

The Pennsylvania State University
The Graduate School
The Harold and Inge Marcus
Department of Industrial & Manufacturing Engineering

**THE MODELING OF A SIX DEGREE-OF-FREEDOM INDUSTRIAL ROBOT
FOR THE PURPOSE OF EFFICIENT PATH PLANNING**

A Thesis in
Industrial Engineering
by
Tyler J. Carter

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2009

The thesis of Tyler J. Carter was reviewed and approved* by the following:

Richard A. Wysk
Professor of Industrial Engineering and Leonhard Chair in Engineering
Thesis Advisor

Vittaladas V. Prabhu
Professor of Industrial Engineering

Richard J. Koubek
Professor of Industrial Engineering
Head of the Harold and Inge Marcus Department of
Industrial & Manufacturing Engineering

*Signatures are on file in the Graduate School

ABSTRACT

In order to compete globally, most manufacturers are looking for ways to reduce costs and, at the same time, are facing increasing demands to become environmentally responsible. One method to achieving both goals is to find ways to maximize process efficiency. In this research, robotic paths of various geometries are examined in order to determine the paths of lowest energy consumption, traveling distance, and process time. The paths are designed to meet the goals of a simple hypothetical scenario, requiring the robot to move from a specific start position to an end position while circumventing an obstacle. The various geometries include triangular, square, rectangular, semi-circular, and a fourth degree polynomial. Each path geometry is tested under two cases: one where the path moves strictly horizontal, while the other allows for vertical movement.

Several observations are discussed regarding the various paths and points of interest regarding variables such as torque or energy consumption. For the robot modeled in this work, it is shown that of all the paths examined, the horizontal triangular path geometry has the shortest traveling time, shortest traveling distance, and consumes the least amount of energy.

TABLE OF CONTENTS

LIST OF FIGURES	vi
LIST OF TABLES	viii
Chapter 1 INTRODUCTION.....	1
1.1 Motivation.....	1
1.2 Problem Statement.....	2
1.3 Overview of Thesis.....	4
Chapter 2 LITERATURE REVIEW	5
2.1 Introduction.....	5
2.2 Robot Modeling and Kinematics	5
2.3 Jacobian Matrices and Robot Dynamics.....	9
2.4 Path Planning.....	12
Chapter 3 METHODOLOGY	16
3.1 Introduction.....	16
3.2 Modeling the ABB IRB 140 Robot	17
3.2.1 Notation and Modified Denavit-Hartenburg Parameters	18
3.2.2 Assumptions	19
3.2.3 Forward Kinematics	20
3.2.4 Inverse Kinematics	21
3.2.5 Inertia Tensors	24
3.2.6 The Jacobian Matrix	25
3.2.7 Robot Dynamics	26
3.3 Path Planning	26
3.3.1 Various Path Geometries	27
3.4 Programming and Simulating the ABB IRB 140 Robot	30
3.4.1 Programming Using RAPID.....	31
3.4.2 Simulating Using RobotStudio.....	32
3.5 Calculation of Consumed Energy	33
Chapter 4 RESULTS.....	34
4.1 Introduction.....	34
4.2 Results.....	34
4.3 Case 1 Discussion	37
4.3.1 Triangular Path Geometry	37
4.3.2 Square Path Geometry	38
4.3.3 Rectangular Path Geometry.....	40

4.3.4 Semi-Circular Path Geometry	41
4.3.5 Polynomial Path Geometry.....	41
4.4 Case 2 Discussion	42
4.4.1 Triangular Path Geometry	43
4.4.2 Square Path Geometry	44
4.4.3 Rectangular Path Geometry.....	45
4.4.4 Semi-Circular Path Geometry	47
4.4.5 Polynomial Path Geometry.....	48
4.5 Travel Time Validation and Sensitivity Analysis.....	49
4.6 Recommendations.....	51
4.7 General Observations.....	51
 Chapter 5 CONCLUSIONS AND FUTURE RESEARCH.....	 53
5.1 Conclusions.....	53
5.2 Future Research	55
 REFERENCES	 56
 Appendix A Transformation Matrices	 58
 Appendix B Inverse Kinematics Derivations	 60
 Appendix D Jacobian Matrix Derivations	 65
 Appendix E Sample RAPID Code	 67
 Appendix F Sample MATLAB Code	 69
 Appendix G MATLAB Results	 74

