KINEMATIC, DYNAMIC AND ACCURACY RELIABLITY ANALYSIS OF 6 DEGREE-OF-FREEDOM ROBOTIC ARM

A Thesis of Master of Science Presented To

The Department of Mechanical and Aerospace Engineering

College of Engineering

University of Missouri-Columbia

By

ZHENTAO XIE

JULY, 2013

The undersigned have examined this thesis entitled

KINEMATIC, DYNAMIC AND ACCURACY RELIABLITY ANALYSIS

OF 6 DEGREE-OF-FREEDOM ROBOTIC ARM

Presented by

Zhentao Xie

A candidate for the degree of

Master of Science

And hereby certify that, in their opinion, it is worthy of acceptance.

Dr. A.Sherif El-Gizawy (Advisor)	Date
Dr. Yuyi Lin	Date
Dr. Hani A Salim	Date

ACKNOWLEDGEMENTS

First of all, I would like to thank my parents, families and friends for their continuous support and encouragement.

Most importantly, I would like to thank my advisor Dr. A.Sherif El-Gizawy to provide me financial and academic support in my master's period. And I benefited a lot from his rigorous academic attitude and patiently teaching.

Besides, I am thankful for my master's committee members, Dr. Yuyi Lin and Dr. Hani A Salim. Their wise ideas and suggestions inspired me so much on my research and study.

Then, I am really grateful for Dr. Roger Fales from his great recommendation on a book named *Modeling and Control of Robot Manipulator* written by Lorenzo Sciavicco and Bruno Siciliano. This book is significant for me as an important reference on my research.

Lastly, I would like to thank my colleagues: Brian Graybill, Annemarie Hoyer, Shane P Corl, Mike Scheller, Amer Krvavac, Jacob Harris, Andy Gunn, Tony Adamson, Laurent Eap. We are working as a team, sharing ideas, helping each other and promoting together.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

In this article, first of all, it describes the procedure for kinematic and dynamic analysis of a 6-degrees-of-freedom robotic arm. In kinematic analysis, it includes kinematics and differential kinematics. In which, the *Denavit-Hartenberg Parameters*, *Homogeneous Transformation Matrix*, *Direct Kinematic Function* and *Geometric Jacobian* are derived. In the dynamic analysis, the Lagrange Formulation is derived and the equations of motion have been formulated in joint space using *Lagrangian* equation. Then, it presents the accuracy reliability analysis based on kinematic parameters.

INTRODUCTION

My master's thesis is about modeling a six-degree-of -freedom robot arm and analyzing its accuracy reliability.

Figure 1 shows the 6 degree-of-freedom robot arm. There are two dgrees in "base", one degree in "arm elbow", one degree in "arm wrist" and another two in fingers. This robot arm is designed to work in hot cell. The inner environment of hot cell is high irradiation and high hazardous. Figure 2 shows the hot cell. There is a pair of manual manipulators working inside of the hot cell which are controlled manually. The robot arm will realize the working automation to replace manual manipulator for safer, faster, and cheaper operation procedure.

In the modeling part of this article includes kinematic analysis, differential kinematic analysis and dynamic analysis. Kinematics described the relationship between the joint positions and the end-effecter position and orientation. Differential kinematics describes the relationship of velocity between joints and end-effecter. For dynamics analysis, it describes the relationship between the joint actuator torques and the motion of the structure. In this part, I used the book named *Modeling and Control of Robot Manipulator* written by Lorenzo Sciavicco and Bruno Siciliano [1] as my main reference. I followed the method from book to create model for this robot arm and presented the detail of formula derivation. Besides, I also referred other researchers' job on this subject. Kazuhiro Kosuage, Katsuhisa Furuta [2], they analyzed the static problem of robot arm using Jacobian of the transformation between the joint coordinate and task coordinate. Kok-Meng Lee and Dharmank Shan [5], they studied the dynamic of robot arm by formulating the equations of motion in joint space using Largragian Equation. C.S.G.Lee and R.Nlgam [6] analyzed the dynamic of a three-degree-of-freedom arm by using Alembert equations of motion. In this article of my work, I presented kinematic analysis by using Homogeneous Transformation method, and presented differential kinematic analysis by using Geometric Jacobian and presented dynamics analysis by using Lagrange Formulation.

For the accuracy reliability, it is the critical index to value a robot performance. There are many researchers working this subject. Mohamed Abderrahim, Allakhamis, Santiago Garrido, Luis Moreno [3] elaborated the definition of repeatability and accuracy, and highlighted the error sources. Kevin L. Conrad, Panayiotis S. Shiakolas [4] examined the concepy of accuracy, repeatability, and resolution under the homogeneous transformation method. Recently, Mashesh D Pandey and Xufang Zhang [10] analyzed the system reliability using entropy-based probility distribution using fractional moments as constraints. Zhao Liang, Su Meng, Miao Yunchen [11] analyzed the accuracy of robotic arm by building a pose error model with screw theory. In this article I used lognormal distribution density function to calculate the accuracy probability.

In this article, there are four main chapters. Chapter 1 is about Kinematics. Chapter 2 is about Differential Kinematics. Chapter 3 is about Dynamics. Chapter 4 is about robotic position accuracy reliability analysis.

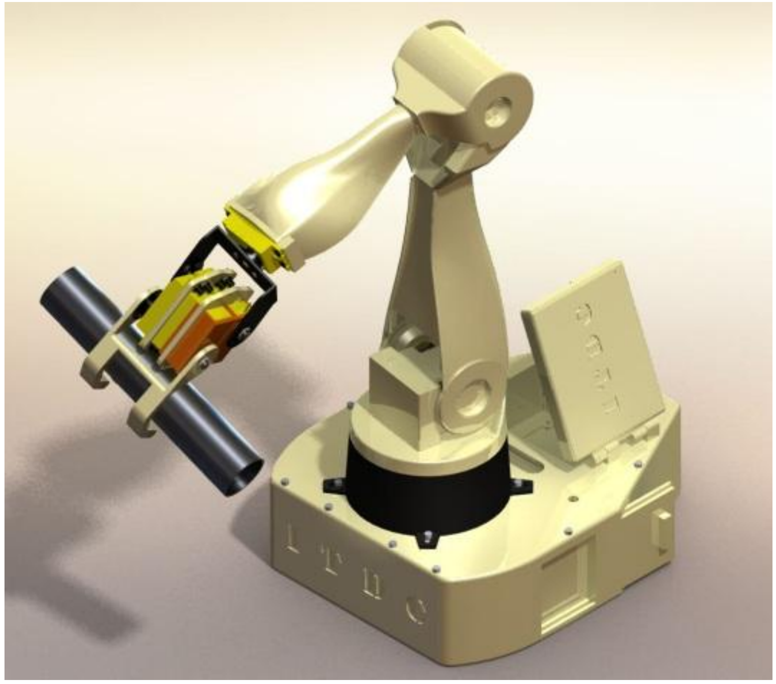

Figure 1 Six Degree-of-Freedom Robotic Arm

Figure 2 Hot Cell

Chapter 1. KINEMATIC ANALYSIS

From a mechanical structure point of view, a robot arm is an open kinematic chain which connected by revolute or prismatic joints. One end is mounted on the base and the other end is the end-effector. The motion of robot arm is obtained by the whole elementary motions of all links, and the motion of each link is respect to the previous one. Hence, the end-effector position and orientation are critical description to manipulate the robot arm in a 3-D working space. In this chapter, firstly, it sets up Denavit-Hartenberg parameters, derives rotation matrix and solves the Homogeneous Transformation matrix. Then it derives the direct kinematics equation, which expresses the “end-effector position and orientation as a function of the joint variables of the mechanical structure with respect to a reference frame” [1]. Finally, the chapter derives the inverse kinematics equation, which “consists of the determination of the joints variables corresponding to a given end-effector configuration” [1].

1.1 Denavit-Hartenberg Convention

“The Danavit-Hartenberg parameters (DH parameters) are the four parameters associated with a particular convention for attaching reference frames to the links of a spatial kinematic chain, or robot manipulator.” [1] The links frame of Denavit-Hartenberg are define as follows:

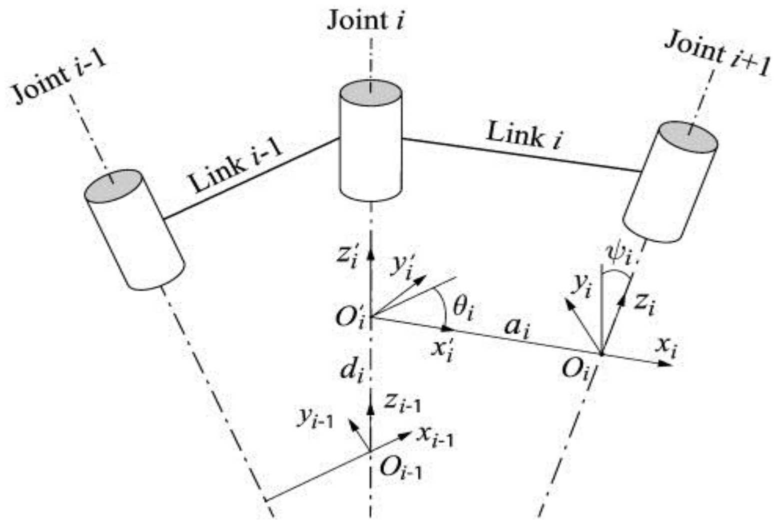

Figure 3 Denavit-Hartenberg Convention

Set up link frame

- O_i is the origin of frame O_i-$x_iy_iz_i$, and a_i is common normal of axis z_i and z_{i-1}. Also, O_i is the intersection of the axis z_i and common normal a_i.
- Axis x_i is along the common normal. a_i and its direction is from joint i to joint i+1.
- y_i is to complete a right-handed frame.

After setting up the link frame, the position and orientation of frame i and frame i-1can be represented by Denavit-Hartenberg parameters (shown in Figure 3):

- a_i : Distance of common normal between joint i and joint i+1.
- d_i : Distance of two perpendicular points of common normal joint i-1 with i and joint i with i+1
- α_i : angle between axes $z_{i-1}$and z_i about axis x_i
- θ_i : angle between axes $x_{i-1}$and x_i about axis z_{i-1}

1.1.1 Set Up Denavit-Hartenberg Parameters

As shown in figure 4. The origin of the base frame was located at the intersection of z_0 with $z_1$so that d_1= 0, a_1= 0; the angle between axes z_0 and z_1 is π/2; joint 1 rotates by angle θ_1.

z_1 and z_2 are parallel and x_1 and x_2 are along the direction of the relative links, in this way, d_2 = 0, distance between o_1 and o_2 is a_2; the angle between axes z_1 and z_2 is 0; joint 2 rotates $\theta_{2.}$

z_2 and z_3 are at intersection, so that d_3=0, a_3 = 0; the angle between axes z_2 and z_3 is π/2; link 3 rotates θ_3.

z_4 is along with $z_{3,}$ so the distance between o_3 and o_4 is d_4 and α_4= 0; joint 4 rotates θ_4.

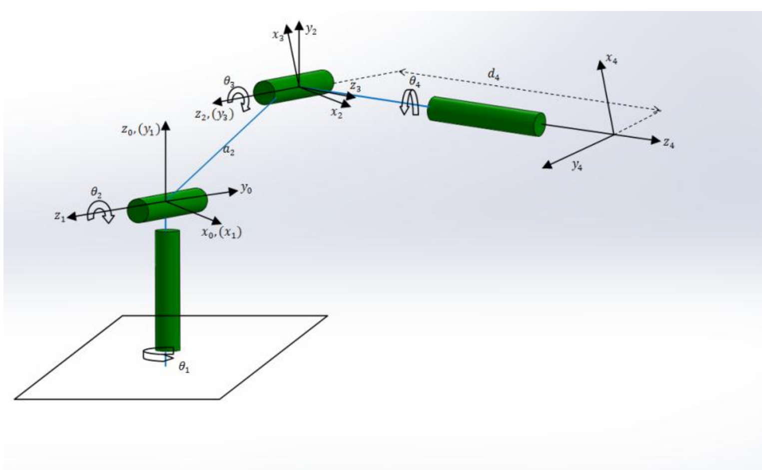

Figure 4 Robotic Arm Frame

The Denavit-Hartenberg parameters are specified in the table 1 below:

Table 1 Denavit-Hartenberg Parameters

Link	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	0	θ_1
2	a_2	0	0	θ_2
3	0	$\pi/2$	0	θ_3
4	0	0	d_4	θ_4

1.2 Rotation Matrix

"A rotation matrix is a matrix used to perform a rotation in 3-D space. It is constituted by three unit vectors **x′**, **y′**, **z′** which describe the body orientation with respect to the reference frame can be combined in the (3x3) matrix"[1].

$$\mathbf{R}=[\boldsymbol{x}' \quad \boldsymbol{y}' \quad \boldsymbol{z}']=\begin{bmatrix} x'_x & y'_x & z'_x \\ x'_y & y'_y & z'_y \\ x'_z & y'_z & z'_z \end{bmatrix} \tag{1.1}$$

Elementary-Rotation-Matrix represents position various in each axis direction, it is also the component of rotation-matrix in 3-D space. As figure 5 shows, the reference frame is O-xyz, and frame O'-x'y'z' is the rotated frame with angle α . The elementary rotation can be represented as follows

$$\boldsymbol{x}'=\begin{bmatrix} cos\alpha \\ sin\alpha \\ 0 \end{bmatrix} \qquad \boldsymbol{y}'=\begin{bmatrix} -sin\alpha \\ cos\alpha \\ 0 \end{bmatrix} \qquad \boldsymbol{z}'=\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{1.2}$$

Hence, the elementary rotation matrix of frame O-x′y′z′ with respect to frame O-xyz is

$$\mathbf{R}_z(\alpha)=\begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

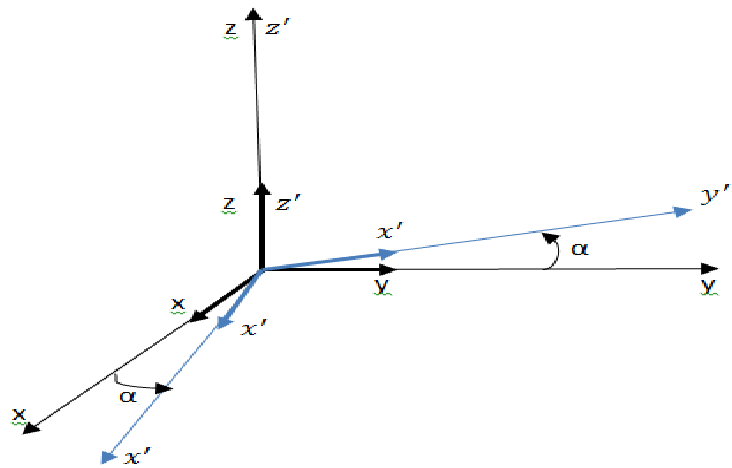

Figure 5 Rotation of the frame O-xyz by an angle about axis z

With a same concept, the rotations by an angle β about axis y and by an angle γ about axis x are respectively presented by

$$\mathbf{R}_y(\beta) = \begin{bmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix} \tag{1.4}$$

$$\mathbf{R}_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\gamma & -sin\gamma \\ 0 & sin\gamma & cos\gamma \end{bmatrix} \tag{1.5}$$

1.3 Homogeneous Transformation Matrix

In Fig 6, P is an arbitrary point in space. p^0 is the coordinate vector of P with respect to the reference frame $0(O_0\text{-}x_0y_0z_0)$. Frame $1(O_1\text{-}x_1y_1z_1)$ is the second frame in space. o_1^0 is the vector of frame 1's origin O_1 with respect to frame 0's origin O_0, and $R_1^0$is the rotation matrix of frame 1 with respect to frame 0. p^1 is the coordinate vector of P with respect to frame 1. On the basis of simple geometry, the position of point P with respect to the reference frame can be expressed as

$$p^0 = o_1^0 + R_1^0 p^1 \qquad (1.6)$$

Therefore, the position in space with two frames can be represented by translation term plus rotation term.

Based on the above explanation and equation 1.6, we can say the coordinate transformation can be represented by a 4x4 matrix which is called *homogeneous transformation matrix.*

$$A_1^0 = \begin{bmatrix} R_1^0 & o_1^0 \\ 0^T & 1 \end{bmatrix} \qquad (1.7)$$

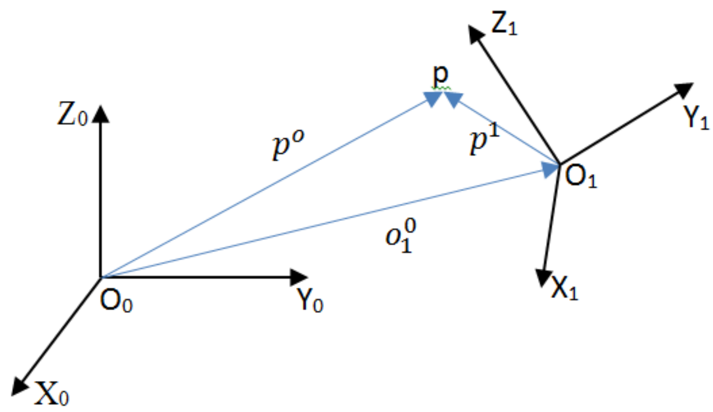

Figure 6 Representation of a point P in different coordinate frames

1.3.1 Calculation of Homogeneous Transformation

First, choose a frame aligned with frame i-1. Doing translation for the chosen frame by d_i along axis $z_{i-1}$and then doing rotation for it by θ_i about axis z_{i-1}.This sequence aligns the current frame i' and it has same process with Eq.(1.3). The homogeneous transformation matrix can be shown as

$$A_{i'}^{i-1} = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1.8)$$

Second, doing translation for the frame aligned with frame i' by a_i along axis $x_{i'}$ and doing rotation for current frame by α_i about axis $x_{i'}$; this sequence aligns the current frame i and it has the same process as Eq. (1.5). The transformation matrix is shown as following.

$$A_i^{i'} = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & c\alpha_i & -s\alpha_i & 0 \\ 0 & s\alpha_i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.9}$$

Multiplying Eq.(1.8) by Eq. (1.9), we can get the homogeneous transformation matrix from frame i to frame i-1.

$$A_i^{i-1}(q_i) = A_{i'}^{i-1} A_i^{i'} = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.10}$$

For the robotic arm, based on equation (1.10), the homogeneous transformation matrix for each single joint can be represented as follows.

$$\text{Link 1} \quad A_1^0(\theta_1) = \begin{bmatrix} C\theta_1 & 0 & S\theta_1 & 0 \\ S\theta_1 & 0 & -C\theta_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.11}$$

$$\text{Link 2} \quad A_2^1(\theta_2) = \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 & a_2 C\theta_2 \\ S\theta_2 & C\theta_2 & 0 & a_2 S\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.12}$$

$$\text{Link 3} \quad A_3^2(\theta_3) = \begin{bmatrix} C\theta_3 & 0 & S\theta_3 & 0 \\ S\theta_3 & 0 & -C\theta_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.13}$$

$$\text{Link 4} \qquad A_4^3(\theta_4)=\begin{bmatrix} C\theta_4 & -S\theta_4 & 0 & 0 \\ S\theta_4 & C\theta_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.14)$$

1.4 Direct Kinematic Function

"The aim of direct kinematics is to determine the end-effector position and orientation as a function of the joint variables".[1] The format of direct kinematic function can be expressed by a homogeneous transformation matrix with respect to the reference frame.

$$\boldsymbol{T}^0(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{n}^0(\boldsymbol{q}) & \boldsymbol{s}^0(\boldsymbol{q}) & \boldsymbol{a}^0(\boldsymbol{q}) & \boldsymbol{p}^0(\boldsymbol{q}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.15)$$

where, **q** is joint variables which is $n\times 1$ vector. **n**, **s**, **a** are unit vectors. **p** is the position vector of the origin frame with respect to the reference frame O_0-$x_0y_0z_0$.

"TheDenavit-Hartenberg convention allows constructing the direct kinematics function by composition of the individual coordinate transformations expressed by (1.7) into one homogeneous transformation matrix."[1] Therefore, multiplying each individual homogeneous transformation matrix together, we can get the direct kinematic function

$$T_n^0(q) = A_1^0(q_1)A_2^1(q_2)\ldots A_n^{n-1}(q_n) \qquad (1.16)$$

1.4.1 Solution of Direct Kinematic Function

Computation of the direct kinematic function as in equation (2.2) yields

$$T_4^0(q) = A_1^0(\theta_1)A_2^1(\theta_2)A_3^2(\theta_3)A_4^3(\theta_4)$$

Therefore,

$$T_4^0(q) = \begin{bmatrix} C1C2C3C4 - C1C2C3C4 + S1S4 & -C1C2C3S4 + C1S2S3S4 + S1C4 & C1C2S3 + C1S2C3 & C1C2S3d4 + C1S2C3d4 + C1C2a2 \\ S1C2C3C4 - S1S2S3C4 - C1S4 & -S1C2C3S4 + S1S2S3S4 - C1C4 & S1C2S3 + S1S2C3 & S1C2S3d4 + S1S2C3d4 + S1a2C2 \\ S2C3C4 + C2S3C4 & -S2C3S4 - C2S3S4 & S2S3 - C2C3 & S2S3d4 - C2C3d4 + a2S2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.17}$$

where q=$[\theta_1 \theta_2 \theta_3\ \theta_4]^T$, since z_4 align with z_3, frame 4 represents a end-effector frame.

1.5 INVERSE KINEMATICS PROBLEM

"The inverse kinematics consists of the determination of the joint variables corresponding to a given end-effector position and orientation."[1] The inverse kinematics solution is critical for control the manipulator. Since given the desired position and orientation of end-effector, the motion of each joint can be derived.

1.5.1 Solution of Inverse Kinematics

Consider the arm shown in Fig.4, the direct kinematics was given in equation (1.17). It is desired to find the joints variables $\theta_1,\theta_2,\theta_3$. The end-effector position p_x, p_y, p_z are given. The dimensions of a_2 and d_4 are constant.

$$\Theta_1=\tan^{-1}(\frac{p_y}{p_x}) \tag{1.18}$$

$$\theta_2=\tan^{-1}(\frac{\sin\theta_2}{\cos\theta_2}) \tag{1.19}$$

$$\theta_3=\tan^{-1}(\frac{\sin\theta_3}{\cos\theta_3}) \tag{1.20}$$

where,

$$\sin\theta_2=\frac{(a_2+d_4\cos\theta_3)p_z-d_4\sin\theta_3\sqrt[2]{(p_x)^2+(p_y)^2}}{(p_x)^2+(p_y)^2+(p_z)^2} \tag{1.21}$$

$$\cos\theta_2=\frac{(a_2+d_4\cos\theta_3)\sqrt[2]{(p_x)^2+(p_y)^2}+d_4p_x\sin\theta_3}{(p_x)^2+(p_y)^2+(p_z)^2} \tag{1.22}$$

$$\sin\theta_3=\pm\sqrt[2]{1-(\cos\theta_3)^2} \tag{1.23}$$

$$\cos\theta_3=\frac{(p_x)^2+(p_y)^2+(p_x)^2-a_2^2-d_4^2}{2a_2\ d_4} \tag{1.24}$$

Chapter 2. DIFFERENTIAL KINEMATICS ANALYSIS

In this section, differential kinematics expressed the relationship between end-effector velocity and each joint linear and angular velocity, which described by the Geometric Jacobian.

2.1 GEOMETRIC JACOBIAN

The differential kinematics represents the relationship between joint velocity $\dot{\boldsymbol{q}}$ and end-effector linear velocity $\dot{\boldsymbol{p}}$ and angular velocity $\boldsymbol{\omega}$. The relationship can be written as following two functions.

$$\dot{\boldsymbol{p}} = \mathbf{J}\mathrm{p}(\mathbf{q})\dot{\boldsymbol{q}} \tag{2.1}$$

$$\boldsymbol{\omega} = \mathbf{J}\mathrm{o}(\mathbf{q})\dot{\boldsymbol{q}} \tag{2.2}$$

In (2.1) **J**p is the (3xn) matrix, which is the jacobian for linear velocity. It expresses the relationship between the joint velocities $\dot{q}$ and the end-effector linear velocity $\dot{p}$, while in (2.2) Jo is the (3 x n) matrix, which is the jacobian of angular velocity. It expresses the relationship between the joint velocities $\dot{q}$ and the end-effector angular velocity ω.

Briefly, Equations (2.1) and (2.2) can be written as

$$\mathrm{V} = \begin{bmatrix} \dot{p} \\ \omega \end{bmatrix} = \mathrm{J(q)}\dot{q} \tag{2.3}$$

J is the 6xn matrix, which is the manipulator Geometry Jacobian.

$$\mathrm{J} = \begin{bmatrix} Jp \\ Jo \end{bmatrix} \tag{2.4}$$

2.1.1 Derivative of a Rotation Matrix

The robot motion relates translation and rotation. Considering the robot motion velocity, it is necessary to consider the first derivate of its displacement, in which the derivative of a rotation matrix is critical. Since the orthogonally of **R,** it has the relation

$$\mathbf{R}(\mathrm{t})\mathbf{R}^{T}(\mathrm{t}) = \mathbf{I} \tag{2.5}$$

Do the first derivate on both sides of the equation (2.5)

$$\dot{\boldsymbol{R}}(\mathrm{t})\mathbf{R}^{\mathrm{T}}(\mathrm{t}) + \mathbf{R}(\mathrm{t})\dot{\mathbf{R}}^{\mathrm{T}}(\mathrm{t}) = 0 \tag{2.6}$$

Set

$$\mathbf{S}(\mathrm{t}) = \dot{\mathbf{R}}(\mathrm{t})\mathbf{R}^{\mathrm{T}}(\mathrm{t}) \tag{2.7}$$

From equation (2.7) it is obviously showing that the (3 x 3) matrix **S(t)** is skew-symmetric (skew-symmetric matrix is a matrix that its transpose matrix equals to its negative matrix, which is $-\mathrm{A}=A^{T}$)

$$\mathbf{S}(\mathrm{t}) + \boldsymbol{S}^{T}(\mathrm{t}) = \mathbf{0} \tag{2.8}$$

Post-multiplying both sides of equation (2.7) by **R**(t) and we get the derivative of the rotation matrix.

$$\dot{\mathbf{R}}(\mathrm{t}) = \mathbf{S}(\mathrm{t})\mathbf{R}(\mathrm{t}) \tag{2.9}$$

If there is a constant vector **p**' and the vector **p**(t)=**R**(t)**p**'. Doing time derivative of p(t) is

$$\dot{\boldsymbol{p}}(\mathrm{t}) = \dot{\boldsymbol{R}}(\mathrm{t})\boldsymbol{p}' \tag{2.10}$$

Based on (2.9), substitute the derivative rotation matrix into equation (2.10)

$$\dot{\boldsymbol{p}}(t)=\mathbf{S}(t)\mathbf{R}(t)\boldsymbol{p}' \tag{2.11}$$

The vector $\boldsymbol{\omega}(t)$ is the angular velocity of frame $\mathbf{R}(t)$ respecting to the reference frame at time t, from mechanics point of view

$$\dot{\boldsymbol{p}}(t) = \boldsymbol{\omega}(t) \times \mathbf{R}(t)\boldsymbol{p}' \tag{2.12}$$

Hence, the matrix $\mathbf{S}(t)$ is the vector product between the vector $\boldsymbol{\omega}$ and vector $\mathbf{R}(t)p'$.

"The matrix $\mathbf{S}(t)$ is so that its symmetric elements with respect to the main diagonal represent the components of the vector $\boldsymbol{\omega}(t) = [\omega_x \quad \omega_y \quad \omega_z]^T$ in the form" [1]

$$S=\begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2.13}$$

Besides, we can get an important equation which will be used later.

$$\mathbf{RS}(\boldsymbol{\omega})\boldsymbol{R}^T = \mathbf{S}(\mathbf{R}\boldsymbol{\omega}) \tag{2.14}$$

Based on equation (1.6), transform a point p from frame 1 to frame 0, we can get

$$p^0 = o_1^0 + R_1^0 p^1 \tag{2.15}$$

Doing differential for (2.15) with respect to time gives

$$\dot{p}^0=\dot{0}_1^0+R_1^0\dot{p}^1+\dot{R}_1^0 p^1 \tag{2.16}$$

Then, based on equation (2.9), and substitute the rotation matrix derivative into equation (2.16) we can get

$$\dot{p}^0 = \dot{0}_1^0 + R_1^0 \dot{p}^1 + S(\omega_1^0) R_1^0 p^1 \quad (2.17)$$

Replacing the vector $R_1^0 p^1$ by r_1^0 ,

$$\dot{p}^0 = \dot{0}_1^0 + R_1^0 \dot{p}^1 + \omega_1^0 \times r_1^0 \quad (2.18)$$

2.1.2 Link Velocity

As shown in figure 8, joint i and joint i+1 are connected by link i; the frame i attaches on link i and its origin is along joint i+1; the frame i-1 attaches on link i too, while its origin is along joint i.

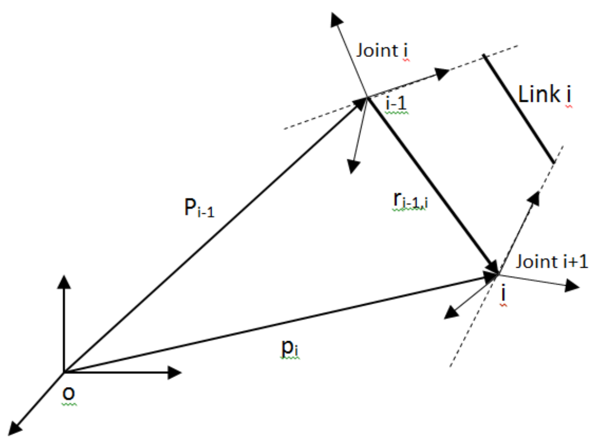

Figure 7 Characterization of generic link i of a manipulator

p_i is the position vector of origin of frame i with respect to origin frame o. Similarly, $p_{i-1}$is the position vector of origin of frame i-1 with respect to origin frame o. $r_{i-1,i}^{i-1}$ is the position of the origin of frame i with respect to frame i-1 expressed in frame i-1. Based on coordinate transformation Eq.(2.15), it can be written

$$p_i = p_{i-1} + R_{i-1} r_{i-1,i}^{i-1} \tag{2.19}$$

Then, based on equation (2.18), it is

$$\dot{p}_i = \dot{p}_{i-1} + R_{i-1} \dot{r}_{i-1,i}^{i-1} + \omega_{i-1} \times R_{i-1} r_{i-1,i}^{i-1}$$

$$\dot{p}_i = \dot{p}_{i-1} + v_{i-1,i} + \omega_{i-1} \times r_{i-1,i} \tag{2.20}$$

Equation (2.20) is the **linear velocity** of link i as a function of the translational and rotational velocities of link i-1. $v_{i-1,i}$ is the velocity of the origin of frame i with respect to the origin of frame i-1, expressed in the base frame.

Prismatic Joint. Because "orientation of frame i with respect to frame i-1 does not vary by moving joint i"[1], the angular velocity of link i with respect to link i-1 is

$$\omega_{i-1,i} = 0 \tag{2.21}$$

Further, the linear velocity is

$$v_{i-1,i} = \dot{d}_i z_{i-1} \tag{2.22}$$

Where z_{i-1} is the unit vector of joint i axis. Therefore, the angular velocity and linear velocity can be expressed as

$$\omega_i = \omega_{i-1} \tag{2.23}$$

$$\dot{p}_i = \dot{p}_{i-1} + \dot{d}_i z_{i-1,i} + \omega_i \times r_{i-1,i} \tag{2.24}$$

Revolute Joint. For the angular velocity it is

$$\omega_{i-1,i} = \dot{\theta}_i z_{i-1} \tag{2.25}$$

while for the linear velocity it is

$$v_{i-1,i} = \omega_i \text{ x } r_{i-1,i} \tag{2.26}$$

Therefore, the expressions of angular velocity (2.25) and linear velocity (2.20) of link i become

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + \dot{\theta}_i \boldsymbol{z}_{i-1}$$

$$\dot{\boldsymbol{p}}_i = \dot{\boldsymbol{p}}_{i-1} + \boldsymbol{\omega}_{i-1} \text{ x } \boldsymbol{r}_{i-1,i}$$

2.2 Jacobian Computation

"The term $\dot{q}_i Jp_i$ represents the contribution of single joint i to the end-effector linear velocity, while the term $\dot{q}_i Jo_i$ represents the contribution of single joint i to the end-effector angular velocity"[1]. For convenient computation, it is necessary to separate the calculation for prismatic joint and revolute joint.

For the angular velocity:

- If joint i is prismatic, from (2.21) it is

$$\dot{\mathbf{q}}_i \mathbf{J}o_i = 0 \tag{2.27}$$

and then

$$\mathbf{J}o_i=0 \tag{2.28}$$

- If joint i is revolute, from (2.25) it is

$$\dot{\mathbf{q}}_i \mathbf{J}o_i = \dot{\theta}_i \boldsymbol{z}_{i-1} \tag{2.29}$$

and then

$$\mathrm{J}o_i=\boldsymbol{z}_{i-1} \tag{2.30}$$

For the linear velocity:

- If joint i is prismatic, from (2.22) it is

$$\dot{\mathrm{q}}_i \mathrm{J}p_i = \dot{d}_i \boldsymbol{z}_{i-1} \tag{2.31}$$

and then

$$\mathrm{J}p_i=\boldsymbol{z}_{i-1} \tag{2.32}$$

- If joint i is revolute, it is

$$\dot{\mathbf{q}}_i \mathbf{J}p_i = \boldsymbol{\omega}_{i-1,i} \times \mathbf{r}_{i-1,i} \tag{2.33}$$

$$\dot{\mathbf{q}}_i \mathbf{J}p_i=\dot{\boldsymbol{\theta}}_i \boldsymbol{z}_{i-1} \times (\mathbf{p}\text{-}\mathbf{p}_{i-1}) \tag{2.34}$$

and then

$$\mathbf{J}p_i=\boldsymbol{z}_{i-1} \times (\mathbf{p}\text{-}\mathbf{p}_{i-1}) \tag{2.35}$$

In sum :

$$\begin{bmatrix} \mathbf{J}p_i \\ \mathbf{J}o_i \end{bmatrix} = \begin{cases} \begin{bmatrix} \mathbf{z}_{i-1} \\ 0 \end{bmatrix} & \text{for a prismatic joint} \\ \begin{bmatrix} \mathbf{z}_{i-1} \times (\mathbf{p} - \mathbf{p}_{i-1}) \\ \mathbf{z}_{i-1} \end{bmatrix} & \text{for a revolute joint} \end{cases} \tag{2.36}$$

Where, the vectors $\mathbf{z}_{i-1}$, $\mathbf{p}$, and $\mathbf{p}_{i-1}$ are all functions of the joint variables. In particular:

- z_{i-1} is the third column of the rotation matrix R_{i-1}^0
- p is the first three elements of the fourth column of the transformation matrix T_n^0.
- p_{i-1} is the first three elements of the fourth column of the transformation matrix T_n^0.

2.2.1 SOLUTION OF JACOBIAN OF THE 6 DEGREE OF FREEDOM ROBOT ARM

In this case, all the links 1, 2, 3, 4 are revolute joints. From (2.41) the jacobian is

$$J = \begin{bmatrix} z_0(p - p_0) & z_1(p - p_1) & z_2(p - p_2) & z_3(p - p_3) \\ z_0 & z_1 & z_2 & z_3 \end{bmatrix} \tag{2.37}$$

The transformation matrices are:

$$T_1^0 = A_1^0 = \begin{array}{c} \begin{array}{cccc} \quad & R_1^0 & \quad z_1 & \quad p_1 \end{array} \\ \begin{bmatrix} C\theta_1 & 0 & S\theta_1 & 0 \\ S\theta_1 & 0 & -C\theta_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

$$
\begin{array}{cccc} & R_2^0 & z_2 & p_2 \end{array}
$$

$$
T_2^0 = A_2^0 = A_1^0 A_2^1 = \begin{bmatrix} \mathrm{C1C2} & -\mathrm{C1S2} & \mathrm{S1} & \mathrm{a1C1C2} \\ S1C2 & -\mathrm{S1S2} & -\mathrm{C1} & \mathrm{a2S1C2} \\ \mathrm{S2} & \mathrm{C2} & 0 & \mathrm{a2S2} \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
\begin{array}{cccc} & R_3^0 & z_3 & p_3 \end{array}
$$

$$
T_3^0 = A_3^0 = A_1^0 A_2^1 A_3^2 = \begin{bmatrix} \mathrm{C1C2C3 - C1S2S3} & \mathrm{S1} & \mathrm{C1C2S3 + C1S2C3} & \mathrm{a1C1C2} \\ \mathrm{S1C2C3 - S1S2S3} & -\mathrm{C1} & \mathrm{S1C2S3 + S1S2C3} & \mathrm{a2S1C2} \\ \mathrm{S2C3 + C2S3} & 0 & \mathrm{S2S3 - C2C3} & \mathrm{a2S2} \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
\mathrm{T}_4^0(\mathrm{q}) = \qquad \mathrm{p}
$$

$$
\begin{bmatrix} \mathrm{C1C2C3C4 - C1C2C3C4 + S1S4} & \mathrm{-C1C2C3S4 + C1S2S3S4 + S1C4} & \mathrm{C1C2S3 + C1S2C3} & \mathrm{C1C2S3d4 + C1S2C3d4 + C1C2a2} \\ \mathrm{S1C2C3C4 - S1S2S3C4 - C1S4} & \mathrm{-S1C2C3S4 + S1S2S3S4 - C1C4} & \mathrm{S1C2S3 + S1S2C3} & \mathrm{S1C2S3d4 + S1S2C3d4 + S1a2C2} \\ \mathrm{S2C3C4 + C2S3C4} & \mathrm{-S2C3S4 - C2S3S4} & \mathrm{S2S3 - C2C3} & \mathrm{S2S3d4 - C2C3d4 + a2S2} \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Therefore

$$
p_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad ; \quad p_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad ; \quad p_2 = \begin{bmatrix} a2C1C2 \\ a2S1C2 \\ a2S2 \end{bmatrix} \quad ; \quad p_0 = \begin{bmatrix} a2C1C2 \\ a2S1C2 \\ a2S2 \end{bmatrix}
$$

$$
\mathrm{P} = \begin{bmatrix} \mathrm{C1C2S3d4 + C1S2C3d4 + C1C2a2} \\ \mathrm{S1C2S3d4 + S1S2C3d4 + S1a2C2} \\ \mathrm{S2S3d4 - C2C3d4 + a2S2} \end{bmatrix}
$$

$$z_0=\begin{bmatrix}0\\0\\1\end{bmatrix} \quad ; \quad z_1=\begin{bmatrix}S1\\-C1\\0\end{bmatrix} \quad ; \quad z_2=\begin{bmatrix}S1\\-C1\\0\end{bmatrix} \quad ; \quad z_3=\begin{bmatrix}C1C2S3+C1S2C3\\S1C2S3+S1S2C3\\S2S3-C2C3\end{bmatrix}$$

Chapter 3. DYNAMICS ANALYSIS

"Dynamic model of a manipulator provides a description of the relationship between the joint actuator torques and the motion of the structure".[1] "Derivation of the dynamic model of a manipulator plays an important role for a simulation of motion, analysis of manipulator structures, and design of control algorithm"[1]. "Computation of the forces and torques required for the execution of typical motions provides useful information for designing joints, drives and actuators".[1] In this section, the Lagrange formulation method is used to derive the equations of motion of a manipulator in the joint space.

3.1 LAGRANGE FORMULATION

"With Lagrange formulation, the equations of motion can be derived in a systematic way independently of the reference coordinate frame". [1]"The Lagrangian of the mechanical system can be defined as a function of the generalized coordinates"[1]:

$$L = T - U \tag{3.1}$$

where T represents the total kinetic energy, and U represents the total potential energy.

Equation (3.2) is the Lagrange equation

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\lambda_i}} - \frac{\partial L}{\partial \lambda_i} = \xi_i \quad i = 1,\ldots,n \tag{3.2}$$

Where ξ_i is the generalized force associated with the generalized coordinate λ_i.

3.1.1 Computation of Kinetic Energy

Link Dynamics

The total kinetic energy equals the summation of kinetic energy of links and kinetic energy of motors.

$$T = \sum_{i=1}^{n}(T_{li} + T_{mi}) \tag{3.3}$$

T_{li} is the kinetic energy of link i and T_{mi} is the kinetic energy of the motor actuating joint i.

The kinetic energy of link is

$$T_{li} = \frac{1}{2}\int_{v_{li}} \dot{p}_i^{*T}\, \dot{p}_i^{*} \rho dV \tag{3.4}$$

$\dot{p}_i^{*}$ is the linear velocity vector and ρ is the density of the elementary particle of volume dV; Vl_i is the volume of link i.

p_i^* is the position vector of the elementary particle and $p_{li}$is the position vector of the link center of mass, both expressed in the base frame, which are shown in equation (3.5) and figure 9.

$$r_i = [r_{ix} r_{iy} r_{iz}]^T = p_i^* - p_{li} \tag{3.5}$$

The link point velocity can be expressed as

$$\dot{p}_i^* = \dot{p}_{li} + \omega_i \times r_i \tag{3.6}$$

$$\dot{p}_i^* = \dot{p}_{li} + S(\omega_i) r_i \tag{3.7}$$

Where $\dot{p}_{li}$ is the linear velocity of the center of mass and ω_i is the angular velocity of the link (Fig. 9)

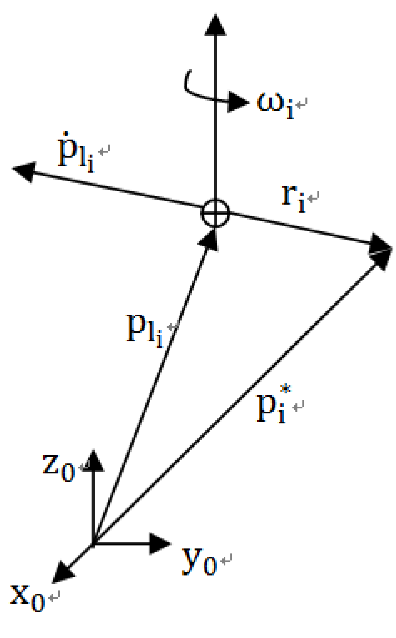

Figure 8 Kinematic description of link i for Lagrange formulation

Substitute the equation (3.7) into equation (3.4), the kinetic energy of each link can be separated into translation contribution and rotation contribution.

