A DUAL HOIST ROBOT CRANE FOR LARGE AREA SENSING

A Thesis
Presented to
The Academic Faculty

by

John A. Harber

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
George W. Woodruff School of Mechanical Engineering

Georgia Institute of Technology
May 2016

A DUAL HOIST ROBOT CRANE FOR LARGE AREA SENSING

Approved by:

Professor William Singhose, Advisor
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Professor Kok-Meng Lee
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Professor Jonathan Rogers
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: 22 April 2016

To my family,

without whose support and encouragement I never would have made it this far.

ACKNOWLEDGEMENTS

First and foremost I would like to thank my advisor, Dr. Singhose. His guidance, optimism, and belief in me, which often exceeded my own, were vital to the completion of this thesis. I would also like to thank my parents and siblings for their love and support, I never could have done it without you. Finally, a big thank you to my friends, who made my time here at Tech so enjoyable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Cranes are used to lift and move large objects in a wide variety of applications at constructions sites, shipping ports, and manufacturing facilities, etc. If the load to be moved is too long or heavy for a single crane, then two, or more, cranes must work in cooperation to move the payload. In a factory setting this can be accomplished using two trolleys running along the same bridge forming a dual hoist crane. Using two hoists not only increases lifting capacity, it also improves stability of the payload over traditional single hoist configurations. This research takes advantage of that increased stability and explores a novel application for dual hoist cranes: suspending a robot arm from the two trolleys. This increases the workspace of the robot to the entirety of the space covered by the crane, opening up numerous applications not possible with a stationary robot. In order to better understand and characterize the dynamics of the system, a numerical model was developed and tested against a physical system to confirm its validity.

A vision system has the potential to greatly increase the usefulness of a robotic system such as the one presented in previous paragraph. The Asus Xtion was used in this work due to its versatility and low cost. An evaluation of this sensor was performed. Various tests were conducted to determine its accuracy in a range of scenarios. It was found that crane oscillations degraded the quality of data returned. This is effect is especially detrimental if the crane is moved to a specified point and sensing begins immediately. The data collection process could be delayed until the residual oscillations subside, however the time penalty incurred by waiting is large because the oscillations are lightly damped and have a long period. To address this issue a control method called input shaping was introduced to reduce the residual

oscillations thereby increasing the quality of the sensor data.

Finally, two promising uses of the robot arm dual-hoist crane system were introduced: painting and sandblasting. The efficiency of a factory equipped with this system can be increased at relatively low cost by automating manual tasks such as these.

CHAPTER I

INTRODUCTION

Cranes are used in construction sites, nuclear facilities, manufacturing plants, warehouses, and in many other locations where large and heavy objects must be moved. Two cranes are used in tandem if the item to be moved exceeds the weight or geometry constraints of a single crane. If the lift is properly executed, then a dual crane system cannot only lift more weight, but also provides more stability to the payload being positioned than a single crane. In a manufacturing or warehouse setting a dual hoist setup can be accomplished by running two trolleys on the same bridge. A photograph of such a dual-hoist bridge crane is shown in Figure 1.

Figure 1: Two-Ton Dual-Hoist Bridge Crane

The research in this thesis takes advantage of the increased stability of a dual hoist bridge crane and explores a novel application: mounting a robotic arm on a platform suspended between the crane's two hooks. A photograph of such a robot crane is shown in Figure 2. This dramatically increases the workspace of the robot. However, it also adds increases the dynamic complexity of the robotic system which can lead to large transient and residual oscillations.

Figure 2: Two-Ton Dual-Hoist Bridge Crane with Robot.

To gain a better understanding of the system's dynamics and to aid in testing, a numerical model of the robot crane system was developed. A sufficiently accurate model can help with testing by decreasing the time it takes to test different system configurations and controllers.

The next step after completing the model was to choose a vision sensor for the

system. Color vision systems can be used for inspection purposes when inconsistencies and imperfections in the inspected surfaces are based on color. If sensing of objects in three dimensions is required or if workspace mapping is necessary, infrared (IR) sensors can be used. These sensors project an infrared grid onto an object and determined its 3D characteristics by examining the distortion of the grid. The Asus Xtion was chosen due to its capabilities as both a vision and depth sensor and its low cost. This gives the system useful sensing capabilities as well as the possibility to incorporate workspace mapping and obstacle detection and avoidance. Because the sensor is to be mounted on a highly flexible system, it is essential to determine the quality of the sensor data if the system is oscillating. This was found to be dependent upon the magnitude of the oscillation of the robot crane system. These fluctuations in accuracy were explored and characterized and input shaping was used to decrease residual vibration, thereby increasing sensor data quality.

Two additional applications of this system are also considered, painting and sand-blasting. Painting and sanding of large areas such as airplanes and ships is often done by hand and is time intensive. Automation of these tasks has the potential do greatly increase the efficiency of such operations.

1.1 Dual Hoist Crane Model

An important step in understanding the dynamics of a complex system is creating a numerical model. Ower et. al [10] explored a Lagrangian approach to model a two-link flexible manipulator, but Lee [16] showed that the Lagrangian approach is not fully equipped to model flexible links as it allows elongation of the links. Rosado et. al [28] used the Newton-Euler approach, again for a two link flexible manipulator, with satisfactory results. Meghdara et. al [1] used Kane's multi body method to decouple the equations of motion of a robotic arm. Using a similar method, Maleki [8] and Miller [2] developed a numerical model for a dual hoist crane carrying rigid

body payloads with centered and offset moments of inertia, respectively. [5] explored the modeling and control of a robotic manipulator mounted on an omnidirectional platform to position heavy payloads on the deck of a ship. Kane's method and the model structure developed by Maleki were used as a starting point to create the model developed in this thesis.

1.2 Input Shaping

The robot crane system is highly flexible and requires an effective controller to reduce the residual oscillations. Input shaping is introduced as a viable controller option due to its ease of implementation, effective control of flexible systems, and high robustness to modeling uncertainty and parameter variations. Unlike some other control methods that require extensive system models and numerical computation, only the natural frequency and damping ratio of the system are needed.

Starr [24] introduced a method to reduce the payload swing while transporting suspended objects with a path controlled robot manipulator. He derived a zero vibration (ZV) input shaper and achieved a large reduction in residual oscillation. Singhose et. al [30] presented a vector diagram method to derive input shapers. This method uses the fact that any two vectors can be effectively cancelled by a strategically-placed third vector. This idea can be used to chose the impulses of an input shaper; three inputs, if properly selected, can result in very little system vibration. If some level of system vibration is acceptable, a more robust shaper can be designed.

Singer et. al [20] [19] explored the effects of input shaping to reduce residual vibration of gantry crane payloads. Two shapers were considered, a Unity Magnitude Fixed Duration (UM FD) and a Unity Magnitude Zero Vibration (UM ZV) shaper. The procedure used to design these shapers took into account the properties of a gantry crane such as, the single-mode dynamics, the known frequency range, and standard acceleration period. The UM FD shaper exhibited lower residual vibrations

than the UM ZV shaper. The UM FD shaper also maintained a constant deceleration period over all cable lengths, whereas the UM ZV shapers acceleration periods varied with cable length. A control mode incorporating input shaping with small step inputs to precisely place a payload that is already near its final destination without sway, was also introduced.

Chu et. al considered a dynamically complex scenario: control of cranes mounted on boats. This configuration adds a degree of complexity to the system because, in addition to the flexible dynamics of the swinging payload, the whole ship is moving in response to waves, currents, etc. Specifically, a three joint boom crane was examined. The existing control method on this crane required the operator to individually control each link of the crane and did not include input shaping or disturbance rejection of any kind. The proposed method included input shaping and only required the operator to control the position of the payload. In order to bridge the gap between controlling the individual booms and just the tip of the crane the Denavit-Hartenberg (D-H) method, a classical way to solve kinematic chains, was used. The Jacobian was then used to relate the joint angular velocities to Cartesian velocities of the arm tip. The Bond Graph (BG) method was used to model the system. The BG method is a highly modular energy-based approach for modeling and simulation of multi-domain dynamic systems. 20-Sim was also used to implement a drivable 3-D Model.

Singhose et. al [22] studied the effects of input shaping on trajectory following in two dimensions. Two types of shapers were used, Zero Vibration Zero Derivative (ZVD) and Extra-Insensitive (EI). The modeling and simulation showed that the EI controller performed better over a larger range of frequencies than the ZVD controller. A flexible gantry with two degrees of freedom was constructed with an airbrush mounted on top. A command was then sent to the device and the actual path traveled was recorded by the pattern painted by the airbrush. This was done both with and without input shaping. It was found that input shaping considerably

increased the accuracy of the tracking.

Han et. al [11] explored an input shaping technique for under-damped systems that only requires the signals of the systems outputs. Because this method only uses the system output and does not rely on an accurate model of the system, problems arising from model parameter uncertainty are avoided. Simulation and experimental results showed that input shapers designed using this method can effectively eliminate vibrations in the system output within the desired response time. Although input shapers were developed from linear system theory, they could also be used to control nonlinear systems.

Smith et. al [12] explored several techniques to design input shapers for nonlinear systems and presented an effective nonlinear ZV shaper. This ZV shaper was calculated using an energy-based method and, while its scope is somewhat limited due to assumptions and conditions of the system, it effectively cancels residual vibrations. Chen et. al [15] devised a general input shaper design methodology for single degree of freedom nonlinear systems was successfully devised using an energy approach. Following this method, two-step and three-step shapers are developed, which in the linear limit reduce to the traditional zero-vibration and zero-vibration-and-derivative shaper respectively. The robustness of these nonlinear shapers was investigated numerically through several case studies and the results showed that the three-step was sufficiently robust to resist significant amounts of parameter variations without exciting significant residual vibrations. An improvement in vibration suppression was noted in comparison to traditional linearized ZV and ZVD shapers.

1.3 3D Sensing

Microsoft Kinect and Asus Xtion sensors, developed by PrimeSense, have recently become popular in the field of robotic vision due to their low cost and depth sensing capabilities. A photograph of the sensor used in this thesis is shown in Figure 3.

Laser scanners have traditionally been the vision sensor of choice for many vision type applications such as robust map building, localization, and collision avoidance. However, they are more expensive than sensors like the Microsoft Kinect and Asus Xtion. Zug et. al [21] compared the PrimeSense sensors with a laser scanner to determine if they could replace the laser scanner for these applications. Mapping, localization, and collision avoidance were all tested with both sensors and the results of the tests were analyzed. It was found that due to the smaller monitoring angle, the PrimeSense sensors did not perform as well in mapping and localization scenarios. However, they performed better than the laser scanner in obstacle avoidance in this test.

Figure 3: Asus Xtion Vision Sensor

Karen [13] [14] and Gonzalez et. al [9] further explored the accuracy and precision

of the PrimeSense sensors, as well as methods of improving them through calibration. [27] presents a technique for three-dimensional camera calibration using standard cameras and lenses. It is a two stage process designed to increase the computational efficiency of calibrating external position and orientation relative a reference coordinate system. It was found that the factory calibration that comes loaded on the sensor is effective as long as the measurement range is less than 2 meters and the accuracy requirements are not very strict. However, if greater range or accuracy is required, a simple checkerboard calibration can improve the accuracy by three to five times. It was also discovered that the sensor errors could be reduced by an additional 50 percent if depth measurement correction was implemented. Testing of the combined calibration yielded a level of precision comparable to a well calibrated RGB camera.

1.4 Thesis Contributions

This thesis makes contributions to understanding the dynamics of a dual-hoist crane with an active payload, as well as sensor integration on such a system. The main contributions of this thesis are:

1. Numerical modeling of a dual-hoist robot crane.
2. Experimental Verification of Model
3. Integration of the robot, crane, and sensors.
4. Evaluation of sensing system.

1.5 Thesis Outline

Chapter 2 presents the robot crane system used in this thesis. Then, a numerical model is derived and presented. An energy balance is used to check if the model is viable. Chapter 3 provides an experimental verification of the model, comparing

the simulation to the data gathered with the physical system. Chapter 4 investigates integration the vision sensor. The accuracy of the sensor is examined, as well as some ways to improve the accuracy. Chapter 5 explores future work and system applications. Finally, Chapter 6 presents the conclusions.

CHAPTER II

NUMERICAL MODELING OF A CRANE MOUNTED ROBOT

2.1 Mechanical Structure

This chapter presents a numerical simulation for the dynamic response of a dual hoist crane with a PRR robot mounted on a platform suspended between the hooks. Figure 4 shows the crane and robot used in this research. It is a two-ton dual hoist crane located in the Georgia Tech Manufacturing Institute.

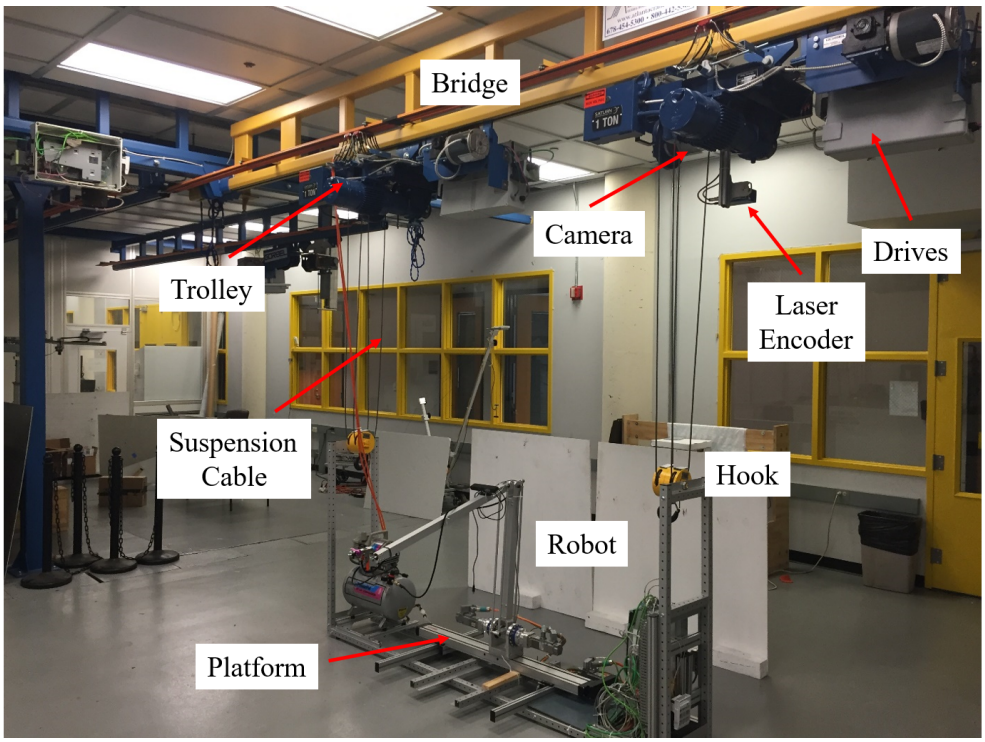

Figure 4: Two-Ton Dual Hoist Bridge Crane with Robot.

The two trolleys move along the bridge either independently or in unison separated by a fixed distance. The bridge is the yellow truss structure shown along the top of Figure 4. The bridge moves perpendicular to the trolleys along two rails mounted on the ceiling on either side of the room. One of the two rails is the blue truss structure shown along the top-left side of Figure 4. Each hoist is capable of lifting one ton. The entire system is controlled using Siemens Simotion drives and PLCs. Laser encoders are used to determine the position of the bridge and trolleys in the workspace, and downward pointing overhead cameras mounted to the trolleys are used to measure the deflection of the hooks.

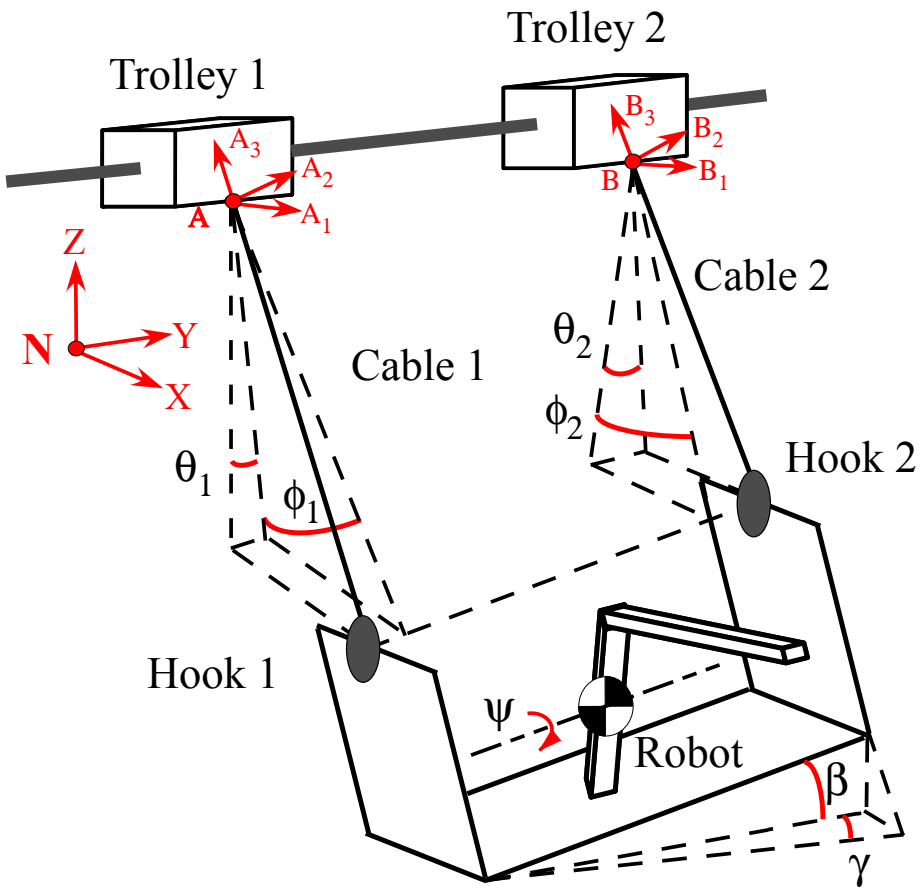

Figure 5: Crane Schematic with Angle Variables Labeled.

Figure 5 presents a schematic representation of the crane with angle variables labeled. θ denotes the cable angles parallel to the bridge, ϕ denotes the cable angles perpendicular to the bridge, and β, γ, and ψ are roll, yaw, and pitch of the robot, respectively. Figure 6 shows the labeled components of the robot. The robot has a unique configuration that allows lighter materials to be used while maintaining structural stability. The motor driving the second link is located in the base instead of at the joint between links 1 and 2 as in a conventional setup. This minimizes the amount of moving mass at the end of link 1 which reduces residual oscillations.

The frame serves as a mounting location for the robot, motor drives, and air compressor, as well as providing connection points to attach to the crane's hooks. The air compressor is used to run the air-brush end effector pictured. Figure 7 shows the coordinate axes and important variables associated with the robot. qLink1 is the angle of the first link of the robot with respect to the vertical axis and qLink2 is the angle of the second link with respect to the first link. Coordinate frames C, D, and F are the local coordinate frames of the first and second link of the robot and the structure on which the robot is mounted, respectively.

2.2 Numerical Simulation

In the Newtonian frame the trolleys move in the y-direction along the bridge and the bridge moves in the x-direction, perpendicular to trolley motion. A multi-body dynamic approach was used to obtain the equations of motion of the system. The trolleys were assumed to be massless points, the cables were modeled as massless rigid bodies, and the hooks were modeled as point masses. The platform on which the robot is mounted was modeled as a rigid body mass with inertial properties and the robot's base was constrained to translate parallel to it.

The trolleys move relative to the Newtonian frame governed by:

$$\vec{P}_{Trolley1/N} = x_1\vec{N}_x + y_1\vec{N}_y \tag{1}$$

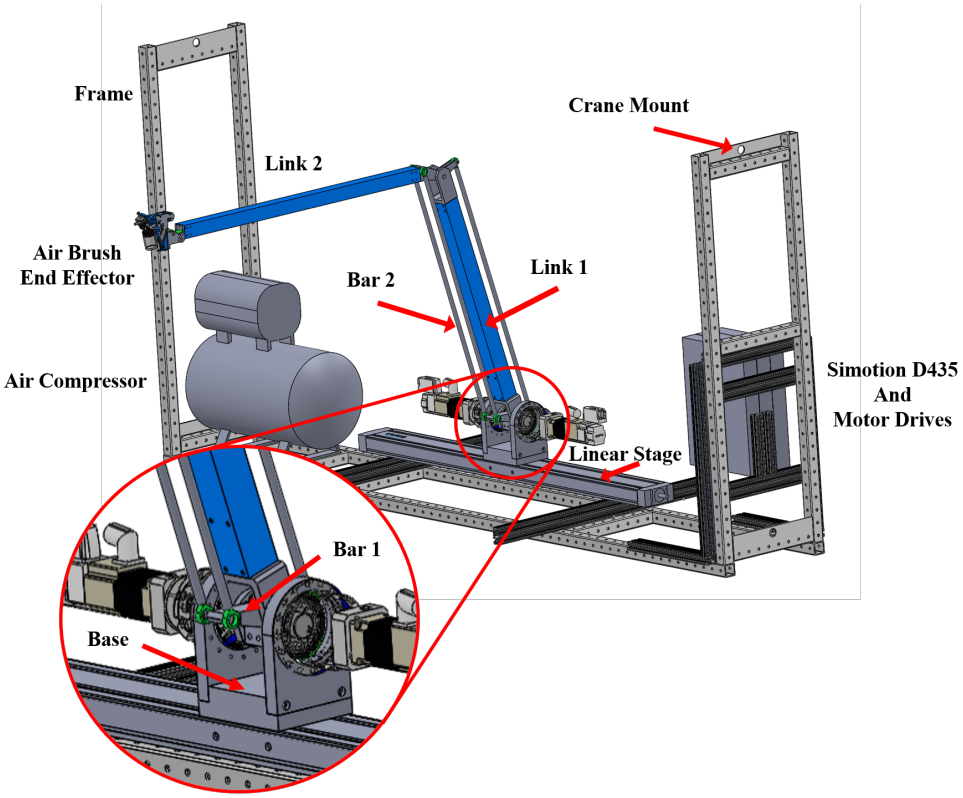

Figure 6: Labeled Robot Components.

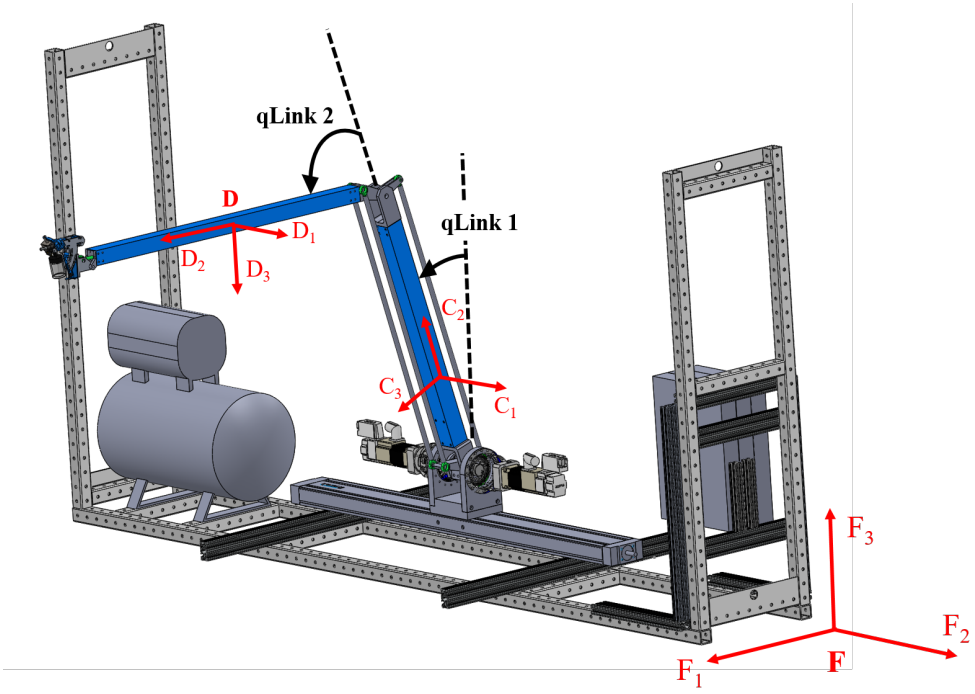

Figure 7: Robot with Variables and Axes Labeled.

$$\vec{P}_{Trolley2/N} = x_2\vec{N}_x + y_2\vec{N}_y \tag{2}$$

Movement of the bridge is simulated by setting x_1 equal to x_2 thereby constraining the trolleys to move in unison in the bridge direction. The hoisting cables are attached to the trolleys with lengths of L_1 and L_2 corresponding to Trolley 1 and Trolley 2, respectively. The cables rotate with respect to the Newtonian frame according to:

$$\vec{\omega}_{Cable1/N} = \dot{\theta}_1\vec{N}_x + \dot{\phi}_1\vec{D}_{1y} \tag{3}$$

$$\vec{\omega}_{Cable2/N} = \dot{\theta}_2\vec{N}_x + \dot{\phi}_2\vec{D}_{2y} \tag{4}$$

where $\vec{D}_1$ and $\vec{D}_2$ are intermediate frames of Cable 1 and Cable 2, respectively. Damping, if present, is added here in the form of a resistive torque. The robot is mounted to the hooks and its rotation is described by:

$$\vec{\omega}_{Payload/N} = \dot{\beta}\vec{N}_x + \dot{\psi}\vec{A}_y + \dot{\gamma}\vec{B}_z \tag{5}$$

where $\vec{A}$ and $\vec{B}$ are intermediate frames produced by each successive rotation. The system can be thought of as four-bar linkage that allows a set of velocity constraints to be introduced. These equations are found by differentiating the loop equations found from sequentially moving to the position of the corner points of the four bar linkage to complete a closed chain:

$$\frac{d}{dt}[x_1\vec{N}_x + y_1\vec{N}_y - L_1\vec{C}_{1z} + L\vec{F}_y + L_2\vec{C}_{2z} - x_2\vec{N}_x - y_2\vec{N}_y] \bullet \vec{N}_x = 0 \tag{6}$$

$$\frac{d}{dt}[x_1\vec{N}_x + y_1\vec{N}_y - L_1\vec{C}_{1z} + L\vec{F}_y + L_2\vec{C}_{2z} - x_2\vec{N}_x - y_2\vec{N}_y] \bullet \vec{N}_y = 0 \tag{7}$$

$$\frac{d}{dt}[x_1\vec{N}_x + y_1\vec{N}_y - L_1\vec{C}_{1z} + L\vec{F}_y + L_2\vec{C}_{2z} - x_2\vec{N}_x - y_2\vec{N}_y] \bullet \vec{N}_z = 0 \tag{8}$$

where $\vec{C}_1$, $\vec{C}_2$, and $\vec{F}$ are the local coordinate axes of Cable 1, Cable 2, and the robot's frame, respectively.

The base of the robot and the first link are directly driven by motors that are modeled as inputs to the system. The second link is indirectly driven by a four-bar linkage. The same method for determining the velocity constraints of the crane

was also used for the robot's four-bar linkage. Only two differentiations of the loop equation are needed because rotational joints constrain the robot arms to move in only one plane:

$$\frac{d}{dt}[D_{1R}\vec{L}_{2Ry} - B_{2R}\vec{B}_{2Ry} - B_{3R}\vec{B}_{3Ry} + L_{1R}\vec{L}_{1Ry}] \bullet \vec{L}_{1Ry} = 0 \tag{9}$$

$$\frac{d}{dt}[D_{1R}\vec{L}_{2Ry} - B_{2R}\vec{B}_{2Ry} - B_{3R}\vec{B}_{3Ry} + L_{1R}\vec{L}_{1Ry}] \bullet \vec{L}_{1Rz} = 0 \tag{10}$$

Table 1 lists the crane's velocity and acceleration limits used to model the system. The values were obtained by testing the actual limits of the physical system [17] [18]. Table 2 shows the parameters for the robot.

Table 1: Dual-Hoist Crane Parameters

Parameter	Value
Hook Mass	7.65 kg
Max Trolley Acceleration and Velocity	1 m/s^2 and 0.33 m/s
Max Bridge Acceleration and Velocity	1 m/s^2 and 0.33 m/s

Table 2: Robot Parameters

Parameter	Value
Max Base Acceleration and Velocity	1 m/s^2 and 0.1 m/s
Max Link 1 Acceleration and Velocity	360 deg/s^2 and 5 deg/s
Max Bar 3 Acceleration and Velocity	360 deg/s^2 and 5 deg/s

The equations of motion were found using MotionGenesis [7], a dynamics modeling software package. Inputs to the model are the acceleration commands to the motors driving the trolleys, bridge, and robot, as well as the initial condition of the system. The MotionGenesis code is included in Appendix A, and the equations of motion output by this code are included in Appendix B. In the next section, some important model outputs are examined to provide a check to ensure the model behaves as expected.

2.3 Energy Balance Check of Simulation

The first check that was performed to ensure the model made physical sense was an energy balance with the crane in a symmetrical configuration. In order for the model to be correct, there should be no increase of energy in the system provided no energy is added from outside sources. To observe whether or not this is the case, the simulation was run with initial cable angles of 10 degrees in the θ direction for both suspension cables. The simulation was allowed to oscillate for 35 seconds. Both kinetic and potential energies were plotted over this time period in Figure 8.

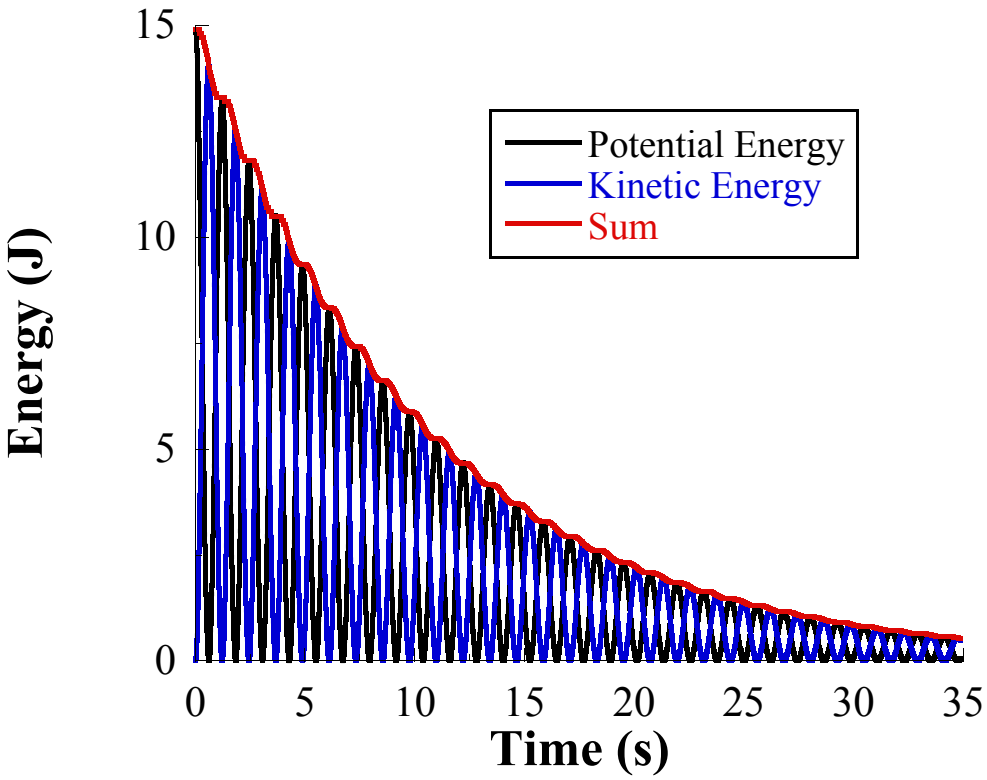

Figure 8: Potential and Kinetic Energy for a Move in the Trolley Direction

Energy is lost through the resistive forces resulting from damping in the system. These resistive losses are directly related to the velocity of the system, and therefore

the kinetic energy as well. No energy is lost when the velocity of the system is zero, causing the plateaus in the sum of kinetic and potential energy. Figure 9 shows a closer view of the first five seconds of the response, along with the hook velocity. This clearly shows the correlation between the points of zero velocity and the plateaus in the total energy.

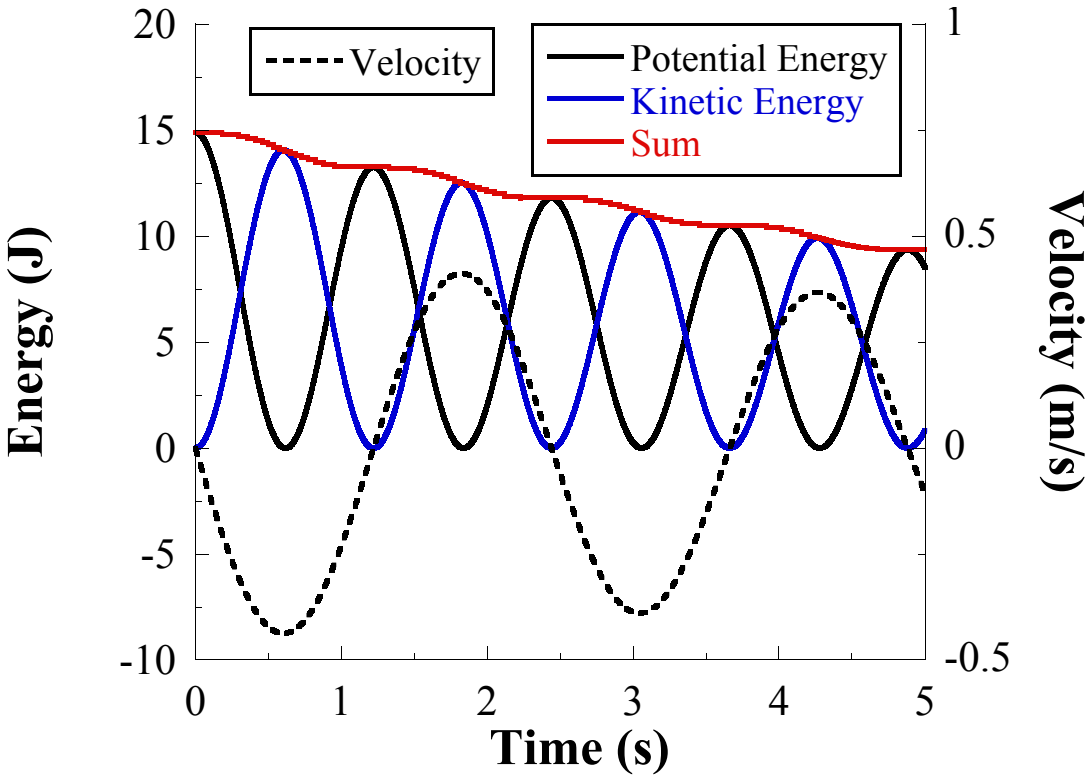

Figure 9: Energy and Velocity Comparison

As visible from the plot, the energy of the system alternates between kinetic and potential energy as the system oscillates and gradually approaches zero due to damping. This is what was expected of the model and provides confidence in its viability.

A similar test was also performed for an initial angle of 10 degrees in the ϕ direction. Unlike the θ direction, which only has one frequency, the ϕ direction

has the compounding effect of the robot pitching, as well as the payload oscillating, which introduces a second frequency. Figure 10 shows the potential energy, kinetic energy, and sum of the two for this trial. Figure 11 shows a closer view of the first five seconds of the response, with the addition of the hook velocity. The dual frequencies are clearly visible in this plot. The continual decay in the total energy again verifies that the simulation behaves in a reasonable manner.

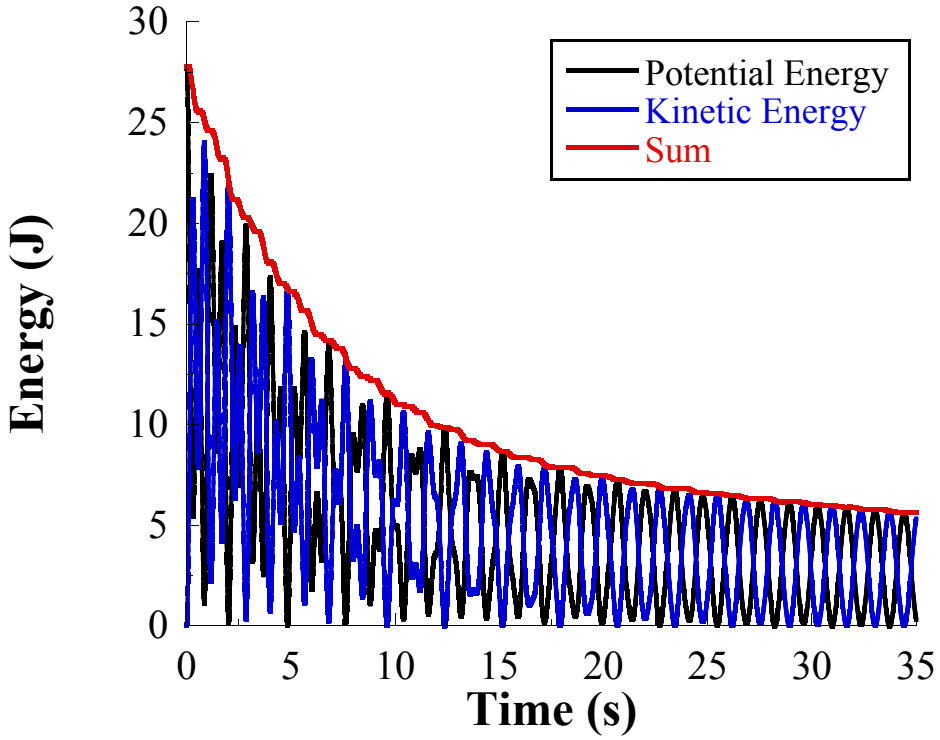

Figure 10: Potential and Kinetic Energy for a Move in the Bridge Direction

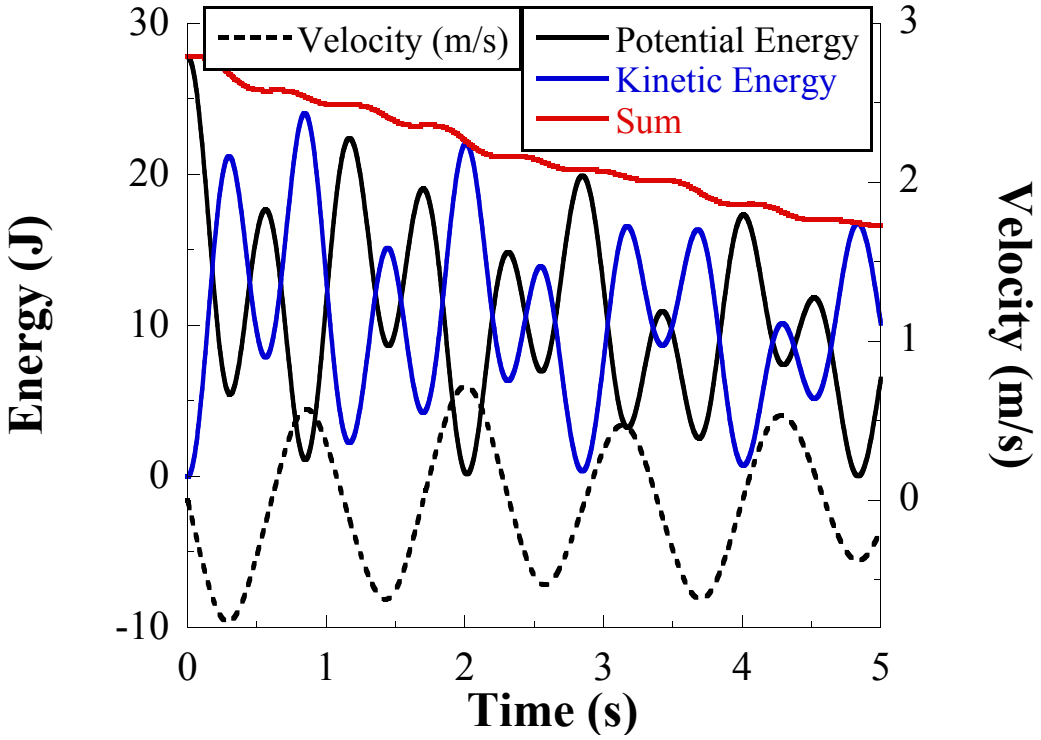

Figure 11: Energy and Velocity Comparison

2.4 Robot Crane Dynamic Behavior

2.4.1 Trolley Motion

The next test of the model was conducted to ensure its outputs were logical and made physical sense. This was achieved by running the model with various inputs and examining the results. The plots in the following sections show the output of the model for the input tests described. A brief evaluation is performed for each outcome to ensure the results align with expectations based on physical reasoning.

Figure 12 shows the crane angles, θ and ϕ, for a trolley move of 1 meter. The θ angle corresponds to the cable angle parallel to the bridge, and ϕ is the angle perpendicular to bridge, as was shown in Figure 5.