LIST OF FIGURES

Figure 1-1 : The ABB IRB 140 robot with labeled points for the described scenario.....	3
Figure 2-1 : Compounded link frames, each described relative to the previous one with the use of a transformation matrix. This concept is used to relate the base frame to the tool frame. (Image Source: Craig 35).....	6
Figure 2-2 : Link frames and kinematic parameters of an arbitrary joint. (Image Source: Craig 68).....	7
Figure 3-1 : Kinematic parameters and frame assignments of the ABB IRB 140 manipulator.	18
Figure 3-2 : Projection of the wrist center onto the xy plane. (Image Source: Vicente 58).....	22
Figure 3-3 : Projection onto the plane formed by links 2 and 3. (Image Source: Vicente 59).....	23
Figure 3-4 : Triangular paths for Case 1 (horizontal path) and Case 2 (vertical path).	28
Figure 3-5 : Square paths for Case 1 (horizontal path) and Case 2 (vertical path).	28
Figure 3-6 : Rectangular paths for Case 1 (horizontal path) and Case 2 (vertical path).	29
Figure 3-7 : Semi-circular paths for Case 1 (horizontal path) and Case 2 (vertical path).	29
Figure 3-8 : Polynomial paths for Case 1 (horizontal path) and Case 2 (vertical path).	30
Figure 3-9 : MATLAB flowchart.	33
Figure 4-1 : Power requirements of the horizontal triangular path also follow a somewhat triangular curve.....	38
Figure 4-2 : Joint torque of the square path (Case 1) with spikes at changes in direction.	39
Figure 4-3 : Joint power requirements of the horizontal (Case 1) rectangular path with peaks of power when the robot uses Joint 2 to move away and towards the base.	40

Figure 4-4: Power requirements of the vertical (Case 2) triangular path with peaks at the initial and final points.	44
Figure 4-5: Joint torque of the vertical square path with peaks at the beginning and end of the path.....	45
Figure 4-6: Joint torque of the vertical (Case 2) rectangular path with spikes when the robot changes direction and larger peaks of torque during the first and last few points.	46
Figure 4-7: Joint position of the vertical (Case 2) semi-circular path, which is very smooth since there are no major direction changes like other paths examined. ...	47
Figure 4-8: Cumulative energy consumption of the vertical (Case 2) polynomial path, which is very typical for all of the vertical paths examined.	48
Figure 4-9: Energy consumption of the triangular (Case 1) path as travel time changes.	50

LIST OF TABLES

Table 3-1 : Modified Denavit-Hartenburg Parameters for the ABB IRB 140 manipulator.	19
Table 4-1 : Summary of travel time, travel distance, and energy consumed for each path.	35
Table 4-2 : Percent differences compared to the horizontal triangular path geometry.	36

Chapter 1

INTRODUCTION

1.1 Motivation

Since their initial debut, the use of industrial robots has continued to gain popularity. Robots are found in many fields, including medical, military, and space exploration, and have become commonplace in many large manufacturing facilities, such as automotive factories.

In the manufacturing industry alone, robots are capable of numerous tasks when equipped with the proper hardware, including activities such as welding, painting, assembling, palletizing, and material handling. When compared to humans, they are typically much more reliable, accurate, and capable of reaching further, moving faster, and carrying more weight—all for much longer periods of time. Industrial robots can be programmed to do tasks that humans do not typically wish to do, such as jobs that are: repetitive, known to be messy or hazardous to humans, or jobs that require such precision and stability that a human operator is not capable of completing.

