SANDIA REPORT

SAND86—2972 • UC—13
Unlimited Release
Printed November 1987

Kinematic and Dynamic Analyses of the Stanford/JPL Robot Hand

Victor J. Johnson, Gregory P. Starr

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

When printing a copy of any digitized SAND Report, you are required to update the markings to current standards.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof or any of their contractors or subcontractors.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes
Printed copy: A04
Microfiche copy: A01

Distribution
Category UC-13

SAND 86-2972
Unlimited Release
Printed November 1987

KINEMATIC AND DYNAMIC ANALYSES OF THE STANFORD/JPL ROBOT HAND

Victor J. Johnson
Switching Devices Division
Sandia National Laboratories
Albuquerque, NM 87185

Gregory P. Starr, Professor
Department of Mechanical Engineering
University of New Mexico
Albuquerque, NM 87131

Abstract

This report develops the kinematic and dynamic equations for one finger of the three-fingered Stanford/JPL robot hand and documents the physical parameters needed to implement the equations. The equations can be used in control schemes for position and force control of the Stanford/JPL robot hand.

Acknowledgments

This work was supported by Technology Based Research and Development funds. A debt of gratitude is owed Dr. J. Kenneth Salisbury, Massachusetts Institute of Technology, for permission to reprint from his papers the physical parameters of the Stanford/JPL robot hand. Debts are also owed to Jon H. Barnette, Switching Devices Division, Gary N. Beeler, Electromechanical Components Department, and Patrick J. Eicker, Intelligent Computing Department for their continuing support of the Sandia dexterous hand project.

The authors are also indebted to Mark A. Powell and Clifford S. Loucks, 1414 for reviewing and editing this report.

Contents

Introduction 7
1.0 Homogeneous Transformations and Frame Assignments 11
1.1 Coordinate Frame Conventions and Link Parameters .. 11
1.2 Finger Coordinate Frames 13
2.0 Forward Kinematics 17
3.0 Inverse Kinematics 18
4.0 Iterative Newton-Euler Dynamics 22
4.1 Velocity Equations 23
4.2 Static Forces 25
4.3 Iterative Newton-Euler Dynamics Equations 29
4.3.1 Acceleration "Propagation" 29
4.3.2 Force "Propagation" 31
4.3.3 Iterative Algorithm Application 34
APPENDIX A - Physical Parameters of the Stanford/JPL Hand . 40
APPENDIX B - MACSYMA Output 44
References 60

Figures

1. Stanford/JPL Hand..... 8
2. Denavit-Hartenberg Parameters..... 12
3. Finger-Frame Attachments..... 13
4. Flex and Hyperextension of Joint 3..... 21
5. Free-body Diagram of One Link..... 26
6. Free-body Diagram of a Generalized Link..... 33

Tables

1. Denavit-Hartenberg Parameters..... 14

KINEMATIC AND DYNAMIC ANALYSES OF THE STANFORD/JPL ROBOT HAND

Introduction

A robot manipulator must be controlled before it can be used to perform useful tasks. This report documents the analyses used to derive the kinematic and dynamic equations describing motion of a Stanford/JPL robot hand and also documents the physical parameters needed to apply the developed equations. The latter information can be found in Salisbury [1,2], but it is republished here for completeness, and to provide a basis for the software developed to control the hand.

The Stanford/JPL dexterous hand, designed by Dr. J. Kenneth Salisbury Jr., is a three-fingered robotic hand containing nine joints (Figure 1). Each finger is identical; three revolute joints, with joint 1 rotating perpendicularly to joints 2 and 3. The three fingers are positioned as two fingers and an opposing thumb. Thus, an analysis of one finger describes each of the three fingers. Only one finger is analyzed in this report. The results can be extended to all three.

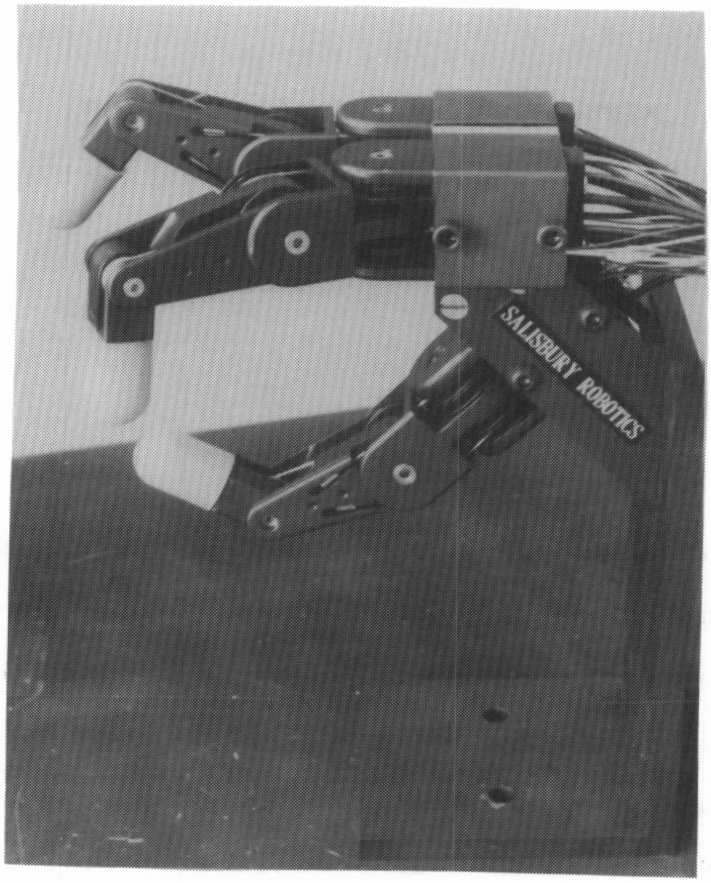

Figure 1. Stanford/JPL hand

Successful completion of any task that the hand is commanded to perform is dependent on the ability to accurately control the fingertip position. This requires development of relationships between defined points in space and the angular joint positions. The equations that give the finger positions, based on a set of known joint angles, are called "forward kinematics". The equations that give the finger joint angles, based on knowledge of the cartesian location of the tip of the manipulator, are called the "inverse kinematic" equations.

Kinematics is the study of motion and the interrelationships between acceleration, velocity, and position; the forces that cause the motions are not considered. However, motion is not the only concern for accurate position control; also important are the forces or torques required to impart the motion. The dynamic equations are derived in this report using the Newton-Euler technique because it yields the manipulator Jacobian in the derivation process. The Jacobian is a matrix that relates fingertip cartesian velocity to joint velocities and can also relate fingertip forces to joint torques.

Dynamic equations can be used to close a control loop internal to the lowest level position controller. Ideally, this technique decouples the inertial and positional nonlinearities of the system. However, the friction in the Stanford/JPL hand is a major contributer to the driver load, which negates the efforts of nonlinearity decoupling unless the friction loading can be mathematically modeled. This feat has not yet been satisfactorily accomplished. Another problem that needs addressing for successful implementation of this control scheme is development of a method that quickly performs the computations required to update the feedback model.

Due to the difficulties, the hand is currently controlled by keeping the forward-path gains low enough to avoid exciting the

higher order frequencies of the system. This technique produces a system with a response that is quick enough to use in all our current research studies. Description of the dynamic equation derivation is the goal of this report. Solution to the problems of using the equations is not presented.

The first section of this report introduces briefly the homogeneous transformations and frame assignments used to simplify the analyses. The conventions that were used are described in Craig [3]. In the second section the development of the forward kinematics for one finger is covered. Section three develops the inverse kinematic equations. Section four explains how the velocity, acceleration, and Newton-Euler dynamic equations are derived. The references used to develop the kinematics and dynamics were Craig [3], and Asada and Slotine [4]. Appendix A describes the physical parameters of the drive system of the Stanford/JPL hand. These parameters are needed to transform the equations developed in sections two and three into forms that can be implemented in the control system software. These parameters convert the equations derived in "manipulator" space to "tendon" and "motor" space. Major portions of the

information contained in Appendix A are a reprint of material from Salisbury [1]. Appendix B is the output file from the MACSYMA1* * symbolic manipulation program that defines the dynamic equations of the fingers.

1.0 Homogeneous Transformations and Frame Assignments

Standard practice in the field of robotics is to use homogeneous transformations to describe the position and orientation of objects and manipulator links relative to each other. The following two subsections describe how this convention has been applied to the Stanford/JPL robot hand.

1.1 Coordinate Frame Conventions and Link Parameters

In addition to manipulator position and orientation, descriptions of position and orientation of other objects may be necessary to perform useful tasks with a robot. Therefore, standardized conventions are desirable for representing positions and orientations in space. The conventions used here are those detailed in Craig [3].

1*MACSYMA is licensed by the Digital Equipment Corporation

Link parameters are defined using the Denavit-Hartenberg notation. This allows unique definition of the relationship between two joints and two links with four parameters: (1) the link length, (a), (2) the link twist, (alpha), (3) the joint offset, (d), and (4) the joint rotation, (theta). See Figure 2.

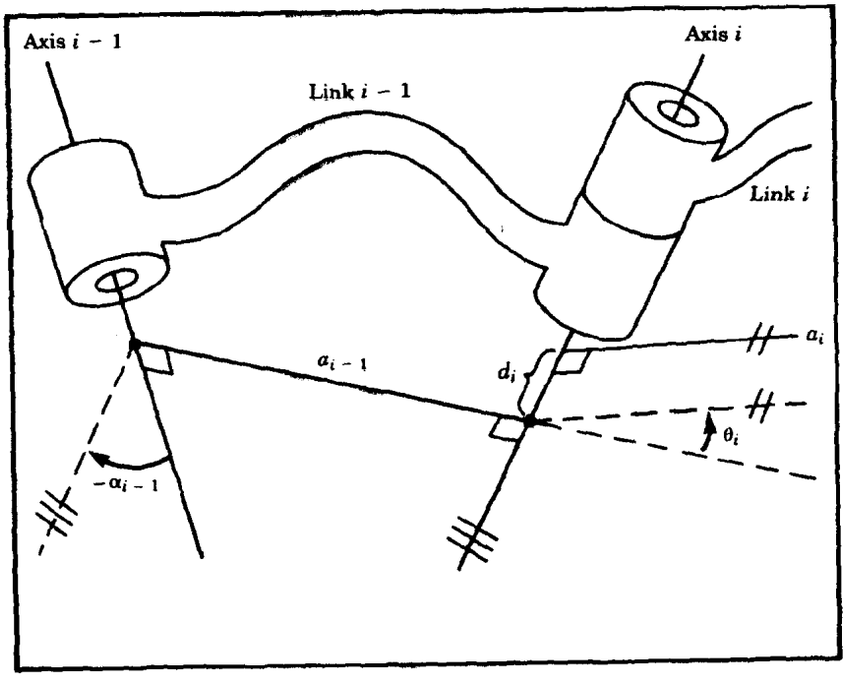

Figure 2. Denavit-Hartenberg parameters from Craig [3].