Because the motion was in the trolley direction, the large oscillations in the the θ direction that slowly diminish due to damping and near-zero activity in the ϕ direction

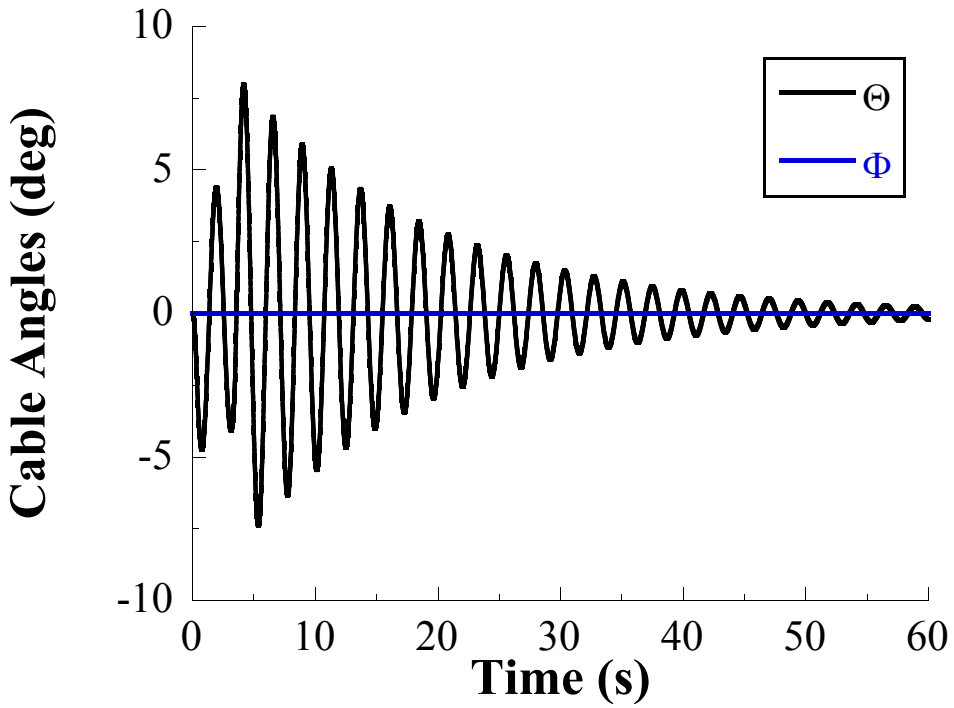

Figure 12: Trolley 1 Cable Angles After a Move in the Trolley Direction.

matches expectations. Figure 13 shows the corresponding response of the end effector of the robot. A 1 meter displacement in the y-direction (EEy) is visible, along with the residual oscillations. No change is visible in the x- and z-directions. This is as expected because the input was exclusively in the y-direction.

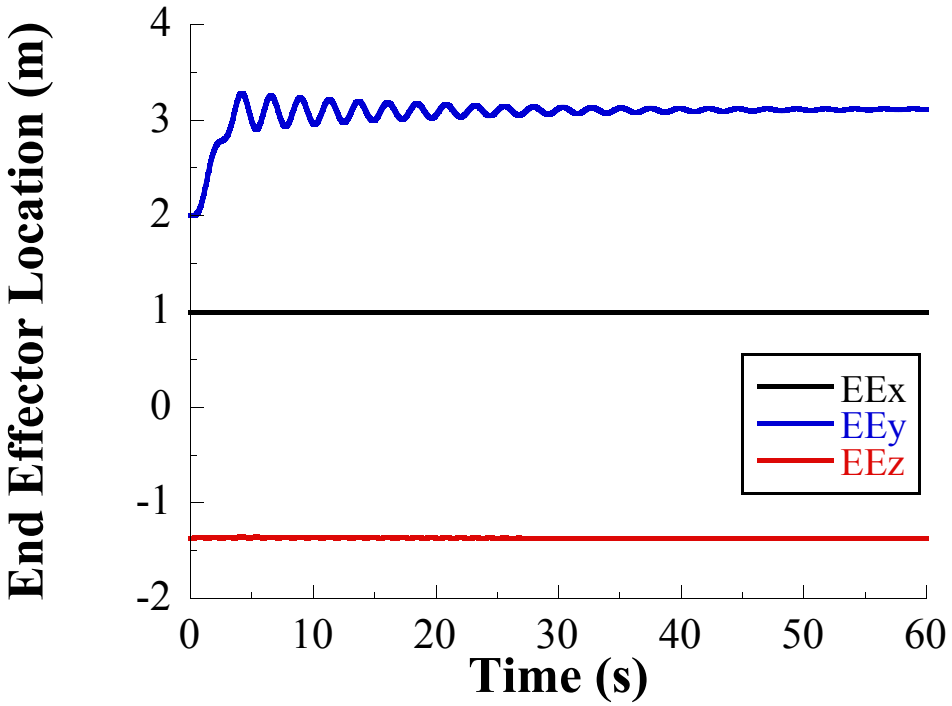

Figure 13: End Effector Location after a 1 m Move in the Trolley Direction

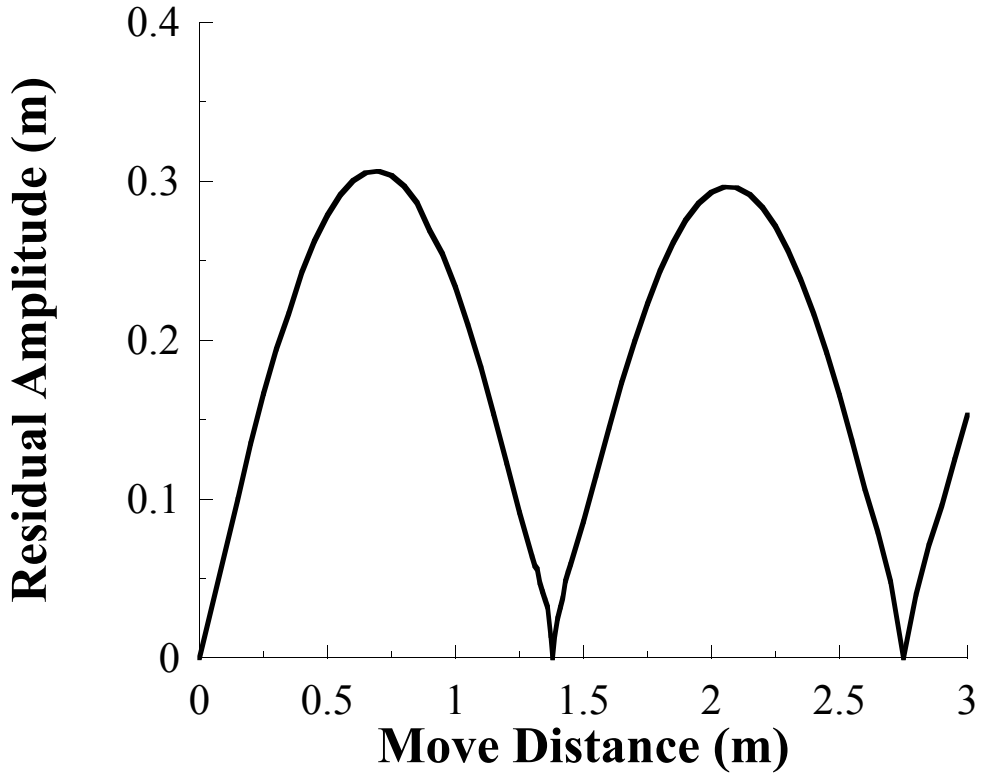

Figure 14: Amplitude vs. Move Distance for Trolley Direction Move Distances

Figure 14 shows a plot of maximum residual amplitude versus move distance in the trolley direction. From knowledge of flexible systems of this type [17], the magnitude of the residual oscillation amplitude is known to depend on move distance with some distances resulting in no residual vibration, while other move distances result in a maximum. These minimums and maximums alternate as move distance increases. This plot follows the general shape that is expected.

The effects of changing cable length were also explored. Figure 15 shows the hook response for various cable lengths for a move in the trolley direction. The response behaves as expected with residual amplitude increasing with cable length.

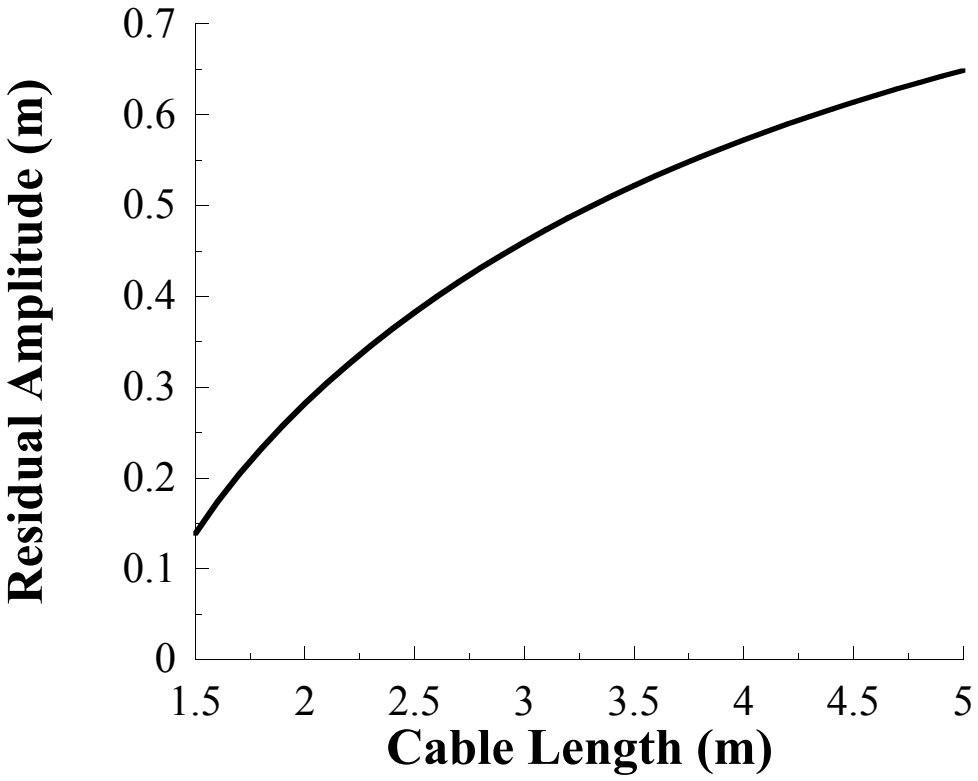

Figure 15: Hook Response for Varying Cable Lengths: Trolley Move

2.4.2 Bridge Motion

The response of the crane and angles for a move of 1.5 meters in the bridge direction is given in Figure 16. For this trial we expect large oscillations in the ϕ-direction and little activity in θ.

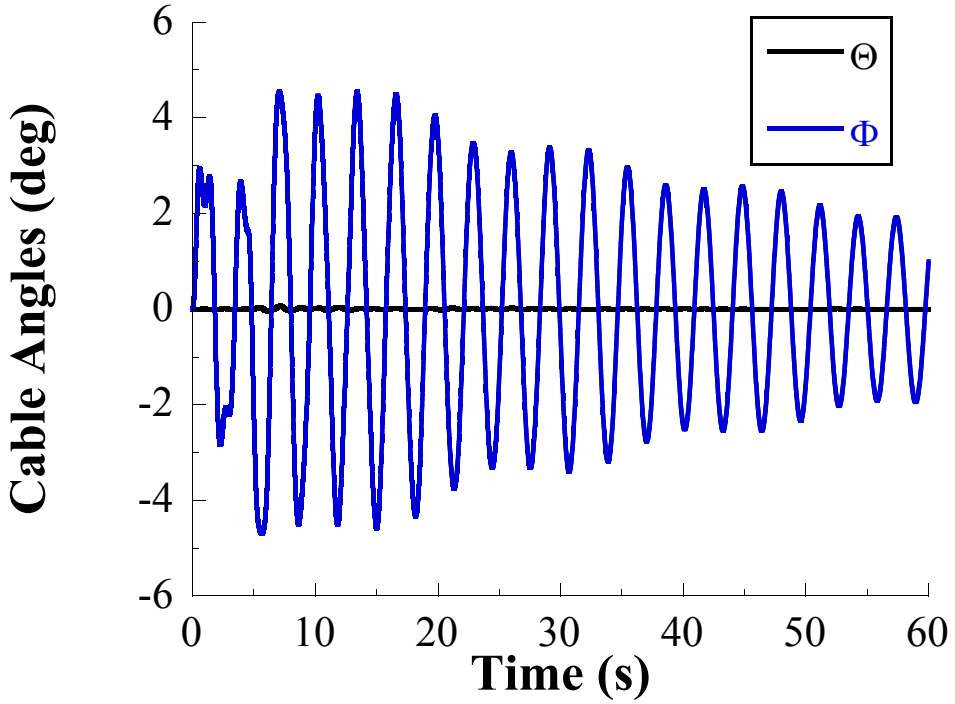

Figure 16: Cable Angles Resulting From Movement of the Bridge

Figure 17 shows the payload angles for the same move bridge move. The end effector response was also recorded and is shown in Figure 18.

The end effector displacement will be largest in the x- and z-directions due to pitching of the robot with some oscillations in the y direction. This small oscillation in the y-direction results from the yawing of the payload due to non-uniform mass distribution on the payload. The cable angles and end effector location closely followed the expected results.

A plot of residual amplitude versus move distance was also created for moves in

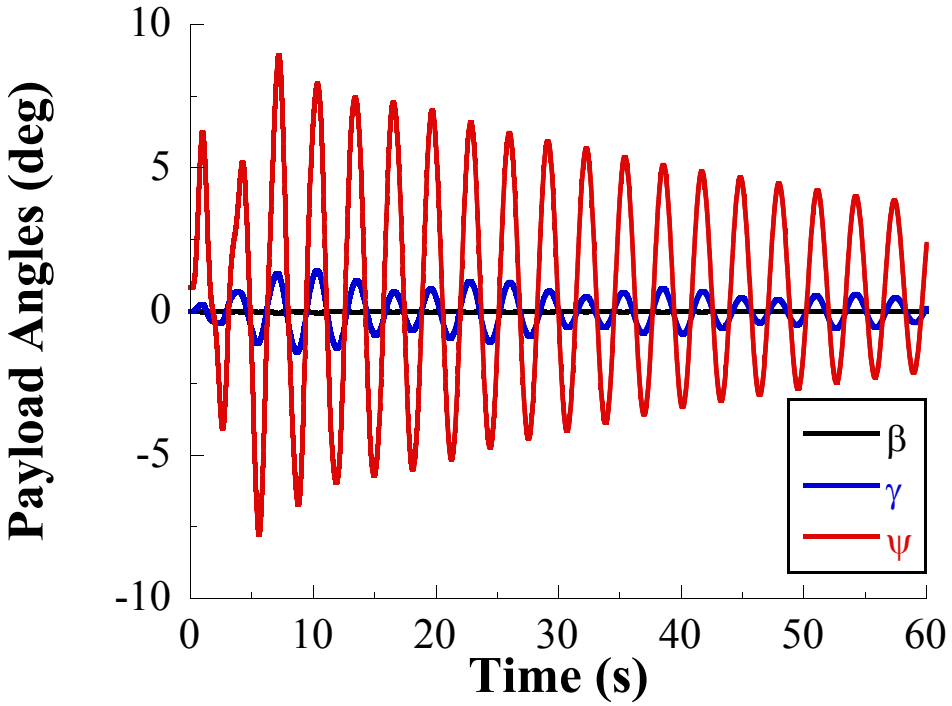

Figure 17: Payload Angles Resulting From a Move in the Bridge Direction

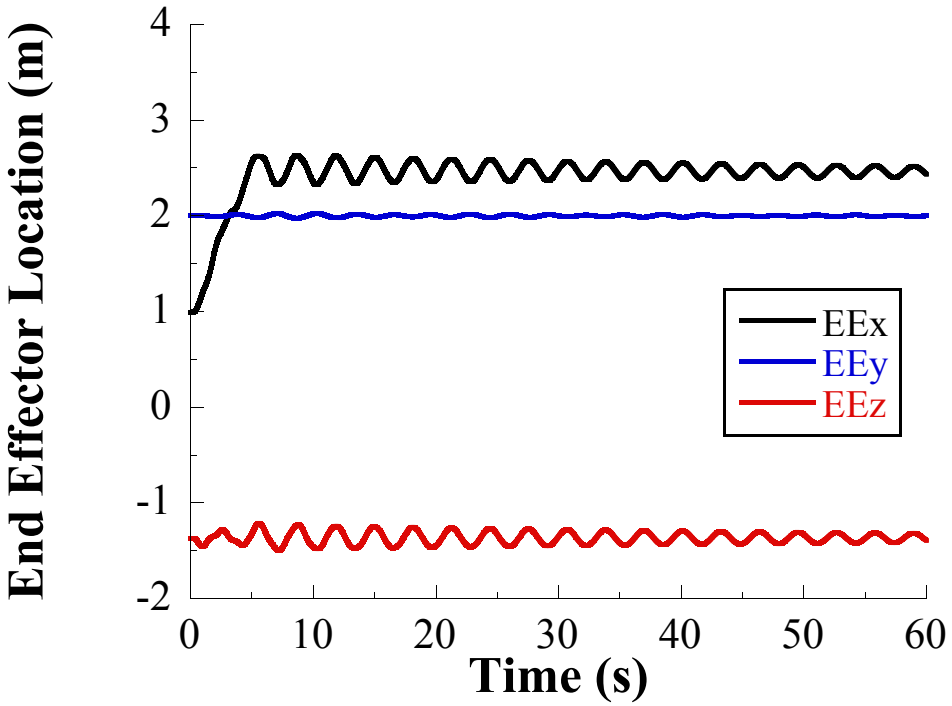

Figure 18: End Effector Location after a Move in the Bridge Direction

the bridge direction, shown in Figure 19. The significant pitching response of the robot that exists in this direction complicates the dynamics and makes the resulting performance less intuitive than for variations in trolley move distance. A comparison to the physical system is needed to ensure the results are correct and is performed in the next chapter.

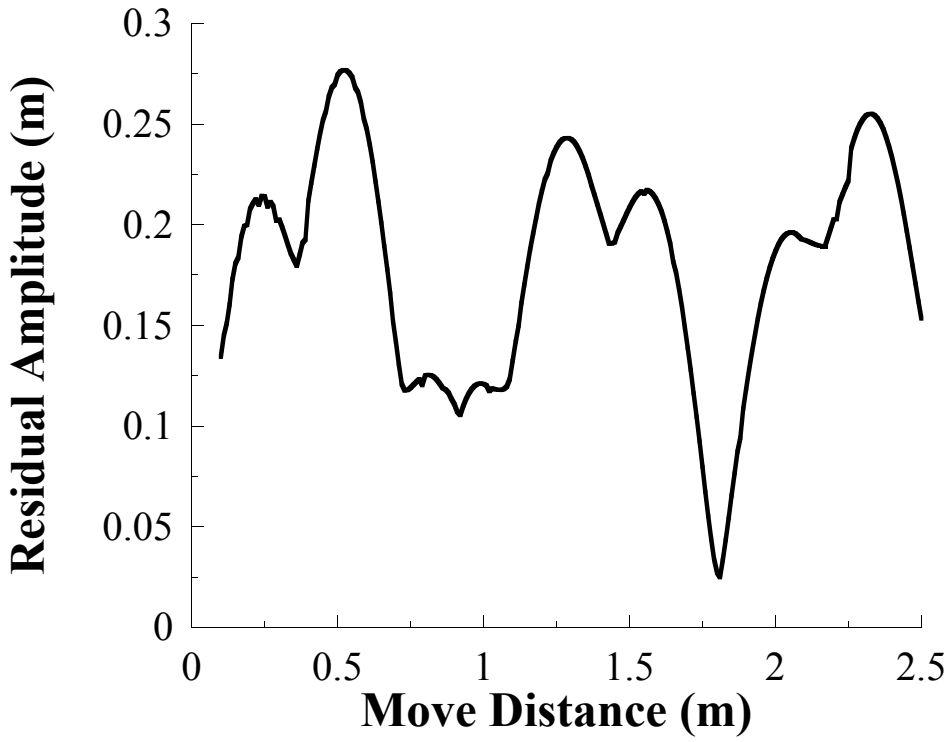

Figure 19: Amplitude vs. Move Distance for Moves in the Bridge Direction

Figure 20 shows the response of hook 1 for changing cable lengths for a move in the bridge direction. This is a straight forward response with residual amplitude increasing with cable length as expected.

2.4.3 Robot Motion

Lastly, Link1 of the robot was rotated 20 degrees and the resulting response is shown in Figure 21. The high frequency visible in the first ten seconds of data is the transient

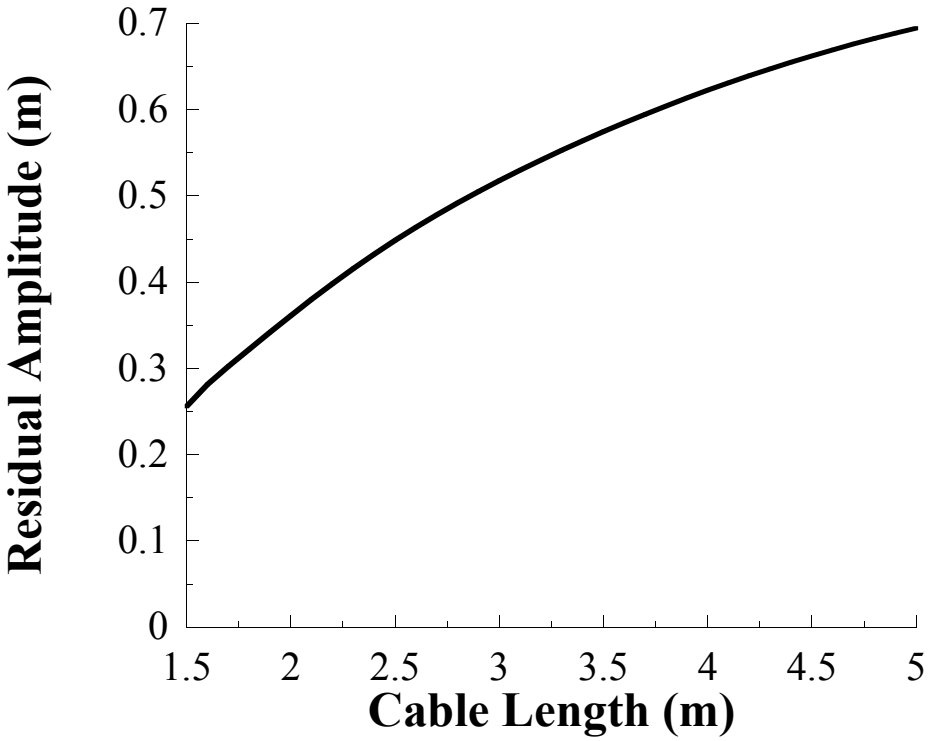

Figure 20: Hook Response for Varying Cable Lengths: Bridge Move

response induced by the rotation. A beating effect in the cable angles is apparent. This is caused by the coupling between of the platform pitch oscillation and the swinging oscillation of the entire system. The maximum deflection angle of the hooks is approximately 0.2 degrees which corresponds to an oscillation amplitude of 0.5 cm. Given the scale of the system an amplitude of this magnitude is barely noticeable and can effectively be ignored.

Figure 22, which shows the end effector response for the same move, confirms that the residual vibration of the system is negligible. Due to the small oscillations, the end effector locations were shifted to zero for this plot to make them easier to see.

Figure 23 shows the maximum residual oscillation of the end effector of the robot resulting from a 0.5 m move in the bridge direction for different configurations of the robot and different end effector masses. The robot started in the configuration shown in Figure 24, and link 1 of the robot was incrementally rotated, while maintaining a

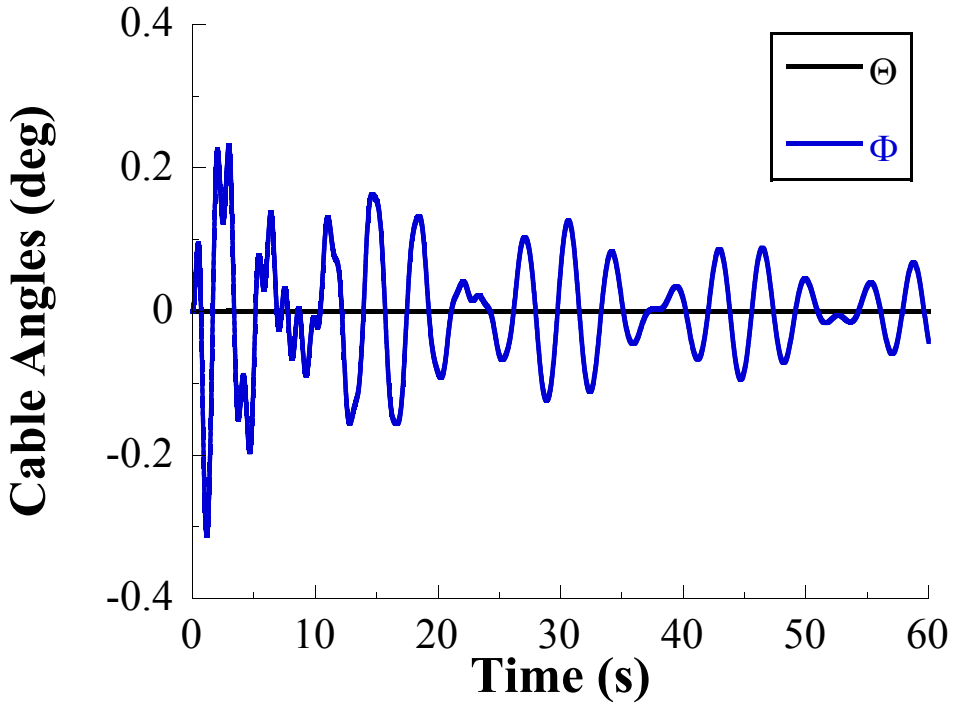

Figure 21: Cable Angles Resulting From a Rotation of Link 1

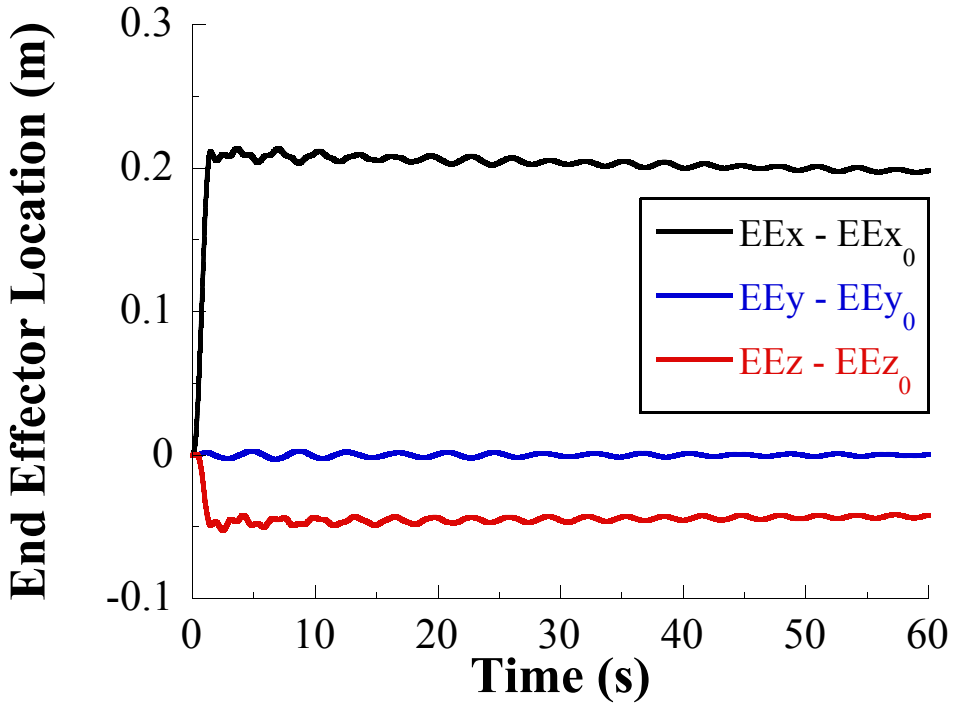

Figure 22: End Effector Location after a Rotation of Link 1.

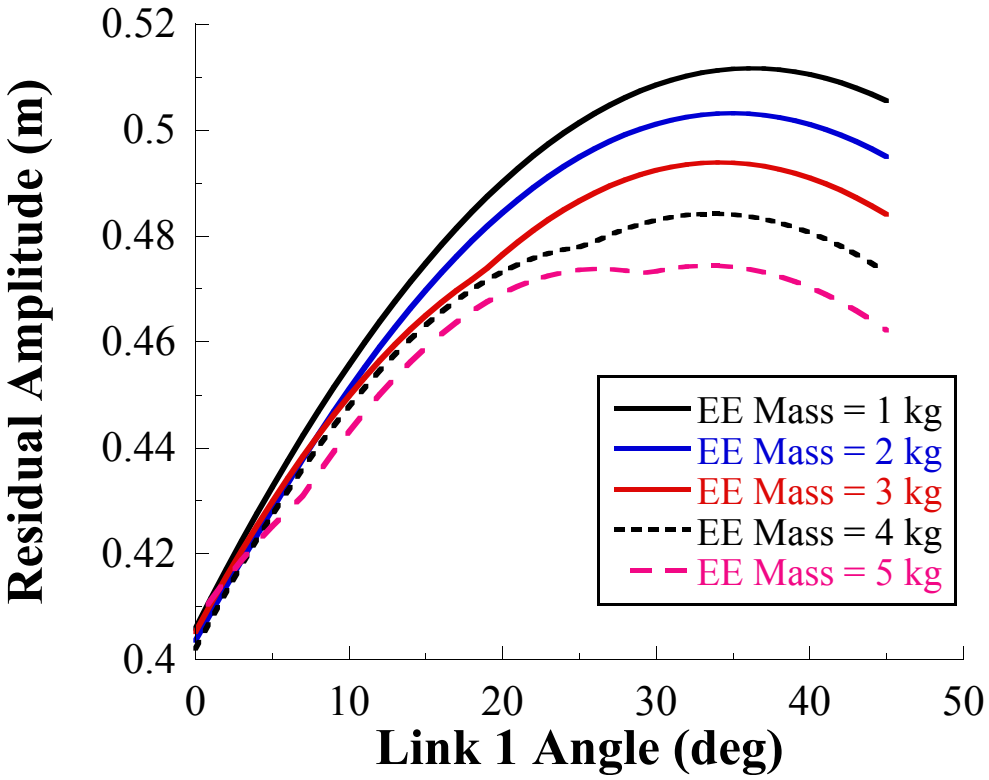

Figure 23: End Effector Response for Varying Link 1 Angle

constant angle between link 1 and link 2. The amplitude of the residual oscillations increase as the end effector of the robot extends further from the frame's center of mass at first. The residual amplitude eventually levels out and starts to decrease.

Figure 25 shows the pitch response of the payload for the same case shown in Figure 23. The response has an interesting shape, most likely caused by the shifting center of mass as the robot arm extends. The overall difference is small however, with a maximum change of less than one degree.

Figure 26 shows the residual oscillations of the end effector the robot resulting from a 1 m move in the bridge direction for varying end effector masses. With the robot in the configuration shown in Figure 24, the mass of the end effector was varied from 0.5 kg to 5 kg and the maximum residual amplitude of the end effector was recorded. The amplitude increased with mass as expected, however the change was

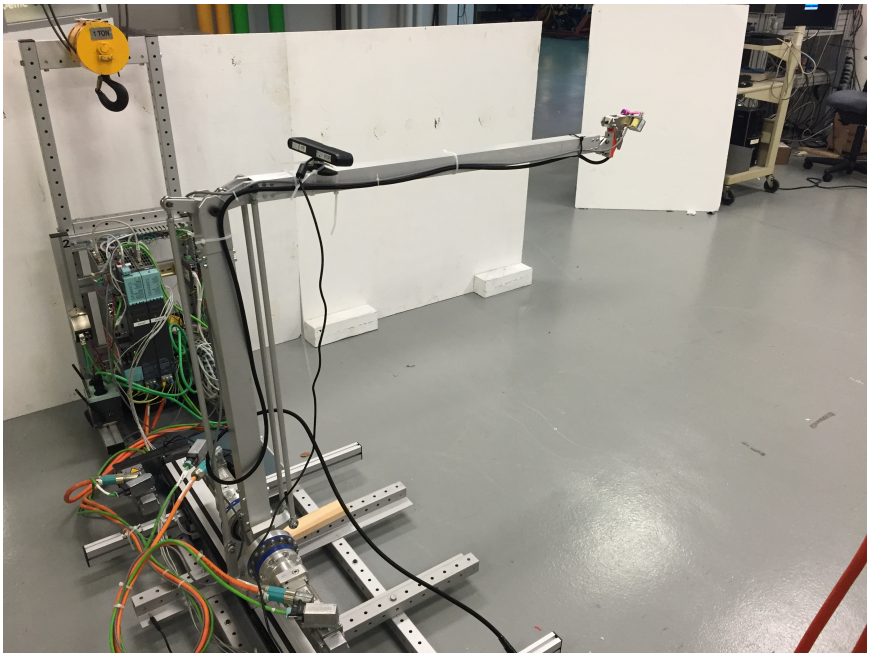

Figure 24: Initial Robot Configuration

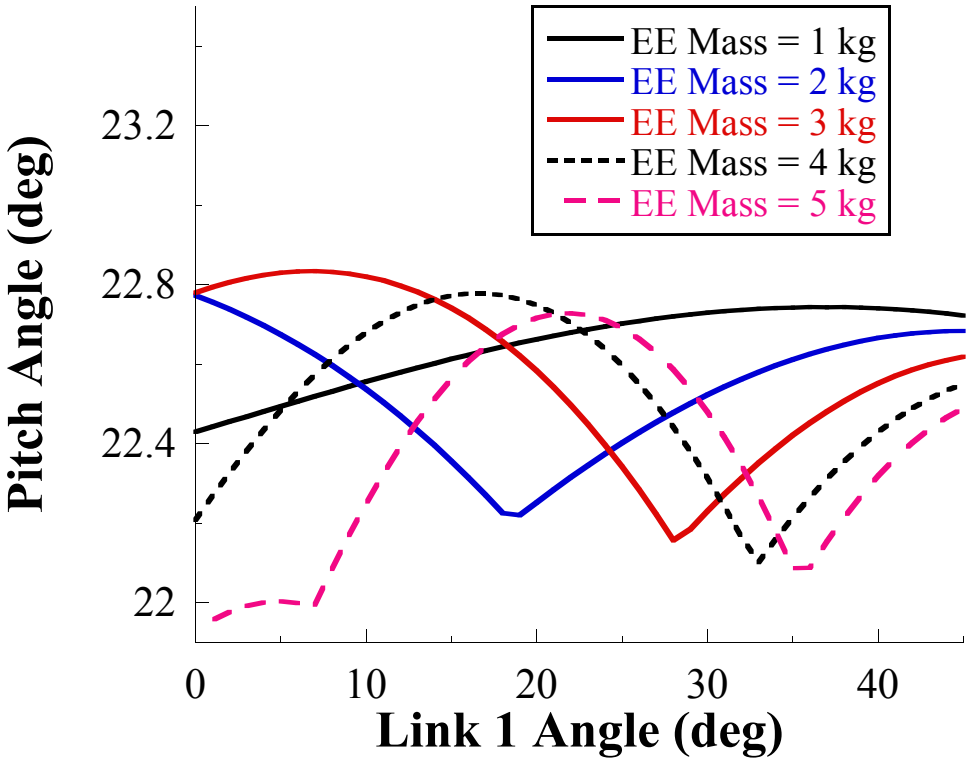

Figure 25: Payload Response for Varying Link 1 Angle

small with a range of less than 1 cm.

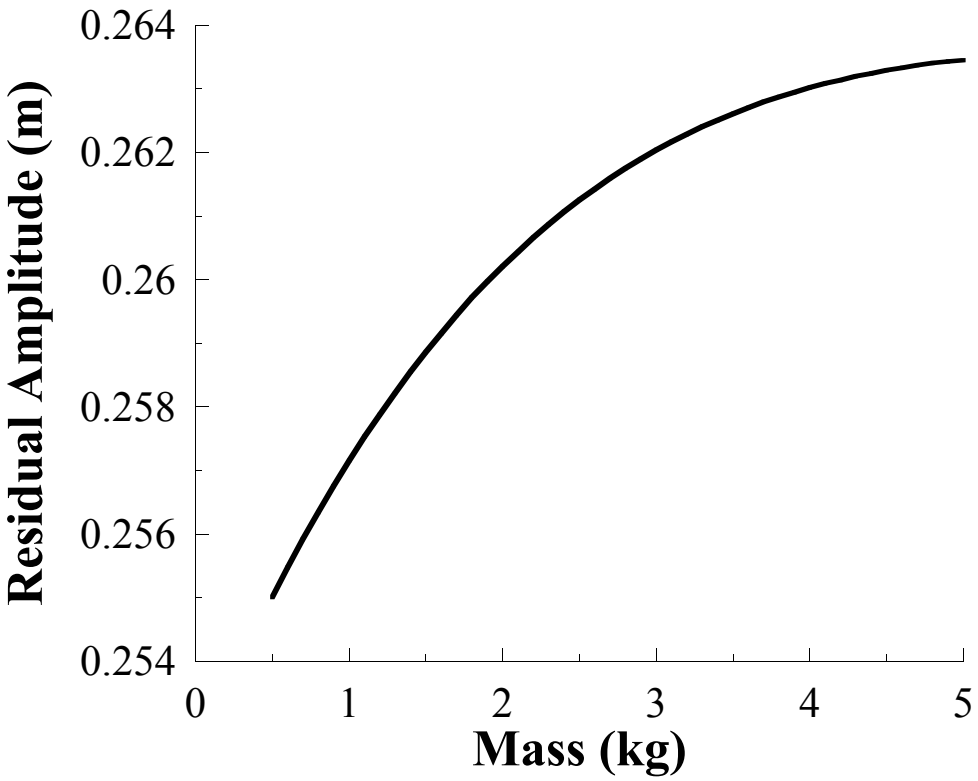

Figure 26: End Effector Response for Varying End Effector Mass

Figure 27 shows the hook response for a linear move of the robot's base with varying end effector mass. This test was performed for different masses of the robot frame. As the mass of the frame increased, the effects of increasing end effector mass decreased.

2.5 Summary

A multi-body mathematical model of the crane robot was developed. The behavior of the model was extensively examined. The results presented so far appear to make sense; however a more quantitative evaluation is necessary to ultimately confirm its accuracy. In the next chapter, a comparison to the physical system is performed in order to further demonstrate the model's fidelity.

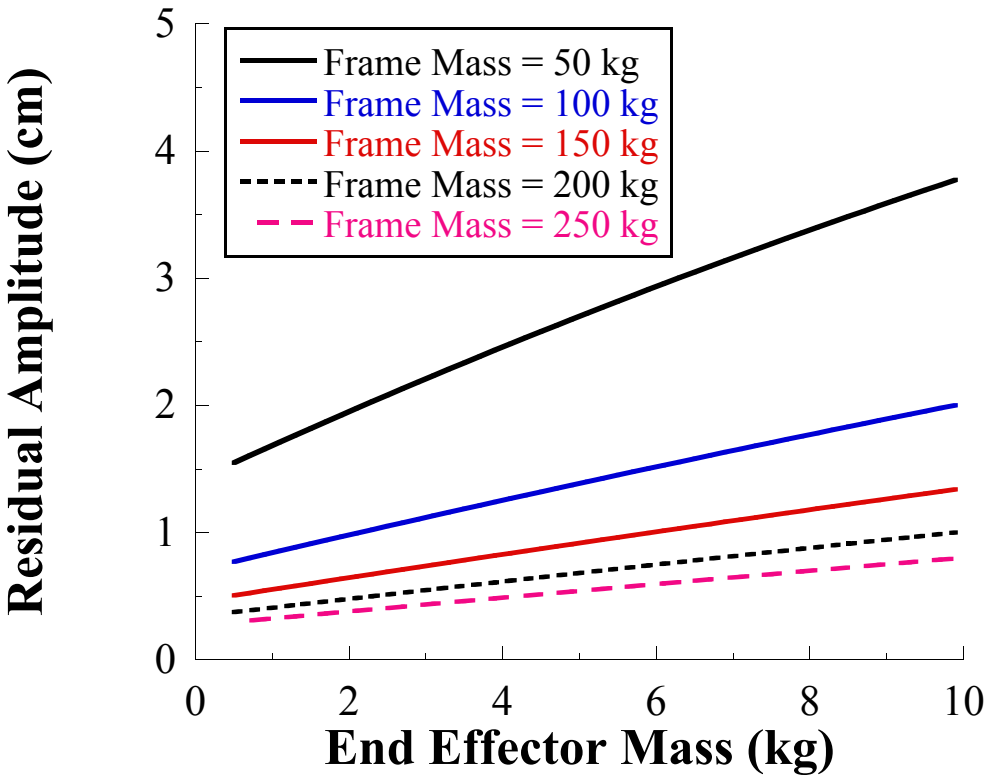

Figure 27: Hook Response for Varying End Effector Mass

CHAPTER III

EXPERIMENTAL VALIDATION OF ROBOT CRANE DYNAMIC MODEL

3.1 Comparing Simulation Response to Physical System Response

This chapter compares the outputs of the dynamic model to the response of the physical system in order to quantitatively validate the model. The following sections analyze crane moves in both the bridge and trolley direction, as well as robot moves. For the bridge and trolley directions, the system was moved a specified distance along the axis being tested and residual oscillations were recorded. The residual vibration was calculated by finding the maximum oscillation amplitude and multiplying it by two. The hook locations were chosen as the points to compare because the crane has an overhead camera system that accurately tracks the location of the hooks. The hook location data can also be directly output from the simulation, allowing for a straightforward comparison between the two data sets.

There is no onboard system to document the response of the robot so experimental data was collected using an external side view camera for rotations of link 1 and 2 and a camera positioned behind the robot to record movements of the robot along the linear stage. A colored marker was placed on the end effector of the robot and a post processing video tracking software was used to extract its location throughout the move. A ruler was placed in the same plane as the marker to provide a reference to convert pixels to meters.

3.1.1 Trolley Motion

Figure 28 shows a comparison of hook 1 location between the actual system and the model for a move distance of 0.6 m in the trolley direction. Even though there are two hooks, data is only shown for one because the hooks behave similarly, allowing the dynamics of the system to be adequately portrayed by one data set. The simulation closely matches the results from the system in both amplitude and frequency. With a simulation frequency of 2.64 rad/s and an actual system frequency of 2.53 rad/s, the experimental error in this case is 4.4%. The control system used is robust to modeling errors of up to 14.3% given an allowable system vibration of 5%. Therefore an error of 4.4% is acceptable and will not adversely effect the results of the model.

