofile.fil'
  open (unit = 14, file = 'perrors.fil'
  open (unit = 15, file = 'ptrajectory.fil'
end if

Call subroutine 'fcn' to set up the variables for the coefficients
and torques, call 'calctorqu' to calculate the result at time 0 and
call 'error' to calculate the position of the end-effector at time
zero.

call fcn (neq, x, y, yprime)
call calctorqu (x)
call calforce
call error
call outputs (x)
Set values for the IMSL subroutine.

ido = 1
inorm = 1
meth = 2
miter = 0
tol = tolerance

Call the IMSL subroutines. Then call 'calcruq' to calculate the
 corresponding torque for the three joints and call 'error' to
calculate the position of the end-effector.

call sset (nparam, 0.0, param, 1)

param (1) = hinit
param (4) = maxstep
param (10) = inorm
param (12) = meth
param (13) = miter

do 20 i = 1, numstep
    xend = float (i) * interval
    call ivpag (ido, neq, fcn, fcnj, a, x, xend, tol, param, y)
call fcn (neq, x, y, yprime)
call calcruq (x)
call calfor

call error
call outputs (x)
20 continue

ido = 3

call ivpag (ido, neq, fcn, fcnj, a, x, xend, tol, param, y)

Close the output files.

close (unit = 10)
close (unit = 11)
close (unit = 12)
close (unit = 13)
close (unit = 14)
close (unit = 15)

Mission accomplished, see you...

return
end

subroutine history (x)

Program name: History.f

This subroutine helps to set up the time history of the
acceleration profile, velocity profile and the displacement
profile.

Declare variables and include the common blocks.
implicit real (a-z)

integer numstep
integer i, j

character * 4 prolife

dimension polyd1 (5), polyel (5), polyd2 (5), polye2 (5),
* polyd3 (5), polye3 (5)

common /theta1/ th1, th1f, th1d, th1dsq, th1dd
common /theta2/ th2, th2f, th2d, th2dsq, th2dd
common /theta3/ th3, th3f, th3d, th3dsq, th3dd
common /profile/ prolife, totalt, motiont, numstep, interval,
* bang1, bang2, bang3,
* polyd1, polye1, polyd2, polye2, polyd3, polye3

Calculate the input angle time history.

if ( prolife .eq. "bang" ) then

if ( x .lt. 0.5 * motiont ) then

th1 = th1i + ( 0.5 * bang1 * ( x ** 2.0 ) )
th1d = bang1 * x
th1dsq = th1d ** 2.0
th1dd = bang1
th2 = th2i + ( 0.5 * bang2 * ( x ** 2.0 ) )
th2d = bang2 * x
th2dsq = th2d ** 2.0
th2dd = bang2
th3 = th3i + ( 0.5 * bang3 * ( x ** 2.0 ) )
th3d = bang3 * x
th3dsq = th3d ** 2.0
th3dd = bang3

else if ( ( x .ge. 0.5 * motiont ) .and.
* ( x .lt. motiont ) ) then

th1 = th1f - ( 0.5 * bang1 * ( ( motiont-x ) ** 2.0 ) )
th1d = bang1 * ( motiont - x )

** th1dsq = th1d ** 2.0
th1dd = -bang1

th2 = th2f - ( 0.5 * bang2 * ( ( motiont-x ) ** 2.0 ) )
th2d = bang2 * ( motiont - x )

** th2dsq = th2d ** 2.0
th2dd = -bang2

th3 = th3f - ( 0.5 * bang3 * ( ( motiont-x ) ** 2.0 ) )
th3d = bang3 * ( motiont - x )

** th3dsq = th3d ** 2.0
th3dd = -bang3

else

th1 = th1f
th1d = 0.0
th1dsq = 0.0
th1dd = 0.0

th2 = th2f
th2d = 0.0
th2dsq = 0.0
th2dd = 0.0

th3 = th3f
th3d = 0.0
th3dsq = 0.0
th3dd = 0.0

end if
else

if ( x .lt. 0.5 * motiont ) then

  th1 = th1i
  th1d = 0.0
  th1dd = 0.0
  th2 = th2i
  th2d = 0.0
  th2dd = 0.0
  th3 = th3i
  th3d = 0.0
  th3dd = 0.0

  do 10 i = 4, 8
      j = i - 3
      th1 = th1 + polydl ( j ) * ( x ** float ( i ) )
      th1d = th1d + float ( i ) * polydl ( j ) * ( x ** float ( i-1 ) )
      th1dd = th1dd + float ( i ) * float ( i-1 ) * polydl ( j ) * ( x ** float ( i-2 ) )
      th2 = th2 + polyd2 ( j ) * ( x ** float ( i ) )
      th2d = th2d + float ( i ) * polyd2 ( j ) * ( x ** float ( i-1 ) )
      th2dd = th2dd + float ( i ) * float ( i-1 ) * polyd2 ( j ) * ( x ** float ( i-2 ) )
      th3 = th3 + polyd3 ( j ) * ( x ** float ( i ) )
      th3d = th3d + float ( i ) * polyd3 ( j ) * ( x ** float ( i-1 ) )
      th3dd = th3dd + float ( i ) * float ( i-1 ) * polyd3 ( j ) * ( x ** float ( i-2 ) )
      continue
  enddo

  10 th1sq = th1 ** 2.0
      th2dsq = th2d ** 2.0
      th3dsq = th3d ** 2.0

else if ( ( x .ge. 0.5 * motiont ) .and. ( x .lt. motiont ) ) then

  th1 = th1f
  th1d = 0.0
  th1dd = 0.0
  th2 = th2f
  th2d = 0.0
  th2dd = 0.0
  th3 = th3f
  th3d = 0.0
  th3dd = 0.0

  do 20 i = 4, 8
      j = i - 3
      th1 = th1 + polyel ( j ) * ( ( motiont - x ) ** float ( i ) )
      th1d = th1d + float ( i ) * polyel ( j ) * ( ( motiont - x ) ** float ( i-1 ) )
      th1dd = th1dd + float ( i ) * float ( i-1 ) * polyel ( j ) * ( ( motiont - x ) ** float ( i-2 ) )
      th2 = th2 + polye2 ( j ) * ( ( motiont - x ) ** float ( i ) )
      th2d = th2d + float ( i ) * polye2 ( j ) * ( ( motiont - x ) ** float ( i-1 ) )
      th2dd = th2dd + float ( i ) * float ( i-1 ) * polye2 ( j ) * ( ( motiont - x ) ** float ( i-2 ) )
      th3 = th3 + polye3 ( j ) * ( ( motiont - x ) ** float ( i ) )
subroutine tlink1

Program name: Tlink1.f

This subroutine helps to setup and calculate the torques for link 1.