Translation contribution

$$\frac{1}{2}\int_{v_{li}} \dot{p}_{li}^{T}\,\dot{p}_{li}\rho dV = \frac{1}{2}m_{li}\dot{p}_{li}^{T}\dot{p}_{li} \tag{3.8}$$

Rotation contribution

$$\frac{1}{2}\int_{v_{li}} r_i^{T}S^{T}(\omega_i)S(\omega_i)\,r_i\rho dV = \frac{1}{2}\omega_i^{T}I_{li}\omega_i \tag{3.9}$$

The matrix

$$I_{l_i} = \begin{bmatrix} \int(r_{iy}^2 + r_{iz}^2)\rho dV & -\int r_{ix}r_{iy}\,\rho dV & -\int r_{ix}r_{iz}\,\rho dV \\ -\int r_{ix}r_{iy}\,\rho dV & \int(r_{ix}^2 + r_{iz}^2)\rho dV & -\int r_{iy}r_{iz}\,\rho dV \\ -\int r_{ix}r_{iz}\,\rho dV & -\int r_{iy}r_{iz}\,\rho dV & \int(r_{ix}^2 + r_{iy}^2)\rho dV \end{bmatrix}$$

$$I_{l_i} = \begin{bmatrix} I_{lixx} & -I_{lixy} & -I_{lixz} \\ I_{lixx} & I_{liyy} & -I_{liyz} \\ -I_{lixz} & -I_{liyz} & I_{lizz} \end{bmatrix} \tag{3.10}$$

Summing the translational and rotational contributions (3.8) and (3.9), the kinetic energy of link i is

$$T_{li} = \frac{1}{2}m_{li}\dot{p}_{li}^{T}\dot{p}_{li} + \frac{1}{2}\omega_i^{T}R_i I_{li}R_i^{T}\omega_i \tag{3.11}$$

“To express the kinetic energy as a function of the generalized coordinate of the system”[1] Jacobian computation can be applied as an intermediate between links velocity and end-effector velocity.

$$\dot{p}_{li} = J_{p1}^{li}\dot{q}_1 + \cdots + J_{pi}^{li}\dot{q}_i = J_p^{li}\dot{q} \tag{3.12}$$

$$\omega_i = J_{O1}^{li}\dot{q}_1 + \cdots + J_{Oi}^{li}\dot{q}_i = J_O^{li}\dot{q} \tag{3.13}$$

The Jacobians are then:

$$\mathbf{J}_p^{li} = [J_{p1}^{li} \dots J_{pi}^{li}\mathbf{0} \dots \mathbf{0}] \tag{3.14}$$

$$\mathbf{J}_O^{li} = [J_{O1}^{li} \dots J_{Oi}^{li}\mathbf{0} \dots \mathbf{0}] \tag{3.15}$$

based on equation (2.41), we can calculate the columns of (3.14) and (3.15)

$$\mathbf{J}_p^{li} = \begin{cases} z_{j-1} & \text{for a prismatic joint} \\ z_{j-1} \times (p_{li} - p_{j-1}) & \text{for a revolute joint} \end{cases} \tag{3.16}$$

$$\mathbf{J}_O^{li} = \begin{cases} 0 & \text{for a prismatic joint} \\ z_{j-1} & \text{for a revolute joint} \end{cases} \tag{3.17}$$

Where $p_{j-1}$is the position vector of joint j-1, and z_{j-1} is the unit vector of axis z of frame j-1.

Summary, the kinetic energy of link i in (3.11) can be written as

$$T_{li} = \frac{1}{2}m_{li}\dot{q}^T J_p^{(li)T} J_p^{(li)}\dot{q} + \frac{1}{2}\omega_i^T I_{li}\omega_i + \frac{1}{2}\dot{q}^T J_O^{(li)T} I_{li} J_O^{(li)}\dot{q} \tag{3.18}$$

Motor Dynamics

The method to calculate motor dynamics is same with the method for calculation link dynamics. Assuming the motor i is located on link i-1.

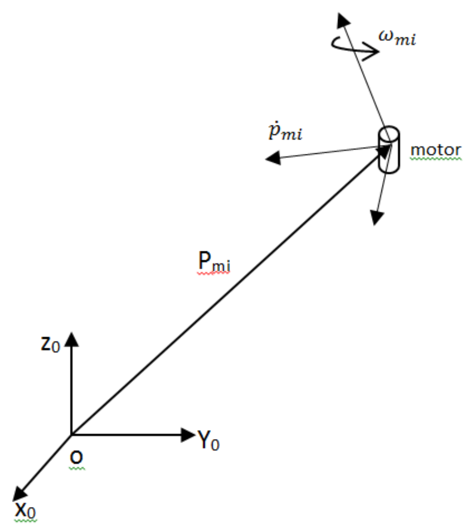

Figure 9 Kinematic description of motor i

The kinetic energy of rotor i can be written as

$$T_{mi} = \frac{1}{2} m_{mi} \dot{p}_{mi}^T \dot{p}_{mi} + \frac{1}{2} \omega_{mi}^T I_{mi} \omega_{mi} \tag{3.19}$$

where m_{mi} is the mass of the rotor, $\dot{p}_{mi}$ is the linear velocity of the center of mass of the rotor, I_{mi} is the inertia tensor of the motor relative to its center of mass, and ω_{mi} is the angular velocity of the rotor.

Assuming rigid transmission and k_{ri} is gear reduction ratio, the angular velocity of motor is

$$k_{ri} \dot{q}_i = \dot{\theta}_{mi} \tag{3.20}$$

Then, the total angular velocity of the motor is

$$\omega_{mi} = \omega_{i-1} + k_{ri}\dot{q}_i z_{mi} \tag{3.21}$$

where ω_{i-1} is the angular velocity of link i-1,and the motor is mounted on link i-1, and z_{mi} is the unit vector along the motor axis.

The linear velocity of center mass of motor is

$$\dot{p}_{mi} = J_p^{mi}\dot{q} \tag{3.22}$$

The Jacobian is

$$\mathbf{J}_{\mathrm{p}}^{\mathrm{mi}} = [\mathrm{J}_{\mathrm{p1}}^{\mathrm{mi}} \ldots \mathrm{J}_{\mathrm{p,i-1}}^{\mathrm{mi}} \mathbf{0} \ldots \mathbf{0}] \tag{3.23}$$

the columns are

$$\mathbf{J}_{p_j}^{\mathrm{mi}} = \begin{cases} \mathrm{z}_{\mathrm{j-1}} & \text{for a prismatic joint} \\ \mathrm{z}_{\mathrm{j-1}} \times (\mathrm{p}_{\mathrm{mi}} - \mathrm{p}_{\mathrm{j-1}}) & \text{for a revolute joint} \end{cases} \tag{3.24}$$

where p_{j-1} is the position vector of the joint j-1. Because the center of mass of the motor is along its axis of rotation, J_{pi}^{mi} equals zero in (3.23).

The angular velocity of center of motor is,

$$\omega_{mi} = J_0^{mi}\dot{q} \tag{3.25}$$

The Jacobian is

$$\mathbf{J}_{O}^{mi} = [\mathrm{J}_{O1}^{mi} \ldots \mathrm{J}_{O,i-1}^{mi} \, \mathrm{J}_{Oi}^{mi}\mathbf{0} \ldots \mathbf{0}\,] \tag{3.26}$$

whose columns, are respectively given by

$$\mathbf{J}_{O}^{mi} = \begin{cases} J_{Oj}^{li} & j = 1, \ldots, i-1 \\ k_{ri} z_{mi} & j = i \end{cases} \tag{3.27}$$

Hence, the kinetic energy of rotor i can be written as

$$\mathrm{T}_{mi} = \frac{1}{2}\mathrm{m}_{mi}\dot{\mathrm{q}}^{T}\mathrm{J}_{p}^{(mi)T}\mathrm{J}_{p}^{(mi)}\dot{\mathrm{q}} + \frac{1}{2}\dot{\mathrm{q}}^{T}\mathrm{J}_{O}^{(mi)T}\mathrm{I}_{mi}\mathrm{J}_{O}^{(mi)}\dot{\mathrm{q}} \tag{3.28}$$

Finally, adding the kinetic energy of single links (3.18) and single motor motors (3.28) togeter, the total kinetic energy is

$$T = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}\,(q)\dot{q}_i\dot{q}_j = \frac{1}{2}\dot{\boldsymbol{q}}^{T}\boldsymbol{B}(q)\dot{\boldsymbol{q}} \tag{3.29}$$

in which, **B**(q) is the $(n \times n)$ inertia matrix:

$$\mathbf{B}(\mathrm{q}) = \sum_{i=1}^{n}(\mathrm{m}_{li}\mathrm{J}_{p}^{(li)T}\mathrm{J}_{p}^{(li)} + \mathrm{J}_{O}^{(li)T}\mathrm{I}_{li}\mathrm{J}_{O}^{(li)} + \mathrm{m}_{mi}\mathrm{J}_{p}^{(mi)T}\mathrm{J}_{p}^{(mi)} + \mathrm{J}_{O}^{(mi)T}\mathrm{I}_{mi}\mathrm{J}_{O}^{(mi)}) \tag{3.30}$$

3.1.2 Computation of Potential Energy

The potential energy of the manipulator equals the sum of the potential energy of each link and motor:

$$U = \sum_{i=1}^{n}(U_{li} + Um_i) \tag{3.31}$$

Assuming rigid links and motors, the contribution due to gravitational forces are expressed by

$$u_{li} = -m_{li} g_0^T p_{li} \tag{3.32}$$

$$u_{mi} = -m_{mi} g_o^T p_{mi} \tag{3.33}$$

where, g_o is the gravity acceleration vector in the base frame.

By substituting (3.32) and (3.33) into (3.31), the potential energy is given by

$$U = -\sum_{i=1}^{n} (m_{li} g_0^T p_{li} + m_{mi} g_o^T p_{mi}) \tag{3.34}$$

3.2 Equation of Motion

After calculating the total kinetic and potential energy of the system, the Lagrangian (3.1) for the manipulator can be written as

$$\begin{aligned} \mathrm{L}(\mathrm{q}, \dot{q}) = {} & \mathrm{T}(\mathrm{q}, \dot{q}) - \mathrm{U}(\mathrm{q}) \\ & = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij}(q) \dot{q}_i \dot{q}_j \\ & + \sum_{i=1}^{n} \left(m_{li} g_0^T p_{li}(q) + m_{mi} g_o^T p_{mi}(q) \right) \end{aligned} \tag{3.35}$$

Taking the derivatives required by Lagrange equations (3.2)

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) = \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) = \sum_{j=1}^{n} b_{ij}(q)\ddot{q}_j + \sum_{j=1}^{n} \frac{db_{ij}(q)}{dt}\dot{q}_j$$

$$= \sum_{j=1}^{n} b_{ij}\,(q)\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial b_{ij}(q)}{\partial qk}\dot{q}_k\dot{q}_j$$

and

$$\frac{\partial T}{\partial q_i} = \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial b_{jk}(q)}{\partial q_i}\dot{q}_k\dot{q}_j \tag{3.36}$$

$$\frac{\partial u}{\partial q_i} = -\sum_{j=1}^{n}\left(m_{lj}g_0^T\frac{\partial p_{lj}}{\partial q_i} + m_{mj}g_0^T\frac{\partial p_{mj}}{\partial q_i}\right)$$

$$= -\sum_{j=1}^{n}\left(m_{li}g_0^T J_{pi}^{li}(q) + m_{mj}g_o^T J_{pi}^{mi}(q)\right) \quad (3.37)$$

As a result, the equations of motion are

$$\sum_{j=1}^{n} b_{ij}\,(q)\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} h_{ijk}\,(q)\dot{q}_k\dot{q}_j + g_i(q) = \xi_i \qquad i = 1, \dots, n \tag{3.38}$$

where

$$h_{ijk} = \frac{\partial b_{ij}}{\partial q_k} - \frac{1}{2}\frac{\partial b_{jk}}{\partial q_i} \tag{3.39}$$

Followings are explanation for equation (3.38),

- For the acceleration terms:

  The coefficient b_{ij} represents the moment of inertia.

- For the quadratic velocity terms:

  The term $h_{ijj}\dot{q}_j^2$ represent the centrifugal effect on joint i by velocity of joint j;

  The term $h_{ijk}\dot{q}_j\dot{q}_k$ represents the Coriolis effect on joint i by velocity of joints j and k.

- For the configuration-dependent terms:

  The term g_i represents the gravity effect.

3.3 Solution of Dynamics Model

Consider the arm in figure.1, the vector of generalized coordinates is q=$[\theta_1\theta_2\theta_3\theta_4]^T$. Lets l_1, l_2 ,l_3, l_4 be the distance of the center of mass of the two links from the respective joint axes. Let also $m_{l1}, m_{l2}, m_{l3}, m_{l4}$ be the masses of the links, and $m_{m1}, m_{m2}, m_{m3}, m_{m4}$ the masses of the rotors of the joint motors. Let I_{m1}, I_{m2}, I_{m3} , I_{m4} be the moments of inertia with respect to the axes of the rotors, and $I_{l1}, I_{l2}, I_{l3}, I_{l4}$ are the moments of inertia relative to the center of mass of the links, respectively. It is assumed that $p_{mi} = p_{i-1}$. The motors are located on the joints axes with centers of mass located at the origins of the respective frames.

With the chosen coordinate frames, computation of the link Jacobian in (3.16) and (3.17) yields

Calculation of J_p^{l1}

$$J_{p1}^{l1} = z_0 \times (p_{l1} - p_0)$$

$$p_{l1} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad p_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$p_{l1} - p_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$J_{p1}^{l1} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$J_p^{l1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Calculation of J_p^{l2}. As shown is Figure 11.

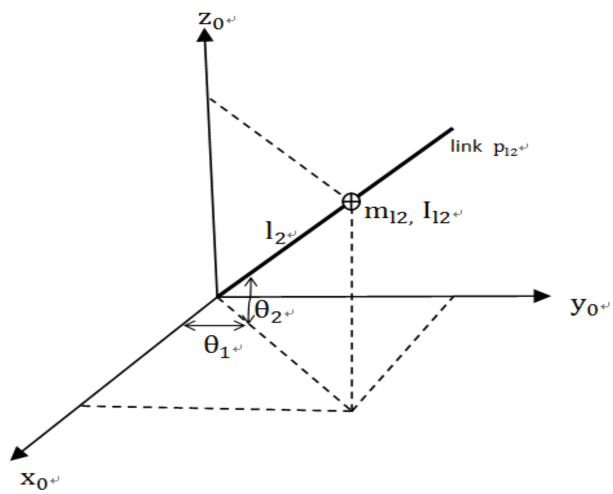

Figure 10 Calculation of J_p^l2

$$\mathbf{J}_{p1}^{l2} = \mathbf{z}_0 \times (\mathbf{p}_{l2} - \mathbf{p}_0)$$

$$= \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} l_2 c_2 c_1 - 0 \\ l_2 c_2 s_1 - 0 \\ l_2 s_2 - 0 \end{bmatrix}$$

$$= \begin{bmatrix} l_2 c_2 s_1 \\ l_2 c_2 c_1 \\ 0 \end{bmatrix}$$

$$\mathbf{J}_{p2}^{l2} = \mathbf{z}_1 \times (\mathbf{p}_{l2} - \mathbf{p}_1)$$

$$= \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix} \times \begin{bmatrix} l_2 c_2 c_1 - 0 \\ l_2 c_2 s_1 - 0 \\ l_2 s_2 - 0 \end{bmatrix}$$

$$=\begin{bmatrix} -l_2 c_1 s_2 \\ l_2 s_1 s_2 \\ l_2 c_2 \end{bmatrix}$$

$$J_p^{l2} = \begin{bmatrix} J_{p2}^{l2} & J_{p2}^{l2} & 0 & 0 \end{bmatrix}$$

$$=\begin{bmatrix} l_2 c_2 c_1 & l_2 c_2 c_1 & 0 & 0 \\ l_2 c_2 s_1 & l_2 c_2 s_1 & 0 & 0 \\ 0 & l_2 s_2 & 0 & 0 \end{bmatrix}$$

Calculation of J_p^{l3}. As shown in figure 12

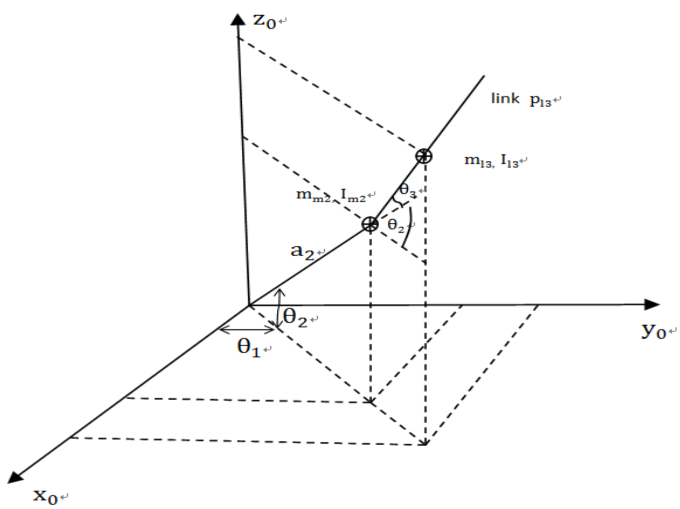

Figure 11 Calculation of J_p^l3

$$\mathbf{J}_{p1}^{l3} = \mathbf{z}_0 \times (\mathbf{p}_{l3} - \mathbf{p}_0)$$

$$= \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} a_2c_2c_2 - l_3c_{32}c_1 - 0 \\ a_2c_2s_1 + l_3c_{32}s_1 - 0 \\ a_2s_2 + l_3s_{32} - 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_2c_2s_1 - l_3c_{32}s_1 \\ a_2c_2c_1 + l_3c_{32}c_1 \\ 0 \end{bmatrix}$$

$$J_{p2}^{l3} = z_1 \times (p_{l3} - p_1)$$

$$= \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix} \times \begin{bmatrix} a_2c_2c_2 + l_3c_{32}c_1 - 0 \\ a_2c_2s_1 + l_3c_{32}s_1 - 0 \\ a_2s_2 + l_3s_{32} - 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_2s_2c_1 - l_3s_{32}c_1 \\ -a_2s_2s_1 - l_3s_{32}s_1 \\ a_2c_2 + l_3c_{32} \end{bmatrix}$$

$$J_{p3}^{l3} = z_2 \times (p_{l3} - p_2)$$

$$= \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix} \times \begin{bmatrix} a_2c_2c_1 + l_3c_{32}c_1 - a_2c_2c_1 \\ a_2c_2s_1 + l_3c_{32}s_1 - a_2c_2s_1 \\ a_2s_2 + l_3s_{32} - a_2s_2 \end{bmatrix}$$

$$= \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix} \times \begin{bmatrix} l_3c_{32}c_1 \\ l_3c_{32}s_1 \\ l_3s_{32} \end{bmatrix}$$

$$= \begin{bmatrix} -l_3 s_{32} c_1 \\ -l_3 s_{32} s_1 \\ l_3 c_{32} \end{bmatrix}$$

$$J_p^{l3} = \begin{bmatrix} J_{p1}^{l3} & J_{p2}^{l3} & J_{p3}^{l3} & 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_2 c_2 s_1 - l_3 c_{32} s_1 & -a_2 s_2 c_1 - l_3 s_{32} c_1 & -l_3 s_{32} c_1 & 0 \\ a_2 c_2 c_1 + l_3 c_{32} c_1 & -a_2 s_2 s_1 - l_3 s_{32} s_1 & -l_3 s_{32} s_1 & 0 \\ 0 & a_2 c_2 + l_3 c_{32} & l_3 c_{32} & 0 \end{bmatrix}$$

Calculation of J_p^{l4}. As shown in figure 13

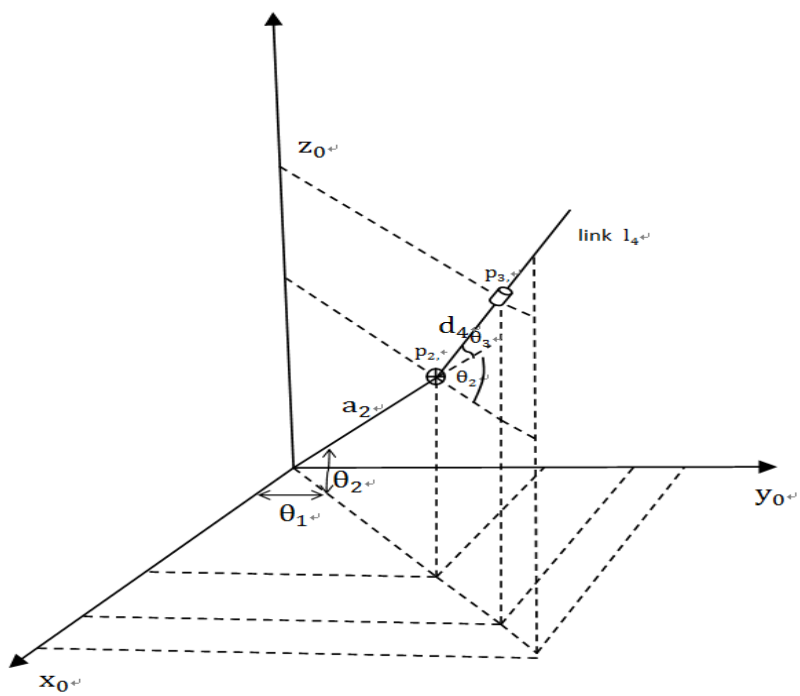

Figure 12 13Calculation of J_p^l4

$$J_{p1}^{l4} = z_0 \times (p_{l4} - p_0)$$

$$= \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} a_2c_2c_1 + d_4c_{32}c_1 + l_4c_{32}c_1 - 0 \\ a_2c_2s_1 + d_4c_{32}s_1 + l_4c_{32}s_1 - 0 \\ a_2s_2 + d_4s_{32} + l_4s_{32} - 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_2c_2s_1 - d_4c_{32}s_1 - l_4c_{32}s_1 \\ a_2c_2c_1 + d_4c_{32}c_1 + l_4c_{32}c_1 \\ 0 \end{bmatrix}$$

$$J_{p2}^{l4} = z_0 \times (p_{l4} - p_1)$$

$$= \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix} \times \begin{bmatrix} a_2c_2c_1 + d_4c_{32}c_1 + l_4c_{32}c_1 - 0 \\ a_2c_2s_1 + d_4c_{32}s_1 + l_4c_{32}s_1 - 0 \\ a_2s_2 + d_4s_{32} + l_4s_{32} - 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_2s_2c_1 - d_4s_{32}c_1 - l_4s_{32}c_1 \\ -a_2s_2s_1 - d_4s_{32}s_1 - l_4s_{32}s_1 \\ a_2c_2 + d_4c_{32} + l_4c_{32} \end{bmatrix}$$

$$J_{p3}^{l4} = z_2 \times (p_{l4} - p_2)$$

$$= \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix} \times \begin{bmatrix} a_2c_2c_1 + d_4c_{32}c_1 + l_4c_{32}c_1 - a_2c_2c_1 \\ a_2c_2s_1 + d_4c_{32}s_1 + l_4c_{32}s_1 - a_2c_2s_1 \\ a_2s_2 + d_4s_{32} + l_4s_{32} - a_2s_2 \end{bmatrix}$$

$$= \begin{bmatrix} -d_4s_{32}c_1 - l_4s_{32}c_1 \\ -d_4s_{32}s_1 - l_4s_{32}s_1 \\ d_4c_{32} + l_4c_{32} \end{bmatrix}$$

$$J_{p4}^{l4} = z_3 \times (p_{l4} - p_3)$$

$$= \begin{bmatrix} c_1c_2c_3 + c_1s_2c_3 \\ s_1c_2s_3 + s_1s_2c_3 \\ s_2s_3 - c_2c_3 \end{bmatrix} \times \begin{bmatrix} l_4c_{32}c_1 \\ l_4c_{32}s_1 \\ l_4s_{32} \end{bmatrix}$$

$$= \begin{bmatrix} l_4s_1c_2s_3s_{32} + l_4s_1s_2c_3s_{32} - l_4s_2s_3c_{32}s_1 + l_4c_2c_3c_{32}s_1 \\ l_4s_2s_3c_{32}c_1 - l_4c_1c_2c_3c_{32} - l_4c_1c_2c_3s_{32} - l_4c_1s_2c_3s_{32} \\ l_4c_1c_2c_3c_{32} + l_4c_1s_2c_3s_1 - l_4s_2s_3c_{32}c_1 + l_4c_1c_2c_3c_{32} \end{bmatrix}$$

$$J_p^{l4} = \begin{bmatrix} J_{p1}^{l4} & J_{p2}^{l4} & J_{p3}^{l4} & J_{p4}^{l4} \end{bmatrix}$$

$$= \begin{bmatrix} -a_2c_2s_1 - d_4c_{32}s_1 - l_4c_{32}s_1 & -a_2s_2c_1 - d_4s_{32}c_1 - l_4s_{32}c_1 & -d_4s_{32}c_1 - l_4s_{32}c_1 & l_4s_1c_2s_3s_{32} + l_4s_1s_2c_3s_{32} - l_4s_2s_3c_{32}s_1 + l_4c_2c_3c_{32}s_1 \\ a_2c_2c_1 + d_4c_{32}c_1 + l_4c_{32}c_1 & -a_2s_2s_1 - d_4s_{32}s_1 - l_4s_{32}s_1 & -d_4s_{32}s_1 - l_4s_{32}s_1 & l_4s_2s_3c_{32}c_1 - l_4c_1c_2c_3c_{32} - l_4c_1c_2c_3s_{32} - l_4c_1s_2c_3s_{32} \\ 0 & a_2c_2 + d_4c_{32} + l_4c_{32} & d_4c_{32} + l_4c_{32} & l_4c_1c_2c_3c_{32} + l_4c_1s_2c_3s_1 - l_4s_2s_3c_{32}c_1 + l_4c_1c_2c_3c_{32} \end{bmatrix}$$

The computation of the Jacobian in (3.15) and (3.17) yields

$$J_{o1}^{l1} = z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J_o^{l1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$J_{o1}^{l2} = z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J_{o2}^{l2} = z_1 = \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix}$$

$$J_o^{l2} = \begin{bmatrix} 0 & s_1 & 0 & 0 \\ 0 & -c_1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$J_o^{l3} = \begin{bmatrix} 0 & s_1 & s_1 & 0 \\ 0 & -c_1 & -c_1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$J_o^{l4} = \begin{bmatrix} 0 & s_1 & s_1 & c_1c_2c_3 + c_1s_2c_3 \\ 0 & -c_1 & -c_1 & s_1c_2s_3 + s_1s_2c_3 \\ 1 & 0 & 0 & s_2s_3 - c_2c_3 \end{bmatrix}$$

Computation of the Jacobian in (3.23) and (3.24) yields

$$J_p^{m1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$J_{p1}^{m2} = z_0 \times (p_{m2} - p_0)$$

$$= \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} 0 - 0 \\ 0 - 0 \\ 0 - 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$J_p^{m2} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$J_{p1}^{m3} = z_0 \times (p_{m3} - p_0)$$

$$= \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} a_2c_1c_2 - 0 \\ a_2c_2s_1 - 0 \\ a_2s_2 - 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_2c_2s_1 \\ a_2c_2c_1 \\ 0 \end{bmatrix}$$

$$J_{p2}^{m3} = z_1 \times (p_{m3} - p_1)$$

$$= \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix} \times \begin{bmatrix} a_2c_1c_2 - 0 \\ a_2c_2s_1 - 0 \\ a_2s_2 - 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_2s_2c_1 \\ -a_2s_2s_1 \\ a_2c_2 \end{bmatrix}$$

$$J_p^{m3} = \begin{bmatrix} -a_2c_2s_1 & -a_2s_2c_1 & 0 & 0 \\ a_2c_2c_1 & -a_2s_2s_1 & 0 & 0 \\ 0 & a_2c_2 & 0 & 0 \end{bmatrix}$$

$$J_{p1}^{m4} = z_0 \times (p_{m4} - p_0)$$

$$= \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} a_2c_2c_1 + d_4c_{32}c_1 - 0 \\ a_2c_2s_1 + d_4c_{32}s_1 - 0 \\ a_2s_2 + d_4s_{32} - 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_2c_2s_1 - d_4c_{32}s_1 \\ a_2c_2c_1 + d_4c_{32}c_1 \\ 0 \end{bmatrix}$$

$$J_{p2}^{m4} = z_1 \times (p_{m4} - p_1)$$

$$= \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix} \times \begin{bmatrix} a_2c_2c_1 + d_4c_{32}c_1 - 0 \\ a_2c_2s_1 + d_4c_{32}s_1 - 0 \\ a_2s_2 + d_4s_{32} - 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_2s_2c_1 - d_4s_{32}c_1 \\ -a_2s_1s_2 - d_4s_{32}s_1 \\ a_2c_2 + d_4c_{32} \end{bmatrix}$$

$$J_{p3}^{m4} = z_2 \times (p_{m4} - p_2)$$

$$= \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix} \times \begin{bmatrix} a_2c_2c_1 + d_4c_{32}c_1 - a_2c_2c_1 \\ a_2c_2s_1 + d_4c_{32}s_1 - a_2c_2s_1 \\ a_2s_2 + d_4s_{32} - a_2s_2 \end{bmatrix}$$

$$= \begin{bmatrix} -d_4s_{32}c_1 \\ -d_4s_{32}s_1 \\ d_4s_{32}c_1 \end{bmatrix}$$

$$J_p^{m4} = \begin{bmatrix} -a_2c_2s_1 - d_4c_{32}s_1 & -a_2s_2c_1 - d_4s_{32}c_1 & -d_4s_{32}c_1 & 0 \\ a_2c_2c_1 + d_4c_{32}c_1 & -a_2s_1s_2 - d_4s_{32}s_1 & -d_4s_{32}s_1 & 0 \\ 0 & a_2c_2 + d_4c_{32} & d_4s_{32}c_1 & 0 \end{bmatrix}$$

The computation of the Jacobian in (3.26) and (3.27) yields

$$J_o^{m1} = [J_{01}^{m1} \quad 0 \quad 0 \quad 0]$$

$$J_o^{m1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ k_{r1} & 0 & 0 & 0 \end{bmatrix}$$

$$J_{o1}^{m2} = J_{o1}^{l2} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J_{o2}^{m2} = k_{r2} z_{m2} = k_{r2} \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix} = \begin{bmatrix} k_{r2} s_1 \\ -k_{r2} c_1 \\ 0 \end{bmatrix}$$

$$J_o^{m2} = [J_{o1}^{m2} \quad J_{o2}^{m2} \quad 0 \quad 0]$$

$$= \begin{bmatrix} 0 & k_{r2} s_1 & 0 & 0 \\ 0 & -k_{r2} c_1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$J_{o1}^{m3} = J_{o1}^{l3} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J_{o2}^{m3} = J_{o2}^{l3} = \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix}$$

$$J_{o3}^{m3} = k_{r3} z_{m3} = k_{r3} \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix}$$

$$J_o^{m3} = [J_{o1}^{m3} \quad J_{o2}^{m3} \quad J_{o2}^{m3} \quad 0]$$

$$= \begin{bmatrix} 0 & s_1 & k_{r3} s_1 & 0 \\ 0 & -c_1 & -k_{r3} c_1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$J_{o1}^{m4} = J_{o1}^{l4} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J_{o2}^{m4} = J_{o2}^{l4} = \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix}$$

$$J_{o3}^{m4} = J_{o3}^{l4} = \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix}$$

$$J_{o4}^{m4} = k_{r4} z_{m4} = k_{r4} \begin{bmatrix} c_1 c_2 c_3 + c_1 s_2 c_3 \\ s_1 c_2 s_3 + s_1 s_2 c_3 \\ s_2 s_3 - c_2 c_3 \end{bmatrix}$$

$$J_o^{m4} = [J_{o1}^{m4} \quad J_{o2}^{m4} \quad J_{o3}^{m4} \quad J_{o4}^{m4}]$$

$$= \begin{bmatrix} 0 & s_1 & s_1 & k_{r4} c_1 c_2 c_3 + k_{r4} c_1 s_2 c_3 \\ 0 & -c_1 & -c_1 & k_{r4} s_1 c_2 s_3 + k_{r4} s_1 s_2 c_3 \\ 1 & 0 & 0 & k_{r4} s_2 s_3 - k_{r4} c_2 c_3 \end{bmatrix}$$

Based on the Eq.(3.30), the inertia matrix can be calculated. From (3.38) and (3.39) the result of equation of motion can be got. In this case, since the calculation of Inertia Matrix and Equation of Motion are tedious, it is advisable to perform it with the aid of program software.

Inertia Matrix

$$\mathrm{B(q)} = \begin{bmatrix} \mathrm{b}_{11} & \mathrm{b}_{12} & \mathrm{b}_{13} & \mathrm{b}_{14} \\ \mathrm{b}_{21} & \mathrm{b}_{22} & \mathrm{b}_{23} & \mathrm{b}_{24} \\ \mathrm{b}_{31} & \mathrm{b}_{32} & \mathrm{b}_{33} & \mathrm{b}_{34} \\ \mathrm{b}_{41} & \mathrm{b}_{42} & \mathrm{b}_{43} & \mathrm{b}_{44} \end{bmatrix}$$

Equation of Motion

$$\sum_{j=1}^{n} b_{ij}\,(q)\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} h_{ijk}\,(q)\dot{q}_k\dot{q}_j + g_i(q) = \xi_i \qquad i = 1, \dots, n$$

$$T_1 = \xi_1 T_2 = \xi_2 T_3 = \xi_3 T_4 = \xi_4$$

The results of inertia matrix and torques on each joint are shown in Appendix 1 and Appendix 2.

Torque Results Test

Assuming the all the length of links are unit 1; distance of a_2 and d_4 are unit 1; $\theta_1 = 0$, $\theta_2 = \frac{1}{3}\pi$, $\theta_3 = \frac{1}{2}\pi$, $\theta_4 = 0$; $\dot{\theta}_1 = 0$, $\dot{\theta}_2 = 0.1$, $\dot{\theta}_3 = 0.2$, $\dot{\theta}_4 = 0$; $\ddot{\theta}_1 = 0$, $\ddot{\theta}_2 = 0.6$, $\ddot{\theta}_3 = 0.4$, $\ddot{\theta}_4 = 0$; all the links and motors mass are unit 1; the moment of inertia are unit 1. Substitude these values into the equation of motion, and get the values of torque.

Torque 1 equals 4.2883; Torque 2 equals 24.8279; Torque 3 equals 37.678; Torques 4 equals 0.534.

Chapter 4. Position Accuracy Reliability Analysis

Due to the serial chain structure, one error will lead to a series error happening. Therefore, the main technological barriers of robotic arm are the position error between the target position and real position. Besides, duo to their serial structure, low stiffness is another weakness. As a result, the industry requirements of large loads and high accuracy are limited. In this chapter, firstly it defines the robotic error and error resource, then it analyses the robot accuracy. Finally, it calculates the probability density function for the accuracy reliability.

4.1 Robotic Error

There are two criteria to define the error sources of robot. One is pose repeatability and the other is pose accuracy [3]. Repeatability is a measure of the ability of the robot to move back to the same position and orientation. Accuracy is defined as the ability of the robot to precisely move to a desired position in 3_D space. Figure 14 shows these concepts graphically.

Figure 13Robotic Accuracy and Repeatability

4.2 Error Sources

Since robots are mechanical devices, errors can be caused by slight changes due to manufacturing tolerance, which are "changes of working environment temperature, wear of parts and component replacement"[4]. In the other words, the end-effector pose is not determined by the joints of a single joint-link train but by all joint-link trains. In summary, Figure 15 [4] shows kinds of errors which affect the robot accuracy.

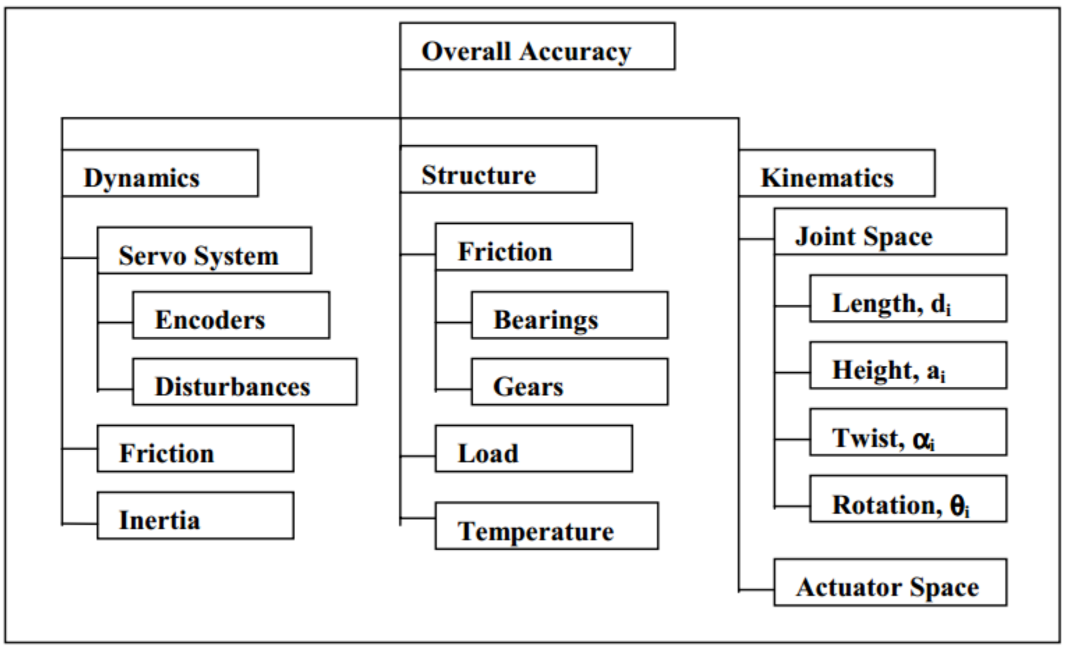

Figure 14 Error Tree

4.3 Robot Accuracy Analysis

To achieve high accuracy, “the kinematic model used in the controller, should be a faithful mathematical description of the relationship, which related the end-effector pose to the individual joint values”. [3] In other words, the accuracy of a robotic arm to reach a specified location in a space depends on the Denavit-Hartenberg parameters of each joint. Table 2 shows the error modeling by kinematic parameters.

Table 2 Error Sources defined by Kinematic Parameters

T	Parameter	Error Sources
T_1	θ_1	Motor accuracy (gear friction, gear backlash, etc.)
T_2	θ_2, a_2	Motor accuracy (gear friction, gear backlash, etc.); Link length(manufacturing tolerance, over load bending)
T_3	θ_3	Motor accuracy (gear friction, gear backlash, etc.)
T_4	θ_4, d_4	Motor accuracy (gear friction, gear backlash, etc.); Link length(manufacturing tolerance, over load bending)

4.4 Accuracy Reliability Calculation

To calculate the reliability, it mainly uses lognormal distribution and probability function. The calculation process shows in follows.

First, create 100 random values which the range is from 0 to 10, which assume the data collecting by experiment. These values represent the distance (in centimeter) from goal target point to the points that the end effector arrives. Then find the mean value M, and the variance is V. Then we can get the standard deviation shown in (Eq. 4.1) and zero mean μ.

$$\sigma = \sqrt{\log(\frac{V}{M^2} + 1)} \tag{4.1}$$

Substitute the σ and μ into the lognormal distribution function

$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{[-\frac{1}{2}(\frac{\ln x-\mu}{\sigma})^2]} \tag{4.2}$$

Then, substitute the lognormal distribution function into the probability density function

$$p(a \le x \le b) = \int_a^b f(x)dx \tag{4.3}$$

We can get the probability lognormal distribution density function

$$p(a \le x \le b) = \int_a^b \frac{1}{x\sigma\sqrt{2\pi}} e^{[-\frac{1}{2}(\frac{\ln x-\mu}{\sigma})^2]} dx \tag{4.4}$$

Based on equation (4.4), we can apply it on accuracy reliability analysis. For example:

We get the 100 experiment data (replaced by 100 random data ranging from 0-10 created by MatLab). From these results, the mean value (M) is 4.95 and Variance (V) is 6.62.We set a=0, b=2, which means acceptable error ranging from 0 cm to 2 cm. Then, the accuracy probability result is 23.9%. (Notes: this value is not the actural experiment result value.) Figure 16 shows the graph of lognormal distribution.