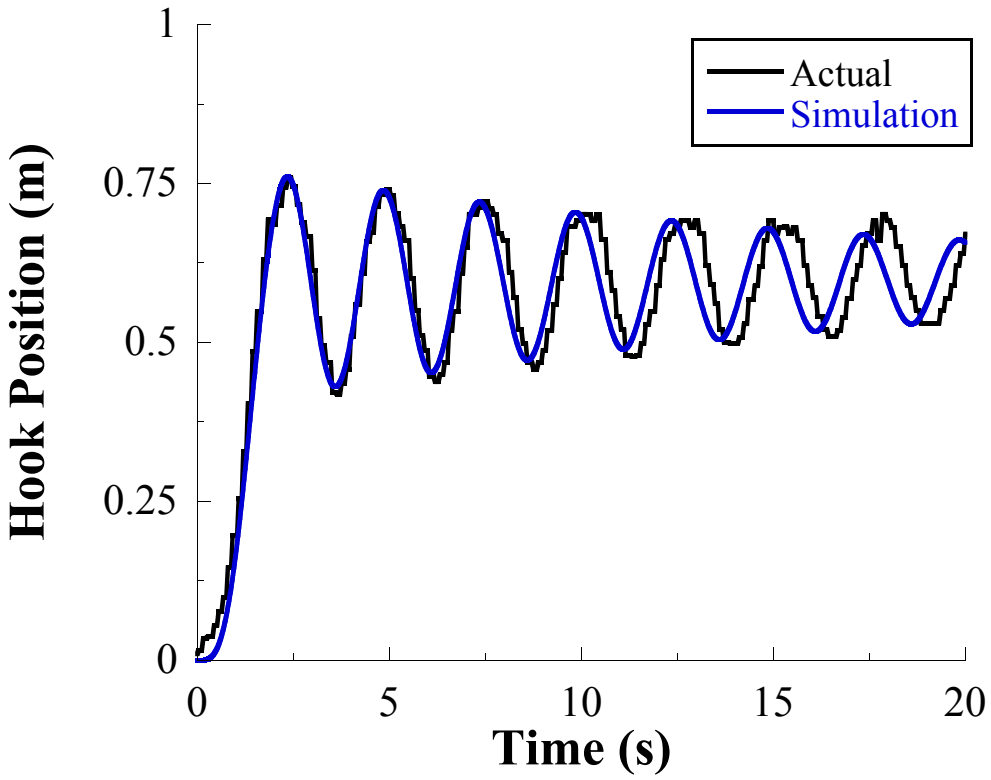

Figure 28: Comparing Hook Response to Simulation for a 0.6 m Trolley Move

The experimental validation was extended by comparing responses over a range of

move distances. Figure 29 displays the residual amplitude versus move distances for trolley moves ranging from zero to three meters. The data point called out in Figure 29 represents the test trial shown in the previous plot.

Movement in the trolley direction creates a simple, single-mode response and closely aligns with the experimental results throughout the range examined.

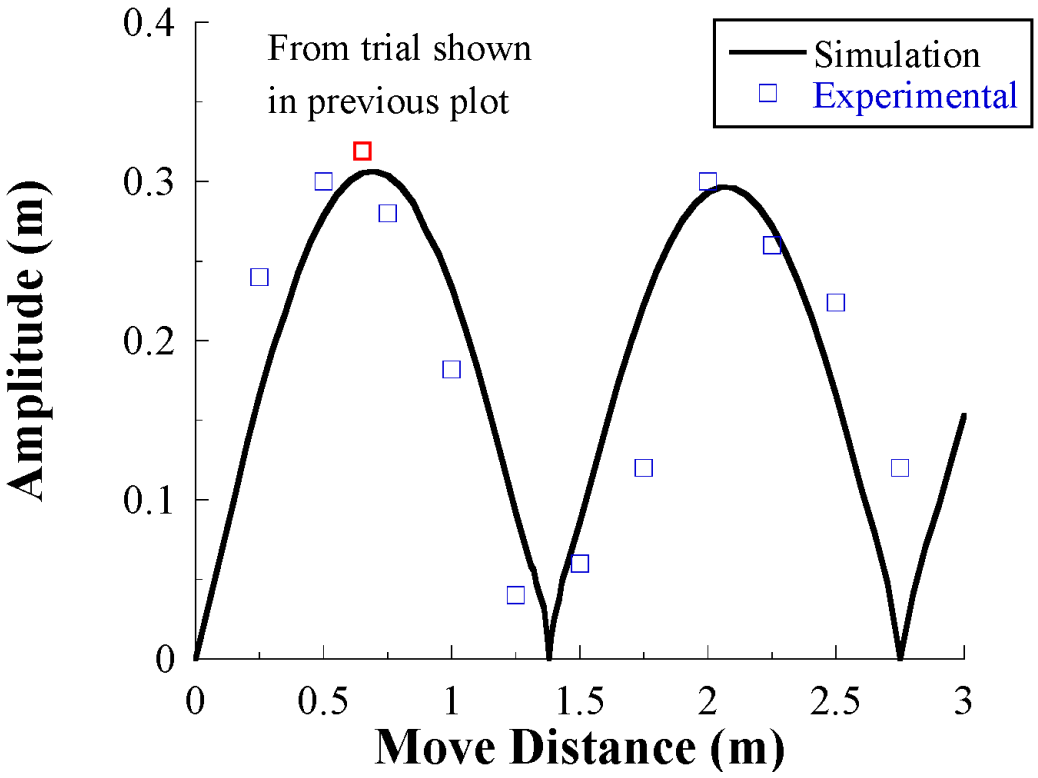

Figure 29: Amplitude vs. Move Distance With Experimental Data

3.1.2 Bridge Motion

Bridge motion produces a more complex response than trolley motion. This is due to an additional frequency induced by pitching of the robot. Figure 30 shows the system response for a 0.75 m move of the bridge, note the multimode response. The residual oscillations compare closely in both frequency and amplitude.

As in the trolley direction, a plot of residual amplitude versus move distance was created for bridge moves ranging from 0 to 2.5 m. This is shown in Figure 31. As expected, this plot does not follow the shape of a single mode system as in the trolley direction. The data point called out on this plot correlates to the data set shown in the previous figure. Overall the physical system closely tracks the results of the simulation for this test.

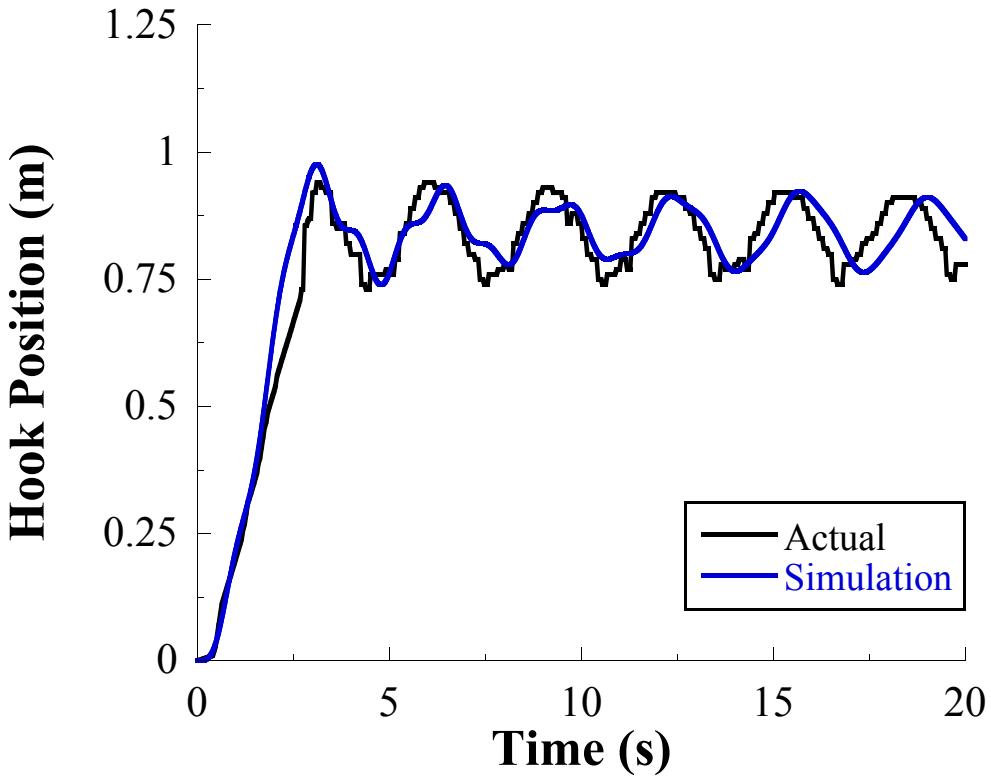

Figure 30: Comparing Hook Response to Simulation for a 0.75 m Bridge Move

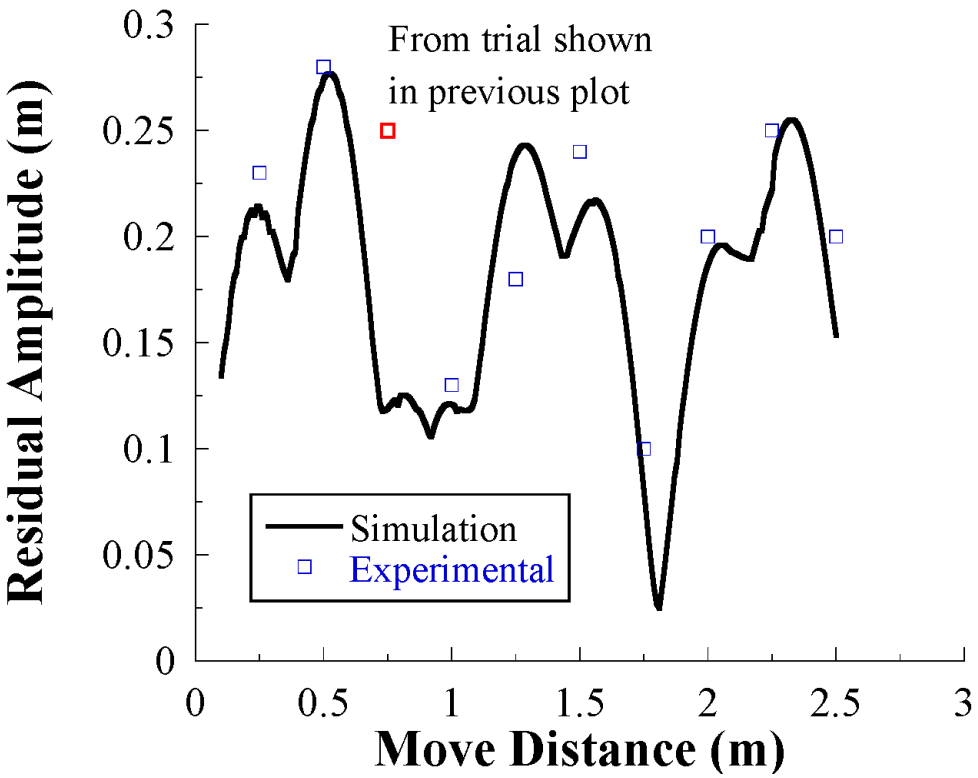

Figure 31: Comparing Hook Response to Simulation for a 0.75 m Bridge Move

3.1.3 Robot Motion

The experimental data for the following plots was collected using a camera because there is no onboard data collection system as for the hooks. Figure 32 shows the response of the end effector in the y direction after a 0.3 m move of the robot's base. The experimental and simulation align closely.

Figure 33 shows the y and z locations of the end effector after a 26 degree rotation of the first link of the robot. For this move the robot started in the same configuration as earlier tests, pictured in Figure 24, and rotated forward. Very little residual oscillation is induced by this move.

Figure 34 shows the result of rotating link 2 of the robot by 14 degrees. Again, the same starting configuration as in previous tests was used. The mass of the robot links is small in comparison to the motors in the base and the frame on which the robot

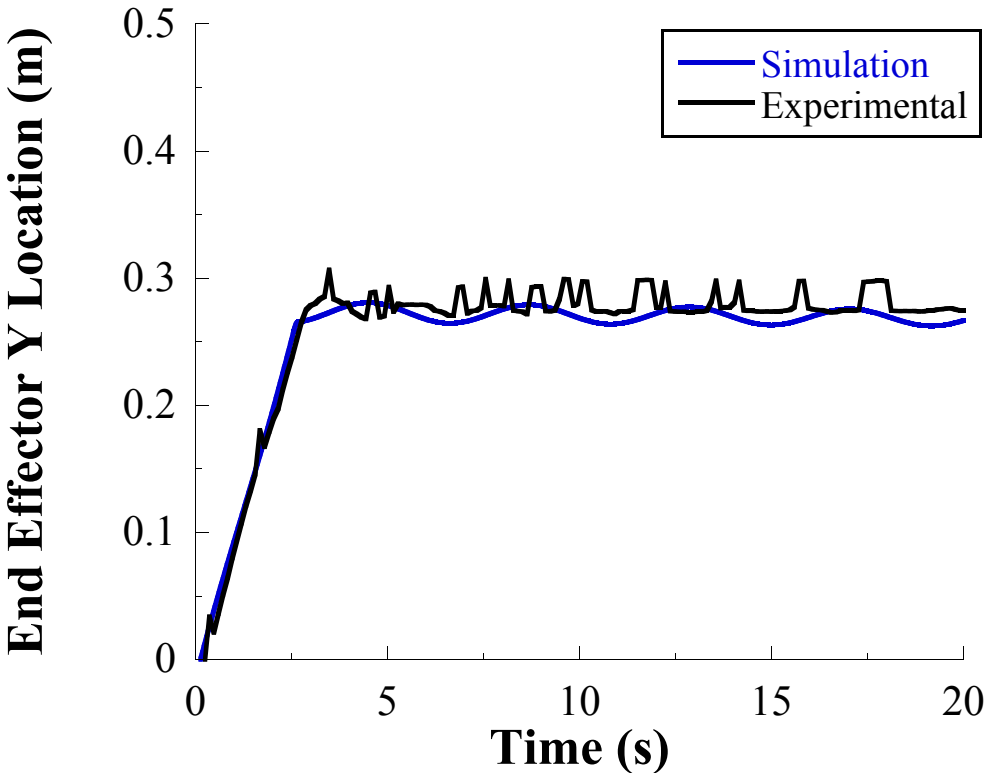

Figure 32: A 0.3 m Move Distance of the Robot's Base

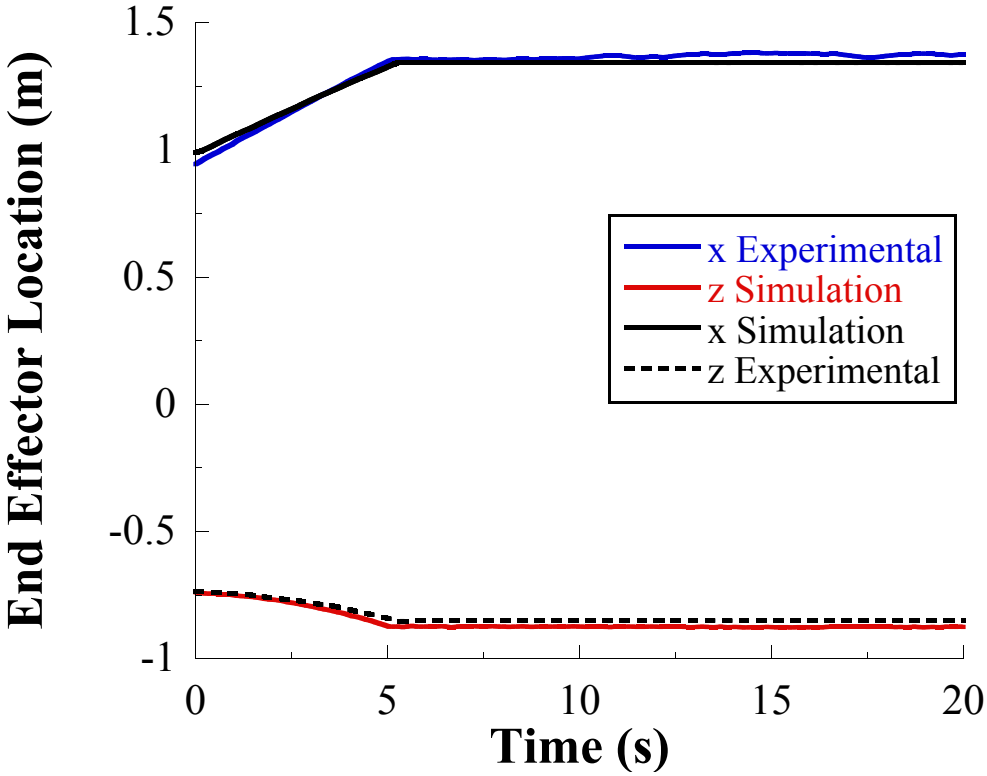

Figure 33: A 26 Degree Rotation of Link 1

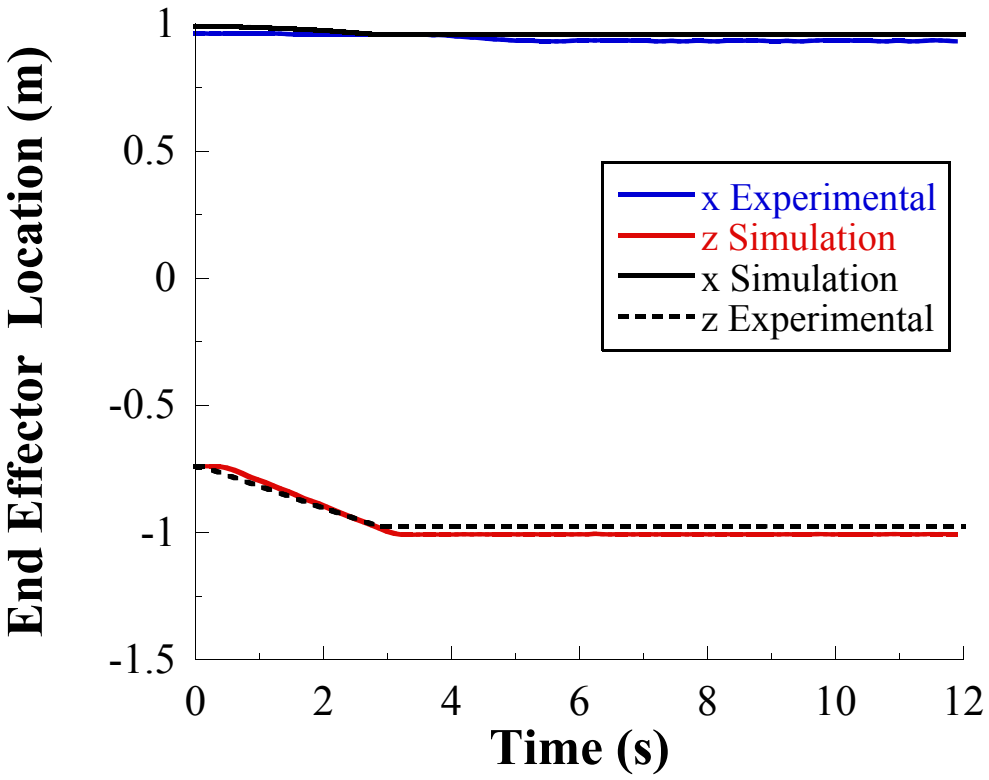

Figure 34: A 14 Degree Rotation of Link 2

rests. Consequently, rotations of links 1 and 2 cause little to no residual oscillation in the system after a move. A move of the robot along the linear stage induces a small amount of vibration in the system due to the comparatively larger mass of the robots motors and base.

Figure 35 shows the residual amplitude of the end effector for a move distance of 0.5 m in the bridge direction with varying end effector mass. This plot is important because a knowledge of how the system responds to different masses can assist with the design of a potential end effector. The simulation data shows a slight upward trend in residual amplitude as mass increases, closely followed by the experimental results.

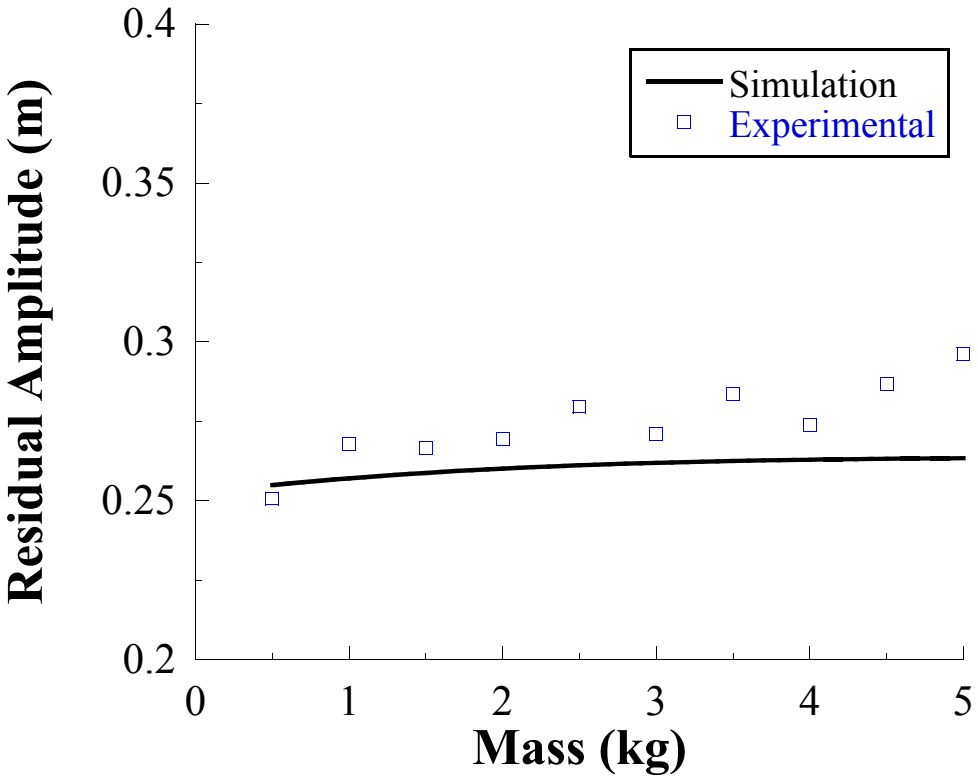

Figure 35: End Effector Response for Varying End Effector Mass

3.2 Summary

The model closely tracks the physical system for all tests performed. This allows the model to be used to test various inputs to the system with reasonable confidence in the accuracy of the results without the need to compare to the physical system.

CHAPTER IV

THREE DIMENSIONAL SENSING

4.1 Objective of Sensor

Three dimensional sensors often have a small effective range that limits their usefulness. The robot crane investigated in this thesis provides a sensor platform that has a very large workspace. Therefore, the system can greatly expand the effective volume of any three-dimensional scanner that is placed on the platform. Sensors of this type have many applications for autonomous systems, the most important of which is arguably increasing the safety through obstacle detection and avoidance. These sensors can also be used for workspace mapping, for general path planning, and to complement or replace manual inspections. The process of using a vision sensor follows a four step cycle: Imaging, Processing, Communication, and Action. This thesis focuses on the first two steps of this process.

An Asus Xtion sensor mounted on the robotic arm will be used to collect and process images after the crane moves. These tests will be performed both with and without the use of input-shaping control on the system. Analysis of the collected data will be used to determine if input shaping can effectively increase the quality of the sensor data. This will also give a metric of how accurate the sensor is with different levels of vibration and movement. Such information is helpful for evaluating and perhaps redesigning the system.

4.2 Sensor Calibration

Calibrating the sensor can greatly increase the data quality. Two types of calibrations exist for sensors of this type, intrinsic and extrinsic. Intrinsic calibration is related

to the internal parameters of the camera such as lens distortion and focal length. Extrinsic calibration is only needed when two sensors are being utilized. It is used to determine the relative transform between the two sensors. The extrinsic calibration was performed for the RGB and IR sensors in the Xtion. Although the RGB and IR sensors are optimally placed to reduce the need for an extrinsic calibration, a small increase in accuracy can still be gained. A built-in calibration tool was used for all calibrations.

4.2.1 Intrinsic Calibration

A 6x8 checkerboard with 108 mm squares was used for the intrinsic calibration of the RGB camera. This board is shown in Figure 36.

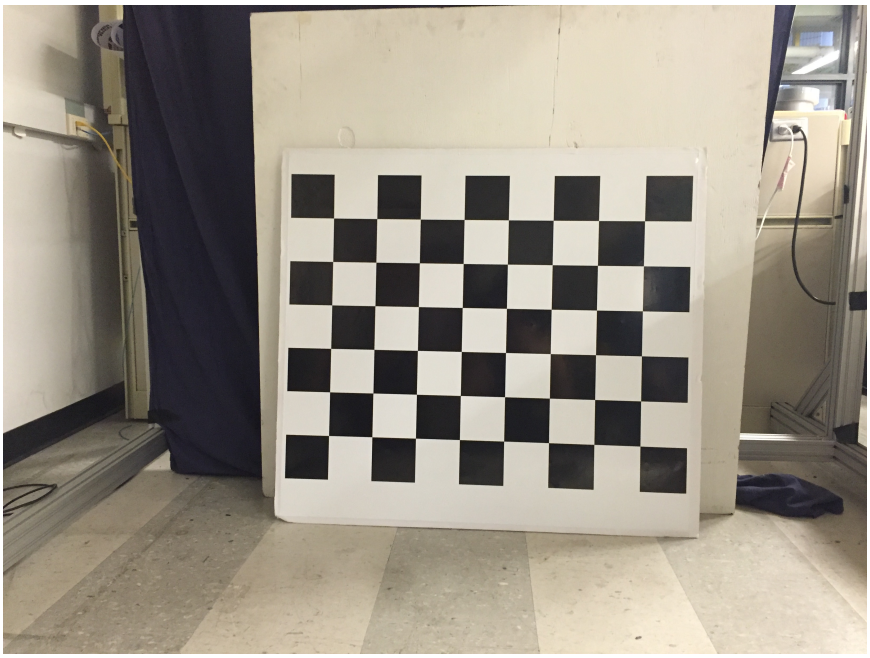

Figure 36: Checkerboard Used for Calibration of the Vision Sensor

After starting the camera, approximately forty images of the checkerboard in various orientations encompassing the full range of view of the camera were collected.

Figure 37 shows a sample of these images.

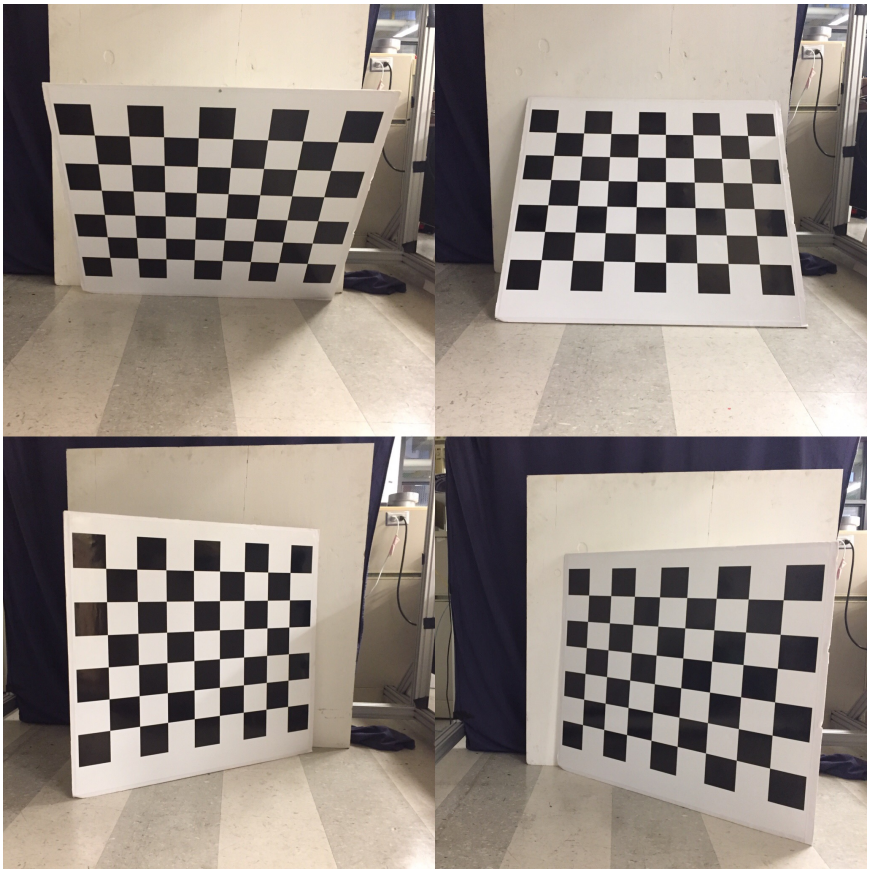

Figure 37: Samples of the Different Orientations Used to Calibrate the Sensor

The calibration tool then generated a file of calibration constants which is used each time the sensor is launched. This process was then repeated for the IR sensor.

4.2.2 Extrinsic Calibration

The extrinsic calibration was performed using the same checkerboard pattern in much the same way as the intrinsic calibration, with a few important differences. First, a different calibration package was used. Secondly, data must be streamed from both the RGB and IR sensors simultaneously in order to successfully complete the calibration. However, the Asus Xtion is not able to stream from both the IR and RGB

cameras at the same time, requiring a modified procedure to simulate simultaneous streaming. To achieve this, the checkerboard was set up opposite the sensor and several seconds of data were collected with the RGB camera and saved. A setting was then changed in the sensor code, enabling simulated time and turning off actual clock time. Finally, the IR sensor was initialized and the previous RGB data set was played simultaneously. This tricks the sensor into believing both streams are occurring in real time and allows the calibration to take place. If either the camera or checkerboard moves during the process, it will result in a faulty calibration. This method only allows for one view of the checkerboard which is not ideal, but it provides better results than no calibration and is adequate for this research.

An additional calibration of the sensor for various depths was performed. This calibration is not vital for the applications considered in this work because the majority of the sensing takes place at a fixed distance. However, if the sensor is used for other operations requiring tight tolerances across a range of depths this calibration will prove useful.

4.3 Input Shaping to Reduce Sensor Noise

After completing the calibration, a test was performed to determine the baseline accuracy of the sensor. Figure 38 shows the setup for this test.

A circle and square of known dimensions were placed on a board 1.5 meters away from the sensor. With the system stationary, several seconds of data were recorded with the RGB camera. Through post processing, the locations and sizes of the circle and square were extracted and compared to the actual dimensions. A ruler placed in the same plane as the shapes enabled a scale factor to convert from pixels to inches. Figure 39 shows the results of this test for the square, and Figure 40 shows the data for the circle. The data from the sensor closely matched the actual dimensions for both the circle and the square.

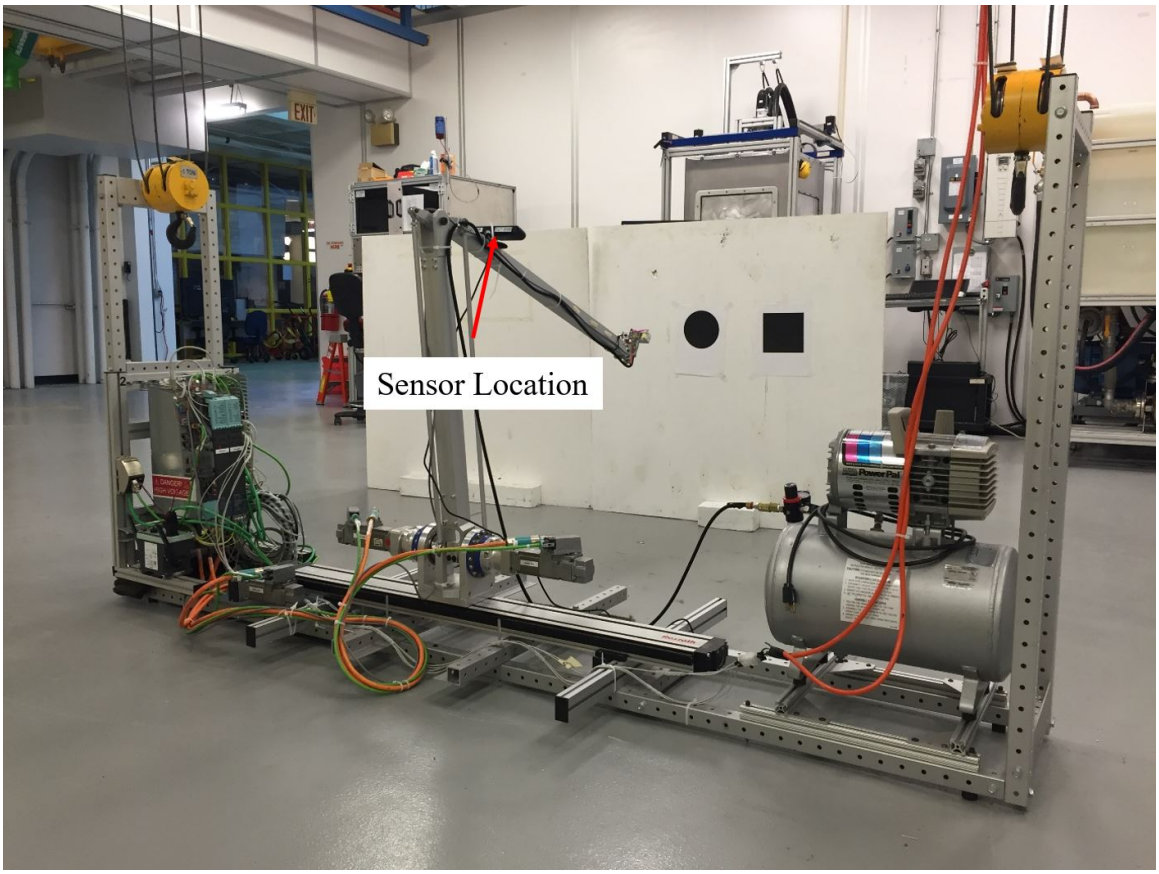

Figure 38: Setup Used to Perform the Sensor Tests

To test sensor performance during motion, the system was moved 0.5 m in the trolley direction. The sensor started data collection after the end location had been reached. Note that the crane platform generally oscillates about the desired location, therefore the sensor is not stationary during its measurements. This emulates a typical function in a factory setting; the system driving for some distance and collecting data once a specified location has been reached.

Figure 41 shows the data for the square and Figure 42 shows the circle data. In this test the measured dimensions of the shapes oscillate as the system moves. It is important to note that the measured dimensions closely match the actual dimension when the system reaches the peak of its swing in any direction. This is because the velocity of the system is zero at these locations, essentially allowing the sensor to

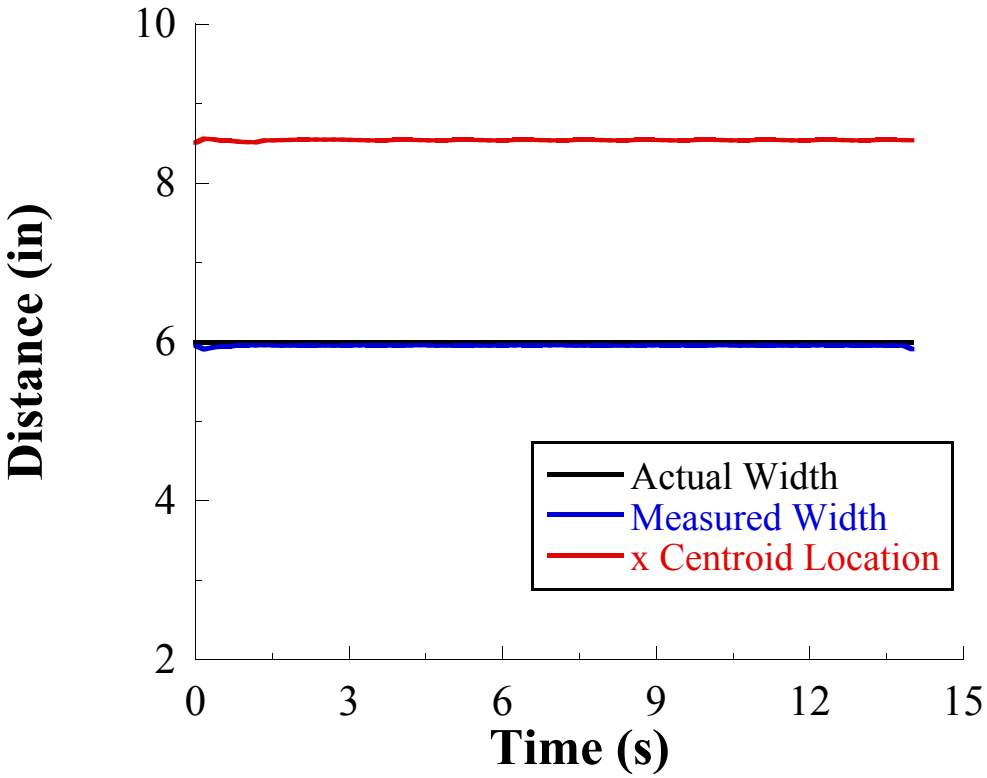

Figure 39: Data For the Square With Stationary Sensor

get a good "look" at the shape. Maximum error occurs when the system reaches maximum velocity.

Fluctuations in the measured dimensions on the scale shown in Figures 41 and 42 are not acceptable if accurate measurements are required. Waiting for the residual oscillation of the system to subside before using the sensor, incurs a large time penalty because the oscillations are lightly damped and have a long period. Greatly reducing the oscillation via a control system is obviously a requirement to enable high-level system performance.

Input shaping is a common control method for flexible systems that reduces residual oscillations and only requires knowledge of the system's frequencies and damping ratios. With input shaping, the command that is sent to the system is preprocessed through convolution with a set of strategically placed impulses. Figure 43 shows the first step of the input-shaping process and Figure 44 shows results of applying the

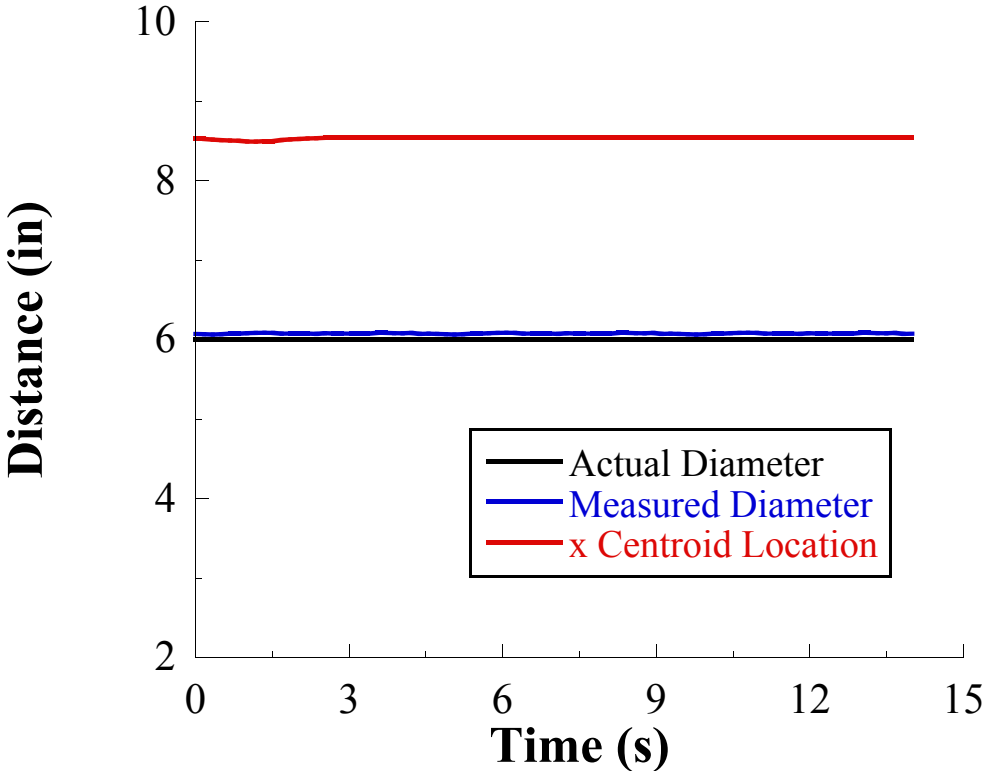

Figure 40: Data For the Circle With Stationary Sensor

input-shaping process. The input shaper type depicted in this figure is a zero vibration (ZV) shaper containing only two impulses [23]. A more robust input shaper, the zero vibration and derivative (ZVD) shaper [19], is used for the following tests.

Figure 45 shows the sensor data for the square using the same move distance performed earlier; however, this time with input-shaping control. Figure 46 shows the corresponding data for the circle. The input-shaped response has much reduced residual oscillation, and the measured and actual measurements closely match.

The residual amplitude of the system under input-shaping control was tested for a range of move distances in both the trolley and bridge directions. Figure 47 shows the results for the trolley direction. As visible from the plot, the amplitude of residual oscillation of an input-shaped system is independent of move distance. This feature of input shaping is extremely useful because it allows the system to travel any distance

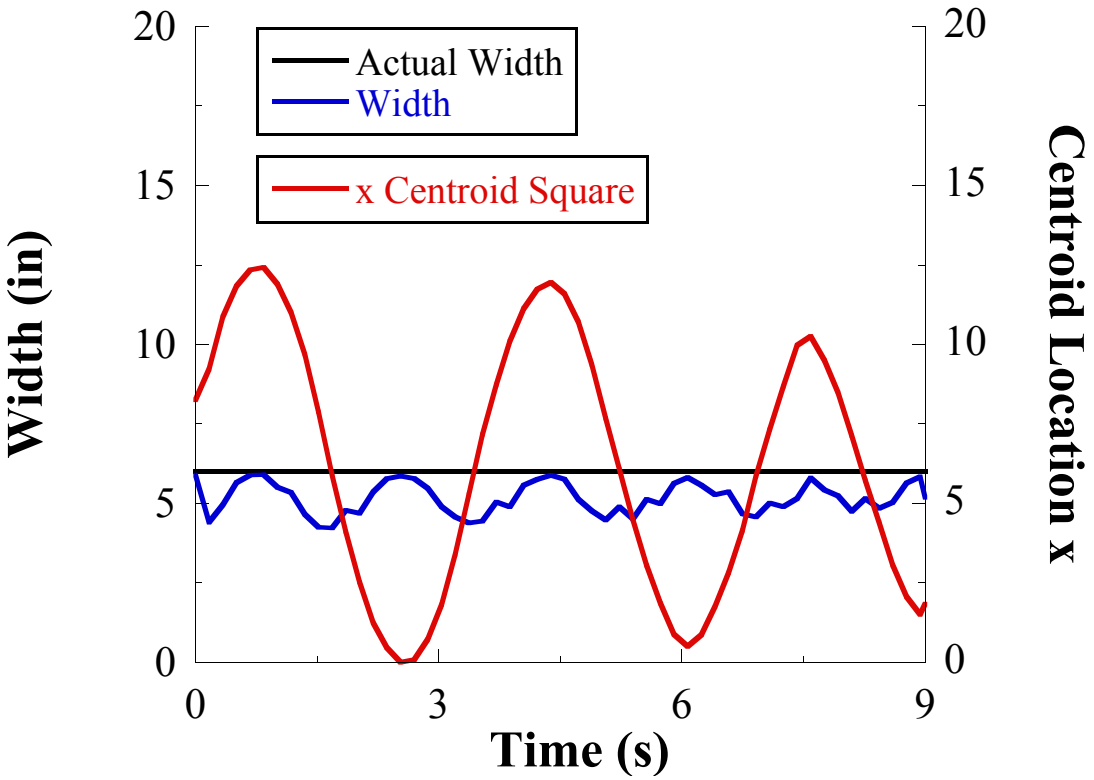

Figure 41: Sensor Data For the Square After System Move

with essentially no residual oscillation.