1.2 Finger Coordinate Frames

The frame attachments used (Figure 3) follow the conventions outlined in Craig. Frame one is placed with z_1 along its axis of rotation (the joint axis), with the x_1 axis pointing in the direction of joint two. Frame two is placed with z_2 along the axis of rotation of joint two, and x_2 points in the direction of joint three. Now, frame 0 is placed coincident with frame 1 where theta1 is zero, and the last frame (frame 3) is placed such that the x_3 axis aligns with x_2 axis and d_3 is zero. The z_3 axis is placed along the joint three axis.

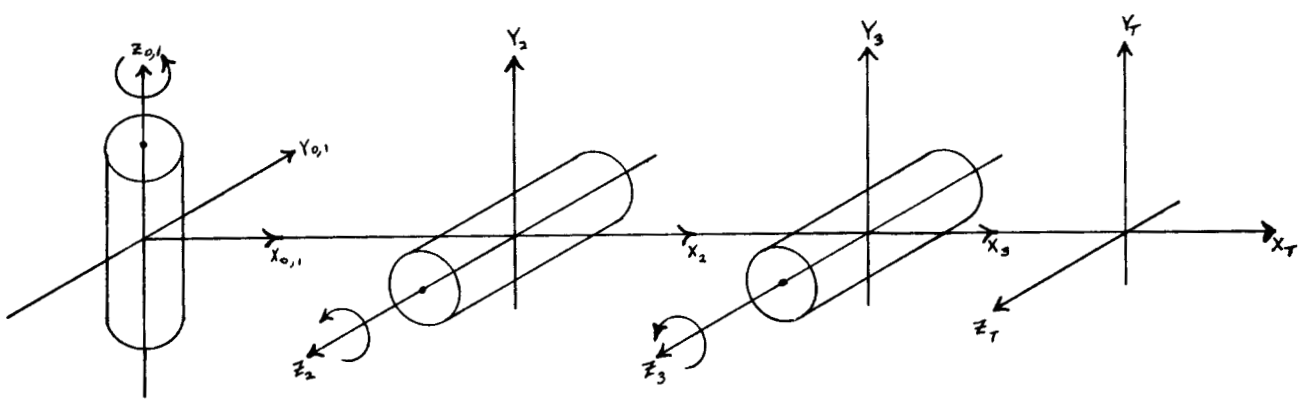

Figure 3. Finger-frame attachments

An additional frame, called a tool frame, is conventionally placed at some arbitrary point on the end of the last link. The

tool frame is placed at the center of the finger's tip with the same axis orientation as frame 3.

The Denavit-Hartenberg parameters are shown in Table 1.

Table 1. Denavit-Hartenberg parameters

i	$alpha_{i-1}$	a_{i-1}	d_i	$theta_i$
1	0	0	0	$theta_1$
2	90	L_1	0	$theta_2$
3	0	L_2	0	$theta_3$
t	0	L_3	0	

The transformation relating a position in frame i to the same position described in terms of frame i-1 is described using the Denavit-Hartenberg parameters in the following equation:

$$
{}^{i-1}T_i = \begin{bmatrix} c(th_i) & -s(th_i) & 0 & a_{i-1} \\ s(th_i)c(al_{i-1}) & c(th_i)c(al_{i-1}) & -s(al_{i-1}) & -d_i s(al_{i-1}) \\ s(th_i)s(al_{i-1}) & c(th_i)s(al_{i-1}) & c(al_{i-1}) & d_i c(al_{i-1}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)
$$

where ${}^{i-1}T_i$ = the transformation relating a position and orientation in frame i to the same position and orientation described in terms of frame i-1.

$s(th_i)$ = the sine of the link i joint angle ($theta_i$).

$c(th_i)$ = the cosine of the link i joint angle ($theta_i$).

$s(al_{i-1})$ = the sine of the link i-1 twist angle ($alpha_{i-1}$).

$c(al_{i-1})$ = the cosine of the link i-1 twist angle ($alpha_{i-1}$).

a_{i-1} = the length of link i-1.

d_i = the link offset of link i.

Using Equation (1) and the Denavit-Hartenberg parameters, the transformations to needed to describe the finger motions are those shown in Equations (2) through (5).

$$ {}^{0}T_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2) $$

$$
{}^{1}T_{2} = \begin{bmatrix} c_2 & -s_2 & 0 & L_1 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}
$$

$$
{}^{2}T_{3} = \begin{bmatrix} c_3 & -s_3 & 0 & L_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}
$$

$$
{}^{3}T_{t} = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}
$$

where s_i is the sine of $theta_i$.

c_i is the cosine of $theta_i$.

L_i is the length of link i.

These transformations are all that are necessary to find the forward kinematics of a finger of the Stanford/JPL hand.

2.0 Forward Kinematics

The forward kinematic equations define the position and orientation of a fingertip when the joint rotation angles are known. That relationship is given by the transformation ${}^{0}T_{t}$, which is found by multiplying equations (2) through (5) together. Equation (6) is the forward kinematic equation, ${}^{0}T_{t}$.

$$
{}^{0}T_{t} = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & L_1c_1 + L_2c_1c_2 + L_3c_1c_{23} \\ s_1c_{23} & -s_1s_{23} & -c_1 & L_1s_1 + L_2s_1c_2 + L_3s_1c_{23} \\ s_{23} & c_{23} & 0 & L_2s_2 + L_3s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6)
$$

In Equation (6), c_{23} is defined as cosine($theta_2$ + $theta_3$) and similarly, s_{23} is sine($theta_2$ + $theta_3$). The position and orientation vectors of the manipulator are determined for any given set of joint angles by putting the joint angles into matrix Equation (6). The upper left 3x3 sub-matrix is the orientation matrix, and describes the tool-frame unit vectors in terms of the base-frame vectors. The first three entries in the last column are the components of the position vector: p_x, p_y, and p_z.

3.0 Inverse Kinematics

The inverse kinematic problem is, as would be expected, the inverse of the forward kinematics problem. Given an (x,y,z) positional value and a desired orientation, the joint angle values necessary to place the finger in that position and orientation must be determined. The derivation of the inverse kinematic equations requires solving the forward kinematic equations for joint angles. The equations are very nonlinear in terms of the joint angles, making solution difficult. For some manipulators, in fact, there is no closed-form solution.

For the three-degree-of-freedom finger there is a solution. However, as the finger has only three degrees of freedom, either position, orientation, or combination of three positional coordinates. To grasp an object, the position of the point of contact of the fingers is required. Thus, the joint angles needed to place the fingertip in the desired position are the parameters of interest.

The three equations needed for the solution are the p_x, p_y, and p_z equations from the ${}^0T_t$ transform: Equation (6). They are

reprinted here for easy reference as Equations (7), (8), and (9).

$$p_x = L_1c_1 + L_2c_1c_2 + L_3c_1c_{23} \tag{7}$$

$$p_y = L_1s_1 + L_2s_1c_2 + L_3s_1c_{23} \tag{8}$$

$$p_z = L_2s_2 + L_3s_{23} \tag{9}$$

To find the solution for $theta_1$, multiply Equation (7) by s_1, Equation (8) by c_1, and divide the results. The result is equation (10).

$$s_1 / c_1 = p_y / p_x \tag{10}$$

Equation (10) can be solved for $theta_1$ by using a two-sided arctangent function, as in Equation (11).

$$theta_1 = ATAN2(p_y,p_x) \tag{11}$$

Now, multiplying Equation (7) by c_1 and Equation (8) by s_1 and adding the two results gives Equation (12).

$$p_xc_1 + p_ys_1 = L_3c_{23} + L_1 + L_2c_2 \tag{12}$$

Since $p_x c_1 + p_y s_1$ is equal to $(p_x^2 + p_y^2)^{0.5}$, Equation (12) can be rewritten to obtain Equation (13).

$$a_1 = (p_x^2 + p_y^2)^{0.5} - L_1 = L_3 c_{23} + L_2 c_2 \tag{13}$$

Squaring equations (9) and (13), adding, and rearranging the results gives equation (14).

$$c_3 = (a_1^2 + p_z^2 - L_3^2 - L_2^2) \ / \ (2L_2L_3) \tag{14}$$

The cosine function is an even function [f(x) = f(-x)], and is periodic; therefore, there are four unique solutions for $theta_3$. See Equation (15). Because of physical constraints, two solutions are attainable. These two solutions correspond to flexing the finger or hyperextending it: $theta_3$ negative, or $theta_3$ positive (Figure 4).

$$theta_3 = ACOS[(a_1^2 + p_z^2 - L_3^2 - L_2^2) \ / \ (2L_2L_3)] \tag{15}$$

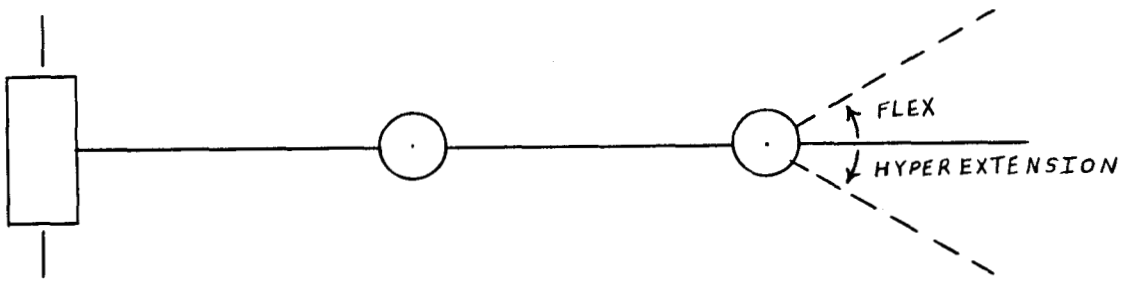

Figure 4. Flex and hyperextension of joint 3.