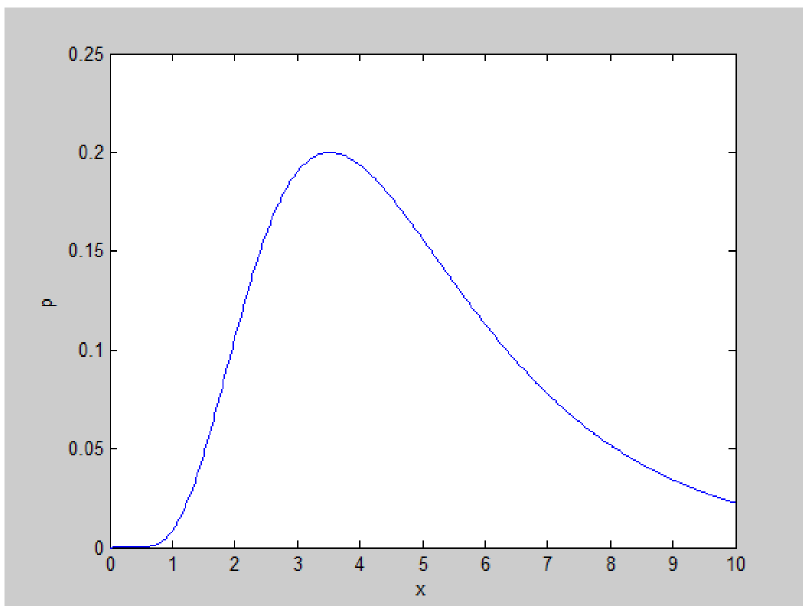

Figure 15 Lognormal Distribution graph

CONCLUSION

In this article, it is about the modeling of the 6 Degree-Of-Freedom robotic arm. It describes the kinematic analysis and dynamic analysis. For the kinematic analysis, it is separated into two parts, one is kinematics which determines the position and orientation relationship between joints and end-effector. In kinematics, the Danavit-Haryenberg Parameters, Homogeneous Transformation Matrix and Direct Kinematic Function are derived. The other is Differential Kinematics, which determines the velocity relationship between joints and end-effector. In Dynamic analysis, the equation of motion is derived based on the Lagrange Equation, which describes the relationship between the joint actuator torques and the motion of the structure. Finally, the robotic position accuracy analysis is done by means of lognormal distribution density function. And the position accuracy analysis is based on the kinematic parameters.

REFERENCE

1. Sciavicco, Lorenzo, and Bruno Siciliano. *Modeling and Control of Robot Manipulators*. New York [u.a.: McGraw-Hill, 1996. Print.

2. Kazuhiro Kosuge, Katsuhisa Furuta. *KINEMATIC AND DYNAMIC ANALYSIS OF ROBOT ARM***.**Tokyo Institute of Technology Oh-Okayama, Meguro-ku, Tokyo, JAPAN

3. Mohamed Abderrahim, Alla Khamis, Santiago Garrido, Luis Moreno University Carlos III of Madrid, Spain. *Accuracy and Calibration Issues of Industrial Manipulators.*

4. Kevin L. Conrad, Panayiotis S. Shiakolas. Robotic Calibration Issues: Accuracy, Repeatability and Calibration.

5. KOK-MENG LEE AND DHARMAN K. SHAH. *Dynamic Analysis of a Three-Degrees-of-Freedom In-Parallel Actuated Manipulator* .IEEE JOURNAL OF ROBOTICS AND AUTOMATION, VOL. 4, NO. 3, JUNE 1988

6. C.S.G.Lee, B.H.Lee, R.Nlgam. *An Efficient Formulation of Robot Arm Dynamics For Control Analysis And Manipulator Design.*Department of Electrical and Computer Engineering, The University of Michigan, Ann Arbor, Michigan April 1982.

7. K, Nishiwaki, T. Sugihara, S. Kagami, F.Kanehiro, M.Inaba, H.Inoue . *"Design and Development of Research Platform for Perception-action Integration in Humanoid Robot: H6.*" *IEEE Xplore*. N.p., Mar. 2000.

8. Hayati, Samad A. "Robot Arm Geometric Link Parameter Estimation." *Robot Arm Geometric Link Parameter Estimation*. N.p., n.d.

9. Mashesh D. Pandey, Xufang Zhang. Department of Civil and Environment Engineering, University of Waterloo, ON, Canada N2L 3G1. *System Reliability Analysis of the robotic Manipulator With Random Joint Clearances.2012.*

10. Zhao Liang, Su Meng, Miao Yunchen. Department of Mechanical Engineering, Zhejiang University, Hangzhou, China. *High Accuracy Analysis of Scara Industrial Robot Based ON Screw Theory.*

APPENDICES 1

Values of Inertia Matrix

b_{11}=

IL1zz + IL2zz + IL3zz + IL4zz + Im2zz + Im3zz + Im4zz + mL3*(L3*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2))^2 + mm4*(d4*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2))^2 + mL3*(L3*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1))^2 + Im1zz*Kr1^2 + mm4*(d4*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1))^2 + mL4*(L4*cos(theta2 + theta3)*cos(theta1) + d4*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2))^2 + mL4*(L4*cos(theta2 + theta3)*sin(theta1) + d4*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1))^2 + a2^2*mm3*cos(theta2)^2*sin(theta1)^2 + L2^2*mL2*cos(theta1)^2*cos(theta2)^2 + L2^2*mL2*cos(theta2)^2*sin(theta1)^2 + a2^2*mm3*cos(theta1)^2*cos(theta2)^2

b_{12} =

IL2yz*cos(theta1) + IL3yz*cos(theta1) + IL4yz*cos(theta1) + Im3yz*cos(theta1) + Im4yz*cos(theta1) - IL2xz*sin(theta1) - IL3xz*sin(theta1) - IL4xz*sin(theta1) - Im3xz*sin(theta1) - Im4xz*sin(theta1) + mL3*(L3*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1))*(L3*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2)) - mm4*(d4*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2))*(d4*sin(theta2 + theta3)*sin(theta1) + a2*sin(theta1)*sin(theta2)) + mm4*(d4*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1))*(d4*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2)) - mL4*(L4*cos(theta2 + theta3)*cos(theta1) + d4*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2))*(L4*sin(theta2 + theta3)*sin(theta1) + d4*sin(theta2 + theta3)*sin(theta1) + a2*sin(theta1)*sin(theta2)) + mL4*(L4*cos(theta2 + theta3)*sin(theta1) + d4*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1))*(L4*sin(theta2 + theta3)*cos(theta1) + d4*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2)) + Im2yz*Kr2*cos(theta1) - Im2xz*Kr2*sin(theta1) - mL3*(L3*sin(theta2 + theta3)*cos(theta1) + a2*sin(theta1)*sin(theta2))*(L3*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2)) - L2^2*mL2*cos(theta1)^2*cos(theta2)*sin(theta2) + L2^2*mL2*cos(theta2)*sin(theta1)^2*sin(theta2)

b_{13}=

IL3yz*cos(theta1) + IL4yz*cos(theta1) + Im4yz*cos(theta1) - IL3xz*sin(theta1) - IL4xz*sin(theta1) - Im4xz*sin(theta1) - mL4*(L4*sin(theta2 + theta3)*sin(theta1) + d4*sin(theta2 + theta3)*sin(theta1))*(L4*cos(theta2 + theta3)*cos(theta1) + d4*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2)) + mL4*(L4*sin(theta2 + theta3)*cos(theta1) + d4*sin(theta2 + theta3)*cos(theta1))*(L4*cos(theta2 + theta3)*sin(theta1) + d4*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1)) + Im3yz*Kr3*cos(theta1) - Im3xz*Kr3*sin(theta1) - L3*mL3*sin(theta2 + theta3)*sin(theta1)*(L3*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2)) + L3*mL3*sin(theta2 + theta3)*cos(theta1)*(L3*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1)) - d4*mm4*sin(theta2 + theta3)*sin(theta1)*(d4*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2)) + d4*mm4*sin(theta2 + theta3)*cos(theta1)*(d4*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1))

b_{14} =

- IL4yz*(cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2)) - IL4zz*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) - Im4xz*(Kr4*cos(theta1)*cos(theta2)*cos(theta3) + Kr4*cos(theta1)*cos(theta3)*sin(theta2)) - Im4yz*(Kr4*cos(theta2)*sin(theta1)*sin(theta3) + Kr4*cos(theta3)*sin(theta1)*sin(theta2)) - Im4zz*(Kr4*cos(theta2)*cos(theta3) - Kr4*sin(theta2)*sin(theta3)) -

IL4xz*(cos(theta1)*cos(theta2)*cos(theta3) + cos(theta1)*cos(theta3)*sin(theta2)) - mL4*(L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) + 2*L4*sin(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) - L4*cos(theta2 + theta3)*cos(theta1)*sin(theta2)*sin(theta3))*(L4*cos(theta2 + theta3)*cos(theta1) + d4*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2)) - mL4*(L4*cos(theta2 + theta3)*sin(theta1) + d4*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1))*(L4*sin(theta2 + theta3)*cos(theta2)*sin(theta1)*sin(theta3) - L4*cos(theta2 + theta3)*sin(theta1)*sin(theta2)*sin(theta3) + L4*sin(theta2 + theta3)*cos(theta3)*sin(theta1)*sin(theta2) + L4*cos(theta2 + theta3)*cos(theta2)*cos(theta3)*sin(theta1))

$b_{21} =$

IL2yz*cos(theta1) + IL3yz*cos(theta1) + IL4yz*cos(theta1) + Im3yz*cos(theta1) + Im4yz*cos(theta1) - IL2xz*sin(theta1) - IL3xz*sin(theta1) - IL4xz*sin(theta1) - Im3xz*sin(theta1) - Im4xz*sin(theta1) + mL3*(L3*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1))*(L3*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2)) - mm4*(d4*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2))*(d4*sin(theta2 + theta3)*sin(theta1) + a2*sin(theta1)*sin(theta2)) + mm4*(d4*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1))*(d4*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2)) - mL4*(L4*cos(theta2 + theta3)*cos(theta1) + d4*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2))*(L4*sin(theta2 + theta3)*sin(theta1) + d4*sin(theta2 + theta3)*sin(theta1) + a2*sin(theta1)*sin(theta2)) + mL4*(L4*cos(theta2 + theta3)*sin(theta1) + d4*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1))*(L4*sin(theta2 + theta3)*cos(theta1) + d4*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2)) + Im2yz*Kr2*cos(theta1) - Im2xz*Kr2*sin(theta1) - mL3*(L3*sin(theta2 + theta3)*cos(theta1) + a2*sin(theta1)*sin(theta2))*(L3*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2)) - L2^2*mL2*cos(theta1)^2*cos(theta2)*sin(theta2) + L2^2*mL2*cos(theta2)*sin(theta1)^2*sin(theta2)

$b_{22} =$

mL3*(L3*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2))^2 + mL3*(L3*sin(theta2 + theta3)*cos(theta1) + a2*sin(theta1)*sin(theta2))^2 + mm4*(d4*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2))^2 + mL3*(L3*cos(theta2 + theta3) - a2*cos(theta2))^2 + mm4*(d4*sin(theta2 + theta3)*sin(theta1) + a2*sin(theta1)*sin(theta2))^2 + mL4*(L4*sin(theta2 + theta3)*cos(theta1) + d4*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2))^2 + mm4*(d4*sin(theta2 + theta3) + a2*cos(theta2))^2 + cos(theta1)*(IL2yy*cos(theta1) + IL2xy*sin(theta1)) + cos(theta1)*(IL3yy*cos(theta1) + IL3xy*sin(theta1)) + cos(theta1)*(IL4yy*cos(theta1) + IL4xy*sin(theta1)) + cos(theta1)*(Im3yy*cos(theta1) + Im3xy*sin(theta1)) + cos(theta1)*(Im4yy*cos(theta1) + Im4xy*sin(theta1)) + sin(theta1)*(IL2xy*cos(theta1) + IL2xx*sin(theta1)) + sin(theta1)*(IL3xy*cos(theta1) + IL3xx*sin(theta1)) + sin(theta1)*(IL4xy*cos(theta1) + IL4xx*sin(theta1)) + sin(theta1)*(Im3xy*cos(theta1) + Im3xx*sin(theta1)) + sin(theta1)*(Im4xy*cos(theta1) + Im4xx*sin(theta1)) + mL4*(L4*sin(theta2 + theta3)*sin(theta1) + d4*sin(theta2 + theta3)*sin(theta1) + a2*sin(theta1)*sin(theta2))^2 + mL4*(L4*cos(theta2 + theta3) + d4*cos(theta2 + theta3) + a2*cos(theta2))^2 + Kr2*cos(theta1)*(Im2yy*Kr2*cos(theta1) + Im2xy*Kr2*sin(theta1)) + Kr2*sin(theta1)*(Im2xy*Kr2*cos(theta1) + Im2xx*Kr2*sin(theta1)) + L2^2*mL2*cos(theta2)^2 + a2^2*mm3*cos(theta2)^2 + L2^2*mL2*sin(theta1)^2*sin(theta2)^2 + a2^2*mm3*cos(theta1)^2*sin(theta2)^2 + a2^2*mm3*sin(theta1)^2*sin(theta2)^2 + L2^2*mL2*cos(theta1)^2*sin(theta2)^2

$b_{23} =$

cos(theta1)*(IL3yy*cos(theta1) + IL3xy*sin(theta1)) + cos(theta1)*(IL4yy*cos(theta1) + IL4xy*sin(theta1)) + cos(theta1)*(Im4yy*cos(theta1) + Im4xy*sin(theta1)) + sin(theta1)*(IL3xy*cos(theta1) + IL3xx*sin(theta1)) +

sin(theta1)*(IL4xy*cos(theta1) + IL4xx*sin(theta1)) + sin(theta1)*(Im4xy*cos(theta1) + Im4xx*sin(theta1)) + Kr3*sin(theta1)*(Im3xy*cos(theta1) + Im3xx*sin(theta1)) + mL4*(L4*sin(theta2 + theta3)*cos(theta1) + d4*sin(theta2 + theta3)*cos(theta1))*(L4*sin(theta2 + theta3)*cos(theta1) + d4*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2)) + mL4*(L4*sin(theta2 + theta3)*sin(theta1) + d4*sin(theta2 + theta3)*sin(theta1))*(L4*sin(theta2 + theta3)*sin(theta1) + d4*sin(theta2 + theta3)*sin(theta1) + a2*sin(theta1)*sin(theta2)) + mL4*(L4*cos(theta2 + theta3) + d4*cos(theta2 + theta3))*(L4*cos(theta2 + theta3) + d4*cos(theta2 + theta3) + a2*cos(theta2)) + Kr3*cos(theta1)*(Im3yy*cos(theta1) + Im3xy*sin(theta1)) - L3*mL3*cos(theta2 + theta3)*(L3*cos(theta2 + theta3) - a2*cos(theta2)) + d4*mm4*sin(theta2 + theta3)*sin(theta1)*(d4*sin(theta2 + theta3)*sin(theta1) + a2*sin(theta1)*sin(theta2)) + d4*mm4*sin(theta2 + theta3)*cos(theta1)*(d4*sin(theta2 + theta3) + a2*cos(theta2)) + L3*mL3*sin(theta2 + theta3)*cos(theta1)*(L3*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2)) + L3*mL3*sin(theta2 + theta3)*sin(theta1)*(L3*sin(theta2 + theta3)*cos(theta1) + a2*sin(theta1)*sin(theta2)) + d4*mm4*sin(theta2 + theta3)*cos(theta1)*(d4*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2))

$b_{24} =$

(cos(theta1)*cos(theta2)*cos(theta3) + cos(theta1)*cos(theta3)*sin(theta2))*(IL4xy*cos(theta1) + IL4xx*sin(theta1)) - (Kr4*cos(theta2)*cos(theta3) - Kr4*sin(theta2)*sin(theta3))*(Im4yz*cos(theta1) - Im4xz*sin(theta1)) - (cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2))*(IL4yy*cos(theta1) + IL4xy*sin(theta1)) - (IL4yz*cos(theta1) - IL4xz*sin(theta1))*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) + (Im4xy*cos(theta1) + Im4xx*sin(theta1))*(Kr4*cos(theta1)*cos(theta2)*cos(theta3) + Kr4*cos(theta1)*cos(theta3)*sin(theta2)) - (Kr4*cos(theta2)*sin(theta1)*sin(theta3) + Kr4*cos(theta3)*sin(theta1)*sin(theta2))*(Im4yy*cos(theta1) + Im4xy*sin(theta1)) + mL4*(L4*cos(theta2 + theta3) + d4*cos(theta2 + theta3) + a2*cos(theta2))*(L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) - L4*cos(theta2 + theta3)*cos(theta1)*sin(theta2)*sin(theta3) + L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) + L4*cos(theta2 + theta3)*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2)) + mL4*(L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) + 2*L4*sin(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) - L4*cos(theta2 + theta3)*cos(theta1)*sin(theta2)*sin(theta3))*(L4*sin(theta2 + theta3)*sin(theta1) + d4*sin(theta2 + theta3)*sin(theta1) + a2*sin(theta1)*sin(theta2)) - mL4*(L4*sin(theta2 + theta3)*cos(theta1) + d4*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2))*(L4*sin(theta2 + theta3)*cos(theta2)*sin(theta1)*sin(theta3) - L4*cos(theta2 + theta3)*sin(theta1)*sin(theta2)*sin(theta3) + L4*sin(theta2 + theta3)*cos(theta3)*sin(theta1)*sin(theta2) + L4*cos(theta2 + theta3)*cos(theta2)*cos(theta3)*sin(theta1))

$b_{31} =$

IL3yz*cos(theta1) + IL4yz*cos(theta1) + Im4yz*cos(theta1) - IL3xz*sin(theta1) - IL4xz*sin(theta1) - Im4xz*sin(theta1) - mL4*(L4*sin(theta2 + theta3)*sin(theta1) + d4*sin(theta2 + theta3)*sin(theta1))*(L4*cos(theta2 + theta3)*cos(theta1) + d4*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2)) + mL4*(L4*sin(theta2 + theta3)*cos(theta1) + d4*sin(theta2 + theta3)*cos(theta1))*(L4*cos(theta2 + theta3)*sin(theta1) + d4*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1)) + Im3yz*Kr3*cos(theta1) - Im3xz*Kr3*sin(theta1) - L3*mL3*sin(theta2 + theta3)*sin(theta1)*(L3*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2)) + L3*mL3*sin(theta2 + theta3)*cos(theta1)*(L3*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1)) - d4*mm4*sin(theta2 + theta3)*sin(theta1)*(d4*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2)) + d4*mm4*sin(theta2 + theta3)*cos(theta1)*(d4*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1))

$b_{32} =$

cos(theta1)*(Im3yy*Kr3*cos(theta1) + Im3xy*Kr3*sin(theta1)) + sin(theta1)*(Im3xy*Kr3*cos(theta1) + Im3xx*Kr3*sin(theta1)) + cos(theta1)*(IL3yy*cos(theta1) + IL3xy*sin(theta1)) + cos(theta1)*(IL4yy*cos(theta1) + IL4xy*sin(theta1)) + cos(theta1)*(Im4yy*cos(theta1) + Im4xy*sin(theta1)) + sin(theta1)*(IL3xy*cos(theta1) + IL3xx*sin(theta1)) + sin(theta1)*(IL4xy*cos(theta1) + IL4xx*sin(theta1)) + sin(theta1)*(Im4xy*cos(theta1) + Im4xx*sin(theta1)) + mL4*(L4*sin(theta2 + theta3)*cos(theta1) + d4*sin(theta2 + theta3)*cos(theta1))*(L4*sin(theta2 + theta3)*cos(theta1) + d4*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2)) + mL4*(L4*sin(theta2 + theta3)*sin(theta1) + d4*sin(theta2 + theta3)*sin(theta1))*(L4*sin(theta2 + theta3)*sin(theta1) + d4*sin(theta2 + theta3)*sin(theta1) + a2*sin(theta1)*sin(theta2)) + mL4*(L4*cos(theta2 + theta3) + d4*cos(theta2 + theta3))*(L4*cos(theta2 + theta3) + d4*cos(theta2 + theta3) + a2*cos(theta2)) - L3*mL3*cos(theta2 + theta3)*(L3*cos(theta2 + theta3) - a2*cos(theta2)) + d4*mm4*sin(theta2 + theta3)*sin(theta1)*(d4*sin(theta2 + theta3)*sin(theta1) + a2*sin(theta1)*sin(theta2)) + d4*mm4*sin(theta2 + theta3)*cos(theta1)*(d4*sin(theta2 + theta3) + a2*cos(theta2)) + L3*mL3*sin(theta2 + theta3)*cos(theta1)*(L3*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2)) + L3*mL3*sin(theta2 + theta3)*sin(theta1)*(L3*sin(theta2 + theta3)*cos(theta1) + a2*sin(theta1)*sin(theta2)) + d4*mm4*sin(theta2 + theta3)*cos(theta1)*(d4*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2))

$b_{33} =$

mL4*(L4*sin(theta2 + theta3)*cos(theta1) + d4*sin(theta2 + theta3)*cos(theta1))^2 + mL4*(L4*sin(theta2 + theta3)*sin(theta1) + d4*sin(theta2 + theta3)*sin(theta1))^2 + mL4*(L4*cos(theta2 + theta3) + d4*cos(theta2 + theta3))^2 + cos(theta1)*(IL3yy*cos(theta1) + IL3xy*sin(theta1)) + cos(theta1)*(IL4yy*cos(theta1) + IL4xy*sin(theta1)) + cos(theta1)*(Im4yy*cos(theta1) + Im4xy*sin(theta1)) + sin(theta1)*(IL3xy*cos(theta1) + IL3xx*sin(theta1)) + sin(theta1)*(IL4xy*cos(theta1) + IL4xx*sin(theta1)) + sin(theta1)*(Im4xy*cos(theta1) + Im4xx*sin(theta1)) + L3^2*mL3*cos(theta2 + theta3)^2 + Kr3*cos(theta1)*(Im3yy*Kr3*cos(theta1) + Im3xy*Kr3*sin(theta1)) + Kr3*sin(theta1)*(Im3xy*Kr3*cos(theta1) + Im3xx*Kr3*sin(theta1)) + L3^2*mL3*sin(theta2 + theta3)^2*cos(theta1)^2 + L3^2*mL3*sin(theta2 + theta3)^2*sin(theta1)^2 + 2*d4^2*mm4*sin(theta2 + theta3)^2*cos(theta1)^2 + d4^2*mm4*sin(theta2 + theta3)^2*sin(theta1)^2

$b_{34} =$

(cos(theta1)*cos(theta2)*cos(theta3) + cos(theta1)*cos(theta3)*sin(theta2))*(IL4xy*cos(theta1) + IL4xx*sin(theta1)) - (Kr4*cos(theta2)*cos(theta3) - Kr4*sin(theta2)*sin(theta3))*(Im4yz*cos(theta1) - Im4xz*sin(theta1)) - (cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2))*(IL4yy*cos(theta1) + IL4xy*sin(theta1)) - (IL4yz*cos(theta1) - IL4xz*sin(theta1))*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) + (Im4xy*cos(theta1) + Im4xx*sin(theta1))*(Kr4*cos(theta1)*cos(theta2)*cos(theta3) + Kr4*cos(theta1)*cos(theta3)*sin(theta2)) - (Kr4*cos(theta2)*sin(theta1)*sin(theta3) + Kr4*cos(theta3)*sin(theta1)*sin(theta2))*(Im4yy*cos(theta1) + Im4xy*sin(theta1)) + mL4*(L4*sin(theta2 + theta3)*sin(theta1) + d4*sin(theta2 + theta3)*sin(theta1))*(L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) + 2*L4*sin(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) - L4*cos(theta2 + theta3)*cos(theta1)*sin(theta2)*sin(theta3)) - mL4*(L4*sin(theta2 + theta3)*cos(theta1) + d4*sin(theta2 + theta3)*cos(theta1))*(L4*sin(theta2 + theta3)*cos(theta2)*sin(theta1)*sin(theta3) - L4*cos(theta2 + theta3)*sin(theta1)*sin(theta2)*sin(theta3) + L4*sin(theta2 + theta3)*cos(theta3)*sin(theta1)*sin(theta2) + L4*cos(theta2 + theta3)*cos(theta2)*cos(theta3)*sin(theta1)) + mL4*(L4*cos(theta2 + theta3) + d4*cos(theta2 + theta3))*(L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) - L4*cos(theta2 + theta3)*cos(theta1)*sin(theta2)*sin(theta3) + L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) + L4*cos(theta2 + theta3)*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2))

$b_{41} =$

- IL4yz*(cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2)) - IL4zz*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) - Im4xz*(Kr4*cos(theta1)*cos(theta2)*cos(theta3) + Kr4*cos(theta1)*cos(theta3)*sin(theta2)) - Im4yz*(Kr4*cos(theta2)*sin(theta1)*sin(theta3) + Kr4*cos(theta3)*sin(theta1)*sin(theta2)) - Im4zz*(Kr4*cos(theta2)*cos(theta3) - Kr4*sin(theta2)*sin(theta3)) - IL4xz*(cos(theta1)*cos(theta2)*cos(theta3) + cos(theta1)*cos(theta3)*sin(theta2)) - mL4*(L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) + 2*L4*sin(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) - L4*cos(theta2 + theta3)*cos(theta1)*sin(theta2)*sin(theta3))*(L4*cos(theta2 + theta3)*cos(theta1) + d4*cos(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*cos(theta2)) - mL4*(L4*cos(theta2 + theta3)*sin(theta1) + d4*cos(theta2 + theta3)*sin(theta1) + a2*cos(theta2)*sin(theta1))*(L4*sin(theta2 + theta3)*cos(theta2)*sin(theta1)*sin(theta3) - L4*cos(theta2 + theta3)*sin(theta1)*sin(theta2)*sin(theta3) + L4*sin(theta2 + theta3)*cos(theta3)*sin(theta1)*sin(theta2) + L4*cos(theta2 + theta3)*cos(theta2)*cos(theta3)*sin(theta1))

$b_{42} =$

sin(theta1)*(Im4xx*(Kr4*cos(theta1)*cos(theta2)*cos(theta3) + Kr4*cos(theta1)*cos(theta3)*sin(theta2)) - Im4xy*(Kr4*cos(theta2)*sin(theta1)*sin(theta3) + Kr4*cos(theta3)*sin(theta1)*sin(theta2)) + Im4xz*(Kr4*cos(theta2)*cos(theta3) - Kr4*sin(theta2)*sin(theta3))) - cos(theta1)*(Im4yy*(Kr4*cos(theta2)*sin(theta1)*sin(theta3) + Kr4*cos(theta3)*sin(theta1)*sin(theta2)) - Im4xy*(Kr4*cos(theta1)*cos(theta2)*cos(theta3) + Kr4*cos(theta1)*cos(theta3)*sin(theta2)) + Im4yz*(Kr4*cos(theta2)*cos(theta3) - Kr4*sin(theta2)*sin(theta3))) - cos(theta1)*(IL4yy*(cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2)) + IL4yz*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) - IL4xy*(cos(theta1)*cos(theta2)*cos(theta3) + cos(theta1)*cos(theta3)*sin(theta2))) + sin(theta1)*(IL4xz*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) - IL4xy*(cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2)) + IL4xx*(cos(theta1)*cos(theta2)*cos(theta3) + cos(theta1)*cos(theta3)*sin(theta2))) + mL4*(L4*cos(theta2 + theta3) + d4*cos(theta2 + theta3) + a2*cos(theta2))*(L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) - L4*cos(theta2 + theta3)*cos(theta1)*sin(theta2)*sin(theta3) + L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) + L4*cos(theta2 + theta3)*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2)) + mL4*(L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) + 2*L4*sin(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) - L4*cos(theta2 + theta3)*cos(theta1)*sin(theta2)*sin(theta3))*(L4*sin(theta2 + theta3)*sin(theta1) + d4*sin(theta2 + theta3)*sin(theta1) + a2*sin(theta1)*sin(theta2)) - mL4*(L4*sin(theta2 + theta3)*cos(theta1) + d4*sin(theta2 + theta3)*cos(theta1) + a2*cos(theta1)*sin(theta2))*(L4*sin(theta2 + theta3)*cos(theta2)*sin(theta1)*sin(theta3) - L4*cos(theta2 + theta3)*sin(theta1)*sin(theta2)*sin(theta3) + L4*sin(theta2 + theta3)*cos(theta3)*sin(theta1)*sin(theta2) + L4*cos(theta2 + theta3)*cos(theta2)*cos(theta3)*sin(theta1))

$b_{43} =$

sin(theta1)*(Im4xx*(Kr4*cos(theta1)*cos(theta2)*cos(theta3) + Kr4*cos(theta1)*cos(theta3)*sin(theta2)) - Im4xy*(Kr4*cos(theta2)*sin(theta1)*sin(theta3) + Kr4*cos(theta3)*sin(theta1)*sin(theta2)) + Im4xz*(Kr4*cos(theta2)*cos(theta3) - Kr4*sin(theta2)*sin(theta3))) - cos(theta1)*(Im4yy*(Kr4*cos(theta2)*sin(theta1)*sin(theta3) + Kr4*cos(theta3)*sin(theta1)*sin(theta2)) - Im4xy*(Kr4*cos(theta1)*cos(theta2)*cos(theta3) + Kr4*cos(theta1)*cos(theta3)*sin(theta2)) + Im4yz*(Kr4*cos(theta2)*cos(theta3) - Kr4*sin(theta2)*sin(theta3))) - cos(theta1)*(IL4yy*(cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2)) + IL4yz*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) - IL4xy*(cos(theta1)*cos(theta2)*cos(theta3) + cos(theta1)*cos(theta3)*sin(theta2))) + sin(theta1)*(IL4xz*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) - IL4xy*(cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2)) +

IL4xx*(cos(theta1)*cos(theta2)*cos(theta3) + cos(theta1)*cos(theta3)*sin(theta2))) + mL4*(L4*sin(theta2 + theta3)*sin(theta1) + d4*sin(theta2 + theta3)*sin(theta1))*(L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) + 2*L4*sin(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) - L4*cos(theta2 + theta3)*cos(theta1)*sin(theta2)*sin(theta3)) - mL4*(L4*sin(theta2 + theta3)*cos(theta1) + d4*sin(theta2 + theta3)*cos(theta1))*(L4*sin(theta2 + theta3)*cos(theta2)*sin(theta1)*sin(theta3) - L4*cos(theta2 + theta3)*sin(theta1)*sin(theta2)*sin(theta3) + L4*sin(theta2 + theta3)*cos(theta3)*sin(theta1)*sin(theta2) + L4*cos(theta2 + theta3)*cos(theta2)*cos(theta3)*sin(theta1)) + mL4*(L4*cos(theta2 + theta3) + d4*cos(theta2 + theta3))*(L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) - L4*cos(theta2 + theta3)*cos(theta1)*sin(theta2)*sin(theta3) + L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) + L4*cos(theta2 + theta3)*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2))

$b_{44} =$

mL4*(L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) - L4*cos(theta2 + theta3)*cos(theta1)*sin(theta2)*sin(theta3) + L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) + L4*cos(theta2 + theta3)*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2))^2 + (cos(theta1)*cos(theta2)*cos(theta3) + cos(theta1)*cos(theta3)*sin(theta2))*(IL4xz*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) - IL4xy*(cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2)) + IL4xx*(cos(theta1)*cos(theta2)*cos(theta3) + cos(theta1)*cos(theta3)*sin(theta2))) + mL4*(L4*cos(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) + 2*L4*sin(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3) - L4*cos(theta2 + theta3)*cos(theta1)*sin(theta2)*sin(theta3))^2 + (cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2))*(IL4yy*(cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2)) + IL4yz*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) - IL4xy*(cos(theta1)*cos(theta2)*cos(theta3) + cos(theta1)*cos(theta3)*sin(theta2))) + (Kr4*cos(theta1)*cos(theta2)*cos(theta3) + Kr4*cos(theta1)*cos(theta3)*sin(theta2))*(Im4xx*(Kr4*cos(theta1)*cos(theta2)*cos(theta3) + Kr4*cos(theta1)*cos(theta3)*sin(theta2)) - Im4xy*(Kr4*cos(theta2)*sin(theta1)*sin(theta3) + Kr4*cos(theta3)*sin(theta1)*sin(theta2)) + Im4xz*(Kr4*cos(theta2)*cos(theta3) - Kr4*sin(theta2)*sin(theta3))) + (cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3))*(IL4yz*(cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2)) + IL4zz*(cos(theta2)*cos(theta3) - sin(theta2)*sin(theta3)) + IL4xz*(cos(theta1)*cos(theta2)*cos(theta3) + cos(theta1)*cos(theta3)*sin(theta2))) + mL4*(L4*sin(theta2 + theta3)*cos(theta2)*sin(theta1)*sin(theta3) - L4*cos(theta2 + theta3)*sin(theta1)*sin(theta2)*sin(theta3) + L4*sin(theta2 + theta3)*cos(theta3)*sin(theta1)*sin(theta2) + L4*cos(theta2 + theta3)*cos(theta2)*cos(theta3)*sin(theta1))^2 + (Kr4*cos(theta2)*sin(theta1)*sin(theta3) + Kr4*cos(theta3)*sin(theta1)*sin(theta2))*(Im4yy*(Kr4*cos(theta2)*sin(theta1)*sin(theta3) + Kr4*cos(theta3)*sin(theta1)*sin(theta2)) - Im4xy*(Kr4*cos(theta1)*cos(theta2)*cos(theta3) + Kr4*cos(theta1)*cos(theta3)*sin(theta2)) + Im4yz*(Kr4*cos(theta2)*cos(theta3) - Kr4*sin(theta2)*sin(theta3))) + (Kr4*cos(theta2)*cos(theta3) - Kr4*sin(theta2)*sin(theta3))*(Im4xz*(Kr4*cos(theta1)*cos(theta2)*cos(theta3) + Kr4*cos(theta1)*cos(theta3)*sin(theta2)) + Im4yz*(Kr4*cos(theta2)*sin(theta1)*sin(theta3) + Kr4*cos(theta3)*sin(theta1)*sin(theta2)) + Im4zz*(Kr4*cos(theta2)*cos(theta3) - Kr4*sin(theta2)*sin(theta3)))

APPEMDICES 2

Torques Results on each joint

T_1=

IL1zz*theta1dd + IL2zz*theta1dd + IL3zz*theta1dd + IL4zz*theta1dd + Im2zz*theta1dd + Im3zz*theta1dd + Im4zz*theta1dd - IL2xz*theta2dd*sin(theta1) - IL3xz*theta2dd*sin(theta1) - IL3xz*theta3dd*sin(theta1) - IL4xz*theta2dd*sin(theta1) - IL4xz*theta3dd*sin(theta1) - Im3xz*theta2dd*sin(theta1) - Im4xz*theta2dd*sin(theta1) - Im4xz*theta3dd*sin(theta1) + Im1zz*Kr1^2*theta1dd + IL2yz*theta2dd*cos(theta1) + IL3yz*theta2dd*cos(theta1) + IL3yz*theta3dd*cos(theta1) + IL4yz*theta2dd*cos(theta1) + IL4yz*theta3dd*cos(theta1) + Im3yz*theta2dd*cos(theta1) + Im4yz*theta2dd*cos(theta1) + Im4yz*theta3dd*cos(theta1) + Im2yz*Kr2*theta2dd*cos(theta1) + Im3yz*Kr3*theta3dd*cos(theta1) - Im2xz*Kr2*theta2dd*sin(theta1) - Im3xz*Kr3*theta3dd*sin(theta1) - IL4zz*theta4dd*cos(theta2)*cos(theta3) + IL4zz*theta4dd*sin(theta2)*sin(theta3) - Im4zz*Kr4*theta4dd*cos(theta2)*cos(theta3) + L3^2*mL3*theta1dd*cos(theta2 + theta3)^2*cos(theta1)^2 + L4^2*mL4*theta1dd*cos(theta2 + theta3)^2*cos(theta1)^2 + IL4xy*theta4d^2*cos(theta1)^2*cos(theta3)^2*sin(theta2)^2 + Im4zz*Kr4*theta4dd*sin(theta2)*sin(theta3) + L3^2*mL3*theta1dd*cos(theta2 + theta3)^2*sin(theta1)^2 + L4^2*mL4*theta1dd*cos(theta2 + theta3)^2*sin(theta1)^2 - IL4xy*theta4d^2*cos(theta3)^2*sin(theta1)^2*sin(theta2)^2 + d4^2*mL4*theta1dd*cos(theta2 + theta3)^2*cos(theta1)^2 + d4^2*mm4*theta1dd*cos(theta2 + theta3)^2*cos(theta1)^2 + d4^2*mL4*theta1dd*cos(theta2 + theta3)^2*sin(theta1)^2 + d4^2*mm4*theta1dd*cos(theta2 + theta3)^2*sin(theta1)^2 + L2^2*mL2*theta1dd*cos(theta1)^2*cos(theta2)^2 + L2^2*mL2*theta1dd*cos(theta2)^2*sin(theta1)^2 + a2^2*mL3*theta1dd*cos(theta1)^2*cos(theta2)^2 + a2^2*mL4*theta1dd*cos(theta1)^2*cos(theta2)^2 + a2^2*mm3*theta1dd*cos(theta1)^2*cos(theta2)^2 + a2^2*mm4*theta1dd*cos(theta1)^2*cos(theta2)^2 + a2^2*mL3*theta1dd*cos(theta2)^2*sin(theta1)^2 + a2^2*mL4*theta1dd*cos(theta2)^2*sin(theta1)^2 + a2^2*mm3*theta1dd*cos(theta2)^2*sin(theta1)^2 + a2^2*mm4*theta1dd*cos(theta2)^2*sin(theta1)^2 - IL4xz*theta4dd*cos(theta1)*cos(theta2)*cos(theta3) - IL4xz*theta4dd*cos(theta1)*cos(theta3)*sin(theta2) - IL4yz*theta4dd*cos(theta2)*sin(theta1)*sin(theta3) - IL4yz*theta4dd*cos(theta3)*sin(theta1)*sin(theta2) + IL4xz*theta4d^2*cos(theta2)^2*cos(theta3)^2*sin(theta1) - Im4xz*Kr4*theta4dd*cos(theta1)*cos(theta3)*sin(theta2) - Im4yz*Kr4*theta4dd*cos(theta2)*sin(theta1)*sin(theta3) - Im4yz*Kr4*theta4dd*cos(theta3)*sin(theta1)*sin(theta2) - 1/2*IL4xz*theta2d*theta4d*cos(theta1)*cos(theta2)*cos(theta3) - 1/2*IL4xz*theta3d*theta4d*cos(theta1)*cos(theta2)*cos(theta3) - 1/2*IL4xz*theta1d*theta4d*cos(theta2)*cos(theta3)*sin(theta1) + 1/2*IL4yz*theta1d*theta4d*cos(theta1)*cos(theta2)*sin(theta3) + 1/2*IL4yz*theta1d*theta4d*cos(theta1)*cos(theta3)*sin(theta2) - 1/2*IL4yz*theta2d*theta4d*cos(theta2)*cos(theta3)*sin(theta1) - 1/2*IL4yz*theta3d*theta4d*cos(theta2)*cos(theta3)*sin(theta1) + 2*L4*d4*mL4*theta1dd*cos(theta2 + theta3)^2*cos(theta1)^2 - 1/2*IL4xz*theta1d*theta4d*cos(theta3)*sin(theta1)*sin(theta2) + 1/2*IL4xz*theta2d*theta4d*cos(theta1)*sin(theta2)*sin(theta3) + 1/2*IL4xz*theta3d*theta4d*cos(theta1)*sin(theta2)*sin(theta3) + 2*L4*d4*mL4*theta1dd*cos(theta2 + theta3)^2*sin(theta1)^2 + 1/2*IL4yz*theta2d*theta4d*sin(theta1)*sin(theta2)*sin(theta3) + 1/2*IL4yz*theta3d*theta4d*sin(theta1)*sin(theta2)*sin(theta3) - IL4yz*theta4d^2*cos(theta1)*cos(theta2)*cos(theta3)^2*sin(theta2) - IL4yz*theta4d^2*cos(theta1)*cos(theta2)^2*cos(theta3)*sin(theta3) + IL4xz*theta4d^2*cos(theta2)*cos(theta3)^2*sin(theta1)*sin(theta2) + IL4yz*theta4d^2*cos(theta1)*cos(theta2)*sin(theta2)*sin(theta3)^2 + IL4yz*theta4d^2*cos(theta1)*cos(theta3)*sin(theta2)^2*sin(theta3) - IL4xz*theta4d^2*cos(theta3)*sin(theta1)*sin(theta2)^2*sin(theta3) - 1/2*IL4xx*theta2d*theta4d*cos(theta1)^2*cos(theta2)*cos(theta3) - 1/2*IL4xx*theta3d*theta4d*cos(theta1)^2*cos(theta2)*cos(theta3) + Im4xz*Kr4^2*theta4d^2*cos(theta2)^2*cos(theta3)^2*sin(theta1) + 1/2*IL4xx*theta2d*theta4d*cos(theta2)*cos(theta3)*sin(theta1)^2 - 1/2*IL4xx*theta2d*theta4d*cos(theta1)^2*cos(theta3)*sin(theta2) + 1/2*IL4xx*theta3d*theta4d*cos(theta2)*cos(theta3)*sin(theta1)^2 - 1/2*IL4xx*theta3d*theta4d*cos(theta1)^2*cos(theta3)*sin(theta2) + 1/2*IL4yy*theta2d*theta4d*cos(theta1)^2*cos(theta2)*sin(theta3) + 1/2*IL4yy*theta2d*theta4d*cos(theta1)^2*cos(theta3)*sin(theta2) + 1/2*IL4yy*theta3d*theta4d*cos(theta1)^2*cos(theta2)*sin(theta3) + 1/2*IL4yy*theta3d*theta4d*cos(theta1)^2*cos(theta3)*sin(theta2) - L3^2*mL3*theta2dd*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta1)^2 + 1/2*IL4xx*theta2d*theta4d*cos(theta3)*sin(theta1)^2*sin(theta2) + 1/2*IL4xx*theta3d*theta4d*cos(theta3)*sin(theta1)^2*sin(theta2) - 1/2*IL4yy*theta2d*theta4d*cos(theta2)*sin(theta1)^2*sin(theta3) - 1/2*IL4yy*theta2d*theta4d*cos(theta3)*sin(theta1)^2*sin(theta2) - 1/2*IL4yy*theta3d*theta4d*cos(theta2)*sin(theta1)^2*sin(theta3) - 1/2*IL4yy*theta3d*theta4d*cos(theta3)*sin(theta1)^2*sin(theta2) + IL4xx*theta4d^2*cos(theta1)*cos(theta2)^2*cos(theta3)^2*sin(theta1) + IL4xy*theta4d^2*cos(theta1)^2*cos(theta2)*cos(theta3)^2*sin(theta2) + IL4xy*theta4d^2*cos(theta1)^2*cos(theta2)^2*cos(theta3)*sin(theta3) + IL4xx*theta4d^2*cos(theta1)*cos(theta3)^2*sin(theta1)*sin(theta2)^2 - IL4xy*theta4d^2*cos(theta2)*cos(theta3)^2*sin(theta1)^2*sin(theta2) - IL4yy*theta4d^2*cos(theta1)*cos(theta2)^2*sin(theta1)*sin(theta3)^2 - IL4yy*theta4d^2*cos(theta1)*cos(theta3)^2*sin(theta1)*sin(theta2)^2 - IL4xy*theta4d^2*cos(theta2)^2*cos(theta3)*sin(theta1)^2*sin(theta3) + Im4xy*Kr4^2*theta4d^2*cos(theta1)^2*cos(theta3)^2*sin(theta2)^2 - L2^2*mL2*theta2dd*cos(theta1)^2*cos(theta2)*sin(theta2) - Im4xy*Kr4^2*theta4d^2*cos(theta3)^2*sin(theta1)^2*sin(theta2)^2 - Im4xz*Kr4*theta4dd*cos(theta1)*cos(theta2)*cos(theta3) + L2^2*mL2*theta2dd*cos(theta2)*sin(theta1)^2*sin(theta2) + IL4xy*theta2d*theta4d*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) + IL4xy*theta3d*theta4d*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) + Im4xz*Kr4^2*theta4d^2*cos(theta2)*cos(theta3)^2*sin(theta1)*sin(theta2) + Im4yz*Kr4^2*theta4d^2*cos(theta1)*cos(theta2)*sin(theta2)*sin(theta3)^2 + Im4yz*Kr4^2*theta4d^2*cos(theta1)*cos(theta3)*sin(theta2)^2*sin(theta3) + L4^2*mL4*theta4dd*cos(theta2 + theta3)^2*sin(theta1)^2*sin(theta2)*sin(theta3) + IL4xy*theta2d*theta4d*cos(theta1)*cos(theta2)*sin(theta1)*sin(theta3) + 2*IL4xy*theta2d*theta4d*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2) + IL4xy*theta3d*theta4d*cos(theta1)*cos(theta2)*sin(theta1)*sin(theta3) + 2*IL4xy*theta3d*theta4d*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2) + L3^2*mL3*theta2dd*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta1)*sin(theta1) - Im4xz*Kr4^2*theta4d^2*cos(theta3)*sin(theta1)*sin(theta2)^2*sin(theta3) + 2*IL4xx*theta4d^2*cos(theta1)*cos(theta2)*cos(theta3)^2*sin(theta1)*sin(theta2) + IL4xy*theta4d^2*cos(theta1)^2*cos(theta2)*cos(theta3)*sin(theta2)*sin(theta3) - IL4xy*theta4d^2*cos(theta2)*cos(theta3)*sin(theta1)^2*sin(theta2)*sin(theta3) + Im4xx*Kr4^2*theta4d^2*cos(theta1)*cos(theta2)^2*cos(theta3)^2*sin(theta1) + Im4xy*Kr4^2*theta4d^2*cos(theta1)^2*cos(theta2)*cos(theta3)^2*sin(theta2) + Im4xy*Kr4^2*theta4d^2*cos(theta1)^2*cos(theta2)^2*cos(theta3)*sin(theta3) -