Figure 48 shows the results for the bridge direction.

It is a very important result that varying move distance has no effect on the response of an input shaper. However, one thing that has an adverse effect on an input-shaped system is changes in system frequency. An input shaper is designed for a specific frequency or range of frequencies. If the system frequency shifts too far outside that range, then undesirable results can arise.

The most common way for the frequeny of the crane system to change is through hoisting or lowering the payload. Figure 49 shows the maximum error in the sensor data for varying frequencies for both ZV and ZVD shapers. In these tests, the cable lengths down to each hook was initially set to match the design frequency of the input shaper. The cables were then shortened in increments of 10 cm to increase

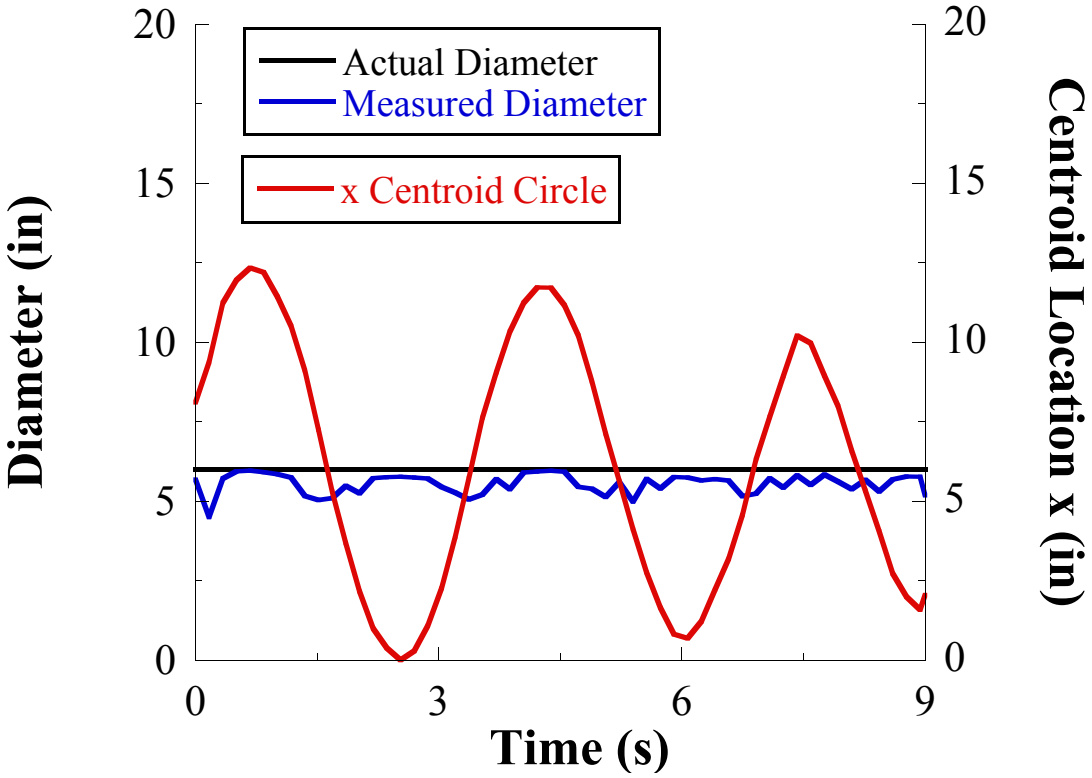

Figure 42: Sensor Data For the Circle After System Move

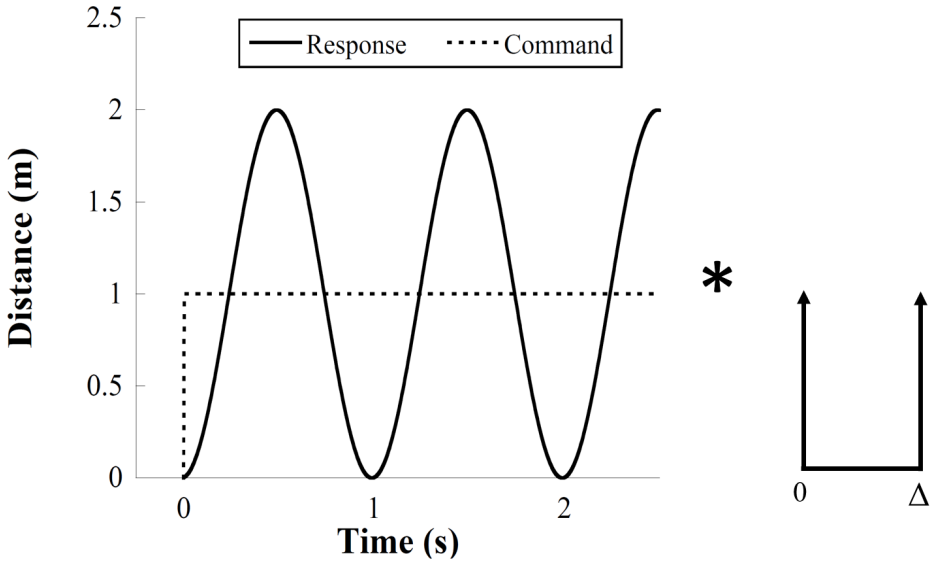

Figure 43: General Schematic of Input-shaping Process: Part 1

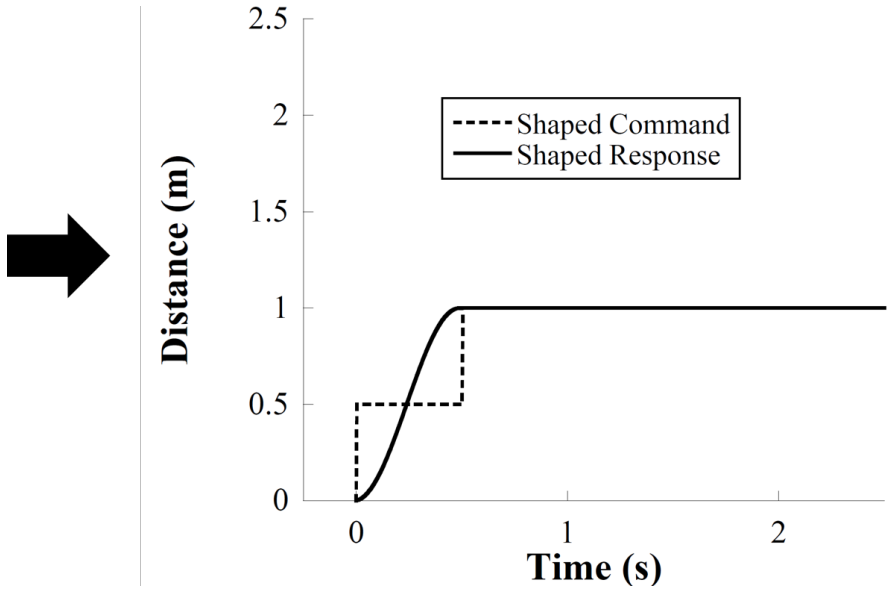

Figure 44: General Schematic of Input-shaping Process: Part 2

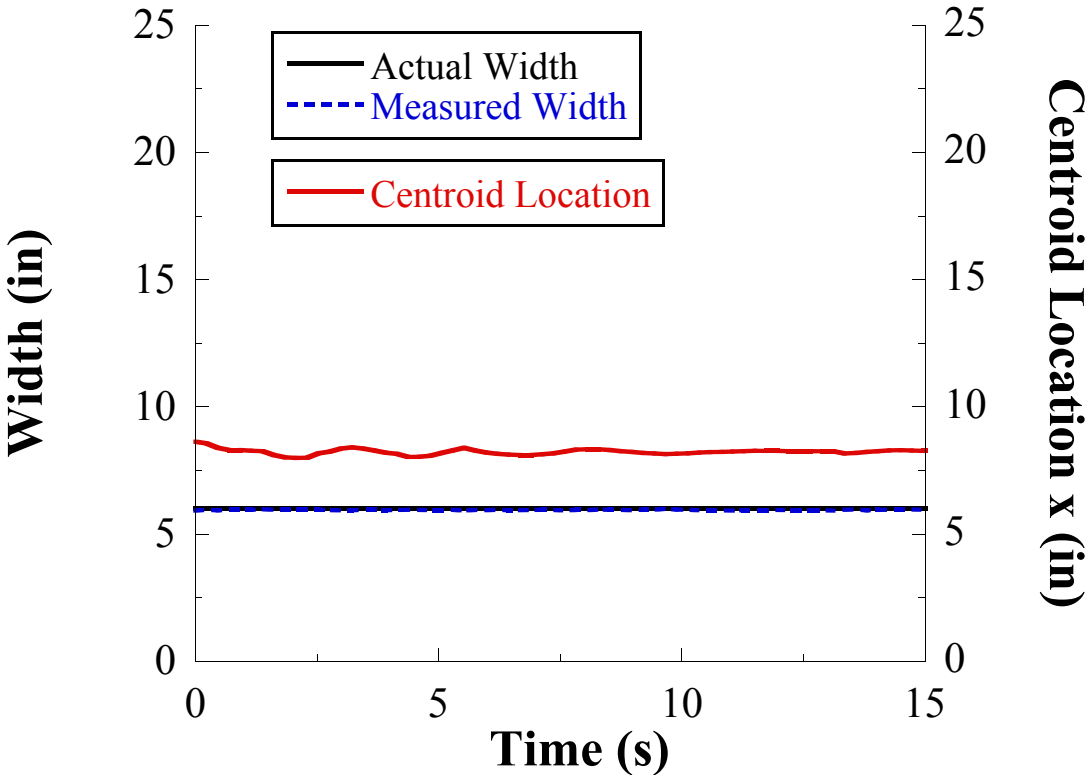

Figure 45: Sensor Data For the Square After System Move With ZVD Shaper

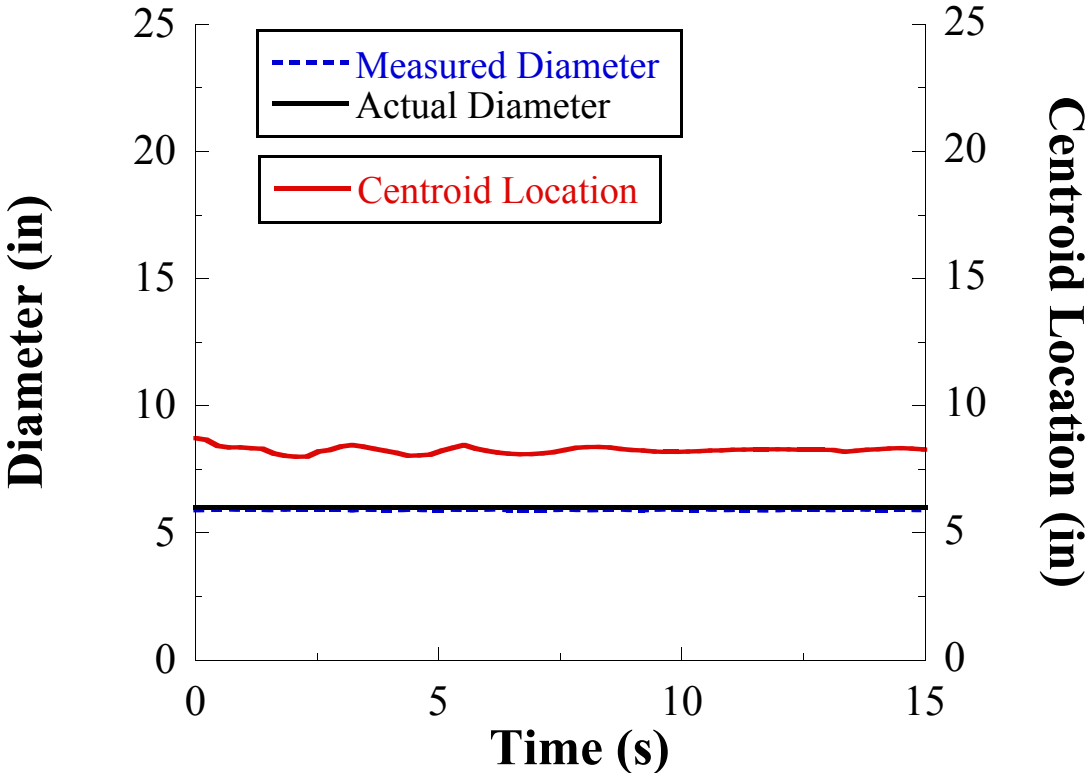

Figure 46: Sensor Data For the Circle After System Move With ZVD Shaper

the system's oscillation frequency. Both shapers were designed for a frequency of approximately 2.6 rad/s. For frequencies above 3 rad/s, the data quality using the ZV shaper begins to decline rapidly; however, the data quality for the ZVD shaper remains constant. This is because the ZVD shaper is more robust to changes in frequency that the ZV shaper. If the frequency continued to increase, then eventually a similar trend would be visible in the ZVD shaper data as well.

4.4 Summary

A calibration of the Xtion sensor has been completed and a thorough evaluation performed. It was found that the sensor responds poorly to oscillations of the platform on which it is mounted. However, input shaping effectively reduces the residual oscillations of the system, thereby increasing the quality of the sensor data. With the

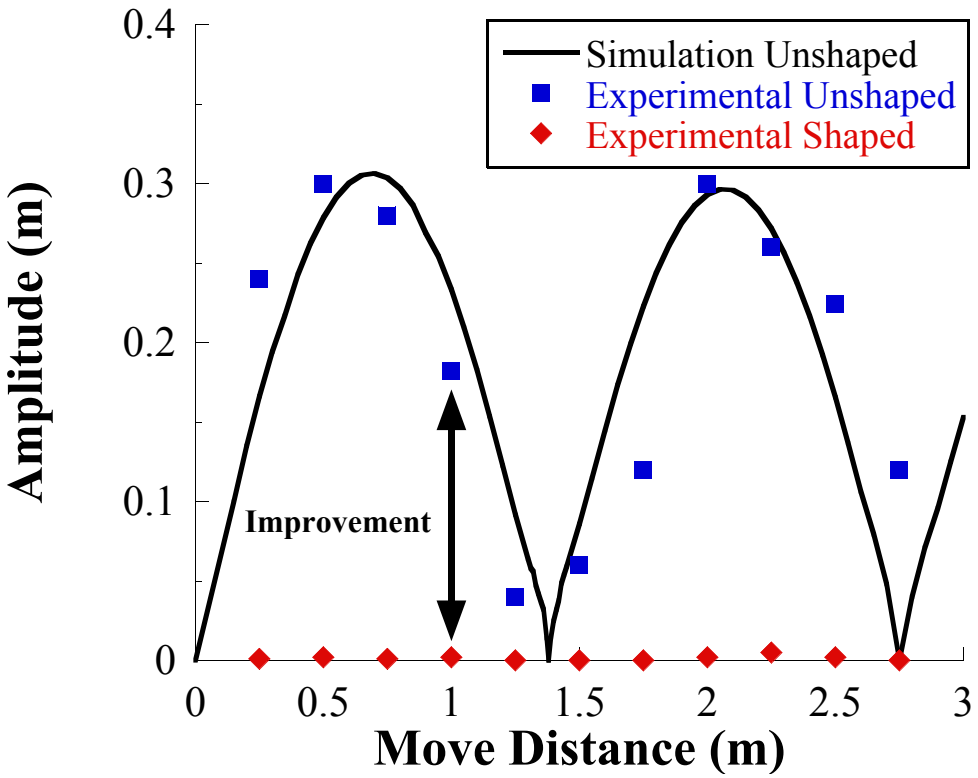

Figure 47: Amplitude vs. Move Distance in the Trolley Direction

addition of input shaping the sensor can be effectively used for inspection purposes over the massive workspace of the robot crane.

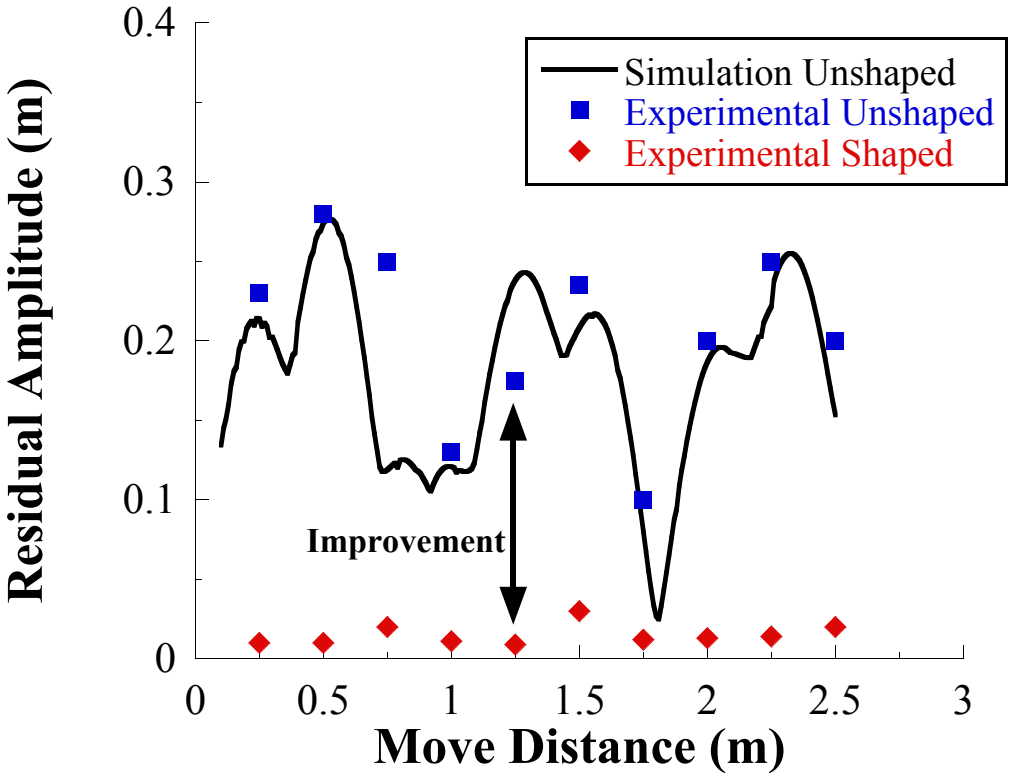

Figure 48: Amplitude vs. Move Distance in the Bridge Direction

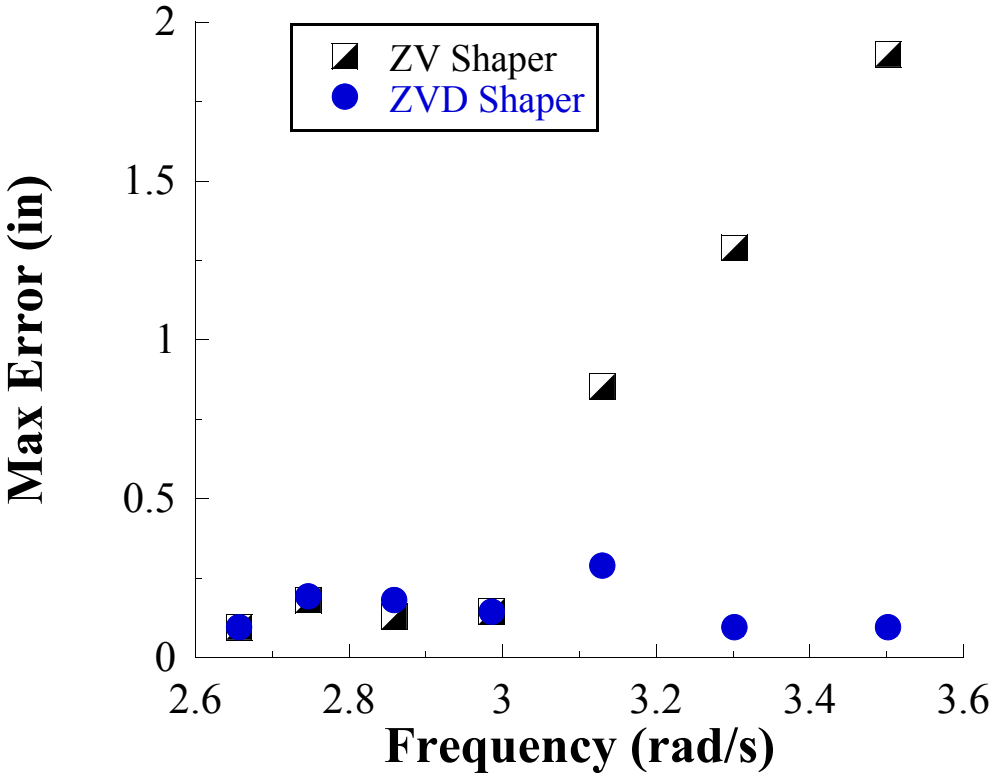

Figure 49: Comparing Robustness of ZV and ZVD Shapers Through Data Quality

CHAPTER V

IMPROVEMENTS AND POTENTIAL APPLICATIONS OF THE ROBOT CRANE

The work presented in this thesis has resulted in the development of a massive workspace robot crane that can accurately be positioned throughout its workspace. The system is equipped with a three dimensional sensor that can accurately acquire visual information. Therefore, the system can monitor its environment to avoid obstacles and locate target positions. This allows the work in this thesis to be extended in an important area, increasing the autonomy of the system. With the automation of any system, especially one in an uncontrolled environment such as a crane workspace, it is important for the system to be able to sense and respond to changes in the surroundings. If the end effector of the robot approaches an object, commands can be sent to the crane and robot to either stop or avoid the obstacle and continue work. This will reduce the likelihood of dangerous and costly collisions.

In order to achieve such autonomy, the sensor must be able to to send and receive data from the programmable logic controller (PLC) running the robot crane. This feature is not yet functional due to a delay with the company that supplies the necessary hardware. Once the sensor can send and receive information from the PLC, real time data processing is essential for obstacle detection and avoidance.

Another area that deserves attention is the data collection process used to document the system response. The crane has overhead cameras that track the hook location in 3D space but that is the only onboard sensor available to track system response. In this thesis an external camera was used to collect data in many cases, but this has its limitations. Inertial measurement units (IMU) could be mounted on

the frame and robot to more effectively and efficiently extract data documenting the response of the payload.

With the above improvements in place, additional applications for the system become very promising. Two applications that are briefly examined here are painting and sand blasting. Figure 51 shows the airbrush end effector to be used for painting operations. The airbrush is powered by an air compressor mounted to the platform and painting is actuated using a small solenoid. Figure 50 shows a sample that was painted with the system using a preprogrammed path and a stencil.

Figure 50: Boeing Logo Painted Using Air Brush End Effector

Figure 52 shows a close up view of the sand blaster. This end effector is also powered by the onboard air compressor and operates on the same principle as the air brush. However, instead of paint an abrasive media is fed through with the air stream. Figure 53 shows the sand blaster mounted on the robot arm, the black hose runs to a receptacle for the abrasive and the yellow hose runs to the air compressor. Sand blasting can be used for operations such as removing paint and rust or smoothing surface imperfections, depending on the coarseness of abrasive media used. This end effector has not yet been tested due to safety concerns arising from inhaling the dust that is generated through use. An enclosure must first be constructed to contain the abrasive dust and debris.

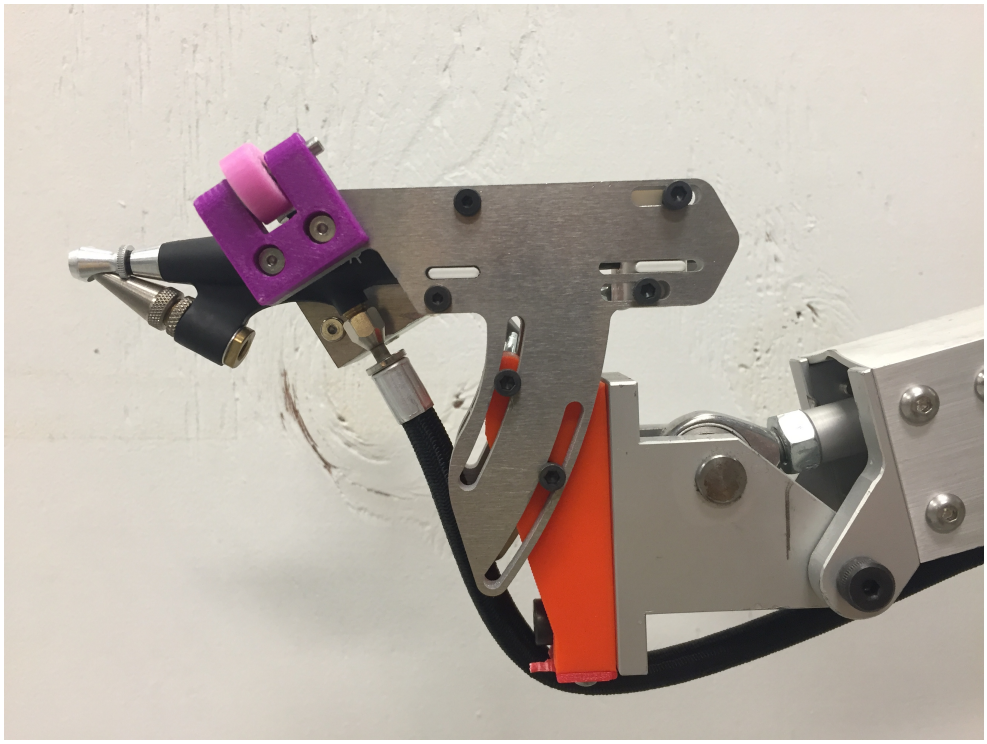

Figure 51: Painting End Effector

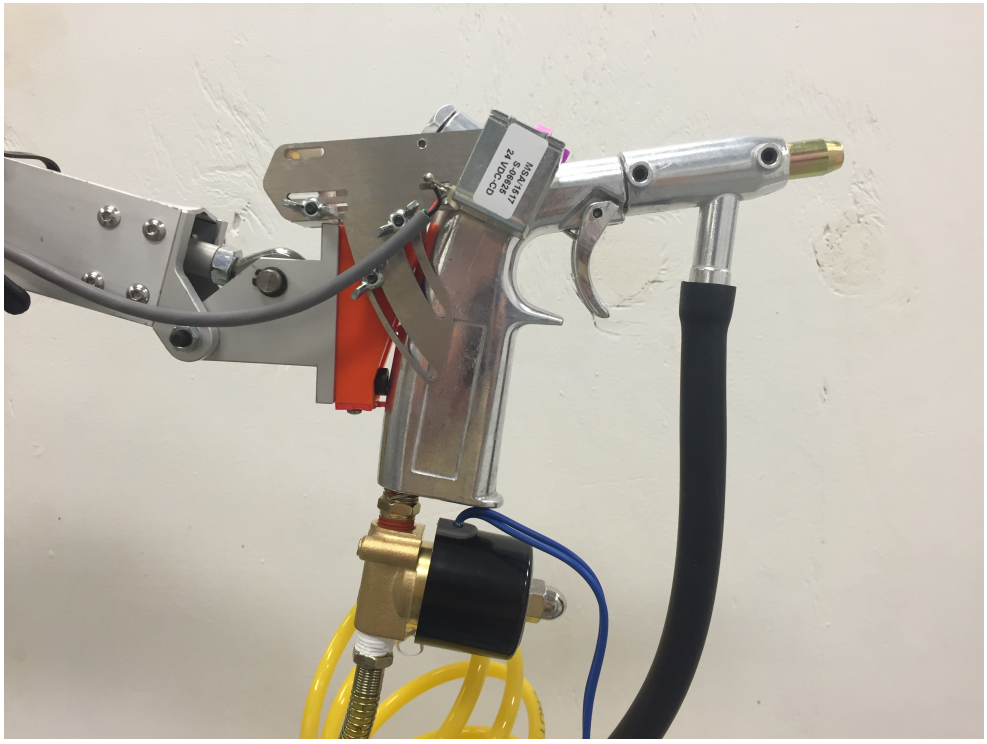

Figure 52: Sand Blasting End Effector

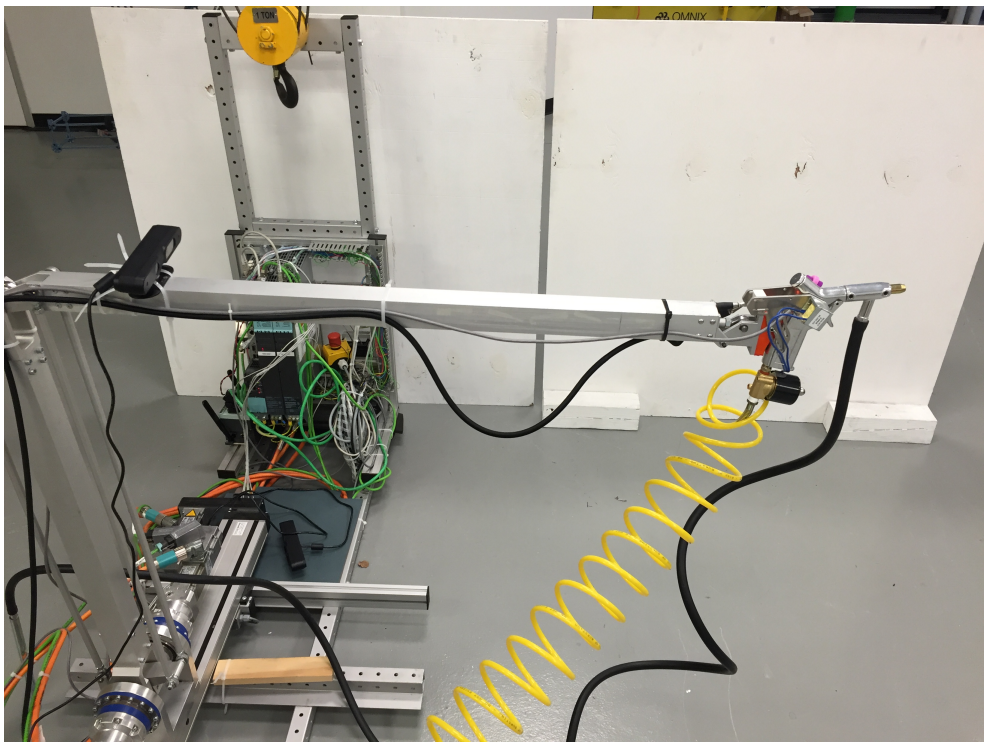

Figure 53: Sand Blasting End Effector

CHAPTER VI

CONCLUSIONS

Chapter 1 presented a background for the problem and explored prior work in the field. In Chapter 2, a numerical model was created and extensively tested to ensure its validity. This allows the dynamics of the system to be better understood and provides a test bed on which large numbers of inputs can be tested in considerably less time than on the physical system. In this chapter the model was used to explore several interesting system inputs. This multi-body mathematical model of the robot crane is one of the major contributions of this thesis. Chapter 3 validates the model by comparing it to experimental results. The results from the physical system closely matched all results found with the model thereby confirming the accuracy of the model.

Next, in Chapter 4, the Asus Xtion vision sensor was calibrated and evaluated to determine the quality of sensor data. The data quality was found to be low while the system was oscillating. In an attempt to reduce the amplitude of residual oscillations, input shaping was used. The decreased residual oscillations of the crane after a move, increasing the data quality returned from the sensor. After a thorough evaluation the Xtion sensor was found to be a suitable solution for inspection purposes if paired with input shaping control.

In Chapter 5, several areas of future work were presented. These include: increasing the autonomy of the system and implementing sensors onboard the robot to improve the data collection process. Two promising applications were also presented; painting and sandblasting.

APPENDIX A

MOTION GENESIS MODEL

```
SetAutoZee(ON)
%---------------------------------------------------------------------
% FRAMES, PARTICLES, AND POINTS
%---------------------------------------------------------------------
       NewtonianFrame N             % Default Newtonian frame
       RigidFrame Cable1, Cable2    % Rigid, inflexible cables
       RigidBody Link               % Rigid payload between cables
       Point T1(Cable1), T2(Cable2) % Trolleys

% Two ends of rigid link, for convenience - also allows adding hook mass
% to simulation
       Particle P1, P2                    % E/W cable hooks

% Rigid Frames for each link of the robot
       RigidBody Base, Link1, Link2, Bar2, Bar3
       Particle EndEffector
       Point PB1L2(Link2)                 % Pivot between bar 1 and link 2
%---------------------------------------------------------------------
% VARIABLES AND CONSTANTS
%---------------------------------------------------------------------
% Crane Variables
Variable theta_1'', theta_2'',beta''  % cables angles for trolley motion
Variable phi_1'',phi_2'', gamma''     % cable angles for bridge motion
```

```
Variable psi''                          % payload rotation angle
Variable y1'', y2''                     % Trolley pos, vel, accel
Variable x1'', x2''  % Bridge pos, vel, accel (same for both trolleys)
Constant LE+, LW+, LC+, g+    % cable lengths and gravity
Constant MC+, ME+, MW+                                % Masses
Specified a_trol1, a_trol2, a_bridge % accel of trolleys & bridge
Constant B_cable1, B_cable2                           % cable damping
Constant Ixx, Iyy, Izz, Ixy, Iyz, Izx % moments of inertia of payload
Constant LCx, LCy, LCz                                % to COM of payload

% Robot Variables
       Constant LBase, LLink1, LLink2, LBar2, LBar3
       Constant MBase, MLink1, MLink2, MEndEffector, MBar2, MBar3
       Constant Distance1
       Constant RobotOffset
       Constant EEForcex, EEForcey, EEForcez
       Variable BasePos''    % Position, Velocity, and Acceleration of
          robot base
       Variable qLink1''              % Link 1 angle
       Variable qBar2'',qBar3''       % Angles of linkages driving Link2
       Variable qLink2''              % Link 2 angle

       Specified AngAccLink1 % Acceleration commands from the motors
       Specified AngAccBar3
       Specified Base_Acceleration

% Hook masses
       P1.SetMass(ME)
       P2.SetMass(MW)
```

```
% Payload mass and inertia
        Link.SetMass(MC)
        Link.SetInertia( LinkCM, Ixx, Iyy, Izz, Ixy, Iyz, Izx )
        Base.SetMass( MBase )
        Base.SetInertia( Basecm, IBasexx, IBaseyy, IBasezz, 0, 0, 0 )
        Link1.SetMass( MLink1 )
        Link1.SetInertia( Link1cm, ILink1xx, ILink1yy, ILink1zz, 0, 0, 0 )
        Bar2.SetMass( MBar2 )
        Bar2.SetInertia( Bar2cm, IBar2xx, IBar2yy, IBar2zz, 0, 0, 0 )
        Bar3.SetMass( MBar3 )
        Bar3.SetInertia( Bar3cm, IBar3xx, IBar3yy, IBar3zz, 0, 0, 0 )
        Link2.SetMass( MLink2 )
        Link2.SetInertia( Link2cm, ILink2xx, ILink2yy, ILink2zz, 0, 0, 0 )
        EndEffector.SetMass( MEndEffector )

% Set derivatives of trolley motion
        SetDt( y1'' = a_trol1)
        SetDt( y2'' = a_trol2)
        SetDt( x1'' = a_bridge)
        SetDt( x2'' = a_bridge)

% Set derivatives for robot motion
        SetDt( BasePos'' = Base_Acceleration )
        SetDt( qBar3'' = AngAccBar3 )
        SetDt( qLink1'' = AngAccLink1 )


%-----------------------------------------------------------------------
```

```
% KINEMATICS
%------------------------------------------------------------------------

% Movement of the two trolleys in 3D space. Only move in X/Y - no Z motion
       T1.Translate(No, x1*Nx> + y1*Ny>)
       T2.Translate(No, x2*Nx> + y2*Ny>)

% Rotation of two cables
       Cable1.Rotate(N, BodyXYZ, theta_1, phi_1, 0) % Cable 1 (E)
       Cable2.Rotate(N, BodyXYZ, theta_2, phi_2, 0) % Cable 2 (W)

% Movement of hook/cable-link connections
       P1.Translate(No,p_No_T1> - LE*Cable1z>)
       P2.Translate(No,p_No_T2> - LW*Cable2z>)

       Link.Rotate( N, BodyXYZ, beta, psi, gamma ) % Rotate the payload
          link
       LinkCM.Translate( No, (p_No_P1> + LCx*Linkx> + LCy*Linky> + LCz*
          Linkz>) )

       Base.Rotate( Link, [0,1, 0; 0, 0, 1; 1, 0, 0] )
       Link1.RotateX( Base, qLink1 )
       Link2.RotateX( Link1, qLink2 )
       Bar2.RotateX( Link2, qBar2 )
       Bar3.RotateX( Base, qBar3 )
       Basecm.Translate( P1, BasePos*Linky> + LBase/2*Basey> + ...
              (LCz+RobotOffset)*Linkz> )
       Link1cm.Translate( Basecm, LBase/2*Basey> + LLink1/2*Link1y> )
       Bar3cm.Translate( Basecm, LBar3/2*Bar3y> )
```

```
        Bar2cm.Translate( Bar3cm, LBar3/2*Bar3y> + LBar2/2*Bar2y> )
        PB1L2.Translate( Link1cm, LLink1/2*Link1y> + Distance1*Link2y> )
        Link2cm.Translate( Link1cm, LLink1/2*Link1y> + LLink2/2*Link2y> )
        EndEffector.Translate( Link2cm, LLink2/2*Link2y> )

% Save hooks/cable-link connection for checking Eq of Motion
        P1_x = Dot( p_No_P1>, Nx> )
        P1_y = Dot( p_No_P1>, Ny> )
        P1_z = Dot( p_No_P1>, Nz> )

        P2_x = Dot( p_No_P2>, Nx> )
        P2_y = Dot( p_No_P2>, Ny> )
        P2_z = Dot( p_No_P2>, Nz> )

% Save position of End Effector for outputting
        EndEffector_x = Dot(p_No_EndEffector>,Nx>)
        EndEffector_y = Dot(p_No_EndEffector>,Ny>)
        EndEffector_z = Dot(p_No_EndEffector>,Nz>)



%-----------------------------------------------------------------------
% CONSTRAINTS
%-----------------------------------------------------------------------
% Variables that are going to be in equations of motion

% Crane
        SetGeneralizedSpeed(theta_1',theta_2',beta',phi_1',phi_2',gamma',psi
            ')
% Robot
```

```
        SetGeneralizedSpeed( qLink2', qBar2')

%------ Equations of motion
% Set up 4-bar linkage velocity constraints in X, Y, Z directions.
% The velocity in each direction = 0
        Loop1> = p_No_T1> - LE*Cable1z> + LC*Linky> + LW*Cable2z> - p_No_T2>
        Dependent[1] = Dot( Dt( Loop1>, N), Nx>)
        Dependent[2] = Dot( Dt( Loop1>, N), Ny>)
        Dependent[3] = Dot( Dt( Loop1>, N), Nz>)

        Loop2> = Distance1*Link2y> - LBar2*Bar2y> - LBar3*Bar3y> + LLink1*
            Link1y>
        Dependent[4] = Dot( Dt(Loop2>,Link1), Link1y> )
        Dependent[5] = Dot( Dt(Loop2>,Link1), Link1z> )
        Constrain(Dependent[theta_2', beta', gamma', qLink2', qBar2'])


%-----------------------------------------------------------------------
% FORCES
%-----------------------------------------------------------------------

% External Forces
        System.AddForceGravity( -g*Nz> )
        EndEffector.AddForce(EEForcex*Nx>)
        EndEffector.AddForce(EEForcey*Ny>)
        EndEffector.AddForce(EEForcez*Nz>)
% Rotary damping between the trolleys and cable links
        Cable1.AddTorque( -B_cable1 * Cable1.GetAngularVelocity(N) )
        Cable2.AddTorque( -B_cable2 * Cable2.GetAngularVelocity(N) )
```

```
% Inputs from robot motors
%       Link1.SetAngularVelocity( Base, qLink1'*Basex> )
%       Bar3.SetAngularVelocity( Base, qBar3'*Basex> )
%       Basecm.SetVelocity( Link, BasePos'*Linky>)


%----------------------------------------------------------------------
% EQUATIONS OF MOTION
%----------------------------------------------------------------------

% Kane's method dynamics - gives equations of motion
        Dynamics = System.GetDynamicsKane()


%----------------------------------------------------------------------
% SIMPLIFY AND SOLVE
%----------------------------------------------------------------------

        Solve( Dynamics, theta_1'', phi_1'', phi_2'', psi'')

        PotentialEnergy = System.GetForceGravityPotentialEnergy(-9.81*Nz>,
            No)
        KineticEnergy = System.GetKineticEnergy(N)
%----------------------------------------------------------------------
% CREATE MATLAB FILES AND SAVE OUTPUT
%----------------------------------------------------------------------

%----- Setup parameters to pass to Matlab Code
```

```
% Integration parameters
        Input tFinal=10,   integStp=0.02,  absError=1.0E-07, relError=1.0E
           -07

% Constant values
        Input LE = 2 m, LW = 2 m, LC = 3 m, MC = 10 kg, ME = 10 kg, MW = 10
           kg
        Input x1 = 0 m, x2 = 0 m, x1' = 0 m/s, x2' = 0 m/s
        Input y1 = 0 m, y2 = 3 m, y1' = 0 m/s, y2' = 0 m/s
        Input g = 9.81 m/s^2
        Input B_cable1 = 0.0, B_cable2 = 0.0

% Initial conditions
        Input theta_1 = 0 deg, theta_2 = 0 deg, theta_1' = 0 deg/sec, beta =
            0 deg
        Input phi_1 = 0 deg, phi_2 = 0 deg, phi_1' = 0 deg/sec, phi_2' = 0
           deg/sec
        Input gamma = 0 deg
        Input psi = 0 deg, psi' = 0 deg/sec

% Quantities to output
        Output t, x1 m, x2 m, y1 m, y2 m, P1_x m, P1_y m, P2_x m, P2_y m
        Output theta_1 deg, theta_2 deg, beta deg, theta_1', theta_2'
        Output beta' deg/sec, phi_1 deg, phi_2 deg, gamma deg, phi_1'
        Output phi_2', gamma', psi, psi', qLink2, qLink2', qLink2''
        Output EndEffector_x, EndEffector_y, EndEffector_z
        Output PotentialEnergy, KineticEnergy

% Create MATLAB code
```

```
        ODE() two_crane_planar_3dwithRobot2.m


% Save MotionGenesis Output

        save two_crane_planar_3dwithRobot2.all
```