Now, by multiplying Equation (13) by c_2, Equation (9) by s_2, and adding equation (16) results in:

$$a_1c_2 + p_zs_2 = L_3c_3 + L_2 \tag{16}$$

Equation (16) can be rearranged to obtain a form used to find $theta_2$:

$$-(L_3c_3 + L_2) = -p_zs_2 + (-a_1c_2) \tag{17}$$

Let $p_z = Rcos(psi)$, $-a_1 = Rsin(psi)$, $W = (L_3c_3 + L_2)$, $R = (p_z^2 + a_1^2)^{0.5}$, and $psi = ATAN2(-a_1,p_z)$. Substituting these expressions

into equation (17), equation (18) results:

$$\text{theta}_2 = \text{ATAN2}(-a_1, p_z) - \text{ATAN2}[(L_3 c_3 + L_2), \pm(a_1^{\,2} + p_z^{\,2} - W^2)^{0.5}] \tag{18}$$

Once again, there are multiple solutions. This time there are two solutions but one is not attainable because of physical constraints. Thus, the solution for theta_2 is the one resulting from a positive value in the second argument of the second arctangent function call.

4.0 Iterative Newton-Euler Dynamics

The equations necessary to find the position and orientation of a robot manipulator were developed in the previous sections. This allows the next step, development of the equations of motion to be attacked. The velocity and static force equations are derived in subsections 4.1 and 4.2. The concepts discussed in those two subsections will be extended to an acceleration and force-of-motion analysis in subsection 4.3.

4.1 Velocity Equations

For any open kinematic chain, such as a robot manipulator, the velocity of each successive link can be calculated by "propagating" the velocity of the previous link; Craig [3]. The velocity of each link has a linear and a rotational component, equations (19) and (20). However, care must be taken to describe the components of velocity in the proper reference frame.

$$ {}^{i+1}o_{i+1} = {}^{i+1}R_i \, {}^{i}o_i + td_{i+1} \, {}^{i+1}Z_{i+1} \tag{19} $$

where ${}^{i+1}o_{i+1}$ = the angular velocity of link i+1 written in terms of the i+1 reference frame.

${}^{i+1}R_i$ = the rotational matrix relating frames i+1 and i and is the upper left three by three matrix of ${}^{i+1}T_i$.

td_{i+1} = the time derivative of $theta_{i+1}$.

${}^{i+1}Z_{i+1}$ = the vector describing the axis of rotation of joint i+1. It is usually the vector $[0\ 0\ 1]^T$ where ^{T} signifies the transpose.

Notice that the velocity vectors are free vectors. Therefore, a positional transformation to change descriptions from one

reference frame to the next is not required. Only ${}^{i+1}R_i$ is needed to change the reference frame description, and not ${}^{i+1}T_i$.

$$ {}^{i+1}v_{i+1} = {}^{i+1}R_i\left({}^{i}v_i + {}^{i}o_i x {}^{i}P_{i+1}\right) \tag{20} $$

where ${}^{i+1}v_{i+1}$ = the linear velocity of the origin of link frame i+1 relative to reference frame i+1.

${}^{i}v_i$ = the linear velocity of the origin of link frame i relative to reference frame i.

${}^{i}o_i x {}^{i}P_{i+1}$ = the cross product of the angular velocity of link i and the position of the origin of frame i+1 referenced to frame i.

Applying Equations (19) and (20) to each sequential link of the robot finger, the velocity equation describing the motion of the tip in terms of the tip reference frame can be determined. See Equations (21) and (22).

$$ {}^{t}o_t = {}^{t}R_3\,{}^{3}o_3 + td_t Z_t = \begin{bmatrix} s_{23}td_1 \\ c_{23}td_1 \\ td_2+td_3 \end{bmatrix} \tag{21} $$

$$ {}^{t}v_t = {}^{t}R_3({}^{3}v_3 + {}^{3}o_3 x {}^{2}P_3) = \begin{bmatrix} L_2 s_3 td_2 \\ L_2 c_3 td_2 + L_3(td_2+td_3) \\ -(L_1+L_2c_2)td_1 - L_3c_{23}td_1 \end{bmatrix} \tag{22} $$

Equation (22) can be rewritten as shown in Equation (23) to illustrate the relationship between the tip velocity and the joint velocities. The 3x3 matrix on the right-hand side of equation (23)is defined as the Jacobian matrix.

$$ {}^{t}v_t = \begin{bmatrix} 0 & L_2 s_3 & 0 \\ 0 & L_2 c_3 + L_3 & L_3 \\ -(L_3c_{23} + L_1 + L_2c_2) & 0 & 0 \end{bmatrix} \begin{bmatrix} td_1 \\ td_2 \\ td_3 \end{bmatrix} \tag{23} $$

The tip velocity is found in terms of the stationary base frame by premultiplying Equation (23) by ${}^{0}R_t$.

4.2 Static Forces

Analyzing the static forces acting on a manipulator is not unlike analyzing the static forces on any structure. The structure of a manipulator suggests performing a force-moment balance on each link in terms of the locally defined link frames. Then, the joint torques required to maintain equilibrium can be determined. Figure 5 shows a generalized link with the forces acting on the

link. Equation (24) results from a force balance procedure:

$$ {}^{i}f_{i} - {}^{i}f_{i+1} = 0 \tag{24} $$

where ${}^{i}f_{i}$ = the force exerted on link i by link i-1 referenced to frame i.

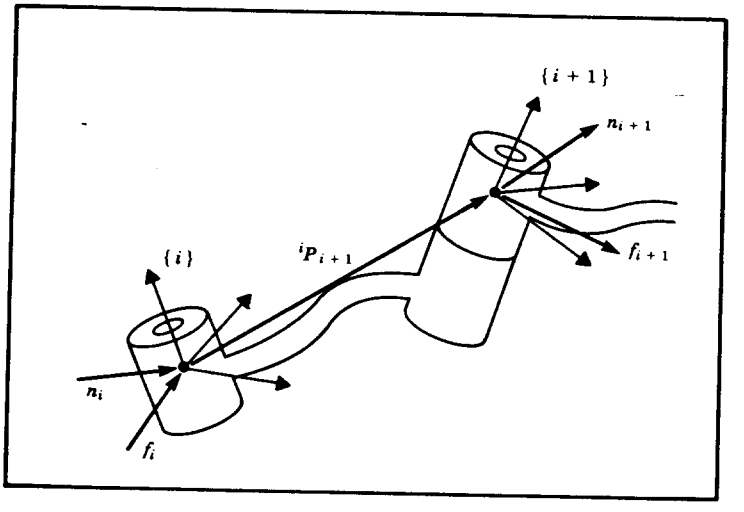

Figure 5. Free-body diagram of one link [from Craig].

A torque summation relative to frame i, gives Equation (25).

$$ {}^{i}n_{i} - {}^{i}n_{i+1} - {}^{i}P_{i+1} x {}^{i}f_{i+1} = 0 \tag{25} $$

where ${}^{i}n_{i}$ = the torque exerted on link i by link i-1 referenced to frame i.

${}^{i}P_{i+1} x {}^{i}f_{i+1}$ = the cross product of the position vector of the origin of frame i+1 and the force exerted on link i by link i-1, both referenced to frame i.

If Equations (24) and (25) are rewritten to find ${}^{i}f_{i}$ and ${}^{i}n_{i}$, and using the known values of ${}^{i+1}f_{i+1}$ and ${}^{i+1}n_{i+1}$ and the appropriate rotation matrices, Equations (26) and (27) are obtained.

$$ {}^{i}f_{i} = {}^{i}R_{i+1} \, {}^{i+1}f_{i+1} \tag{26} $$

$$ {}^{i}n_{i} = {}^{i}R_{i+1} \, {}^{i+1}n_{i+1} + {}^{i}P_{i+1} x {}^{i}f_{i} \tag{27} $$

The form of these equations allows "propagation" of the forces from the end link back to the base link. All components of the force and moment vectors are balanced by the structure of the mechanism, with the exception of the torque about the joint axis. Thus, the joint torque required for static equilibrium is equal to the component of the torque on link i directed along the joint axis (the dot product of the joint axis vector with the moment vector). Equation (28) gives the joint torque.

$$ Tau_{i} = {}^{i}n_{i} \, {}^{T}i_{Z_{i}} \tag{28} $$

Using Equations (26), (27), and (29) on the three-link finger results in a torque equation of the form:

$$\text{Tau} = \begin{bmatrix} 0 & 0 & -(L_3c_{23}+L_2c_2+L_1) \\ L_2s_3 & L_3+L_2c_2 & 0 \\ 0 & L_3 & 0 \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{29}$$

where f_x, f_y, and f_z = the x, y, and z direction of a force applied to the tip of a finger.

The 3x3 matrix on the right-hand side of equation (29) is the transposition of the Jacobian. The remarkable fact about the Jacobian is that it relates the cartesian forces and velocities to joint torques and angular velocities without performing an inverse transformation.

4.3 Iterative Newton-Euler Dynamics Equations

Derivation of the dynamic equations of a manipulator can be accomplished using many different techniques. Each technique has both advantages and disadvantages. The technique chosen for this report is the iterative Newton-Euler method; it was selected because it is an intuitive approach for engineers with a mechanical background.