1/2*Im4xz*Kr4*theta2d*theta4d*cos(theta1)*cos(theta2)*cos(theta3) - 1/2*Im4xz*Kr4*theta3d*theta4d*cos(theta1)*cos(theta2)*cos(theta3) +
Im4xx*Kr4^2*theta4d^2*cos(theta1)*cos(theta3)^2*sin(theta1)*sin(theta2)^2 -
Im4xy*Kr4^2*theta4d^2*cos(theta2)*cos(theta3)^2*sin(theta1)^2*sin(theta2) -
Im4yy*Kr4^2*theta4d^2*cos(theta1)*cos(theta2)^2*sin(theta1)*sin(theta3)^2 -
Im4yy*Kr4^2*theta4d^2*cos(theta1)*cos(theta3)^2*sin(theta1)*sin(theta2)^2 -
Im4xy*Kr4^2*theta4d^2*cos(theta2)^2*cos(theta3)*sin(theta1)^2*sin(theta3) -
1/2*Im4xz*Kr4*theta1d*theta4d*cos(theta2)*cos(theta3)*sin(theta1) + 1/2*Im4yz*Kr4*theta1d*theta4d*cos(theta1)*cos(theta2)*sin(theta3) +
1/2*Im4yz*Kr4*theta1d*theta4d*cos(theta1)*cos(theta3)*sin(theta2) - 1/2*Im4yz*Kr4*theta2d*theta4d*cos(theta2)*cos(theta3)*sin(theta1) -
1/2*Im4yz*Kr4*theta3d*theta4d*cos(theta2)*cos(theta3)*sin(theta1) - 1/2*Im4xz*Kr4*theta1d*theta4d*cos(theta3)*sin(theta1)*sin(theta2) +
1/2*Im4xz*Kr4*theta2d*theta4d*cos(theta1)*sin(theta2)*sin(theta3) + 1/2*Im4xz*Kr4*theta3d*theta4d*cos(theta1)*sin(theta2)*sin(theta3) +
1/2*Im4yz*Kr4*theta2d*theta4d*sin(theta1)*sin(theta2)*sin(theta3) + 1/2*Im4yz*Kr4*theta3d*theta4d*sin(theta1)*sin(theta2)*sin(theta3) +
2*L3*a2*mL3*theta1dd*cos(theta2 + theta3)*cos(theta1)^2*cos(theta2) + 2*L4*a2*mL4*theta1dd*cos(theta2 +
theta3)*cos(theta1)^2*cos(theta2) + 2*L3*a2*mL3*theta1dd*cos(theta2 + theta3)*cos(theta2)*sin(theta1)^2 - L3*a2*mL3*theta2dd*sin(theta2 +
theta3)*cos(theta1)^2*cos(theta2) + 2*L4*a2*mL4*theta1dd*cos(theta2 + theta3)*cos(theta2)*sin(theta1)^2 +
2*a2*d4*mL4*theta1dd*cos(theta2 + theta3)*cos(theta1)^2*cos(theta2) + 2*a2*d4*mm4*theta1dd*cos(theta2 +
theta3)*cos(theta1)^2*cos(theta2) - 1/2*Im4xx*Kr4*theta2d*theta4d*cos(theta1)^2*cos(theta2)*cos(theta3) -
1/2*Im4xx*Kr4*theta3d*theta4d*cos(theta1)^2*cos(theta2)*cos(theta3) + 2*a2*d4*mL4*theta1dd*cos(theta2 + theta3)*cos(theta2)*sin(theta1)^2
+ 2*a2*d4*mm4*theta1dd*cos(theta2 + theta3)*cos(theta2)*sin(theta1)^2 +
1/2*Im4xx*Kr4*theta2d*theta4d*cos(theta2)*cos(theta3)*sin(theta1)^2 - 1/2*Im4xx*Kr4*theta2d*theta4d*cos(theta1)^2*cos(theta3)*sin(theta2)
+ 1/2*Im4xx*Kr4*theta3d*theta4d*cos(theta2)*cos(theta3)*sin(theta1)^2 -
1/2*Im4xx*Kr4*theta3d*theta4d*cos(theta1)^2*cos(theta3)*sin(theta2) + 1/2*Im4yy*Kr4*theta2d*theta4d*cos(theta1)^2*cos(theta2)*sin(theta3)
+ 1/2*Im4yy*Kr4*theta2d*theta4d*cos(theta1)^2*cos(theta3)*sin(theta2) +
1/2*Im4yy*Kr4*theta3d*theta4d*cos(theta1)^2*cos(theta2)*sin(theta3) + 1/2*Im4yy*Kr4*theta3d*theta4d*cos(theta1)^2*cos(theta3)*sin(theta2)
- L4^2*mL4*theta4dd*cos(theta2 + theta3)^2*cos(theta1)^2*cos(theta2)*cos(theta3) -
IL4xz*theta4d^2*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2)*sin(theta3) +
1/2*Im4xx*Kr4*theta2d*theta4d*cos(theta3)*sin(theta1)^2*sin(theta2) + 1/2*Im4xx*Kr4*theta3d*theta4d*cos(theta3)*sin(theta1)^2*sin(theta2)
- 1/2*Im4yy*Kr4*theta2d*theta4d*cos(theta2)*sin(theta1)^2*sin(theta3) - 1/2*Im4yy*Kr4*theta2d*theta4d*cos(theta3)*sin(theta1)^2*sin(theta2)
- 1/2*Im4yy*Kr4*theta3d*theta4d*cos(theta2)*sin(theta1)^2*sin(theta3) - 1/2*Im4yy*Kr4*theta3d*theta4d*cos(theta3)*sin(theta1)^2*sin(theta2)
- L4^2*mL4*theta4dd*cos(theta2 + theta3)^2*cos(theta2)*cos(theta3)*sin(theta1)^2 -
Im4yz*Kr4^2*theta4d^2*cos(theta1)*cos(theta2)*cos(theta3)^2*sin(theta2) -
Im4yz*Kr4^2*theta4d^2*cos(theta1)*cos(theta2)^2*cos(theta3)*sin(theta3) - L4^2*mL4*theta4d^2*cos(theta2 +
theta3)^2*cos(theta1)^3*cos(theta2)^2*cos(theta3)^2 + L4^2*mL4*theta4dd*cos(theta2 + theta3)^2*cos(theta1)^2*sin(theta2)*sin(theta3) +
L4*d4*mL4*theta4dd*cos(theta2 + theta3)^2*cos(theta1)^2*sin(theta2)*sin(theta3) + 1/2*L4^2*mL4*theta2d*theta4d*cos(theta2 +
theta3)^2*cos(theta2)*cos(theta3)*sin(theta1) + 1/2*L4^2*mL4*theta3d*theta4d*cos(theta2 + theta3)^2*cos(theta2)*cos(theta3)*sin(theta1) -
L4*a2*mL4*theta4dd*sin(theta2 + theta3)*cos(theta2)^2*sin(theta1)^2*sin(theta3) + L4*d4*mL4*theta4dd*cos(theta2 +
theta3)^2*sin(theta1)^2*sin(theta2)*sin(theta3) - 1/2*L4^2*mL4*theta2d*theta4d*cos(theta2 + theta3)^2*sin(theta1)*sin(theta2)*sin(theta3) -
1/2*L4^2*mL4*theta3d*theta4d*cos(theta2 + theta3)^2*sin(theta1)*sin(theta2)*sin(theta3) - 2*L4^2*mL4*theta4dd*cos(theta2 +
theta3)*sin(theta2 + theta3)*cos(theta1)^2*cos(theta2)*cos(theta3) + L3*a2*mL3*theta2dd*sin(theta2 +
theta3)*cos(theta1)*cos(theta2)*sin(theta1) + Im4xy*Kr4*theta2d*theta4d*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) +
Im4xy*Kr4*theta3d*theta4d*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) - 1/2*L4^2*mL4*theta2d*theta4d*cos(theta2 +
theta3)^2*cos(theta1)^2*cos(theta2)*cos(theta3) - 1/2*L4^2*mL4*theta3d*theta4d*cos(theta2 + theta3)^2*cos(theta1)^2*cos(theta2)*cos(theta3)
- L4^2*mL4*theta4dd*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta2)*sin(theta1)^2*sin(theta3) - L4^2*mL4*theta4dd*cos(theta2 +
theta3)*sin(theta2 + theta3)*cos(theta3)*sin(theta1)^2*sin(theta2) + Im4xy*Kr4*theta2d*theta4d*cos(theta1)*cos(theta2)*sin(theta1)*sin(theta3)
+ 2*Im4xy*Kr4*theta2d*theta4d*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2) +
Im4xy*Kr4*theta3d*theta4d*cos(theta1)*cos(theta2)*sin(theta1)*sin(theta3) +
2*Im4xy*Kr4*theta3d*theta4d*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2) + 1/2*L4^2*mL4*theta2d*theta4d*cos(theta2 +
theta3)^2*cos(theta2)*cos(theta3)*sin(theta1)^2 - 1/2*L4^2*mL4*theta2d*theta4d*cos(theta2 + theta3)^2*cos(theta1)^2*cos(theta3)*sin(theta2) -
L4^2*mL4*theta2d*theta4d*sin(theta2 + theta3)^2*cos(theta1)^2*cos(theta2)*cos(theta3) + 1/2*L4^2*mL4*theta3d*theta4d*cos(theta2 +
theta3)^2*cos(theta2)*cos(theta3)*sin(theta1)^2 - 1/2*L4^2*mL4*theta3d*theta4d*cos(theta2 + theta3)^2*cos(theta1)^2*cos(theta3)*sin(theta2) -
L4^2*mL4*theta3d*theta4d*sin(theta2 + theta3)^2*cos(theta1)^2*cos(theta2)*cos(theta3) + 1/2*L4^2*mL4*theta2d*theta4d*cos(theta2 +
theta3)^2*cos(theta3)*sin(theta1)^2*sin(theta2) + L4^2*mL4*theta2d*theta4d*sin(theta2 + theta3)^2*cos(theta2)*cos(theta3)*sin(theta1)^2 +
1/2*L4^2*mL4*theta2d*theta4d*sin(theta2 + theta3)^2*cos(theta1)^2*cos(theta2)*sin(theta3) + 1/2*L4^2*mL4*theta2d*theta4d*sin(theta2 +
theta3)^2*cos(theta1)^2*cos(theta3)*sin(theta2) + 1/2*L4^2*mL4*theta3d*theta4d*cos(theta2 + theta3)^2*cos(theta3)*sin(theta1)^2*sin(theta2)
+ L4^2*mL4*theta3d*theta4d*sin(theta2 + theta3)^2*cos(theta2)*cos(theta3)*sin(theta1)^2 + 1/2*L4^2*mL4*theta3d*theta4d*sin(theta2 +
theta3)^2*cos(theta1)^2*cos(theta2)*sin(theta3) + 1/2*L4^2*mL4*theta3d*theta4d*sin(theta2 + theta3)^2*cos(theta1)^2*cos(theta3)*sin(theta2) -
1/2*L4^2*mL4*theta2d*theta4d*sin(theta2 + theta3)^2*cos(theta2)*sin(theta1)^2*sin(theta3) - 1/2*L4^2*mL4*theta2d*theta4d*sin(theta2 +
theta3)^2*cos(theta3)*sin(theta1)^2*sin(theta2) - 1/2*L4^2*mL4*theta3d*theta4d*sin(theta2 + theta3)^2*cos(theta2)*sin(theta1)^2*sin(theta3) -
1/2*L4^2*mL4*theta3d*theta4d*sin(theta2 + theta3)^2*cos(theta3)*sin(theta1)^2*sin(theta2) + L4^2*mL4*theta4d^2*cos(theta2 +
theta3)^2*cos(theta1)*cos(theta2)^2*cos(theta3)^2*sin(theta1) - L4^2*mL4*theta4d^2*cos(theta2 +
theta3)^2*cos(theta1)^3*cos(theta2)*cos(theta3)^2*sin(theta2) -
Im4xz*Kr4^2*theta4d^2*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2)*sin(theta3) + 4*L4^2*mL4*theta4d^2*sin(theta2 +
theta3)^2*cos(theta1)*cos(theta2)^2*cos(theta3)^2*sin(theta1) + L4^2*mL4*theta4d^2*cos(theta2 +
theta3)^2*cos(theta1)^3*cos(theta3)*sin(theta2)^2*sin(theta3) + L4^2*mL4*theta4d^2*cos(theta2 +
theta3)^2*cos(theta1)*sin(theta1)*sin(theta2)^2*sin(theta3)^2 - L4^2*mL4*theta4d^2*sin(theta2 +
theta3)^2*cos(theta1)*cos(theta2)^2*sin(theta1)*sin(theta3)^2 - L4^2*mL4*theta4d^2*sin(theta2 +
theta3)^2*cos(theta1)*cos(theta3)^2*sin(theta1)*sin(theta2)^2 -
2*IL4yy*theta4d^2*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2)*sin(theta3) +
2*Im4xx*Kr4^2*theta4d^2*cos(theta1)*cos(theta2)*cos(theta3)^2*sin(theta1)*sin(theta2) +
Im4xy*Kr4^2*theta4d^2*cos(theta1)^2*cos(theta2)*cos(theta3)*sin(theta2)*sin(theta3) - L4*a2*mL4*theta4dd*cos(theta2 +
theta3)*cos(theta1)^2*cos(theta2)^2*cos(theta3) - L4*d4*mL4*theta4dd*cos(theta2 + theta3)^2*cos(theta1)^2*cos(theta2)*cos(theta3) +

2*L4^2*mL4*theta4d^2*cos(theta2 + theta3)^2*cos(theta1)*cos(theta2)^2*cos(theta3)^2*sin(theta1)^2 + L4^2*mL4*theta4d^2*cos(theta2 + theta3)^2*cos(theta1)*cos(theta2)^2*cos(theta3)^2*sin(theta1)^3 - L4^2*mL4*theta4d^2*cos(theta2 + theta3)^2*cos(theta1)^3*cos(theta2)^2*cos(theta3)^2*sin(theta1) - Im4xy*Kr4^2*theta4d^2*cos(theta2)*cos(theta3)*sin(theta1)^2*sin(theta2)*sin(theta3) - L4*a2*mL4*theta4dd*cos(theta2 + theta3)*cos(theta2)^2*cos(theta3)*sin(theta1)^2 - 2*L4*a2*mL4*theta4dd*sin(theta2 + theta3)*cos(theta1)^2*cos(theta2)^2*cos(theta3) - L4*d4*mL4*theta4dd*cos(theta2 + theta3)^2*cos(theta2)*cos(theta3)*sin(theta1)^2 + L4^2*mL4*theta4d^2*cos(theta2 + theta3)^2*cos(theta1)*cos(theta3)^2*sin(theta1)^3*sin(theta2)^2 - L4^2*mL4*theta4d^2*cos(theta2 + theta3)^2*cos(theta1)^3*cos(theta3)^2*sin(theta1)*sin(theta2)^2 + 1/2*L4*a2*mL4*theta2d*theta4d*cos(theta2 + theta3)*cos(theta2)^2*cos(theta3)*sin(theta1) + 1/2*L4*d4*mL4*theta2d*theta4d*cos(theta2 + theta3)^2*cos(theta2)*cos(theta3)*sin(theta1) + 1/2*L4*d4*mL4*theta3d*theta4d*cos(theta2 + theta3)^2*cos(theta2)*cos(theta3)*sin(theta1) + L4^2*mL4*theta4d^2*cos(theta2 + theta3)^2*cos(theta1)^3*cos(theta2)*cos(theta3)*sin(theta2)*sin(theta3) - 1/2*L4*d4*mL4*theta2d*theta4d*cos(theta2 + theta3)^2*sin(theta1)*sin(theta2)*sin(theta3) - 1/2*L4*d4*mL4*theta3d*theta4d*cos(theta2 + theta3)^2*sin(theta1)*sin(theta2)*sin(theta3) - 2*L4*d4*mL4*theta4dd*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta1)^2*cos(theta2)*cos(theta3) - 1/2*L4*a2*mL4*theta2d*theta4d*cos(theta2 + theta3)*cos(theta1)^2*cos(theta2)^2*cos(theta3) - 1/2*L4*d4*mL4*theta2d*theta4d*cos(theta2 + theta3)^2*cos(theta1)^2*cos(theta2)*cos(theta3) - 1/2*L4*d4*mL4*theta3d*theta4d*cos(theta2 + theta3)^2*cos(theta1)^2*cos(theta2)*cos(theta3) - L4*d4*mL4*theta4dd*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta2)*sin(theta1)^2*sin(theta3) - L4*d4*mL4*theta4dd*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta3)*sin(theta1)^2*sin(theta2) + 1/2*L4*a2*mL4*theta2d*theta4d*cos(theta2 + theta3)*cos(theta2)^2*cos(theta3)*sin(theta1)^2 - 2*Im4yy*Kr4^2*theta4d^2*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2)*sin(theta3) + 1/2*L4*d4*mL4*theta2d*theta4d*cos(theta2 + theta3)^2*cos(theta2)*cos(theta3)*sin(theta1)^2 - 1/2*L4*d4*mL4*theta2d*theta4d*cos(theta2 + theta3)^2*cos(theta1)^2*cos(theta3)*sin(theta2) - L4*d4*mL4*theta2d*theta4d*sin(theta2 + theta3)^2*cos(theta1)^2*cos(theta2)*cos(theta3) + 1/2*L4*d4*mL4*theta3d*theta4d*cos(theta2 + theta3)^2*cos(theta2)*cos(theta3)*sin(theta1)^2 - 1/2*L4*d4*mL4*theta3d*theta4d*cos(theta2 + theta3)^2*cos(theta1)^2*cos(theta3)*sin(theta2) - L4*d4*mL4*theta3d*theta4d*sin(theta2 + theta3)^2*cos(theta1)^2*cos(theta2)*cos(theta3) + 2*L4^2*mL4*theta4d^2*cos(theta2 + theta3)^2*cos(theta1)*cos(theta2)*cos(theta3)^2*sin(theta1)^2*sin(theta2) + 2*L4^2*mL4*theta4d^2*cos(theta2 + theta3)^2*cos(theta1)*cos(theta2)*cos(theta3)^2*sin(theta1)^3*sin(theta2) - 2*L4^2*mL4*theta4d^2*cos(theta2 + theta3)^2*cos(theta1)^3*cos(theta2)*cos(theta3)^2*sin(theta1)*sin(theta2) + 1/2*L4*a2*mL4*theta2d*theta4d*sin(theta2 + theta3)*cos(theta1)^2*cos(theta3)*sin(theta2)^2 + 1/2*L4*d4*mL4*theta2d*theta4d*cos(theta2 + theta3)^2*cos(theta3)*sin(theta1)^2*sin(theta2) + L4*d4*mL4*theta2d*theta4d*sin(theta2 + theta3)^2*cos(theta2)*cos(theta3)*sin(theta1)^2 + 1/2*L4*d4*mL4*theta2d*theta4d*sin(theta2 + theta3)^2*cos(theta1)^2*cos(theta2)*sin(theta3) + 1/2*L4*d4*mL4*theta2d*theta4d*sin(theta2 + theta3)^2*cos(theta1)^2*cos(theta3)*sin(theta2) + 1/2*L4*d4*mL4*theta3d*theta4d*cos(theta2 + theta3)^2*cos(theta3)*sin(theta1)^2*sin(theta2) + L4*d4*mL4*theta3d*theta4d*sin(theta2 + theta3)^2*cos(theta2)*cos(theta3)*sin(theta1)^2 + 1/2*L4*d4*mL4*theta3d*theta4d*sin(theta2 + theta3)^2*cos(theta1)^2*cos(theta2)*sin(theta3) + 1/2*L4*d4*mL4*theta3d*theta4d*sin(theta2 + theta3)^2*cos(theta1)^2*cos(theta3)*sin(theta2) + L4*a2*mL4*theta4dd*cos(theta2 + theta3)*cos(theta1)^2*cos(theta2)*sin(theta2)*sin(theta3) - 2*L4^2*mL4*theta4d^2*cos(theta2 + theta3)^2*cos(theta1)*cos(theta3)*sin(theta1)^2*sin(theta2)^2*sin(theta3) - 1/2*L4*a2*mL4*theta2d*theta4d*sin(theta2 + theta3)*cos(theta3)*sin(theta1)^2*sin(theta2)^2 - 1/2*L4*d4*mL4*theta2d*theta4d*sin(theta2 + theta3)^2*cos(theta2)*sin(theta1)^2*sin(theta3) - 1/2*L4*d4*mL4*theta2d*theta4d*sin(theta2 + theta3)^2*cos(theta3)*sin(theta1)^2*sin(theta2) - 1/2*L4*d4*mL4*theta3d*theta4d*sin(theta2 + theta3)^2*cos(theta2)*sin(theta1)^2*sin(theta3) - 1/2*L4*d4*mL4*theta3d*theta4d*sin(theta2 + theta3)^2*cos(theta3)*sin(theta1)^2*sin(theta2) + L4*a2*mL4*theta4dd*cos(theta2 + theta3)*cos(theta2)*sin(theta1)^2*sin(theta2)*sin(theta3) - L4*a2*mL4*theta4dd*sin(theta2 + theta3)*cos(theta2)*cos(theta3)*sin(theta1)^2*sin(theta2) + 4*L4^2*mL4*theta4d^2*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta1)*cos(theta2)^2*cos(theta3)^2*sin(theta1) - 1/2*L4*a2*mL4*theta2d*theta4d*sin(theta2 + theta3)*cos(theta2)*sin(theta1)^2*sin(theta2)*sin(theta3) - 2*L4^2*mL4*theta1d*theta4d*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) + L4^2*mL4*theta1d*theta4d*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta1)*cos(theta2)*sin(theta1)*sin(theta3) + L4^2*mL4*theta1d*theta4d*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2) - 2*L4^2*mL4*theta4d^2*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3)^2*sin(theta1)*sin(theta2) - 2*L4^2*mL4*theta4d^2*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta1)*cos(theta2)^2*cos(theta3)*sin(theta1)*sin(theta3) + 2*L4^2*mL4*theta4d^2*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta1)*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3)^2 + 2*L4^2*mL4*theta4d^2*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2)^2*sin(theta3) - 2*L4^2*mL4*theta4d^2*cos(theta2 + theta3)^2*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2)*sin(theta3) - 1/2*L4*a2*mL4*theta2d*theta4d*cos(theta2 + theta3)*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3) - 2*L4^2*mL4*theta4d^2*sin(theta2 + theta3)^2*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2)*sin(theta3) - 1/2*L4*a2*mL4*theta2d*theta4d*cos(theta2 + theta3)*cos(theta1)^2*cos(theta2)*cos(theta3)*sin(theta2) - 2*L4*a2*mL4*theta1d*theta4d*sin(theta2 + theta3)*cos(theta1)*cos(theta2)^2*cos(theta3)*sin(theta1) + 1/2*L4*a2*mL4*theta2d*theta4d*cos(theta2 + theta3)*cos(theta2)*cos(theta3)*sin(theta1)^2*sin(theta2) - L4*a2*mL4*theta2d*theta4d*sin(theta2 + theta3)*cos(theta1)^2*cos(theta2)*cos(theta3)*sin(theta2) - 2*L4^2*mL4*theta4d^2*cos(theta2 + theta3)^2*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1)^2*sin(theta2)*sin(theta3) + L4*a2*mL4*theta1d*theta4d*sin(theta2 + theta3)*cos(theta1)*cos(theta2)^2*sin(theta1)*sin(theta3) + L4*a2*mL4*theta2d*theta4d*sin(theta2 + theta3)*cos(theta2)*cos(theta3)*sin(theta1)^2*sin(theta2) + 1/2*L4*a2*mL4*theta2d*theta4d*sin(theta2 + theta3)*cos(theta1)^2*cos(theta2)*sin(theta2)*sin(theta3) - 2*L4*d4*mL4*theta1d*theta4d*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) + L4*d4*mL4*theta1d*theta4d*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta1)*cos(theta2)*sin(theta1)*sin(theta3) + L4*d4*mL4*theta1d*theta4d*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2) + L4*a2*mL4*theta1d*theta4d*sin(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2) - 4*L4^2*mL4*theta4d^2*cos(theta2 + theta3)*sin(theta2 + theta3)*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2)*sin(theta3)

T_2=

IL1zz*theta1dd + 1/2*IL2xx*theta2dd + 1/2*IL2yy*theta2dd + IL2zz*theta1dd + 1/2*IL3xx*theta2dd + 1/2*IL3xx*theta3dd + 1/2*IL3yy*theta2dd + 1/2*IL3yy*theta3dd + IL3zz*theta1dd + 1/2*IL4xx*theta2dd + 1/2*IL4xx*theta3dd + 1/2*IL4yy*theta2dd + 1/2*IL4yy*theta3dd + IL4zz*theta1dd + Im2zz*theta1dd + 1/2*Im3xx*theta2dd + 1/2*Im3yy*theta2dd + Im3zz*theta1dd + 1/2*Im4xx*theta2dd + 1/2*Im4xx*theta3dd + 1/2*Im4yy*theta2dd + 1/2*Im4yy*theta3dd + Im4zz*theta1dd + 1/2*a2^2*mL3*theta1dd + a2^2*mL3*theta2dd + 1/2*a2^2*mL4*theta1dd + a2^2*mL4*theta2dd + 1/2*a2^2*mm3*theta1dd + a2^2*mm3*theta2dd + 1/2*a2^2*mm4*theta1dd + a2^2*mm4*theta2dd + 1/2*d4^2*mL4*theta1dd + d4^2*mL4*theta2dd + d4^2*mL4*theta3dd + 1/2*d4^2*mm4*theta1dd + d4^2*mm4*theta2dd + 1/2*d4^2*mm4*theta3dd - IL2xz*theta1dd*sin(theta1) - IL2xz*theta2dd*sin(theta1) - IL3xz*theta1dd*sin(theta1) - IL3xz*theta2dd*sin(theta1) - IL3xz*theta3dd*sin(theta1) - IL4xz*theta1dd*sin(theta1) - IL4xz*theta2dd*sin(theta1) - IL4xz*theta3dd*sin(theta1) - Im3xz*theta1dd*sin(theta1) - Im3xz*theta2dd*sin(theta1) - Im4xz*theta1dd*sin(theta1) - Im4xz*theta2dd*sin(theta1) - Im4xz*theta3dd*sin(theta1) + 1/8*IL4xx*theta4dd*sin(2*theta1 + theta2 - theta3) + 1/8*IL4xx*theta4dd*sin(2*theta1 - theta2 + theta3) + 1/8*IL4xy*theta4dd*sin(2*theta1 + theta2 - theta3) - 1/8*IL4xy*theta4dd*sin(2*theta1 - theta2 + theta3) + 3/4*IL4xz*theta4dd*sin(theta1 - theta2 - theta3) + 1/4*IL4xy*theta4dd*cos(theta2 - theta3) + 1/4*IL4xy*theta4dd*sin(theta2 - theta3) - 1/4*IL4xz*theta4dd*cos(theta1 + theta2 + theta3) + 1/4*IL4xz*theta4dd*sin(theta1 + theta2 + theta3) - 1/2*IL2xx*theta2dd*cos(2*theta1) + 1/2*IL2yy*theta2dd*cos(2*theta1) - 1/2*IL3xx*theta2dd*cos(2*theta1) - 1/2*IL3xx*theta3dd*cos(2*theta1) + 1/2*IL3yy*theta2dd*cos(2*theta1) + 1/2*IL3yy*theta3dd*cos(2*theta1) - 1/2*IL4xx*theta2dd*cos(2*theta1) - 1/2*IL4xx*theta3dd*cos(2*theta1) + 1/2*IL4yy*theta2dd*cos(2*theta1) + 1/2*IL4yy*theta3dd*cos(2*theta1) - 1/2*Im3xx*theta2dd*cos(2*theta1) + 1/2*Im3yy*theta2dd*cos(2*theta1) - 1/2*Im4xx*theta2dd*cos(2*theta1) - 1/2*Im4xx*theta3dd*cos(2*theta1) + 1/2*Im4yy*theta2dd*cos(2*theta1) + 1/2*Im4yy*theta3dd*cos(2*theta1) + 1/8*IL4xx*theta4dd*cos(2*theta1 - theta2 - theta3) + 1/8*IL4xy*theta4dd*cos(2*theta1 - theta2 - theta3) - 1/4*IL4yy*theta4dd*cos(2*theta1 - theta2 - theta3) - 1/8*IL4xz*theta4d^2*cos(theta1 - 2*theta2 - 2*theta3) - 3/8*IL4xz*theta4d^2*cos(theta1 + 2*theta2 + 2*theta3) + IL2xy*theta2dd*sin(2*theta1) + IL3xy*theta2dd*sin(2*theta1) + IL3xy*theta3dd*sin(2*theta1) + IL4xy*theta2dd*sin(2*theta1) + IL4xy*theta3dd*sin(2*theta1) + 1/4*IL4xz*theta4d^2*sin(theta1) + Im3xy*theta2dd*sin(2*theta1) + Im4xy*theta2dd*sin(2*theta1) + Im4xy*theta3dd*sin(2*theta1) + 1/2*Im3xx*Kr3*theta3dd + 1/2*Im3yy*Kr3*theta3dd + 1/8*IL4xx*theta4dd*sin(2*theta1 - theta2 - theta3) - 3/8*IL4xy*theta4dd*sin(2*theta1 - theta2 - theta3) - 1/8*IL4xz*theta4d^2*sin(theta1 - 2*theta2 - 2*theta3) + 3/8*IL4xz*theta4d^2*sin(theta1 + 2*theta2 + 2*theta3) - 1/4*IL4yz*theta4d^2*sin(theta1 - 2*theta2 - 2*theta3) - 3/4*IL4yz*theta4d^2*sin(theta1 + 2*theta2 + 2*theta3) - 1/8*IL4xz*theta4d^2*cos(theta1 - 2*theta2) - 3/8*IL4xz*theta4d^2*cos(theta1 + 2*theta2) - 1/8*IL4xz*theta4d^2*cos(theta1 - 2*theta3) + 1/8*IL4xz*theta4d^2*cos(theta1 + 2*theta3) - 1/8*IL4xz*theta4d^2*sin(theta1 - 2*theta2) + 3/8*IL4xz*theta4d^2*sin(theta1 + 2*theta2) + 1/8*IL4xz*theta4d^2*sin(theta1 - 2*theta3) + 1/8*IL4xz*theta4d^2*sin(theta1 + 2*theta3) - 1/8*IL4xx*theta4dd*cos(2*theta1 + theta2 + theta3) + 1/8*IL4xy*theta4dd*cos(2*theta1 + theta2 + theta3) - 1/4*IL4xz*theta4dd*cos(theta1 + theta2 - theta3) - 1/4*IL4xz*theta4dd*cos(theta1 - theta2 + theta3) + 1/4*IL4yy*theta4dd*cos(2*theta1 + theta2 + theta3) + 1/8*IL4xx*theta4dd*sin(2*theta1 + theta2 + theta3) + 3/8*IL4xy*theta4dd*sin(2*theta1 + theta2 + theta3) - 1/4*IL4xz*theta4dd*sin(theta1 + theta2 - theta3) + 1/4*IL4xz*theta4dd*sin(theta1 - theta2 + theta3) - 1/4*IL4xx*theta4d^2*cos(2*theta2) + 1/4*IL4xy*theta4d^2*cos(2*theta1) - 1/8*IL4xx*theta4d^2*cos(2*theta1 + 2*theta2 - 2*theta3) - 1/8*IL4xx*theta4d^2*cos(2*theta1 + 2*theta2 + 2*theta3) - 1/4*IL4xy*theta4d^2*cos(2*theta1 + 2*theta2 + 2*theta3) + 1/4*IL4xx*theta4d^2*sin(2*theta1) - 1/4*IL4yy*theta4d^2*sin(2*theta1) + Im1zz*Kr1^2*theta1dd + 1/2*Im2xx*Kr2^2*theta2dd + 1/2*Im2yy*Kr2^2*theta2dd + 1/4*IL4xy*theta4dd*cos(theta2 + theta3) - IL4zz*theta4dd*cos(theta2 + theta3) + 1/4*IL4xy*theta4d^2*sin(2*theta1 + 2*theta2 + 2*theta3) + 1/4*IL4yy*theta4d^2*sin(2*theta1 + 2*theta2 + 2*theta3) - 1/4*IL4xy*theta4dd*sin(theta2 + theta3) - 1/4*IL4xx*theta4d^2*cos(2*theta1 + 2*theta2) - 1/4*IL4xy*theta4d^2*cos(2*theta1 + 2*theta2) - 1/8*IL4xx*theta4d^2*cos(2*theta2 - 2*theta3) - 1/8*IL4xx*theta4d^2*cos(2*theta2 + 2*theta3) + 1/8*IL4xy*theta4d^2*cos(2*theta1 - 2*theta3) + 1/8*IL4xy*theta4d^2*cos(2*theta1 + 2*theta3) + 1/2*L2^2*mL2*theta1dd + L2^2*mL2*theta2dd + 1/2*L3^2*mL3*theta1dd + L3^2*mL3*theta2dd - 1/4*L3^2*mL3*theta3dd + 1/2*L4^2*mL4*theta1dd + L4^2*mL4*theta2dd + L4^2*mL4*theta3dd + IL2yz*theta1dd*cos(theta1) + IL2yz*theta2dd*cos(theta1) + IL3yz*theta1dd*cos(theta1) + IL3yz*theta2dd*cos(theta1) + IL3yz*theta3dd*cos(theta1) + IL4yz*theta1dd*cos(theta1) + IL4yz*theta2dd*cos(theta1) + IL4yz*theta3dd*cos(theta1) + Im3yz*theta1dd*cos(theta1) + Im3yz*theta2dd*cos(theta1) + Im4yz*theta1dd*cos(theta1) + Im4yz*theta2dd*cos(theta1) + Im4yz*theta3dd*cos(theta1) + 1/8*IL4xx*theta4d^2*sin(2*theta1 - 2*theta3) + 1/8*IL4xx*theta4d^2*sin(2*theta1 + 2*theta3) + 1/4*IL4xy*theta4d^2*sin(2*theta1 + 2*theta2) - 1/8*IL4xy*theta4d^2*sin(2*theta1 - 2*theta3) + 1/8*IL4xy*theta4d^2*sin(2*theta1 + 2*theta3) - 1/32*IL4xy*theta4d^2*sin(2*theta3 - 2*theta2) - 1/32*IL4xy*theta4d^2*sin(2*theta2 - 2*theta3) - 1/32*IL4yy*theta4d^2*sin(2*theta3 - 2*theta2) - 1/32*IL4yy*theta4d^2*sin(2*theta2 - 2*theta3) - 1/4*IL4yy*theta4d^2*sin(2*theta2 + 2*theta3) + 1/2*IL4zz*theta4d^2*sin(2*theta2 + 2*theta3) - 1/8*IL4xx*theta4dd*cos(2*theta1 + theta2 - theta3) + 1/8*IL4xx*theta4dd*cos(2*theta1 - theta2 + theta3) + 1/8*IL4xy*theta4dd*cos(2*theta1 + theta2 - theta3) + 1/8*IL4xy*theta4dd*cos(2*theta1 - theta2 + theta3) - 1/4*IL4xz*theta4dd*cos(theta1 - theta2 - theta3) - IL4yz*theta4dd*cos(theta1 - theta2 - theta3) - 1/8*Im4xz*Kr4^2*theta4d^2*cos(theta1 - 2*theta2 - 2*theta3) - 3/8*Im4xz*Kr4^2*theta4d^2*cos(theta1 + 2*theta2 + 2*theta3) - 1/8*L4^2*mL4*theta4dd*sin(theta2 + 3*theta3) - 1/8*L4^2*mL4*theta4dd*sin(theta3 - 3*theta2) - 1/8*L4^2*mL4*theta4dd*sin(3*theta2 + theta3) + 3/8*L4^2*mL4*theta4dd*cos(theta1 + theta2 + theta3) + Im2xy*Kr2^2*theta2dd*sin(2*theta1) + 1/4*Im4xz*Kr4^2*theta4d^2*sin(theta1) - 3/16*IL4xx*theta2d*theta4d*cos(2*theta1 + theta2 + theta3) + 1/4*IL4xz*theta1d*theta4d*cos(theta1 + theta2 - theta3) - 3/16*IL4xx*theta3d*theta4d*cos(2*theta1 + theta2 + theta3) - 9/16*IL4xy*theta2d*theta4d*cos(2*theta1 + theta2 + theta3) - 9/16*IL4xy*theta3d*theta4d*cos(2*theta1 + theta2 + theta3) + 1/4*Im4xy*Kr4*theta4dd*cos(theta2 + theta3) - Im4zz*Kr4*theta4dd*cos(theta2 + theta3) - 1/8*Im4xz*Kr4^2*theta4d^2*sin(theta1 - 2*theta2 - 2*theta3) + 3/8*Im4xz*Kr4^2*theta4d^2*sin(theta1 + 2*theta2 + 2*theta3) - 1/4*Im4yz*Kr4^2*theta4d^2*sin(theta1 - 2*theta2 - 2*theta3) - 3/4*Im4yz*Kr4^2*theta4d^2*sin(theta1 + 2*theta2 + 2*theta3) - 3/16*IL4xx*theta2d*theta4d*sin(2*theta1 + theta2 + theta3) - 1/4*IL4xz*theta1d*theta4d*sin(theta1 + theta2 - theta3) - 3/16*IL4xx*theta3d*theta4d*sin(2*theta1 + theta2 + theta3) + 3/16*IL4xy*theta2d*theta4d*sin(2*theta1 + theta2 + theta3) + 3/16*IL4xy*theta3d*theta4d*sin(2*theta1 + theta2 + theta3) + 3/8*IL4yy*theta2d*theta4d*sin(2*theta1 + theta2 + theta3) + 3/8*IL4yy*theta3d*theta4d*sin(2*theta1 + theta2 + theta3) + 1/2*L2^2*mL2*theta1dd*cos(2*theta2) + 1/2*L3^2*mL3*theta2dd*cos(2*theta1) + 1/4*L3^2*mL3*theta3dd*cos(2*theta1) - 3/64*L4^2*mL4*theta4d^2*cos(theta1) - 1/4*Im4xy*Kr4*theta4dd*sin(theta2 + theta3) + 1/8*L3^2*mL3*theta1dd*cos(2*theta1 - 2*theta2 - 2*theta3) - 1/8*L3^2*mL3*theta1dd*cos(2*theta1 + 2*theta2 + 2*theta3) - 1/8*L3^2*mL3*theta2dd*cos(2*theta1 - 2*theta2 - 2*theta3) - 3/8*L3^2*mL3*theta2dd*cos(2*theta1 + 2*theta2 + 2*theta3) - 1/8*L3^2*mL3*theta3dd*cos(2*theta1 - 2*theta2 - 2*theta3) - 1/8*L3^2*mL3*theta3dd*cos(2*theta1 + 2*theta2 + 2*theta3) - 3/32*L4^2*mL4*theta4dd*cos(2*theta1 - theta2 - theta3) - 1/32*L4^2*mL4*theta4dd*cos(2*theta1 - theta2 - 3*theta3) + 1/32*L4^2*mL4*theta4dd*cos(2*theta1 - 3*theta2 - theta3) + 1/32*L4^2*mL4*theta4dd*cos(2*theta1 - 3*theta2 - 3*theta3) - 5/32*L4^2*mL4*theta4dd*cos(2*theta1 + 3*theta2 + 3*theta3) + 1/32*L4^2*mL4*theta4d^2*cos(theta1 - 2*theta2 - 2*theta3) - 3/32*L4^2*mL4*theta4d^2*cos(theta1 + 2*theta2 + 2*theta3) + 1/128*L4^2*mL4*theta4d^2*cos(theta1 - 2*theta2 - 4*theta3) - 3/128*L4^2*mL4*theta4d^2*cos(theta1 + 2*theta2 + 4*theta3) +