In order to compete globally, most manufacturers are looking for ways to reduce costs and, at the same time, are facing increasing demands to become environmentally responsible. Industrial robots could be part of their solution, as robots are cost effective over the long term and can increase productivity while allowing human operators to focus on other crucial tasks. If these same industrial robots are analyzed and programmed

carefully, they could perform efficiently in such a way that minimizes production time, travel distance, and/or energy consumed. The analysis of robots and their planned paths could potentially help manufacturers produce at the lowest rate of energy consumption possible and, in turn, save energy costs. Performance at low energy consumption could also be useful in applications when there is a power supply with a finite energy capacity, such as a battery or gas generator. Analyzing paths based on travel distance and sudden direction changes can directly be related to movements that create wear on the motors. When motors need to be replaced, there are numerous costs introduced: spare parts costs, maintenance and repair personnel costs, and loss of sales due to production downtime. Likewise, process or travel time of a particular path can dictate the production rate. If this process slows the facility's production capabilities, there is money lost in sales, or could even mean that the operator supervising the process may need to be paid for longer hours or can not move to other production areas. Depending on the various costs and priorities of a particular manufacturer, one of the above process efficiency metrics may have more precedence over the others.

1.2 Problem Statement

Consider a manufacturing environment that is automated with an industrial robot such as a six-axis ABB IRB 140, shown in Figure 1-1 below. The articulated robotic arm must pick up a product dispensed at the point (400, -250, 200) mm. It must move the product to another point to be boxed at (400, 250, 200) mm. However, it must avoid a

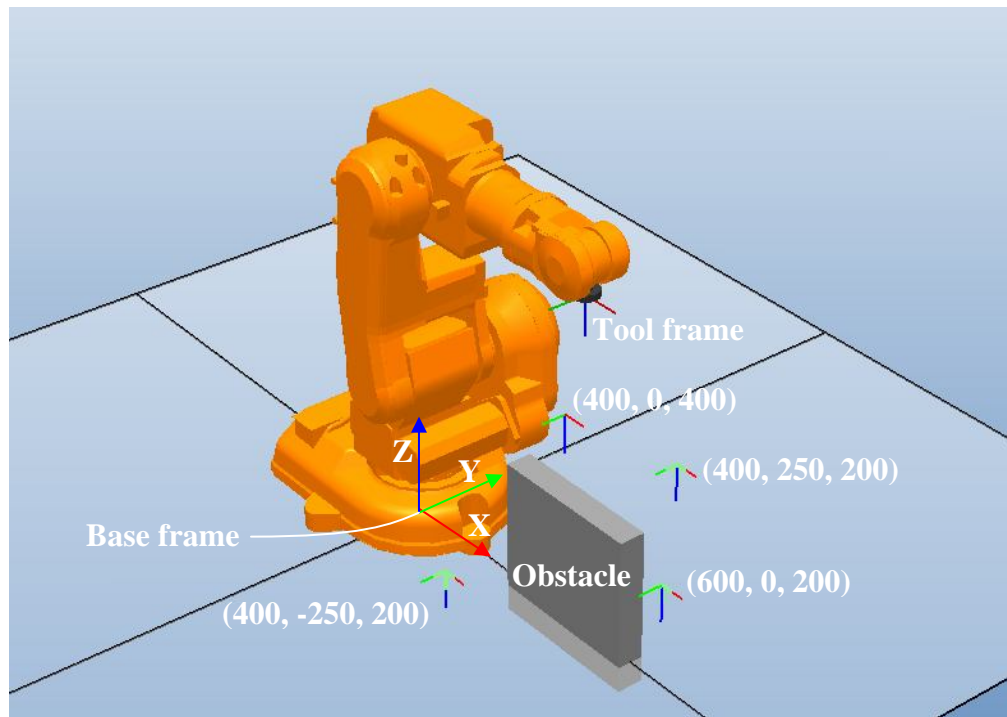


Figure 1-1: The ABB IRB 140 robot with labeled points for the described scenario.

safety barrier to prevent workers from reaching in too close to the product dispenser. The robot arm can avoid this obstacle in two different cases:

- 1) if it stays in the $z = 200$ mm plane and passes around point $(600, 0, 200)$ mm or
- 2) if it stays in the $x = 400$ mm plane and passes above point $(400, 0, 400)$ mm.