4.3.1 Acceleration "Propagation"

Before calculating the inertial forces on each link, the rotational velocity, and the linear and rotational accelerations of the mass center of each link must be found. In subsection 4.1 the rotational velocity equation, that propagates velocities from the base link to the last link, was presented (Equation (19)). Now, a similar technique will be used to propagate linear and rotational accelerations. Equation (30) is a reprint of equation (19), whereas Equations (31) and (32) are the rotational and linear acceleration equations, respectively.

$$ {}^{i+1}o_{i+1} = {}^{i+1}R_i \, {}^{i}o_i + td_{i+1} \, {}^{i+1}Z_{i+1} \tag{30} $$

$$^{i+1}od_{i+1} = {}^{i+1}R_i\,{}^{i}od_i + {}^{i+1}R_i\,{}^{i}o_i \times td_{i+1}\,{}^{i+1}Z_{i+1} \tag{31}$$

$$+ tdd_{i+1}\,{}^{i+1}Z_{i+1}$$

where $^{i+1}od_{i+1}$ = the rotational acceleration of link i+1 referenced to frame i+1

$^{i}od_i$ = the rotational acceleration of link i referenced to frame i+1

tdd_{i+1} = the rotational acceleration of joint i+1

$$^{i+1}a_{i+1} = {}^{i+1}R_i[{}^{i}od_i \times {}^{i}P_{i+1} + {}^{i}o_i \times ({}^{i}o_i \times {}^{i}P_{i+1}) + {}^{i}a_i] \tag{32}$$

where $^{i+1}a_{i+1}$ = the linear acceleration of the origin of link frame i+1 in terms of frame i+1

$^{i}P_{i+1}$ = the position of frame i+1 referenced to frame i

$^{i}a_i$ = the linear acceleration of the origin of link frame i in terms of frame i

The last acceleration equation needed is one that describes the acceleration of the center of mass of each link: Equation (33).

$$^{i}a_{Ci} = {}^{i}od_i \times {}^{i}P_{Ci} + {}^{i}o_i \times ({}^{i}o_i \times {}^{i}P_{Ci}) + {}^{i}a_i \tag{33}$$

where ${}^{i}a_{Ci}$ = the linear acceleration of the center of mass of link i relative to frame 0.

${}^{i}P_{Ci}$ = the position of the center of link frame i referenced to frame i

Equations (31), (32), and (33) have been written to propagate the accelerations from link 0 to the end link, since the acceleration and velocity of link 0 is known. The gravitational acceleration effects on the links are accounted for by letting the acceleration of link 0, ${}^{0}a_{0}$, be that of gravity.

4.3.2 Force "Propagation"

The inertial force and torque acting at the mass center of each link can be found if the mass, the mass moments of inertia, and the linear acceleration of the center of mass are known. The acceleration is found using equations (31) and (32). The inertial force and torque of each link is given by Equations (34) and (35).

$$F_i = m_i \, {}^{i}a_{Ci} \tag{34}$$

where F_i = the inertial force of link i

m_i = the mass of link i

${}^{i}a_{Ci}$ = the linear acceleration of the center of link i in terms of frame i

$$N_i = I_{Ci}\,{}^{i}od_i + {}^{i}o_i \times I_{Ci}\,{}^{i}o_i \tag{35}$$

where N_i = the inertial torque of link i

I_{Ci} = the moment of inertia tensor evaluated at the mass center of link i

Now, all that remains to compute the joint torques required for motion is determination of the forces and torques exerted by the joints on each other. The forces and torques can be found by performing a force balance on a generalized link. The free-body diagram of Figure 6 results in Equations (36) and (37).

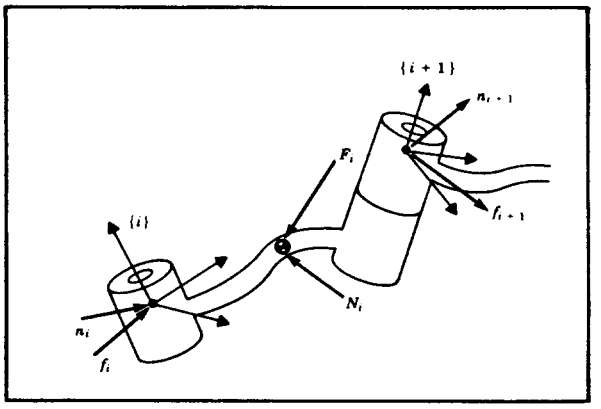

Figure 6. Free body diagram of a generalized link [from Craig].

$$ {}^{i}f_{i} = {}^{i}R_{i+1}\,{}^{i+1}f_{i+i} + {}^{i}F_{i} \tag{36} $$

where ${}^{i}f_{i}$ = the force exerted on link i by link i-1

$$ {}^{i}n_{i} = {}^{i}N_{i} + {}^{i}R_{i+1}\,{}^{i+1}n_{i+1} + {}^{i}P_{Ci} \times {}^{i}F_{i} \tag{37} $$

$$ + {}^{i}P_{i+1} \times {}^{i}R_{i+1}\,{}^{i+1}f_{i+1} $$

where ${}^{i}n_{i}$ = the torque exerted on link i by link i-1

The joint torque is the component of ${}^{i}n_{i}$ along the link joint axis, Equation (38).

$$Tau_i = {}^{i}n_i{}^{T}\,{}^{i}Z_i \tag{38}$$

where Tau_i = the joint torque of joint i

${}^{i}n_i{}^{T}$ = the transpose of the ${}^{i}n_i$ vector

Equations (36) and (37) are written to propagate forces from the last link inward to link 1.

4.3.3 Iterative Algorithm Application

All equations necessary to find the joint torques were presented in the previous two subsections. The equations can be applied by iterating forward from the base link to the end link with Equations (30), (31), (32), (33), (34) and (35), and backward from the end link to the first link with Equations (36), (37), and (38).

When applying the equations to the three-link Stanford/JPL hand, the solutions soon became too complex to compute by hand, so the MACSYMA symbolic manipulation program was used. The complete file generated representing the dynamic equations is contained in Appendix B. The resulting torques can be written in a vector equation as shown in Equation (39).

$$Tau(t,td,tdd) = M(t)tdd + V(t,td) + G(t) \tag{39}$$

where Tau = a vector of joint torques that is a function of theta (t), the first derivative of theta (td), and the second derivative of theta, (tdd)

M(t) = an inertia matrix, (a function of theta)

V(t,td) = a vector of velocity terms, containing centrifugal-force and Coriolis-force terms

Equations (40) through (54) give the terms generated in the analyses.

$$\begin{aligned} M_{11} = {} & c_2*I_{x3}*s_{23}*s_3 - c_{23}*(L_2/2)^2*m_3*s_2*s_3 \\ & - c_2*L_2*(L_2/2)*m_3*s_2*s_3 - L_1*(L_2/2)*m_3*s_2*s_3 \\ & - c_{23}*I_{y3}*s_2*s_3 + c_3*I_{x3}*s_2*s_{23} + I_{x2}*s_2^2 \\ & + c_2*c_3*c_{23}*(L_2/2)^2*m_3 + c_2*c_{23}*L_2*(L_2/2)*m_3 \\ & + c_2^2*c_3*L_2*(L_2/2)*m_3 + c_{23}*L_1*(L_2/2)*m_3 \\ & + c_2*c_3*L_1*(L_2/2)*m_3 + c_2^2*L_2^2*m_3 + 2*c_2*L_1*L_2*m_3 \\ & + (L_1/2)*m_3+c_2^2*(L_2/2)^2*m_2 + 2*c_2*L_1*(L_2/2)*m_2 \\ & + (L_1/2)*m_2 + (L_1/2)^2*m_1 + I_{z1} + c_2*c_3*c_{23}*I_{y3} \\ & + c_2^2*I_{y2} \end{aligned} \tag{40}$$

$$M_{12} = 0 \tag{41}$$

$$M_{13} = 0 \tag{42}$$

$$M_{21} = 0 \tag{43}$$

$$M_{23} = (L_2/2)^2*m_3+c_3*L_2*(L_2/2)*m_3+I_{z3} \tag{44}$$

$$M_{22} = L_2^{\ 2}*m_3*s_3^{\ 2}+(L_2/2)^2*m_3+2*c_3*L_2*(L_2/2)*m_3+c_3^{\ 2}*L_2^{\ 2}*m_3 + (L_2/2)^2*m_2+I_{z3}+I_{z2} \tag{45}$$

$$M_{31} = 0 \tag{46}$$

$$M_{32} = (L_2/2)^2*m_3+c_3*L_2*(L_2/2)*m_3+I_{z3} \tag{47}$$

$$M_{33} = (L_2/2)^2*m_3+I_{z3} \tag{48}$$

$$
\begin{aligned}
V_1 = {} & 2*(L_2/2)^2*m_3*s_2*s_{23}*s_3*td_1*td_3 \\
& +I_{z3}*s_2*s_{23}*s_3*td_1*td_3+I_{y3}*s_2*s_{23}*s_3*td_1*td_3 \\
& -I_{x3}*s_2*s_{23}*s_3*td_1*td_3+c_2*c_{23}*I_{z3}*s_3*td_1*td_3 \\
& -c_2*c_{23}*I_{y3}*s_3*td_1*td_3+c_2*c_{23}*I_{x3}*s_3*td_1*td_3 \\
& -2*c_2*c_3*(L_2/2)^2*m_3*s_{23}*td_1*td_3 \\
& -2*c_2*L_2*(L_2/2)*m_3*s_{23}*td_1*td_3 \\
& -2*L_1*(L_2/2)*m_3*s_{23}*td_1*td_3-c_2*c_3*I_{z3}*s_{23}*td_1*td_3 \\
& -c_2*c_3*I_{y3}*s_{23}*td_1*td_3+c_2*c_3*I_{x3}*s_{23}*td_1*td_3 \\
& +c_3*c_{23}*I_{z3}*s_2*td_1*td_3-c_3*c_{23}*I_{y3}*s_2*td_1*td_3 \\
& +c_3*c_{23}*I_{x3}*s_2*td_1*td_3 \\
& +2*(L_2/2)^2*m_3*s_2*s_{23}*s_3*td_1*td_2 \\
& +I_{z3}*s_2*s_{23}*s_3*td_1*td_2+I_{y3}*s_2*s_{23}*s_3*td_1*td_2 \\
& -I_{x3}*s_2*s_{23}*s_3*td_1*td_2 \\
& +2*L_2*(L_2/2)*m_3*s_2^2*s_3*td_1*td_2 \\
& +c_2*c_{23}*I_{z3}*s_3*td_1*td_2-c_2*c_{23}*I_{y3}*s_3*td_1*td_2 \\
& +c_2*c_{23}*I_{x3}*s_3*td_1*td_2 \\
& -2*c_2*c_3*(L_2/2)^2*m_3*s_{23}*td_1*td_2 \\
& -2*c_2*L_2*(L_2/2)*m_3*s_{23}*td_1*td_2 \\
& -2*L_1*(L_2/2)*m_3*s_{23}*td_1*td_2-c_2*c_3*I_{z3}*s_{23}*td_1*td_2 \\
& -c_2*c_3*I_{y3}*s_{23}*td_1*td_2+c_2*c_3*I_{x3}*s_{23}*td_1*td_2 \\
& -2*c_2*c_3*L_2*(L_2/2)*m_3*s_2*td_1*td_2 \\
& -2*c_2*L_2^2*m_3*s_2*td_1*td_2-2*L_1*L_2*m_3*s_2*td_1*td_2
\end{aligned}
\tag{49}
$$