3/128*L4^2*mL4*theta4d^2*cos(theta1 - 4*theta2 - 2*theta3) - 5/128*L4^2*mL4*theta4d^2*cos(theta1 + 4*theta2 + 2*theta3) +
3/128*L4^2*mL4*theta4d^2*cos(theta1 - 4*theta2 - 4*theta3) - 5/128*L4^2*mL4*theta4d^2*cos(theta1 + 4*theta2 + 4*theta3) -
1/2*L2*g*mL2*cos(theta1 + theta2) - 1/8*Im4xz*Kr4^2*theta4d^2*cos(theta1 - 2*theta2) - 3/8*Im4xz*Kr4^2*theta4d^2*cos(theta1 + 2*theta2)
- 1/8*Im4xz*Kr4^2*theta4d^2*cos(theta1 - 2*theta3) + 1/8*Im4xz*Kr4^2*theta4d^2*cos(theta1 + 2*theta3) +
1/4*L3^2*mL3*theta3dd*sin(2*theta1) + 1/2*a2^2*mL3*theta1dd*cos(2*theta2) + 1/2*a2^2*mL4*theta1dd*cos(2*theta2) +
1/2*a2^2*mm3*theta1dd*cos(2*theta2) + 1/2*a2^2*mm4*theta1dd*cos(2*theta2) + 1/8*L3^2*mL3*theta1dd*sin(2*theta1 - 2*theta2 - 2*theta3)
- 1/8*L3^2*mL3*theta1dd*sin(2*theta1 + 2*theta2 + 2*theta3) + 1/8*L3^2*mL3*theta2dd*sin(2*theta1 - 2*theta2 - 2*theta3) -
1/8*L3^2*mL3*theta2dd*sin(2*theta1 + 2*theta2 + 2*theta3) - 1/8*L3^2*mL3*theta3dd*sin(2*theta1 - 2*theta2 - 2*theta3) -
1/8*L3^2*mL3*theta3dd*sin(2*theta1 + 2*theta2 + 2*theta3) + 7/32*L4^2*mL4*theta4dd*sin(2*theta1 - theta2 - theta3) +
1/32*L4^2*mL4*theta4dd*sin(2*theta1 - theta2 - 3*theta3) + 1/32*L4^2*mL4*theta4dd*sin(2*theta1 - 3*theta2 - theta3) +
1/32*L4^2*mL4*theta4dd*sin(2*theta1 - 3*theta2 - 3*theta3) - 3/32*L4^2*mL4*theta4dd*sin(2*theta1 + 3*theta2 + 3*theta3) -
1/64*L4^2*mL4*theta4d^2*sin(theta1 - 2*theta2 - 2*theta3) - 3/64*L4^2*mL4*theta4d^2*sin(theta1 + 2*theta2 + 2*theta3) +
1/128*L4^2*mL4*theta4d^2*sin(theta1 - 2*theta2 - 4*theta3) + 3/128*L4^2*mL4*theta4d^2*sin(theta1 + 2*theta2 + 4*theta3) -
3/128*L4^2*mL4*theta4d^2*sin(theta1 - 4*theta2 - 2*theta3) - 5/128*L4^2*mL4*theta4d^2*sin(theta1 + 4*theta2 + 2*theta3) -
3/128*L4^2*mL4*theta4d^2*sin(theta1 - 4*theta2 - 4*theta3) - 5/128*L4^2*mL4*theta4d^2*sin(theta1 + 4*theta2 + 4*theta3) -
1/16*IL4xx*theta2d*theta4d*cos(theta2 + theta3) - 1/16*IL4xx*theta3d*theta4d*cos(theta2 + theta3) + 1/16*IL4xy*theta2d*theta4d*cos(theta2 +
theta3) + 1/16*IL4xy*theta3d*theta4d*cos(theta2 + theta3) + Im2yz*Kr2*theta1dd*cos(theta1) + Im2yz*Kr2*theta2dd*cos(theta1) +
Im3yz*Kr3*theta3dd*cos(theta1) + 1/2*a2*g*mL3*cos(theta1 + theta2) + 1/2*a2*g*mL4*cos(theta1 + theta2) + 1/2*a2*g*mm3*cos(theta1 +
theta2) + 1/2*a2*g*mm4*cos(theta1 + theta2) - 1/8*Im4xz*Kr4^2*theta4d^2*sin(theta1 - 2*theta2) + 3/8*Im4xz*Kr4^2*theta4d^2*sin(theta1 +
2*theta2) + 1/8*Im4xz*Kr4^2*theta4d^2*sin(theta1 - 2*theta3) + 1/8*Im4xz*Kr4^2*theta4d^2*sin(theta1 + 2*theta3) -
1/8*Im4xx*Kr4*theta4dd*cos(2*theta1 + theta2 - theta3) + 1/8*Im4xx*Kr4*theta4dd*cos(2*theta1 - theta2 + theta3) +
1/8*Im4xy*Kr4*theta4dd*cos(2*theta1 + theta2 - theta3) + 1/8*Im4xy*Kr4*theta4dd*cos(2*theta1 - theta2 + theta3) -
1/4*Im4xz*Kr4*theta4dd*cos(theta1 - theta2 - theta3) - Im4yz*Kr4*theta4dd*cos(theta1 - theta2 - theta3) +
1/8*IL4xy*theta2d*theta4d*sin(theta2 + theta3) + 1/8*IL4xy*theta3d*theta4d*sin(theta2 + theta3) - 1/2*IL4zz*theta1d*theta4d*sin(theta2 +
theta3) - Im2xz*Kr2*theta1dd*sin(theta1) - Im2xz*Kr2*theta2dd*sin(theta1) - Im3xz*Kr3*theta3dd*sin(theta1) +
1/32*L3^2*mL3*theta1dd*cos(- 2*theta2 - 2*theta3) + 1/32*L3^2*mL3*theta1dd*cos(2*theta3 - 2*theta2) -
1/32*L3^2*mL3*theta1dd*cos(2*theta2 - 2*theta3) + 15/32*L3^2*mL3*theta1dd*cos(2*theta2 + 2*theta3) -
3/4*L3^2*mL3*theta3dd*cos(2*theta2 + 2*theta3) + 1/32*L4^2*mL4*theta1dd*cos(- 2*theta2 - 2*theta3) +
1/32*L4^2*mL4*theta1dd*cos(2*theta3 - 2*theta2) - 1/32*L4^2*mL4*theta1dd*cos(2*theta2 - 2*theta3) +
15/32*L4^2*mL4*theta1dd*cos(2*theta2 + 2*theta3) - 5/64*L4^2*mL4*theta4dd*cos(- theta2 - theta3) - 1/64*L4^2*mL4*theta4dd*cos(- theta2
- 3*theta3) - 1/128*L4^2*mL4*theta4dd*cos(3*theta3 - theta2) - 3/128*L4^2*mL4*theta4dd*cos(- 3*theta2 - theta3) +
1/32*L4^2*mL4*theta4dd*cos(3*theta2 - theta3) - 1/128*L4^2*mL4*theta4dd*cos(- 3*theta2 - 3*theta3) -
15/128*L4^2*mL4*theta4dd*cos(3*theta2 + 3*theta3) + 3/128*L4^2*mL4*theta4d^2*cos(theta1 - 2*theta2) -
9/128*L4^2*mL4*theta4d^2*cos(theta1 + 2*theta2) - 3/128*L4^2*mL4*theta4d^2*cos(theta1 - 2*theta3) -
3/128*L4^2*mL4*theta4d^2*cos(theta1 + 2*theta3) + 1/8*Im4xx*Kr4*theta4dd*sin(2*theta1 + theta2 - theta3) +
1/8*Im4xx*Kr4*theta4dd*sin(2*theta1 - theta2 + theta3) + 1/8*Im4xy*Kr4*theta4dd*sin(2*theta1 + theta2 - theta3) -
1/8*Im4xy*Kr4*theta4dd*sin(2*theta1 - theta2 + theta3) + 3/4*Im4xz*Kr4*theta4dd*sin(theta1 - theta2 - theta3) -
1/4*Im4xx*Kr4^2*theta4d^2*cos(2*theta2) + 1/4*Im4xy*Kr4^2*theta4d^2*cos(2*theta1) - 1/2*L4*g*mL4*cos(theta1 - theta2 - theta3) -
1/8*Im4xx*Kr4^2*theta4d^2*cos(2*theta1 + 2*theta2 - 2*theta3) - 1/8*Im4xx*Kr4^2*theta4d^2*cos(2*theta1 + 2*theta2 + 2*theta3) -
1/4*Im4xy*Kr4^2*theta4d^2*cos(2*theta1 + 2*theta2 + 2*theta3) + 1/4*L2^2*mL2*theta1dd*sin(2*theta1 - 2*theta2) -
1/4*L2^2*mL2*theta1dd*sin(2*theta1 + 2*theta2) + 1/4*L2^2*mL2*theta2dd*sin(2*theta1 - 2*theta2) - 1/4*L2^2*mL2*theta2dd*sin(2*theta1 +
2*theta2) - 1/4*L3^2*mL3*theta1dd*sin(2*theta2 + 2*theta3) - 1/4*L3^2*mL3*theta2dd*sin(2*theta2 + 2*theta3) -
1/8*L4^2*mL4*theta4dd*sin(3*theta2 - theta3) - 1/8*L4^2*mL4*theta4dd*sin(3*theta2 + 3*theta3) - 3/128*L4^2*mL4*theta4d^2*sin(theta1 -
2*theta2) - 9/128*L4^2*mL4*theta4d^2*sin(theta1 + 2*theta2) - 3/128*L4^2*mL4*theta4d^2*sin(theta1 - 2*theta3) +
3/128*L4^2*mL4*theta4d^2*sin(theta1 + 2*theta3) + 7/32*L4^2*mL4*theta4dd*cos(2*theta1 + theta2 + theta3) +
1/4*Im4xx*Kr4^2*theta4d^2*sin(2*theta1) - 1/4*Im4yy*Kr4^2*theta4d^2*sin(2*theta1) + 1/2*L3*g*mL3*sin(theta1 - theta2 - theta3) +
1/32*d4^2*mL4*theta1dd*cos(- 2*theta2 - 2*theta3) + 1/32*d4^2*mL4*theta1dd*cos(2*theta3 - 2*theta2) -
1/32*d4^2*mL4*theta1dd*cos(2*theta2 - 2*theta3) + 15/32*d4^2*mL4*theta1dd*cos(2*theta2 + 2*theta3) + 1/32*d4^2*mm4*theta1dd*cos(-
2*theta2 - 2*theta3) + 1/32*d4^2*mm4*theta1dd*cos(2*theta3 - 2*theta2) - 1/32*d4^2*mm4*theta1dd*cos(2*theta2 - 2*theta3) +
15/32*d4^2*mm4*theta1dd*cos(2*theta2 + 2*theta3) - d4^2*mm4*theta2dd*cos(2*theta2 + 2*theta3) - 1/2*d4^2*mm4*theta3dd*cos(2*theta2 +
2*theta3) - 3/16*IL4xx*theta2d*theta4d*cos(2*theta1 + theta2 - theta3) - 1/16*IL4xx*theta2d*theta4d*cos(2*theta1 - theta2 + theta3) -
3/16*IL4xx*theta3d*theta4d*cos(2*theta1 + theta2 - theta3) - 1/16*IL4xx*theta3d*theta4d*cos(2*theta1 - theta2 + theta3) -
3/16*IL4xy*theta2d*theta4d*cos(2*theta1 + theta2 - theta3) + 1/16*IL4xy*theta2d*theta4d*cos(2*theta1 - theta2 + theta3) -
3/16*IL4xy*theta3d*theta4d*cos(2*theta1 + theta2 - theta3) + 1/16*IL4xy*theta3d*theta4d*cos(2*theta1 - theta2 + theta3) +
1/4*Im4xy*Kr4*theta4dd*cos(theta2 - theta3) + 1/4*Im4xy*Kr4^2*theta4d^2*sin(2*theta1 + 2*theta2 + 2*theta3) +
1/4*Im4yy*Kr4^2*theta4d^2*sin(2*theta1 + 2*theta2 + 2*theta3) - 1/2*d4*g*mL4*cos(theta1 - theta2 - theta3) - 1/2*d4*g*mm4*cos(theta1 -
theta2 - theta3) + 3/32*L4^2*mL4*theta4dd*sin(2*theta1 + theta2 + theta3) - 3/16*IL4xx*theta2d*theta4d*sin(2*theta1 + theta2 - theta3) +
1/16*IL4xx*theta2d*theta4d*sin(2*theta1 - theta2 + theta3) - 3/16*IL4xx*theta3d*theta4d*sin(2*theta1 + theta2 - theta3) +
1/16*IL4xx*theta3d*theta4d*sin(2*theta1 - theta2 + theta3) + 3/16*IL4xy*theta2d*theta4d*sin(2*theta1 + theta2 - theta3) +
1/16*IL4xy*theta2d*theta4d*sin(2*theta1 - theta2 + theta3) + 3/16*IL4xy*theta3d*theta4d*sin(2*theta1 + theta2 - theta3) +
1/16*IL4xy*theta3d*theta4d*sin(2*theta1 - theta2 + theta3) - 9/64*L4^2*mL4*theta4d^2*cos(3*theta1) -
5/32*L4^2*mL4*theta4d^2*cos(2*theta2) - 1/64*L4^2*mL4*theta4d^2*cos(4*theta2) + 1/4*Im4xy*Kr4*theta4dd*sin(theta2 - theta3) -
1/4*Im4xz*Kr4*theta4dd*cos(theta1 + theta2 + theta3) - 1/4*L4^2*mL4*theta4d^2*cos(2*theta1 + 2*theta2 + 2*theta3) +
9/512*L4^2*mL4*theta4d^2*cos(2*theta2 - 3*theta1 - 2*theta3) - 9/512*L4^2*mL4*theta4d^2*cos(2*theta2 - 3*theta1 + 2*theta3) -
7/512*L4^2*mL4*theta4d^2*cos(3*theta1 - 2*theta2 - 2*theta3) - 9/512*L4^2*mL4*theta4d^2*cos(3*theta1 - 2*theta2 + 2*theta3) -
5/32*L4^2*mL4*theta4d^2*cos(3*theta1 + 2*theta2 + 2*theta3) - 1/8*L4^2*mL4*theta4d^2*cos(2*theta1 + 2*theta2 + 4*theta3) -
1/16*L4^2*mL4*theta4d^2*cos(2*theta1 - 4*theta2 - 2*theta3) - 3/16*L4^2*mL4*theta4d^2*cos(2*theta1 + 4*theta2 + 2*theta3) +
15/1024*L4^2*mL4*theta4d^2*cos(2*theta2 - 4*theta1 - 2*theta3) + 9/1024*L4^2*mL4*theta4d^2*cos(2*theta2 - 4*theta1 + 2*theta3) -
17/1024*L4^2*mL4*theta4d^2*cos(4*theta1 - 2*theta2 - 2*theta3) - 23/1024*L4^2*mL4*theta4d^2*cos(4*theta1 - 2*theta2 + 2*theta3) +
3/128*L4^2*mL4*theta4d^2*cos(4*theta1 + 2*theta2 - 2*theta3) + 3/128*L4^2*mL4*theta4d^2*cos(4*theta1 + 2*theta2 + 2*theta3) -

```
9/512*L4^2*mL4*theta4d^2*cos(2*theta2 - 3*theta1 + 4*theta3) + 5/512*L4^2*mL4*theta4d^2*cos(3*theta1 - 2*theta2 - 4*theta3) -
5/128*L4^2*mL4*theta4d^2*cos(3*theta1 + 2*theta2 + 4*theta3) + 1/128*L4^2*mL4*theta4d^2*cos(3*theta1 - 4*theta2 - 2*theta3) -
7/128*L4^2*mL4*theta4d^2*cos(3*theta1 + 4*theta2 + 2*theta3) - 1/16*L4^2*mL4*theta4d^2*cos(2*theta1 - 4*theta2 - 4*theta3) -
3/16*L4^2*mL4*theta4d^2*cos(2*theta1 + 4*theta2 + 4*theta3) + 3/512*L4^2*mL4*theta4d^2*cos(2*theta2 - 4*theta1 - 4*theta3) +
3/1024*L4^2*mL4*theta4d^2*cos(2*theta2 - 4*theta1 + 4*theta3) + 1/128*L4^2*mL4*theta4d^2*cos(2*theta3 - 4*theta2 - 4*theta1) -
9/2048*L4^2*mL4*theta4d^2*cos(4*theta2 - 4*theta1 - 2*theta3) - 3/1024*L4^2*mL4*theta4d^2*cos(4*theta2 - 4*theta1 + 2*theta3) -
3/1024*L4^2*mL4*theta4d^2*cos(4*theta1 - 2*theta2 - 4*theta3) - 3/512*L4^2*mL4*theta4d^2*cos(4*theta1 - 2*theta2 + 4*theta3) +
3/1024*L4^2*mL4*theta4d^2*cos(4*theta1 - 4*theta2 - 2*theta3) + 9/2048*L4^2*mL4*theta4d^2*cos(4*theta1 - 4*theta2 + 2*theta3) -
1/128*L4^2*mL4*theta4d^2*cos(4*theta1 + 4*theta2 - 2*theta3) + 1/32*L4^2*mL4*theta4d^2*cos(4*theta1 + 4*theta2 + 2*theta3) +
1/128*L4^2*mL4*theta4d^2*cos(3*theta1 - 4*theta2 - 4*theta3) - 7/128*L4^2*mL4*theta4d^2*cos(3*theta1 + 4*theta2 + 4*theta3) -
1/512*L4^2*mL4*theta4d^2*cos(4*theta2 - 4*theta1 - 4*theta3) + 5/2048*L4^2*mL4*theta4d^2*cos(4*theta2 - 4*theta1 + 4*theta3) -
5/2048*L4^2*mL4*theta4d^2*cos(4*theta1 - 4*theta2 - 4*theta3) + 1/512*L4^2*mL4*theta4d^2*cos(4*theta1 - 4*theta2 + 4*theta3) +
1/64*L4^2*mL4*theta4d^2*cos(4*theta1 + 4*theta2 + 4*theta3) + 1/2*L2*g*mL2*cos(theta1 - theta2) -
1/4*Im4xx*Kr4^2*theta4d^2*cos(2*theta1 + 2*theta2) - 1/4*Im4xy*Kr4^2*theta4d^2*cos(2*theta1 + 2*theta2) -
1/8*Im4xx*Kr4^2*theta4d^2*cos(2*theta2 - 2*theta3) - 1/8*Im4xx*Kr4^2*theta4d^2*cos(2*theta2 + 2*theta3) +
1/8*Im4xy*Kr4^2*theta4d^2*cos(2*theta1 - 2*theta3) + 1/8*Im4xy*Kr4^2*theta4d^2*cos(2*theta1 + 2*theta3) +
1/16*L4^2*mL4*theta4d^2*sin(2*theta1) + 9/64*L4^2*mL4*theta4d^2*sin(2*theta2) - 1/16*L4^2*mL4*theta4d^2*sin(4*theta1) -
1/8*L4^2*mL4*theta4d^2*sin(4*theta2) - 51/64*L4^2*mL4*theta4dd*cos(theta2 + theta3) + 1/4*Im4xz*Kr4*theta4dd*sin(theta1 + theta2 +
theta3) + 1/8*L4^2*mL4*theta4d^2*sin(2*theta1 + 2*theta2 - 2*theta3) + 3/8*L4^2*mL4*theta4d^2*sin(2*theta1 + 2*theta2 + 2*theta3) -
9/512*L4^2*mL4*theta4d^2*sin(2*theta2 - 3*theta1 - 2*theta3) - 27/512*L4^2*mL4*theta4d^2*sin(2*theta2 - 3*theta1 + 2*theta3) -
19/512*L4^2*mL4*theta4d^2*sin(3*theta1 - 2*theta2 - 2*theta3) - 9/512*L4^2*mL4*theta4d^2*sin(3*theta1 - 2*theta2 + 2*theta3) -
5/64*L4^2*mL4*theta4d^2*sin(3*theta1 + 2*theta2 + 2*theta3) - 1/8*L4^2*mL4*theta4d^2*sin(2*theta1 + 2*theta2 + 4*theta3) +
1/16*L4^2*mL4*theta4d^2*sin(2*theta1 - 4*theta2 - 2*theta3) - 3/16*L4^2*mL4*theta4d^2*sin(2*theta1 + 4*theta2 + 2*theta3) -
7/2048*L4^2*mL4*theta4d^2*sin(2*theta2 - 4*theta1 - 2*theta3) - 1/2048*L4^2*mL4*theta4d^2*sin(2*theta2 - 4*theta1 + 2*theta3) -
33/2048*L4^2*mL4*theta4d^2*sin(4*theta1 - 2*theta2 - 2*theta3) - 7/2048*L4^2*mL4*theta4d^2*sin(4*theta1 - 2*theta2 + 2*theta3) -
3/64*L4^2*mL4*theta4d^2*sin(4*theta1 + 2*theta2 + 2*theta3) - 3/512*L4^2*mL4*theta4d^2*sin(2*theta2 - 3*theta1 + 4*theta3) -
7/512*L4^2*mL4*theta4d^2*sin(3*theta1 - 2*theta2 - 4*theta3) + 5/128*L4^2*mL4*theta4d^2*sin(3*theta1 + 2*theta2 + 4*theta3) -
1/128*L4^2*mL4*theta4d^2*sin(3*theta1 - 4*theta2 - 2*theta3) - 7/128*L4^2*mL4*theta4d^2*sin(3*theta1 + 4*theta2 + 2*theta3) -
1/32*L4^2*mL4*theta4d^2*sin(2*theta1 - 4*theta2 - 4*theta3) + 3/32*L4^2*mL4*theta4d^2*sin(2*theta1 + 4*theta2 + 4*theta3) +
1/1024*L4^2*mL4*theta4d^2*sin(2*theta2 - 4*theta1 - 4*theta3) + 5/2048*L4^2*mL4*theta4d^2*sin(2*theta2 - 4*theta1 + 4*theta3) +
3/1024*L4^2*mL4*theta4d^2*sin(4*theta2 - 4*theta1 - 2*theta3) - 3/1024*L4^2*mL4*theta4d^2*sin(4*theta2 - 4*theta1 + 2*theta3) -
11/2048*L4^2*mL4*theta4d^2*sin(4*theta1 - 2*theta2 - 4*theta3) + 1/1024*L4^2*mL4*theta4d^2*sin(4*theta1 - 2*theta2 + 4*theta3) -
3/128*L4^2*mL4*theta4d^2*sin(4*theta1 + 2*theta2 + 4*theta3) - 3/1024*L4^2*mL4*theta4d^2*sin(4*theta1 - 4*theta2 - 2*theta3) +
3/1024*L4^2*mL4*theta4d^2*sin(4*theta1 - 4*theta2 + 2*theta3) - 3/1024*L4^2*mL4*theta4d^2*sin(4*theta2 - 3*theta1 + 4*theta3) -
11/1024*L4^2*mL4*theta4d^2*sin(3*theta1 - 4*theta2 - 4*theta3) - 7/128*L4^2*mL4*theta4d^2*sin(3*theta1 + 4*theta2 + 4*theta3) +
1/2048*L4^2*mL4*theta4d^2*sin(4*theta2 - 4*theta1 - 4*theta3) + 1/2048*L4^2*mL4*theta4d^2*sin(4*theta2 - 4*theta1 + 4*theta3) +
1/2048*L4^2*mL4*theta4d^2*sin(4*theta1 - 4*theta2 - 4*theta3) + 1/2048*L4^2*mL4*theta4d^2*sin(4*theta1 - 4*theta2 + 4*theta3) -
1/16*IL4xx*theta2d*theta4d*cos(theta2 - theta3) + 1/16*IL4xx*theta2d*theta4d*cos(theta3 - theta2) - 1/16*IL4xx*theta3d*theta4d*cos(theta2 -
theta3) + 1/16*IL4xx*theta3d*theta4d*cos(theta3 - theta2) - 3/16*IL4xy*theta2d*theta4d*cos(theta2 - theta3) +
1/16*IL4xy*theta2d*theta4d*cos(theta3 - theta2) - 3/16*IL4xy*theta3d*theta4d*cos(theta2 - theta3) + 1/16*IL4xy*theta3d*theta4d*cos(theta3 -
theta2) - 1/2*Im3xx*Kr3*theta3dd*cos(2*theta1) + 1/2*Im3yy*Kr3*theta3dd*cos(2*theta1) + 1/2*L4*g*mL4*cos(theta1 + theta2 + theta3) -
1/2*a2*g*mL3*cos(theta1 - theta2) - 1/2*a2*g*mL4*cos(theta1 - theta2) - 1/2*a2*g*mm3*cos(theta1 - theta2) - 1/2*a2*g*mm4*cos(theta1 -
theta2) + 1/8*Im4xx*Kr4^2*theta4d^2*sin(2*theta1 - 2*theta3) + 1/8*Im4xx*Kr4^2*theta4d^2*sin(2*theta1 + 2*theta3) +
1/4*Im4xy*Kr4^2*theta4d^2*sin(2*theta1 + 2*theta2) - 1/8*Im4xy*Kr4^2*theta4d^2*sin(2*theta1 - 2*theta3) +
1/8*Im4xy*Kr4^2*theta4d^2*sin(2*theta1 + 2*theta3) - 1/32*Im4xy*Kr4^2*theta4d^2*sin(2*theta3 - 2*theta2) -
1/32*Im4xy*Kr4^2*theta4d^2*sin(2*theta2 - 2*theta3) - 1/32*Im4yy*Kr4^2*theta4d^2*sin(2*theta3 - 2*theta2) -
1/32*Im4yy*Kr4^2*theta4d^2*sin(2*theta2 - 2*theta3) - 1/4*Im4yy*Kr4^2*theta4d^2*sin(2*theta2 + 2*theta3) +
1/2*Im4zz*Kr4^2*theta4d^2*sin(2*theta2 + 2*theta3) + 1/8*Im4xx*Kr4*theta4dd*cos(2*theta1 - theta2 - theta3) +
1/8*Im4xy*Kr4*theta4dd*cos(2*theta1 - theta2 - theta3) - 1/4*Im4yy*Kr4*theta4dd*cos(2*theta1 - theta2 - theta3) -
1/8*L4^2*mL4*theta4dd*sin(theta2 + theta3) + 1/8*IL4xy*theta2d*theta4d*sin(theta2 - theta3) + 1/8*IL4xy*theta3d*theta4d*sin(theta2 - theta3)
+ Im3xy*Kr3*theta3dd*sin(2*theta1) - 1/2*L3*g*mL3*sin(theta1 + theta2 + theta3) + 1/4*IL4xz*theta1d*theta4d*cos(theta1 + theta2 + theta3) -
1/2*IL4xz*theta2d*theta4d*cos(theta1 + theta2 + theta3) - 1/2*IL4xz*theta3d*theta4d*cos(theta1 + theta2 + theta3) -
1/8*L4^2*mL4*theta4d^2*cos(2*theta1 + 2*theta2) + 1/16*L4^2*mL4*theta4d^2*cos(2*theta1 - 2*theta3) -
1/16*L4^2*mL4*theta4d^2*cos(2*theta1 + 2*theta3) + 9/512*L4^2*mL4*theta4d^2*cos(2*theta2 - 3*theta1) -
21/512*L4^2*mL4*theta4d^2*cos(3*theta1 - 2*theta2) - 15/128*L4^2*mL4*theta4d^2*cos(3*theta1 + 2*theta2) -
9/512*L4^2*mL4*theta4d^2*cos(- 3*theta1 - 2*theta3) - 9/512*L4^2*mL4*theta4d^2*cos(2*theta3 - 3*theta1) -
27/512*L4^2*mL4*theta4d^2*cos(3*theta1 - 2*theta3) - 27/512*L4^2*mL4*theta4d^2*cos(3*theta1 + 2*theta3) +
1/2048*L4^2*mL4*theta4d^2*cos(- 2*theta2 - 2*theta3) - 15/2048*L4^2*mL4*theta4d^2*cos(2*theta3 - 2*theta2) -
17/2048*L4^2*mL4*theta4d^2*cos(2*theta2 - 2*theta3) - 545/2048*L4^2*mL4*theta4d^2*cos(2*theta2 + 2*theta3) +
15/1024*L4^2*mL4*theta4d^2*cos(2*theta2 - 4*theta1) - 31/1024*L4^2*mL4*theta4d^2*cos(4*theta1 - 2*theta2) +
3/64*L4^2*mL4*theta4d^2*cos(4*theta1 + 2*theta2) - 21/2048*L4^2*mL4*theta4d^2*cos(- 4*theta1 - 2*theta3) +
15/1024*L4^2*mL4*theta4d^2*cos(2*theta3 - 4*theta1) + 1/1024*L4^2*mL4*theta4d^2*cos(4*theta1 - 2*theta3) -
11/2048*L4^2*mL4*theta4d^2*cos(4*theta1 + 2*theta3) + 9/512*L4^2*mL4*theta4d^2*cos(4*theta3 - 3*theta1) -
9/512*L4^2*mL4*theta4d^2*cos(3*theta1 - 4*theta3) + 3/2048*L4^2*mL4*theta4d^2*cos(- 2*theta2 - 4*theta3) -
1/1024*L4^2*mL4*theta4d^2*cos(4*theta3 - 2*theta2) + 1/1024*L4^2*mL4*theta4d^2*cos(2*theta2 - 4*theta3) -
259/2048*L4^2*mL4*theta4d^2*cos(2*theta2 + 4*theta3) - 1/128*L4^2*mL4*theta4d^2*cos(- 4*theta1 - 4*theta2) +
7/1024*L4^2*mL4*theta4d^2*cos(4*theta2 - 4*theta1) - 7/1024*L4^2*mL4*theta4d^2*cos(4*theta1 - 4*theta2) +
3/128*L4^2*mL4*theta4d^2*cos(4*theta1 + 4*theta2) - 1/512*L4^2*mL4*theta4d^2*cos(- 4*theta2 - 2*theta3) -
49/256*L4^2*mL4*theta4d^2*cos(2*theta3 - 4*theta2) + 49/256*L4^2*mL4*theta4d^2*cos(4*theta2 - 2*theta3) -
143/512*L4^2*mL4*theta4d^2*cos(4*theta2 + 2*theta3) - 3/1024*L4^2*mL4*theta4d^2*cos(- 4*theta1 - 4*theta3) -
```

9/2048*L4^2*mL4*theta4d^2*cos(4*theta3 - 4*theta1) + 25/2048*L4^2*mL4*theta4d^2*cos(4*theta1 - 4*theta3) - 5/1024*L4^2*mL4*theta4d^2*cos(4*theta1 + 4*theta3) - 1/2048*L4^2*mL4*theta4d^2*cos(- 4*theta2 - 4*theta3) - 1/2048*L4^2*mL4*theta4d^2*cos(4*theta3 - 4*theta2) + 1/2048*L4^2*mL4*theta4d^2*cos(4*theta2 - 4*theta3) - 543/2048*L4^2*mL4*theta4d^2*cos(4*theta2 + 4*theta3) + 1/8*Im4xx*Kr4*theta4dd*sin(2*theta1 - theta2 - theta3) - 3/8*Im4xy*Kr4*theta4dd*sin(2*theta1 - theta2 - theta3) + 1/2*d4*g*mL4*cos(theta1 + theta2 + theta3) + 1/2*d4*g*mm4*cos(theta1 + theta2 + theta3) - 1/4*IL4xz*theta1d*theta4d*sin(theta1 + theta2 + theta3) + 1/2*IL4yz*theta1d*theta4d*sin(theta1 + theta2 + theta3) - 1/2*IL4yz*theta2d*theta4d*sin(theta1 + theta2 + theta3) - 1/2*IL4yz*theta3d*theta4d*sin(theta1 + theta2 + theta3) + 1/8*L4^2*mL4*theta4d^2*sin(2*theta1 + 2*theta2) + 1/16*L4^2*mL4*theta4d^2*sin(2*theta1 - 2*theta3) + 1/16*L4^2*mL4*theta4d^2*sin(2*theta1 + 2*theta3) - 33/512*L4^2*mL4*theta4d^2*sin(2*theta2 - 3*theta1) - 21/512*L4^2*mL4*theta4d^2*sin(3*theta1 - 2*theta2) - 15/128*L4^2*mL4*theta4d^2*sin(3*theta1 + 2*theta2) + 1/32*L4^2*mL4*theta4d^2*sin(2*theta1 - 4*theta2) - 3/32*L4^2*mL4*theta4d^2*sin(2*theta1 + 4*theta2) + 9/512*L4^2*mL4*theta4d^2*sin(- 3*theta1 - 2*theta3) + 63/1024*L4^2*mL4*theta4d^2*sin(2*theta3 - 3*theta1) - 9/1024*L4^2*mL4*theta4d^2*sin(3*theta1 - 2*theta3) + 45/512*L4^2*mL4*theta4d^2*sin(3*theta1 + 2*theta3) + 7/2048*L4^2*mL4*theta4d^2*sin(- 2*theta2 - 2*theta3) + 15/2048*L4^2*mL4*theta4d^2*sin(2*theta3 - 2*theta2) + 271/2048*L4^2*mL4*theta4d^2*sin(2*theta2 - 2*theta3) + 839/2048*L4^2*mL4*theta4d^2*sin(2*theta2 + 2*theta3) - 27/2048*L4^2*mL4*theta4d^2*sin(2*theta2 - 4*theta1) - 43/2048*L4^2*mL4*theta4d^2*sin(4*theta1 - 2*theta2) - 3/128*L4^2*mL4*theta4d^2*sin(4*theta1 + 2*theta2) - 1/32*L4^2*mL4*theta4d^2*sin(2*theta1 - 4*theta3) - 1/32*L4^2*mL4*theta4d^2*sin(2*theta1 + 4*theta3) + 3/1024*L4^2*mL4*theta4d^2*sin(4*theta2 - 3*theta1) + 3/1024*L4^2*mL4*theta4d^2*sin(3*theta1 - 4*theta2) + 3/2048*L4^2*mL4*theta4d^2*sin(- 4*theta1 - 2*theta3) + 9/2048*L4^2*mL4*theta4d^2*sin(2*theta3 - 4*theta1) - 55/2048*L4^2*mL4*theta4d^2*sin(4*theta1 - 2*theta3) - 61/2048*L4^2*mL4*theta4d^2*sin(4*theta1 + 2*theta3) + 9/1024*L4^2*mL4*theta4d^2*sin(4*theta3 - 3*theta1) + 9/1024*L4^2*mL4*theta4d^2*sin(3*theta1 - 4*theta3) + 1/2048*L4^2*mL4*theta4d^2*sin(- 2*theta2 - 4*theta3) + 3/1024*L4^2*mL4*theta4d^2*sin(4*theta3 - 2*theta2) + 3/1024*L4^2*mL4*theta4d^2*sin(2*theta2 - 4*theta3) - 223/2048*L4^2*mL4*theta4d^2*sin(2*theta2 + 4*theta3) + 5/1024*L4^2*mL4*theta4d^2*sin(4*theta2 - 4*theta1) + 5/1024*L4^2*mL4*theta4d^2*sin(4*theta1 - 4*theta2) + 1/2048*L4^2*mL4*theta4d^2*sin(- 4*theta2 - 2*theta3) - 65/512*L4^2*mL4*theta4d^2*sin(2*theta3 - 4*theta2) - 65/512*L4^2*mL4*theta4d^2*sin(4*theta2 - 2*theta3) - 511/2048*L4^2*mL4*theta4d^2*sin(4*theta2 + 2*theta3) + 3/2048*L4^2*mL4*theta4d^2*sin(4*theta3 - 4*theta1) + 3/2048*L4^2*mL4*theta4d^2*sin(4*theta1 - 4*theta3) + 3/64*L4^2*mL4*theta4d^2*sin(4*theta3 - 4*theta2) + 3/64*L4^2*mL4*theta4d^2*sin(4*theta2 - 4*theta3) + 1/8*L4^2*mL4*theta4d^2*sin(4*theta2 + 4*theta3) + 3/8*L4^2*mL4*theta4dd*cos(theta1 - theta2 - theta3) - 1/16*L4^2*mL4*theta4dd*cos(2*theta1 + theta2 - theta3) + 1/16*L4^2*mL4*theta4dd*cos(2*theta1 - theta2 + theta3) + 1/32*L4^2*mL4*theta4dd*cos(2*theta1 + theta2 + 3*theta3) - 1/32*L4^2*mL4*theta4dd*cos(2*theta1 + 3*theta2 + theta3) + 1/8*L4^2*mL4*theta4dd*cos(theta1 - 3*theta2 - 3*theta3) + 1/8*L4^2*mL4*theta4dd*cos(theta1 + 3*theta2 + 3*theta3) - 1/16*IL4xx*theta2d*theta4d*cos(2*theta1 - theta2 - theta3) - 1/16*IL4xx*theta3d*theta4d*cos(2*theta1 - theta2 - theta3) + 3/16*IL4xy*theta2d*theta4d*cos(2*theta1 - theta2 - theta3) + 3/16*IL4xy*theta3d*theta4d*cos(2*theta1 - theta2 - theta3) + 3/16*L4^2*mL4*theta4dd*sin(2*theta1 + theta2 - theta3) + 3/16*L4^2*mL4*theta4dd*sin(2*theta1 - theta2 + theta3) - 3/32*L4^2*mL4*theta4dd*sin(2*theta1 + theta2 + 3*theta3) - 3/32*L4^2*mL4*theta4dd*sin(2*theta1 + 3*theta2 + theta3) + 1/2*d4^2*mm4*theta3dd*cos(theta1) + 1/16*IL4xx*theta2d*theta4d*sin(2*theta1 - theta2 - theta3) + 1/16*IL4xx*theta3d*theta4d*sin(2*theta1 - theta2 - theta3) + 1/16*IL4xy*theta2d*theta4d*sin(2*theta1 - theta2 - theta3) + 1/16*IL4xy*theta3d*theta4d*sin(2*theta1 - theta2 - theta3) - 1/8*IL4yy*theta2d*theta4d*sin(2*theta1 - theta2 - theta3) - 1/8*IL4yy*theta3d*theta4d*sin(2*theta1 - theta2 - theta3) - 1/4*d4^2*mm4*theta3dd*cos(theta1 - 2*theta2 - 2*theta3) - 1/4*d4^2*mm4*theta3dd*cos(theta1 + 2*theta2 + 2*theta3) - 1/8*Im4xx*Kr4*theta4dd*cos(2*theta1 + theta2 + theta3) + 1/8*Im4xy*Kr4*theta4dd*cos(2*theta1 + theta2 + theta3) - 1/4*Im4xz*Kr4*theta4dd*cos(theta1 + theta2 - theta3) - 1/4*Im4xz*Kr4*theta4dd*cos(theta1 - theta2 + theta3) + 1/4*Im4yy*Kr4*theta4dd*cos(2*theta1 + theta2 + theta3) + L4*d4*mL4*theta1dd + 2*L4*d4*mL4*theta2dd + 2*L4*d4*mL4*theta3dd + 11/128*L4^2*mL4*theta4dd*cos(theta2 - theta3) - 11/128*L4^2*mL4*theta4dd*cos(theta3 - theta2) + 1/128*L4^2*mL4*theta4dd*cos(theta2 - 3*theta3) + 1/64*L4^2*mL4*theta4dd*cos(theta2 + 3*theta3) - 1/32*L4^2*mL4*theta4dd*cos(theta3 - 3*theta2) + 3/128*L4^2*mL4*theta4dd*cos(3*theta2 + theta3) + 1/8*Im4xx*Kr4*theta4dd*sin(2*theta1 + theta2 + theta3) + 3/8*Im4xy*Kr4*theta4dd*sin(2*theta1 + theta2 + theta3) - 1/4*Im4xz*Kr4*theta4dd*sin(theta1 + theta2 - theta3) + 1/4*Im4xz*Kr4*theta4dd*sin(theta1 - theta2 + theta3) + 1/16*IL4xx*theta2d*theta4d*cos(- theta2 - theta3) + 1/16*IL4xx*theta3d*theta4d*cos(- theta2 - theta3) + 1/16*IL4xy*theta2d*theta4d*cos(- theta2 - theta3) + 1/16*IL4xy*theta3d*theta4d*cos(- theta2 - theta3) - 1/2*Im2xx*Kr2^2*theta2dd*cos(2*theta1) + 1/2*Im2yy*Kr2^2*theta2dd*cos(2*theta1) - 1/8*L3*a2*mL3*theta1dd*sin(2*theta1 + theta3) - 1/4*L3*a2*mL3*theta1dd*sin(2*theta2 + theta3) + 1/8*L3*a2*mL3*theta2dd*sin(2*theta1 + theta3) - 1/4*L3*a2*mL3*theta2dd*sin(2*theta2 + theta3) + 3/32*L4*a2*mL4*theta4dd*sin(2*theta1 + theta2) + 1/8*L4*a2*mL4*theta4dd*sin(theta2 - 2*theta3) - 1/4*L4*a2*mL4*theta4dd*sin(theta2 + 2*theta3) - 1/8*L4*d4*mL4*theta4dd*sin(theta2 + 3*theta3) - 1/8*L4*d4*mL4*theta4dd*sin(theta3 - 3*theta2) - 1/8*L4*d4*mL4*theta4dd*sin(3*theta2 + theta3) + 1/16*L4^2*mL4*theta1d*theta4d*cos(theta2 + theta3) - 7/256*L4^2*mL4*theta2d*theta4d*cos(theta2 + theta3) - 7/256*L4^2*mL4*theta3d*theta4d*cos(theta2 + theta3) - 1/4*Im4xz*Kr4*theta1d*theta4d*sin(theta1 + theta2 + theta3) + 1/2*Im4yz*Kr4*theta1d*theta4d*sin(theta1 + theta2 + theta3) - 1/2*Im4yz*Kr4*theta2d*theta4d*sin(theta1 + theta2 + theta3) - 1/2*Im4yz*Kr4*theta3d*theta4d*sin(theta1 + theta2 + theta3) + 3/8*L4*d4*mL4*theta4dd*cos(theta1 + theta2 + theta3) + 1/8*a2*d4*mL4*theta1dd*cos(theta3 - 2*theta2) + 7/8*a2*d4*mL4*theta1dd*cos(2*theta2 + theta3) + 1/8*a2*d4*mm4*theta1dd*cos(theta3 - 2*theta2) + 7/8*a2*d4*mm4*theta1dd*cos(2*theta2 + theta3) - a2*d4*mm4*theta2dd*cos(2*theta2 + theta3) - 1/2... Output truncated. Text exceeds maximum line length of 25,000 characters for Command Window display.