Declare variables and include the common blocks.

implicit real (a-z)

common /link1/  d1, rho1, mass1, j1
common /theta1/  th1i, thf, th1, th1d, th1dsq, th1dd
common /torque1/  torque1l1, torque2l1, torque3l1

Set variables.

torque1l1 = j1*th1dd

Mission complete, see you...
**Program name: Tlink2.f**

This subroutine helps to setup and calculate the torques for link 2.

Declare variables and include the common blocks.

```
implicit real (a-z)
integer i, maxarray, numsegment
parameter ( maxarray = 4, numsegment = 100 )
dimension u2 (0:numsegment), l2sub (6)
dimension gv2 (0:numsegment), gv2sq (0:numsegment),
*   gv2p (0:numsegment), gv2psq (0:numsegment),
*   gv2pp (0:numsegment), gv2ppsq (0:numsegment)
dimension gw2 (0:numsegment), gw2sq (0:numsegment),
*   gw2p (0:numsegment), gw2psq (0:numsegment),
*   gw2pp (0:numsegment), gw2ppsq (0:numsegment)
dimension element (maxarray,0:numsegment), integral (maxarray)
```

Calculate the values of the integrals of link 2.

```
do 10 i = 0, numsegment
  l2sub(1) = costh2 * u2(i) - sinh2 * qv2 * gv2(i)
  l2sub(2) = costh2 * gw2(i) * qv2 * th1d - gv2(i) * qv2 * th2d
  l2sub(3) = sinh2 * th1d * u2(i) + costh2 * gv2(i) * qv2 * th1d
  l2sub(4) = -costh2 * th1d * u2(i) + gv2(i) * qv2 * sinh2 * th1d +
*   gw2(i) * qv2d
  l2sub(5) = sinh2 * th2d * u2(i) + costh2 * gw2(i) * qv2 * th2d +
*   qv2(i) * qv2d * sinh2
  l2sub(6) = th2d * u2(i) - gw2(i) * qv2 * sinh2 * th1d +
*   qv2(i) * qv2d
```

```
element(1,i) = 2*costh2*gw2(i)*qv2*(-gv2(i)*qv2*th2dd-gw2(i)*qv2\*s
 1 *inh2*th1d*th2d-gv2(i)*qv2*th2dd+costh2*gw2(i)*qv2*th1dd+costh2
 2 *gw2(i)*qv2*th1d*th2d+gw2(i)*qv2*th1dd+costh2*gw2(i)
 3 *qv2*th1d*th2d-gw2(i)*qv2*th1dd+gw2(i)*qv2*th2dd+costh2*gw2(i)
 4 *qv2*th1d*th2d-gv2(i)*qv2*th1dd+costh2*gw2(i)*qv2*th2dd+costh2*gw2(i)
 5 *qv2*th1d*th2d+gw2(i)*qv2*th1dd+costh2*gw2(i)*th1dd+gw2(i)
```

return
end
```
element(2,i) = -2*gw2(i)*qw2*(-gw2(i)*qw2*th2dd-gw2(i)*qw2*sinth2*th1d+costh2*gw2(i)*qw2d*th1d-cos th2*gw2(i)*qw2d*sinth2)-2*l2sub(2)*gw2(i)*qw2d

10 continue

call integr( element, maxarray, numsegment, a2r, a2, integral )
c
***********************************************
c
Set variables.

e2 = 0.5*integral(1)*rho2e
e4 = (-2.0)*a2rcu*costh2*rho2r*sinth2*th1d*th2d/3.0
e5 = a2rcu*costh2*q2*rho2r*th2dd/3.0
e6 = 0.5*integral(2)*rho2e
e7 = -0.5*integral(3)*rho2e
e8 = integral(4)*q*rho2e
e9 = sm2*(-2*gw2a2*qv2*(-gw2a2*qv2*th2dd-gw2a2*qw2*sinth2*th1d*th2d-gw2a2*qv2d*th2d+costh2*gw2a2*qw2*th1dd)+2*a2*(a2*th2dd-cos th2*gw2a2*qw2*th1d*th2d-gw2a2*qw2*sinth2*th1dd-gw2a2*qw2d*sinth2*th1d)+gv2a2*qv2dd)-2*l2sub(2)*gw2a2*qw2d)/4.0
e10 = a2rcu*rho2r*th2dd/3.0
e11 = -sm2*((-2*l2sub(2)*gw2a2*qw2*sinth2*th1d*th2d-2*l2sub(6)*costh2*gw2a2*qv2d)-2*l2sub(2)*gw2a2*qv2d)
e12 = a2rcu*cos th2*rho2r*sinth2*th1d*th2d/3.0

c Calculate the torques with the variables just calculated.
torque1l2 = e5+e4+e3+e2
torque2l2 = e9+e8+e7+e6+e13+e12+e11+e10
c
***********************************************
c
Mission complete, see you...

c return
end

subroutine tlink3

c Program name: tlink3.f

c
This subroutine helps to setup and calculate the torques for

c link 3.

c
***********************************************
c Declare variables and include the common blocks.