Effectively, Case 1 is a strictly horizontal movement, while Case 2 is a vertical movement. Manipulator orientation remains constant throughout the entire path and the robot follows the same path back to the product dispenser. This cycle of operations is repeated thousands of times a day. For safety and fatigue reasons, the robot will not operate at full speed, but at an engineer's suggested speed of "v1000", which limits the linear velocity of the robot manipulator to 1000 mm/s. The manufacturer is a competitive firm and is facing increasing pressure from both environmentalists and the finance department to increase the process efficiency. Although energy consumption is the main

path selection criteria, the management would like a path to be programmed that also does not take a long time to complete or put unnecessary wear on the motors by making the robot travel excessively far. In other words, the metrics for process efficiency will include process time, travel distance, and energy consumed.

1.3 Overview of Thesis

The remainder of the thesis is organized as follows. In Chapter 2, related research is discussed. General kinematics and dynamics are presented to establish a basis for what is often used in the robotics field. Various path planning strategies are presented, along with literature utilized to calculate energy consumption.

Chapter 3 is a direct application of much of the literature review to an ABB IRB 140 robot. The methodology is presented and several geometric paths are created, each with a case for horizontal movement and a case for vertical movement. Several types of computer software are utilized to aid in the implementation of methods introduced in the literature review.

Chapter 4 presents selected graphical results and compares the various paths. A sensitivity analysis is performed on the effect of the simulated travel time on the estimated energy results. Finally, Chapter 5 provides concluding comments, along with several suggestions for future research.

Chapter 2

LITERATURE REVIEW

2.1 Introduction

Much of the information in the beginning of this chapter, such as robot modeling, kinematics, and dynamics, is common knowledge in the robotics field. There are numerous textbooks on the topics, such as those written or edited by Manseur, Kurfess, and Craig, most of which have only slight differences and methodologies (Manseur; Kurfess; Craig). Therefore, this chapter will be primarily used to give a general overview of the terms and notation used throughout this paper. For a more detailed explanation, please consult the textbooks. The final sections in this chapter will summarize various works of literature with respect to path planning and energy consumption of robotic manipulators. In Chapter 3, the general overview and concepts will be directly applied to the specific scenario.

2.2 Robot Modeling and Kinematics

Robotic arms consist of links and joints. Each link can move with respect to the preceding link with the use of either rotational or translational joints. As the names imply, a rotational joint allows a link to rotate and a translational joint allows the link to perform a sliding motion. A human arm, for example, can be modeled with bones as the

links connected by numerous rotational joints, such as the elbow or shoulder. Some robots have joints that intersect at the same point, allowing a wrist-like capability (roll, pitch, and yaw). Furthermore, robotic arms often have an end-effector, such as a gripper, that is similar to two fingers that can pinch and pick objects up. Oftentimes, robotic arms with revolute joints are referred to as articulated arms or industrial robotic arms. The remainder of this paper will discuss only this type of robot.

In order to mathematically model a robot so that we can know the location and orientation of the tip of the arm compared to the base, or any other point, we must assign coordinate frames to the base and tip of the robot and at each joint. A frame's translation with respect to another frame i can be described by the 3×1 vector ${}^iP_{i+1}$. A frame's