$T_3 =$

IL1zz*theta1dd + 1/2*IL2xx*theta2dd + 1/2*IL2yy*theta2dd + IL2zz*theta1dd + IL3xx*theta2dd + IL3xx*theta3dd + IL3yy*theta2dd + IL3yy*theta3dd + IL3zz*theta1dd + IL4xx*theta2dd + IL4xx*theta3dd + IL4yy*theta2dd + IL4yy*theta3dd + IL4zz*theta1dd + Im2zz*theta1dd + 1/2*Im3xx*theta2dd + 1/2*Im3yy*theta2dd + Im3zz*theta1dd + Im4xx*theta2dd + Im4xx*theta3dd + Im4yy*theta2dd + Im4yy*theta3dd +

Im4zz*theta1dd + 1/2*a2^2*mL3*theta1dd + a2^2*mL3*theta2dd + 1/2*a2^2*mL4*theta1dd + a2^2*mL4*theta2dd + 1/2*a2^2*mm3*theta1dd + a2^2*mm3*theta2dd + 1/2*a2^2*mm4*theta1dd + a2^2*mm4*theta2dd + 1/2*d4^2*mL4*theta1dd + 2*d4^2*mL4*theta2dd + 2*d4^2*mL4*theta3dd + 1/2*d4^2*mm4*theta1dd + 3/2*d4^2*mm4*theta2dd + 5/4*d4^2*mm4*theta3dd - IL2xz*theta1dd*sin(theta1) - IL2xz*theta2dd*sin(theta1) - 2*IL3xz*theta1dd*sin(theta1) - IL3xz*theta2dd*sin(theta1) - IL3xz*theta3dd*sin(theta1) - 2*IL4xz*theta1dd*sin(theta1) - IL4xz*theta2dd*sin(theta1) - IL4xz*theta3dd*sin(theta1) - Im3xz*theta1dd*sin(theta1) - Im3xz*theta2dd*sin(theta1) - 2*Im4xz*theta1dd*sin(theta1) - Im4xz*theta2dd*sin(theta1) - Im4xz*theta3dd*sin(theta1) + 1/4*IL4xx*theta4dd*sin(2*theta1 + theta2 - theta3) + 1/4*IL4xx*theta4dd*sin(2*theta1 - theta2 + theta3) + 1/4*IL4xy*theta4dd*sin(2*theta1 + theta2 - theta3) - 1/4*IL4xy*theta4dd*sin(2*theta1 - theta2 + theta3) + 5/4*IL4xz*theta4dd*sin(theta1 - theta2 - theta3) + 1/2*IL4xy*theta4dd*cos(theta2 - theta3) + 1/2*IL4xy*theta4dd*sin(theta2 - theta3) - 1/4*IL4xz*theta4dd*cos(theta1 + theta2 + theta3) - 1/2*IL4yz*theta4dd*cos(theta1 + theta2 + theta3) + 3/4*IL4xz*theta4dd*sin(theta1 + theta2 + theta3) - 1/2*IL2xx*theta2dd*cos(2*theta1) + 1/2*IL2yy*theta2dd*cos(2*theta1) - IL3xx*theta2dd*cos(2*theta1) - IL3xx*theta3dd*cos(2*theta1) + IL3yy*theta2dd*cos(2*theta1) + IL3yy*theta3dd*cos(2*theta1) - IL4xx*theta2dd*cos(2*theta1) - IL4xx*theta3dd*cos(2*theta1) + IL4yy*theta2dd*cos(2*theta1) + IL4yy*theta3dd*cos(2*theta1) - 1/2*Im3xx*theta2dd*cos(2*theta1) + 1/2*Im3yy*theta2dd*cos(2*theta1) - Im4xx*theta2dd*cos(2*theta1) - Im4xx*theta3dd*cos(2*theta1) + Im4yy*theta2dd*cos(2*theta1) + Im4yy*theta3dd*cos(2*theta1) + 1/4*IL4xx*theta4dd*cos(2*theta1 - theta2 - theta3) + 1/4*IL4xy*theta4dd*cos(2*theta1 - theta2 - theta3) - 1/2*IL4yy*theta4dd*cos(2*theta1 - theta2 - theta3) - 3/8*IL4xz*theta4d^2*cos(theta1 - 2*theta2 - 2*theta3) - 5/8*IL4xz*theta4d^2*cos(theta1 + 2*theta2 + 2*theta3) + IL2xy*theta2dd*sin(2*theta1) + 2*IL3xy*theta2dd*sin(2*theta1) + 2*IL3xy*theta3dd*sin(2*theta1) + 2*IL4xy*theta2dd*sin(2*theta1) + 2*IL4xy*theta3dd*sin(2*theta1) + 1/4*IL4xz*theta4d^2*sin(theta1) + Im3xy*theta2dd*sin(2*theta1) + 2*Im4xy*theta2dd*sin(2*theta1) + 2*Im4xy*theta3dd*sin(2*theta1) + 1/2*Im3xx*Kr3*theta2dd + 1/2*Im3xx*Kr3*theta3dd + 1/2*Im3yy*Kr3*theta2dd + 1/2*Im3yy*Kr3*theta3dd + 1/4*IL4xx*theta4dd*sin(2*theta1 - theta2 - theta3) - 3/4*IL4xy*theta4dd*sin(2*theta1 - theta2 - theta3) - 3/8*IL4xz*theta4d^2*sin(theta1 - 2*theta2 - 2*theta3) + 5/8*IL4xz*theta4d^2*sin(theta1 + 2*theta2 + 2*theta3) - 3/4*IL4yz*theta4d^2*sin(theta1 - 2*theta2 - 2*theta3) - 5/4*IL4yz*theta4d^2*sin(theta1 + 2*theta2 + 2*theta3) - 1/8*IL4xz*theta4d^2*cos(theta1 - 2*theta2) - 3/8*IL4xz*theta4d^2*cos(theta1 + 2*theta2) + 1/8*IL4xz*theta4d^2*cos(theta1 - 2*theta3) + 3/8*IL4xz*theta4d^2*cos(theta1 + 2*theta3) - 1/8*IL4xz*theta4d^2*sin(theta1 - 2*theta2) + 3/8*IL4xz*theta4d^2*sin(theta1 + 2*theta2) - 1/8*IL4xz*theta4d^2*sin(theta1 - 2*theta3) + 3/8*IL4xz*theta4d^2*sin(theta1 + 2*theta3) - 1/4*IL4xx*theta4dd*cos(2*theta1 + theta2 + theta3) + 1/4*IL4xy*theta4dd*cos(2*theta1 + theta2 + theta3) - 1/4*IL4xz*theta4dd*cos(theta1 + theta2 - theta3) - 1/4*IL4xz*theta4dd*cos(theta1 - theta2 + theta3) + 1/2*IL4yy*theta4dd*cos(2*theta1 + theta2 + theta3) + 1/4*IL4xx*theta4dd*sin(2*theta1 + theta2 + theta3) + 3/4*IL4xy*theta4dd*sin(2*theta1 + theta2 + theta3) - 1/4*IL4xz*theta4dd*sin(theta1 + theta2 - theta3) + 1/4*IL4xz*theta4dd*sin(theta1 - theta2 + theta3) - 1/4*IL4xx*theta4d^2*cos(2*theta2) + 1/4*IL4xy*theta4d^2*cos(2*theta1) - 1/16*IL4xx*theta4d^2*cos(2*theta1 - 2*theta2 - 2*theta3) + 1/16*IL4xx*theta4d^2*cos(2*theta1 - 2*theta2 + 2*theta3) - 1/16*IL4xx*theta4d^2*cos(2*theta1 + 2*theta2 - 2*theta3) - 3/16*IL4xx*theta4d^2*cos(2*theta1 + 2*theta2 + 2*theta3) + 1/8*IL4xy*theta4d^2*cos(2*theta1 - 2*theta2 - 2*theta3) - 3/8*IL4xy*theta4d^2*cos(2*theta1 + 2*theta2 + 2*theta3) + 1/4*IL4xx*theta4d^2*sin(2*theta1) + 1/4*IL4xx*theta4d^2*sin(2*theta3) - 1/4*IL4yy*theta4d^2*sin(2*theta1) + Im1zz*Kr1^2*theta1dd + 1/2*Im2xx*Kr2^2*theta2dd + 1/2*Im2yy*Kr2^2*theta2dd + 1/2*Im3xx*Kr3^2*theta3dd + 1/2*Im3yy*Kr3^2*theta3dd + 1/2*IL4xy*theta4dd*cos(theta2 + theta3) - IL4zz*theta4dd*cos(theta2 + theta3) + 1/8*IL4xy*theta4d^2*sin(2*theta1 - 2*theta2 - 2*theta3) + 3/8*IL4xy*theta4d^2*sin(2*theta1 + 2*theta2 + 2*theta3) - 1/8*IL4yy*theta4d^2*sin(2*theta1 - 2*theta2 - 2*theta3) + 3/8*IL4yy*theta4d^2*sin(2*theta1 + 2*theta2 + 2*theta3) - 1/2*IL4xy*theta4dd*sin(theta2 + theta3) - 1/4*IL4xx*theta4d^2*cos(2*theta1 + 2*theta2) - 1/4*IL4xy*theta4d^2*cos(2*theta1 + 2*theta2) - 1/4*IL4xx*theta4d^2*cos(2*theta2 + 2*theta3) + 1/4*IL4xy*theta4d^2*cos(2*theta1 + 2*theta3) + 1/2*L2^2*mL2*theta1dd + L2^2*mL2*theta2dd + 1/2*L3^2*mL3*theta1dd + 3/4*L3^2*mL3*theta2dd + 3/4*L3^2*mL3*theta3dd + 1/2*L4^2*mL4*theta1dd + 2*L4^2*mL4*theta2dd + 2*L4^2*mL4*theta3dd + IL2yz*theta1dd*cos(theta1) + IL2yz*theta2dd*cos(theta1) + 2*IL3yz*theta1dd*cos(theta1) + IL3yz*theta2dd*cos(theta1) + IL3yz*theta3dd*cos(theta1) + 2*IL4yz*theta1dd*cos(theta1) + IL4yz*theta2dd*cos(theta1) + IL4yz*theta3dd*cos(theta1) + Im3yz*theta1dd*cos(theta1) + Im3yz*theta2dd*cos(theta1) + 2*Im4yz*theta1dd*cos(theta1) + Im4yz*theta2dd*cos(theta1) + Im4yz*theta3dd*cos(theta1) + 1/4*IL4xx*theta4d^2*sin(2*theta1 + 2*theta3) + 1/4*IL4xy*theta4d^2*sin(2*theta1 + 2*theta2) + 1/4*IL4xy*theta4d^2*sin(2*theta1 + 2*theta3) - 1/32*IL4xy*theta4d^2*sin(2*theta3 - 2*theta2) - 1/32*IL4xy*theta4d^2*sin(2*theta2 - 2*theta3) - 1/16*IL4yy*theta4d^2*sin(2*theta3 - 2*theta2) - 1/16*IL4yy*theta4d^2*sin(2*theta2 - 2*theta3) - 1/2*IL4yy*theta4d^2*sin(2*theta2 + 2*theta3) + IL4zz*theta4d^2*sin(2*theta2 + 2*theta3) - 1/4*IL4xx*theta4dd*cos(2*theta1 + theta2 - theta3) + 1/4*IL4xx*theta4dd*cos(2*theta1 - theta2 + theta3) + 1/4*IL4xy*theta4dd*cos(2*theta1 + theta2 - theta3) + 1/4*IL4xy*theta4dd*cos(2*theta1 - theta2 + theta3) - 1/4*IL4xz*theta4dd*cos(theta1 - theta2 - theta3) - 3/2*IL4yz*theta4dd*cos(theta1 - theta2 - theta3) - 3/8*Im4xz*Kr4^2*theta4d^2*cos(theta1 - 2*theta2 - 2*theta3) - 5/8*Im4xz*Kr4^2*theta4d^2*cos(theta1 + 2*theta2 + 2*theta3) - 1/8*L4^2*mL4*theta4dd*sin(theta2 + 3*theta3) - 1/8*L4^2*mL4*theta4dd*sin(theta3 - 3*theta2) - 1/8*L4^2*mL4*theta4dd*sin(3*theta2 + theta3) + 3/4*L4^2*mL4*theta4dd*cos(theta1 + theta2 + theta3) + Im2xy*Kr2^2*theta2dd*sin(2*theta1) + Im3xy*Kr3^2*theta3dd*sin(2*theta1) + 1/4*Im4xz*Kr4^2*theta4d^2*sin(theta1) - 1/4*L3*g*mL3*sin(theta1 + theta2 + 2*theta3) - 1/4*L4*g*mL4*sin(theta1 + theta2 + 2*theta3) - 1/4*IL4xx*theta2d*theta4d*cos(2*theta1 + theta2 + theta3) + 1/8*IL4xz*theta1d*theta4d*cos(theta1 + theta2 - theta3) - 1/8*IL4xz*theta1d*theta4d*cos(theta1 - theta2 + theta3) - 1/4*IL4xx*theta3d*theta4d*cos(2*theta1 + theta2 + theta3) - 3/4*IL4xy*theta2d*theta4d*cos(2*theta1 + theta2 + theta3) - 3/4*IL4xy*theta3d*theta4d*cos(2*theta1 + theta2 + theta3) + 1/2*Im4xy*Kr4*theta4dd*cos(theta2 + theta3) - Im4zz*Kr4*theta4dd*cos(theta2 + theta3) - 3/8*Im4xz*Kr4^2*theta4d^2*sin(theta1 - 2*theta2 - 2*theta3) + 5/8*Im4xz*Kr4^2*theta4d^2*sin(theta1 + 2*theta2 + 2*theta3) - 3/4*Im4yz*Kr4^2*theta4d^2*sin(theta1 - 2*theta2 - 2*theta3) - 5/4*Im4yz*Kr4^2*theta4d^2*sin(theta1 + 2*theta2 + 2*theta3) - 1/4*d4*g*mm4*cos(theta1 + theta2 + 2*theta3) - 1/4*IL4xx*theta2d*theta4d*sin(2*theta1 + theta2 + theta3) - 1/8*IL4xz*theta1d*theta4d*sin(theta1 + theta2 - theta3) - 1/8*IL4xz*theta1d*theta4d*sin(theta1 - theta2 + theta3) - 1/4*IL4xx*theta3d*theta4d*sin(2*theta1 + theta2 + theta3) + 1/4*IL4xy*theta2d*theta4d*sin(2*theta1 + theta2 + theta3) + 1/4*IL4xy*theta3d*theta4d*sin(2*theta1 + theta2 + theta3) + 1/2*IL4yy*theta2d*theta4d*sin(2*theta1 + theta2 + theta3) + 1/2*IL4yy*theta3d*theta4d*sin(2*theta1 + theta2 + theta3) + 1/2*L2^2*mL2*theta1dd*cos(2*theta2) + 3/4*L3^2*mL3*theta2dd*cos(2*theta1) + 1/4*L3^2*mL3*theta3dd*cos(2*theta1) - 3/64*L4^2*mL4*theta4d^2*cos(theta1) - 1/2*Im4xy*Kr4*theta4dd*sin(theta2 + theta3) - 1/4*d4*g*mL4*sin(theta1 + theta2 + 2*theta3) - 1/4*d4*g*mm4*sin(theta1 + theta2 + 2*theta3) + 1/8*L3^2*mL3*theta1dd*cos(2*theta1 - 2*theta2 - 2*theta3) - 1/8*L3^2*mL3*theta1dd*cos(2*theta1 + 2*theta2 + 2*theta3) - 1/4*L3^2*mL3*theta2dd*cos(2*theta1 - 2*theta2 - 2*theta3) - 1/2*L3^2*mL3*theta2dd*cos(2*theta1 + 2*theta2 + 2*theta3) - 1/8*L3^2*mL3*theta3dd*cos(2*theta1 - 2*theta2 - 2*theta3) - 1/8*L3^2*mL3*theta3dd*cos(2*theta1 + 2*theta2 + 2*theta3) - 1/4*L4^2*mL4*theta4dd*cos(2*theta1 - theta2 - theta3) - 1/16*L4^2*mL4*theta4dd*cos(2*theta1 - theta2 - 3*theta3) + 1/16*L4^2*mL4*theta4dd*cos(2*theta1 - 3*theta2 - theta3) + 1/8*L4^2*mL4*theta4dd*cos(2*theta1 - 3*theta2 - 3*theta3) - 1/4*L4^2*mL4*theta4dd*cos(2*theta1 + 3*theta2 + 3*theta3) + 3/32*L4^2*mL4*theta4d^2*cos(theta1 - 2*theta2 - 2*theta3) - 5/32*L4^2*mL4*theta4d^2*cos(theta1 + 2*theta2 + 2*theta3) +

5/128*L4^2*mL4*theta4d^2*cos(theta1 - 2*theta2 - 4*theta3) - 7/128*L4^2*mL4*theta4d^2*cos(theta1 + 2*theta2 + 4*theta3) +
5/128*L4^2*mL4*theta4d^2*cos(theta1 - 4*theta2 - 2*theta3) - 7/128*L4^2*mL4*theta4d^2*cos(theta1 + 4*theta2 + 2*theta3) +
7/128*L4^2*mL4*theta4d^2*cos(theta1 - 4*theta2 - 4*theta3) - 9/128*L4^2*mL4*theta4d^2*cos(theta1 + 4*theta2 + 4*theta3) -
1/2*L2*g*mL2*cos(theta1 + theta2) - 1/8*Im4xz*Kr4^2*theta4d^2*cos(theta1 - 2*theta2) - 3/8*Im4xz*Kr4^2*theta4d^2*cos(theta1 + 2*theta2)
+ 1/8*Im4xz*Kr4^2*theta4d^2*cos(theta1 - 2*theta3) + 3/8*Im4xz*Kr4^2*theta4d^2*cos(theta1 + 2*theta3) +
1/4*L3^2*mL3*theta2dd*sin(2*theta1) + 1/4*L3^2*mL3*theta3dd*sin(2*theta1) + 1/2*a2^2*mL3*theta1dd*cos(2*theta2) +
1/2*a2^2*mL4*theta1dd*cos(2*theta2) + 1/2*a2^2*mm3*theta1dd*cos(2*theta2) + 1/2*a2^2*mm4*theta1dd*cos(2*theta2) +
1/8*L3^2*mL3*theta1dd*sin(2*theta1 - 2*theta2 - 2*theta3) - 1/8*L3^2*mL3*theta1dd*sin(2*theta1 + 2*theta2 + 2*theta3) -
1/4*L3^2*mL3*theta2dd*sin(2*theta1 + 2*theta2 + 2*theta3) - 1/8*L3^2*mL3*theta3dd*sin(2*theta1 - 2*theta2 - 2*theta3) -
1/8*L3^2*mL3*theta3dd*sin(2*theta1 + 2*theta2 + 2*theta3) + 3/8*L4^2*mL4*theta4dd*sin(2*theta1 - theta2 - theta3) -
1/8*L4^2*mL4*theta4dd*sin(2*theta1 + 3*theta2 + 3*theta3) - 3/64*L4^2*mL4*theta4d^2*sin(theta1 - 2*theta2 - 2*theta3) -
5/64*L4^2*mL4*theta4d^2*sin(theta1 + 2*theta2 + 2*theta3) + 5/128*L4^2*mL4*theta4d^2*sin(theta1 - 2*theta2 - 4*theta3) +
7/128*L4^2*mL4*theta4d^2*sin(theta1 + 2*theta2 + 4*theta3) - 5/128*L4^2*mL4*theta4d^2*sin(theta1 - 4*theta2 - 2*theta3) -
7/128*L4^2*mL4*theta4d^2*sin(theta1 + 4*theta2 + 2*theta3) - 7/128*L4^2*mL4*theta4d^2*sin(theta1 - 4*theta2 - 4*theta3) -
9/128*L4^2*mL4*theta4d^2*sin(theta1 + 4*theta2 + 4*theta3) - 1/4*L3*g*mL3*sin(theta1 + theta2) - 1/4*L4*g*mL4*sin(theta1 + theta2) +
1/4*d4^2*mm4*theta3dd*cos(2*theta1) - 1/16*IL4xx*theta2d*theta4d*cos(theta2 + theta3) - 1/16*IL4xx*theta3d*theta4d*cos(theta2 + theta3) +
1/8*IL4xy*theta2d*theta4d*cos(theta2 + theta3) + 1/8*IL4xy*theta3d*theta4d*cos(theta2 + theta3) + Im2yz*Kr2*theta1dd*cos(theta1) +
Im2yz*Kr2*theta2dd*cos(theta1) + Im3yz*Kr3*theta1dd*cos(theta1) + Im3yz*Kr3*theta3dd*cos(theta1) + 1/2*a2*g*mL3*cos(theta1 + theta2) +
1/2*a2*g*mL4*cos(theta1 + theta2) + 1/2*a2*g*mm3*cos(theta1 + theta2) + 1/2*a2*g*mm4*cos(theta1 + theta2) -
1/8*Im4xz*Kr4^2*theta4d^2*sin(theta1 - 2*theta2) + 3/8*Im4xz*Kr4^2*theta4d^2*sin(theta1 + 2*theta2) -
1/8*Im4xz*Kr4^2*theta4d^2*sin(theta1 - 2*theta3) + 3/8*Im4xz*Kr4^2*theta4d^2*sin(theta1 + 2*theta3) -
1/8*d4^2*mm4*theta3dd*cos(2*theta1 - 2*theta2 - 2*theta3) - 1/8*d4^2*mm4*theta3dd*cos(2*theta1 + 2*theta2 + 2*theta3) +
1/4*d4*g*mm4*cos(theta1 + theta2) - 1/4*Im4xx*Kr4*theta4dd*cos(2*theta1 + theta2 - theta3) + 1/4*Im4xx*Kr4*theta4dd*cos(2*theta1 - theta2
+ theta3) + 1/4*Im4xy*Kr4*theta4dd*cos(2*theta1 + theta2 - theta3) + 1/4*Im4xy*Kr4*theta4dd*cos(2*theta1 - theta2 + theta3) -
1/4*Im4xz*Kr4*theta4dd*cos(theta1 - theta2 - theta3) - 3/2*Im4yz*Kr4*theta4dd*cos(theta1 - theta2 - theta3) +
1/4*IL4xy*theta2d*theta4d*sin(theta2 + theta3) + 1/4*IL4xy*theta3d*theta4d*sin(theta2 + theta3) - IL4zz*theta1d*theta4d*sin(theta2 + theta3) -
Im2xz*Kr2*theta1dd*sin(theta1) - Im2xz*Kr2*theta2dd*sin(theta1) - Im3xz*Kr3*theta1dd*sin(theta1) - Im3xz*Kr3*theta3dd*sin(theta1) -
1/4*d4*g*mL4*sin(theta1 + theta2) - 1/4*d4*g*mm4*sin(theta1 + theta2) + 1/32*L3^2*mL3*theta1dd*cos(- 2*theta2 - 2*theta3) +
1/32*L3^2*mL3*theta1dd*cos(2*theta3 - 2*theta2) - 1/32*L3^2*mL3*theta1dd*cos(2*theta2 - 2*theta3) +
15/32*L3^2*mL3*theta1dd*cos(2*theta2 + 2*theta3) - 3/4*L3^2*mL3*theta2dd*cos(2*theta2 + 2*theta3) -
3/4*L3^2*mL3*theta3dd*cos(2*theta2 + 2*theta3) + 1/32*L4^2*mL4*theta1dd*cos(- 2*theta2 - 2*theta3) +
1/32*L4^2*mL4*theta1dd*cos(2*theta3 - 2*theta2) - 1/32*L4^2*mL4*theta1dd*cos(2*theta2 - 2*theta3) +
15/32*L4^2*mL4*theta1dd*cos(2*theta2 + 2*theta3) - 5/64*L4^2*mL4*theta4dd*cos(- theta2 - theta3) - 1/64*L4^2*mL4*theta4dd*cos(- theta2
- 3*theta3) - 1/128*L4^2*mL4*theta4dd*cos(3*theta3 - theta2) - 3/128*L4^2*mL4*theta4dd*cos(- 3*theta2 - theta3) +
1/32*L4^2*mL4*theta4dd*cos(3*theta2 - theta3) - 1/128*L4^2*mL4*theta4dd*cos(- 3*theta2 - 3*theta3) -
15/128*L4^2*mL4*theta4dd*cos(3*theta2 + 3*theta3) + 3/128*L4^2*mL4*theta4d^2*cos(theta1 - 2*theta2) -
9/128*L4^2*mL4*theta4d^2*cos(theta1 + 2*theta2) + 3/128*L4^2*mL4*theta4d^2*cos(theta1 - 2*theta3) -
9/128*L4^2*mL4*theta4d^2*cos(theta1 + 2*theta3) + 1/4*Im4xx*Kr4*theta4dd*sin(2*theta1 + theta2 - theta3) +
1/4*Im4xx*Kr4*theta4dd*sin(2*theta1 - theta2 + theta3) + 1/4*Im4xy*Kr4*theta4dd*sin(2*theta1 + theta2 - theta3) -
1/4*Im4xy*Kr4*theta4dd*sin(2*theta1 - theta2 + theta3) + 5/4*Im4xz*Kr4*theta4dd*sin(theta1 - theta2 - theta3) -
1/4*Im4xx*Kr4^2*theta4d^2*cos(2*theta2) + 1/4*Im4xy*Kr4^2*theta4d^2*cos(2*theta1) - 1/2*L4*g*mL4*cos(theta1 - theta2 - theta3) -
1/16*Im4xx*Kr4^2*theta4d^2*cos(2*theta1 - 2*theta2 - 2*theta3) + 1/16*Im4xx*Kr4^2*theta4d^2*cos(2*theta1 - 2*theta2 + 2*theta3) -
1/16*Im4xx*Kr4^2*theta4d^2*cos(2*theta1 + 2*theta2 - 2*theta3) - 3/16*Im4xx*Kr4^2*theta4d^2*cos(2*theta1 + 2*theta2 + 2*theta3) +
1/8*Im4xy*Kr4^2*theta4d^2*cos(2*theta1 - 2*theta2 - 2*theta3) - 3/8*Im4xy*Kr4^2*theta4d^2*cos(2*theta1 + 2*theta2 + 2*theta3) +
1/4*L2^2*mL2*theta1dd*sin(2*theta1 - 2*theta2) - 1/4*L2^2*mL2*theta1dd*sin(2*theta1 + 2*theta2) + 1/4*L2^2*mL2*theta2dd*sin(2*theta1 -
2*theta2) - 1/4*L2^2*mL2*theta2dd*sin(2*theta1 + 2*theta2) - 1/4*L3^2*mL3*theta1dd*sin(2*theta2 + 2*theta3) -
1/4*L3^2*mL3*theta2dd*sin(2*theta2 + 2*theta3) - 1/8*L4^2*mL4*theta4dd*sin(3*theta2 - theta3) - 1/8*L4^2*mL4*theta4dd*sin(3*theta2 +
3*theta3) - 3/128*L4^2*mL4*theta4d^2*sin(theta1 - 2*theta2) - 9/128*L4^2*mL4*theta4d^2*sin(theta1 + 2*theta2) +
3/128*L4^2*mL4*theta4d^2*sin(theta1 - 2*theta3) + 9/128*L4^2*mL4*theta4d^2*sin(theta1 + 2*theta3) - 1/2*L3*g*mL3*sin(theta2) -
1/2*L4*g*mL4*sin(theta2) + 3/8*L4^2*mL4*theta4dd*cos(2*theta1 + theta2 + theta3) + 1/4*Im4xx*Kr4^2*theta4d^2*sin(2*theta1) +
1/4*Im4xx*Kr4^2*theta4d^2*sin(2*theta3) - 1/4*Im4yy*Kr4^2*theta4d^2*sin(2*theta1) + 1/2*L3*g*mL3*sin(theta1 - theta2 - theta3) +
1/4*L3*g*mL3*sin(theta1 - theta2 - 2*theta3) + 1/4*L4*g*mL4*sin(theta1 - theta2 - 2*theta3) + 1/32*d4^2*mL4*theta1dd*cos(- 2*theta2 -
2*theta3) + 1/32*d4^2*mL4*theta1dd*cos(2*theta3 - 2*theta2) - 1/32*d4^2*mL4*theta1dd*cos(2*theta2 - 2*theta3) +
15/32*d4^2*mL4*theta1dd*cos(2*theta2 + 2*theta3) + 1/32*d4^2*mm4*theta1dd*cos(- 2*theta2 - 2*theta3) +
1/32*d4^2*mm4*theta1dd*cos(2*theta3 - 2*theta2) - 1/32*d4^2*mm4*theta1dd*cos(2*theta2 - 2*theta3) +
15/32*d4^2*mm4*theta1dd*cos(2*theta2 + 2*theta3) - 3/2*d4^2*mm4*theta2dd*cos(2*theta2 + 2*theta3) -
5/4*d4^2*mm4*theta3dd*cos(2*theta2 + 2*theta3) - 1/8*IL4xx*theta2d*theta4d*cos(2*theta1 + theta2 - theta3) -
1/8*IL4xx*theta2d*theta4d*cos(2*theta1 - theta2 + theta3) + 1/8*IL4xz*theta1d*theta4d*cos(theta1 - theta2 - theta3) -
1/8*IL4xx*theta3d*theta4d*cos(2*theta1 + theta2 - theta3) - 1/8*IL4xx*theta3d*theta4d*cos(2*theta1 - theta2 + theta3) -
1/8*IL4xy*theta2d*theta4d*cos(2*theta1 + theta2 - theta3) + 1/8*IL4xy*theta2d*theta4d*cos(2*theta1 - theta2 + theta3) +
1/4*IL4xz*theta2d*theta4d*cos(theta1 - theta2 - theta3) - 1/8*IL4xy*theta3d*theta4d*cos(2*theta1 + theta2 - theta3) +
1/8*IL4xy*theta3d*theta4d*cos(2*theta1 - theta2 + theta3) + 1/4*IL4xz*theta3d*theta4d*cos(theta1 - theta2 - theta3) +
1/2*Im4xy*Kr4*theta4dd*cos(theta2 - theta3) + 1/8*Im4xy*Kr4^2*theta4d^2*sin(2*theta1 - 2*theta2 - 2*theta3) +
3/8*Im4xy*Kr4^2*theta4d^2*sin(2*theta1 + 2*theta2 + 2*theta3) - 1/8*Im4yy*Kr4^2*theta4d^2*sin(2*theta1 - 2*theta2 - 2*theta3) +
3/8*Im4yy*Kr4^2*theta4d^2*sin(2*theta1 + 2*theta2 + 2*theta3) - 1/2*d4*g*mL4*cos(theta1 - theta2 - theta3) - 1/2*d4*g*mm4*cos(theta1 -
theta2 - theta3) - 1/4*d4*g*mm4*cos(theta1 - theta2 - 2*theta3) + 1/4*L4^2*mL4*theta4dd*sin(2*theta1 + theta2 + theta3) -
1/2*d4*g*mL4*sin(theta2) - 1/8*IL4xx*theta2d*theta4d*sin(2*theta1 + theta2 - theta3) + 1/8*IL4xx*theta2d*theta4d*sin(2*theta1 - theta2 +
theta3) + 1/8*IL4xz*theta1d*theta4d*sin(theta1 - theta2 - theta3) - 1/8*IL4xx*theta3d*theta4d*sin(2*theta1 + theta2 - theta3) +
1/8*IL4xx*theta3d*theta4d*sin(2*theta1 - theta2 + theta3) + 1/8*IL4xy*theta2d*theta4d*sin(2*theta1 + theta2 - theta3) +
1/8*IL4xy*theta2d*theta4d*sin(2*theta1 - theta2 + theta3) + 1/4*IL4yz*theta1d*theta4d*sin(theta1 - theta2 - theta3) +

1/8*IL4xy*theta3d*theta4d*sin(2*theta1 + theta2 - theta3) + 1/8*IL4xy*theta3d*theta4d*sin(2*theta1 - theta2 + theta3) +
1/4*IL4yz*theta2d*theta4d*sin(theta1 - theta2 - theta3) + 1/4*IL4yz*theta3d*theta4d*sin(theta1 - theta2 - theta3) -
9/64*L4^2*mL4*theta4d^2*cos(3*theta1) - 5/32*L4^2*mL4*theta4d^2*cos(2*theta2) - 7/64*L4^2*mL4*theta4d^2*cos(2*theta3) -
1/64*L4^2*mL4*theta4d^2*cos(4*theta2) + 1/64*L4^2*mL4*theta4d^2*cos(4*theta3) + 1/2*Im4xy*Kr4*theta4dd*sin(theta2 - theta3) +
1/4*d4*g*mL4*sin(theta1 - theta2 - 2*theta3) + 1/4*d4*g*mm4*sin(theta1 - theta2 - 2*theta3) - 1/4*Im4xz*Kr4*theta4dd*cos(theta1 + theta2 +
theta3) - 1/2*Im4yz*Kr4*theta4dd*cos(theta1 + theta2 + theta3) - 1/8*L4^2*mL4*theta4d^2*cos(2*theta1 - 2*theta2 - 2*theta3) -
3/8*L4^2*mL4*theta4d^2*cos(2*theta1 + 2*theta2 + 2*theta3) + 9/512*L4^2*mL4*theta4d^2*cos(2*theta2 - 3*theta1 - 2*theta3) -
9/512*L4^2*mL4*theta4d^2*cos(2*theta2 - 3*theta1 + 2*theta3) + 25/512*L4^2*mL4*theta4d^2*cos(3*theta1 - 2*theta2 - 2*theta3) -
9/512*L4^2*mL4*theta4d^2*cos(3*theta1 - 2*theta2 + 2*theta3) - 7/32*L4^2*mL4*theta4d^2*cos(3*theta1 + 2*theta2 + 2*theta3) -
1/8*L4^2*mL4*theta4d^2*cos(2*theta1 - 2*theta2 - 4*theta3) - 1/4*L4^2*mL4*theta4d^2*cos(2*theta1 + 2*theta2 + 4*theta3) -
1/8*L4^2*mL4*theta4d^2*cos(2*theta1 - 4*theta2 - 2*theta3) - 1/4*L4^2*mL4*theta4d^2*cos(2*theta1 + 4*theta2 + 2*theta3) +
17/1024*L4^2*mL4*theta4d^2*cos(2*theta2 - 4*theta1 - 2*theta3) + 23/1024*L4^2*mL4*theta4d^2*cos(2*theta2 - 4*theta1 + 2*theta3) -
23/1024*L4^2*mL4*theta4d^2*cos(4*theta1 - 2*theta2 - 2*theta3) - 33/1024*L4^2*mL4*theta4d^2*cos(4*theta1 - 2*theta2 + 2*theta3) +
1/64*L4^2*mL4*theta4d^2*cos(4*theta1 + 2*theta2 - 2*theta3) + 1/32*L4^2*mL4*theta4d^2*cos(4*theta1 + 2*theta2 + 2*theta3) -
9/512*L4^2*mL4*theta4d^2*cos(2*theta2 - 3*theta1 + 4*theta3) + 21/512*L4^2*mL4*theta4d^2*cos(3*theta1 - 2*theta2 - 4*theta3) -
9/128*L4^2*mL4*theta4d^2*cos(3*theta1 + 2*theta2 + 4*theta3) + 3/128*L4^2*mL4*theta4d^2*cos(3*theta1 - 4*theta2 - 2*theta3) -
9/128*L4^2*mL4*theta4d^2*cos(3*theta1 + 4*theta2 + 2*theta3) - 3/16*L4^2*mL4*theta4d^2*cos(2*theta1 - 4*theta2 - 4*theta3) -
5/16*L4^2*mL4*theta4d^2*cos(2*theta1 + 4*theta2 + 4*theta3) + 3/256*L4^2*mL4*theta4d^2*cos(2*theta2 - 4*theta1 - 4*theta3) +
7/1024*L4^2*mL4*theta4d^2*cos(2*theta2 - 4*theta1 + 4*theta3) + 7/512*L4^2*mL4*theta4d^2*cos(2*theta3 - 4*theta2 - 4*theta1) -
11/2048*L4^2*mL4*theta4d^2*cos(4*theta2 - 4*theta1 - 2*theta3) + 9/1024*L4^2*mL4*theta4d^2*cos(4*theta2 - 4*theta1 + 2*theta3) -
7/1024*L4^2*mL4*theta4d^2*cos(4*theta1 - 2*theta2 - 4*theta3) - 3/256*L4^2*mL4*theta4d^2*cos(4*theta1 - 2*theta2 + 4*theta3) -
1/1024*L4^2*mL4*theta4d^2*cos(4*theta1 - 4*theta2 - 2*theta3) + 11/2048*L4^2*mL4*theta4d^2*cos(4*theta1 - 4*theta2 + 2*theta3) -
7/512*L4^2*mL4*theta4d^2*cos(4*theta1 + 4*theta2 - 2*theta3) + 5/128*L4^2*mL4*theta4d^2*cos(4*theta1 + 4*theta2 + 2*theta3) +
5/128*L4^2*mL4*theta4d^2*cos(3*theta1 - 4*theta2 - 4*theta3) - 11/128*L4^2*mL4*theta4d^2*cos(3*theta1 + 4*theta2 + 4*theta3) -
3/1024*L4^2*mL4*theta4d^2*cos(4*theta2 - 4*theta1 - 4*theta3) + 9/2048*L4^2*mL4*theta4d^2*cos(4*theta2 - 4*theta1 + 4*theta3) +
7/2048*L4^2*mL4*theta4d^2*cos(4*theta1 - 4*theta2 - 4*theta3) + 3/1024*L4^2*mL4*theta4d^2*cos(4*theta1 - 4*theta2 + 4*theta3) +
3/128*L4^2*mL4*theta4d^2*cos(4*theta1 + 4*theta2 + 4*theta3) + 1/2*L2*g*mL2*cos(theta1 - theta2) -
1/4*Im4xx*Kr4^2*theta4d^2*cos(2*theta1 + 2*theta2) - 1/4*Im4xy*Kr4^2*theta4d^2*cos(2*theta1 + 2*theta2) -
1/4*Im4xx*Kr4^2*theta4d^2*cos(2*theta2 + 2*theta3) + 1/4*Im4xy*Kr4^2*theta4d^2*cos(2*theta1 + 2*theta3) +
1/16*L4^2*mL4*theta4d^2*sin(2*theta1) + 9/64*L4^2*mL4*theta4d^2*sin(2*theta2) - 1/16*L4^2*mL4*theta4d^2*sin(4*theta1) +
5/32*L4^2*mL4*theta4d^2*sin(2*theta3) - 1/8*L4^2*mL4*theta4d^2*sin(4*theta2) - 1/8*L4^2*mL4*theta4d^2*sin(4*theta3) -
51/64*L4^2*mL4*theta4dd*cos(theta2 + theta3) + 3/4*Im4xz*Kr4*theta4dd*sin(theta1 + theta2 + theta3) -
3/16*L4^2*mL4*theta4d^2*sin(2*theta1 - 2*theta2 - 2*theta3) + 1/16*L4^2*mL4*theta4d^2*sin(2*theta1 - 2*theta2 + 2*theta3) +
1/16*L4^2*mL4*theta4d^2*sin(2*theta1 + 2*theta2 - 2*theta3) + 9/16*L4^2*mL4*theta4d^2*sin(2*theta1 + 2*theta2 + 2*theta3) -
9/512*L4^2*mL4*theta4d^2*sin(2*theta2 - 3*theta1 - 2*theta3) - 27/512*L4^2*mL4*theta4d^2*sin(2*theta2 - 3*theta1 + 2*theta3) -
35/512*L4^2*mL4*theta4d^2*sin(3*theta1 - 2*theta2 - 2*theta3) - 9/512*L4^2*mL4*theta4d^2*sin(3*theta1 - 2*theta2 + 2*theta3) -
7/64*L4^2*mL4*theta4d^2*sin(3*theta1 + 2*theta2 + 2*theta3) + 1/8*L4^2*mL4*theta4d^2*sin(2*theta1 - 2*theta2 - 4*theta3) -
1/4*L4^2*mL4*theta4d^2*sin(2*theta1 + 2*theta2 + 4*theta3) + 1/8*L4^2*mL4*theta4d^2*sin(2*theta1 - 4*theta2 - 2*theta3) -
1/4*L4^2*mL4*theta4d^2*sin(2*theta1 + 4*theta2 + 2*theta3) - 45/2048*L4^2*mL4*theta4d^2*sin(2*theta2 - 4*theta1 - 2*theta3) -
11/512*L4^2*mL4*theta4d^2*sin(2*theta2 - 4*theta1 + 2*theta3) - 11/512*L4^2*mL4*theta4d^2*sin(4*theta1 - 2*theta2 - 2*theta3) -
45/2048*L4^2*mL4*theta4d^2*sin(4*theta1 - 2*theta2 + 2*theta3) - 1/16*L4^2*mL4*theta4d^2*sin(4*theta1 + 2*theta2 + 2*theta3) -
3/512*L4^2*mL4*theta4d^2*sin(2*theta2 - 3*theta1 + 4*theta3) + 9/512*L4^2*mL4*theta4d^2*sin(3*theta1 - 2*theta2 - 4*theta3) +
9/128*L4^2*mL4*theta4d^2*sin(3*theta1 + 2*theta2 + 4*theta3) - 3/128*L4^2*mL4*theta4d^2*sin(3*theta1 - 4*theta2 - 2*theta3) -
9/128*L4^2*mL4*theta4d^2*sin(3*theta1 + 4*theta2 + 2*theta3) - 3/32*L4^2*mL4*theta4d^2*sin(2*theta1 - 4*theta2 - 4*theta3) +
5/32*L4^2*mL4*theta4d^2*sin(2*theta1 + 4*theta2 + 4*theta3) - 5/1024*L4^2*mL4*theta4d^2*sin(2*theta2 - 4*theta1 - 4*theta3) -
7/512*L4^2*mL4*theta4d^2*sin(2*theta2 - 4*theta1 + 4*theta3) + 1/256*L4^2*mL4*theta4d^2*sin(4*theta2 - 4*theta1 - 2*theta3) +
1/256*L4^2*mL4*theta4d^2*sin(4*theta2 - 4*theta1 + 2*theta3) - 3/512*L4^2*mL4*theta4d^2*sin(4*theta1 - 2*theta2 - 4*theta3) -
5/1024*L4^2*mL4*theta4d^2*sin(4*theta1 - 2*theta2 + 4*theta3) - 5/128*L4^2*mL4*theta4d^2*sin(4*theta1 + 2*theta2 + 4*theta3) +
1/256*L4^2*mL4*theta4d^2*sin(4*theta1 - 4*theta2 - 2*theta3) + 1/256*L4^2*mL4*theta4d^2*sin(4*theta1 - 4*theta2 + 2*theta3) -
3/1024*L4^2*mL4*theta4d^2*sin(4*theta2 - 3*theta1 + 4*theta3) - 43/1024*L4^2*mL4*theta4d^2*sin(3*theta1 - 4*theta2 - 4*theta3) -
11/128*L4^2*mL4*theta4d^2*sin(3*theta1 + 4*theta2 + 4*theta3) + 1/1024*L4^2*mL4*theta4d^2*sin(4*theta2 - 4*theta1 - 4*theta3) +
3/1024*L4^2*mL4*theta4d^2*sin(4*theta2 - 4*theta1 + 4*theta3) + 3/1024*L4^2*mL4*theta4d^2*sin(4*theta1 - 4*theta2 - 4*theta3) +
1/1024*L4^2*mL4*theta4d^2*sin(4*theta1 - 4*theta2 + 4*theta3) + 1/4*L3*g*mL3*sin(theta1 - theta2) + 1/2*L3*g*mL3*sin(theta2 + 2*theta3)
+ 1/4*L4*g*mL4*sin(theta1 - theta2) + 1/2*L4*g*mL4*sin(theta2 + 2*theta3) - 1/16*IL4xx*theta2d*theta4d*cos(theta2 - theta3) +
1/16*IL4xx*theta2d*theta4d*cos(theta3 - theta2) - 1/16*IL4xx*theta3d*theta4d*cos(theta2 - theta3) + 1/16*IL4xx*theta3d*theta4d*cos(theta3 -
theta2) - 1/8*IL4xy*theta2d*theta4d*cos(theta2 - theta3) + 1/8*IL4xy*theta2d*theta4d*cos(theta3 - theta2) -
1/8*IL4xy*theta3d*theta4d*cos(theta2 - theta3) + 1/8*IL4xy*theta3d*theta4d*cos(theta3 - theta2) - 1/2*Im3xx*Kr3*theta2dd*cos(2*theta1) -
1/2*Im3xx*Kr3*theta3dd*cos(2*theta1) + 1/2*Im3yy*Kr3*theta2dd*cos(2*theta1) + 1/2*Im3yy*Kr3*theta3dd*cos(2*theta1) +
1/2*L4*g*mL4*cos(theta1 + theta2 + theta3) - 1/2*a2*g*mL3*cos(theta1 - theta2) - 1/2*a2*g*mL4*cos(theta1 - theta2) -
1/2*a2*g*mm3*cos(theta1 - theta2) - 1/2*a2*g*mm4*cos(theta1 - theta2) + 1/4*Im4xx*Kr4^2*theta4d^2*sin(2*theta1 + 2*theta3) +
1/4*Im4xy*Kr4^2*theta4d^2*sin(2*theta1 + 2*theta2) + 1/4*Im4xy*Kr4^2*theta4d^2*sin(2*theta1 + 2*theta3) -
1/32*Im4xy*Kr4^2*theta4d^2*sin(2*theta3 - 2*theta2) - 1/32*Im4xy*Kr4^2*theta4d^2*sin(2*theta2 - 2*theta3) -
1/16*Im4yy*Kr4^2*theta4d^2*sin(2*theta3 - 2*theta2) - 1/16*Im4yy*Kr4^2*theta4d^2*sin(2*theta2 - 2*theta3) -
1/2*Im4yy*Kr4^2*theta4d^2*sin(2*theta2 + 2*theta3) + Im4zz*Kr4^2*theta4d^2*sin(2*theta2 + 2*theta3) + 1/4*d4*g*mm4*cos(theta1 - theta2)
+ 1/4*Im4xx*Kr4*theta4dd*cos(2*theta1 - theta2 - theta3) + 1/4*Im4xy*Kr4*theta4dd*cos(2*theta1 - theta2 - theta3) -
1/2*Im4yy*Kr4*theta4dd*cos(2*theta1 - theta2 - theta3) - 1/8*L4^2*mL4*theta4dd*sin(theta2 + theta3) + Im3xy*Kr3*theta2dd*sin(2*theta1) +
Im3xy*Kr3*theta3dd*sin(2*theta1) - 1/2*L3*g*mL3*sin(theta1 + theta2 + theta3) + 3/8*IL4xz*theta1d*theta4d*cos(theta1 + theta2 + theta3) -
3/4*IL4xz*theta2d*theta4d*cos(theta1 + theta2 + theta3) - 3/4*IL4xz*theta3d*theta4d*cos(theta1 + theta2 + theta3) + 1/4*d4*g*mL4*sin(theta1 -
theta2) + 1/2*d4*g*mL4*sin(theta2 + 2*theta3) + 1/4*d4*g*mm4*sin(theta1 - theta2) - 1/8*L4^2*mL4*theta4d^2*cos(2*theta1 + 2*theta2) -
1/8*L4^2*mL4*theta4d^2*cos(2*theta1 + 2*theta3) + 9/512*L4^2*mL4*theta4d^2*cos(2*theta2 - 3*theta1) -