```fortran
implicit real (a-z)

integer i, maxarray, numsegment

parameter ( maxarray = 7, numsegment = 100 )

dimension u2 (0:numsegment), u3 (0:numsegment), I3sub (30)
dimension gv2 (0:numsegment), gv2sq (0:numsegment),
*   gv2p (0:numsegment), gv2psq (0:numsegment),
*   gv2pp (0:numsegment), gv2ppsq (0:numsegment)
dimension gw2 (0:numsegment), gw2sq (0:numsegment),
*   gw2p (0:numsegment), gw2psq (0:numsegment),
*   gw2pp (0:numsegment), gw2ppsq (0:numsegment)
dimension gv3 (0:numsegment), gv3sq (0:numsegment),
*   gv3p (0:numsegment), gv3psq (0:numsegment),
*   gv3pp (0:numsegment), gv3ppsq (0:numsegment)
dimension gw3 (0:numsegment), gw3sq (0:numsegment),
*   gw3p (0:numsegment), gw3psq (0:numsegment),
*   gw3pp (0:numsegment), gw3ppsq (0:numsegment)
dimension element (maxarray,0:numsegment), integral (maxarray)

cомmon /constant/ g, pi, segment

cомmon /link2/
    a2r, a2rsq, a2rcu, rho2r, mass2r,
    a2e, rho2e, mass2e, sm2, lm2, eiv2, eiw2,
    u2, a2, a2sq, costh2, costhsq, cos2th2,
    sinh2, sinh2th2

cомmon /link3/
    u3, a3, a3cg, rho3, mass3, sm3, lm3, m3cg, eiv3,
    eiw3, psid, psidsq, psidsd, cospsi, cospsiq,
    sinpsi, sinpsisq,
    chi, chid, chidd, coschi, sinchi, th3dth2dd,
    omegasub1, omegasub2, omegasub3

cомmon /link2gs/
    gv2, gv2sq, gv2p, gv2psq, gv2pp, gv2ppsq,
    gv2a2, gv2a2sq, gv2a2p, gv2a2psq,
    gv2a2pp, gv2a2ppsq,
    gw2, gw2sq, gw2p, gw2psq, gw2pp, gw2ppsq,
    gw2a2, gw2a2sq, gw2a2p, gw2a2psq,
    gw2a2pp, gw2a2ppsq

cомmon /link3gs/
    gv3, gv3sq, gv3p, gv3psq, gv3pp, gv3ppsq,
    gv3a3, gv3a3sq, gv3a3p,
    gw3, gw3sq, gw3p, gw3psq, gw3pp, gw3ppsq,
    gw3a3, gw3a3sq, gw3a3p

cомmon /link2qs/
    qv2, qv2sq, qv2d, qv2dsq, qv2dd,
    qv2, qv2sq, qv2d, qv2dsq, qv2dd

cомmon /link3qs/
    qv3, qv3sq, qv3d, qv3dsq, qv3dd,
    qv3, qv3sq, qv3d, qv3dsq, qv3dd

cомmon /theta1/
    th1i, th1f, th1, th1d, th1dsq, th1dd

cомmon /theta2/
    th2i, th2f, th2, th2d, th2dsq, th2dd

cомmon /theta3/
    th3i, th3f, th3, th3d, th3dsq, th3dd

cомmon /torque13/ torque13, torque213, torque313

```

***

C Calculate the values of the integrals of link 3.