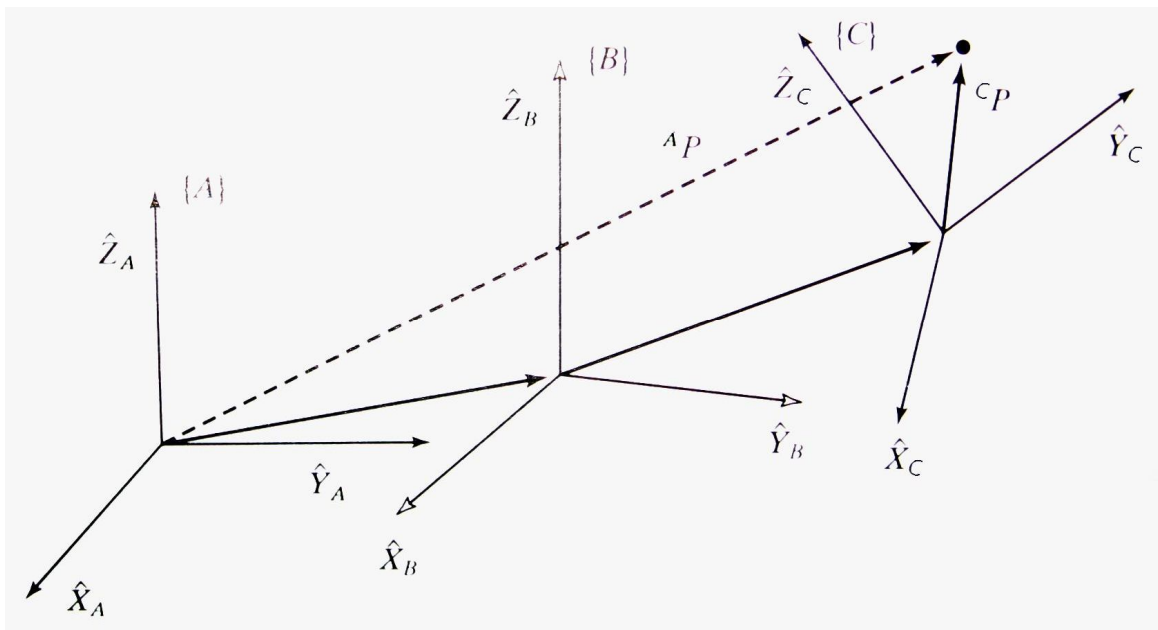


Figure 2-1: Compounded link frames, each described relative to the previous one with the use of a transformation matrix. This concept is used to relate the base frame to the tool frame. (Image Source: Craig 35)

rotation with respect to another frame i can be described by a rotation matrix ${}^iR_{i+1}$, which is a 3×3 matrix composed of dot products of unit vectors. Finally, to describe the translation *and* rotation of a frame $i+1$ with respect to frame i , a 4×4 homogeneous transformation matrix ${}^iT_{i+1}$ is used with the following structure:

$${}^iT_{i+1} = \left[\begin{array}{ccc|c} & & & \\ & & & \\ & & & \\ \hline 0 & 0 & 0 & 1 \end{array} \right]. \quad (2.1)$$

To address the translation and rotation portions of equation 2.1, some specific measurements must be included to describe how a robot's frames are assigned. One common method to assigning the frames (and therefore link descriptions) properly is by using the Denavit-Hartenberg (DH) notation, which is used by authors such as Manseur.