21/512*L4^2*mL4*theta4d^2*cos(3*theta1 - 2*theta2) - 15/128*L4^2*mL4*theta4d^2*cos(3*theta1 + 2*theta2) - 9/512*L4^2*mL4*theta4d^2*cos(- 3*theta1 - 2*theta3) - 9/512*L4^2*mL4*theta4d^2*cos(2*theta3 - 3*theta1) - 3/512*L4^2*mL4*theta4d^2*cos(3*theta1 - 2*theta3) - 51/512*L4^2*mL4*theta4d^2*cos(3*theta1 + 2*theta3) + 11/2048*L4^2*mL4*theta4d^2*cos(- 2*theta2 - 2*theta3) - 21/2048*L4^2*mL4*theta4d^2*cos(2*theta3 - 2*theta2) + 21/2048*L4^2*mL4*theta4d^2*cos(2*theta2 - 2*theta3) - 1099/2048*L4^2*mL4*theta4d^2*cos(2*theta2 + 2*theta3) + 21/1024*L4^2*mL4*theta4d^2*cos(2*theta2 - 4*theta1) - 37/1024*L4^2*mL4*theta4d^2*cos(4*theta1 - 2*theta2) + 3/64*L4^2*mL4*theta4d^2*cos(4*theta1 + 2*theta2) - 25/2048*L4^2*mL4*theta4d^2*cos(- 4*theta1 - 2*theta3) + 3/128*L4^2*mL4*theta4d^2*cos(2*theta3 - 4*theta1) - 1/64*L4^2*mL4*theta4d^2*cos(4*theta1 - 2*theta3) - 23/2048*L4^2*mL4*theta4d^2*cos(4*theta1 + 2*theta3) + 9/512*L4^2*mL4*theta4d^2*cos(4*theta3 - 3*theta1) - 9/512*L4^2*mL4*theta4d^2*cos(3*theta1 - 4*theta3) + 9/2048*L4^2*mL4*theta4d^2*cos(- 2*theta2 - 4*theta3) + 1/1024*L4^2*mL4*theta4d^2*cos(4*theta3 - 2*theta2) - 1/1024*L4^2*mL4*theta4d^2*cos(2*theta2 - 4*theta3) - 777/2048*L4^2*mL4*theta4d^2*cos(2*theta2 + 4*theta3) - 7/512*L4^2*mL4*theta4d^2*cos(- 4*theta1 - 4*theta2) - 5/1024*L4^2*mL4*theta4d^2*cos(4*theta2 - 4*theta1) + 5/1024*L4^2*mL4*theta4d^2*cos(4*theta1 - 4*theta2) + 15/512*L4^2*mL4*theta4d^2*cos(4*theta1 + 4*theta2) - 1/512*L4^2*mL4*theta4d^2*cos(- 4*theta2 - 2*theta3) - 181/512*L4^2*mL4*theta4d^2*cos(2*theta3 - 4*theta2) + 181/512*L4^2*mL4*theta4d^2*cos(4*theta2 - 2*theta3) - 215/512*L4^2*mL4*theta4d^2*cos(4*theta2 + 2*theta3) - 5/1024*L4^2*mL4*theta4d^2*cos(- 4*theta1 - 4*theta3) - 27/2048*L4^2*mL4*theta4d^2*cos(4*theta3 - 4*theta1) + 27/2048*L4^2*mL4*theta4d^2*cos(4*theta1 - 4*theta3) - 11/1024*L4^2*mL4*theta4d^2*cos(4*theta1 + 4*theta3) - 1/2048*L4^2*mL4*theta4d^2*cos(- 4*theta2 - 4*theta3) - 17/2048*L4^2*mL4*theta4d^2*cos(4*theta3 - 4*theta2) + 17/2048*L4^2*mL4*theta4d^2*cos(4*theta2 - 4*theta3) - 1087/2048*L4^2*mL4*theta4d^2*cos(4*theta2 + 4*theta3) + 1/4*Im4xx*Kr4*theta4dd*sin(2*theta1 - theta2 - theta3) - 3/4*Im4xy*Kr4*theta4dd*sin(2*theta1 - theta2 - theta3) + 1/2*d4*g*mL4*cos(theta1 + theta2 + theta3) + 1/2*d4*g*mm4*cos(theta1 + theta2 + theta3) - 3/8*IL4xz*theta1d*theta4d*sin(theta1 + theta2 + theta3) + 3/4*IL4yz*theta1d*theta4d*sin(theta1 + theta2 + theta3) - 3/4*IL4yz*theta2d*theta4d*sin(theta1 + theta2 + theta3) - 3/4*IL4yz*theta3d*theta4d*sin(theta1 + theta2 + theta3) + 1/8*L4^2*mL4*theta4d^2*sin(2*theta1 + 2*theta2) + 1/8*L4^2*mL4*theta4d^2*sin(2*theta1 + 2*theta3) - 33/512*L4^2*mL4*theta4d^2*sin(2*theta2 - 3*theta1) - 21/512*L4^2*mL4*theta4d^2*sin(3*theta1 - 2*theta2) - 15/128*L4^2*mL4*theta4d^2*sin(3*theta1 + 2*theta2) + 1/32*L4^2*mL4*theta4d^2*sin(2*theta1 - 4*theta2) - 3/32*L4^2*mL4*theta4d^2*sin(2*theta1 + 4*theta2) + 9/512*L4^2*mL4*theta4d^2*sin(- 3*theta1 - 2*theta3) + 63/1024*L4^2*mL4*theta4d^2*sin(2*theta3 - 3*theta1) + 39/1024*L4^2*mL4*theta4d^2*sin(3*theta1 - 2*theta3) + 69/512*L4^2*mL4*theta4d^2*sin(3*theta1 + 2*theta3) + 17/2048*L4^2*mL4*theta4d^2*sin(- 2*theta2 - 2*theta3) + 3/128*L4^2*mL4*theta4d^2*sin(2*theta3 - 2*theta2) + 3/128*L4^2*mL4*theta4d^2*sin(2*theta2 - 2*theta3) + 1681/2048*L4^2*mL4*theta4d^2*sin(2*theta2 + 2*theta3) - 29/1024*L4^2*mL4*theta4d^2*sin(2*theta2 - 4*theta1) - 37/1024*L4^2*mL4*theta4d^2*sin(4*theta1 - 2*theta2) - 3/128*L4^2*mL4*theta4d^2*sin(4*theta1 + 2*theta2) + 1/32*L4^2*mL4*theta4d^2*sin(2*theta1 - 4*theta3) - 3/32*L4^2*mL4*theta4d^2*sin(2*theta1 + 4*theta3) + 3/1024*L4^2*mL4*theta4d^2*sin(4*theta2 - 3*theta1) + 3/1024*L4^2*mL4*theta4d^2*sin(3*theta1 - 4*theta2) + 1/512*L4^2*mL4*theta4d^2*sin(- 4*theta1 - 2*theta3) + 7/2048*L4^2*mL4*theta4d^2*sin(2*theta3 - 4*theta1) - 25/2048*L4^2*mL4*theta4d^2*sin(4*theta1 - 2*theta3) - 23/512*L4^2*mL4*theta4d^2*sin(4*theta1 + 2*theta3) + 9/1024*L4^2*mL4*theta4d^2*sin(4*theta3 - 3*theta1) + 9/1024*L4^2*mL4*theta4d^2*sin(3*theta1 - 4*theta3) + 1/1024*L4^2*mL4*theta4d^2*sin(- 2*theta2 - 4*theta3) + 17/2048*L4^2*mL4*theta4d^2*sin(4*theta3 - 2*theta2) + 17/2048*L4^2*mL4*theta4d^2*sin(2*theta2 - 4*theta3) - 335/1024*L4^2*mL4*theta4d^2*sin(2*theta2 + 4*theta3) + 15/2048*L4^2*mL4*theta4d^2*sin(4*theta2 - 4*theta1) + 15/2048*L4^2*mL4*theta4d^2*sin(4*theta1 - 4*theta2) + 3/2048*L4^2*mL4*theta4d^2*sin(- 4*theta2 - 2*theta3) - 327/2048*L4^2*mL4*theta4d^2*sin(2*theta3 - 4*theta2) - 327/2048*L4^2*mL4*theta4d^2*sin(4*theta2 - 2*theta3) - 765/2048*L4^2*mL4*theta4d^2*sin(4*theta2 + 2*theta3) + 9/2048*L4^2*mL4*theta4d^2*sin(4*theta3 - 4*theta1) + 9/2048*L4^2*mL4*theta4d^2*sin(4*theta1 - 4*theta3) + 159/2048*L4^2*mL4*theta4d^2*sin(4*theta3 - 4*theta2) + 159/2048*L4^2*mL4*theta4d^2*sin(4*theta2 - 4*theta3) + 1/4*L4^2*mL4*theta4d^2*sin(4*theta2 + 4*theta3) + 3/4*L4^2*mL4*theta4dd*cos(theta1 - theta2 - theta3) - 1/8*L4^2*mL4*theta4dd*cos(2*theta1 + theta2 - theta3) + 1/8*L4^2*mL4*theta4dd*cos(2*theta1 - theta2 + theta3) + 1/16*L4^2*mL4*theta4dd*cos(2*theta1 + theta2 + 3*theta3) - 1/16*L4^2*mL4*theta4dd*cos(2*theta1 + 3*theta2 + theta3) + 1/4*L4^2*mL4*theta4dd*cos(theta1 - 3*theta2 - 3*theta3) + 1/4*L4^2*mL4*theta4dd*cos(theta1 + 3*theta2 + 3*theta3) + 3/8*L4^2*mL4*theta4dd*sin(2*theta1 + theta2 - theta3) + 3/8*L4^2*mL4*theta4dd*sin(2*theta1 - theta2 + theta3) - 1/8*L4^2*mL4*theta4dd*sin(2*theta1 + theta2 + 3*theta3) - 1/8*L4^2*mL4*theta4dd*sin(2*theta1 + 3*theta2 + theta3) + 1/2*d4^2*mm4*theta2dd*cos(theta1) + 1/2*d4^2*mm4*theta3dd*cos(theta1) - 1/4*d4^2*mm4*theta2dd*cos(theta1 - 2*theta2 - 2*theta3) - 1/4*d4^2*mm4*theta2dd*cos(theta1 + 2*theta2 + 2*theta3) - 1/4*d4^2*mm4*theta3dd*cos(theta1 - 2*theta2 - 2*theta3) - 1/4*d4^2*mm4*theta3dd*cos(theta1 + 2*theta2 + 2*theta3) - 1/4*Im4xx*Kr4*theta4dd*cos(2*theta1 + theta2 + theta3) + 1/4*Im4xy*Kr4*theta4dd*cos(2*theta1 + theta2 + theta3) - 1/4*Im4xz*Kr4*theta4dd*cos(theta1 + theta2 - theta3) - 1/4*Im4xz*Kr4*theta4dd*cos(theta1 - theta2 + theta3) + 1/2*Im4yy*Kr4*theta4dd*cos(2*theta1 + theta2 + theta3) + L4*d4*mL4*theta1dd + 4*L4*d4*mL4*theta2dd + 4*L4*d4*m... Output truncated. Text exceeds maximum line length of 25,000 characters for Command Window display.

$T_4 =$

IL1zz*theta1dd + IL2zz*theta1dd + IL3zz*theta1dd + IL4zz*theta1dd + Im2zz*theta1dd + Im3zz*theta1dd + Im4zz*theta1dd - IL2xz*theta1dd*sin(theta1) - IL2xz*theta2dd*sin(theta1) - 2*IL3xz*theta1dd*sin(theta1) - IL3xz*theta2dd*sin(theta1) - IL3xz*theta3dd*sin(theta1) - 2*IL4xz*theta1dd*sin(theta1) - IL4xz*theta2dd*sin(theta1) - IL4xz*theta3dd*sin(theta1) - Im3xz*theta1dd*sin(theta1) - Im3xz*theta2dd*sin(theta1) - 2*Im4xz*theta1dd*sin(theta1) - Im4xz*theta2dd*sin(theta1) - Im4xz*theta3dd*sin(theta1) + IL2yy*theta2dd*cos(theta1)^2 + 2*IL3yy*theta2dd*cos(theta1)^2 + 2*IL3yy*theta3dd*cos(theta1)^2 + 2*IL4yy*theta2dd*cos(theta1)^2 + 2*IL4yy*theta3dd*cos(theta1)^2 + Im3yy*theta2dd*cos(theta1)^2 + 2*Im4yy*theta2dd*cos(theta1)^2 + 2*Im4yy*theta3dd*cos(theta1)^2 + IL2xx*theta2dd*sin(theta1)^2 + 2*IL3xx*theta2dd*sin(theta1)^2 + 2*IL3xx*theta3dd*sin(theta1)^2 +

2*IL4xx*theta2dd*sin(theta1)^2 + 2*IL4xx*theta3dd*sin(theta1)^2 + Im3xx*theta2dd*sin(theta1)^2 + 2*Im4xx*theta2dd*sin(theta1)^2 +
2*Im4xx*theta3dd*sin(theta1)^2 + Im1zz*Kr1^2*theta1dd + IL2yz*theta1dd*cos(theta1) + IL2yz*theta2dd*cos(theta1) +
2*IL3yz*theta1dd*cos(theta1) + IL3yz*theta2dd*cos(theta1) + IL3yz*theta3dd*cos(theta1) + 2*IL4yz*theta1dd*cos(theta1) +
IL4yz*theta2dd*cos(theta1) + IL4yz*theta3dd*cos(theta1) + Im3yz*theta1dd*cos(theta1) + Im3yz*theta2dd*cos(theta1) +
2*Im4yz*theta1dd*cos(theta1) + Im4yz*theta2dd*cos(theta1) + Im4yz*theta3dd*cos(theta1) + Im2xx*Kr2^2*theta2dd*sin(theta1)^2 +
Im3xx*Kr3^2*theta3dd*sin(theta1)^2 + L2^2*mL2*theta2dd*cos(theta2)^2 + a2^2*mL3*theta2dd*cos(theta2)^2 +
a2^2*mL4*theta2dd*cos(theta2)^2 + a2^2*mm3*theta2dd*cos(theta2)^2 + a2^2*mm4*theta2dd*cos(theta2)^2 +
Im2yz*Kr2*theta1dd*cos(theta1) + Im2yz*Kr2*theta2dd*cos(theta1) + Im3yz*Kr3*theta1dd*cos(theta1) + Im3yz*Kr3*theta3dd*cos(theta1) -
Im2xz*Kr2*theta1dd*sin(theta1) - Im2xz*Kr2*theta2dd*sin(theta1) - Im3xz*Kr3*theta1dd*sin(theta1) - Im3xz*Kr3*theta3dd*sin(theta1) +
IL4zz*theta4dd*cos(theta2)^2*cos(theta3)^2 + Im3yy*Kr3*theta2dd*cos(theta1)^2 + Im3yy*Kr3*theta3dd*cos(theta1)^2 +
IL4zz*theta4dd*sin(theta2)^2*sin(theta3)^2 + Im3xx*Kr3*theta2dd*sin(theta1)^2 + Im3xx*Kr3*theta3dd*sin(theta1)^2 -
IL4zz*theta1dd*cos(theta2)*cos(theta3) - IL4zz*theta4dd*cos(theta2)*cos(theta3) + 2*IL2xy*theta2dd*cos(theta1)*sin(theta1) +
4*IL3xy*theta2dd*cos(theta1)*sin(theta1) + 4*IL3xy*theta3dd*cos(theta1)*sin(theta1) + 4*IL4xy*theta2dd*cos(theta1)*sin(theta1) +
4*IL4xy*theta3dd*cos(theta1)*sin(theta1) + 2*Im3xy*theta2dd*cos(theta1)*sin(theta1) + 4*Im4xy*theta2dd*cos(theta1)*sin(theta1) +
4*Im4xy*theta3dd*cos(theta1)*sin(theta1) + IL4zz*theta1dd*sin(theta2)*sin(theta3) + IL4zz*theta4dd*sin(theta2)*sin(theta3) +
Im2yy*Kr2^2*theta2dd*cos(theta1)^2 + Im3yy*Kr3^2*theta3dd*cos(theta1)^2 + IL4xy*theta2dd*cos(theta1)^2*cos(theta2)*cos(theta3) +
IL4xy*theta3dd*cos(theta1)^2*cos(theta2)*cos(theta3) + 2*IL4xy*theta4dd*cos(theta1)^2*cos(theta2)*cos(theta3) +
IL4xy*theta2dd*cos(theta1)^2*cos(theta3)*sin(theta2) + IL4xy*theta3dd*cos(theta1)^2*cos(theta3)*sin(theta2) +
2*IL4xy*theta4dd*cos(theta1)^2*cos(theta3)*sin(theta2) - Im4zz*Kr4*theta1dd*cos(theta2)*cos(theta3) -
Im4zz*Kr4*theta4dd*cos(theta2)*cos(theta3) - IL4xx*theta4d^2*cos(theta1)^2*cos(theta2)^2*cos(theta3)^2 -
IL4xy*theta2dd*cos(theta2)*sin(theta1)^2*sin(theta3) - IL4xy*theta2dd*cos(theta3)*sin(theta1)^2*sin(theta2) -
IL4xy*theta3dd*cos(theta2)*sin(theta1)^2*sin(theta3) - IL4xy*theta3dd*cos(theta3)*sin(theta1)^2*sin(theta2) -
2*IL4xy*theta4dd*cos(theta2)*sin(theta1)^2*sin(theta3) - 2*IL4xy*theta4dd*cos(theta3)*sin(theta1)^2*sin(theta2) +
2*Im3xy*Kr3*theta2dd*cos(theta1)*sin(theta1) + 2*Im3xy*Kr3*theta3dd*cos(theta1)*sin(theta1) +
IL4xx*theta4d^2*cos(theta1)^2*cos(theta3)^2*sin(theta2)^2 + IL4xy*theta4d^2*cos(theta1)^2*cos(theta3)^2*sin(theta2)^2 +
Im4zz*Kr4^2*theta4dd*cos(theta2)^2*cos(theta3)^2 + Im4zz*Kr4*theta1dd*sin(theta2)*sin(theta3) +
Im4zz*Kr4*theta4dd*sin(theta2)*sin(theta3) - IL4xy*theta4d^2*cos(theta3)^2*sin(theta1)^2*sin(theta2)^2 + L2*g*mL2*sin(theta1)*sin(theta2) -
IL4zz*theta1d*theta4d*cos(theta2)*sin(theta3) - IL4zz*theta1d*theta4d*cos(theta3)*sin(theta2) +
Im4zz*Kr4^2*theta4dd*sin(theta2)^2*sin(theta3)^2 + L2^2*mL2*theta1dd*cos(theta1)^2*cos(theta2)^2 +
2*L4^2*mL4*theta2dd*cos(theta2)^2*cos(theta3)^2 + 2*L4^2*mL4*theta3dd*cos(theta2)^2*cos(theta3)^2 +
2*IL4xz*theta4dd*cos(theta1)*cos(theta2)^2*cos(theta3)^2 - a2*g*mL3*sin(theta1)*sin(theta2) - a2*g*mL4*sin(theta1)*sin(theta2) -
a2*g*mm3*sin(theta1)*sin(theta2) - a2*g*mm4*sin(theta1)*sin(theta2) + L2^2*mL2*theta1dd*cos(theta2)^2*sin(theta1)^2 +
L2^2*mL2*theta2dd*cos(theta1)^2*sin(theta2)^2 + a2^2*mL3*theta1dd*cos(theta1)^2*cos(theta2)^2 +
a2^2*mL4*theta1dd*cos(theta1)^2*cos(theta2)^2 + a2^2*mm3*theta1dd*cos(theta1)^2*cos(theta2)^2 +
a2^2*mm4*theta1dd*cos(theta1)^2*cos(theta2)^2 + 2*d4^2*mL4*theta2dd*cos(theta2)^2*cos(theta3)^2 +
2*d4^2*mL4*theta3dd*cos(theta2)^2*cos(theta3)^2 + 2*IL4zz*theta4d^2*cos(theta2)*cos(theta3)^2*sin(theta2) +
2*IL4zz*theta4d^2*cos(theta2)^2*cos(theta3)*sin(theta3) + L2^2*mL2*theta2dd*sin(theta1)^2*sin(theta2)^2 +
2*L4^2*mL4*theta2dd*sin(theta2)^2*sin(theta3)^2 + 2*L4^2*mL4*theta3dd*sin(theta2)^2*sin(theta3)^2 +
a2^2*mL3*theta1dd*cos(theta2)^2*sin(theta1)^2 + a2^2*mL3*theta2dd*cos(theta1)^2*sin(theta2)^2 +
a2^2*mL4*theta1dd*cos(theta2)^2*sin(theta1)^2 + a2^2*mL4*theta2dd*cos(theta1)^2*sin(theta2)^2 +
a2^2*mm3*theta1dd*cos(theta2)^2*sin(theta1)^2 + a2^2*mm3*theta2dd*cos(theta1)^2*sin(theta2)^2 +
a2^2*mm4*theta1dd*cos(theta2)^2*sin(theta1)^2 + a2^2*mm4*theta2dd*cos(theta1)^2*sin(theta2)^2 +
d4^2*mm4*theta2dd*cos(theta2)^2*sin(theta3)^2 + d4^2*mm4*theta2dd*cos(theta3)^2*sin(theta2)^2 -
2*IL4zz*theta4d^2*cos(theta2)*sin(theta2)*sin(theta3)^2 - 2*IL4zz*theta4d^2*cos(theta3)*sin(theta2)^2*sin(theta3) +
2*Im2xy*Kr2^2*theta2dd*cos(theta1)*sin(theta1) + 2*Im3xy*Kr3^2*theta3dd*cos(theta1)*sin(theta1) +
a2^2*mL3*theta2dd*sin(theta1)^2*sin(theta2)^2 + a2^2*mL4*theta2dd*sin(theta1)^2*sin(theta2)^2 +
a2^2*mm3*theta2dd*sin(theta1)^2*sin(theta2)^2 + a2^2*mm4*theta2dd*sin(theta1)^2*sin(theta2)^2 +
2*d4^2*mL4*theta2dd*sin(theta2)^2*sin(theta3)^2 + 2*d4^2*mL4*theta3dd*sin(theta2)^2*sin(theta3)^2 -
IL4xz*theta1dd*cos(theta1)*cos(theta2)*cos(theta3) - IL4yz*theta2dd*cos(theta1)*cos(theta2)*cos(theta3) -
IL4xz*theta4dd*cos(theta1)*cos(theta2)*cos(theta3) - IL4yz*theta3dd*cos(theta1)*cos(theta2)*cos(theta3) -
2*IL4yz*theta4dd*cos(theta1)*cos(theta2)*cos(theta3) - L3*g*mL3*sin(theta2)*sin(theta3)^2 - L4*g*mL4*sin(theta2)*sin(theta3)^2 -
IL4xz*theta1dd*cos(theta1)*cos(theta3)*sin(theta2) + IL4xz*theta2dd*cos(theta2)*cos(theta3)*sin(theta1) +
IL4xz*theta3dd*cos(theta2)*cos(theta3)*sin(theta1) - IL4xz*theta4dd*cos(theta1)*cos(theta3)*sin(theta2) +
2*IL4xz*theta4dd*cos(theta2)*cos(theta3)*sin(theta1) - IL4xz*theta4d^2*cos(theta1)*cos(theta2)^2*cos(theta3)^2 +
IL4xx*theta4dd*cos(theta1)^2*cos(theta2)^2*cos(theta3)^2 - IL4yz*theta1dd*cos(theta2)*sin(theta1)*sin(theta3) -
IL4yz*theta1dd*cos(theta3)*sin(theta1)*sin(theta2) + IL4yz*theta2dd*cos(theta1)*sin(theta2)*sin(theta3) +
IL4yz*theta3dd*cos(theta1)*sin(theta2)*sin(theta3) + 2*IL4yz*theta4dd*cos(theta1)*sin(theta2)*sin(theta3) -
IL4yz*theta4dd*cos(theta2)*sin(theta1)*sin(theta3) - IL4yz*theta4dd*cos(theta3)*sin(theta1)*sin(theta2) - d4*g*mL4*sin(theta2)*sin(theta3)^2 +
2*IL4xz*theta4d^2*cos(theta1)*cos(theta3)^2*sin(theta2)^2 + IL4xz*theta4d^2*cos(theta2)^2*cos(theta3)^2*sin(theta1) -
2*IL4yz*theta4d^2*cos(theta2)^2*cos(theta3)^2*sin(theta1) + IL4xx*theta4dd*cos(theta1)^2*cos(theta3)^2*sin(theta2)^2 -
IL4xz*theta2dd*sin(theta1)*sin(theta2)*sin(theta3) - IL4xz*theta3dd*sin(theta1)*sin(theta2)*sin(theta3) -
2*IL4xz*theta4dd*sin(theta1)*sin(theta2)*sin(theta3) - IL4xz*theta4d^2*cos(theta1)*sin(theta2)^2*sin(theta3)^2 +
2*IL4yz*theta4d^2*cos(theta2)^2*sin(theta1)*sin(theta3)^2 + 2*IL4yz*theta4d^2*cos(theta3)^2*sin(theta1)*sin(theta2)^2 +
IL4yy*theta4dd*cos(theta2)^2*sin(theta1)^2*sin(theta3)^2 + IL4yy*theta4dd*cos(theta3)^2*sin(theta1)^2*sin(theta2)^2 -
2*IL4yz*theta4d^2*sin(theta1)*sin(theta2)^2*sin(theta3)^2 - Im4xz*Kr4*theta1dd*cos(theta1)*cos(theta3)*sin(theta2) +
Im4xz*Kr4*theta2dd*cos(theta2)*cos(theta3)*sin(theta1) + Im4xz*Kr4*theta3dd*cos(theta2)*cos(theta3)*sin(theta1) -
Im4xz*Kr4*theta4dd*cos(theta1)*cos(theta3)*sin(theta2) + 2*Im4xz*Kr4*theta4dd*cos(theta2)*cos(theta3)*sin(theta1) -
2*IL4yz*theta4dd*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3)^2 - 2*IL4yz*theta4dd*cos(theta3)*sin(theta1)*sin(theta2)^2*sin(theta3) +
2*Im4xz*Kr4^2*theta4dd*cos(theta1)*cos(theta2)^2*cos(theta3)^2 - Im4yz*Kr4*theta1dd*cos(theta2)*sin(theta1)*sin(theta3) -
Im4yz*Kr4*theta1dd*cos(theta3)*sin(theta1)*sin(theta2) + Im4yz*Kr4*theta2dd*cos(theta1)*sin(theta2)*sin(theta3) +

Im4yz*Kr4*theta3dd*cos(theta1)*sin(theta2)*sin(theta3) + 2*Im4yz*Kr4*theta4dd*cos(theta1)*sin(theta2)*sin(theta3) -
Im4yz*Kr4*theta4dd*cos(theta2)*sin(theta1)*sin(theta3) - Im4yz*Kr4*theta4dd*cos(theta3)*sin(theta1)*sin(theta2) +
3*a2*d4*mL4*theta2dd*cos(theta2)^2*cos(theta3) + a2*d4*mL4*theta3dd*cos(theta2)^2*cos(theta3) -
L3*g*mL3*cos(theta1)*cos(theta2)*sin(theta3) - L3*g*mL3*cos(theta1)*cos(theta3)*sin(theta2) +
L3*g*mL3*cos(theta2)*cos(theta3)*sin(theta3) + L4*g*mL4*cos(theta2)*cos(theta3)*sin(theta3) +
1/2*IL4xz*theta1d*theta4d*cos(theta1)*cos(theta2)*cos(theta3) - 1/2*IL4xz*theta2d*theta4d*cos(theta1)*cos(theta2)*cos(theta3) -
1/2*IL4xz*theta3d*theta4d*cos(theta1)*cos(theta2)*cos(theta3) + 2*Im4zz*Kr4^2*theta4d^2*cos(theta2)*cos(theta3)^2*sin(theta2) +
2*Im4zz*Kr4^2*theta4d^2*cos(theta2)^2*cos(theta3)*sin(theta3) - Im4xz*Kr4*theta2dd*sin(theta1)*sin(theta2)*sin(theta3) -
Im4xz*Kr4*theta3dd*sin(theta1)*sin(theta2)*sin(theta3) - 2*Im4xz*Kr4*theta4dd*sin(theta1)*sin(theta2)*sin(theta3) +
2*a2*d4*mm4*theta2dd*cos(theta2)^2*sin(theta3) - L4*g*mL4*cos(theta2)*sin(theta1)*sin(theta3) -
L4*g*mL4*cos(theta3)*sin(theta1)*sin(theta2) - 1/2*IL4xz*theta1d*theta4d*cos(theta1)*cos(theta2)*sin(theta3) -
1/2*IL4xz*theta1d*theta4d*cos(theta1)*cos(theta3)*sin(theta2) - 1/2*IL4xz*theta1d*theta4d*cos(theta2)*cos(theta3)*sin(theta1) +
1/2*IL4yz*theta1d*theta4d*cos(theta1)*cos(theta2)*sin(theta3) + 1/2*IL4yz*theta1d*theta4d*cos(theta1)*cos(theta3)*sin(theta2) +
IL4yz*theta1d*theta4d*cos(theta2)*cos(theta3)*sin(theta1) - IL4yz*theta2d*theta4d*cos(theta1)*cos(theta2)*sin(theta3) -
IL4yz*theta2d*theta4d*cos(theta1)*cos(theta3)*sin(theta2) - 1/2*IL4yz*theta2d*theta4d*cos(theta2)*cos(theta3)*sin(theta1) -
IL4yz*theta3d*theta4d*cos(theta1)*cos(theta2)*sin(theta3) - IL4yz*theta3d*theta4d*cos(theta1)*cos(theta3)*sin(theta2) -
1/2*IL4yz*theta3d*theta4d*cos(theta2)*cos(theta3)*sin(theta1) - 2*Im4zz*Kr4^2*theta4d^2*cos(theta2)*sin(theta2)*sin(theta3)^2 -
2*Im4zz*Kr4^2*theta4d^2*cos(theta3)*sin(theta2)^2*sin(theta3) + d4*g*mL4*cos(theta2)*cos(theta3)*sin(theta3) +
L4^2*mL4*theta2dd*cos(theta1)*cos(theta2)^3*cos(theta3)^3 + L4^2*mL4*theta3dd*cos(theta1)*cos(theta2)^3*cos(theta3)^3 +
2*L4^2*mL4*theta4dd*cos(theta1)*cos(theta2)^3*cos(theta3)^3 - 1/2*IL4xz*theta1d*theta4d*cos(theta1)*sin(theta2)*sin(theta3) -
1/2*IL4xz*theta1d*theta4d*cos(theta3)*sin(theta1)*sin(theta2) + 1/2*IL4xz*theta2d*theta4d*cos(theta1)*sin(theta2)*sin(theta3) +
IL4xz*theta2d*theta4d*cos(theta2)*sin(theta1)*sin(theta3) + IL4xz*theta2d*theta4d*cos(theta3)*sin(theta1)*sin(theta2) +
1/2*IL4xz*theta3d*theta4d*cos(theta1)*sin(theta2)*sin(theta3) + IL4xz*theta3d*theta4d*cos(theta2)*sin(theta1)*sin(theta3) +
IL4xz*theta3d*theta4d*cos(theta3)*sin(theta1)*sin(theta2) - d4*g*mL4*cos(theta2)*sin(theta1)*sin(theta3) -
d4*g*mL4*cos(theta3)*sin(theta1)*sin(theta2) - d4*g*mm4*cos(theta2)*sin(theta1)*sin(theta3) - d4*g*mm4*cos(theta3)*sin(theta1)*sin(theta2)
- IL4yz*theta1d*theta4d*sin(theta1)*sin(theta2)*sin(theta3) + 1/2*IL4yz*theta2d*theta4d*sin(theta1)*sin(theta2)*sin(theta3) +
1/2*IL4yz*theta3d*theta4d*sin(theta1)*sin(theta2)*sin(theta3) + 3*IL4xz*theta4d^2*cos(theta1)*cos(theta2)*cos(theta3)^2*sin(theta2) +
3*IL4xz*theta4d^2*cos(theta1)*cos(theta2)^2*cos(theta3)*sin(theta3) - IL4yz*theta4d^2*cos(theta1)*cos(theta2)*cos(theta3)^2*sin(theta2) -
IL4yz*theta4d^2*cos(theta1)*cos(theta2)^2*cos(theta3)*sin(theta3) + 2*IL4xx*theta4dd*cos(theta1)^2*cos(theta2)*cos(theta3)^2*sin(theta2) +
Im4xy*Kr4*theta2dd*cos(theta1)^2*cos(theta2)*cos(theta3) + Im4xy*Kr4*theta3dd*cos(theta1)^2*cos(theta2)*cos(theta3) +
2*Im4xy*Kr4*theta4dd*cos(theta1)^2*cos(theta2)*cos(theta3) - L4^2*mL4*theta2dd*cos(theta1)*sin(theta2)^3*sin(theta3)^3 -
L4^2*mL4*theta3dd*cos(theta1)*sin(theta2)^3*sin(theta3)^3 - 2*L4^2*mL4*theta4dd*cos(theta1)*sin(theta2)^3*sin(theta3)^3 +
d4^2*mm4*theta2dd*cos(theta1)*cos(theta2)^2*sin(theta3)^2 + d4^2*mm4*theta2dd*cos(theta1)*cos(theta3)^2*sin(theta2)^2 +
d4^2*mm4*theta3dd*cos(theta1)*cos(theta2)^2*sin(theta3)^2 + d4^2*mm4*theta3dd*cos(theta1)*cos(theta3)^2*sin(theta2)^2 -
IL4xz*theta4d^2*cos(theta1)*cos(theta2)*sin(theta2)*sin(theta3)^2 + IL4xz*theta4d^2*cos(theta2)*cos(theta3)^2*sin(theta1)*sin(theta2) -
IL4xz*theta4d^2*cos(theta1)*cos(theta3)*sin(theta2)^2*sin(theta3) + IL4yz*theta4d^2*cos(theta1)*cos(theta2)*sin(theta2)*sin(theta3)^2 +
IL4yz*theta4d^2*cos(theta1)*cos(theta3)*sin(theta2)^2*sin(theta3) - 2*IL4xy*theta4dd*cos(theta1)*cos(theta3)^2*sin(theta1)*sin(theta2)^2 +
Im4xy*Kr4*theta2dd*cos(theta1)^2*cos(theta3)*sin(theta2) + Im4xy*Kr4*theta3dd*cos(theta1)^2*cos(theta3)*sin(theta2) +
2*Im4xy*Kr4*theta4dd*cos(theta1)^2*cos(theta3)*sin(theta2) + 4*L4*d4*mL4*theta2dd*cos(theta2)^2*cos(theta3)^2 +
4*L4*d4*mL4*theta3dd*cos(theta2)^2*cos(theta3)^2 - IL4xz*theta4d^2*cos(theta3)*sin(theta1)*sin(theta2)^2*sin(theta3) -
Im4xz*Kr4^2*theta4d^2*cos(theta1)*cos(theta2)^2*cos(theta3)^2 + Im4xx*Kr4^2*theta4dd*cos(theta1)^2*cos(theta2)^2*cos(theta3)^2 -
Im4xy*Kr4*theta2dd*cos(theta2)*sin(theta1)^2*sin(theta3) - Im4xy*Kr4*theta2dd*cos(theta3)*sin(theta1)^2*sin(theta2) -
Im4xy*Kr4*theta3dd*cos(theta2)*sin(theta1)^2*sin(theta3) - Im4xy*Kr4*theta3dd*cos(theta3)*sin(theta1)^2*sin(theta2) -
2*Im4xy*Kr4*theta4dd*cos(theta2)*sin(theta1)^2*sin(theta3) - 2*Im4xy*Kr4*theta4dd*cos(theta3)*sin(theta1)^2*sin(theta2) -
L3*g*mL3*cos(theta1)*cos(theta3)^2*sin(theta2) - L4*g*mL4*cos(theta1)*cos(theta3)^2*sin(theta2) -
1/2*IL4xx*theta2d*theta4d*cos(theta1)^2*cos(theta2)*cos(theta3) - 1/2*IL4xx*theta3d*theta4d*cos(theta1)^2*cos(theta2)*cos(theta3) -
1/2*IL4xy*theta2d*theta4d*cos(theta1)^2*cos(theta2)*cos(theta3) - 1/2*IL4xy*theta3d*theta4d*cos(theta1)^2*cos(theta2)*cos(theta3) +
2*Im4xz*Kr4^2*theta4d^2*cos(theta1)*cos(theta3)^2*sin(theta2)^2 + Im4xz*Kr4^2*theta4d^2*cos(theta2)^2*cos(theta3)^2*sin(theta1) -
2*Im4yz*Kr4^2*theta4d^2*cos(theta2)^2*cos(theta3)^2*sin(theta1) + Im4xx*Kr4^2*theta4dd*cos(theta1)^2*cos(theta3)^2*sin(theta2)^2 +
4*L4*d4*mL4*theta2dd*sin(theta2)^2*sin(theta3)^2 + 4*L4*d4*mL4*theta3dd*sin(theta2)^2*sin(theta3)^2 +
1/2*IL4xx*theta2d*theta4d*cos(theta2)*cos(theta3)*sin(theta1)^2 - 1/2*IL4xx*theta2d*theta4d*cos(theta1)^2*cos(theta3)*sin(theta2) +
1/2*IL4xx*theta3d*theta4d*cos(theta2)*cos(theta3)*sin(theta1)^2 - 1/2*IL4xx*theta3d*theta4d*cos(theta1)^2*cos(theta3)*sin(theta2) +
IL4xy*theta2d*theta4d*cos(theta2)*cos(theta3)*sin(theta1)^2 + 1/2*IL4xy*theta2d*theta4d*cos(theta1)^2*cos(theta2)*sin(theta3) +
1/2*IL4xy*theta2d*theta4d*cos(theta1)^2*cos(theta3)*sin(theta2) + IL4xy*theta3d*theta4d*cos(theta2)*cos(theta3)*sin(theta1)^2 +
1/2*IL4xy*theta3d*theta4d*cos(theta1)^2*cos(theta2)*sin(theta3) + 1/2*IL4xy*theta3d*theta4d*cos(theta1)^2*cos(theta3)*sin(theta2) +
1/2*IL4yy*theta2d*theta4d*cos(theta1)^2*cos(theta2)*sin(theta3) + 1/2*IL4yy*theta2d*theta4d*cos(theta1)^2*cos(theta3)*sin(theta2) +
1/2*IL4yy*theta3d*theta4d*cos(theta1)^2*cos(theta2)*sin(theta3) + 1/2*IL4yy*theta3d*theta4d*cos(theta1)^2*cos(theta3)*sin(theta2) +
IL4xx*theta2dd*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) + IL4xx*theta3dd*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) +
2*IL4xx*theta4dd*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) - Im4xz*Kr4^2*theta4d^2*cos(theta1)*sin(theta2)^2*sin(theta3)^2 +
2*Im4yz*Kr4^2*theta4d^2*cos(theta2)^2*sin(theta1)*sin(theta3)^2 + 2*Im4yz*Kr4^2*theta4d^2*cos(theta3)^2*sin(theta1)*sin(theta2)^2 +
Im4yy*Kr4^2*theta4dd*cos(theta2)^2*sin(theta1)^2*sin(theta3)^2 + Im4yy*Kr4^2*theta4dd*cos(theta3)^2*sin(theta1)^2*sin(theta2)^2 -
d4*g*mL4*cos(theta1)*cos(theta3)^2*sin(theta2) + d4*g*mm4*cos(theta1)*cos(theta2)*sin(theta3)^2 -
d4*g*mm4*cos(theta1)*cos(theta3)^2*sin(theta2) + L3^2*mL3*theta1dd*cos(theta1)^2*cos(theta2)^2*cos(theta3)^2 +
L4^2*mL4*theta1dd*cos(theta1)^2*cos(theta2)^2*cos(theta3)^2 - L4^2*mL4*theta1dd*cos(theta1)^2*cos(theta2)^3*cos(theta3)^3 -
L4^2*mL4*theta4dd*cos(theta1)^2*cos(theta2)^3*cos(theta3)^3 + 2*L4^2*mL4*theta4dd*cos(theta1)^2*cos(theta2)^4*cos(theta3)^4 +
1/2*IL4xx*theta2d*theta4d*cos(theta3)*sin(theta1)^2*sin(theta2) + 1/2*IL4xx*theta3d*theta4d*cos(theta3)*sin(theta1)^2*sin(theta2) +
1/2*IL4xy*theta2d*theta4d*cos(theta1)^2*sin(theta2)*sin(theta3) + 1/2*IL4xy*theta3d*theta4d*cos(theta1)^2*sin(theta2)*sin(theta3) -
1/2*IL4yy*theta2d*theta4d*cos(theta2)*sin(theta1)^2*sin(theta3) - 1/2*IL4yy*theta2d*theta4d*cos(theta3)*sin(theta1)^2*sin(theta2) -
1/2*IL4yy*theta3d*theta4d*cos(theta2)*sin(theta1)^2*sin(theta3) - 1/2*IL4yy*theta3d*theta4d*cos(theta3)*sin(theta1)^2*sin(theta2) +
IL4xx*theta2dd*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2) + IL4xx*theta3dd*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2) +