```fortran
do 10 i = 0, numsegment

I3sub(1) = cospsi * gv3 * gv3(i) + sinpsi * u3(i)
I3sub(2) = I3sub(1) ** 2.0
I3sub(3) = cospsi * u3(i) - sinpsi * gv3 * gv3(i)
I3sub(4) = cospsi * gv2a2p * u3(i) -
*   gv2a2p * gv3(i) * cospsi
I3sub(5) = gv2a2p * sinpsi * u3(i) +
*   cospsi * gv2a2p * gv3(i) * gv3
I3sub(6) = psid * sinpsi * u3(i) + sinpsi * gv3(i) * qv3d +
*   psid * cospsi * gv3(i) * qv3
I3sub(7) = psid * cospsi * u3(i) + cospsi * gv3(i) * qv3d -
*   psid * sinpsi * gv3(i) * qv3
I3sub(8) = sinpsi * u3(i) + cospsi * gv3(i) * qv3 + gv2a2 * qv2
```

\[ \text{l3sub}(9) = \cos\psi \cdot u_3(i) - qv_3 \cdot \sin\psi + a_2 \]
\[ \text{l3sub}(10) = \cos\theta_2 \cdot g_3(i) \cdot q_3 + \cos\theta_2 \cdot g_2 a_2 \cdot g_3 \]
\[ \text{l3sub}(11) = \sin\theta_2 \cdot g_3(i) \cdot q_3 + \sin\theta_2 \cdot g_2 a_2 \cdot g_3 \]
\[ \text{l3sub}(12) = \cos\chi \cdot u_3(i) \cdot \sin\chi - qv_3 \cdot g_3(i) \]
\[ \text{l3sub}(13) = g_2 a_2 \cdot \text{l3sub}(3) + g_2 a_2 \]
\[ \text{l3sub}(14) = \text{l3sub}(13) \cdot 2.0 \]
\[ \text{l3sub}(15) = -g_2 a_2 \cdot \theta_2 - \cos\psi \cdot g_2 a_2 \cdot g_3(i) \cdot \theta_3 \cdot qv_3 - 3.0 \]
\[ \text{l3sub}(16) = \cos\psi \cdot g_2 a_2 \cdot g_3(i) \cdot \theta_3 \cdot qv_3 - \text{l3sub}(3) \cdot \chi \]
\[ \text{l3sub}(17) = \cos\psi \cdot g_2 a_2 \cdot u_3(i) - g_2 a_2 \cdot \theta_3 \cdot qv_3 \]
\[ \text{l3sub}(18) = \cos\psi \cdot g_2 a_2 \cdot g_3(i) \cdot q_3 + \cos\psi \cdot g_2 a_2 \cdot g_3(i) \]
\[ \text{l3sub}(19) = \cos\psi \cdot g_2 a_2 \cdot g_3(i) \cdot \theta_3 \cdot qv_3 - \cos\psi \cdot g_2 a_2 \cdot g_3(i) \]
\[ \text{l3sub}(20) = \cos\psi \cdot g_2 a_2 \cdot g_3(i) \cdot \theta_3 \cdot qv_3 - \cos\psi \cdot g_2 a_2 \cdot g_3(i) \]
\[ \text{l3sub}(21) = \cos\psi \cdot g_2 a_2 \cdot g_3(i) \cdot \theta_3 \cdot qv_3 - \cos\psi \cdot g_2 a_2 \cdot g_3(i) \]
\[ \text{l3sub}(22) = \cos\psi \cdot g_2 a_2 \cdot g_3(i) \cdot \theta_3 \cdot qv_3 - \cos\psi \cdot g_2 a_2 \cdot g_3(i) \]
\[ \text{l3sub}(23) = \cos\psi \cdot g_2 a_2 \cdot g_3(i) \cdot \theta_3 \cdot qv_3 - \cos\psi \cdot g_2 a_2 \cdot g_3(i) \]
\[ \text{l3sub}(24) = \cos\psi \cdot g_2 a_2 \cdot g_3(i) \cdot \theta_3 \cdot qv_3 - \cos\psi \cdot g_2 a_2 \cdot g_3(i) \]
\[ \text{l3sub}(25) = \cos\psi \cdot g_2 a_2 \cdot g_3(i) \cdot \theta_3 \cdot qv_3 - \cos\psi \cdot g_2 a_2 \cdot g_3(i) \]
\[ \text{l3sub}(26) = \cos\psi \cdot g_2 a_2 \cdot g_3(i) \cdot \theta_3 \cdot qv_3 - \cos\psi \cdot g_2 a_2 \cdot g_3(i) \]
\[ \text{l3sub}(27) = \cos\psi \cdot g_2 a_2 \cdot g_3(i) \cdot \theta_3 \cdot qv_3 - \cos\psi \cdot g_2 a_2 \cdot g_3(i) \]
\[ \text{l3sub}(28) = \cos\psi \cdot g_2 a_2 \cdot g_3(i) \cdot \theta_3 \cdot qv_3 - \cos\psi \cdot g_2 a_2 \cdot g_3(i) \]
\[ \text{l3sub}(29) = \cos\psi \cdot g_2 a_2 \cdot g_3(i) \cdot \theta_3 \cdot qv_3 - \cos\psi \cdot g_2 a_2 \cdot g_3(i) \]
\[ \text{l3sub}(30) = \cos\psi \cdot g_2 a_2 \cdot g_3(i) \cdot \theta_3 \cdot qv_3 - \cos\psi \cdot g_2 a_2 \cdot g_3(i) \]
\[ \text{element}(1,i) = 2*\text{l3sub}(10)*(-g_2 a_2 \cdot qv_2 \cdot \theta_2 - g_2 a_2 \cdot g_2 a_2 \cdot qv_2 \cdot \sin\psi + g_3(i) \cdot \xi \cdot \omega_{sub1} \cdot qw_3 + g_3(i) \cdot \omega_{sub3} - \chi \cdot \omega_{sub1} \cdot qw_3 - \chi \cdot \omega_{sub3} \cdot \xi) \]
\[ \text{element}(2,i) = -2*\text{l3sub}(8)*(-g_2 a_2 \cdot qv_2 \cdot \theta_2 - g_2 a_2 \cdot g_2 a_2 \cdot qv_2 \cdot \sin\psi + g_3(i) \cdot \xi \cdot \omega_{sub1} \cdot qw_3 + g_3(i) \cdot \omega_{sub3} - \chi \cdot \omega_{sub1} \cdot qw_3 - \chi \cdot \omega_{sub3} \cdot \xi) \]
\[ \text{element}(3,i) = 2*(a_2 \cdot \sin\theta_2 \cdot \theta_2 \cdot \sin\theta_2 \cdot \theta_2 \cdot \sin\theta_2 \cdot \xi) \cdot \omega_{sub1} \cdot qw_3 + g_3(i) \cdot \omega_{sub3} - \chi \cdot \omega_{sub1} \cdot qw_3 - \chi \cdot \omega_{sub3} \cdot \xi) \]
\[ v3*\text{th1d}*\cos\theta2*g2a2*q2*\text{th1d}*(g2a2*q2*\sin\theta2*\text{th1d})*g3(1)*q2*3*\\] 
\[ 3*\sin\theta1d*\cos\theta1d*\sin\chi1d*\\text{th1d}*(-a2*\cos\theta2*\\text{th1d}+g2a2*g2*\text{th1d})*g3(1)*q2*3*\\] 