APPENDIX B

MOTIONGENESIS OUTPUT

```
function [tSoln, ySoln] = craneWithRobot_SOLVE3(Inputs, Parameters)
%
% Solve crane and robot Equations of Motion for the case with
% the given inputs (Uses ODE45 to solve)
%
% ************************************************************************
%
% AUTHOR: John Harber (jharber6@gatech.edu)
% Based on code by Ehsan Maleki and C.J. Adams
% DATE: 3/28/2015
%
% MODIFIED: 11/23/15
% BY: John Harber
% MODIFICATIONS: Potential and Kinetic energy calculations for the system
% were added.
%
% ************************************************************************

%% Function Handle for Equation of Motion in (Nonlinear) State-Space form

craneWithRobotDEOM = @(t,y) craneWithRobot_DEOM3(t, y, Inputs, Parameters);

%% Simulate System for the given input parameters
```

```
tSim = Inputs.tSim; % [s]
t0 = tSim(1); tf = tSim(end); % [s]
x0 = Parameters.InitialConditions; % initial conditions

% set "options" to prescribe tolerances for accuracy.
% Reduce reltol and abstol for more accurate results.
  options = odeset('reltol',1e-7,'abstol',1e-8,'maxstep',(tf-t0)/500);

  [tSoln, ySoln]= ode45(craneWithRobotDEOM,[t0, tf],x0,options);
% [tSoln, ySoln]= ode45(craneWithRobotDEOM,[t0, tf],x0);
% tSoln is a px1 column vector of times (ranging from t0 to tf)
% ySoln is a px2 matrix; the first column gives x1, the second x2, etc


%% Evaluate additonal output quantities

% Extract needed quantities
    BasePos = ySoln(:,1);
    beta  = ySoln(:,2);
    gamma  = ySoln(:,3);
    phi_1 = ySoln(:,4);
    phi_2 = ySoln(:,5);
    psi = ySoln(:,6);
    qBar2 = ySoln(:,7);
    qBar3 = ySoln(:,8);
    qLink1 = ySoln(:,9);
    qLink2 = ySoln(:,10);
    theta_1 = ySoln(:,11);
```

```
theta_2 = ySoln(:,12);

x1 = ySoln(:,13);

x2 = ySoln(:,14);

y1 = ySoln(:,15);

y2 = ySoln(:,16);

BasePosp = ySoln(:,17);

phi_1p = ySoln(:,18);

phi_2p = ySoln(:,19);

psip = ySoln(:,20);

qBar3p = ySoln(:,21);

qLink1p = ySoln(:,22);

theta_1p = ySoln(:,23);

x1p = ySoln(:,24);

x2p = ySoln(:,25);

y1p = ySoln(:,26);

y2p = ySoln(:,27);


B_cable1 = Parameters.B_cable1;

B_cable2 = Parameters.B_cable2;

EEForcex = Parameters.EEForcex;

EEForcey = Parameters.EEForcey;

EEForcez = Parameters.EEForcez;

Distance1 = Parameters.Distance1;

g = Parameters.g;          % [m/s^2]

IBar2xx = Parameters.IBar2xx;

IBar2yy = Parameters.IBar2yy;

IBar2zz = Parameters.IBar2zz;

IBar3xx = Parameters.IBar3xx;

IBar3yy = Parameters.IBar3yy;
```

```
IBar3zz = Parameters.IBar3zz;

IBasexx = Parameters.IBasexx;

IBaseyy = Parameters.IBaseyy;

IBasezz = Parameters.IBasezz;

ILink1xx = Parameters.ILink1xx;

ILink1yy = Parameters.ILink1yy;

ILink1zz = Parameters.ILink1zz;

ILink2xx = Parameters.ILink2xx;

ILink2yy = Parameters.ILink2yy;

ILink2zz = Parameters.ILink2zz;

Ixx = Parameters.Ixx;

Ixy = Parameters.Ixy;

Iyy = Parameters.Iyy;

Iyz = Parameters.Iyz;

Izx = Parameters.Izx;

Izz = Parameters.Izz;

LBar2 = Parameters.LBar2;

LBar3 = Parameters.LBar3;

LBase = Parameters.LBase;

LC = Parameters.LC;

LCx = Parameters.LCx;

LCy = Parameters.LCy;

LCz = Parameters.LCz;

LE = Parameters.LE;

LLink1 = Parameters.LLink1;

LLink2 = Parameters.LLink2;

LW = Parameters.LW;

MBar2 = Parameters.MBar2;

MBar3 = Parameters.MBar3;
```

```
    MBase = Parameters.MBase;
    MC = Parameters.MC;
    ME = Parameters.ME;
    MEndEffector = Parameters.MEndEffector;
    MLink1 = Parameters.MLink1;
    MLink2 = Parameters.MLink2;
    MW = Parameters.MW;
    RobotOffset = Parameters.RobotOffset;
    Base_Acceleration = Inputs.Base_Acceleration;
    AngAccLink1 = Inputs.AngAccLink1;
    AngAccBar3 = Inputs.AngAccBar3;
    a_bridge = Inputs.a_bridge;
    a_trol1 = Inputs.a_trol1;
    a_trol2 = Inputs.a_trol2;

    % Interpolate to find input values at each solution time step
    qLink1pp = interp1(tSim, AngAccLink1, tSoln);
    qBar3pp = interp1(tSim, AngAccBar3, tSoln);
    BasePospp = interp1(tSim, Base_Acceleration, tSoln);
    y1pp = interp1(tSim, a_trol1, tSoln);
    y2pp = interp1(tSim, a_trol2, tSoln);
    x1pp = interp1(tSim, a_bridge, tSoln);
    x2pp = interp1(tSim, a_bridge, tSoln);

%     BasePospp = Base_Acceleration;
%     qLink1pp = AngAccLink1;
%     qBar3pp = AngAccBar3;
%     y1pp = a_trol1;
%     y2pp = a_trol2;
```

```
%     x2pp = a_bridge;
%     x1pp = a_bridge;


    DEGtoRAD = pi / 180.0;
    RADtoDEG = 180.0 / pi;
%     beta = beta * DEGtoRAD;
%     psi = psi * DEGtoRAD;
%     theta_1 = theta_1 * DEGtoRAD;
%     theta_2 = theta_2 * DEGtoRAD;
%     psip = psip * DEGtoRAD;
%     theta_1p = theta_1p * DEGtoRAD;


% Evaluate constants
   z(142) = LCz + RobotOffset + 0.5*LBase;
z(302) = 0.5*LBase + z(142);
z(414) = g*MBar2;
z(415) = g*MBar3;
z(416) = g*MBase;
z(417) = g*MEndEffector;
z(418) = g*MLink1;
z(419) = g*MLink2;
z(420) = g*MC;
z(421) = g*ME;
z(422) = g*MW;
z(423) = EEForcez - z(417);
z(634) = MBar2 + MBar3 + MBase + MC + ME + MEndEffector + MLink1 + MLink2;
z(640) = MBar2 + MBar3 + MBase + MEndEffector + MLink1 + MLink2;
z(642) = LBar3*(MBar3+2*MBar2);
z(644) = LBar2*MBar2;
```

```
z(645) = LLink1*MLink1;

z(647) = LLink2*MLink2;

z(648) = LLink2*MEndEffector;

z(655) = LE*(MBar2+MBar3+MBase+MC+ME+MEndEffector+MLink1+MLink2);

z(658) = LE*(MBar2+MBar3+MBase+MEndEffector+MLink1+MLink2);

z(660) = LE^2*(MBar2+MBar3+MBase+MC+ME+MEndEffector+MLink1+MLink2);

z(661) = LBar3*LE*(MBar3+2*MBar2);

z(665) = LBar2*LE*MBar2;

z(668) = MW*LW^2;

z(669) = LW*MW;

z(682) = LBar3*MBar3;

z(683) = LBar3*MBar2;

z(700) = IBaseyy + Izz + MC*(LCx^2+LCy^2);

z(876) = LE*z(423);

z(877) = EEForcex*LE;

z(878) = EEForcey*LE;

z(879) = LE*z(421);

z(932) = LE*ME;


 gammap = zeros(length(gamma),1);

 betap = zeros(length(gamma),1);

 qBar2p = zeros(length(gamma),1);

 qLink2p = zeros(length(gamma),1);

 theta_2p = zeros(length(gamma),1);

 P1_x = zeros(length(gamma),1);

 P1_y = zeros(length(gamma),1);

 P2_x = zeros(length(gamma),1);

 P2_y = zeros(length(gamma),1);

 EndEffector_x = zeros(length(gamma),1);
```

```
 EndEffector_y = zeros(length(gamma),1);
 EndEffector_z = zeros(length(gamma),1);
 theta_1pp = zeros(length(gamma),1);
 phi_1pp = zeros(length(gamma),1);
 phi_2pp = zeros(length(gamma),1);
 psipp = zeros(length(gamma),1);
 KineticEnergy = zeros(length(gamma),1);
 PotentialEnergy = zeros(length(gamma),1);


 for i = 1:length(gamma)
z(44) = sin(gamma(i));
z(146) = z(142)*z(44);
z(2) = sin(phi_1(i));
z(3) = sin(theta_1(i));
z(4) = z(2)*z(3);
z(41) = cos(gamma(i));
z(45) = cos(beta(i));
z(46) = sin(beta(i));
z(47) = sin(psi(i));
z(51) = z(41)*z(45) - z(44)*z(46)*z(47);
z(1) = cos(phi_1(i));
z(42) = cos(psi(i));
z(50) = z(42)*z(44);
z(5) = cos(theta_1(i));
z(6) = z(2)*z(5);
z(52) = z(41)*z(46) + z(44)*z(45)*z(47);
z(450) = z(4)*z(51) - z(1)*z(50) - z(6)*z(52);
z(143) = z(142)*z(41);
```

```
z(43) = z(41)*z(42);

z(48) = z(44)*z(45) + z(41)*z(46)*z(47);

z(49) = z(44)*z(46) - z(41)*z(45)*z(47);

z(472) = z(1)*z(43) + z(4)*z(48) - z(6)*z(49);

z(148) = BasePos(i)*z(44);

z(53) = z(42)*z(46);

z(54) = z(42)*z(45);

z(471) = z(1)*z(47) - z(4)*z(53) - z(6)*z(54);

z(66) = LCz*z(44);

z(64) = LCy*z(44) - LCx*z(41);

z(67) = LCz*z(41);

z(84) = cos(qLink1(i));

z(89) = z(44)*z(84);

z(96) = cos(qLink2(i));

z(85) = sin(qLink1(i));

z(87) = z(44)*z(85);

z(97) = sin(qLink2(i));

z(103) = z(89)*z(96) - z(87)*z(97);

z(113) = cos(qBar2(i));

z(100) = z(87)*z(96) + z(89)*z(97);

z(114) = sin(qBar2(i));

z(120) = z(103)*z(113) - z(100)*z(114);

z(220) = LBar2*z(120);

z(194) = LBar3*z(41);

z(130) = cos(qBar3(i));

z(131) = sin(qBar3(i));

z(313) = z(84)*z(130) + z(85)*z(131);

z(88) = z(43)*z(84) - z(47)*z(85);

z(305) = z(48)*z(84) + z(53)*z(85);
```

```
z(306) = z(49)*z(84) - z(54)*z(85);

z(452) = z(1)*z(88) + z(4)*z(305) - z(6)*z(306);

z(314) = z(85)*z(130) - z(84)*z(131);

z(86) = z(43)*z(85) + z(47)*z(84);

z(303) = z(48)*z(85) - z(53)*z(84);

z(304) = z(49)*z(85) + z(54)*z(84);

z(451) = z(1)*z(86) + z(4)*z(303) - z(6)*z(304);

z(466) = z(313)*z(452) + z(314)*z(451);

z(135) = z(44)*z(130);

z(198) = LBar3*z(135);

z(200) = -z(146) - 0.5*z(198);

z(222) = z(200) - 0.5*z(198);

z(216) = LBar2*z(41);

z(316) = z(96)*z(113) - z(97)*z(114);

z(318) = -z(96)*z(114) - z(97)*z(113);

z(460) = z(316)*z(452) + z(318)*z(451);

z(168) = LLink1*z(41);

z(167) = -0.5*LBase*z(44) - 0.5*LLink1*z(89);

z(171) = z(167) - z(146);

z(163) = LBase*z(41);

z(172) = z(143) + 0.5*z(163);

z(272) = LLink2*z(41);

z(102) = z(88)*z(96) - z(86)*z(97);

z(309) = z(96)*z(305) - z(97)*z(303);

z(310) = z(96)*z(306) - z(97)*z(304);

z(622) = z(1)*z(102) + z(4)*z(309) - z(6)*z(310);

z(271) = -0.5*LLink1*z(89) - 0.5*LLink2*z(103);

z(275) = z(171) + z(271);

z(294) = LLink2*z(103);
```

```
z(653) = LE*(2*MBase*(z(146)*z(450)-z(143)*z(472)-z(148)*z(471))+2*MC*(z
    (66)*z(450)-z(64)*z(471)-z(67)*z(472))+MBar2*(z(220)*z(450)-2*z(143)*z
    (472)-2*z(148)*z(471)-2*z(194)*z(466)-2*z(222)*z(450)-z(216)*z(460))-
    MBar3*(z(194)*z(466)+2* ...
z(143)*z(472)+2*z(148)*z(471)+2*z(200)*z(450))-MLink1*(z(168)*z(452)+2*z
    (148)*z(471)+2*z(171)*z(450)+2*z(172)*z(472))-MLink2*(z(272)*z(622)+2*z
    (148)*z(471)+2*z(168)*z(452)+2*z(172)*z(472)+2*z(275)*z(450))-
    MEndEffector*(2*z(148)*z(471)+ ...
2*z(168)*z(452)+2*z(172)*z(472)+2*z(272)*z(622)+2*z(275)*z(450)-z(294)*z
    (450)));
z(319) = LC*z(44)*z(47);
z(320) = LC*z(43);
z(342) = z(319)/z(320);
z(101) = -z(84)*z(97) - z(85)*z(96);
z(269) = 0.5*LLink1*z(85) - 0.5*LLink2*z(101);
z(165) = LLink1*z(85);
z(276) = z(269) + 0.5*z(165);
z(196) = LBar3*z(131);
z(98) = z(84)*z(96) - z(85)*z(97);
z(118) = z(101)*z(113) - z(98)*z(114);
z(218) = LBar2*z(118);
z(292) = LLink2*z(101);
z(654) = LE*(2*MBase*BasePos(i)*z(472)+2*MLink2*(BasePos(i)*z(472)-z(276)*z
    (450))+MBar3*(2*BasePos(i)*z(472)-z(196)*z(450))+MLink1*(2*BasePos(i)*z
    (472)-z(165)*z(450))-2*MC*(LCx*z(450)-LCy*z(472))-MBar2*(2*z(196)*z
    (450)-2*BasePos(i)*z(472)-z(218)* ...
z(450))-MEndEffector*(2*z(276)*z(450)-2*BasePos(i)*z(472)-z(292)*z(450)));
z(13) = cos(phi_2(i));
z(17) = cos(theta_2(i));
```

```
z(324) = LW*z(13)*z(17);

z(334) = LC*z(49);

z(331) = LC*z(44)*z(54);

z(351) = z(334)*z(342) - z(331);

z(15) = sin(theta_2(i));

z(333) = LW*z(13)*z(15);

z(327) = LC*z(44)*z(53);

z(325) = LC*z(48);

z(348) = z(327) + z(325)*z(342);

z(330) = LC*(z(43)*z(54)-z(47)*z(49));

z(326) = LC*(z(43)*z(53)+z(47)*z(48));

z(345) = -z(324)*z(330) - z(326)*z(333);

z(364) = (z(324)*z(351)-z(333)*z(348))/z(345);

z(145) = z(142)*z(43);

z(144) = -z(142)*z(50) - BasePos(i)*z(47);

z(147) = BasePos(i)*z(43);

z(195) = LBar3*z(50);

z(134) = z(43)*z(130) - z(47)*z(131);

z(197) = LBar3*z(134);

z(199) = -z(145) - 0.5*z(197);

z(169) = LLink1*z(50);

z(166) = -0.5*LBase*z(43) - 0.5*LLink1*z(88);

z(170) = z(166) - z(145);

z(164) = LBase*z(50);

z(173) = z(144) - 0.5*z(164);

z(273) = LLink2*z(50);

z(270) = -0.5*LLink1*z(88) - 0.5*LLink2*z(102);

z(274) = z(170) + z(270);

z(217) = LBar2*z(50);
```

```
z(99) = z(86)*z(96) + z(88)*z(97);

z(119) = z(102)*z(113) - z(99)*z(114);

z(219) = LBar2*z(119);

z(221) = z(199) - 0.5*z(197);

z(63) = LCx*z(50) + LCy*z(43);

z(65) = LCx*z(47) - LCz*z(43);

z(68) = -LCy*z(47) - LCz*z(50);

z(293) = LLink2*z(102);

z(652) = LE*(2*MBase*(z(145)*z(450)-z(144)*z(472)-z(147)*z(471))+MBar3*(z
    (195)*z(466)-2*z(144)*z(472)-2*z(147)*z(471)-2*z(199)*z(450))+MLink1*(z
    (169)*z(452)-2*z(147)*z(471)-2*z(170)*z(450)-2*z(173)*z(472))+MLink2*(z
    (273)*z(622)+2* ...
z(169)*z(452)-2*z(147)*z(471)-2*z(173)*z(472)-2*z(274)*z(450))+MBar2*(z
    (217)*z(460)+z(219)*z(450)+2*z(195)*z(466)-2*z(144)*z(472)-2*z(147)*z
    (471)-2*z(221)*z(450))-2*MC*(z(63)*z(471)+z(65)*z(450)+z(68)*z(472))-
    MEndEffector*(2*z(147)* ...
z(471)+2*z(173)*z(472)+2*z(274)*z(450)-2*z(169)*z(452)-2*z(273)*z(622)-z
    (293)*z(450)));

z(779) = 0.5*z(653) + 0.5*z(342)*z(654) - 0.5*z(364)*z(652);

z(328) = LE*z(6);

z(25) = LE*z(1);

z(344) = z(25)/z(320);

z(350) = -z(328) - z(334)*z(344);

z(323) = LE*z(4);

z(347) = z(323) - z(325)*z(344);

z(357) = (z(326)*z(350)+z(330)*z(347))/z(345);

z(358) = (z(326)*z(351)+z(330)*z(348))/z(345);

z(33) = LW*z(13);

z(686) = MW*z(33)^2;
```

```
z(784) = z(358)*z(686);

z(584) = ILink1yy*z(84);

z(610) = ILink2yy*z(98);

z(614) = ILink2zz*z(101);

z(117) = z(100)*z(113) + z(103)*z(114);

z(115) = z(98)*z(113) + z(101)*z(114);

z(439) = IBar2yy*z(115);

z(443) = IBar2zz*z(118);

z(133) = z(44)*z(131);

z(517) = IBar3yy*z(130);

z(484) = z(84)*z(316) + z(85)*z(318);

z(499) = z(52)*z(54) - z(47)*z(50) - z(51)*z(53);

z(590) = ILink1zz*z(85);

z(523) = IBar3zz*z(131);

z(673) = Iyz*z(41) + Izx*z(44) + z(87)*z(584) + z(100)*z(610) + z(103)*z
    (614) + z(117)*z(439) + z(120)*z(443) + z(133)*z(517) + 0.5*MBar3*(z
    (196)*z(200)-2*BasePos(i)*z(143)-BasePos(i)*z(130)*z(194)) + 0.5*MLink1
    *(z(165)*z(171)-2*BasePos(i)* ...
z(172)-BasePos(i)*z(84)*z(168)) + 0.5*MLink2*(2*z(275)*z(276)-2*BasePos(i)*
    z(172)-2*BasePos(i)*z(84)*z(168)-BasePos(i)*z(98)*z(272)) + 0.25*MBar2
    *(z(218)*z(220)+4*z(196)*z(222)-4*BasePos(i)*z(143)-2*z(196)*z(220)-2*z
    (218)*z(222)-4*BasePos(i)*z(130)* ...
z(194)-2*BasePos(i)*z(216)*z(484)-2*z(148)*z(218)*z(499)) + 0.25*
    MEndEffector*(z(292)*z(294)+4*z(275)*z(276)-4*BasePos(i)*z(172)-2*z
    (275)*z(292)-2*z(276)*z(294)-4*BasePos(i)*z(84)*z(168)-4*BasePos(i)*z
    (98)*z(272)-2*z(148)*z(292)*z(499)) - z(89)* ...
z(590) - z(135)*z(523) - MBase*BasePos(i)*z(143) - MC*(LCx*z(66)+LCy*z(67))
    ;
```

```
z(701) = z(700) + MBase*BasePos(i)^2 + z(84)*z(584) + z(85)*z(590) + z(98)*
    z(610) + z(101)*z(614) + z(115)*z(439) + z(118)*z(443) + z(130)*z(517)
    + z(131)*z(523) + MLink2*(BasePos(i)^2+z(276)^2) + 0.25*MBar3*(4*
    BasePos(i)^2+z(196)^2) + 0.25* ...
MLink1*(4*BasePos(i)^2+z(165)^2) - 0.25*MBar2*(4*z(196)*z(218)-4*BasePos(i)
    ^2-4*z(196)^2-z(218)^2) - 0.25*MEndEffector*(4*z(276)*z(292)-4*BasePos(
    i)^2-4*z(276)^2-z(292)^2);
z(116) = z(99)*z(113) + z(102)*z(114);
z(132) = z(43)*z(131) + z(47)*z(130);
z(689) = IBaseyy*z(47) + Izx*z(43) + Izz*z(47) + z(86)*z(584) + z(99)*z
    (610) + z(102)*z(614) + z(116)*z(439) + z(119)*z(443) + z(132)*z(517) +
     MC*(LCx*z(65)-LCy*z(68)) + 0.25*MBar2*(z(218)*z(219)+4*z(196)*z(221)
    +2*BasePos(i)*z(217)* ...
z(484)+4*BasePos(i)*z(130)*z(195)-4*BasePos(i)*z(144)-2*z(196)*z(219)-2*z
    (218)*z(221)-2*z(147)*z(218)*z(499)) + 0.25*MEndEffector*(z(292)*z(293)
    +4*z(274)*z(276)+4*BasePos(i)*z(84)*z(169)+4*BasePos(i)*z(98)*z(273)-4*
    BasePos(i)*z(173)-2*z(274)*z(292)-2* ...
z(276)*z(293)-2*z(147)*z(292)*z(499)) - Iyz*z(50) - z(88)*z(590) - z(134)*z
    (523) - MBase*BasePos(i)*z(144) - 0.5*MBar3*(2*BasePos(i)*z(144)-z(196)
    *z(199)-BasePos(i)*z(130)*z(195)) - 0.5*MLink1*(2*BasePos(i)*z(173)-z
    (165)*z(170)-BasePos(i)*z(84)* ...
z(169)) - 0.5*MLink2*(2*BasePos(i)*z(173)-2*z(274)*z(276)-2*BasePos(i)*z
    (84)*z(169)-BasePos(i)*z(98)*z(273));
z(772) = z(673) + z(342)*z(701) - z(364)*z(689);
z(363) = (z(324)*z(350)-z(333)*z(347))/z(345);
z(551) = Ixy*z(43);
z(568) = Iyz*z(47);
z(539) = IBasezz*z(43);
z(548) = Ixx*z(43);
```

```
z(566) = Izx*z(47);

z(585) = ILink1yy*z(86);

z(588) = ILink1zz*z(88);

z(611) = ILink2yy*z(99);

z(615) = ILink2zz*z(102);

z(440) = IBar2yy*z(116);

z(444) = IBar2zz*z(119);

z(518) = IBar3yy*z(132);

z(521) = IBar3zz*z(134);

z(492) = z(49)*z(130) - z(54)*z(131);

z(491) = z(48)*z(130) + z(53)*z(131);

z(507) = z(47)*z(134) + z(54)*z(492) - z(53)*z(491);

z(600) = z(47)*z(88) + z(54)*z(306) - z(53)*z(305);

z(633) = z(47)*z(102) + z(54)*z(310) - z(53)*z(309);

z(483) = z(84)*z(318) - z(85)*z(316);

z(308) = z(96)*z(304) + z(97)*z(306);

z(488) = z(113)*z(310) - z(114)*z(308);

z(307) = z(96)*z(303) + z(97)*z(305);

z(487) = z(113)*z(309) - z(114)*z(307);

z(501) = z(47)*z(119) + z(54)*z(488) - z(53)*z(487);

z(480) = z(313)*z(316) + z(314)*z(318);

z(437) = IBar2xx*z(50);

z(515) = IBar3xx*z(50);

z(535) = IBasexx*z(50);

z(561) = Iyy*z(50);

z(582) = ILink1xx*z(50);

z(608) = ILink2xx*z(50);

z(558) = Ixy*z(50);
```

```
z(674) = z(41)*z(551) + z(41)*z(568) + z(44)*z(539) + z(44)*z(548) + z(44)*
    z(566) + z(87)*z(585) + z(89)*z(588) + z(100)*z(611) + z(103)*z(615) +
    z(117)*z(440) + z(120)*z(444) + z(133)*z(518) + z(135)*z(521) + MBase*(
    z(143)*z(144)+ ...
z(145)*z(146)+z(147)*z(148)) + 0.25*MBar3*(4*z(143)*z(144)+4*z(147)*z(148)
    +4*z(199)*z(200)+2*z(130)*z(144)*z(194)-z(194)*z(195)-2*z(130)*z(143)*z
    (195)-2*z(131)*z(147)*z(194)-2*z(148)*z(195)*z(507)) + 0.25*MLink1*(4*z
    (147)*z(148)+4* ...
z(170)*z(171)+4*z(172)*z(173)+2*z(84)*z(168)*z(173)-z(168)*z(169)-2*z(84)*z
    (169)*z(172)-2*z(85)*z(147)*z(168)-2*z(148)*z(169)*z(600)) + 0.25*
    MLink2*(4*z(147)*z(148)+4*z(172)*z(173)+4*z(274)*z(275)+2*z(98)*z(173)*
    z(272)+2*z(101)*z(147)* ...
z(272)+4*z(84)*z(168)*z(173)-4*z(168)*z(169)-z(272)*z(273)-4*z(84)*z(169)*z
    (172)-4*z(85)*z(147)*z(168)-4*z(148)*z(169)*z(600)-2*z(96)*z(168)*z
    (273)-2*z(96)*z(169)*z(272)-2*z(98)*z(172)*z(273)-2*z(148)*z(273)*z
    (633)) + 0.25*MBar2*( ...
z(219)*z(220)+4*z(143)*z(144)+4*z(147)*z(148)+4*z(221)*z(222)+2*z(144)*z
    (216)*z(484)+2*z(147)*z(216)*z(483)+4*z(130)*z(144)*z(194)-4*z(194)*z
    (195)-2*z(219)*z(222)-2*z(220)*z(221)-z(216)*z(217)-4*z(130)*z(143)*z
    (195)-4*z(131)*z(147)* ...
z(194)-4*z(148)*z(195)*z(507)-2*z(143)*z(217)*z(484)-2*z(148)*z(217)*z(501)
    -2*z(148)*z(219)*z(499)-2*z(194)*z(217)*z(480)-2*z(195)*z(216)*z(480))
    + 0.25*MEndEffector*(z(293)*z(294)+4*z(147)*z(148)+4*z(172)*z(173)+4*z
    (274)*z(275)+4* ...
z(84)*z(168)*z(173)+4*z(98)*z(173)*z(272)+4*z(101)*z(147)*z(272)-4*z(168)*z
    (169)-4*z(272)*z(273)-2*z(274)*z(294)-2*z(275)*z(293)-4*z(84)*z(169)*z
    (172)-4*z(85)*z(147)*z(168)-4*z(96)*z(168)*z(273)-4*z(96)*z(169)*z(272)
    -4*z(98)*z(172)* ...
```

```
z(273)-4*z(148)*z(169)*z(600)-4*z(148)*z(273)*z(633)-2*z(148)*z(293)*z(499)
    ) - z(41)*z(437) - z(41)*z(515) - z(41)*z(535) - z(41)*z(561) - z(41)*z
    (582) - z(41)*z(608) - z(44)*z(558) - MC*(z(65)*z(66)-z(63)*z(64)-z(67)
    *z(68));
z(537) = IBaseyy*z(47);
z(554) = Izx*z(43);
z(570) = Izz*z(47);
z(564) = Iyz*z(50);
z(702) = z(537) + z(554) + z(570) + z(84)*z(585) + z(98)*z(611) + z(101)*z
    (615) + z(115)*z(440) + z(118)*z(444) + z(130)*z(518) + MC*(LCx*z(65)-
    LCy*z(68)) - z(564) - z(85)*z(588) - z(131)*z(521) - MBase*BasePos(i)*z
    (144) - 0.5*MBar3*(2* ...
BasePos(i)*z(144)-z(196)*z(199)-BasePos(i)*z(130)*z(195)) - 0.5*MLink1*(2*
    BasePos(i)*z(173)-z(165)*z(170)-BasePos(i)*z(84)*z(169)) - 0.5*MLink2
    *(2*BasePos(i)*z(173)-2*z(274)*z(276)-2*BasePos(i)*z(84)*z(169)-BasePos
    (i)*z(98)*z(273)) - 0.25*MBar2*(2*z(196)* ...
z(219)+2*z(218)*z(221)+4*BasePos(i)*z(144)-4*z(196)*z(221)-z(218)*z(219)-4*
    BasePos(i)*z(130)*z(195)-2*BasePos(i)*z(217)*z(484)) - 0.25*
    MEndEffector*(2*z(274)*z(292)+2*z(276)*z(293)+4*BasePos(i)*z(173)-4*z
    (274)*z(276)-z(292)*z(293)-4*BasePos(i)*z(84)* ...
z(169)-4*BasePos(i)*z(98)*z(273));
z(690) = z(43)*z(539) + z(43)*z(548) + z(43)*z(566) + z(47)*z(537) + z(47)*
    z(554) + z(47)*z(570) + z(50)*z(437) + z(50)*z(515) + z(50)*z(535) + z
    (50)*z(561) + z(50)*z(582) + z(50)*z(608) + z(86)*z(585) + z(88)*z(588)
     + z(99)*z(611) + ...
z(102)*z(615) + z(116)*z(440) + z(119)*z(444) + z(132)*z(518) + z(134)*z
    (521) + MBase*(z(144)^2+z(145)^2+z(147)^2) + MC*(z(63)^2+z(65)^2+z(68)
    ^2) + 0.25*MBar3*(z(195)^2+4*z(144)^2+4*z(147)^2+4*z(199)^2+2*z(131)*z
    (147)*z(195)-4*z(130)* ...
```

```
z(144)*z(195)-2*z(147)*z(195)*z(507)) + 0.25*MLink1*(z(169)^2+4*z(147)^2+4*
    z(170)^2+4*z(173)^2+2*z(85)*z(147)*z(169)-4*z(84)*z(169)*z(173)-2*z
    (147)*z(169)*z(600)) + 0.25*MLink2*(z(273)^2+4*z(147)^2+4*z(169)^2+4*z
    (173)^2+4*z(274)^2+4* ...
z(85)*z(147)*z(169)+4*z(96)*z(169)*z(273)-8*z(84)*z(169)*z(173)-4*z(98)*z
    (173)*z(273)-4*z(147)*z(169)*z(600)-2*z(101)*z(147)*z(273)-2*z(147)*z
    (273)*z(633)) + 0.25*MEndEffector*(z(293)^2+4*z(147)^2+4*z(169)^2+4*z
    (173)^2+4*z(273)^2+4* ...
z(274)^2+4*z(85)*z(147)*z(169)+8*z(96)*z(169)*z(273)-4*z(274)*z(293)-8*z
    (84)*z(169)*z(173)-8*z(98)*z(173)*z(273)-4*z(101)*z(147)*z(273)-4*z
    (147)*z(169)*z(600)-4*z(147)*z(273)*z(633)-2*z(147)*z(293)*z(499)) - z
    (43)*z(558) - z(47)* ...
z(564) - z(50)*z(551) - z(50)*z(568) - 0.25*MBar2*(4*z(219)*z(221)+2*z(147)
    *z(217)*z(483)+2*z(147)*z(217)*z(501)+2*z(147)*z(219)*z(499)+4*z(144)*z
    (217)*z(484)+4*z(147)*z(195)*z(507)+8*z(130)*z(144)*z(195)-4*z(144)
    ^2-4*z(147)^2-4* ...
z(195)^2-4*z(221)^2-z(217)^2-z(219)^2-4*z(131)*z(147)*z(195)-4*z(195)*z
    (217)*z(480));
z(773) = z(674) + z(342)*z(702) - z(364)*z(690);
z(788) = z(779) + z(357)*z(784) - z(344)*z(772) - z(363)*z(773);
z(341) = z(33)/z(320);
z(734) = 0.5*z(654) - z(344)*z(701) - z(363)*z(689);
z(14) = sin(phi_2(i));
z(18) = z(14)*z(17);
z(332) = LW*z(18);
z(352) = z(332) + z(334)*z(341);
z(16) = z(14)*z(15);
z(321) = LW*z(16);
z(346) = z(325)*z(341) - z(321);
```

```
z(354) = (z(326)*z(352)+z(330)*z(346))/z(345);

z(743) = z(357)*z(686);

z(360) = (z(324)*z(352)-z(333)*z(346))/z(345);

z(732) = 0.5*z(652) - z(344)*z(702) - z(363)*z(690);

z(749) = z(341)*z(734) + z(354)*z(743) - z(360)*z(732);

z(555) = Izx*z(44);

z(563) = Iyz*z(41);

z(586) = ILink1yy*z(87);

z(612) = ILink2yy*z(100);

z(616) = ILink2zz*z(103);

z(441) = IBar2yy*z(117);

z(445) = IBar2zz*z(120);

z(519) = IBar3yy*z(133);

z(589) = ILink1zz*z(89);

z(522) = IBar3zz*z(135);

z(703) = z(555) + z(563) + z(84)*z(586) + z(98)*z(612) + z(101)*z(616) + z
    (115)*z(441) + z(118)*z(445) + z(130)*z(519) + 0.5*MBar3*(z(196)*z(200)
    -2*BasePos(i)*z(143)-BasePos(i)*z(130)*z(194)) + 0.5*MLink1*(z(165)*z
    (171)-2*BasePos(i)*z(172)- ...
BasePos(i)*z(84)*z(168)) + 0.5*MLink2*(2*z(275)*z(276)-2*BasePos(i)*z(172)
    -2*BasePos(i)*z(84)*z(168)-BasePos(i)*z(98)*z(272)) + 0.25*MBar2*(z
    (218)*z(220)+4*z(196)*z(222)-4*BasePos(i)*z(143)-2*z(196)*z(220)-2*z
    (218)*z(222)-4*BasePos(i)*z(130)*z(194)-2* ...
BasePos(i)*z(216)*z(484)) + 0.25*MEndEffector*(z(292)*z(294)+4*z(275)*z
    (276)-4*BasePos(i)*z(172)-2*z(275)*z(292)-2*z(276)*z(294)-4*BasePos(i)*
    z(84)*z(168)-4*BasePos(i)*z(98)*z(272)) - z(85)*z(589) - z(131)*z(522)
    - MBase*BasePos(i)*z(143) - MC*(LCx* ...
z(66)+LCy*z(67));

z(540) = IBasezz*z(44);
```

```
z(549) = Ixx*z(44);

z(557) = Ixy*z(41);

z(436) = IBar2xx*z(41);

z(514) = IBar3xx*z(41);

z(534) = IBasexx*z(41);

z(552) = Ixy*z(44);

z(560) = Iyy*z(41);

z(581) = ILink1xx*z(41);

z(607) = ILink2xx*z(41);

z(691) = z(43)*z(540) + z(43)*z(549) + z(43)*z(557) + z(47)*z(555) + z(47)*
    z(563) + z(86)*z(586) + z(88)*z(589) + z(99)*z(612) + z(102)*z(616) + z
    (116)*z(441) + z(119)*z(445) + z(132)*z(519) + z(134)*z(522) + MBase*(z
    (143)*z(144)+ ...

z(145)*z(146)+z(147)*z(148)) + 0.25*MBar2*(z(219)*z(220)+4*z(143)*z(144)+4*
    z(147)*z(148)+4*z(221)*z(222)+2*z(144)*z(216)*z(484)+2*z(147)*z(216)*z
    (501)+4*z(130)*z(144)*z(194)+4*z(131)*z(148)*z(195)+4*z(147)*z(194)*z
    (507)-4*z(194)* ...

z(195)-2*z(219)*z(222)-2*z(220)*z(221)-z(216)*z(217)-4*z(130)*z(143)*z(195)
    -2*z(143)*z(217)*z(484)-2*z(147)*z(220)*z(499)-2*z(148)*z(217)*z(483)
    -2*z(194)*z(217)*z(480)-2*z(195)*z(216)*z(480)) + 0.25*MEndEffector*(z
    (293)*z(294)+4* ...

z(147)*z(148)+4*z(172)*z(173)+4*z(274)*z(275)+4*z(84)*z(168)*z(173)+4*z(85)
    *z(148)*z(169)+4*z(98)*z(173)*z(272)+4*z(147)*z(168)*z(600)+4*z(147)*z
    (272)*z(633)-4*z(168)*z(169)-4*z(272)*z(273)-2*z(274)*z(294)-2*z(275)*z
    (293)-4*z(84)* ...

z(169)*z(172)-4*z(96)*z(168)*z(273)-4*z(96)*z(169)*z(272)-4*z(98)*z(172)*z
    (273)-4*z(101)*z(148)*z(273)-2*z(147)*z(294)*z(499)) - z(50)*z(436) - z
    (50)*z(514) - z(50)*z(534) - z(50)*z(552) - z(50)*z(560) - z(50)*z(581)
     - z(50)*z(607) - MC*( ...
```

```
z(65)*z(66)-z(63)*z(64)-z(67)*z(68)) - 0.25*MBar3*(z(194)*z(195)+2*z(130)*z
    (143)*z(195)-4*z(143)*z(144)-4*z(147)*z(148)-4*z(199)*z(200)-2*z(130)*z
    (144)*z(194)-2*z(131)*z(148)*z(195)-2*z(147)*z(194)*z(507)) - 0.25*
    MLink1*(z(168)*z(169)+ ...
2*z(84)*z(169)*z(172)-4*z(147)*z(148)-4*z(170)*z(171)-4*z(172)*z(173)-2*z
    (84)*z(168)*z(173)-2*z(85)*z(148)*z(169)-2*z(147)*z(168)*z(600)) -
    0.25*MLink2*(z(272)*z(273)+4*z(168)*z(169)+2*z(96)*z(168)*z(273)+2*z
    (96)*z(169)*z(272)+2*z(98)* ...
z(172)*z(273)+2*z(101)*z(148)*z(273)+4*z(84)*z(169)*z(172)-4*z(147)*z(148)
    -4*z(172)*z(173)-4*z(274)*z(275)-4*z(84)*z(168)*z(173)-4*z(85)*z(148)*z
    (169)-4*z(147)*z(168)*z(600)-2*z(98)*z(173)*z(272)-2*z(147)*z(272)*z
    (633));
z(756) = z(341)*z(703) - z(360)*z(691);
z(754) = z(341)*z(701) - z(360)*z(689);
z(753) = z(354)*z(686);
z(755) = z(341)*z(702) - z(360)*z(690);
z(770) = z(756) + z(342)*z(754) + z(358)*z(753) - z(364)*z(755);
z(733) = 0.5*z(653) - z(344)*z(703) - z(363)*z(691);
z(750) = z(733) + z(342)*z(734) + z(358)*z(743) - z(364)*z(732);
z(769) = z(668) + z(341)*z(754) + z(354)*z(753) - z(360)*z(755);
z(902) = z(749)*z(770) - z(750)*z(769);
z(675) = z(41)*z(436) + z(41)*z(514) + z(41)*z(534) + z(41)*z(552) + z(41)*
    z(560) + z(41)*z(581) + z(41)*z(607) + z(44)*z(540) + z(44)*z(549) + z
    (44)*z(557) + z(87)*z(586) + z(89)*z(589) + z(100)*z(612) + z(103)*z
    (616) + z(117)*z(441) + ...
z(120)*z(445) + z(133)*z(519) + z(135)*z(522) + MBase*(z(143)^2+z(146)^2+z
    (148)^2) + MC*(z(64)^2+z(66)^2+z(67)^2) + 0.25*MEndEffector*(z(294)
    ^2+4*z(148)^2+4*z(168)^2+4*z(172)^2+4*z(272)^2+4*z(275)^2+4*z(101)*z
    (148)*z(272)+4*z(148)* ...
```

```
z(168)*z(600)+4*z(148)*z(272)*z(633)+8*z(84)*z(168)*z(172)+8*z(96)*z(168)*z
    (272)+8*z(98)*z(172)*z(272)-4*z(275)*z(294)-4*z(85)*z(148)*z(168)-2*z
    (148)*z(294)*z(499)) - 0.25*MBar3*(2*z(131)*z(148)*z(194)-4*z(143)^2-4*
    z(148)^2-4*z(200)^2- ...
z(194)^2-4*z(130)*z(143)*z(194)-2*z(148)*z(194)*z(507)) - 0.25*MLink1*(2*z
    (85)*z(148)*z(168)-4*z(148)^2-4*z(171)^2-4*z(172)^2-z(168)^2-4*z(84)*z
    (168)*z(172)-2*z(148)*z(168)*z(600)) - 0.25*MLink2*(4*z(85)*z(148)*z
    (168)-4*z(148)^2-4* ...
z(168)^2-4*z(172)^2-4*z(275)^2-z(272)^2-8*z(84)*z(168)*z(172)-4*z(96)*z
    (168)*z(272)-4*z(98)*z(172)*z(272)-4*z(148)*z(168)*z(600)-2*z(101)*z
    (148)*z(272)-2*z(148)*z(272)*z(633)) - 0.25*MBar2*(4*z(220)*z(222)+2*z
    (148)*z(220)*z(499)+4* ...
z(131)*z(148)*z(194)-4*z(143)^2-4*z(148)^2-4*z(194)^2-4*z(222)^2-z(216)^2-z
    (220)^2-8*z(130)*z(143)*z(194)-4*z(143)*z(216)*z(484)-4*z(148)*z(194)*z
    (507)-4*z(194)*z(216)*z(480)-2*z(148)*z(216)*z(483)-2*z(148)*z(216)*z
    (501));
z(774) = z(675) + z(342)*z(703) - z(364)*z(691);
z(790) = z(774) + z(342)*z(772) + z(358)*z(784) - z(364)*z(773);
z(738) = z(660) - 0.5*z(344)*z(654) - 0.5*z(363)*z(652);
z(748) = z(738) + z(357)*z(743) - z(344)*z(734) - z(363)*z(732);
z(761) = 0.5*z(341)*z(654) - 0.5*z(360)*z(652);
z(768) = z(761) + z(357)*z(753) - z(344)*z(754) - z(363)*z(755);
z(900) = z(748)*z(769) - z(749)*z(768);
z(789) = z(341)*z(772) + z(354)*z(784) - z(360)*z(773);
z(901) = z(748)*z(770) - z(750)*z(768);
z(903) = z(788)*z(902) + z(790)*z(900) - z(789)*z(901);
z(28) = LE*z(2)*phi_1p(i);
z(29) = theta_1p(i)*z(28);
z(10) = z(2)*theta_1p(i);
```

```
z(26) = LE*phi_1p(i);
z(32) = -z(29) - z(10)*z(26);
z(453) = z(3)*z(52) + z(5)*z(51);
z(343) = 1/z(320);
z(371) = z(343)*(x1p(i)-x2p(i));
gammap(i) = z(341)*phi_2p(i) + z(342)*psip(i) + z(371) - z(344)*phi_1p(i);
z(152) = z(142)*z(41)*gammap(i);
z(362) = z(333)/z(345);
z(353) = z(334)*z(343);
z(349) = z(325)*z(343);
z(365) = (z(324)*z(353)-z(333)*z(349))/z(345);
z(370) = z(362)*y2p(i) + z(365)*x2p(i) - z(362)*y1p(i) - z(365)*x1p(i);
z(322) = LE*z(1)*z(5);
z(329) = LE*z(1)*z(3);
z(361) = (z(322)*z(333)-z(324)*z(329))/z(345);
betap(i) = z(370) - z(360)*phi_2p(i) - z(361)*theta_1p(i) - z(363)*phi_1p(i
    ) - z(364)*psip(i);
z(58) = -z(41)*z(47)*psip(i) - z(42)*z(44)*gammap(i);
z(153) = -psip(i)*z(152) - z(142)*betap(i)*z(58);
z(57) = gammap(i) + z(47)*betap(i);
z(150) = z(143)*psip(i) + z(144)*betap(i) - BasePos(i)*gammap(i);
z(55) = z(43)*betap(i) + z(44)*psip(i);
z(151) = z(147)*betap(i) + z(148)*psip(i);
z(160) = z(153) + z(57)*z(150) - z(55)*z(151);
z(473) = z(3)*z(54) - z(5)*z(53);
z(157) = z(43)*BasePosp(i) + BasePos(i)*z(58);
z(158) = z(44)*BasePosp(i) + BasePos(i)*z(41)*gammap(i);
z(159) = betap(i)*z(157) + psip(i)*z(158);
z(149) = BasePosp(i) - z(145)*betap(i) - z(146)*psip(i);
```

```
z(56) = z(41)*psip(i) - z(50)*betap(i);

z(162) = z(159) + z(55)*z(149) - z(56)*z(150);

z(474) = z(3)*z(49) + z(5)*z(48);

z(60) = z(41)*z(42)*gammap(i) - z(44)*z(47)*psip(i);

z(155) = -z(47)*BasePosp(i) - BasePos(i)*z(42)*psip(i) - z(142)*z(60);

z(154) = z(142)*z(44)*gammap(i);

z(156) = betap(i)*z(155) - BasePosp(i)*gammap(i) - psip(i)*z(154);

z(161) = z(156) + z(56)*z(151) - z(57)*z(149);

z(75) = LCx*z(42)*psip(i) - LCz*z(58);

z(76) = LCz*z(41)*gammap(i);

z(77) = betap(i)*z(75) - psip(i)*z(76);

z(69) = z(67)*psip(i) + z(68)*betap(i) - LCy*gammap(i);

z(71) = z(63)*betap(i) + z(64)*psip(i);

z(83) = z(77) + z(57)*z(69) - z(55)*z(71);

z(78) = LCx*z(60) + LCy*z(58);

z(79) = (LCx*z(44)+LCy*z(41))*gammap(i);

z(80) = betap(i)*z(78) + psip(i)*z(79);

z(70) = LCx*gammap(i) + z(65)*betap(i) - z(66)*psip(i);

z(82) = z(80) + z(55)*z(70) - z(56)*z(69);

z(73) = -LCy*z(42)*psip(i) - LCz*z(60);

z(72) = LCz*z(44)*gammap(i);

z(74) = betap(i)*z(73) - psip(i)*z(72);

z(81) = z(74) + z(56)*z(71) - z(57)*z(70);

z(203) = LBar3*z(130)*qBar3p(i);

z(204) = z(130)*z(58) - z(42)*z(131)*psip(i) - z(43)*z(131)*qBar3p(i) - z
    (47)*z(130)*qBar3p(i);

z(205) = z(41)*z(130)*gammap(i) - z(44)*z(131)*qBar3p(i);

z(206) = 0.5*gammap(i)*z(203) - 0.5*LBar3*betap(i)*z(204) - 0.5*LBar3*psip(
    i)*z(205);
```

```
z(202) = 0.5*LBar3*qBar3p(i) + 0.5*z(194)*psip(i) - 0.5*z(195)*betap(i);

z(208) = z(130)*gammap(i) + z(132)*betap(i) + z(133)*psip(i);

z(212) = z(206) + z(202)*z(208);

z(215) = z(160) + z(212);

z(454) = z(3)*z(304) + z(5)*z(303);

z(315) = z(84)*z(131) - z(85)*z(130);

z(455) = z(3)*z(306) + z(5)*z(305);

z(467) = z(313)*z(454) + z(315)*z(455);

z(201) = 0.5*z(196)*gammap(i) - 0.5*z(197)*betap(i) - 0.5*z(198)*psip(i);

z(209) = z(134)*betap(i) + z(135)*psip(i) - z(131)*gammap(i);

z(207) = qBar3p(i) + z(41)*psip(i) - z(50)*betap(i);

z(214) = z(201)*z(209) - z(202)*z(207);

z(468) = z(313)*z(455) + z(314)*z(454);

z(210) = LBar3*z(44)*gammap(i);

z(211) = -0.5*psip(i)*z(210) - 0.5*LBar3*betap(i)*z(60);

z(213) = z(211) - z(201)*z(208);

z(177) = z(84)*z(58) - z(42)*z(85)*psip(i) - z(43)*z(85)*qLink1p(i) - z(47)
   *z(84)*qLink1p(i);

z(178) = -0.5*LBase*z(58) - 0.5*LLink1*z(177);

z(179) = z(41)*z(84)*gammap(i) - z(44)*z(85)*qLink1p(i);

z(180) = -0.5*LBase*z(41)*gammap(i) - 0.5*LLink1*z(179);

z(181) = LLink1*z(84)*qLink1p(i);

z(182) = betap(i)*z(178) + psip(i)*z(180) + 0.5*gammap(i)*z(181);

z(175) = 0.5*z(163)*psip(i) - 0.5*z(164)*betap(i);

z(105) = z(84)*gammap(i) + z(86)*betap(i) + z(87)*psip(i);

z(176) = 0.5*LLink1*qLink1p(i) + 0.5*z(168)*psip(i) - 0.5*z(169)*betap(i);

z(187) = z(182) + z(57)*z(175) + z(105)*z(176);

z(191) = z(160) + z(187);

z(336) = LBar2*z(318);
```

```
z(340) = LBar3*z(313);
z(337) = LBar3*z(314);
z(339) = LBar2*z(316);
z(338) = Distance1*z(96) - LBar2*z(316);
z(335) = -Distance1*z(97) - LBar2*z(318);
z(366) = z(336)*z(338) - z(335)*z(339);
z(367) = (z(336)*z(340)-z(337)*z(339))/z(366);
z(372) = z(367)*(qBar3p(i)-qLink1p(i));
qLink2p(i) = z(372);
z(228) = z(42)*z(84)*psip(i) + z(43)*z(84)*qLink1p(i) + z(85)*z(58) - z(47)
    *z(85)*qLink1p(i);
z(229) = z(96)*z(177) - z(86)*z(96)*qLink2p(i) - z(88)*z(97)*qLink2p(i) - z
    (97)*z(228);
z(280) = -0.5*LLink1*z(177) - 0.5*LLink2*z(229);
z(225) = z(85)*z(97)*qLink1p(i) + z(85)*z(97)*qLink2p(i) - z(84)*z(96)*
    qLink1p(i) - z(84)*z(96)*qLink2p(i);
z(279) = 0.5*LLink1*z(84)*qLink1p(i) - 0.5*LLink2*z(225);
z(232) = z(41)*z(85)*gammap(i) + z(44)*z(84)*qLink1p(i);
z(233) = z(96)*z(179) - z(87)*z(96)*qLink2p(i) - z(89)*z(97)*qLink2p(i) - z
    (97)*z(232);
z(281) = -0.5*LLink1*z(179) - 0.5*LLink2*z(233);
z(282) = betap(i)*z(280) + gammap(i)*z(279) + psip(i)*z(281);
z(122) = z(98)*gammap(i) + z(99)*betap(i) + z(100)*psip(i);
z(278) = 0.5*LLink2*qLink1p(i) + 0.5*LLink2*qLink2p(i) + 0.5*z(272)*psip(i)
     - 0.5*z(273)*betap(i);
z(285) = z(282) + z(105)*z(176) + z(122)*z(278);
z(289) = z(191) + z(285);
z(296) = LLink2*(betap(i)*z(229)+gammap(i)*z(225)+psip(i)*z(233));
z(297) = z(122)*z(278) - 0.5*z(296);
```

```
z(106) = z(88)*betap(i) + z(89)*psip(i) - z(85)*gammap(i);

z(277) = z(269)*gammap(i) + z(270)*betap(i) + z(271)*psip(i);

z(104) = qLink1p(i) + z(41)*psip(i) - z(50)*betap(i);