$$-2*c_2*(L_2/2)^2*m_2*s_2*td_1*td_2$$
$$-2*L_1*(L_2/2)*m_2*s_2*td_1*td_2+c_3*c_{23}*I_{z3}*s_2*td_1*td_2$$
$$-c_3*c_{23}*I_{y3}*s_2*td_1*td_2-2*c_2*I_{y2}*s_2*td_1*td_2$$
$$+c_3*c_{23}*I_{x3}*s_2*td_1*td_2+2*c_2*I_{x2}*s_2*td_1*td_2$$

$$\begin{aligned} V_2 = & -L_2*(L_2/2)*m_3*s_3*td_3^2-2*L_2*(L_2/2)*m_3*s_3*td_2*td_3 \\ & +c_2*L_2^2*m_3*s_2*s_3^2*td_1^2+L_1*L_2*m_3*s_2*s_3^2*td_1^2 \\ & -c_{23}^2*L_2*(L_2/2)*m_3*s_3*td_1^2+c_2^2*L_2*(L_2/2)*m_3*s_3*td_1^2 \\ & +c_2*L_1*(L_2/2)*m_3*s_3*td_1^2+c_{23}*(L_2/2)^2*m_3*s_{23}*td_1^2 \\ & +c_3*c_{23}*L_2*(L_2/2)*m_3*s_{23}*td_1^2+c_{23}*I_{y3}*s_{23}*td_1^2 \\ & -c_{23}*I_{x3}*s_{23}*td_1^2+c_2*c_3*L_2*(L_2/2)*m_3*s_2*td_1^2 \\ & +c_3*L_1*(L_2/2)*m_3*s_2*td_1^2+c_2*c_3^2*L_2^2*m_3*s_2*td_1^2 \\ & +c_3^2*L_1*L_2*m_3*s_2*td_1^2+c_2*(L_2/2)^2*m_2*s_2*td_1^2 \\ & +L_1*(L_2/2)*m_2*s_2*td_1^2+c_2*I_{y2}*s_2*td_1^2-c_2*I_{x2}*s_2*td_1^2 \end{aligned} \tag{50}$$

$$\begin{aligned} V_3 = & L_2*(L_2/2)*m_3*s_3*td_2^2+c_2^2*L_2*(L_2/2)*m_3*s_3*td_1^2 \\ & +c_2*L_1*(L_2/2)*m_3*s_3*td_1^2+c_{23}*(L_2/2)^2*m_3*s_{23}*td_1^2 \\ & +c_{23}*I_{y3}*s_{23}*td_1^2-c_{23}*I_{x3}*s_{23}*td_1^2 \\ & +c_2*c_3*L_2*(L_2/2)*m_3*s_2*td_1^2+c_3*L_1*(L_2/2)*m_3*s_2*td_1^2 \end{aligned} \tag{51}$$

$$\begin{aligned} G_1 = & -n_{4y}*s_2*s_3+f_{4z}*L_3*s_2*s_3+c_2*n_{4x}*s_3 \\ & +c_3*n_{4x}*s_2+c_2*c_3*n_{4y}-c_2*c_3*f_{4z}*L_3-c_2*f_{4z}*L_2 \\ & -f_{4z}*L_1 \end{aligned} \tag{52}$$

$$G_2 = c_2*g*L_2*m_3*s_{32}-g*(L_2/2)*m_3*s_2*s_3+f_{4x}*L_2*s_3+n_{4z} \\ +c_2*c_3*g*(L_2/2)*m_3+c_2*c_{32}*g*L_2*m_3+c_2*g*(L_2/2)*m_2 \\ +f_{4y}*L_3+c_3*f_{4y}*L_2 \tag{53}$$

$$G_3 = -g*(L_2/2)*m_3*s_2*s_3+n_{4z}+c_2*c_3*g*(L_2/2)*m_3+f_{4y}*L_3 \tag{54}$$

Included in the gravity terms are forces and moments applied to the end of the last link.

APPENDIX A - Physical Parameters of the Stanford/JPL Hand

The equations developed for the Stanford/JPL hand in sections 2., 3., and 4. are written in "manipulator" space: the derived torques are the torques required by the manipulator at the joints, and the derived joint angles and velocities are the angles of the finger joints. The fingers are driven by electric motors through a gear train and a series of pulleys. The position and velocity controls must be accomplished at the motors, not the joints, and thus the calculated torques, positions, and velocities must be converted to units applicable to the motors.

Between the following dotted lines is information reprinted from Salisbury[1] defining the relationships necessary to transform joint space terms to tendon space.

PROCEDURE FOR DETERMINING TENDON TENSIONS, (T1,T2,T3,T4) GIVEN JOINT MOMENTS (M1,M2,M3):

```
Tmin = 0.0;                          : set min tendon tension
a = 0.147082081;                      : constants a and b
b = 0.227833027;
f1 =  0.351757117*M1 + 0.4403178320*M2 - 0.415855728*M3
f2 = -0.227083653*M1 - 0.0233141675*M2 + 0.751094470*M3
f3 = -0.227083653*M1 - 0.0233141675*M2 - 0.707056610*M3
f4 =  0.351757117*M1 - 0.3680896100*M2 + 0.347640183*M3
y = max((Tmin-f1)/a,(Tmin-f2)/b,(Tmin-f3)/b,(Tmin-f4)/a);
T1 = a*y + f1;  ;compute cable tensions
T2 = b*y + f2;
T3 = b*y + f3;
T4 = a*y + f4;
```

SUMMARY OF MATRICES:

$M = R.T$ and $T = R^{-1}.M$ (joint moments derived from tendon tensions and vice-versa), where $M = (M1,M2,M3,Y)^t$, $T = (T1,T2,T3,T4)^t$ and

$$R = \begin{bmatrix} 0.86763 & -0.6477 & -0.6477 & 1.1303 \\ 1.237 & 0.6477 & -0.6477 & -1.237 \\ 0.0 & 0.6858 & -0.6858 & 0.0 \\ 1.0 & 1.54902 & 1.54902 & 1.0 \end{bmatrix}$$

$$R^{-1} = \begin{bmatrix} 0.351757117 & 0.440317832 & -0.415855728 & 0.147082081 \\ -0.227083653 & -0.0233141676 & 0.75109447 & 0.227833027 \\ -0.227083653 & -0.0233141676 & -0.70705661 & 0.227833027 \\ 0.351757117 & -0.36808961 & 0.347640183 & 0.147082081 \end{bmatrix}$$

NOTES:

Dimensions are in cm. For example if T is in grams M = R.T will be the joint moments in gm-cm (except Y which will be in gms).

^{t} indicates transpose (vector or matrix)

R^{-1} indicates R inverse matrix

R.T indicates matrix vector product

Tmin (minimum cable tension) should be set sufficiently positive to keep noise in the system from causing it to attempt to exert negative tensions.

The fourth row in R is orthogonal (by construction) to the other 3 rows. Any value of T with elements proportional to the elements in row 4 will cause no net moment (M1,M2,M3) to be exerted at the joints; it will only increase the internal tension in the system.

These same matrices also relate joint and tendon velocities. If V = (vector of tendon velocities) and W = (vector of joint

angular velocities and "bearing velocity") then $V = R^t.W$ can be used to find the tendon rates necessary for a given set of joint rates.

The conversion from tendon space to motor space is simply a matter of considering the radius of the capstan that the tendons wrap onto, the gearing between the motor and capstan, and the motor encoder resolution. These values follow.

Capstan radius = .375 in.

Gearing reduction = 25/1

Encoder resolution = 2000/rev.