\[ 2*\text{th1d}+\cos\theta2*\sin\theta2*\\text{th1d}+2*\text{th2d}+\cos\theta2*\sin\theta2*\\text{th2d}+g2a2*g2*\text{th1d}+g2a2*g2*\text{th2d}+g3(1)*q2*3*\\] 
\[ 2*\text{th1d}+\cos\theta2*\sin\theta2*\\text{th1d}+2*\text{th2d}+\cos\theta2*\sin\theta2*\\text{th2d}+g2a2*g2*\text{th1d}+g2a2*g2*\text{th2d}+g3(1)*q2*3*\] 
\[ \text{element}(4, i) = -13*\text{sub}(26)\] 
\[ \text{element}(5, i) = -2*\text{th1d}-(\cos\chi1d+\cos\theta2)*\text{th1d}+g2a2*g2*\text{th1d}+g2a2*g2*\text{th2d}+g3(1)*q2*3*\] 
\[ 2*\text{th1d}+\cos\theta2*\sin\theta2*\\text{th1d}+2*\text{th2d}+\cos\theta2*\sin\theta2*\\text{th2d}+g2a2*g2*\text{th1d}+g2a2*g2*\text{th2d}+g3(1)*q2*3*\] 
\[ \text{element}(6, i) = 2*\text{u3}(i)*\sin\chi1d*\cos\theta1d+g3(1)*g2*\text{th1d}+g2a2*g2*\text{th1d}+g2a2*g2*\text{th2d}+g3(1)*q2*3*\] 
\[ \text{element}(7, i) = 13*\text{sub}(12)\] 

10 continue 

call integrat (element, maxarray, numsegment, 0, a3, integral ) 

c Set variables. 

e2 = 0.5*integral(1)*rh03 

e3 = s83*(2*\text{sub}(6))*(-g2a2*g2*\text{th2d}+g2a2*g2*\text{th1d}+\cos\theta2*\sin\chi1d*\\text{th1d}+\cos\theta2*\\text{th1d}+2*\text{th2d}+\cos\theta2*\sin\theta2*\\text{th1d}+2*\text{th2d}+g2a2*g2*\text{th1d}+g2a2*g2*\text{th2d}+g3(1)*q2*3*\] 
\[ 2*\text{th1d}+\cos\theta2*\sin\theta2*\\text{th1d}+2*\text{th2d}+\cos\theta2*\sin\theta2*\\text{th2d}+g2a2*g2*\text{th1d}+g2a2*g2*\text{th2d}+g3(1)*q2*3*\] 
\[ v3*\text{th1d}+\cos\theta2*g2a2*q2*\text{th1d}*(g2a2*q2*\sin\theta2*\\text{th1d}+g3(1)*q2*3*\] 
\[ e4 = 0.5*integral(2)*rh03 

e5 = -0.5*integral(3)*rh03 

e6 = integral(4)*rh03 

e7 = s83*(-2*\text{sub}(9))*(-g2a2*g2*\text{th2d}+g2a2*g2*\text{th1d}+\cos\theta2*\sin\chi1d*\\text{th1d}+\cos\theta2*\\text{th1d}+2*\text{th2d}+\cos\theta2*\sin\theta2*\\text{th1d}+2*\text{th2d}+g2a2*g2*\text{th1d}+g2a2*g2*\text{th2d}+g3(1)*q2*3*\] 
\[ 2*\text{th1d}+\cos\theta2*\sin\theta2*\\text{th1d}+2*\text{th2d}+\cos\theta2*\sin\theta2*\\text{th2d}+g2a2*g2*\text{th1d}+g2a2*g2*\text{th2d}+g3(1)*q2*3*\] 
\[ v3*\text{th1d}+\cos\theta2*g2a2*q2*\text{th1d}*(g2a2*q2*\sin\theta2*\\text{th1d}+g3(1)*q2*3*\] 
\[ d = -2*\text{sub}(6)*(-g2a2*g2*\text{th2d}+g2a2*g2*\text{th1d}+\cos\theta2*\sin\chi1d*\\text{th1d}+\cos\theta2*\\text{th1d}+2*\text{th2d}+\cos\theta2*\sin\theta2*\\text{th1d}+2*\text{th2d}+g2a2*g2*\text{th1d}+g2a2*g2*\text{th2d}+g3(1)*q2*3*\] 
\[ e8 = -s83*(-2*\text{sub}(22))*(-g2a2*g2*\text{th2d}+g2a2*g2*\text{th1d}+\cos\theta2*\sin\chi1d*\\text{th1d}+\cos\theta2*\\text{th1d}+2*\text{th2d}+\cos\theta2*\sin\theta2*\\text{th1d}+2*\text{th2d}+g2a2*g2*\text{th1d}+g2a2*g2*\text{th2d}+g3(1)*q2*3*\] 
\[ 2*\text{th1d}+\cos\theta2*\sin\theta2*\\text{th1d}+2*\text{th2d}+\cos\theta2*\sin\theta2*\\text{th2d}+g2a2*g2*\text{th1d}+g2a2*g2*\text{th2d}+g3(1)*q2*3*\] 
\[ v3*\text{th1d}+\cos\theta2*g2a2*q2*\text{th1d}*(g2a2*q2*\sin\theta2*\\text{th1d}+g3(1)*q2*3*\] 
\[ e9 = 0.5*integral(5)*rh03 

e10 = -0.5*integral(6)*rh03 

e11 = integral(7)*rh03 

Calculate the torques with the variables just calculated.

\[ \text{torque1}_{l3} = e_3 + e_2 \]
\[ \text{torque2}_{l3} = e_8 + e_7 + e_6 + e_5 + e_4 \]
\[ \text{torque3}_{l3} = e_9 + e_{13} + e_{12} + e_{11} + e_{10} \]

Mission complete, see you...

return
d
Set variables.

\[
\begin{align*}
\text{lcsub}(1) &= 2 \cdot \text{th3d} + \text{th2d} + 2 \cdot \text{gv2a2p} \cdot \text{qv2d} \\
\text{lcsub}(2) &= \text{th3d} + 2 \cdot \text{th2d} + \text{gv2a2p} \cdot \text{qv2d} \\
\text{lcsub}(3) &= \text{th3d} - \text{th2d} + \text{gv2a2p} \cdot \text{qv2d} \\
\text{lcsub}(4) &= 3 \cdot \text{a2} \cdot \text{acsq} - \text{gv2a2p} - 6 \cdot \text{acsq} \cdot \text{gw2a2} \\
\text{lcsub}(5) &= 2 \cdot \text{accu} = \text{gw2a2p} - 12 \cdot \text{a2} \cdot \text{acsq} + \text{gw2a2} \\
\text{lcsub}(6) &= 3 \cdot \text{a2sq} \cdot \text{ac} - 3 \cdot \text{ac} \cdot \text{gv2a2sq} - \text{qv2sq} \\
\text{lcsub}(7) &= 12 \cdot \text{ac} = \text{gv2a2sq} \cdot \text{qv2} - \text{qv2d} + 6 \cdot \text{ac} \cdot \text{gv2a2} - \text{qv2} + \text{qv2d} \\
\text{lcsub}(8) &= 6 \cdot \text{ac} \cdot \text{gv2a2} - \text{qv2} - \text{qv2d} \\