z(287) = z(106)*z(277) - z(104)*z(176);

z(190) = z(104)*z(176);

z(290) = z(287) - z(190);

z(185) = LLink1*z(44)*gammap(i);

z(186) = -0.5*psip(i)*z(185) - 0.5*LLink1*betap(i)*z(60);

z(286) = z(186) - z(105)*z(277);

z(291) = z(186) + z(286);

z(174) = z(166)*betap(i) + z(167)*psip(i) + 0.5*z(165)*gammap(i);

z(189) = z(55)*z(174) - z(56)*z(175);

z(192) = z(162) + z(189);

z(183) = LBase*z(44)*gammap(i);

z(184) = -0.5*psip(i)*z(183) - 0.5*LBase*betap(i)*z(60);

z(188) = z(184) - z(57)*z(174);

z(193) = z(161) + z(188);

z(623) = z(3)*z(308) + z(5)*z(307);

z(123) = z(101)*gammap(i) + z(102)*betap(i) + z(103)*psip(i);

z(295) = -0.5*z(292)*gammap(i) - 0.5*z(293)*betap(i) - 0.5*z(294)*psip(i);

z(121) = qLink1p(i) + qLink2p(i) + z(41)*psip(i) - z(50)*betap(i);

z(299) = z(123)*z(295) - z(121)*z(278);

z(288) = z(121)*z(278);

z(300) = z(299) - z(288);

z(624) = z(3)*z(310) + z(5)*z(309);

z(283) = LLink2*z(44)*gammap(i);

z(284) = -0.5*psip(i)*z(283) - 0.5*LLink2*betap(i)*z(60);

z(298) = z(284) - z(122)*z(295);

z(301) = z(284) + z(298);
```

```
z(368) = (z(335)*z(340)-z(337)*z(338))/z(366);
z(373) = z(368)*(qBar3p(i)-qLink1p(i));
qBar2p(i) = z(373);
z(224) = 0.5*LBar2*qBar2p(i) + 0.5*LBar2*qLink1p(i) + 0.5*LBar2*qLink2p(i)
   + 0.5*z(216)*psip(i) - 0.5*z(217)*betap(i);
z(238) = z(115)*gammap(i) + z(116)*betap(i) + z(117)*psip(i);
z(230) = z(88)*z(96)*qLink2p(i) + z(96)*z(228) + z(97)*z(177) - z(86)*z(97)
   *qLink2p(i);
z(231) = z(113)*z(229) - z(99)*z(113)*qBar2p(i) - z(102)*z(114)*qBar2p(i) -
    z(114)*z(230);
z(226) = -z(84)*z(97)*qLink1p(i) - z(84)*z(97)*qLink2p(i) - z(85)*z(96)*
   qLink1p(i) - z(85)*z(96)*qLink2p(i);
z(227) = z(113)*z(225) - z(98)*z(113)*qBar2p(i) - z(101)*z(114)*qBar2p(i) -
    z(114)*z(226);
z(234) = z(89)*z(96)*qLink2p(i) + z(96)*z(232) + z(97)*z(179) - z(87)*z(97)
   *qLink2p(i);
z(235) = z(113)*z(233) - z(100)*z(113)*qBar2p(i) - z(103)*z(114)*qBar2p(i)
   - z(114)*z(234);
z(236) = LBar2*(betap(i)*z(231)+gammap(i)*z(227)+psip(i)*z(235));
z(242) = z(224)*z(238) - 0.5*z(236);
z(245) = z(212) + z(215);
z(317) = z(96)*z(114) + z(97)*z(113);
z(461) = z(316)*z(454) + z(317)*z(455);
z(223) = -0.5*z(218)*gammap(i) - 0.5*z(219)*betap(i) - 0.5*z(220)*psip(i);
z(239) = z(118)*gammap(i) + z(119)*betap(i) + z(120)*psip(i);
z(237) = qBar2p(i) + qLink1p(i) + qLink2p(i) + z(41)*psip(i) - z(50)*betap(
   i);
z(244) = z(223)*z(239) - z(224)*z(237);
z(462) = z(316)*z(455) + z(318)*z(454);
```

```
z(240) = LBar2*z(44)*gammap(i);
z(241) = -0.5*psip(i)*z(240) - 0.5*LBar2*betap(i)*z(60);
z(243) = z(241) - z(223)*z(238);
z(651) = z(25)*(ME*z(32)+MBase*(z(32)+z(453)*z(160)+z(473)*z(162)+z(474)*z
    (161))+MC*(z(32)+z(453)*z(83)+z(473)*z(82)+z(474)*z(81))+MBar3*(z(32)+z
    (453)*z(215)+z(467)*z(214)+z(468)*z(213)+z(473)*z(162)+z(474)*z(161))+
    MEndEffector*(z(32)+ ...
z(453)*z(289)+z(453)*z(297)+z(454)*z(290)+z(455)*z(291)+z(473)*z(192)+z
    (474)*z(193)+z(623)*z(300)+z(624)*z(301))+MBar2*(z(32)+z(453)*z(242)+z
    (453)*z(245)+z(461)*z(244)+z(462)*z(243)+z(473)*z(162)+z(474)*z(161)+2*
    z(467)*z(214)+2*z(468)* ...
z(213))-MLink1*(z(454)*z(190)-z(32)-z(453)*z(191)-z(455)*z(186)-z(473)*z
    (192)-z(474)*z(193))-MLink2*(z(623)*z(288)-z(32)-z(453)*z(289)-z(454)*z
    (290)-z(455)*z(291)-z(473)*z(192)-z(474)*z(193)-z(624)*z(284)));
z(355) = (z(322)*z(330)+z(326)*z(329))/z(345);
z(356) = z(330)/z(345);
z(359) = (z(326)*z(353)+z(330)*z(349))/z(345);
z(369) = z(356)*y2p(i) + z(359)*x1p(i) - z(356)*y1p(i) - z(359)*x2p(i);
theta_2p(i) = z(354)*phi_2p(i) + z(357)*phi_1p(i) + z(358)*psip(i) + z(369)
     - z(355)*theta_1p(i);
z(36) = LW*z(14)*phi_2p(i);
z(37) = theta_2p(i)*z(36);
z(22) = z(14)*theta_2p(i);
z(34) = LW*phi_2p(i);
z(40) = -z(37) - z(22)*z(34);
z(688) = MW*z(33)*z(40);
z(528) = z(41)*psip(i) - z(50)*betap(i);
z(529) = gammap(i) + z(47)*betap(i);
z(532) = IBaseyy*z(529);
```

```
z(531) = IBasexx*z(528);

z(542) = z(528)*z(532) - z(529)*z(531);

z(530) = z(43)*betap(i) + z(44)*psip(i);

z(547) = Iyz*z(528) + Izx*z(530) + Izz*z(529);

z(546) = Ixy*z(530) + Iyy*z(528) + Iyz*z(529);

z(574) = z(528)*z(547) - z(529)*z(546);

z(533) = IBasezz*z(530);

z(543) = z(530)*z(531) - z(528)*z(533);

z(545) = Ixx*z(530) + Ixy*z(528) + Izx*z(529);

z(572) = z(530)*z(546) - z(528)*z(545);

z(59) = z(41)*gammap(i)*psip(i) + betap(i)*z(58);

z(541) = IBasezz*z(59);

z(550) = Ixx*z(59);

z(61) = -z(44)*gammap(i)*psip(i) - betap(i)*z(60);

z(559) = Ixy*z(61);

z(62) = z(42)*betap(i)*psip(i);

z(567) = Izx*z(62);

z(538) = IBaseyy*z(62);

z(556) = Izx*z(59);

z(565) = Iyz*z(61);

z(571) = Izz*z(62);

z(91) = z(84)*z(55) - z(85)*z(57);

z(93) = qLink1p(i)*z(91);

z(94) = z(93) + z(84)*z(62) + z(85)*z(59);

z(587) = ILink1yy*z(94);

z(577) = z(88)*betap(i) + z(89)*psip(i) - z(85)*gammap(i);

z(575) = z(41)*psip(i) + qLink1p(i) - z(50)*betap(i);

z(578) = ILink1xx*z(575);

z(580) = ILink1zz*z(577);
```

```
z(593) = z(577)*z(578) - z(580)*z(575);

z(90) = z(84)*z(57) + z(85)*z(55);

z(92) = qLink1p(i)*z(90);

z(95) = z(84)*z(59) - z(92) - z(85)*z(62);

z(591) = ILink1zz*z(95);

z(576) = z(84)*gammap(i) + z(86)*betap(i) + z(87)*psip(i);

z(579) = ILink1yy*z(576);

z(592) = z(579)*z(575) - z(576)*z(578);

z(108) = z(96)*z(106) - z(97)*z(105);

z(110) = qLink2p(i)*z(108);

z(111) = z(110) + z(96)*z(94) + z(97)*z(95);

z(613) = ILink2yy*z(111);

z(603) = z(101)*gammap(i) + z(102)*betap(i) + z(103)*psip(i);

z(601) = z(41)*psip(i) + qLink1p(i) + z(372) - z(50)*betap(i);

z(604) = ILink2xx*z(601);

z(606) = ILink2zz*z(603);

z(619) = z(603)*z(604) - z(606)*z(601);

z(107) = z(96)*z(105) + z(97)*z(106);

z(109) = qLink2p(i)*z(107);

z(112) = z(96)*z(95) - z(109) - z(97)*z(94);

z(617) = ILink2zz*z(112);

z(602) = z(98)*gammap(i) + z(99)*betap(i) + z(100)*psip(i);

z(605) = ILink2yy*z(602);

z(618) = z(605)*z(601) - z(602)*z(604);

z(125) = z(113)*z(123) - z(114)*z(122);

z(127) = qBar2p(i)*z(125);

z(128) = z(127) + z(113)*z(111) + z(114)*z(112);

z(442) = IBar2yy*z(128);

z(432) = z(118)*gammap(i) + z(119)*betap(i) + z(120)*psip(i);
```

```
z(430) = z(41)*psip(i) + qLink1p(i) + z(372) + z(373) - z(50)*betap(i);

z(433) = IBar2xx*z(430);

z(435) = IBar2zz*z(432);

z(448) = z(432)*z(433) - z(435)*z(430);

z(124) = z(113)*z(122) + z(114)*z(123);

z(126) = qBar2p(i)*z(124);

z(129) = z(113)*z(112) - z(126) - z(114)*z(111);

z(446) = IBar2zz*z(129);

z(431) = z(115)*gammap(i) + z(116)*betap(i) + z(117)*psip(i);

z(434) = IBar2yy*z(431);

z(447) = z(434)*z(430) - z(431)*z(433);

z(137) = z(130)*z(55) - z(131)*z(57);

z(139) = qBar3p(i)*z(137);

z(140) = z(139) + z(130)*z(62) + z(131)*z(59);

z(520) = IBar3yy*z(140);

z(510) = z(134)*betap(i) + z(135)*psip(i) - z(131)*gammap(i);

z(508) = z(41)*psip(i) + qBar3p(i) - z(50)*betap(i);

z(511) = IBar3xx*z(508);

z(513) = IBar3zz*z(510);

z(526) = z(510)*z(511) - z(513)*z(508);

z(136) = z(130)*z(57) + z(131)*z(55);

z(138) = qBar3p(i)*z(136);

z(141) = z(130)*z(59) - z(138) - z(131)*z(62);

z(524) = IBar3zz*z(141);

z(509) = z(130)*gammap(i) + z(132)*betap(i) + z(133)*psip(i);

z(512) = IBar3yy*z(509);

z(525) = z(512)*z(508) - z(509)*z(511);

z(27) = z(25)*theta_1p(i);

z(30) = z(10)*z(27);
```

```
z(7) = z(1)*z(3);

z(8) = z(1)*z(5);

z(475) = z(2)*z(47) + z(7)*z(53) + z(8)*z(54);

z(9) = z(1)*theta_1p(i);

z(31) = phi_1p(i)*z(26) + z(9)*z(27);

z(599) = z(47)*z(86) + z(54)*z(304) - z(53)*z(303);

z(632) = z(47)*z(99) + z(54)*z(308) - z(53)*z(307);

z(476) = z(2)*z(43) + z(8)*z(49) - z(7)*z(48);

z(627) = z(84)*z(97) + z(85)*z(96);

z(456) = z(8)*z(52) - z(2)*z(50) - z(7)*z(51);

z(458) = z(2)*z(88) + z(8)*z(306) - z(7)*z(305);

z(626) = z(2)*z(102) + z(8)*z(310) - z(7)*z(309);

z(449) = z(431)*z(435) - z(432)*z(434);

z(527) = z(509)*z(513) - z(510)*z(512);

z(544) = z(529)*z(533) - z(530)*z(532);

z(573) = z(529)*z(545) - z(530)*z(547);

z(594) = z(576)*z(580) - z(577)*z(579);

z(620) = z(602)*z(606) - z(603)*z(605);

z(438) = IBar2xx*z(61);

z(516) = IBar3xx*z(61);

z(536) = IBasexx*z(61);

z(553) = Ixy*z(59);

z(562) = Iyy*z(61);

z(569) = Iyz*z(62);

z(583) = ILink1xx*z(61);

z(609) = ILink2xx*z(61);

z(457) = z(2)*z(86) + z(8)*z(304) - z(7)*z(303);

z(470) = z(313)*z(458) + z(314)*z(457);

z(490) = z(49)*z(131) + z(54)*z(130);
```

```
z(489) = z(48)*z(131) - z(53)*z(130);

z(506) = z(47)*z(132) + z(54)*z(490) - z(53)*z(489);

z(464) = z(316)*z(458) + z(318)*z(457);

z(478) = z(313)*z(317) + z(314)*z(316);

z(479) = z(313)*z(318) + z(315)*z(316);

z(482) = z(84)*z(317) + z(85)*z(316);

z(486) = z(113)*z(308) + z(114)*z(310);

z(485) = z(113)*z(307) + z(114)*z(309);

z(500) = z(47)*z(116) + z(54)*z(486) - z(53)*z(485);

z(699) = z(43)*z(542) + z(43)*z(574) + z(47)*z(543) + z(47)*z(572) + z(43)*
   z(541) + z(43)*z(550) + z(43)*z(559) + z(43)*z(567) + z(47)*z(538) + z
   (47)*z(556) + z(47)*z(565) + z(47)*z(571) + z(86)*z(587) + z(86)*z(593)
    + z(88)*z(591) + ...

z(88)*z(592) + z(99)*z(613) + z(99)*z(619) + z(102)*z(617) + z(102)*z(618)
   + z(116)*z(442) + z(116)*z(448) + z(119)*z(446) + z(119)*z(447) + z
   (132)*z(520) + z(132)*z(526) + z(134)*z(524) + z(134)*z(525) + 0.5*
   MEndEffector*(2*z(147)* ...

z(192)+2*z(173)*z(193)+2*z(274)*z(289)+2*z(274)*z(297)+z(293)*z(450)*z(30)
   +2*z(84)*z(173)*z(291)+2*z(85)*z(169)*z(192)+2*z(85)*z(173)*z(290)+2*z
   (97)*z(273)*z(290)+2*z(98)*z(173)*z(301)+2*z(147)*z(473)*z(32)+2*z(147)
   *z(475)*z(31)+2* ...

z(147)*z(499)*z(297)+2*z(147)*z(599)*z(290)+2*z(147)*z(600)*z(291)+2*z(147)
   *z(632)*z(300)+2*z(147)*z(633)*z(301)+2*z(169)*z(452)*z(30)+2*z(173)*z
   (474)*z(32)+2*z(173)*z(476)*z(31)+2*z(173)*z(627)*z(300)+2*z(273)*z
   (622)*z(30)+2*z(274)* ...

z(453)*z(32)+2*z(274)*z(456)*z(31)-2*z(169)*z(291)-2*z(273)*z(301)-z(293)*z
   (289)-z(293)*z(297)-2*z(84)*z(169)*z(193)-2*z(96)*z(169)*z(301)-2*z(96)
   *z(273)*z(291)-2*z(97)*z(169)*z(300)-2*z(98)*z(273)*z(193)-2*z(101)*z
   (273)*z(192)-2* ...
```

```
z(147)*z(471)*z(30)-2*z(169)*z(455)*z(32)-2*z(169)*z(458)*z(31)-2*z(173)*z
    (472)*z(30)-2*z(273)*z(624)*z(32)-2*z(273)*z(626)*z(31)-2*z(274)*z(450)
    *z(30)-z(293)*z(453)*z(32)-z(293)*z(456)*z(31)) - z(50)*z(449) - z(50)*
    z(527) - z(50)* ...
z(544) - z(50)*z(573) - z(50)*z(594) - z(50)*z(620) - z(50)*z(438) - z(50)*
    z(516) - z(50)*z(536) - z(50)*z(553) - z(50)*z(562) - z(50)*z(569) - z
    (50)*z(583) - z(50)*z(609) - MBase*(z(145)*z(160)+z(144)*z(472)*z(30)+z
    (145)*z(453)*z(32)+ ...
z(145)*z(456)*z(31)+z(147)*z(471)*z(30)-z(144)*z(161)-z(147)*z(162)-z(144)*
    z(474)*z(32)-z(144)*z(476)*z(31)-z(145)*z(450)*z(30)-z(147)*z(473)*z
    (32)-z(147)*z(475)*z(31)) - MC*(z(63)*z(471)*z(30)+z(65)*z(450)*z(30)+z
    (68)*z(472)*z(30)- ...
z(63)*z(82)-z(65)*z(83)-z(68)*z(81)-z(63)*z(473)*z(32)-z(63)*z(475)*z(31)-z
    (65)*z(453)*z(32)-z(65)*z(456)*z(31)-z(68)*z(474)*z(32)-z(68)*z(476)*z
    (31)) - 0.5*MBar3*(z(195)*z(213)+z(130)*z(195)*z(161)+z(195)*z(468)*z
    (32)+z(195)*z(470)* ...
z(31)+2*z(144)*z(472)*z(30)+2*z(147)*z(471)*z(30)+2*z(199)*z(450)*z(30)-2*z
    (144)*z(161)-2*z(147)*z(162)-2*z(199)*z(215)-2*z(130)*z(144)*z(213)-2*z
    (131)*z(144)*z(214)-2*z(144)*z(474)*z(32)-2*z(144)*z(476)*z(31)-2*z
    (147)*z(473)*z(32)-2* ...
z(147)*z(475)*z(31)-2*z(147)*z(506)*z(214)-2*z(147)*z(507)*z(213)-2*z(199)*
    z(453)*z(32)-2*z(199)*z(456)*z(31)-z(131)*z(195)*z(162)-z(195)*z(466)*z
    (30)) - 0.5*MLink1*(z(169)*z(186)+z(84)*z(169)*z(193)+z(169)*z(455)*z
    (32)+z(169)*z(458)* ...
z(31)+2*z(85)*z(173)*z(190)+2*z(147)*z(471)*z(30)+2*z(147)*z(599)*z(190)+2*
    z(170)*z(450)*z(30)+2*z(173)*z(472)*z(30)-2*z(147)*z(192)-2*z(170)*z
    (191)-2*z(173)*z(193)-2*z(84)*z(173)*z(186)-2*z(147)*z(473)*z(32)-2*z
    (147)*z(475)*z(31)-2* ...
```

```
z(147)*z(600)*z(186)-2*z(170)*z(453)*z(32)-2*z(170)*z(456)*z(31)-2*z(173)*z
    (474)*z(32)-2*z(173)*z(476)*z(31)-z(85)*z(169)*z(192)-z(169)*z(452)*z
    (30)) - 0.5*MLink2*(z(273)*z(284)+2*z(169)*z(291)+z(96)*z(273)*z(291)+z
    (98)*z(273)*z(193)+ ...
z(101)*z(273)*z(192)+z(273)*z(624)*z(32)+z(273)*z(626)*z(31)+2*z(84)*z(169)
    *z(193)+2*z(96)*z(169)*z(284)+2*z(147)*z(471)*z(30)+2*z(147)*z(632)*z
    (288)+2*z(169)*z(455)*z(32)+2*z(169)*z(458)*z(31)+2*z(173)*z(472)*z(30)
    +2*z(173)*z(627)* ...
z(288)+2*z(274)*z(450)*z(30)-2*z(147)*z(192)-2*z(173)*z(193)-2*z(274)*z
    (289)-2*z(84)*z(173)*z(291)-2*z(85)*z(169)*z(192)-2*z(85)*z(173)*z(290)
    -2*z(97)*z(169)*z(288)-2*z(98)*z(173)*z(284)-2*z(147)*z(473)*z(32)-2*z
    (147)*z(475)*z(31)-2* ...
z(147)*z(599)*z(290)-2*z(147)*z(600)*z(291)-2*z(147)*z(633)*z(284)-2*z(169)
    *z(452)*z(30)-2*z(173)*z(474)*z(32)-2*z(173)*z(476)*z(31)-2*z(274)*z
    (453)*z(32)-2*z(274)*z(456)*z(31)-z(97)*z(273)*z(290)-z(273)*z(622)*z
    (30)) - 0.5*MBar2*( ...
z(217)*z(243)+z(219)*z(242)+z(219)*z(245)+4*z(195)*z(213)+z(217)*z(462)*z
    (32)+z(217)*z(464)*z(31)+z(217)*z(483)*z(162)+z(217)*z(484)*z(161)+z
    (219)*z(453)*z(32)+z(219)*z(456)*z(31)+2*z(130)*z(195)*z(161)+2*z(144)*
    z(472)*z(30)+2*z(147)* ...
z(471)*z(30)+2*z(195)*z(468)*z(32)+2*z(195)*z(470)*z(31)+2*z(195)*z(478)*z
    (244)+2*z(195)*z(480)*z(243)+2*z(217)*z(479)*z(214)+2*z(217)*z(480)*z
    (213)+2*z(221)*z(450)*z(30)-2*z(144)*z(161)-2*z(147)*z(162)-2*z(221)*z
    (242)-2*z(221)*z(245)- ...
4*z(130)*z(144)*z(213)-4*z(131)*z(144)*z(214)-4*z(147)*z(506)*z(214)-4*z
    (147)*z(507)*z(213)-2*z(131)*z(195)*z(162)-2*z(144)*z(474)*z(32)-2*z
    (144)*z(476)*z(31)-2*z(144)*z(482)*z(244)-2*z(144)*z(484)*z(243)-2*z
    (147)*z(473)*z(32)-2* ...
```

```
z(147)*z(475)*z(31)-2*z(147)*z(499)*z(242)-2*z(147)*z(500)*z(244)-2*z(147)*
    z(501)*z(243)-2*z(195)*z(466)*z(30)-2*z(221)*z(453)*z(32)-2*z(221)*z
    (456)*z(31)-z(217)*z(460)*z(30)-z(219)*z(450)*z(30));
z(726) = z(651) - z(355)*z(688) - z(361)*z(699);
z(639) = z(634)*z(5)*z(25);
z(696) = MC*(z(48)*z(68)+z(51)*z(65)-z(53)*z(63)) + MBase*(z(48)*z(144)-z
    (51)*z(145)-z(53)*z(147)) + 0.5*MBar3*(2*z(48)*z(144)+2*z(51)*z(199)-2*
    z(53)*z(147)-z(195)*z(491)) + 0.5*MLink1*(2*z(48)*z(173)+2*z(51)*z(170)
    -2*z(53)*z(147)- ...
z(169)*z(305)) + 0.5*MLink2*(2*z(48)*z(173)+2*z(51)*z(274)-2*z(53)*z(147)
    -2*z(169)*z(305)-z(273)*z(309)) + 0.5*MBar2*(2*z(48)*z(144)+2*z(51)*z
    (221)-2*z(53)*z(147)-2*z(195)*z(491)-z(51)*z(219)-z(217)*z(487)) + 0.5*
    MEndEffector*(2*z(48)* ...
z(173)+2*z(51)*z(274)-2*z(53)*z(147)-2*z(169)*z(305)-2*z(273)*z(309)-z(51)*
    z(293));
z(716) = z(639) - z(361)*z(696);
z(636) = z(25)*(2*MBase*(z(145)*z(453)-z(144)*z(474)-z(147)*z(473))+MBar3*(
    z(195)*z(468)-2*z(144)*z(474)-2*z(147)*z(473)-2*z(199)*z(453))+MLink1*(
    z(169)*z(455)-2*z(147)*z(473)-2*z(170)*z(453)-2*z(173)*z(474))+MLink2*(
    z(273)*z(624)+2* ...
z(169)*z(455)-2*z(147)*z(473)-2*z(173)*z(474)-2*z(274)*z(453))+MBar2*(z
    (217)*z(462)+z(219)*z(453)+2*z(195)*z(468)-2*z(144)*z(474)-2*z(147)*z
    (473)-2*z(221)*z(453))-2*MC*(z(63)*z(473)+z(65)*z(453)+z(68)*z(474))-
    MEndEffector*(2*z(147)* ...
z(473)+2*z(173)*z(474)+2*z(274)*z(453)-2*z(169)*z(455)-2*z(273)*z(624)-z
    (293)*z(453)));
```

```
z(637) = z(25)*(2*MBase*(z(146)*z(453)-z(143)*z(474)-z(148)*z(473))+2*MC*(z
    (66)*z(453)-z(64)*z(473)-z(67)*z(474))+MBar2*(z(220)*z(453)-2*z(143)*z
    (474)-2*z(148)*z(473)-2*z(194)*z(468)-2*z(222)*z(453)-z(216)*z(462))-
    MBar3*(z(194)*z(468)+ ...
2*z(143)*z(474)+2*z(148)*z(473)+2*z(200)*z(453))-MLink1*(z(168)*z(455)+2*z
    (148)*z(473)+2*z(171)*z(453)+2*z(172)*z(474))-MLink2*(z(272)*z(624)+2*z
    (148)*z(473)+2*z(168)*z(455)+2*z(172)*z(474)+2*z(275)*z(453))-
    MEndEffector*(2*z(148)* ...
z(473)+2*z(168)*z(455)+2*z(172)*z(474)+2*z(272)*z(624)+2*z(275)*z(453)-z
    (294)*z(453)));
z(638) = z(25)*(2*MBase*BasePos(i)*z(474)+2*MLink2*(BasePos(i)*z(474)-z
    (276)*z(453))+MBar3*(2*BasePos(i)*z(474)-z(196)*z(453))+MLink1*(2*
    BasePos(i)*z(474)-z(165)*z(453))-2*MC*(LCx*z(453)-LCy*z(474))-MBar2*(2*
    z(196)*z(453)-2*BasePos(i)*z(474)-z(218)* ...
z(453))-MEndEffector*(2*z(276)*z(453)-2*BasePos(i)*z(474)-z(292)*z(453)));
z(778) = 0.5*z(364)*z(636) - 0.5*z(637) - 0.5*z(342)*z(638);
z(787) = z(778) - z(355)*z(784) - z(361)*z(773);
z(745) = 0.5*z(344)*z(638) + 0.5*z(363)*z(636);
z(747) = z(745) - z(355)*z(743) - z(361)*z(732);
z(760) = 0.5*z(360)*z(636) - 0.5*z(341)*z(638);
z(767) = z(760) - z(355)*z(753) - z(361)*z(755);
z(904) = z(747)*z(769) - z(749)*z(767);
z(905) = z(747)*z(770) - z(750)*z(767);
z(906) = z(787)*z(902) + z(790)*z(904) - z(789)*z(905);
z(907) = z(747)*z(768) - z(748)*z(767);
z(908) = z(787)*z(901) + z(790)*z(907) - z(788)*z(905);
z(909) = z(787)*z(900) + z(789)*z(907) - z(788)*z(904);
z(35) = z(33)*theta_2p(i);
z(38) = z(22)*z(35);
```

```
z(374) = LC*(z(41)*z(47)*gammap(i)+z(42)*z(44)*psip(i));
z(375) = psip(i)*z(374) + phi_1p(i)*z(28) - phi_2p(i)*z(36) - LC*gammap(i)*
    z(58);
z(376) = z(13)*z(15)*phi_2p(i) + z(14)*z(17)*theta_2p(i);
z(377) = LE*(z(1)*z(3)*theta_1p(i)+z(2)*z(5)*phi_1p(i));
z(378) = z(1)*z(3)*phi_1p(i) + z(2)*z(5)*theta_1p(i);
z(379) = LW*(z(13)*z(15)*theta_2p(i)+z(14)*z(17)*phi_2p(i));
z(380) = z(41)*z(45)*gammap(i) + z(41)*z(42)*z(46)*psip(i) + z(41)*z(45)*z
    (47)*betap(i) - z(44)*z(46)*betap(i) - z(44)*z(46)*z(47)*gammap(i);
z(381) = z(42)*z(45)*betap(i) - z(46)*z(47)*psip(i);
z(382) = LC*(z(42)*z(48)*psip(i)+z(43)*z(381)+z(47)*z(380)+z(53)*z(58));
z(383) = LC*(z(41)*z(53)*gammap(i)+z(44)*z(381));
z(384) = theta_2p(i)*z(379) + LW*phi_2p(i)*z(376) - psip(i)*z(383) -
    theta_1p(i)*z(377) - LC*gammap(i)*z(380) - LE*phi_1p(i)*z(378) - betap(
    i)*z(382);
z(385) = z(1)*z(5)*phi_1p(i) - z(2)*z(3)*theta_1p(i);
z(386) = LE*(z(1)*z(5)*theta_1p(i)-z(2)*z(3)*phi_1p(i));
z(387) = -z(42)*z(46)*betap(i) - z(45)*z(47)*psip(i);
z(388) = z(41)*z(46)*gammap(i) + z(44)*z(45)*betap(i) + z(41)*z(46)*z(47)*
    betap(i) + z(44)*z(45)*z(47)*gammap(i) - z(41)*z(42)*z(45)*psip(i);
z(389) = LC*(z(42)*z(49)*psip(i)+z(47)*z(388)-z(43)*z(387)-z(54)*z(58));
z(390) = LC*(z(41)*z(54)*gammap(i)+z(44)*z(387));
z(391) = z(13)*z(17)*phi_2p(i) - z(14)*z(15)*theta_2p(i);
z(392) = LW*(z(13)*z(17)*theta_2p(i)-z(14)*z(15)*phi_2p(i));
z(393) = psip(i)*z(390) + theta_1p(i)*z(386) + LE*phi_1p(i)*z(385) -
    theta_2p(i)*z(392) - LC*gammap(i)*z(388) - LW*phi_2p(i)*z(391) - betap(
    i)*z(389);
z(394) = z(97)*z(114)*z(372) + z(97)*z(114)*z(373) - z(96)*z(113)*z(372) -
    z(96)*z(113)*z(373);
```

```
z(395) = -LBar2*z(394) - Distance1*z(96)*z(372);
z(396) = z(84)*z(130)*qLink1p(i) + z(85)*z(131)*qLink1p(i) - z(84)*z(130)*
   qBar3p(i) - z(85)*z(131)*qBar3p(i);
z(397) = z(372)*z(395) + LBar3*qLink1p(i)*z(396) - LBar2*z(373)*z(394) -
   LBar3*qBar3p(i)*z(396);
z(398) = -z(96)*z(114)*z(372) - z(96)*z(114)*z(373) - z(97)*z(113)*z(372) -
    z(97)*z(113)*z(373);
z(399) = -LBar2*z(398) - Distance1*z(97)*z(372);
z(400) = z(84)*z(131)*qLink1p(i) + z(85)*z(130)*qBar3p(i) - z(84)*z(131)*
   qBar3p(i) - z(85)*z(130)*qLink1p(i);
z(401) = z(372)*z(399) + LBar3*qLink1p(i)*z(400) - LBar2*z(373)*z(398) -
   LBar3*qBar3p(i)*z(400);
z(402) = z(375)/z(320);
z(403) = z(325)*z(402) - z(384);
z(404) = z(334)*z(402) - z(393);
z(405) = (z(326)*z(404)+z(330)*z(403))/z(345);
z(406) = (z(324)*z(404)-z(333)*z(403))/z(345);
z(407) = (z(336)*z(401)-z(339)*z(397))/z(366);
z(408) = (z(335)*z(401)-z(338)*z(397))/z(366);
z(424) = B_cable1*z(1)*theta_1p(i);
z(425) = B_cable1*phi_1p(i);
z(426) = B_cable1*z(2)*theta_1p(i);
z(427) = B_cable2*z(13)*theta_2p(i);
z(428) = B_cable2*phi_2p(i);
z(429) = B_cable2*z(14)*theta_2p(i);
z(459) = z(316)*z(451) + z(317)*z(452);
z(465) = z(313)*z(451) + z(315)*z(452);
z(621) = z(1)*z(99) + z(4)*z(307) - z(6)*z(308);
z(635) = z(634)*z(25)^2;
```

```
z(641) = z(640)*z(25)*z(453);
z(643) = z(642)*z(25)*z(468);
z(646) = z(25)*(z(644)*z(462)+z(645)*z(455)+2*MEndEffector*(LLink1*z(455)+
    LLink2*z(624))+MLink2*(LLink2*z(624)+2*LLink1*z(455)));
z(649) = z(25)*(z(644)*z(462)+z(647)*z(624)+2*z(648)*z(624));
z(650) = z(644)*z(25)*z(462);
z(656) = z(655)*z(1);
z(657) = z(655)*z(4);
z(659) = z(658)*z(450);
z(662) = z(661)*z(466);
z(663) = LE*(z(644)*z(460)+z(645)*z(452)+2*MEndEffector*(LLink1*z(452)+
    LLink2*z(622))+MLink2*(LLink2*z(622)+2*LLink1*z(452)));
z(664) = LE*(z(644)*z(460)+z(647)*z(622)+2*z(648)*z(622));
z(666) = z(665)*z(460);
z(667) = LE*(ME*z(30)+MBase*(z(30)-z(450)*z(160)-z(471)*z(162)-z(472)*z
    (161))+MC*(z(30)-z(450)*z(83)-z(471)*z(82)-z(472)*z(81))+MLink1*(z(30)+
    z(451)*z(190)-z(450)*z(191)-z(452)*z(186)-z(471)*z(192)-z(472)*z(193))+
    MBar3*(z(30)-z(450)* ...
z(215)-z(465)*z(214)-z(466)*z(213)-z(471)*z(162)-z(472)*z(161))+MLink2*(z
    (30)+z(621)*z(288)-z(450)*z(289)-z(451)*z(290)-z(452)*z(291)-z(471)*z
    (192)-z(472)*z(193)-z(622)*z(284))+MBar2*(z(30)-2*z(465)*z(214)-2*z
    (466)*z(213)-z(450)* ...
z(242)-z(450)*z(245)-z(459)*z(244)-z(460)*z(243)-z(471)*z(162)-z(472)*z
    (161))+MEndEffector*(z(30)-z(450)*z(289)-z(450)*z(297)-z(451)*z(290)-z
    (452)*z(291)-z(471)*z(192)-z(472)*z(193)-z(621)*z(300)-z(622)*z(301)));
z(670) = z(669)*z(13);
z(671) = z(669)*z(16);
z(672) = z(669)*z(38);
```

```
z(676) = IBar2xx*z(41) + ILink1xx*z(41) + ILink2xx*z(41) + 0.25*z(645)*(z
    (168)+2*z(84)*z(172)+2*z(148)*z(600)) + 0.25*z(644)*(z(216)+2*z(143)*z
    (484)+2*z(148)*z(501)+2*z(194)*z(480)) + MEndEffector*(LLink1*z(168)+
    LLink2*z(272)+LLink1* ...
z(84)*z(172)+LLink1*z(96)*z(272)+LLink1*z(148)*z(600)+LLink2*z(96)*z(168)+
    LLink2*z(98)*z(172)+LLink2*z(148)*z(633)) + 0.25*MLink2*(LLink2*z(272)
    +4*LLink1*z(168)+2*LLink1*z(96)*z(272)+2*LLink2*z(96)*z(168)+2*LLink2*z
    (98)*z(172)+2* ...
LLink2*z(148)*z(633)+4*LLink1*z(84)*z(172)+4*LLink1*z(148)*z(600));
z(677) = IBar2xx*z(41) + ILink2xx*z(41) + z(648)*(z(272)+z(96)*z(168)+z(98)
    *z(172)+z(148)*z(633)) + 0.25*z(644)*(z(216)+2*z(143)*z(484)+2*z(148)*z
    (501)+2*z(194)*z(480)) + 0.25*z(647)*(z(272)+2*z(96)*z(168)+2*z(98)*z
    (172)+2*z(148)* ...
z(633));
z(678) = IBar2xx*z(41) + 0.25*z(644)*(z(216)+2*z(143)*z(484)+2*z(148)*z
    (501)+2*z(194)*z(480));
z(679) = MBase*(z(43)*z(143)+z(47)*z(148)+z(50)*z(146)) + MC*(z(43)*z(67)+z
    (47)*z(64)+z(50)*z(66)) - 0.5*MBar3*(2*z(50)*z(200)-2*z(43)*z(143)-2*z
    (47)*z(148)-z(134)*z(194)) - 0.5*MLink1*(2*z(50)*z(171)-2*z(43)*z(172)
    -2*z(47)*z(148)- ...
z(88)*z(168)) - 0.5*MLink2*(2*z(50)*z(275)-2*z(43)*z(172)-2*z(47)*z(148)-2*
    z(88)*z(168)-z(102)*z(272)) - 0.5*MBar2*(2*z(50)*z(222)-2*z(43)*z(143)
    -2*z(47)*z(148)-2*z(134)*z(194)-z(50)*z(220)-z(119)*z(216)) - 0.5*
    MEndEffector*(2*z(50)* ...
z(275)-2*z(43)*z(172)-2*z(47)*z(148)-2*z(88)*z(168)-2*z(102)*z(272)-z(50)*z
    (294));
```

```
z(680) = MBase*(z(48)*z(143)-z(51)*z(146)-z(53)*z(148)) + MC*(z(48)*z(67)-z
    (51)*z(66)-z(53)*z(64)) - 0.5*MBar3*(2*z(53)*z(148)-2*z(48)*z(143)-2*z
    (51)*z(200)-z(194)*z(491)) - 0.5*MLink1*(2*z(53)*z(148)-2*z(48)*z(172)
    -2*z(51)*z(171)- ...
z(168)*z(305)) - 0.5*MLink2*(2*z(53)*z(148)-2*z(48)*z(172)-2*z(51)*z(275)
    -2*z(168)*z(305)-z(272)*z(309)) - 0.5*MBar2*(z(51)*z(220)+2*z(53)*z
    (148)-2*z(48)*z(143)-2*z(51)*z(222)-2*z(194)*z(491)-z(216)*z(487)) -
    0.5*MEndEffector*(z(51)* ...
z(294)+2*z(53)*z(148)-2*z(48)*z(172)-2*z(51)*z(275)-2*z(168)*z(305)-2*z
    (272)*z(309));
z(681) = MBar3*z(200) + MLink1*z(171) + MLink2*z(275) + 0.5*MEndEffector
    *(2*z(275)-z(294)) - MBase*z(146) - 0.5*MBar2*(z(220)-2*z(222));
z(684) = IBar3xx*z(41) + 0.25*z(682)*(z(194)+2*z(130)*z(143)+2*z(148)*z
    (507)) + 0.5*z(683)*(2*z(194)+z(216)*z(480)+2*z(130)*z(143)+2*z(148)*z
    (507));
z(685) = z(41)*z(449) + z(41)*z(527) + z(41)*z(544) + z(41)*z(573) + z(41)*
    z(594) + z(41)*z(620) + z(44)*z(542) + z(44)*z(574) + z(41)*z(438) + z
    (41)*z(516) + z(41)*z(536) + z(41)*z(553) + z(41)*z(562) + z(41)*z(569)
     + z(41)*z(583) + ...
z(41)*z(609) + z(44)*z(541) + z(44)*z(550) + z(44)*z(559) + z(44)*z(567) +
    z(87)*z(587) + z(87)*z(593) + z(89)*z(591) + z(89)*z(592) + z(100)*z
    (613) + z(100)*z(619) + z(103)*z(617) + z(103)*z(618) + z(117)*z(442) +
     z(117)*z(448) + ...
z(120)*z(446) + z(120)*z(447) + z(133)*z(520) + z(133)*z(526) + z(135)*z
    (524) + z(135)*z(525) + 0.5*MEndEffector*(2*z(148)*z(192)+2*z(168)*z
    (291)+2*z(172)*z(193)+2*z(272)*z(301)+2*z(275)*z(289)+2*z(275)*z(297)+z
    (294)*z(450)*z(30)+2* ...
```

```
z(84)*z(168)*z(193)+2*z(84)*z(172)*z(291)+2*z(85)*z(172)*z(290)+2*z(96)*z
    (168)*z(301)+2*z(96)*z(272)*z(291)+2*z(97)*z(168)*z(300)+2*z(98)*z(172)
    *z(301)+2*z(98)*z(272)*z(193)+2*z(101)*z(272)*z(192)+2*z(148)*z(473)*z
    (32)+2*z(148)*z(475)* ...
z(31)+2*z(148)*z(499)*z(297)+2*z(148)*z(599)*z(290)+2*z(148)*z(600)*z(291)
    +2*z(148)*z(632)*z(300)+2*z(148)*z(633)*z(301)+2*z(168)*z(455)*z(32)+2*
    z(168)*z(458)*z(31)+2*z(172)*z(474)*z(32)+2*z(172)*z(476)*z(31)+2*z
    (172)*z(627)*z(300)+2* ...
z(272)*z(624)*z(32)+2*z(272)*z(626)*z(31)+2*z(275)*z(453)*z(32)+2*z(275)*z
    (456)*z(31)-z(294)*z(289)-z(294)*z(297)-2*z(85)*z(168)*z(192)-2*z(97)*z
    (272)*z(290)-2*z(148)*z(471)*z(30)-2*z(168)*z(452)*z(30)-2*z(172)*z
    (472)*z(30)-2*z(272)* ...
z(622)*z(30)-2*z(275)*z(450)*z(30)-z(294)*z(453)*z(32)-z(294)*z(456)*z(31))
     - MBase*(z(146)*z(160)+z(143)*z(472)*z(30)+z(146)*z(453)*z(32)+z(146)*
    z(456)*z(31)+z(148)*z(471)*z(30)-z(143)*z(161)-z(148)*z(162)-z(143)*z
    (474)*z(32)-z(143)* ...
z(476)*z(31)-z(146)*z(450)*z(30)-z(148)*z(473)*z(32)-z(148)*z(475)*z(31)) -
     MC*(z(66)*z(83)+z(64)*z(471)*z(30)+z(66)*z(453)*z(32)+z(66)*z(456)*z
    (31)+z(67)*z(472)*z(30)-z(64)*z(82)-z(67)*z(81)-z(64)*z(473)*z(32)-z
    (64)*z(475)*z(31)- ...
z(66)*z(450)*z(30)-z(67)*z(474)*z(32)-z(67)*z(476)*z(31)) - 0.5*MBar3*(z
    (131)*z(194)*z(162)+z(194)*z(466)*z(30)+2*z(143)*z(472)*z(30)+2*z(148)*
    z(471)*z(30)+2*z(200)*z(450)*z(30)-2*z(143)*z(161)-2*z(148)*z(162)-2*z
    (200)*z(215)-z(194)* ...
z(213)-2*z(130)*z(143)*z(213)-2*z(131)*z(143)*z(214)-2*z(143)*z(474)*z(32)
    -2*z(143)*z(476)*z(31)-2*z(148)*z(473)*z(32)-2*z(148)*z(475)*z(31)-2*z
    (148)*z(506)*z(214)-2*z(148)*z(507)*z(213)-2*z(200)*z(453)*z(32)-2*z
    (200)*z(456)*z(31)- ...
```

```
z(130)*z(194)*z(161)-z(194)*z(468)*z(32)-z(194)*z(470)*z(31)) - 0.5*MLink1
    *(z(85)*z(168)*z(192)+z(168)*z(452)*z(30)+2*z(85)*z(172)*z(190)+2*z
    (148)*z(471)*z(30)+2*z(148)*z(599)*z(190)+2*z(171)*z(450)*z(30)+2*z
    (172)*z(472)*z(30)-2* ...
z(148)*z(192)-2*z(171)*z(191)-2*z(172)*z(193)-z(168)*z(186)-2*z(84)*z(172)*
    z(186)-2*z(148)*z(473)*z(32)-2*z(148)*z(475)*z(31)-2*z(148)*z(600)*z
    (186)-2*z(171)*z(453)*z(32)-2*z(171)*z(456)*z(31)-2*z(172)*z(474)*z(32)
    -2*z(172)*z(476)* ...
z(31)-z(84)*z(168)*z(193)-z(168)*z(455)*z(32)-z(168)*z(458)*z(31)) - 0.5*
    MLink2*(z(97)*z(272)*z(290)+z(272)*z(622)*z(30)+2*z(85)*z(168)*z(192)
    +2*z(97)*z(168)*z(288)+2*z(148)*z(471)*z(30)+2*z(148)*z(632)*z(288)+2*z
    (168)*z(452)*z(30)+2* ...
z(172)*z(472)*z(30)+2*z(172)*z(627)*z(288)+2*z(275)*z(450)*z(30)-2*z(148)*z
    (192)-2*z(168)*z(291)-2*z(172)*z(193)-2*z(275)*z(289)-z(272)*z(284)-2*z
    (84)*z(168)*z(193)-2*z(84)*z(172)*z(291)-2*z(85)*z(172)*z(290)-2*z(96)*
    z(168)*z(284)-2* ...
z(98)*z(172)*z(284)-2*z(148)*z(473)*z(32)-2*z(148)*z(475)*z(31)-2*z(148)*z
    (599)*z(290)-2*z(148)*z(600)*z(291)-2*z(148)*z(633)*z(284)-2*z(168)*z
    (455)*z(32)-2*z(168)*z(458)*z(31)-2*z(172)*z(474)*z(32)-2*z(172)*z(476)
    *z(31)-2*z(275)* ...
z(453)*z(32)-2*z(275)*z(456)*z(31)-z(96)*z(272)*z(291)-z(98)*z(272)*z(193)-
    z(101)*z(272)*z(192)-z(272)*z(624)*z(32)-z(272)*z(626)*z(31)) - 0.5*
    MBar2*(z(220)*z(242)+z(220)*z(245)+z(216)*z(460)*z(30)+z(220)*z(453)*z
    (32)+z(220)*z(456)* ...
z(31)+2*z(131)*z(194)*z(162)+2*z(143)*z(472)*z(30)+2*z(148)*z(471)*z(30)+2*
    z(194)*z(466)*z(30)+2*z(222)*z(450)*z(30)-4*z(194)*z(213)-2*z(143)*z
    (161)-2*z(148)*z(162)-2*z(222)*z(242)-2*z(222)*z(245)-z(216)*z(243)-4*z
    (130)*z(143)*z(213)-4* ...
```

```
z(131)*z(143)*z(214)-4*z(148)*z(506)*z(214)-4*z(148)*z(507)*z(213)-2*z(130)
    *z(194)*z(161)-2*z(143)*z(474)*z(32)-2*z(143)*z(476)*z(31)-2*z(143)*z
    (482)*z(244)-2*z(143)*z(484)*z(243)-2*z(148)*z(473)*z(32)-2*z(148)*z
    (475)*z(31)-2*z(148)* ...
z(499)*z(242)-2*z(148)*z(500)*z(244)-2*z(148)*z(501)*z(243)-2*z(194)*z(468)
    *z(32)-2*z(194)*z(470)*z(31)-2*z(194)*z(478)*z(244)-2*z(194)*z(480)*z
    (243)-2*z(216)*z(479)*z(214)-2*z(216)*z(480)*z(213)-2*z(222)*z(453)*z
    (32)-2*z(222)*z(456)* ...
z(31)-z(216)*z(462)*z(32)-z(216)*z(464)*z(31)-z(216)*z(483)*z(162)-z(216)*z
    (484)*z(161)-z(220)*z(450)*z(30));
z(687) = MW*z(17)*z(33);
z(692) = 0.25*z(644)*(2*z(144)*z(484)+2*z(147)*z(501)-z(217)-2*z(195)*z
    (480)) - IBar2xx*z(50) - ILink1xx*z(50) - ILink2xx*z(50) - 0.25*z(645)
    *(z(169)-2*z(84)*z(173)-2*z(147)*z(600)) - MEndEffector*(LLink1*z(169)+
    LLink2*z(273)+LLink1* ...
z(96)*z(273)+LLink2*z(96)*z(169)-LLink1*z(84)*z(173)-LLink1*z(147)*z(600)-
    LLink2*z(98)*z(173)-LLink2*z(147)*z(633)) - 0.25*MLink2*(LLink2*z(273)
    +4*LLink1*z(169)+2*LLink1*z(96)*z(273)+2*LLink2*z(96)*z(169)-4*LLink1*z
    (84)*z(173)-4* ...
LLink1*z(147)*z(600)-2*LLink2*z(98)*z(173)-2*LLink2*z(147)*z(633));
z(693) = 0.25*z(644)*(2*z(144)*z(484)+2*z(147)*z(501)-z(217)-2*z(195)*z
    (480)) - IBar2xx*z(50) - ILink2xx*z(50) - z(648)*(z(273)+z(96)*z(169)-z
    (98)*z(173)-z(147)*z(633)) - 0.25*z(647)*(z(273)+2*z(96)*z(169)-2*z(98)
    *z(173)-2*z(147)* ...
z(633));
z(694) = 0.25*z(644)*(2*z(144)*z(484)+2*z(147)*z(501)-z(217)-2*z(195)*z
    (480)) - IBar2xx*z(50);
```

```
z(695) = MBase*(z(43)*z(144)+z(47)*z(147)+z(50)*z(145)) + MC*(z(43)*z(68)+z
    (47)*z(63)-z(50)*z(65)) + 0.5*MBar3*(2*z(43)*z(144)+2*z(47)*z(147)-2*z
    (50)*z(199)-z(134)*z(195)) + 0.5*MLink1*(2*z(43)*z(173)+2*z(47)*z(147)
    -2*z(50)*z(170)- ...
z(88)*z(169)) + 0.5*MLink2*(2*z(43)*z(173)+2*z(47)*z(147)-2*z(50)*z(274)-2*
    z(88)*z(169)-z(102)*z(273)) + 0.5*MBar2*(z(50)*z(219)+2*z(43)*z(144)+2*
    z(47)*z(147)-2*z(50)*z(221)-2*z(134)*z(195)-z(119)*z(217)) + 0.5*
    MEndEffector*(z(50)* ...
z(293)+2*z(43)*z(173)+2*z(47)*z(147)-2*z(50)*z(274)-2*z(88)*z(169)-2*z(102)
    *z(273));
z(697) = 0.5*z(683)*(2*z(130)*z(144)+2*z(147)*z(507)-2*z(195)-z(217)*z(480)
    ) - IBar3xx*z(50) - 0.25*z(682)*(z(195)-2*z(130)*z(144)-2*z(147)*z(507)
    );
z(698) = MBar3*z(199) + MLink1*z(170) + MLink2*z(274) + 0.5*MEndEffector
    *(2*z(274)-z(293)) - MBase*z(145) - 0.5*MBar2*(z(219)-2*z(221));
z(704) = MC*(LCx*z(51)-LCy*z(48)) + 0.5*MBar2*(2*z(51)*z(196)-2*BasePos(i)*
    z(48)-z(51)*z(218)) + 0.5*MEndEffector*(2*z(51)*z(276)-2*BasePos(i)*z
    (48)-z(51)*z(292)) - MBase*BasePos(i)*z(48) - MLink2*(BasePos(i)*z(48)-
    z(51)*z(276)) - 0.5*MBar3*(2* ...
BasePos(i)*z(48)-z(51)*z(196)) - 0.5*MLink1*(2*BasePos(i)*z(48)-z(51)*z
    (165));
z(705) = -MBase*BasePos(i)*z(43) - MC*(LCx*z(50)+LCy*z(43)) - MLink2*(
    BasePos(i)*z(43)+z(50)*z(276)) - 0.5*MBar3*(z(50)*z(196)+2*BasePos(i)*z
    (43)) - 0.5*MLink1*(z(50)*z(165)+2*BasePos(i)*z(43)) - 0.5*MBar2*(2*
    BasePos(i)*z(43)+2*z(50)*z(196)-z(50)* ...
z(218)) - 0.5*MEndEffector*(2*BasePos(i)*z(43)+2*z(50)*z(276)-z(50)*z(292))
    ;
z(706) = MLink2*z(276) + 0.5*MBar3*z(196) + 0.5*MLink1*z(165) + 0.5*MBar2
    *(2*z(196)-z(218)) + 0.5*MEndEffector*(2*z(276)-z(292));
```

```
z(707) = z(642)*BasePos(i)*z(130);
z(708) = BasePos(i)*(z(644)*z(484)+z(645)*z(84)+2*MEndEffector*(LLink1*z
    (84)+LLink2*z(98))+MLink2*(LLink2*z(98)+2*LLink1*z(84)));
z(709) = BasePos(i)*(z(644)*z(484)+z(647)*z(98)+2*z(648)*z(98));
z(710) = z(644)*BasePos(i)*z(484);
z(711) = z(543) + z(572) + z(538) + z(556) + z(565) + z(571) + z(84)*z(587)
     + z(84)*z(593) + z(98)*z(613) + z(98)*z(619) + z(101)*z(617) + z(101)*
    z(618) + z(115)*z(442) + z(115)*z(448) + z(118)*z(446) + z(118)*z(447)
    + z(130)*z(520) + ...
z(130)*z(526) + MBase*BasePos(i)*(z(472)*z(30)-z(161)-z(474)*z(32)-z(476)*z
    (31)) + MC*(LCx*z(83)+LCx*z(453)*z(32)+LCx*z(456)*z(31)+LCy*z(472)*z
    (30)-LCy*z(81)-LCx*z(450)*z(30)-LCy*z(474)*z(32)-LCy*z(476)*z(31)) +
    0.5*MBar2*(2*z(196)* ...
z(242)+2*z(196)*z(245)+z(218)*z(450)*z(30)+2*BasePos(i)*z(472)*z(30)+2*z
    (196)*z(453)*z(32)+2*z(196)*z(456)*z(31)-2*BasePos(i)*z(161)-z(218)*z
    (242)-z(218)*z(245)-4*BasePos(i)*z(130)*z(213)-4*BasePos(i)*z(131)*z
    (214)-2*BasePos(i)*z(474)*z(32)-2* ...
BasePos(i)*z(476)*z(31)-2*BasePos(i)*z(482)*z(244)-2*BasePos(i)*z(484)*z
    (243)-2*z(196)*z(450)*z(30)-z(218)*z(453)*z(32)-z(218)*z(456)*z(31)) +
    0.5*MEndEffector*(2*z(276)*z(289)+2*z(276)*z(297)+z(292)*z(450)*z(30)
    +2*BasePos(i)*z(472)*z(30)+2* ...
z(276)*z(453)*z(32)+2*z(276)*z(456)*z(31)-2*BasePos(i)*z(193)-z(292)*z(289)
    -z(292)*z(297)-2*BasePos(i)*z(84)*z(291)-2*BasePos(i)*z(85)*z(290)-2*
    BasePos(i)*z(98)*z(301)-2*BasePos(i)*z(474)*z(32)-2*BasePos(i)*z(476)*z