2*IL4xx*theta4dd*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2) - IL4yy*theta2dd*cos(theta1)*cos(theta2)*sin(theta1)*sin(theta3) -
IL4yy*theta2dd*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2) - IL4yy*theta3dd*cos(theta1)*cos(theta2)*sin(theta1)*sin(theta3) -
IL4yy*theta3dd*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2) - 2*IL4yy*theta4dd*cos(theta1)*cos(theta2)*sin(theta1)*sin(theta3) -
2*IL4yy*theta4dd*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2) - 2*IL4zz*theta4dd*cos(theta2)*cos(theta3)*sin(theta2)*sin(theta3) -
Im4zz*Kr4*theta1d*theta4d*cos(theta2)*sin(theta3) - Im4zz*Kr4*theta1d*theta4d*cos(theta3)*sin(theta2) -
2*Im4yz*Kr4^2*theta4d^2*sin(theta1)*sin(theta2)^2*sin(theta3)^2 + L3^2*mL3*theta1dd*cos(theta2)^2*cos(theta3)^2*sin(theta1)^2 +
3*L3^2*mL3*theta2dd*cos(theta1)^2*cos(theta2)^2*sin(theta3)^2 + 3*L3^2*mL3*theta2dd*cos(theta1)^2*cos(theta3)^2*sin(theta2)^2 +
2*L3^2*mL3*theta3dd*cos(theta1)^2*cos(theta2)^2*sin(theta3)^2 + 2*L3^2*mL3*theta3dd*cos(theta1)^2*cos(theta3)^2*sin(theta2)^2 +
L4^2*mL4*theta1dd*cos(theta2)^2*cos(theta3)^2*sin(theta1)^2 + 2*L4^2*mL4*theta2dd*cos(theta1)^2*cos(theta2)^2*sin(theta3)^2 +
2*L4^2*mL4*theta2dd*cos(theta1)^2*cos(theta3)^2*sin(theta2)^2 - L4^2*mL4*theta1dd*cos(theta2)^3*cos(theta3)^3*sin(theta1)^2 +
2*L4^2*mL4*theta3dd*cos(theta1)^2*cos(theta2)^2*sin(theta3)^2 + 2*L4^2*mL4*theta3dd*cos(theta1)^2*cos(theta3)^2*sin(theta2)^2 -
L4^2*mL4*theta4dd*cos(theta2)^3*cos(theta3)^3*sin(theta1)^2 + L4^2*mL4*theta4dd*cos(theta2)^4*cos(theta3)^4*sin(theta1)^2 -
IL4xy*theta2d*theta4d*sin(theta1)^2*sin(theta2)*sin(theta3) - IL4xy*theta3d*theta4d*sin(theta1)^2*sin(theta2)*sin(theta3) +
d4^2*mL4*theta1dd*cos(theta1)^2*cos(theta2)^2*cos(theta3)^2 + d4^2*mm4*theta1dd*cos(theta1)^2*cos(theta2)^2*cos(theta3)^2 +
IL4xx*theta4d^2*cos(theta1)*cos(theta2)^2*cos(theta3)^2*sin(theta1) + 2*IL4xy*theta4d^2*cos(theta1)*cos(theta2)^2*cos(theta3)^2*sin(theta1)
+ IL4xx*theta4d^2*cos(theta1)^2*cos(theta2)^2*cos(theta3)*sin(theta3) + IL4xy*theta4d^2*cos(theta1)^2*cos(theta2)*cos(theta3)^2*sin(theta2)
+ IL4xy*theta4d^2*cos(theta1)^2*cos(theta2)^2*cos(theta3)*sin(theta3) + L3^2*mL3*theta1dd*cos(theta1)^2*sin(theta2)^2*sin(theta3)^2 +
L3^2*mL3*theta3dd*cos(theta2)^2*sin(theta1)^2*sin(theta3)^2 + L3^2*mL3*theta3dd*cos(theta3)^2*sin(theta1)^2*sin(theta2)^2 +
L4^2*mL4*theta1dd*cos(theta1)^2*sin(theta2)^2*sin(theta3)^2 + 2*L4^2*mL4*theta2dd*cos(theta2)^2*sin(theta1)^2*sin(theta3)^2 +
2*L4^2*mL4*theta2dd*cos(theta3)^2*sin(theta1)^2*sin(theta2)^2 + L4^2*mL4*theta1dd*cos(theta1)^2*sin(theta2)^3*sin(theta3)^3 +
2*L4^2*mL4*theta3dd*cos(theta2)^2*sin(theta1)^2*sin(theta3)^2 + 2*L4^2*mL4*theta3dd*cos(theta3)^2*sin(theta1)^2*sin(theta2)^2 +
L4^2*mL4*theta4dd*cos(theta1)^2*sin(theta2)^3*sin(theta3)^3 + 2*L4^2*mL4*theta4dd*cos(theta1)^2*sin(theta2)^4*sin(theta3)^4 +
L4^2*mL4*theta4dd*cos(theta2)^4*sin(theta1)^2*sin(theta3)^4 + L4^2*mL4*theta4dd*cos(theta3)^4*sin(theta1)^2*sin(theta2)^4 +
d4^2*mL4*theta1dd*cos(theta2)^2*cos(theta3)^2*sin(theta1)^2 + 2*d4^2*mL4*theta2dd*cos(theta1)^2*cos(theta2)^2*sin(theta3)^2 +
2*d4^2*mL4*theta2dd*cos(theta1)^2*cos(theta3)^2*sin(theta2)^2 + 2*d4^2*mL4*theta3dd*cos(theta1)^2*cos(theta2)^2*sin(theta3)^2 +
2*d4^2*mL4*theta3dd*cos(theta1)^2*cos(theta3)^2*sin(theta2)^2 + d4^2*mm4*theta1dd*cos(theta2)^2*cos(theta3)^2*sin(theta1)^2 +
2*d4^2*mm4*theta2dd*cos(theta1)^2*cos(theta2)^2*sin(theta3)^2 + 2*d4^2*mm4*theta2dd*cos(theta1)^2*cos(theta3)^2*sin(theta2)^2 +
3*d4^2*mm4*theta3dd*cos(theta1)^2*cos(theta2)^2*sin(theta3)^2 + 3*d4^2*mm4*theta3dd*cos(theta1)^2*cos(theta3)^2*sin(theta2)^2 +
IL4xx*theta4d^2*cos(theta1)*cos(theta3)^2*sin(theta1)*sin(theta2)^2 - IL4xy*theta4d^2*cos(theta1)*cos(theta2)^2*sin(theta1)*sin(theta3)^2 -
IL4xy*theta4d^2*cos(theta1)*cos(theta3)^2*sin(theta1)*sin(theta2)^2 + IL4xx*theta4d^2*cos(theta1)^2*cos(theta3)*sin(theta2)^2*sin(theta3) -
IL4xy*theta4d^2*cos(theta2)*cos(theta3)^2*sin(theta1)^2*sin(theta2) - IL4yy*theta4d^2*cos(theta1)*cos(theta2)^2*sin(theta1)*sin(theta3)^2 -
IL4yy*theta4d^2*cos(theta1)*cos(theta3)^2*sin(theta1)*sin(theta2)^2 - IL4xy*theta4d^2*cos(theta2)^2*cos(theta3)*sin(theta1)^2*sin(theta3) -
2*IL4yy*theta4d^2*cos(theta2)*cos(theta3)^2*sin(theta1)^2*sin(theta2) -
2*IL4yy*theta4d^2*cos(theta2)^2*cos(theta3)*sin(theta1)^2*sin(theta3) + L3^2*mL3*theta1dd*sin(theta1)^2*sin(theta2)^2*sin(theta3)^2 +
L4^2*mL4*theta1dd*sin(theta1)^2*sin(theta2)^2*sin(theta3)^2 + L4^2*mL4*theta1dd*sin(theta1)^2*sin(theta2)^3*sin(theta3)^3 +
L4^2*mL4*theta4dd*sin(theta1)^2*sin(theta2)^3*sin(theta3)^3 + L4^2*mL4*theta4dd*sin(theta1)^2*sin(theta2)^4*sin(theta3)^4 +
d4^2*mL4*theta1dd*cos(theta1)^2*sin(theta2)^2*sin(theta3)^2 + 2*d4^2*mL4*theta2dd*cos(theta2)^2*sin(theta1)^2*sin(theta3)^2 +
2*d4^2*mL4*theta2dd*cos(theta3)^2*sin(theta1)^2*sin(theta2)^2 + 2*d4^2*mL4*theta3dd*cos(theta2)^2*sin(theta1)^2*sin(theta3)^2 +
2*d4^2*mL4*theta3dd*cos(theta3)^2*sin(theta1)^2*sin(theta2)^2 + d4^2*mm4*theta1dd*cos(theta1)^2*sin(theta2)^2*sin(theta3)^2 +
2*d4^2*mm4*theta2dd*cos(theta2)^2*sin(theta1)^2*sin(theta3)^2 + 2*d4^2*mm4*theta2dd*cos(theta3)^2*sin(theta1)^2*sin(theta2)^2 +
2*d4^2*mm4*theta3dd*cos(theta2)^2*sin(theta1)^2*sin(theta3)^2 + 2*d4^2*mm4*theta3dd*cos(theta3)^2*sin(theta1)^2*sin(theta2)^2 +
2*IL4yy*theta4d^2*cos(theta2)*sin(theta1)^2*sin(theta2)*sin(theta3)^2 +
2*IL4yy*theta4d^2*cos(theta3)*sin(theta1)^2*sin(theta2)^2*sin(theta3) - Im4xx*Kr4^2*theta4d^2*cos(theta1)^2*cos(theta2)^2*cos(theta3)^2 +
d4^2*mL4*theta1dd*sin(theta1)^2*sin(theta2)^2*sin(theta3)^2 + d4^2*mm4*theta1dd*sin(theta1)^2*sin(theta2)^2*sin(theta3)^2 +
Im4xx*Kr4^2*theta4d^2*cos(theta1)^2*cos(theta3)^2*sin(theta2)^2 + Im4xy*Kr4^2*theta4d^2*cos(theta1)^2*cos(theta3)^2*sin(theta2)^2 -
L2^2*mL2*theta1dd*cos(theta1)^2*cos(theta2)*sin(theta2) - L2^2*mL2*theta2dd*cos(theta1)^2*cos(theta2)*sin(theta2) +
L4*g*mL4*cos(theta1)*sin(theta2)^2*sin(theta3)^3 + 2*IL4xz*theta4dd*cos(theta1)*cos(theta2)*cos(theta3)^2*sin(theta2) -
Im4xy*Kr4^2*theta4d^2*cos(theta3)^2*sin(theta1)^2*sin(theta2)^2 - Im4xz*Kr4*theta1dd*cos(theta1)*cos(theta2)*cos(theta3) -
Im4yz*Kr4*theta2dd*cos(theta1)*cos(theta2)*cos(theta3) - Im4xz*Kr4*theta4dd*cos(theta1)*cos(theta2)*cos(theta3) -
Im4yz*Kr4*theta3dd*cos(theta1)*cos(theta2)*cos(theta3) - 2*Im4yz*Kr4*theta4dd*cos(theta1)*cos(theta2)*cos(theta3) -
4*L4^2*mL4*theta4d^2*cos(theta1)^2*cos(theta2)^4*cos(theta3)^4 - L4^2*mL4*theta4d^2*cos(theta1)^3*cos(theta2)^4*cos(theta3)^4 +
L2^2*mL2*theta1dd*cos(theta2)*sin(theta1)^2*sin(theta2) + L2^2*mL2*theta2dd*cos(theta2)*sin(theta1)^2*sin(theta2) -
2*IL4xz*theta4dd*cos(theta1)*cos(theta3)*sin(theta2)^2*sin(theta3) + 2*IL4yz*theta4dd*cos(theta2)*cos(theta3)^2*sin(theta1)*sin(theta2) +
2*IL4yz*theta4dd*cos(theta2)^2*cos(theta3)*sin(theta1)*sin(theta3) - L3*a2*mL3*theta2dd*cos(theta2)^2*cos(theta3) +
L3*a2*mL3*theta3dd*cos(theta2)^2*cos(theta3) + 3*L4*a2*mL4*theta2dd*cos(theta2)^2*cos(theta3) +
L4*a2*mL4*theta3dd*cos(theta2)^2*cos(theta3) - 1/2*IL4xx*theta2d*theta4d*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) -
1/2*IL4xx*theta3d*theta4d*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) +
IL4xy*theta2d*theta4d*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) + IL4xy*theta3d*theta4d*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1)
+ IL4yy*theta2d*theta4d*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) +
IL4yy*theta3d*theta4d*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta1) -
Im4xz*Kr4^2*theta4d^2*cos(theta1)*cos(theta2)*sin(theta2)*sin(theta3)^2 +
Im4xz*Kr4^2*theta4d^2*cos(theta2)*cos(theta3)^2*sin(theta1)*sin(theta2) -
Im4xz*Kr4^2*theta4d^2*cos(theta1)*cos(theta3)*sin(theta2)^2*sin(theta3) +
Im4yz*Kr4^2*theta4d^2*cos(theta1)*cos(theta2)*sin(theta2)*sin(theta3)^2 +
Im4yz*Kr4^2*theta4d^2*cos(theta1)*cos(theta3)*sin(theta2)^2*sin(theta3) -
2*Im4xy*Kr4^2*theta4dd*cos(theta1)*cos(theta3)^2*sin(theta1)*sin(theta2)^2 +
2*L4^2*mL4*theta1d*theta4d*cos(theta1)^2*cos(theta2)^3*cos(theta3)^3 -
1/2*L4^2*mL4*theta2d*theta4d*cos(theta1)^2*cos(theta2)^3*cos(theta3)^3 -
1/2*L4^2*mL4*theta3d*theta4d*cos(theta1)^2*cos(theta2)^3*cos(theta3)^3 - d4*g*mL4*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta3) -
d4*g*mm4*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta3) + 1/2*IL4xx*theta2d*theta4d*cos(theta1)*cos(theta2)*sin(theta1)*sin(theta3) +

1/2*IL4xx*theta2d*theta4d*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2) +
1/2*IL4xx*theta3d*theta4d*cos(theta1)*cos(theta2)*sin(theta1)*sin(theta3) +
1/2*IL4xx*theta3d*theta4d*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2) +
IL4xy*theta2d*theta4d*cos(theta1)*cos(theta2)*sin(theta1)*sin(theta3) +
2*IL4xy*theta2d*theta4d*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2) +
IL4xy*theta3d*theta4d*cos(theta1)*cos(theta2)*sin(theta1)*sin(theta3) +
2*IL4xy*theta3d*theta4d*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2) -
Im4xz*Kr4^2*theta4d^2*cos(theta3)*sin(theta1)*sin(theta2)^2*sin(theta3) +
1/2*L4^2*mL4*theta2d*theta4d*cos(theta1)^2*cos(theta2)^3*sin(theta3)^3 +
1/2*L4^2*mL4*theta2d*theta4d*cos(theta1)^2*cos(theta3)^3*sin(theta2)^3 +
1/2*L4^2*mL4*theta2d*theta4d*cos(theta2)^3*cos(theta3)^3*sin(theta1)^2 +
1/2*L4^2*mL4*theta3d*theta4d*cos(theta1)^2*cos(theta2)^3*sin(theta3)^3 +
1/2*L4^2*mL4*theta3d*theta4d*cos(theta1)^2*cos(theta3)^3*sin(theta2)^3 +
1/2*L4^2*mL4*theta3d*theta4d*cos(theta2)^3*cos(theta3)^3*sin(theta1)^2 + d4*g*mm4*cos(theta1)*cos(theta3)*sin(theta2)*sin(theta3) -
L3^2*mL3*theta1dd*cos(theta1)^2*cos(theta2)*cos(theta3)^2*sin(theta2) -
L3^2*mL3*theta1dd*cos(theta1)^2*cos(theta2)^2*cos(theta3)*sin(theta3) -
L3^2*mL3*theta2dd*cos(theta1)^2*cos(theta2)*cos(theta3)^2*sin(theta2) -
L3^2*mL3*theta2dd*cos(theta1)^2*cos(theta2)^2*cos(theta3)*sin(theta3) +
L4^2*mL4*theta2dd*cos(theta1)*cos(theta2)^3*cos(theta3)^3*sin(theta1) +
L4^2*mL4*theta3dd*cos(theta1)*cos(theta2)^3*cos(theta3)^3*sin(theta1) +
2*L4^2*mL4*theta4dd*cos(theta1)*cos(theta2)^3*cos(theta3)^3*sin(theta1) +
1/2*IL4xx*theta2d*theta4d*cos(theta1)*sin(theta1)*sin(theta2)*sin(theta3) +
1/2*IL4xx*theta3d*theta4d*cos(theta1)*sin(theta1)*sin(theta2)*sin(theta3) -
IL4yy*theta2d*theta4d*cos(theta1)*sin(theta1)*sin(theta2)*sin(theta3) - IL4yy*theta3d*theta4d*cos(theta1)*sin(theta1)*sin(theta2)*sin(theta3) -
L4^2*mL4*theta1d*theta4d*cos(theta2)^3*sin(theta1)^2*sin(theta3)^3 - L4^2*mL4*theta1d*theta4d*cos(theta3)^3*sin(theta1)^2*sin(theta2)^3 -
1/2*L4^2*mL4*theta2d*theta4d*cos(theta2)^3*sin(theta1)^2*sin(theta3)^3 -
1/2*L4^2*mL4*theta2d*theta4d*cos(theta3)^3*sin(theta1)^2*sin(theta2)^3 -
1/2*L4^2*mL4*theta3d*theta4d*cos(theta2)^3*sin(theta1)^2*sin(theta3)^3 -
1/2*L4^2*mL4*theta3d*theta4d*cos(theta3)^3*sin(theta1)^2*sin(theta2)^3 +
L3^2*mL3*theta1dd*cos(theta1)^2*cos(theta2)*sin(theta2)*sin(theta3)^2 +
L3^2*mL3*theta2dd*cos(theta1)*cos(theta2)^2*sin(theta1)*sin(theta3)^2 +
L3^2*mL3*theta2dd*cos(theta1)*cos(theta3)^2*sin(theta1)*sin(theta2)^2 +
L3^2*mL3*theta1dd*cos(theta1)^2*cos(theta3)*sin(theta2)^2*sin(theta3) +
L3^2*mL3*theta2dd*cos(theta1)^2*cos(theta2)*sin(theta2)*sin(theta3)^2 +
L3^2*mL3*theta3dd*cos(theta1)*cos(theta2)^2*sin(theta1)*sin(theta3)^2 +
L3^2*mL3*theta3dd*cos(theta1)*cos(theta3)^2*sin(theta1)*sin(theta2)^2 +
L3^2*mL3*theta2dd*cos(theta1)^2*cos(theta3)*sin(theta2)^2*sin(theta3) -
L4^2*mL4*theta2dd*cos(theta1)*cos(theta2)^3*sin(theta1)*sin(theta3)^3 -
L4^2*mL4*theta2dd*cos(theta1)*cos(theta3)^3*sin(theta1)*sin(theta2)^3 -
L4^2*mL4*theta3dd*cos(theta1)*cos(theta2)^3*sin(theta1)*sin(theta3)^3 -
L4^2*mL4*theta3dd*cos(theta1)*cos(theta3)^3*sin(theta1)*sin(theta2)^3 -
2*L4^2*mL4*theta4dd*cos(theta1)*cos(theta2)^3*sin(theta1)*sin(theta3)^3 -
2*L4^2*mL4*theta4dd*cos(theta1)*cos(theta3)^3*sin(theta1)*sin(theta2)^3 +
2*IL4xx*theta4d^2*cos(theta1)*cos(theta2)*cos(theta3)^2*sin(theta1)*sin(theta2) +
3*IL4xy*theta4d^2*cos(theta1)*cos(theta2)*cos(theta3)^2*sin(theta1)*sin(theta2) +
2*IL4xx*theta4d^2*cos(theta1)^2*cos(theta2)*cos(theta3)*sin(theta2)*sin(theta3) +
IL4xy*theta4d^2*cos(theta1)*cos(theta2)^2*cos(theta3)*sin(theta1)*sin(theta3) +
IL4xy*theta4d^2*cos(theta1)^2*cos(theta2)*cos(theta3)*sin(theta2)*sin(theta3) +
2*L3*a2*mL3*theta1dd*cos(theta1)^2*cos(theta2)^2*cos(theta3) + 2*L4*a2*mL4*theta1dd*cos(theta1)^2*cos(theta2)^2*cos(theta3) +
L4*a2*mL4*theta2dd*cos(theta1)*cos(theta2)^3*cos(theta3)^2 + L4*a2*mL4*theta4dd*cos(theta1)*cos(theta2)^3*cos(theta3)^2 +
L4*d4*mL4*theta2dd*cos(theta1)*cos(theta2)^3*cos(theta3)^3 + L4*d4*mL4*theta3dd*cos(theta1)*cos(theta2)^3*cos(theta3)^3 +
2*L4*d4*mL4*theta4dd*cos(theta1)*cos(theta2)^3*cos(theta3)^3 +
10*L4^2*mL4*theta4d^2*cos(theta1)^2*cos(theta2)^2*cos(theta3)^4*sin(theta2)^2 +
10*L4^2*mL4*theta4d^2*cos(theta1)^2*cos(theta2)^4*cos(theta3)^2*sin(theta3)^2 -
L4^2*mL4*theta4d^2*cos(theta1)^2*cos(theta2)^4*cos(theta3)^4*sin(theta1)^2 -
IL4xy*theta4d^2*cos(theta1)*cos(theta2)*sin(theta1)*sin(theta2)*sin(theta3)^2 -
3*IL4xy*theta4d^2*cos(theta1)*cos(theta3)*sin(theta1)*sin(theta2)^2*sin(theta3) -
IL4xy*theta4d^2*cos(theta2)*cos(theta3)*sin(theta1)^2*sin(theta2)*sin(theta3) - L3*a2*mL3*theta1dd*cos(theta1)^2*cos(theta2)^2*sin(theta3)
+ 2*L3*a2*mL3*theta1dd*cos(theta2)^2*cos(theta3)*sin(theta1)^2 + 3*L3*a2*mL3*theta2dd*cos(theta1)^2*cos(theta3)*sin(theta2)^2 -
L3*a2*mL3*theta2dd*cos(theta1)^2*cos(theta2)^2*sin(theta3) + L3*a2*mL3*theta3dd*cos(theta1)^2*cos(theta3)*sin(theta2)^2 +
2*L4*a2*mL4*theta1dd*cos(theta2)^2*cos(theta3)*sin(theta1)^2 + 3*L4*a2*mL4*theta2dd*cos(theta1)^2*cos(theta3)*sin(theta2)^2 +
L4*a2*mL4*theta3dd*cos(theta1)^2*cos(theta3)*sin(theta2)^2 - 2*Im4zz*Kr4^2*theta4dd*cos(theta2)*cos(theta3)*sin(theta2)*sin(theta3) +
2*L4^2*mL4*theta4d^2*cos(theta1)^2*cos(theta2)^2*sin(theta2)^2*sin(theta3)^4 +
2*L4^2*mL4*theta4d^2*cos(theta1)^2*cos(theta3)^2*sin(theta2)^4*sin(theta3)^2 +
2*a2*d4*mL4*theta1dd*cos(theta1)^2*cos(theta2)^2*cos(theta3) + 2*a2*d4*mm4*theta1dd*cos(theta1)^2*cos(theta2)^2*cos(theta3) +
L3*a2*mL3*theta2dd*cos(theta3)*sin(theta1)^2*sin(theta2)^2 + L3*a2*mL3*theta3dd*cos(theta3)*sin(theta1)^2*sin(theta2)^2 +
3*L4*a2*mL4*theta2dd*cos(theta3)*sin(theta1)^2*sin(theta2)^2 + L4*a2*mL4*theta3dd*cos(theta3)*sin(theta1)^2*sin(theta2)^2 +
Im4xx*Kr4^2*theta4d^2*cos(theta1)*cos(theta2)^2*cos(theta3)^2*sin(theta1) +
2*Im4xy*Kr4^2*theta4d^2*cos(theta1)*cos(theta2)^2*cos(theta3)^2*sin(theta1) +
Im4xx*Kr4^2*theta4d^2*cos(theta1)^2*cos(theta2)^2*cos(theta3)*sin(theta3) +

Im4xy*Kr4^2*theta4d^2*cos(theta1)^2*cos(theta2)*cos(theta3)^2*sin(theta2) +
Im4xy*Kr4^2*theta4d^2*cos(theta1)^2*cos(theta2)^2*cos(theta3)*sin(theta3) - L4*d4*mL4*theta2dd*cos(theta1)*sin(theta2)^3*sin(theta3)^3 -
L4*d4*mL4*theta3dd*cos(theta1)*sin(theta2)^3*sin(theta3)^3 - 2*L4*d4*mL4*theta4dd*cos(theta1)*sin(theta2)^3*sin(theta3)^3 +
2*a2*d4*mL4*theta1dd*cos(theta2)^2*cos(theta3)*sin(theta1)^2 + 3*a2*d4*mL4*theta2dd*cos(theta1)^2*cos(theta3)*sin(theta2)^2 +
a2*d4*mL4*theta3dd*cos(theta1)^2*cos(theta3)*sin(theta2)^2 + 2*a2*d4*mm4*theta1dd*cos(theta2)^2*cos(theta3)*sin(theta1)^2 +
3*a2*d4*mm4*theta2dd*cos(theta1)^2*cos(theta3)*sin(theta2)^2 + a2*d4*mm4*theta3dd*cos(theta1)^2*cos(theta3)*sin(theta2)^2 +
1/2*Im4xz*Kr4*theta1d*theta4d*cos(theta1)*cos(theta2)*cos(theta3) - 1/2*Im4xz*Kr4*theta2d*theta4d*cos(theta1)*cos(theta2)*cos(theta3) -
1/2*Im4xz*Kr4*theta3d*... Output truncated. Text exceeds maximum line length of 25,000 characters for Command Window display.

APPEMDICES 3

Matlab program for calculation of Inertia Matrix and Equation of Motion.

```
symsL1L2L3L4s1s2s3c1c2c3s32c32a2d4Kr1Kr2Kr3Kr4;
symstheta1theta2theta3theta4theta1ddtheta2ddtheta3ddtheta4dd
symstheta1dtheta2dtheta3dtheta4d
thetad=transpose([theta1d theta2d theta3d theta4d]);
thetadd=transpose([theta1dd theta2dd theta3dd theta4dd]);
s1=sin(theta1);
s2=sin(theta2);
s3=sin(theta3);
s32=sin(theta3+theta2);
c1=cos(theta1);
c2=cos(theta2);
c3=cos(theta3);
c32=cos(theta3+theta2);

% Jacobian for link
JpL1=[0 0 0 0;0 0 0 0;0 0 0 0];
JpL2=[L2*c2*c1 -c1*L2*s2 0 0;L2*c2*s1 L2*s1*s2 0 0;0 L2*c2 0 0];
JpL3=[-a2*c2*s1-L3*c32*s1 -a2*s2*c1-L3*s32*c1 -L3*s32*c1 0;
    a2*c2*c1+L3*c32*c1 -a2*s2*s1-L3*s32*c1 -L3*s32*s1 0;
    0 a2*c2-L3*c32 L3*c32 0];
JpL4=[-a2*c2*s1-d4*c32*s1-L4*c32*s1 -a2*s2*c1-d4*s32*c1-L4*s32*c1 -
d4*s32*c1-L4*s32*c1 L4*s1*c2*s3*s32+L4*s2*s1*c3*s32-
L4*s2*s3*s1*c32+L4*c2*c3*c32*s1;
    a2*c2*c1+d4*c32*c1+L4*c32*c1 -a2*s2*s1-d4*s32*s1-L4*s32*s1 -
d4*s32*s1-L4*s32*s1 L4*c1*s2*s3*c32-L4*c2*c1*c3*c32-L4*c1*c2*c3*s32-
L4*c1*c2*c3*s32;
    0 a2*c2+d4*c32+L4*c32 d4*c32+L4*c32
L4*c1*c2*c3*c32*s1+L4*c1*s2*c3*c32*s1-
L4*c32*c1*s2*s3+L4*c1*c2*c3*c32];

JoL1=[0 0 0 0;0 0 0 0;1 0 0 0];
JoL2=[0 s1 0 0;0 -c1 0 0;1 0 0 0];
JoL3=[0 s1 s1 0;0 -c1 -c1 0;1 0 0 0];
JoL4=[0 s1 s1 c1*c2*c3+c1*s2*c3;0 -c1 -c1 s1*c2*s3+s1*s2*c3;1 0 0
s2*s3-c2*c3];

% Jacobian for motor
Jpm1=[0 0 0 0;0 0 0 0;0 0 0 0];
Jpm2=[0 0 0 0;0 0 0 0;0 0 0 0];
Jpm3=[-a2*c2*s1 -a2*s2*c1 0 0;a2*c2*c1 -a2*s2*s1 0 0;0 a2*c2 0 0];
Jpm4=[-a2*c2*s1-d4*c32*s1 -a2*s2*c1-d4*s32*c1 -d4*s32*c1
0;a2*c2*c1+d4*c32*c1 -a2*s1*s2-d4*s32*s1 -d4*s32*s1 0;0 a2*c2+d4*s32
d4*s32*c1 0];

Jom1=[0 0 0 0;0 0 0 0;Kr1 0 0 0];
Jom2=[0 Kr2*s1 0 0;0 -Kr2*c1 0 0;1 0 0 0];
```

```
Jom3=[0 s1 Kr3*s1 0;0 -c1 -Kr3*c1 0;1 0 0 0];
Jom4=[0 s1 s1 Kr4*c1*c2*c3+Kr4*c1*s2*c3;0 -c1 -c1
Kr4*s1*c2*s3+Kr4*s1*s2*c3;1 0 0 Kr4*s2*s3-Kr4*c2*c3];

% inertia tensor
symsIL1xxIL1xyIL1xzIL1yyIL1yzIL1zzIL2xxIL2xyIL2xzIL2yyIL2yzIL2zzIL3x
xIL3xyIL3xzIL3yyIL3yzIL3zzIL4xxIL4xyIL4xzIL4yyIL4yzIL4zz;
IL1=[IL1xx -IL1xy -IL1xz;-IL1xy IL1yy -IL1yz;-IL1xz -IL1yz IL1zz];
IL2=[IL2xx -IL2xy -IL2xz;-IL2xy IL2yy -IL2yz;-IL2xz -IL2yz IL2zz];
IL3=[IL3xx -IL3xy -IL3xz;-IL3xy IL3yy -IL3yz;-IL3xz -IL3yz IL3zz];
IL4=[IL4xx -IL4xy -IL4xz;-IL4xy IL4yy -IL4yz;-IL4xz -IL4yz IL4zz];
symsIm1xxIm1xyIm1xzIm1yyIm1yzIm1zzIm2xxIm2xyIm2xzIm2yyIm2yzIm2zzIm3x
xIm3xyIm3xzIm3yyIm3yzIm3zzIm4xxIm4xyIm4xzIm4yyIm4yzIm4zz;
Im1=[Im1xx -Im1xy -Im1xz;-Im1xy Im1yy -Im1yz;-Im1xz -Im1yz Im1zz];
Im2=[Im2xx -Im2xy -Im2xz;-Im2xy Im2yy -Im2yz;-Im2xz -Im2yz Im2zz];
Im3=[Im3xx -Im3xy -Im3xz;-Im3xy Im3yy -Im3yz;-Im3xz -Im3yz Im3zz];
Im4=[Im4xx -Im4xy -Im4xz;-Im4xy Im4yy -Im4yz;-Im4xz -Im4yz Im4zz];

% calculate inertia matrix B
symsmL1mL2mL3mL4mm1mm2mm3mm4
B1=mL1*transpose(JpL1)*JpL1+transpose(JoL1)*IL1*JoL1+mm1*transpose(J
pm1)*Jpm1+transpose(Jom1)*Im1*Jom1;
B2=mL2*transpose(JpL2)*JpL2+transpose(JoL2)*IL2*JoL2+mm2*transpose(J
pm2)*Jpm2+transpose(Jom2)*Im2*Jom2;
B3=mL3*transpose(JpL3)*JpL3+transpose(JoL3)*IL3*JoL3+mm3*transpose(J
pm3)*Jpm3+transpose(Jom3)*Im3*Jom3;
B4=mL4*transpose(JpL4)*JpL4+transpose(JoL4)*IL4*JoL4+mm4*transpose(J
pm4)*Jpm4+transpose(Jom4)*Im4*Jom4;
B=B1+B2+B3+B4;
b11=B(1,1);
b12=B(1,2);
b13=B(1,3);
b14=B(1,4);
b21=B(2,1);
b22=B(2,2);
b23=B(2,3);
b24=B(2,4);
b31=B(3,1);
b32=B(3,2);
b33=B(3,3);
b34=B(3,4);
b41=B(4,1);
b42=B(4,2);
b43=B(4,3);
b44=B(4,4);

% equation of motion
diff_theta1=diff(B,1,theta1);
diff_theta2=diff(B,1,theta2);
diff_theta3=diff(B,1,theta3);
diff_theta4=zeros(size(B));

mL=[mL1 mL2 mL3 mL4];
mm=[mm1 mm2 mm3 mm4];
symsg;
```

```
g0=[0 0 g*cos(theta3) g*cos(theta3)*cos(theta4);0 g 0
g*sin(theta4);g 0 g*sin(theta3) g*sin(theta3)];
JpL=[JpL1 JpL2 JpL3 JpL4];
Jpm=[Jpm1 Jpm2 Jpm3 Jpm4];

ddsigma=0;
ssigma=0;
gsigma=0;


fori=1:4
for j=1:4
for k=1:4
if k==1
dqk=diff_theta1;
elseif k==2
dqk=diff_theta2;
elseif k==3
dqk=diff_theta3;
elseif k==4
dqk=diff_theta4;
end
ifi==1
dqi=diff_theta1;
elseifi==2
dqi=diff_theta2;
elseifi==3
dqi=diff_theta3;
elseifi==4
dqi=diff_theta4;
end
            h=dqk(i,j)-1/2*dqi(j,k);
            sum1=h*thetad(k)*thetad(j);
end

ddsigma=ddsigma+sum1;
ssigma=ssigma+B(i,j)*thetadd(j);
      gsigma=gsigma+mL(j)*transpose(g0(:,i))*JpL(:,4*(j-
1)+i)+mm(j)*transpose(g0(:,i))*Jpm(:,4*(j-1)+i);
end
torque(i)=simplify(ddsigma+ssigma+gsigma);

end

% test torque
L1=1; L2=1; L3=1; L4=1;Kr1=1; Kr2=1; Kr3=1; Kr4=1;theta1=0;
theta2=30*pi/180; theta3=45*pi/180; theta4=0;g=10;a2=1;d4=1;
theta1d=0; theta2d=0.1; theta3d=0.2; theta4d=0;theta1dd=0;
theta2dd=0.6; theta3dd=0.4; theta4dd=0;
IL1xx=1; IL1xy=1; IL1xz=1; IL1yy=1; IL1yz=1; IL1zz=1; IL2xx=1;
IL2xy=1; IL2xz=1; IL2yy=1; IL2yz=1; IL2zz=1; IL3xx=1; IL3xy=1;
IL3xz=1; IL3yy=1; IL3yz=1; IL3zz=1; IL4xx=1; IL4xy=1; IL4xz=1;
IL4yy=1; IL4yz=1; IL4zz=1;
Im1xx=1; Im1xy=1; Im1xz=1; Im1yy=1; Im1yz=1; Im1zz=1; Im2xx=1;
Im2xy=1; Im2xz=1; Im2yy=1; Im2yz=1; Im2zz=1; Im3xx=1; Im3xy=1;
```

```
Im3xz=1; Im3yy=1; Im3yz=1; Im3zz=1; Im4xx=1; Im4xy=1; Im4xz=1;
Im4yy=1; Im4yz=1; Im4zz=1;
mL1=1;mL2=1;mL3=1;mL4=1;mm1=1;mm2=1;mm3=1;mm4=1;
testtorque=subs(torque);
% testtorque(:,1)=testtorque;
T1=testtorque(1);
T2=testtorque(2);
T3=testtorque(3);
T4=testtorque(4);
```