\text{lcsub}(9) &= -12 \cdot \text{a2} \cdot \text{ac} \cdot \text{gw2a2} \cdot \text{gw2a2} - \text{qv2sq} + \text{ac} + 3 \cdot \text{a2sq} \cdot \text{ac} \\
\text{lcsub}(10) &= 12 \cdot \text{a2} \cdot \text{ac} \cdot \text{cosh2} - \text{gv2a2} - \text{gw2a2} - \text{qv2} - \text{qv2d} \\
\text{lcsub}(11) &= -12 \cdot \text{a2} \cdot \text{ac} \cdot \text{cosh2} - \text{gw2a2} - \text{qv2} - \text{qv2d} \\
\text{lcsub}(12) &= -6 \cdot \text{acsq} = \text{gw2a2} - \text{qv2} - \text{qv2d} \\
\text{lcsub}(13) &= 12 \cdot \text{a2} \cdot \text{ac} \cdot \text{cosh2} - \text{gw2a2} - \text{qv2} - \text{qv2d} \\
\text{lcsub}(14) &= -6 \cdot \text{acsq} = \text{gw2a2} - \text{qv2} - \text{qv2d} \\
\text{lcsub}(15) &= \text{sin} (2 \cdot \text{zeta} + 2 \cdot \text{th3} + 2 \cdot \text{th2d} + 2 \cdot \text{gv2a2} + \text{qv2}) \\
\text{lcsub}(16) &= \text{cos} (2 \cdot \text{zeta} + 2 \cdot \text{th3} + 2 \cdot \text{th2d} + 2 \cdot \text{gv2a2} + \text{qv2}) \\
\text{lcsub}(17) &= \text{sin} (2 \cdot \text{zeta} + 2 \cdot \text{th3} + 2 \cdot \text{th2d} + 2 \cdot \text{gv2a2} + \text{qv2}) \\
\text{lcsub}(18) &= \text{cos} (2 \cdot \text{zeta} + 2 \cdot \text{th3} + 2 \cdot \text{th2d} + 2 \cdot \text{gv2a2} + \text{qv2}) \\
\text{lcsub}(19) &= \text{sin} (2 \cdot \text{zeta} + 2 \cdot \text{th3} + 2 \cdot \text{th2d} + 2 \cdot \text{gv2a2} + \text{qv2}) \\
\text{lcsub}(20) &= \text{cos} (2 \cdot \text{zeta} + 2 \cdot \text{th3} + 2 \cdot \text{th2d} + 2 \cdot \text{gv2a2} + \text{qv2}) \\
\text{lcsub}(21) &= \text{sin} (2 \cdot \text{zeta} + 2 \cdot \text{th3} + 2 \cdot \text{th2d} + 2 \cdot \text{gv2a2} + \text{qv2}) \\
\text{lcsub}(22) &= \text{cos} (2 \cdot \text{zeta} + 2 \cdot \text{th3} + 2 \cdot \text{th2d} + 2 \cdot \text{gv2a2} + \text{qv2}) \\
\text{lcsub}(23) &= \text{sin} (2 \cdot \text{zeta} + 2 \cdot \text{th3} + 2 \cdot \text{th2d} + 2 \cdot \text{gv2a2} + \text{qv2}) \\
\text{lcsub}(24) &= \text{cos} (2 \cdot \text{zeta} + 2 \cdot \text{th3} + 2 \cdot \text{th2d} + 2 \cdot \text{gv2a2} + \text{qv2}) \\
\text{lcsub}(25) &= \text{sin} (2 \cdot \text{zeta} + 2 \cdot \text{th3} + 2 \cdot \text{th2d} + 2 \cdot \text{gv2a2} + \text{qv2}) \\
\text{lcsub}(26) &= \text{cos} (2 \cdot \text{zeta} + 2 \cdot \text{th3} + 2 \cdot \text{th2d} + 2 \cdot \text{gv2a2} + \text{qv2}) \\
\text{lcsub}(27) &= \text{sin} (2 \cdot \text{zeta} + 2 \cdot \text{th3} + 2 \cdot \text{th2d} + 2 \cdot \text{gv2a2} + \text{qv2}) \\
\text{lcsub}(28) &= \text{cos} (2 \cdot \text{zeta} + 2 \cdot \text{th3} + 2 \cdot \text{th2d} + 2 \cdot \text{gv2a2} + \text{qv2}) 
\end{align*}
\]
c Calculate the torques with the variables just calculated.

torque1lc = e2/12.0
torque2lc = e5+e4+e3
torque3lc = e7+e6+e5

c Mission complete, see you...

return

d end

subroutine tlinkh3
This subroutine helps to setup and calculate the torques for hydraulic cylinder 3.

Declare variables and include the common blocks.

implicit real (a-z)

integer i, maxarray, numsegment

parameter (maxarray = 6, numsegment = 100)

dimension u2 (0:numsegment), uh3 (0:numsegment), lh3sub (25)
dimension gv2 (0:numsegment), gv2sq (0:numsegment),
  *  gv2p (0:numsegment), gv2psq (0:numsegment),
  *  gv2pp (0:numsegment), gv2ppsq (0:numsegment)
dimension gw2 (0:numsegment), gw2sq (0:numsegment),
  *  gw2p (0:numsegment), gw2psq (0:numsegment),
  *  gw2pp (0:numsegment), gw2ppsq (0:numsegment)
dimension gvh3 (0:numsegment), gvh3sq (0:numsegment),
  *  gvh3p (0:numsegment), gvh3psq (0:numsegment),
  *  gvh3pp (0:numsegment), gvh3ppsq (0:numsegment)
dimension gwh3 (0:numsegment), gwh3sq (0:numsegment),
  *  gwh3p (0:numsegment), gwh3psq (0:numsegment),
  *  gwh3pp (0:numsegment), gwh3ppsq (0:numsegment)
dimension element (maxarray,0:numsegment), integral (maxarray)

common /constant/ a, pi, segment
common /link2/ a2r, a2rsa, a2rcu, rho2r, mass2r,
  *  a2e, rho2e, mass2e, em2, mn2, eiv2, eiw2,
  *  u2, a2, a2sq, costh2, costh2sq, cos2th2,
  *  sinh2, sinh2sq
common /linkh3/ ah3s, ah3sq, ah3cu, rhoh3s, massh3s,