    (31)-2*BasePos(i)*z(627)*z(300)-2*z(276)* ...
z(450)*z(30)-z(292)*z(453)*z(32)-z(292)*z(456)*z(31)) - z(85)*z(591) - z
    (85)*z(592) - z(131)*z(524) - z(131)*z(525) - 0.5*MBar3*(2*BasePos(i)*z
    (161)+z(196)*z(450)*z(30)+2*BasePos(i)*z(130)*z(213)+2*BasePos(i)*z
    (131)*z(214)+2*BasePos(i)*z(474)* ...
```

```
z(32)+2*BasePos(i)*z(476)*z(31)-z(196)*z(215)-2*BasePos(i)*z(472)*z(30)-z
    (196)*z(453)*z(32)-z(196)*z(456)*z(31)) - 0.5*MLink1*(2*BasePos(i)*z
    (193)+z(165)*z(450)*z(30)+2*BasePos(i)*z(84)*z(186)+2*BasePos(i)*z(474)
    *z(32)+2*BasePos(i)*z(476)*z(31)-z(165)* ...
z(191)-2*BasePos(i)*z(85)*z(190)-2*BasePos(i)*z(472)*z(30)-z(165)*z(453)*z
    (32)-z(165)*z(456)*z(31)) - MLink2*(BasePos(i)*z(193)+BasePos(i)*z(84)*
    z(291)+BasePos(i)*z(85)*z(290)+BasePos(i)*z(98)*z(284)+BasePos(i)*z
    (474)*z(32)+BasePos(i)*z(476)*z(31)+z(276)* ...
z(450)*z(30)-z(276)*z(289)-BasePos(i)*z(472)*z(30)-BasePos(i)*z(627)*z(288)
    -z(276)*z(453)*z(32)-z(276)*z(456)*z(31));
z(712) = z(635) + 0.5*z(361)*z(636);
z(713) = -0.5*z(636) - z(361)*z(690);
z(714) = -0.5*z(637) - z(361)*z(691);
z(715) = -0.5*z(638) - z(361)*z(689);
z(717) = z(641) - z(361)*z(698);
z(718) = 0.5*z(643) - z(361)*z(697);
z(719) = 0.5*z(646) - z(361)*z(692);
z(720) = 0.5*z(649) - z(361)*z(693);
z(721) = 0.5*z(650) - z(361)*z(694);
z(722) = z(355)*z(686);
z(723) = z(355)*z(687);
z(724) = z(361)*z(652);
z(725) = z(361)*z(695);
z(727) = z(712) + z(355)*z(722) - z(361)*z(713);
z(728) = -0.5*z(724) - z(344)*z(715) - z(357)*z(722) - z(363)*z(713);
z(729) = z(341)*z(715) - z(354)*z(722) - z(360)*z(713);
z(730) = z(714) + z(342)*z(715) - z(358)*z(722) - z(364)*z(713);
z(735) = -z(656) - z(344)*z(705) - z(363)*z(695);
z(736) = -z(657) - z(344)*z(704) - z(363)*z(696);
```

```
z(737) = -z(659) - z(344)*z(706) - z(363)*z(698);
z(739) = 0.5*z(344)*z(707) - 0.5*z(662) - z(363)*z(697);
z(740) = 0.5*z(344)*z(708) - 0.5*z(663) - z(363)*z(692);
z(741) = 0.5*z(344)*z(709) - 0.5*z(664) - z(363)*z(693);
z(742) = 0.5*z(344)*z(710) - 0.5*z(666) - z(363)*z(694);
z(744) = z(357)*z(687);
z(746) = z(667) + z(357)*z(688) - z(344)*z(711) - z(363)*z(699);
z(752) = z(354)*z(687) - z(671);
z(757) = -z(360)*z(692) - 0.5*z(341)*z(708);
z(758) = -z(360)*z(693) - 0.5*z(341)*z(709);
z(759) = -z(360)*z(694) - 0.5*z(341)*z(710);
z(762) = z(341)*z(705) - z(360)*z(695);
z(763) = z(341)*z(704) - z(360)*z(696);
z(764) = -z(360)*z(697) - 0.5*z(341)*z(707);
z(765) = z(341)*z(706) - z(360)*z(698);
z(766) = z(672) + z(341)*z(711) + z(354)*z(688) - z(360)*z(699);
z(775) = z(676) - z(364)*z(692) - 0.5*z(342)*z(708);
z(776) = z(677) - z(364)*z(693) - 0.5*z(342)*z(709);
z(777) = z(678) - z(364)*z(694) - 0.5*z(342)*z(710);
z(780) = z(679) + z(342)*z(705) - z(364)*z(695);
z(781) = z(680) + z(342)*z(704) - z(364)*z(696);
z(782) = z(681) + z(342)*z(706) - z(364)*z(698);
z(783) = z(684) - z(364)*z(697) - 0.5*z(342)*z(707);
z(785) = z(358)*z(687);
z(786) = z(685) + z(342)*z(711) + z(358)*z(688) - z(364)*z(699);
z(792) = z(13)*z(355);
z(793) = z(14)*z(355);
z(794) = z(13)*z(357);
z(795) = z(14)*z(357);
```

```
z(796) = z(13)*z(354);

z(797) = z(14)*z(354);

z(798) = z(13)*z(358);

z(799) = z(14)*z(358);

z(800) = z(219)*z(361);

z(801) = z(217)*z(361);

z(802) = z(195)*z(361);

z(803) = z(221)*z(361);

z(804) = z(144)*z(361);

z(805) = z(147)*z(361);

z(806) = 0.5*z(218)*z(344) + 0.5*z(219)*z(363);

z(807) = z(217)*z(363);

z(808) = z(195)*z(363);

z(809) = -z(196)*z(344) - z(221)*z(363);

z(810) = BasePos(i)*z(344) - z(144)*z(363);

z(811) = z(147)*z(363);

z(812) = 0.5*z(219)*z(360) - 0.5*z(218)*z(341);

z(813) = z(217)*z(360);

z(814) = z(195)*z(360);

z(815) = z(196)*z(341) - z(221)*z(360);

z(816) = -BasePos(i)*z(341) - z(144)*z(360);

z(817) = z(147)*z(360);

z(818) = 0.5*z(219)*z(364) - 0.5*z(220) - 0.5*z(218)*z(342);

z(819) = 0.5*z(216) + 0.5*z(217)*z(364);

z(820) = z(194) + z(195)*z(364);

z(821) = z(222) + z(196)*z(342) - z(221)*z(364);

z(822) = z(143) - BasePos(i)*z(342) - z(144)*z(364);

z(823) = z(148) - z(147)*z(364);

z(824) = z(199)*z(361);
```

```
z(825) = -z(199)*z(363) - 0.5*z(196)*z(344);
z(826) = 0.5*z(196)*z(341) - z(199)*z(360);
z(827) = 0.5*z(194) + 0.5*z(195)*z(364);
z(828) = z(200) + 0.5*z(196)*z(342) - z(199)*z(364);
z(829) = z(145)*z(361);
z(830) = z(145)*z(363);
z(831) = z(145)*z(360);
z(832) = z(145)*z(364) - z(146);
z(833) = z(274)*z(361);
z(834) = z(173)*z(361);
z(835) = z(169)*z(361);
z(836) = z(293)*z(361);
z(837) = z(273)*z(361);
z(838) = -z(274)*z(363) - z(276)*z(344);
z(839) = BasePos(i)*z(344) - z(173)*z(363);
z(840) = z(169)*z(363);
z(841) = 0.5*z(292)*z(344) + 0.5*z(293)*z(363);
z(842) = z(273)*z(363);
z(843) = z(276)*z(341) - z(274)*z(360);
z(844) = -BasePos(i)*z(341) - z(173)*z(360);
z(845) = z(169)*z(360);
z(846) = 0.5*z(293)*z(360) - 0.5*z(292)*z(341);
z(847) = z(273)*z(360);
z(848) = z(275) + z(276)*z(342) - z(274)*z(364);
z(849) = z(172) - BasePos(i)*z(342) - z(173)*z(364);
z(850) = z(168) + z(169)*z(364);
z(851) = 0.5*z(293)*z(364) - 0.5*z(294) - 0.5*z(292)*z(342);
z(852) = z(272) + z(273)*z(364);
z(853) = z(170)*z(361);
```

```
z(854) = -z(170)*z(363) - 0.5*z(165)*z(344);

z(855) = 0.5*z(165)*z(341) - z(170)*z(360);

z(856) = z(171) + 0.5*z(165)*z(342) - z(170)*z(364);

z(857) = 0.5*z(168) + 0.5*z(169)*z(364);

z(858) = 0.5*z(272) + 0.5*z(273)*z(364);

z(859) = z(68)*z(361);

z(860) = z(65)*z(361);

z(861) = z(63)*z(361);

z(862) = LCy*z(344) - z(68)*z(363);

z(863) = -LCx*z(344) - z(65)*z(363);

z(864) = z(63)*z(363);

z(865) = -LCy*z(341) - z(68)*z(360);

z(866) = LCx*z(341) - z(65)*z(360);

z(867) = z(63)*z(360);

z(868) = z(67) - LCy*z(342) - z(68)*z(364);

z(869) = LCx*z(342) - z(66) - z(65)*z(364);

z(870) = z(64) - z(63)*z(364);

z(871) = z(33)*z(355);

z(872) = z(33)*z(357);

z(873) = z(33)*z(354);

z(874) = z(33)*z(358);

z(875) = EEForcex*z(50)*z(833) + EEForcex*z(88)*z(835) + EEForcex*z(102)*z
    (837) + EEForcey*z(5)*z(25) + EEForcey*z(53)*z(805) + EEForcey*z(305)*z
    (835) + EEForcey*z(309)*z(837) + z(422)*z(15)*z(871) + z(423)*z(3)*z
    (25) + z(423)*z(306)* ...
z(835) + z(423)*z(310)*z(837) + 0.5*EEForcey*z(51)*z(836) + 0.5*z(423)*z
    (52)*z(836) + 0.5*z(415)*(2*z(49)*z(804)+2*z(52)*z(824)+2*z(54)*z(805)
    -2*z(3)*z(25)-z(492)*z(802)) + 0.5*z(418)*(2*z(49)*z(834)+2*z(52)*z
    (853)+2*z(54)*z(805)-2*z(3)* ...
```

```
z(25)-z(306)*z(835)) + 0.5*z(419)*(2*z(49)*z(834)+2*z(52)*z(833)+2*z(54)*z
    (805)-2*z(3)*z(25)-2*z(306)*z(835)-z(310)*z(837)) + 0.5*z(414)*(2*z(49)
    *z(804)+2*z(52)*z(803)+2*z(54)*z(805)-2*z(3)*z(25)-2*z(492)*z(802)-z
    (52)*z(800)-z(488)* ...
z(801)) + z(792)*z(427) + z(793)*z(429) - EEForcex*z(43)*z(834) - EEForcex*
    z(47)*z(805) - EEForcey*z(48)*z(834) - EEForcey*z(51)*z(833) - z(421)*z
    (3)*z(25) - z(423)*z(49)*z(834) - z(423)*z(52)*z(833) - z(423)*z(54)*z
    (805) - 0.5* ...
EEForcex*z(50)*z(836) - z(416)*(z(3)*z(25)+z(52)*z(829)-z(49)*z(804)-z(54)*
    z(805)) - z(420)*(z(3)*z(25)-z(49)*z(859)-z(52)*z(860)-z(54)*z(861)) -
    z(1)*z(424) - z(2)*z(426);
z(880) = z(876)*z(6) + EEForcex*z(43)*z(839) + EEForcex*z(88)*z(840) +
    EEForcex*z(102)*z(842) + EEForcey*z(48)*z(839) + EEForcey*z(51)*z(838)
    + EEForcey*z(51)*z(841) + EEForcey*z(53)*z(811) + EEForcey*z(305)*z
    (840) + EEForcey*z(309)* ...
z(842) + z(423)*z(49)*z(839) + z(423)*z(52)*z(838) + z(423)*z(52)*z(841) +
    z(423)*z(306)*z(840) + z(423)*z(310)*z(842) + 0.5*z(415)*(2*z(54)*z
    (811)-2*LE*z(6)-2*z(49)*z(810)-2*z(52)*z(825)-z(492)*z(808)) + 0.5*z
    (418)*(2*z(54)*z(811)-2*LE* ...
z(6)-2*z(49)*z(839)-2*z(52)*z(854)-z(306)*z(840)) + 0.5*z(419)*(2*z(54)*z
    (811)-2*LE*z(6)-2*z(49)*z(839)-2*z(52)*z(838)-2*z(306)*z(840)-z(310)*z
    (842)) + 0.5*z(414)*(2*z(54)*z(811)-2*LE*z(6)-2*z(49)*z(810)-2*z(52)*z
    (806)-2*z(52)*z(809)-2* ...
z(492)*z(808)-z(488)*z(807)) - z(877)*z(1) - z(878)*z(4) - z(879)*z(6) -
    EEForcex*z(47)*z(811) - EEForcex*z(50)*z(838) - EEForcex*z(50)*z(841) -
     z(422)*z(15)*z(872) - z(423)*z(54)*z(811) - z(416)*(LE*z(6)+z(49)*z
    (810)+z(52)*z(830)- ...
z(54)*z(811)) - z(420)*(LE*z(6)+z(49)*z(862)+z(52)*z(863)-z(54)*z(864)) - z
    (425) - z(794)*z(427) - z(795)*z(429);
```

```
z(881) = EEForcex*z(43)*z(844) + EEForcex*z(88)*z(845) + EEForcex*z(102)*z
    (847) + EEForcey*z(48)*z(844) + EEForcey*z(51)*z(843) + EEForcey*z(51)*
    z(846) + EEForcey*z(53)*z(817) + EEForcey*z(305)*z(845) + EEForcey*z
    (309)*z(847) + z(423)* ...
z(49)*z(844) + z(423)*z(52)*z(843) + z(423)*z(52)*z(846) + z(423)*z(306)*z
    (845) + z(423)*z(310)*z(847) + 0.5*z(415)*(2*z(54)*z(817)-2*z(49)*z
    (816)-2*z(52)*z(826)-z(492)*z(814)) + 0.5*z(418)*(2*z(54)*z(817)-2*z
    (49)*z(844)-2*z(52)* ...
z(855)-z(306)*z(845)) + 0.5*z(419)*(2*z(54)*z(817)-2*z(49)*z(844)-2*z(52)*z
    (843)-2*z(306)*z(845)-z(310)*z(847)) + 0.5*z(414)*(2*z(54)*z(817)-2*z
    (49)*z(816)-2*z(52)*z(812)-2*z(52)*z(815)-2*z(492)*z(814)-z(488)*z(813)
    ) - EEForcex*z(47)* ...
z(817) - EEForcex*z(50)*z(843) - EEForcex*z(50)*z(846) - z(423)*z(54)*z
    (817) - z(422)*(LW*z(18)+z(15)*z(873)) - z(416)*(z(49)*z(816)+z(52)*z
    (831)-z(54)*z(817)) - z(420)*(z(49)*z(865)+z(52)*z(866)-z(54)*z(867)) -
     z(428) - z(796)*z(427) - ...
z(797)*z(429);
z(882) = EEForcex*z(43)*z(849) + EEForcex*z(47)*z(823) + EEForcex*z(88)*z
    (850) + EEForcex*z(102)*z(852) + EEForcey*z(48)*z(849) + EEForcey*z(51)
    *z(848) + EEForcey*z(51)*z(851) + EEForcey*z(305)*z(850) + EEForcey*z
    (309)*z(852) + z(423)* ...
z(49)*z(849) + z(423)*z(52)*z(848) + z(423)*z(52)*z(851) + z(423)*z(54)*z
    (823) + z(423)*z(306)*z(850) + z(423)*z(310)*z(852) - EEForcex*z(50)*z
    (848) - EEForcex*z(50)*z(851) - EEForcey*z(53)*z(823) - z(422)*z(15)*z
    (874) - z(416)*(z(49)* ...
z(822)+z(52)*z(832)+z(54)*z(823)) - z(420)*(z(49)*z(868)+z(52)*z(869)+z(54)
    *z(870)) - z(415)*(z(49)*z(822)+z(52)*z(828)+z(54)*z(823)+z(492)*z(827)
    ) - z(418)*(z(49)*z(849)+z(52)*z(856)+z(54)*z(823)+z(306)*z(857)) - z
    (419)*(z(49)*z(849)+ ...
```

```
z(52)*z(848)+z(54)*z(823)+z(306)*z(850)+z(310)*z(858)) - z(414)*(z(49)*z
   (822)+z(52)*z(818)+z(52)*z(821)+z(54)*z(823)+z(488)*z(819)+z(492)*z
   (820)) - z(798)*z(427) - z(799)*z(429);
z(887) = z(727)*z(748) - z(728)*z(747);
z(888) = z(769)*z(790) - z(770)*z(789);
z(889) = z(727)*z(749) - z(729)*z(747);
z(890) = z(768)*z(790) - z(770)*z(788);
z(891) = z(727)*z(750) - z(730)*z(747);
z(892) = z(768)*z(789) - z(769)*z(788);
z(893) = z(728)*z(749) - z(729)*z(748);
z(894) = z(767)*z(790) - z(770)*z(787);
z(895) = z(728)*z(750) - z(730)*z(748);
z(896) = z(767)*z(789) - z(769)*z(787);
z(897) = z(729)*z(750) - z(730)*z(749);
z(898) = z(767)*z(788) - z(768)*z(787);
z(899) = z(887)*z(888) + z(891)*z(892) + z(893)*z(894) + z(897)*z(898) - z
   (889)*z(890) - z(895)*z(896);
z(910) = z(728)*z(769) - z(729)*z(768);
z(911) = z(728)*z(770) - z(730)*z(768);
z(912) = z(729)*z(770) - z(730)*z(769);
z(913) = z(788)*z(912) + z(790)*z(910) - z(789)*z(911);
z(914) = z(727)*z(769) - z(729)*z(767);
z(915) = z(727)*z(770) - z(730)*z(767);
z(916) = z(787)*z(912) + z(790)*z(914) - z(789)*z(915);
z(917) = z(727)*z(768) - z(728)*z(767);
z(918) = z(787)*z(911) + z(790)*z(917) - z(788)*z(915);
z(919) = z(787)*z(910) + z(789)*z(917) - z(788)*z(914);
z(920) = z(788)*z(897) + z(790)*z(893) - z(789)*z(895);
z(921) = z(787)*z(897) + z(790)*z(889) - z(789)*z(891);
```

```
z(922) = z(787)*z(895) + z(790)*z(887) - z(788)*z(891);
z(923) = z(787)*z(893) + z(789)*z(887) - z(788)*z(889);
z(924) = z(768)*z(897) + z(770)*z(893) - z(769)*z(895);
z(925) = z(767)*z(897) + z(770)*z(889) - z(769)*z(891);
z(926) = z(767)*z(895) + z(770)*z(887) - z(768)*z(891);
z(927) = z(767)*z(893) + z(769)*z(887) - z(768)*z(889);


z(410) = z(362)*y2pp(i) + z(365)*x2pp(i) - z(406) - z(362)*y1pp(i) - z(365)
    *x1pp(i);
z(411) = z(402) + z(343)*x1pp(i) - z(343)*x2pp(i);
z(412) = z(367)*qBar3pp(i) - z(407) - z(367)*qLink1pp(i);
z(413) = z(368)*qBar3pp(i) - z(408) - z(368)*qLink1pp(i);
z(409) = z(405) + z(356)*y2pp(i) + z(359)*x1pp(i) - z(356)*y1pp(i) - z(359)
    *x2pp(i);
z(731) = z(726) + z(716)*y1pp(i) + z(717)*BasePospp(i) + z(718)*qBar3pp(i)
    + z(719)*qLink1pp(i) + z(713)*z(410) + z(715)*z(411) + z(720)*z(412) +
    z(721)*z(413) - z(723)*y2pp(i) - z(725)*x1pp(i) - z(722)*z(409);
z(883) = z(731) - z(875);
z(771) = z(766) + z(752)*y2pp(i) + z(757)*qLink1pp(i) + z(762)*x1pp(i) + z
    (763)*y1pp(i) + z(764)*qBar3pp(i) + z(765)*BasePospp(i) + z(753)*z(409)
     + z(754)*z(411) + z(755)*z(410) + z(758)*z(412) + z(759)*z(413) - z
    (670)*x2pp(i);
z(885) = z(771) - z(881);
z(751) = z(746) + z(735)*x1pp(i) + z(736)*y1pp(i) + z(737)*BasePospp(i) + z
    (739)*qBar3pp(i) + z(740)*qLink1pp(i) + z(744)*y2pp(i) + z(732)*z(410)
    + z(734)*z(411) + z(741)*z(412) + z(742)*z(413) + z(743)*z(409);
z(884) = z(751) - z(880);
```

```
z(791) = z(786) + z(775)*qLink1pp(i) + z(780)*x1pp(i) + z(781)*y1pp(i) + z
    (782)*BasePospp(i) + z(783)*qBar3pp(i) + z(785)*y2pp(i) + z(772)*z(411)
     + z(773)*z(410) + z(776)*z(412) + z(777)*z(413) + z(784)*z(409);
z(886) = z(791) - z(882);
z(928) = (z(903)*z(883)+z(920)*z(885)-z(913)*z(884)-z(924)*z(886))/z(899);
theta_1pp(i) = -z(928);
z(929) = (z(906)*z(883)+z(921)*z(885)-z(916)*z(884)-z(925)*z(886))/z(899);
phi_1pp(i) = z(929);
z(930) = (z(908)*z(883)+z(922)*z(885)-z(918)*z(884)-z(926)*z(886))/z(899);
phi_2pp(i) = -z(930);
z(931) = (z(909)*z(883)+z(923)*z(885)-z(919)*z(884)-z(927)*z(886))/z(899);
psipp(i) = z(931);

% Energy
z(933) = 9.81*MC*(LCx*z(49)+LCy*z(52)+LCz*z(54)-LE*z(8)) - 9.81*z(669)*z
    (20) - 9.81*z(932)*z(8) - 9.81*MBase*(LE*z(8)-z(142)*z(54)-BasePos(i)*z
    (52)) - 4.905*MBar3*(2*LE*z(8)-2*z(142)*z(54)-2*BasePos(i)*z(52)-LBar3*
    z(490)) - 4.905*MLink1*(2*LE* ...
z(8)-2*z(302)*z(54)-2*BasePos(i)*z(52)-LLink1*z(304)) - 9.81*MEndEffector*(
    LE*z(8)-LLink1*z(304)-LLink2*z(308)-z(302)*z(54)-BasePos(i)*z(52)) -
    4.905*MBar2*(2*LE*z(8)-2*LBar3*z(490)-2*z(142)*z(54)-2*BasePos(i)*z(52)
    -LBar2*z(486)) - 4.905* ...
MLink2*(2*LE*z(8)-2*LLink1*z(304)-2*z(302)*z(54)-2*BasePos(i)*z(52)-LLink2*
    z(308));
PotentialEnergy(i) = z(933);
```

```
KineticEnergy(i) = 0.5*IBaseyy*(gammap(i)+z(47)*betap(i))^2 + 0.5*IBasezz*(
    z(43)*betap(i)+z(44)*psip(i))^2 + 0.5*IBasexx*(z(41)*psip(i)-z(50)*
    betap(i))^2 + 0.5*IBar2yy*(z(115)*gammap(i)+z(116)*betap(i)+z(117)*psip
    (i))^2 + 0.5*IBar2zz*(z(118)*gammap(i)+z(119)*betap(i)+ ...
z(120)*psip(i))^2 + 0.5*IBar3yy*(z(130)*gammap(i)+z(132)*betap(i)+z(133)*
    psip(i))^2 + 0.5*ILink1yy*(z(84)*gammap(i)+z(86)*betap(i)+z(87)*psip(i)
    )^2 + 0.5*ILink2yy*(z(98)*gammap(i)+z(99)*betap(i)+z(100)*psip(i))^2 +
    0.5*ILink2zz*(z(101)*gammap(i)+z(102)*betap(i)+z(103)*psip(i))^ ...
2 + 0.5*IBar3zz*(z(131)*gammap(i)-z(134)*betap(i)-z(135)*psip(i))^2 + 0.5*
    ILink1zz*(z(85)*gammap(i)-z(88)*betap(i)-z(89)*psip(i))^2 + 0.5*(gammap
    (i)+z(47)*betap(i))*(Izz*(gammap(i)+z(47)*betap(i))+Izx*(z(43)*betap(i)
    +z(44)*psip(i))+Iyz*(z(41)*psip(i)-z(50)*betap(i))) + 0.5*( ...
z(43)*betap(i)+z(44)*psip(i))*(Izx*(gammap(i)+z(47)*betap(i))+Ixx*(z(43)*
    betap(i)+z(44)*psip(i))+Ixy*(z(41)*psip(i)-z(50)*betap(i))) + 0.5*(z
    (41)*psip(i)-z(50)*betap(i))*(Iyz*(gammap(i)+z(47)*betap(i))+Ixy*(z(43)
    *betap(i)+z(44)*psip(i))+Iyy*(z(41)*psip(i)-z(50)*betap(i))) + 0.5* ...
IBar3xx*(z(50)*betap(i)-z(41)*psip(i)-qBar3p(i))^2 + 0.5*ILink1xx*(z(50)*
    betap(i)-z(41)*psip(i)-qLink1p(i))^2 + 0.5*MBase*(LE^2*phi_1p(i)^2+z
    (25)^2*theta_1p(i)^2+(z(147)*betap(i)+z(148)*psip(i))^2+2*z(25)*z(473)*
    theta_1p(i)*(z(147)*betap(i)+z(148)*psip(i))+(BasePos(i)*gammap(i)- ...
z(143)*psip(i)-z(144)*betap(i))^2+2*LE*z(472)*phi_1p(i)*(BasePos(i)*gammap(
    i)-z(143)*psip(i)-z(144)*betap(i))+x1p(i)^2+y1p(i)^2+2*z(5)*z(25)*
    theta_1p(i)*y1p(i)+(z(145)*betap(i)+z(146)*psip(i)-BasePosp(i))^2+2*z
    (47)*(z(147)*betap(i)+z(148)*psip(i))*x1p(i)+2*z(50)*x1p(i)*(z(145)*
    betap(i)+ ...
```

```
z(146)*psip(i)-BasePosp(i))+2*LE*z(450)*phi_1p(i)*(z(145)*betap(i)+z(146)*
    psip(i)-BasePosp(i))-2*LE*z(471)*phi_1p(i)*(z(147)*betap(i)+z(148)*psip
    (i))-2*z(25)*z(474)*theta_1p(i)*(BasePos(i)*gammap(i)-z(143)*psip(i)-z
    (144)*betap(i))-2*LE*z(1)*phi_1p(i)*x1p(i)-2*LE*z(4)*phi_1p(i)*y1p(i)
    -2* ...
z(53)*(z(147)*betap(i)+z(148)*psip(i))*y1p(i)-2*z(51)*y1p(i)*(z(145)*betap(
    i)+z(146)*psip(i)-BasePosp(i))-2*z(25)*z(453)*theta_1p(i)*(z(145)*betap
    (i)+z(146)*psip(i)-BasePosp(i))-2*z(43)*(BasePos(i)*gammap(i)-z(143)*
    psip(i)-z(144)*betap(i))*x1p(i)-2*z(48)*(BasePos(i)*gammap(i)-z(143)*
    psip(i)- ...
z(144)*betap(i))*y1p(i)) + 0.5*MC*(LE^2*phi_1p(i)^2+z(25)^2*theta_1p(i)^2+(
    z(63)*betap(i)+z(64)*psip(i))^2+(LCx*gammap(i)+z(65)*betap(i)-z(66)*
    psip(i))^2+2*z(25)*z(473)*theta_1p(i)*(z(63)*betap(i)+z(64)*psip(i))+(
    LCy*gammap(i)-z(67)*psip(i)-z(68)*betap(i))^2+2*z(25)*z(453)* ...
theta_1p(i)*(LCx*gammap(i)+z(65)*betap(i)-z(66)*psip(i))+2*LE*z(472)*phi_1p
    (i)*(LCy*gammap(i)-z(67)*psip(i)-z(68)*betap(i))+x1p(i)^2+y1p(i)^2+2*z
    (5)*z(25)*theta_1p(i)*y1p(i)+2*z(47)*(z(63)*betap(i)+z(64)*psip(i))*x1p
    (i)+2*z(51)*(LCx*gammap(i)+z(65)*betap(i)-z(66)*psip(i))*y1p(i)-2*LE*z
    (471)* ...
phi_1p(i)*(z(63)*betap(i)+z(64)*psip(i))-2*LE*z(450)*phi_1p(i)*(LCx*gammap(
    i)+z(65)*betap(i)-z(66)*psip(i))-2*z(25)*z(474)*theta_1p(i)*(LCy*gammap
    (i)-z(67)*psip(i)-z(68)*betap(i))-2*LE*z(1)*phi_1p(i)*x1p(i)-2*LE*z(4)*
    phi_1p(i)*y1p(i)-2*z(53)*(z(63)*betap(i)+z(64)*psip(i))*y1p(i)-2*z(50)
    *( ...
LCx*gammap(i)+z(65)*betap(i)-z(66)*psip(i))*x1p(i)-2*z(43)*(LCy*gammap(i)-z
    (67)*psip(i)-z(68)*betap(i))*x1p(i)-2*z(48)*(LCy*gammap(i)-z(67)*psip(i
    )-z(68)*betap(i))*y1p(i)) + 0.5*ILink2xx*(z(50)*betap(i)-z(41)*psip(i)-
    qLink1p(i)-z(372))^2 + 0.5*IBar2xx*(z(50)*betap(i)-z(41)*psip(i)- ...
```

```
qLink1p(i)-z(372)-z(373))^2 - 0.5*ME*(2*LE*z(1)*phi_1p(i)*x1p(i)+2*LE*z(4)*
    phi_1p(i)*y1p(i)-LE^2*phi_1p(i)^2-z(25)^2*theta_1p(i)^2-x1p(i)^2-y1p(i)
    ^2-2*z(5)*z(25)*theta_1p(i)*y1p(i)) - 0.5*MW*(2*LW*z(13)*phi_2p(i)*x2p(
    i)+2*LW*z(16)*phi_2p(i)*y2p(i)-LW^2*phi_2p(i)^2-z(33)^2*theta_2p(i)^2-
    ...
x2p(i)^2-y2p(i)^2-2*z(17)*z(33)*theta_2p(i)*y2p(i)) - 0.125*MBar3*(8*LE*z
    (471)*phi_1p(i)*(z(147)*betap(i)+z(148)*psip(i))+8*z(25)*z(474)*
    theta_1p(i)*(BasePos(i)*gammap(i)-z(143)*psip(i)-z(144)*betap(i))+8*LE*
    z(1)*phi_1p(i)*x1p(i)+8*LE*z(4)*phi_1p(i)*y1p(i)+8*z(53)*(z(147)*betap(
    i)+ ...
z(148)*psip(i))*y1p(i)+4*z(134)*x1p(i)*(z(195)*betap(i)-z(194)*psip(i)-
    LBar3*qBar3p(i))+4*z(491)*y1p(i)*(z(195)*betap(i)-z(194)*psip(i)-LBar3*
    qBar3p(i))+8*z(43)*(BasePos(i)*gammap(i)-z(143)*psip(i)-z(144)*betap(i)
    )*x1p(i)+8*z(48)*(BasePos(i)*gammap(i)-z(143)*psip(i)-z(144)*betap(i))*
    y1p(i)+4* ...
z(25)*z(468)*theta_1p(i)*(z(195)*betap(i)-z(194)*psip(i)-LBar3*qBar3p(i))
    +4*z(50)*x1p(i)*(z(196)*gammap(i)+2*z(199)*betap(i)+2*z(200)*psip(i)+2*
    BasePosp(i))+4*LE*z(450)*phi_1p(i)*(z(196)*gammap(i)+2*z(199)*betap(i)
    +2*z(200)*psip(i)+2*BasePosp(i))+4*z(507)*(z(147)*betap(i)+z(148)* ...
psip(i))*(z(195)*betap(i)-z(194)*psip(i)-LBar3*qBar3p(i))-4*LE^2*phi_1p(i)
    ^2-4*z(25)^2*theta_1p(i)^2-4*(z(147)*betap(i)+z(148)*psip(i))^2-8*z(25)
    *z(473)*theta_1p(i)*(z(147)*betap(i)+z(148)*psip(i))-4*(BasePos(i)*
    gammap(i)-z(143)*psip(i)-z(144)*betap(i))^2-8*LE*z(472)*phi_1p(i)*( ...
BasePos(i)*gammap(i)-z(143)*psip(i)-z(144)*betap(i))-4*x1p(i)^2-4*y1p(i)
    ^2-8*z(5)*z(25)*theta_1p(i)*y1p(i)-8*z(47)*(z(147)*betap(i)+z(148)*psip
    (i))*x1p(i)-(z(195)*betap(i)-z(194)*psip(i)-LBar3*qBar3p(i))^2-4*LE*z
    (466)*phi_1p(i)*(z(195)*betap(i)-z(194)*psip(i)-LBar3*qBar3p(i))-(z
    (196)* ...
```

```
gammap(i)+2*z(199)*betap(i)+2*z(200)*psip(i)+2*BasePosp(i))^2-4*z(51)*y1p(i
    )*(z(196)*gammap(i)+2*z(199)*betap(i)+2*z(200)*psip(i)+2*BasePosp(i))
    -4*z(25)*z(453)*theta_1p(i)*(z(196)*gammap(i)+2*z(199)*betap(i)+2*z
    (200)*psip(i)+2*BasePosp(i))-4*z(130)*(BasePos(i)*gammap(i)-z(143)*psip
    (i)- ...
z(144)*betap(i))*(z(195)*betap(i)-z(194)*psip(i)-LBar3*qBar3p(i))) - 0.125*
    MLink1*(8*LE*z(471)*phi_1p(i)*(z(147)*betap(i)+z(148)*psip(i))+8*z(25)*
    z(474)*theta_1p(i)*(BasePos(i)*gammap(i)-z(172)*psip(i)-z(173)*betap(i)
    )+8*LE*z(1)*phi_1p(i)*x1p(i)+8*LE*z(4)*phi_1p(i)*y1p(i)+8*z(53)*( ...
z(147)*betap(i)+z(148)*psip(i))*y1p(i)+4*z(88)*x1p(i)*(z(169)*betap(i)-z
    (168)*psip(i)-LLink1*qLink1p(i))+4*z(305)*y1p(i)*(z(169)*betap(i)-z
    (168)*psip(i)-LLink1*qLink1p(i))+8*z(43)*(BasePos(i)*gammap(i)-z(172)*
    psip(i)-z(173)*betap(i))*x1p(i)+8*z(48)*(BasePos(i)*gammap(i)-z(172)*
    psip(i)- ...
z(173)*betap(i))*y1p(i)+4*z(25)*z(455)*theta_1p(i)*(z(169)*betap(i)-z(168)*
    psip(i)-LLink1*qLink1p(i))+4*z(50)*x1p(i)*(z(165)*gammap(i)+2*z(170)*
    betap(i)+2*z(171)*psip(i)+2*BasePosp(i))+4*LE*z(450)*phi_1p(i)*(z(165)*
    gammap(i)+2*z(170)*betap(i)+2*z(171)*psip(i)+2*BasePosp(i))+4* ...
z(600)*(z(147)*betap(i)+z(148)*psip(i))*(z(169)*betap(i)-z(168)*psip(i)-
    LLink1*qLink1p(i))-4*LE^2*phi_1p(i)^2-4*z(25)^2*theta_1p(i)^2-4*(z(147)
    *betap(i)+z(148)*psip(i))^2-8*z(25)*z(473)*theta_1p(i)*(z(147)*betap(i)
    +z(148)*psip(i))-4*(BasePos(i)*gammap(i)-z(172)*psip(i)-z(173)* ...
betap(i))^2-8*LE*z(472)*phi_1p(i)*(BasePos(i)*gammap(i)-z(172)*psip(i)-z
    (173)*betap(i))-4*x1p(i)^2-4*y1p(i)^2-8*z(5)*z(25)*theta_1p(i)*y1p(i)
    -8*z(47)*(z(147)*betap(i)+z(148)*psip(i))*x1p(i)-(z(169)*betap(i)-z
    (168)*psip(i)-LLink1*qLink1p(i))^2-4*LE*z(452)*phi_1p(i)*(z(169)*betap(
    i)-z(168)* ...
```

```
psip(i)-LLink1*qLink1p(i))-(z(165)*gammap(i)+2*z(170)*betap(i)+2*z(171)*
    psip(i)+2*BasePosp(i))^2-4*z(51)*y1p(i)*(z(165)*gammap(i)+2*z(170)*
    betap(i)+2*z(171)*psip(i)+2*BasePosp(i))-4*z(25)*z(453)*theta_1p(i)*(z
    (165)*gammap(i)+2*z(170)*betap(i)+2*z(171)*psip(i)+2*BasePosp(i))-4*z
    (84)*( ...
BasePos(i)*gammap(i)-z(172)*psip(i)-z(173)*betap(i))*(z(169)*betap(i)-z
    (168)*psip(i)-LLink1*qLink1p(i))) - 0.125*MLink2*(8*LE*z(471)*phi_1p(i)
    *(z(147)*betap(i)+z(148)*psip(i))+8*z(25)*z(474)*theta_1p(i)*(BasePos(i
    )*gammap(i)-z(172)*psip(i)-z(173)*betap(i))+8*LE*z(1)*phi_1p(i)*x1p(i)
    +8* ...
LE*z(4)*phi_1p(i)*y1p(i)+8*z(53)*(z(147)*betap(i)+z(148)*psip(i))*y1p(i)+8*
    z(50)*x1p(i)*(z(274)*betap(i)+z(275)*psip(i)+z(276)*gammap(i)+BasePosp(
    i))+8*LE*z(450)*phi_1p(i)*(z(274)*betap(i)+z(275)*psip(i)+z(276)*gammap
    (i)+BasePosp(i))+8*z(43)*(BasePos(i)*gammap(i)-z(172)*psip(i)-z(173)*
    ...
betap(i))*x1p(i)+8*z(48)*(BasePos(i)*gammap(i)-z(172)*psip(i)-z(173)*betap(
    i))*y1p(i)+8*z(88)*x1p(i)*(z(169)*betap(i)-z(168)*psip(i)-LLink1*
    qLink1p(i))+8*z(305)*y1p(i)*(z(169)*betap(i)-z(168)*psip(i)-LLink1*
    qLink1p(i))+8*z(25)*z(455)*theta_1p(i)*(z(169)*betap(i)-z(168)*psip(i)-
    LLink1* ...
qLink1p(i))+8*z(600)*(z(147)*betap(i)+z(148)*psip(i))*(z(169)*betap(i)-z
    (168)*psip(i)-LLink1*qLink1p(i))+4*z(102)*x1p(i)*(z(273)*betap(i)-z
    (272)*psip(i)-LLink2*qLink1p(i)-LLink2*z(372))+4*z(309)*y1p(i)*(z(273)*
    betap(i)-z(272)*psip(i)-LLink2*qLink1p(i)-LLink2*z(372))+4*z(25)* ...
z(624)*theta_1p(i)*(z(273)*betap(i)-z(272)*psip(i)-LLink2*qLink1p(i)-LLink2
    *z(372))+4*z(633)*(z(147)*betap(i)+z(148)*psip(i))*(z(273)*betap(i)-z
    (272)*psip(i)-LLink2*qLink1p(i)-LLink2*z(372))-4*LE^2*phi_1p(i)^2-4*z
    (25)^2*theta_1p(i)^2-4*(z(147)*betap(i)+z(148)*psip(i))^2-8* ...
```

```
z(25)*z(473)*theta_1p(i)*(z(147)*betap(i)+z(148)*psip(i))-4*(BasePos(i)*
    gammap(i)-z(172)*psip(i)-z(173)*betap(i))^2-8*LE*z(472)*phi_1p(i)*(
    BasePos(i)*gammap(i)-z(172)*psip(i)-z(173)*betap(i))-4*x1p(i)^2-4*y1p(i
    )^2-8*z(5)*z(25)*theta_1p(i)*y1p(i)-8*z(47)*(z(147)*betap(i)+z(148)*
    psip(i))*x1p(i)- ...
4*(z(274)*betap(i)+z(275)*psip(i)+z(276)*gammap(i)+BasePosp(i))^2-8*z(51)*
    y1p(i)*(z(274)*betap(i)+z(275)*psip(i)+z(276)*gammap(i)+BasePosp(i))
    -4*(z(169)*betap(i)-z(168)*psip(i)-LLink1*qLink1p(i))^2-8*z(25)*z(453)*
    theta_1p(i)*(z(274)*betap(i)+z(275)*psip(i)+z(276)*gammap(i)+BasePosp(i
    ))- ...
8*LE*z(452)*phi_1p(i)*(z(169)*betap(i)-z(168)*psip(i)-LLink1*qLink1p(i))-8*
    z(84)*(BasePos(i)*gammap(i)-z(172)*psip(i)-z(173)*betap(i))*(z(169)*
    betap(i)-z(168)*psip(i)-LLink1*qLink1p(i))-(z(273)*betap(i)-z(272)*psip
    (i)-LLink2*qLink1p(i)-LLink2*z(372))^2-4*LE*z(622)*phi_1p(i)*( ...
z(273)*betap(i)-z(272)*psip(i)-LLink2*qLink1p(i)-LLink2*z(372))-4*z(96)*(z
    (169)*betap(i)-z(168)*psip(i)-LLink1*qLink1p(i))*(z(273)*betap(i)-z
    (272)*psip(i)-LLink2*qLink1p(i)-LLink2*z(372))-4*z(98)*(BasePos(i)*
    gammap(i)-z(172)*psip(i)-z(173)*betap(i))*(z(273)*betap(i)-z(272)*psip(
    i)- ...
LLink2*qLink1p(i)-LLink2*z(372))) - 0.125*MEndEffector*(8*LE*z(471)*phi_1p(
    i)*(z(147)*betap(i)+z(148)*psip(i))+4*z(25)*z(453)*theta_1p(i)*(z(292)*
    gammap(i)+z(293)*betap(i)+z(294)*psip(i))+8*z(25)*z(474)*theta_1p(i)*(
    BasePos(i)*gammap(i)-z(172)*psip(i)-z(173)*betap(i))+4* ...
z(499)*(z(147)*betap(i)+z(148)*psip(i))*(z(292)*gammap(i)+z(293)*betap(i)+z
    (294)*psip(i))+8*LE*z(1)*phi_1p(i)*x1p(i)+8*LE*z(4)*phi_1p(i)*y1p(i)+8*
    z(53)*(z(147)*betap(i)+z(148)*psip(i))*y1p(i)+4*z(51)*(z(292)*gammap(i)
    +z(293)*betap(i)+z(294)*psip(i))*y1p(i)+8*z(50)*x1p(i)*(z(274)*betap(i)
    + ...
```

```
z(275)*psip(i)+z(276)*gammap(i)+BasePosp(i))+8*LE*z(450)*phi_1p(i)*(z(274)*
    betap(i)+z(275)*psip(i)+z(276)*gammap(i)+BasePosp(i))+8*z(43)*(BasePos(
    i)*gammap(i)-z(172)*psip(i)-z(173)*betap(i))*x1p(i)+8*z(48)*(BasePos(i)
    *gammap(i)-z(172)*psip(i)-z(173)*betap(i))*y1p(i)+8*z(88)*x1p(i)*(z
    (169)* ...
betap(i)-z(168)*psip(i)-LLink1*qLink1p(i))+8*z(305)*y1p(i)*(z(169)*betap(i)
    -z(168)*psip(i)-LLink1*qLink1p(i))+8*z(25)*z(455)*theta_1p(i)*(z(169)*
    betap(i)-z(168)*psip(i)-LLink1*qLink1p(i))+4*(z(292)*gammap(i)+z(293)*
    betap(i)+z(294)*psip(i))*(z(274)*betap(i)+z(275)*psip(i)+z(276)* ...
gammap(i)+BasePosp(i))+8*z(600)*(z(147)*betap(i)+z(148)*psip(i))*(z(169)*
    betap(i)-z(168)*psip(i)-LLink1*qLink1p(i))+8*z(102)*x1p(i)*(z(273)*
    betap(i)-z(272)*psip(i)-LLink2*qLink1p(i)-LLink2*z(372))+8*z(309)*y1p(i
    )*(z(273)*betap(i)-z(272)*psip(i)-LLink2*qLink1p(i)-LLink2*z(372))+8*
    ...
z(25)*z(624)*theta_1p(i)*(z(273)*betap(i)-z(272)*psip(i)-LLink2*qLink1p(i)-
    LLink2*z(372))+8*z(633)*(z(147)*betap(i)+z(148)*psip(i))*(z(273)*betap(
    i)-z(272)*psip(i)-LLink2*qLink1p(i)-LLink2*z(372))-4*LE^2*phi_1p(i)
    ^2-4*z(25)^2*theta_1p(i)^2-4*(z(147)*betap(i)+z(148)*psip(i))^ ...
2-8*z(25)*z(473)*theta_1p(i)*(z(147)*betap(i)+z(148)*psip(i))-(z(292)*
    gammap(i)+z(293)*betap(i)+z(294)*psip(i))^2-4*(BasePos(i)*gammap(i)-z
    (172)*psip(i)-z(173)*betap(i))^2-4*LE*z(450)*phi_1p(i)*(z(292)*gammap(i
    )+z(293)*betap(i)+z(294)*psip(i))-8*LE*z(472)*phi_1p(i)*(BasePos(i)*
    ...
gammap(i)-z(172)*psip(i)-z(173)*betap(i))-4*x1p(i)^2-4*y1p(i)^2-8*z(5)*z
    (25)*theta_1p(i)*y1p(i)-8*z(47)*(z(147)*betap(i)+z(148)*psip(i))*x1p(i)
    -4*(z(274)*betap(i)+z(275)*psip(i)+z(276)*gammap(i)+BasePosp(i))^2-4*z
    (50)*(z(292)*gammap(i)+z(293)*betap(i)+z(294)*psip(i))*x1p(i)-8*z(51)*
    y1p(i)*( ...
```

```
z(274)*betap(i)+z(275)*psip(i)+z(276)*gammap(i)+BasePosp(i))-4*(z(169)*
    betap(i)-z(168)*psip(i)-LLink1*qLink1p(i))^2-8*z(25)*z(453)*theta_1p(i)
    *(z(274)*betap(i)+z(275)*psip(i)+z(276)*gammap(i)+BasePosp(i))-8*LE*z
    (452)*phi_1p(i)*(z(169)*betap(i)-z(168)*psip(i)-LLink1*qLink1p(i))-8*
    ...
z(84)*(BasePos(i)*gammap(i)-z(172)*psip(i)-z(173)*betap(i))*(z(169)*betap(i
    )-z(168)*psip(i)-LLink1*qLink1p(i))-4*(z(273)*betap(i)-z(272)*psip(i)-
    LLink2*qLink1p(i)-LLink2*z(372))^2-8*LE*z(622)*phi_1p(i)*(z(273)*betap(
    i)-z(272)*psip(i)-LLink2*qLink1p(i)-LLink2*z(372))-8*z(96)*( ...
z(169)*betap(i)-z(168)*psip(i)-LLink1*qLink1p(i))*(z(273)*betap(i)-z(272)*
    psip(i)-LLink2*qLink1p(i)-LLink2*z(372))-8*z(98)*(BasePos(i)*gammap(i)-
    z(172)*psip(i)-z(173)*betap(i))*(z(273)*betap(i)-z(272)*psip(i)-LLink2*
    qLink1p(i)-LLink2*z(372))) - 0.125*MBar2*(8*LE*z(471)* ...
phi_1p(i)*(z(147)*betap(i)+z(148)*psip(i))+4*z(25)*z(453)*theta_1p(i)*(z
    (218)*gammap(i)+z(219)*betap(i)+z(220)*psip(i))+8*z(25)*z(474)*theta_1p
    (i)*(BasePos(i)*gammap(i)-z(143)*psip(i)-z(144)*betap(i))+4*z(499)*(z
    (147)*betap(i)+z(148)*psip(i))*(z(218)*gammap(i)+z(219)*betap(i)+ ...
z(220)*psip(i))+8*LE*z(1)*phi_1p(i)*x1p(i)+8*LE*z(4)*phi_1p(i)*y1p(i)+8*z
    (53)*(z(147)*betap(i)+z(148)*psip(i))*y1p(i)+4*z(51)*(z(218)*gammap(i)+
    z(219)*betap(i)+z(220)*psip(i))*y1p(i)+8*z(50)*x1p(i)*(z(196)*gammap(i)
    +z(221)*betap(i)+z(222)*psip(i)+BasePosp(i))+8*LE*z(450)*phi_1p(i)*(
    ...
z(196)*gammap(i)+z(221)*betap(i)+z(222)*psip(i)+BasePosp(i))+8*z(43)*(
    BasePos(i)*gammap(i)-z(143)*psip(i)-z(144)*betap(i))*x1p(i)+8*z(48)*(
    BasePos(i)*gammap(i)-z(143)*psip(i)-z(144)*betap(i))*y1p(i)+8*z(134)*
    x1p(i)*(z(195)*betap(i)-z(194)*psip(i)-LBar3*qBar3p(i))+8*z(491)*y1p(i)
    *(z(195)* ...
```

```
betap(i)-z(194)*psip(i)-LBar3*qBar3p(i))+8*z(25)*z(468)*theta_1p(i)*(z(195)
    *betap(i)-z(194)*psip(i)-LBar3*qBar3p(i))+4*(z(218)*gammap(i)+z(219)*
    betap(i)+z(220)*psip(i))*(z(196)*gammap(i)+z(221)*betap(i)+z(222)*psip(
    i)+BasePosp(i))+8*z(507)*(z(147)*betap(i)+z(148)*psip(i))*(z(195)* ...
betap(i)-z(194)*psip(i)-LBar3*qBar3p(i))+4*z(119)*x1p(i)*(z(217)*betap(i)-z
    (216)*psip(i)-LBar2*qLink1p(i)-LBar2*z(372)-LBar2*z(373))+4*z(487)*y1p(
    i)*(z(217)*betap(i)-z(216)*psip(i)-LBar2*qLink1p(i)-LBar2*z(372)-LBar2*
    z(373))+4*z(25)*z(462)*theta_1p(i)*(z(217)*betap(i)- ...
z(216)*psip(i)-LBar2*qLink1p(i)-LBar2*z(372)-LBar2*z(373))+4*z(501)*(z(147)
    *betap(i)+z(148)*psip(i))*(z(217)*betap(i)-z(216)*psip(i)-LBar2*qLink1p
    (i)-LBar2*z(372)-LBar2*z(373))-4*LE^2*phi_1p(i)^2-4*z(25)^2*theta_1p(i)
    ^2-4*(z(147)*betap(i)+z(148)*psip(i))^2-8*z(25)* ...
z(473)*theta_1p(i)*(z(147)*betap(i)+z(148)*psip(i))-(z(218)*gammap(i)+z
    (219)*betap(i)+z(220)*psip(i))^2-4*(BasePos(i)*gammap(i)-z(143)*psip(i)
    -z(144)*betap(i))^2-4*LE*z(450)*phi_1p(i)*(z(218)*gammap(i)+z(219)*
    betap(i)+z(220)*psip(i))-8*LE*z(472)*phi_1p(i)*(BasePos(i)*gammap(i)-z
    (143)*psip(i)- ...
z(144)*betap(i))-4*x1p(i)^2-4*y1p(i)^2-8*z(5)*z(25)*theta_1p(i)*y1p(i)-8*z
    (47)*(z(147)*betap(i)+z(148)*psip(i))*x1p(i)-4*(z(196)*gammap(i)+z(221)
    *betap(i)+z(222)*psip(i)+BasePosp(i))^2-4*z(50)*(z(218)*gammap(i)+z
    (219)*betap(i)+z(220)*psip(i))*x1p(i)-8*z(51)*y1p(i)*(z(196)*gammap(i)+
     ...
z(221)*betap(i)+z(222)*psip(i)+BasePosp(i))-4*(z(195)*betap(i)-z(194)*psip(
    i)-LBar3*qBar3p(i))^2-8*z(25)*z(453)*theta_1p(i)*(z(196)*gammap(i)+z
    (221)*betap(i)+z(222)*psip(i)+BasePosp(i))-8*LE*z(466)*phi_1p(i)*(z
    (195)*betap(i)-z(194)*psip(i)-LBar3*qBar3p(i))-8*z(130)*(BasePos(i)*
    ...
```

```
gammap(i)-z(143)*psip(i)-z(144)*betap(i))*(z(195)*betap(i)-z(194)*psip(i)-
    LBar3*qBar3p(i))-(z(217)*betap(i)-z(216)*psip(i)-LBar2*qLink1p(i)-LBar2
    *z(372)-LBar2*z(373))^2-4*LE*z(460)*phi_1p(i)*(z(217)*betap(i)-z(216)*
    psip(i)-LBar2*qLink1p(i)-LBar2*z(372)-LBar2*z(373))-4* ...
z(480)*(z(195)*betap(i)-z(194)*psip(i)-LBar3*qBar3p(i))*(z(217)*betap(i)-z
    (216)*psip(i)-LBar2*qLink1p(i)-LBar2*z(372)-LBar2*z(373))-4*z(484)*(
    BasePos(i)*gammap(i)-z(143)*psip(i)-z(144)*betap(i))*(z(217)*betap(i)-z
    (216)*psip(i)-LBar2*qLink1p(i)-LBar2*z(372)-LBar2*z(373)));



% Additional outputs
z(19) = z(13)*z(15);
z(20) = z(13)*z(17);
P1_x(i) = x1(i) - LE*z(2);
P1_y(i) = y1(i) + LE*z(7);
P2_x(i) = x2(i) - LW*z(14);
P2_y(i) = y2(i) + LW*z(19);
EndEffector_x(i) = x1(i) + LLink1*z(86) + LLink2*z(99) + z(302)*z(47) - LE*
    z(2) - BasePos(i)*z(50);
EndEffector_y(i) = y1(i) + LE*z(7) + LLink1*z(303) + LLink2*z(307) +
    BasePos(i)*z(51) - z(302)*z(53);
EndEffector_z(i) = LLink1*z(304) + LLink2*z(308) + z(302)*z(54) + BasePos(i
    )*z(52) - LE*z(8);
%    EndEffector_x(i) = LLink1*sin(qLink1(i)) + LLink2*sin(qLink1(i)+qLink2
    (i));
%    EndEffector_y(i) = BasePos(i);
%    EndEffector_z(i) = LBase + LLink1*cos(qLink1(i)) + LLink2*cos(qLink1(i
    )+qLink2(i));
 end
```

```
    % Append to ySoln vector
    ySoln(:,28) = gammap;
    ySoln(:,29) = betap;
    ySoln(:,30) = qBar2p;
    ySoln(:,31) = qLink2p;
    ySoln(:,32) = theta_2p;
    ySoln(:,33) = theta_1pp;
    ySoln(:,34) = phi_1pp;
    ySoln(:,35) = phi_2pp;
    ySoln(:,36) = psipp;
    ySoln(:,37) = P1_x;
    ySoln(:,38) = P1_y;
    ySoln(:,39) = P2_x;
    ySoln(:,40) = P2_y;
    ySoln(:,41) = EndEffector_x;
    ySoln(:,42) = EndEffector_y;
    ySoln(:,43) = EndEffector_z;
    ySoln(:,44) = PotentialEnergy-min(PotentialEnergy);
    ySoln(:,45) = KineticEnergy;

end % craneWithRobot_SOLVE
```