APPENDIX B - MACSYMA Output

This appendix is the output file of the MACSYMA symbolic manipulation program which gives the derivation of the Newton-Euler dynamic equations.

```
(c6) R01:matrix([c1,-s1,0],
                [s1,c1,0],
                [0,0,1])$

(c7) R12:matrix([c2,-s2,0],
                [0,0,-1],
                [s2,c2,0])$

(c8) R23:matrix([c3,-s3,0],
                [s3,c3,0],
                [0,0,1])$

(c9) R34:matrix([1,0,0],
                [0,1,0],
                [0,0,1])$

(c10) R10:transpose(R01)$

(c11) R21:transpose(R12)$

(c12) R32:transpose(R23)$

(c13) R43:transpose(R34)$

(c14) O00:matrix([0],[0],[0])$

(c15) OD00:matrix([0],[0],[0])$

(c16) VD00:matrix([0],[0],[g])$

(c17) Z11:matrix([0],[0],[1])$

(c18) Z22:matrix([0],[0],[1])$

(c19) Z33:matrix([0],[0],[1])$

(c20) Z44:matrix([0],[0],[1])$

(c21) P01:matrix([0],[0],[0])$
```

```
(c22) P12:matrix([L1],[0],[0])$

(c23) P1C1:matrix([L12],[0],[0])$

(c24) P23:matrix([L2],[0],[0])$

(c25) P2C2:matrix([L22],[0],[0])$

(c26) P34:matrix([L3],[0],[0])$

(c27) P3C3:matrix([L22],[0],[0])$

(c28) C1I1:matrix([IX1,0,0],
                  [0,IY1,0],
                  [0,0,IZ1])$

(c29) C2I2:matrix([IX2,0,0],
                  [0,IY2,0],
                  [0,0,IZ2])$

(c30) C3I3:matrix([IX3,0,0],
                  [0,IY3,0],
                  [0,0,IZ3])$

(c31) cross(a,b):=matrix([a[2]*b[3]-a[3]*b[2]],
      [a[3]*b[1]-a[1]*b[3]],[a[1]*b[2]-a[2]*b[1]])$

(c32) lf44:matrix([f4x],[f4y],[f4z])$

(c33) ln44:matrix([n4x],[n4y],[n4z])$

(c34) O11:R10.O00+TD1*Z11;
```

$$(d34) \quad \begin{bmatrix} 0 \\ 0 \\ td1 \end{bmatrix}$$

```
(c35) OD11:R10.O00+cross(R10.O00,TD1*Z11)+TDD1*Z11;
```