  *  ah3p, ah3pstart, ah3pstartsq, ah3pstartcu,
  *  ah3pend, ah3pendsq, ah3pendcu, rhoh3p, massh3p,
  *  ah3r, rhoh3r, massh3r, smh3, eih3,
  *  uh3, eih3, ah3cu, ah3sq, ah3f, ah3f2, ah3se,
  *  eta, coseta, sineta, thh3, costh3, costh3sq,
  *  sinhth3, sinhth3sq, lamda, coslamda, coslamdasq,
  *  sinlamda, sinlamdasq
common /link2gs/ gv2, gv2sq, gv2p, gv2psq, gv2pp, gv2ppsq,
  *  gv2psq2, gv2spsq2, gv2pppsq2,
  *  gw2, gw2sq, gw2p, gw2psq, gw2pp, gw2ppsq,
  *  gw2psq2, gw2spsq2, gw2pppsq2
common /linkh3gs/ gvh3, gvh3sq, gvh3p, gvh3psq, gvh3pp, gvh3ppsq,
  *  gvh3ah3, gvh3ah3sq, gvh3ah3cu, gvh3ah3p,
  *  gvh3ah3pp, gvh3ah3ppsq,
  *  gvh3, gvh3sq, gvh3p, gvh3psq, gvh3pp, gvh3ppsq,
  *  gwh3h3, gwh3h3sq, gwh3h3cu, gwh3h3p,
  *  gwh3h3pp, gwh3h3ppsq
common /link2qs/ qv2, qv2sq, qv2d, qv2dsq, qv2dd,
  *  qv2, qv2sq, qv2d, qv2dsq, qv2dd
common /linkh3qs/ qvh3, qvh3sq, qvh3cu, qvh3d, qvh3dsq, qvh3dd
common /diffh3/ da3t3, da3t3t3, da3t3t3sqv2, da3t3t3sqv2, da3t3t3sqv2,
  *  da3t3t3, dtet, dtett, detaq2v2, detaq3v2,
  *  detaq2v3, detaq2v2, detaq3v3, detaqv3, dth3t3t3t3, dth3t3t3t3
  *  dth3t3t3t3, dth3t3t3sqv2, dth3t3t3sqv2, dth3t3t3sqv2,
  *  dth3t3t3, dth3t3t3sqv2, dth3t3t3sqv2, dth3t3t3sqv2
common /theta1/ th1, th1f, th1d, th1dsq, th1dd
common /theta2/ th2i, th2f, th2d, th2dsq, th2dd
common /theta3/ th3i, th3f, th3d, th3dsq, th3dd
common /torqueh3/ torque1h3, torque2h3, torque3h3

*******************************************************************************
Calculate the values of the integrals of hydraulic cylinder 2.

do 10 i = 0, numsegment

lh3sub(1) = costh3 * qvh3 * qvh3(i) + sinthh3 * uh3(i)

lh3sub(2) = costh3 * uh3(i) - sinthh3 * qvh3 * gvh3(i)

lh3sub(3) = coseta * qvh3 * gvh3(i) + sineta * uh3(i)

lh3sub(4) = coseta * uh3(i) - sineta * qvh3 * gvh3(i)

lh3sub(5) = -gvh3(i) * qvh3d * sineta - detath3 * uh3(i) * sineta -
          * coseta * detath3 * gvh3(i) * qvh3

lh3sub(6) = -detath3 * qvh3(i) * qvh3d * sineta +
          * coseta * gvh3(i) * qvh3d + coseta * detath3 * uh3(i)

lh3sub(7) = gvh3(i) * qvh3d * sinthh3 +
          * dthh3t * uh3(i) * sinthh3 +
          * costh3 * dthh3t * gvh3(i) * qvh3

lh3sub(8) = gwa2a2p * qw2 * sinth2 * th1d * th2d * gwh3(i) /
          * gwh3ah3p

lh3sub(9) = gwa2a2p * qw2 * costh2 * th1dd * gwh3(i) / gwh3ah3p

lh3sub(10) = gwa2a2p * qw2dd * costh2 * th1d * gwh3(i) / gwh3ah3p

lh3sub(11) = gwa2a2p * qw2 * costh2 * th1d * th2d * gwh3(i) /
          * gwh3ah3p

lh3sub(12) = gwa2a2p * qw2 * sinth2 * th1dd * gwh3(i) / gwh3ah3p

lh3sub(13) = gwa2a2p * qw2 * sinth2 * th1d * gwh3(i) / gwh3ah3p

lh3sub(14) = gwa2a2p * qw2 * costh2 * th1d * gwh3(i) / gwh3ah3p

lh3sub(15) = gwa2a2p * qw2 * costh2 * th1d * gwh3(i) / gwh3ah3p

lh3sub(16) = gwa2a2p * qw2d * gwh3(i) / gwh3ah3p

lh3sub(17) = lh3sub(16) - lh3sub(2) * th1d

lh3sub(18) = dah3qyv2 * sineta + lh3sub(4) * dthh3qyv2

lh3sub(19) = coseta + dah3qyv2 - lh3sub(3) * dthh3qyv2

lh3sub(20) = -gvh3(i) * sineta - lh3sub(3) * dthh3qyv3

lh3sub(21) = coseta * gvh3(i) + lh3sub(4) * dthh3qyv3

lh3sub(22) = -gvh3(i) * qvh3d * sineta - lh3sub(3) * dthh3t +
          * coseta * dah3t + lh3sub(15)

lh3sub(23) = dah3t * sineta + coseta * gvh3(i) * qvh3d +
          * costh3 * dthh3t - lh3sub(14)

lh3sub(24) = -detath3 * gvh3(i) * qvh3d + sineta + dah3tt * sineta +
          * coseta * gvh3(i) + qvh3dd + lh3sub(4) * dthh3tt +
          * lh3sub(15) + lh3sub(12) - lh3sub(11)

lh3sub(25) = -gvh3(i) * qvh3d * sineta - dah3tt * detath3 * sineta -
          * coseta * detath3 * gvh3(i) * qvh3d -
          * lh3sub(3) * dthh3tt - lh3sub(6) * dthh3t +
          * coseta + dah3tt + lh3sub(10) + lh3sub(9) -
          * lh3sub(8)

element(1,i) = -2*lh3sub(22)*gwa2a2p*gwh3(i)*qw2*sinth2*th2d/gwh3ah
1 3p-2*lh3sub(23)*costh2*gwa2a2p*gwh3(i)*qw2*th2d/gwh3ah3p-2*lh3sub
2 ah3p-2*lh3sub(23)*costh2*gwa2a2p*gwh3(i)*qw2d*sinint2/gwh3ah3p-2*lh3sub
4 (24)*gwa2a2p*gwh3(1)*qw2*sinint2/gwh3ah3p2*lh3sub(22)*costh2*gwa2
5 a2p*gwh3(i)*qw2d/gwh3ah3p+2*lh3sub(25)*costh2*gwa2a2p*gwh3(i)*qw2
6 2/gwh3ah3p+2*lh3sub(7)*lh3sub(17)