REFERENCES

[1] A. Meghdari, F. F., "On the first-order decoupling of dynamical equations of motion for elastic multibody systems as applied to a two-link flexible manipulator," *Multibody System Dynamics*, 2001.

[2] Alex S. Miller, Padma Sarvepalli, W. S., "Dynamics and control of dual-hoist cranes moving triangular payloads," ASME Dynamic Systems adn Control Conference, 2014.

[3] Bearee, R., "New damped-jerk trajectory for vibration reduction," *Control Engineering Practice*, 2014.

[4] Daniel Maier, Armin Hornung, M. B., "Real-time navigation in 3d environments based on depth camera data," IEEE International Conference on Humanoid Robots, 2012.

[5] DiCicco, M. A., "Force control of heavy lift manipulators for high precision insertion tasks," Master's thesis, Massachusetts Intitute of Technology, 2005.

[6] Ditteon, R., *Modern Geometrical Optics*. John Wiley & Sons, Inc.

[7] Dr. Paul Mitiguy, "Motiongenesis."

[8] Ehsan Maleki, William Singhose, J. H., "Dynamic response of a dual-hoist crane," ASME Dynamic Systems and Control Conference, 2013.

[9] H. Gonzalez, B. Riveiro, E. V.-F. J. M.-S. P. A., "Metrological evaluation of microsoft kinect and asus xtion sensors," *Journal of the International Measurement Confederation*, 2013.

[10] J.C. Ower, J. V. D. V., "Classical control design for a flexible manipulator: Modeling and control system design," *Journal of Robotics and Automation*, 1987.

[11] Jianda Han, Zhiqiang Zhu, Y. H.-J. Q., "A novel input shaping method based on system output," *Journal of Vibration and Sound*, 2014.

[12] Jonathan Y. Smith, Kris Kozak, W. E. S., "Input shaping for a simple nonlinear system," IEEE American Control Conference, 2002.

[13] Karan, B., "Accuracy improvements of consumer-grade 3d sensors for robotic applications," IEEE Symposium on Intelligent Systems and Informatics, 2013.

[14] Karan, B., "Calibration of depth measurement model for kinect-type 3d vision sensors," WSCG Conference on Computer Graphics, Visualization and Computer Vision, 2013.

[15] Kuo-Shen Chen, Tian-Shiang Yang, J.-F. Y., "Residual vibration suppression for duffing nonlinear systems with electromagnetical actuation using nonlinear command shaping techniques," *Journal of Vibration and Acoustics*, 2006.

[16] Lee, H., "New dynamic modeling of flexible link robots," *Journal of Dynamic Systems, Measurement, and Control*, 2005.

[17] Maleki, E., *Control of Human-Operated Machinery with Flexible Dynamics*. PhD thesis, Georgia Intitute of Technology, 2013.

[18] Miller, A., "Dynamics and control of dual-hoist cranes moving distributed payloads," Master's thesis, Georgia Intitute of Technology, 2015.

[19] N. C. Singer, W. P. S., "Preshaping command inputs to reduce system vibration," *Journal of Dynamic Systems, Measurement, and Control*, 1990.

[20] Neil Singer, W. S. and Krikku, E., "An input shaping controller enabling cranes to move without sway," American Nuclear Society 7th Topical Meeting on Robotics and Remote Systems, 1997.

[21] Sabastian Zug, Felix Penzlin, A. D. T. T. N.-S. A., "Are laser scanners replaceable by kinect sensors in robotic applications?," IEEE International Symposium on Robotics and Sensors Environments, 2012.

[22] Singhose, W. E. and Singer, N. C., "Effects of input shaping on two-dimensional trajectory following," IEEE Transactions on Robotics and Automation, 1996.

[23] Smith, O., *Feedback Control Systems*. Mc-Graw Hill Books.

[24] Starr, G. P., "Swing-free transport of suspended objects with a path-controlled robot manipulator," *Journal of Dynamic Systems, Measurement, and Control*, 1985.

[25] Starr, G. P., "Swing-free transport of suspended objects with a path-controlled robot manipulator," December 2003.

[26] Ted Shaneyfelt, M. M. J. and Agaian, S., "A vision feedback robotic docking crane system with application to vanilla pollination," *International Journal of Automation and Control*, 2013.

[27] Tsai, R. Y., "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *Multibody System Dynamics*, 2001.

[28] V. O. G. Rosado, E. A. O. Y., "Dynamic modeling and simulation of a flexible robotic manipulator," *Robotica*, 1999.

[29] William Singhose, Dooroo Kim, M. K., "Input shaping control of double-pendulum bridge crane oscillations," *Journal of Dynamic Systems, Measurement, and Control*, 2008.

[30] William Singhose, W. S. and Singer, N., "Shaping inputs to reduce vibration: A vector diagram approach," IEEE International Conference on Robotics and Automation, 1990.

[31] Yingguang Chu, Filippo Sanfilippo, V. A. and Zhang, H., "An effective heave compensation and anti–sway control approach for offshore hydraulic crane operations," IEEE International Conference on Mechatronics and Automation, 2014.