$$(d35) \quad \begin{bmatrix} [0] \\ [0] \\ [tdd1] \end{bmatrix}$$

```
(c36) VD11:R10.(cross(OD00,P01)+cross(O00,cross(O00,P01))+VD00);

                                   [ [[0]] ]
                                   [       ]
(d36)                              [ [[0]] ]
                                   [       ]
                                   [ [[g]] ]

(c37) VD1C1:cross(OD11,P1C1)+cross(O11,cross(O11,P1C1))+VD11;

                                [                2    ]
                                [ [[- l12 td1 ]] ]
(d37)                           [                    ]
                                [  [[l12 tdd1]]      ]
                                [                    ]
                                [     [[g]]          ]

(c38) F11:M1*VD1C1;

                               [                   2    ]
                               [ [[- l12 m1 td1 ]] ]
(d38)                          [                       ]
                               [  [[l12 m1 tdd1]]      ]
                               [                       ]
                               [     [[g m1]]          ]

(c39) N11:C1I1.OD11+cross(O11,C1I1.O11);

                                  [     [0]      ]
                                  [              ]
(d39)                             [     [0]      ]
                                  [              ]
                                  [ [iz1 tdd1] ]

(c40) O22:R21.O11+TD2*Z22;

                                    [ s2 td1 ]
                                    [        ]
(d40)                               [ c2 td1 ]
                                    [        ]
                                    [  td2   ]

(c41) OD22:R21.OD11+cross(R21.O11,TD2*Z22)+TDD2*Z22;

                            [ [s2 tdd1 + c2 td1 td2] ]
                            [                        ]
(d41)                       [ [c2 tdd1 - s2 td1 td2] ]
                            [                        ]
                            [          [tdd2]        ]
```

```
(c42) VD22:R21.(cross(OD11,P12)+cross(O11,cross(O11,P12))+VD11);

                                 [                      2    ]
                                 [ [[g s2 - c2 l1 td1 ]] ]
                                 [                           ]
(d42)                            [                 2         ]
                                 [ [[l1 s2 td1  + c2 g]] ]
                                 [                           ]
                                 [     [[- l1 tdd1]]         ]

(c43) VD2C2:cross(OD22,P2C2)+cross(O22,cross(O22,P2C2))+VD22;

        [                    2      2          2              2                ]
        [      [[- l22 td2  - c2  l22 td1  - c2 l1 td1  + g s2]]               ]
        [                                                                       ]
(d43)   [                                     2              2                 ]
        [      [[l22 tdd2 + c2 l22 s2 td1  + l1 s2 td1  + c2 g]]               ]
        [                                                                       ]
        [ [[- l22 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s2 td1 td2]] ]

(c44) F22:M2*VD2C2;

      [                      2      2          2              2                 ]
      [      [[m2 (- l22 td2  - c2  l22 td1  - c2 l1 td1  + g s2)]]             ]
      [                                                                         ]
(d44) [                                      2              2                   ]
      [      [[m2 (l22 tdd2 + c2 l22 s2 td1  + l1 s2 td1  + c2 g)]]             ]
      [                                                                         ]
      [ [[m2 (- l22 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s2 td1 td2)]] ]

(c45) N22:C2I2.OD22+cross(O22,C2I2.O22);

      [ [ix2 (s2 tdd1 + c2 td1 td2) + c2 iz2 td1 td2 - c2 iy2 td1 td2] ]
      [                                                                 ]
(d45) [ [iy2 (c2 tdd1 - s2 td1 td2) - iz2 s2 td1 td2 + ix2 s2 td1 td2] ]
      [                                                                 ]
      [                                      2                  2       ]
      [          [iz2 tdd2 + c2 iy2 s2 td1  - c2 ix2 s2 td1 ]          ]

(c46) O33:R32.O22+TD3*Z33;

                                 [ c2 s3 td1 + c3 s2 td1 ]
                                 [                        ]
(d46)                            [ c2 c3 td1 - s2 s3 td1 ]
                                 [                        ]
                                 [       td3 + td2        ]
```

```
(c47) O33:matrix([s23*TD1],
          [co23*TD1],
          [TD2+TD3])$

(c48) OD33:R32.OD22+cross(R32.O22,TD3*Z33)+TDD3*Z33;

(d48) matrix([[c3 (s2 tdd1 + c2 td1 td2) + s3 (c2 tdd1 - s2 td1 td2)

 + (c2 c3 td1 - s2 s3 td1) td3]], [[- s3 (s2 tdd1 + c2 td1 td2)

 + c3 (c2 tdd1 - s2 td1 td2) - (c2 s3 td1 + c3 s2 td1) td3]], [[tdd3 + tdd2]])

(c49) OD33:matrix([s23*TDD1+co23*TD1*TD2+co23*TD1*TD3],
           [co23*TDD1-s23*TD1*TD2-s23*TD1*TD3],
           [TDD2+TDD3]);

                 [ s23 tdd1 + co23 td1 td3 + co23 td1 td2 ]
                 [                                        ]
(d49)            [ co23 tdd1 - s23 td1 td3 - s23 td1 td2  ]
                 [                                        ]
                 [               tdd3 + tdd2              ]

(c50) VD33:R32.(cross(OD22,P23)+cross(O22,cross(O22,P23))+VD22);

                                                  2            2
(d50) matrix([[[s3 (l2 tdd2 + c2 l2 s2 td1  + l1 s2 td1  + c2 g)

                 2      2        2            2
 + c3 (- l2 td2  - c2  l2 td1  - c2 l1 td1  + g s2)]]],

                                  2            2
[[[c3 (l2 tdd2 + c2 l2 s2 td1  + l1 s2 td1  + c2 g)

                 2      2        2            2
 - s3 (- l2 td2  - c2  l2 td1  - c2 l1 td1  + g s2)]]],

[[[- l2 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l2 s2 td1 td2]]])

(c51) VD3C3:cross(OD33,P3C3)+cross(O33,cross(O33,P3C3))+VD33;

                                                  2            2
(d51) matrix([[[s3 (l2 tdd2 + c2 l2 s2 td1  + l1 s2 td1  + c2 g)

                   2                 2      2        2            2
 - l22 (td3 + td2)  + c3 (- l2 td2  - c2  l2 td1  - c2 l1 td1  + g s2)

       2         2
 - co23  l22 td1  ]]], [[[l22 (tdd3 + tdd2)
```

```
 + c3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g)
 - s3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) + co23 l22 s23 td1^2 ]]],
[[[- l22 (co23 tdd1 - s23 td1 td3 - s23 td1 td2) - l2 (c2 tdd1 - s2 td1 td2)
 - l1 tdd1 + l22 s23 td1 (td3 + td2) + l2 s2 td1 td2]]])
(c52) F33:M3*VD3C3;
(d52) matrix([[[m3 (s3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g)
 - l22 (td3 + td2)^2 + c3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2)
 - co23^2 l22 td1^2 )]]], [[[m3 (l22 (tdd3 + tdd2)
 + c3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g)
 - s3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) + co23 l22 s23 td1^2 )]]],
[[[m3 (- l22 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
 - l2 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s23 td1 (td3 + td2)
 + l2 s2 td1 td2)]]])
(c53) N33:C3I3.OD33+cross(O33,C3I3.O33);
(d53) matrix([[ix3 (s23 tdd1 + co23 td1 td3 + co23 td1 td2)
 + co23 iz3 td1 (td3 + td2) - co23 iy3 td1 (td3 + td2)]],
[[iy3 (co23 tdd1 - s23 td1 td3 - s23 td1 td2) - iz3 s23 td1 (td3 + td2)
 + ix3 s23 td1 (td3 + td2)]], [[iz3 (tdd3 + tdd2) + co23 iy3 s23 td1^2
 - co23 ix3 s23 td1^2 ]])
(c54) lf33:R34.lf44+F33;
```

```
(d54) matrix([[[m3 (s3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g)
 - l22 (td3 + td2)^2 + c3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2)
 - co23^2 l22 td1^2) + f4x]]], [[[m3 (l22 (tdd3 + tdd2)
 + c3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g)
 - s3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) + co23 l22 s23 td1^2)
 + f4y]]], [[[m3 (- l22 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
 - l2 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s23 td1 (td3 + td2)
 + l2 s2 td1 td2) + f4z]]])
(c55) ln33:N33+R34.ln44+cross(P3C3,F33)+cross(P34,R34.lf44);
(d55) matrix([[[[ix3 (s23 tdd1 + co23 td1 td3 + co23 td1 td2)
 + co23 iz3 td1 (td3 + td2) - co23 iy3 td1 (td3 + td2) + n4x]]]],
[[[[- l22 m3 (- l22 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
 - l2 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s23 td1 (td3 + td2)
 + l2 s2 td1 td2) + iy3 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
 - iz3 s23 td1 (td3 + td2) + ix3 s23 td1 (td3 + td2) + n4y - f4z l3]]]],
[[[[l22 m3 (l22 (tdd3 + tdd2) + c3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2
 + c2 g) - s3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2)
 + co23 l22 s23 td1^2) + iz3 (tdd3 + tdd2) + co23 iy3 s23 td1^2
 - co23 ix3 s23 td1^2 + n4z + f4y l3]]]])
```

```
(c56) TAU3:grind(ratexpand(transpose(ln33).Z33));

[[[l22^2*m3*tdd3+iz3*tdd3+l22^2*m3*tdd2+c3*l2*l22*m3*tdd2+iz3*tdd2
                +l2*l22*m3*s3*td2^2+c2^2*l2*l22*m3*s3*td1^2
                +c2*l1*l22*m3*s3*td1^2+co23*l22^2*m3*s23*td1^2
                +co23*iy3*s23*td1^2-co23*ix3*s23*td1^2
                +c2*c3*l2*l22*m3*s2*td1^2+c3*l1*l22*m3*s2*td1^2-g*l22*m3*s2*s3
                +n4z+c2*c3*g*l22*m3+f4y*l3]]]$
(d56)                                   done

(c57) lf22:R23.lf33+F22;

(d57) matrix([[[- s3 (m3 (l22 (tdd3 + tdd2)

 + c3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g)

 - s3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) + co23 l22 s23 td1^2) + f4y)

 + c3 (m3 (s3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g) - l22 (td3 + td2)^2

 + c3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) - co23^2 l22 td1^2) + f4x)

 + m2 (- l22 td2^2 - c2^2 l22 td1^2 - c2 l1 td1^2 + g s2)]]],

[[[c3 (m3 (l22 (tdd3 + tdd2) + c3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g)

 - s3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) + co23 l22 s23 td1^2) + f4y)

 + s3 (m3 (s3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g) - l22 (td3 + td2)^2

 + c3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) - co23^2 l22 td1^2) + f4x)

 + m2 (l22 tdd2 + c2 l22 s2 td1^2 + l1 s2 td1^2 + c2 g)]]],

[[[m3 (- l22 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)

 - l2 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s23 td1 (td3 + td2)
```

```
 + l2 s2 td1 td2) + m2 (- l22 (c2 tdd1 - s2 td1 td2) - l1 tdd1
 + l22 s2 td1 td2) + f4z]]])
(c58) ln22:N22+R23.ln33+cross(P2C2,F22)+cross(P23,R23.lf33);
(d58) matrix([[[[- s3 (- l22 m3 (- l22 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
 - l2 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s23 td1 (td3 + td2)
 + l2 s2 td1 td2) + iy3 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
 - iz3 s23 td1 (td3 + td2) + ix3 s23 td1 (td3 + td2) + n4y - f4z l3)
 + c3 (ix3 (s23 tdd1 + co23 td1 td3 + co23 td1 td2) + co23 iz3 td1 (td3 + td2)
 - co23 iy3 td1 (td3 + td2) + n4x) + ix2 (s2 tdd1 + c2 td1 td2)
 + c2 iz2 td1 td2 - c2 iy2 td1 td2]]]],
[[[[c3 (- l22 m3 (- l22 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
 - l2 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s23 td1 (td3 + td2)
 + l2 s2 td1 td2) + iy3 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
 - iz3 s23 td1 (td3 + td2) + ix3 s23 td1 (td3 + td2) + n4y - f4z l3)
 - l2 (m3 (- l22 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
 - l2 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s23 td1 (td3 + td2)
 + l2 s2 td1 td2) + f4z) + s3 (ix3 (s23 tdd1 + co23 td1 td3 + co23 td1 td2)
 + co23 iz3 td1 (td3 + td2) - co23 iy3 td1 (td3 + td2) + n4x)
 - l22 m2 (- l22 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s2 td1 td2)
 + iy2 (c2 tdd1 - s2 td1 td2) - iz2 s2 td1 td2 + ix2 s2 td1 td2]]]],