element(2,i) = 2*lh3sub(4)*lh3sub(23)*dthh3tth2-2*lh3sub(3)*lh3sub
1 (22)*dthh3tth2-2*lh3sub(3)*lh3sub(25)*dthh3tth2-2*lh3sub(4)*lh3s
2 *sub(22)*dthh3tth2-2*lh3sub(5)*lh3sub(25)*dthh3tth2-2*lh3sub(6)*lh3
3 sub(22)*dthh3tth2
element(3,i) = 2*lh3sub(1)*lh3sub(17)*th1d2*lh3sub(23)*(lh3sub(4)
1 *dthh3tth2-lh3sub(15)+2*lh3sub(22)*(-lh3sub(3)*dthh3tth2-lh3s
2 b(14)
element(4,i) = lh3sub(2)
element(5,i) = 2*lh3sub(1)*lh3sub(17)*dethath3*th1d2*lh3sub(23)*(d-
01 dethath3*gvh3(i)*qvh3d*sineta+dah3tt3*sineta+lh3sub(4)*dthh3tth
2 3-lh3sub(3)*dethath3*dthh3t+dethath3dethath3)*2*lh3sub(22)*(d-
3 -dah3tt*dethath3*dethath3dethath3)*2*lh3sub(22)*dethath3t
4 *dethath3*dthh3t+dethath3*lh3sub(6)*dethath3*dethath3dethath3
element(6,i) = lh3sub(23)*dethath3

continue

call integrat ( element, maxarray, numsegment, ah3s, ah3,
          * integral )
Set variables.

e2 = 0.5*rhoh3r*(dah3t*(-2*lh3sub(23)*gw2a2p*gwh3ah3*qw2*sinth2/gw1*h3ah3p+2*Ih3sub(22)*costh2*gw2a2p*guh3ah3*qw2/gwh3ah3p-2*Ih3sub(2))*lh3sub(17))+integral(1))
e3 = lh3sub(28)*costh3sq*dah3t*th1d/3.0
e4 = (2,0)*lh3sub(30)*costh3*dthh3t*sinthh3*th1d/3.0
e5 = lh3sub(30)*costh3sq*th1d/3.0
e6 = 0.5*(dah3t*(2*lh3sub(24)*lh3sub(23)*dthh3th2-2*lh3sub(23)*lh3sub(3))*lh3su1*b(22)*dthh3th2)+integral(2))*rhoh3r
e7 = -0.5*integral(3)*rhoh3r
e8 = integral(4)*g*rhoh3r
e9 = lh3sub(30)*dthh3th2*dthh3tt/3.0
e10 = -lh3sub(30)*dthh3th2*dthh3tt/3.0+costh3*sinthh3*th1dsq)/6.
e11 = lh3sub(30)*dthh3th2*dthh3tt/3.0
e12 = lh3sub(28)*dah3t*dthh3th2/3.0
e13 = lh3sub(29)*costh3sq*th1d/2.0
e14 = -0.5*((lh3sub(23)**2+lh3sub(22)**2)*lh3sub(17)**2)*dah3t*th1d/2+integral(5))*rhoh3r
e15 = (lh3sub(1)*dah3t*integral(6))*g*rhoh3r
e16 = -lh3sub(30)*dthh3th2*costh3*dthh3th2*costh3*detath3*sinthh3*th1d/2+integral(2))/6.0
e17 = -lh3sub(28)*dah3t*costh3sq*th1d/2+integral(2))/2.0
e18 = g*(-2*ah3pend*dah3th3*rhoh3r-2*ah3pend*dah3th3*rhoh3r-2*ah3pend*dah3th3*rhoh3r-2*ah3pend*dah3th3*rhoh3r)*sinthh3/2.0
e19 = lh3sub(29)*costh3-detath3sq*th1d/2.0
e20 = dthh3th2=3*ei3h3*gw2a2p*gwh3ah3psq*gw2sq/gwh3ah3psqgvh3ah3p*psq*gw3ah3p/2.0

Calculate the torques with the variables just calculated.

torque1lh3 = e5+e4+e3+e2
torque2lh3 = e9+e8+e7+e6+e10+e11+e12
torque3lh3 = e20+e19+e18+e17+e16+e15+e14

Mission complete, see you...

return
der

subroutine torqueO

Program name: Torque0.f

This subroutine helps to set the torque of the linkage to zero.

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

Declare variables and include the common blocks.

implicit real (a-z)

common /torque1l1/ torque1l1, torque2l1, torque3l1
common /torque1l2/ torque1l2, torque2l2, torque3l2
common /torque1l3/ torque1l3, torque2l3, torque3l3
common /torquelc/ torque1lc, torque2lc, torque3lc
common /torquehl3/ torque1hl3, torque2hl3, torque3hl3

***************************************************************************
Subroutine torqtot

Program name: torqtot.f

This subroutine helps to add the torques of all links together and store the values into an array.

Declare variables and include the common blocks.

Implicit real (a-z)

dimension torque (3)

common /torque1l1/ torque1l1, torque2l1, torque3l1
common /torque1l2/ torque1l2, torque2l2, torque3l2
common /torque1l3/ torque1l3, torque2l3, torque3l3
common /torque1lc/ torque1lc, torque2lc, torque3lc
common /torque1lh3/ torque1lh3, torque2lh3, torque3lh3
common /torques/ torque

c Set zero torque for link 1.
torque1l1 = 0.0
torque2l1 = 0.0
torque3l1 = 0.0

Set zero torque for link 2.
torque1l2 = 0.0
torque2l2 = 0.0
torque3l2 = 0.0

Set zero torque for link 3.
torque1l3 = 0.0
torque2l3 = 0.0
torque3l3 = 0.0

Set zero torque for link c.
torque1lc = 0.0
torque2lc = 0.0
torque3lc = 0.0

Set zero torque for link h3.
torque1lh3 = 0.0
torque2lh3 = 0.0
torque3lh3 = 0.0

Mission complete, see you...

return
end
c Add the torques together.

torque(1) = torque1h3+torque1lc+torque1l3+torque1l2+torque1l1

torque(2) = torque2h3+torque2lc+torque2l3+torque2l2+torque2l1

torque(3) = torque3h3+torque3lc+torque3l3+torque3l2+torque3l1

c

c Time to go, bye...

return

end
REFERENCES