                                                                  2            2
[[[[l2 (c3 (m3 (l22 (tdd3 + tdd2) + c3 (l2 tdd2 + c2 l2 s2 td1  + l1 s2 td1
                                 2      2        2            2
 + c2 g) - s3 (- l2 td2  - c2  l2 td1  - c2 l1 td1  + g s2)
                   2
 + co23 l22 s23 td1 ) + f4y) + s3 (m3 (s3
```

```
                                  2                   2                                  2
 (l2 tdd2 + c2 l2 s2 td1  + l1 s2 td1  + c2 g) - l22 (td3 + td2)

                 2      2        2              2                 2         2
 + c3 (- l2 td2  - c2  l2 td1  - c2 l1 td1  + g s2) - co23  l22 td1 ) + f4x))

                                                                  2            2
 + l22 m3 (l22 (tdd3 + tdd2) + c3 (l2 tdd2 + c2 l2 s2 td1  + l1 s2 td1  + c2 g)

                 2      2        2              2                                2
 - s3 (- l2 td2  - c2  l2 td1  - c2 l1 td1  + g s2) + co23 l22 s23 td1 )

                                                                 2            2
 + iz3 (tdd3 + tdd2) + l22 m2 (l22 tdd2 + c2 l22 s2 td1  + l1 s2 td1  + c2 g)

                                   2                    2                  2
 + iz2 tdd2 + co23 iy3 s23 td1  - co23 ix3 s23 td1  + c2 iy2 s2 td1

                 2
 - c2 ix2 s2 td1  + n4z + f4y l3]]]]])

(c59) TAU2:grind(ratexpand(transpose(ln22).Z22));

[[[l22^2*m3*tdd3+c3*l2*l22*m3*tdd3+iz3*tdd3+l2^2*m3*s3^2*tdd2+l22^2*m3*tdd2
                +2*c3*l2*l22*m3*tdd2+c3^2*l2^2*m3*tdd2+l22^2*m2*tdd2+iz3*tdd2
                +iz2*tdd2-l2*l22*m3*s3*td3^2-2*l2*l22*m3*s3*td2*td3
                +c2*l2^2*m3*s2*s3^2*td1^2+l1*l2*m3*s2*s3^2*td1^2
                -co23^2*l2*l22*m3*s3*td1^2+c2^2*l2*l22*m3*s3*td1^2
                +c2*l1*l22*m3*s3*td1^2+co23*l22^2*m3*s23*td1^2
                +c3*co23*l2*l22*m3*s23*td1^2+co23*iy3*s23*td1^2
                -co23*ix3*s23*td1^2+c2*c3*l2*l22*m3*s2*td1^2
                +c3*l1*l22*m3*s2*td1^2+c2*c3^2*l2^2*m3*s2*td1^2
                +c3^2*l1*l2*m3*s2*td1^2+c2*l22^2*m2*s2*td1^2
                +l1*l22*m2*s2*td1^2+c2*iy2*s2*td1^2-c2*ix2*s2*td1^2
                +c2*g*l2*m3*s3^2-g*l22*m3*s2*s3+f4x*l2*s3+n4z+c2*c3*g*l22*m3
                +c2*c3^2*g*l2*m3+c2*g*l22*m2+f4y*l3+c3*f4y*l2]]]$
(d59)                                done

(c60) lf11:R12.lf22+F11;

(d60) matrix([[[c2 (- s3 (m3 (l22 (tdd3 + tdd2)

                                  2                   2
 + c3 (l2 tdd2 + c2 l2 s2 td1  + l1 s2 td1  + c2 g)

                 2      2        2              2                                 2
 - s3 (- l2 td2  - c2  l2 td1  - c2 l1 td1  + g s2) + co23 l22 s23 td1 ) + f4y)
```

```
 + c3 (m3 (s3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g) - l22 (td3 + td2)^2
 + c3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) - co23^2 l22 td1^2 ) + f4x)
 + m2 (- l22 td2^2 - c2^2 l22 td1^2 - c2 l1 td1^2 + g s2))
 - s2 (c3 (m3 (l22 (tdd3 + tdd2) + c3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2
 + c2 g) - s3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2)
 + co23 l22 s23 td1^2 ) + f4y) + s3 (m3 (s3
 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g) - l22 (td3 + td2)^2
 + c3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) - co23^2 l22 td1^2 ) + f4x)
 + m2 (l22 tdd2 + c2 l22 s2 td1^2 + l1 s2 td1^2 + c2 g)) - l12 m1 td1^2 ]]],
[[[- m3 (- l22 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
 - l2 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s23 td1 (td3 + td2)
 + l2 s2 td1 td2) - m2 (- l22 (c2 tdd1 - s2 td1 td2) - l1 tdd1
 + l22 s2 td1 td2) + l12 m1 tdd1 - f4z]]],
[[[s2 (- s3 (m3 (l22 (tdd3 + tdd2) + c3
 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g)
 - s3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) + co23 l22 s23 td1^2 ) + f4y)
 + c3 (m3 (s3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g) - l22 (td3 + td2)^2
 + c3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) - co23^2 l22 td1^2 ) + f4x)
```

```
+ m2 (- l22 td2^2 - c2^2 l22 td1^2 - c2 l1 td1^2 + g s2))
+ c2 (c3 (m3 (l22 (tdd3 + tdd2) + c3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2
+ c2 g) - s3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2)
+ co23 l22 s23 td1^2) + f4y) + s3 (m3 (s3
(l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g) - l22 (td3 + td2)^2
+ c3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) - co23^2 l22 td1^2) + f4x)
+ m2 (l22 tdd2 + c2 l22 s2 td1^2 + l1 s2 td1^2 + c2 g)) + g m1]]])
(c61) ln11:N11+R12.ln22+cross(P1C1,F11)+cross(P12,R12.lf22);
(d61) matrix([[[[c2 (- s3 (- l22 m3 (- l22
(co23 tdd1 - s23 td1 td3 - s23 td1 td2) - l2 (c2 tdd1 - s2 td1 td2) - l1 tdd1
+ l22 s23 td1 (td3 + td2) + l2 s2 td1 td2)
+ iy3 (co23 tdd1 - s23 td1 td3 - s23 td1 td2) - iz3 s23 td1 (td3 + td2)
+ ix3 s23 td1 (td3 + td2) + n4y - f4z l3)
+ c3 (ix3 (s23 tdd1 + co23 td1 td3 + co23 td1 td2) + co23 iz3 td1 (td3 + td2)
- co23 iy3 td1 (td3 + td2) + n4x) + ix2 (s2 tdd1 + c2 td1 td2)
+ c2 iz2 td1 td2 - c2 iy2 td1 td2) - s2
(c3 (- l22 m3 (- l22 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
- l2 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s23 td1 (td3 + td2)
+ l2 s2 td1 td2) + iy3 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
- iz3 s23 td1 (td3 + td2) + ix3 s23 td1 (td3 + td2) + n4y - f4z l3)
- l2 (m3 (- l22 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
```

```
- l2 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s23 td1 (td3 + td2)
+ l2 s2 td1 td2) + f4z) + s3 (ix3 (s23 tdd1 + co23 td1 td3 + co23 td1 td2)
+ co23 iz3 td1 (td3 + td2) - co23 iy3 td1 (td3 + td2) + n4x)
- l22 m2 (- l22 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s2 td1 td2)
+ iy2 (c2 tdd1 - s2 td1 td2) - iz2 s2 td1 td2 + ix2 s2 td1 td2)]]]],
[[[[- l1 (s2 (- s3 (m3 (l22 (tdd3 + tdd2)
+ c3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g)
- s3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) + co23 l22 s23 td1^2) + f4y)
+ c3 (m3 (s3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g) - l22 (td3 + td2)^2
+ c3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) - co23^2 l22 td1^2) + f4x)
+ m2 (- l22 td2^2 - c2^2 l22 td1^2 - c2 l1 td1^2 + g s2))
+ c2 (c3 (m3 (l22 (tdd3 + tdd2) + c3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2
+ c2 g) - s3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2)
+ co23 l22 s23 td1^2) + f4y) + s3 (m3 (s3
(l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g) - l22 (td3 + td2)^2
+ c3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) - co23^2 l22 td1^2) + f4x)
+ m2 (l22 tdd2 + c2 l22 s2 td1^2 + l1 s2 td1^2 + c2 g)))
- l2 (c3 (m3 (l22 (tdd3 + tdd2) + c3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2
```

```
+ c2 g) - s3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2)
+ co23 l22 s23 td1^2) + f4y) + s3 (m3 (s3
(l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g) - l22 (td3 + td2)^2
+ c3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) - co23^2 l22 td1^2) + f4x))
- l22 m3 (l22 (tdd3 + tdd2) + c3 (l2 tdd2 + c2 l2 s2 td1^2 + l1 s2 td1^2 + c2 g)
- s3 (- l2 td2^2 - c2^2 l2 td1^2 - c2 l1 td1^2 + g s2) + co23 l22 s23 td1^2)
- iz3 (tdd3 + tdd2) - l22 m2 (l22 tdd2 + c2 l22 s2 td1^2 + l1 s2 td1^2 + c2 g)
- iz2 tdd2 - co23 iy3 s23 td1^2 + co23 ix3 s23 td1^2 - c2 iy2 s2 td1^2
+ c2 ix2 s2 td1^2 - n4z - g l12 m1 - f4y l3]]]],
[[[[s2 (- s3 (- l22 m3 (- l22 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
- l2 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s23 td1 (td3 + td2)
+ l2 s2 td1 td2) + iy3 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
- iz3 s23 td1 (td3 + td2) + ix3 s23 td1 (td3 + td2) + n4y - f4z l3)
+ c3 (ix3 (s23 tdd1 + co23 td1 td3 + co23 td1 td2) + co23 iz3 td1 (td3 + td2)
- co23 iy3 td1 (td3 + td2) + n4x) + ix2 (s2 tdd1 + c2 td1 td2)
+ c2 iz2 td1 td2 - c2 iy2 td1 td2) + c2
(c3 (- l22 m3 (- l22 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
- l2 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s23 td1 (td3 + td2)
+ l2 s2 td1 td2) + iy3 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
- iz3 s23 td1 (td3 + td2) + ix3 s23 td1 (td3 + td2) + n4y - f4z l3)
```

```
 - l2 (m3 (- l22 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
 - l2 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s23 td1 (td3 + td2)
 + l2 s2 td1 td2) + f4z) + s3 (ix3 (s23 tdd1 + co23 td1 td3 + co23 td1 td2)
 + co23 iz3 td1 (td3 + td2) - co23 iy3 td1 (td3 + td2) + n4x)
 - l22 m2 (- l22 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s2 td1 td2)
 + iy2 (c2 tdd1 - s2 td1 td2) - iz2 s2 td1 td2 + ix2 s2 td1 td2)
 + l1 (- m3 (- l22 (co23 tdd1 - s23 td1 td3 - s23 td1 td2)
 - l2 (c2 tdd1 - s2 td1 td2) - l1 tdd1 + l22 s23 td1 (td3 + td2)
 + l2 s2 td1 td2) - m2 (- l22 (c2 tdd1 - s2 td1 td2) - l1 tdd1
                                 2
 + l22 s2 td1 td2) - f4z) + l12  m1 tdd1 + iz1 tdd1]]]])
(c62) TAU1:grind(ratexpand(transpose(ln11).Z11));
[[[c2*ix3*s23*s3*tdd1-co23*l22^2*m3*s2*s3*tdd1-c2*l2*l22*m3*s2*s3*tdd1
                      -l1*l22*m3*s2*s3*tdd1-co23*iy3*s2*s3*tdd1
                      +c3*ix3*s2*s23*tdd1+ix2*s2^2*tdd1
                      +c2*c3*co23*l22^2*m3*tdd1+c2*co23*l2*l22*m3*tdd1
                      +c2^2*c3*l2*l22*m3*tdd1+co23*l1*l22*m3*tdd1
                      +c2*c3*l1*l22*m3*tdd1+c2^2*l2^2*m3*tdd1
                      +2*c2*l1*l2*m3*tdd1+l1^2*m3*tdd1+c2^2*l22^2*m2*tdd1
                      +2*c2*l1*l22*m2*tdd1+l1^2*m2*tdd1+l12^2*m1*tdd1+iz1*tdd1
                      +c2*c3*co23*iy3*tdd1+c2^2*iy2*tdd1
                      +2*l22^2*m3*s2*s23*s3*td1*td3+iz3*s2*s23*s3*td1*td3
                      +iy3*s2*s23*s3*td1*td3-ix3*s2*s23*s3*td1*td3
                      +c2*co23*iz3*s3*td1*td3-c2*co23*iy3*s3*td1*td3
                      +c2*co23*ix3*s3*td1*td3-2*c2*c3*l22^2*m3*s23*td1*td3
                      -2*c2*l2*l22*m3*s23*td1*td3-2*l1*l22*m3*s23*td1*td3
                      -c2*c3*iz3*s23*td1*td3-c2*c3*iy3*s23*td1*td3
                      +c2*c3*ix3*s23*td1*td3+c3*co23*iz3*s2*td1*td3
                      -c3*co23*iy3*s2*td1*td3+c3*co23*ix3*s2*td1*td3
                      +2*l22^2*m3*s2*s23*s3*td1*td2+iz3*s2*s23*s3*td1*td2
                      +iy3*s2*s23*s3*td1*td2-ix3*s2*s23*s3*td1*td2
                      +2*l2*l22*m3*s2^2*s3*td1*td2+c2*co23*iz3*s3*td1*td2
                      -c2*co23*iy3*s3*td1*td2+c2*co23*ix3*s3*td1*td2
                      -2*c2*c3*l22^2*m3*s23*td1*td2-2*c2*l2*l22*m3*s23*td1*td2
                      -2*l1*l22*m3*s23*td1*td2-c2*c3*iz3*s23*td1*td2
                      -c2*c3*iy3*s23*td1*td2+c2*c3*ix3*s23*td1*td2
                      -2*c2*c3*l2*l22*m3*s2*td1*td2-2*c2*l2^2*m3*s2*td1*td2
                      -2*l1*l2*m3*s2*td1*td2-2*c2*l22^2*m2*s2*td1*td2
```

```
-2*l1*l22*m2*s2*td1*td2+c3*co23*iz3*s2*td1*td2
-c3*co23*iy3*s2*td1*td2-2*c2*iy2*s2*td1*td2
+c3*co23*ix3*s2*td1*td2+2*c2*ix2*s2*td1*td2-n4y*s2*s3
+f4z*l3*s2*s3+c2*n4x*s3+c3*n4x*s2+c2*c3*n4y-c2*c3*f4z*l3
-c2*f4z*l2-f4z*l1]]]$
(d62)                                done
```

References

[1] K. Salisbury, "The Stanford/JPL Hand: Mechanical Specifications", (Salisbury Robotics, Inc., Palo Alto, CA, 1984.)

[2] K. Salisbury, "Design and Control of an Articulated Hand", (International Sumposium on Design and Synthesis, Tokyo, July 1984.)

[3] John J. Craig, Introduction to Robotics Mechanics and Control, (Massachusettes: Addison-Wesley Publishing Company, Inc., 1986.)

[4] H. Asada and J.-J. E. Slotine, Robot Analysis and Control, (New York: John Wiley and Sons, Inc., 1986.)

DISTRIBUTION:

2540 G. N. Beeler
2545 J. H. Barnette
2545 V. J. Johnson (25)
2545 D. W. Plummer
2545 M. A. Powell
1414 C. S. Loucks
1414 R. W. Harrigan
3141 S. A. Landenberger (5)
3151 W. L. Garner (3)
3154-1 C. H. Dalin (28) for DOE/OSTI
8024 P. W. Dean
UNM Dr. G. P. Starr