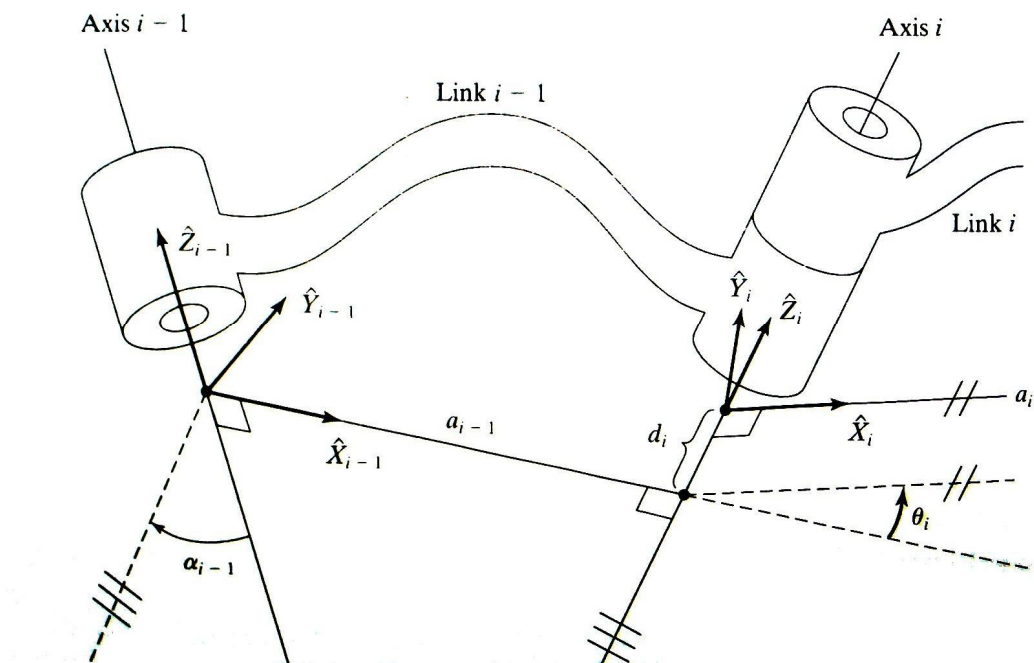


Figure 2-2: Link frames and kinematic parameters of an arbitrary joint. (Image Source: Craig 68)

However, Craig uses a modified version of DH notation which will be used throughout this paper. As shown in Figure 2-1, the DH parameters as modified by Craig are:

$$\begin{aligned}
 a_i &= \text{link length, the distance from } \hat{Z}_i \text{ to } \hat{Z}_{i+1} \text{ measured along } \hat{X}_i; \\
 \alpha_i &= \text{link twist, the angle from } \hat{Z}_i \text{ to } \hat{Z}_{i+1} \text{ measured about } \hat{X}_i; \\
 d_i &= \text{link offset, the distance from } \hat{X}_{i-1} \text{ to } \hat{X}_i \text{ measured along } \hat{Z}_i; \text{ and} \\
 \theta_i &= \text{joint angle, the angle from } \hat{X}_{i-1} \text{ to } \hat{X}_i \text{ measured about } \hat{Z}_i. \quad (64-69)
 \end{aligned}$$

The general form of the transformation matrix using the modified DH parameters is as follows:

$${}^{i-1}T_i = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.2)$$

where $c\theta_i = \cos(\theta_i)$ and $s\theta_i = \sin(\theta_i)$ (Craig 75). A transformation matrix is calculated for $i=1$ to N , where N is the number of joints in the robotic arm. Additionally, a transformation matrix from the base frame to the tip, or 0T_N , can be calculated by multiplying all the transformation matrices together, in order:

$${}^0T_N = {}^0T_1 {}^1T_2 \dots {}^{N-1}T_N \quad (2.3)$$

The process known as forward kinematics “consists of computing the position and orientation of a robot end-effector when all the joint variables are known” (Manseur 118).

Conversely, calculating the joint variables when the position and orientation are given is known as inverse kinematics. This problem is much more complex than the forward kinematics due to the fact that there often exists far more than one set of joint angles that satisfy the position and orientation. Going back to the human arm example, if

a person was holding an object in front of themselves, they could maintain the same position of their fingers but could move their shoulder and elbow to allow a position where the elbow is pointing up or a position where the elbow is pointing down.

Regardless, some robots (like the one used in this paper) are able to be solved in closed-form, meaning equations can be solved analytically to describe the robot. Closed-form solutions can be solved for with an algebraic or geometric approach.

2.3 Jacobian Matrices and Robot Dynamics

In the robotics field, the Jacobian is a matrix of partial derivatives that “relate joint velocities to Cartesian velocities of the tip of the arm” (Craig 150). This matrix is useful because it allows us to easily calculate the velocity of the tip while only measuring the joint velocities with the use of encoders, or sensors, on the joint motors. One of the methods for deriving the Jacobian is by “directly differentiating the kinematic equations of the mechanism,” which is straightforward for linear velocity (Craig 150).

Before the robot dynamics—or equations of motion—are addressed, the mass distribution of each link must be represented to account for inertial properties of the robot. As Hollerbach recognizes, oftentimes, the inertial properties are not known, even to the manufacturer. There are numerous ways to measure or estimate what is known as the inertia tensor, many of which are not feasible for this paper. One such labor-intensive method is to disassemble the robot, weigh the links, counterbalance for the center of mass, and then swing for moments of inertia. Another somewhat intensive method is analyzing 3D CAD models using material properties and geometries. This second

method will be used in this paper, and will be discussed in Chapter 3. Finally, one could estimate the inertial properties using torque sensors at each joint and using an algorithm to solve for the parameters. It is important to note that the inertia tensors, denoted by ${}^{C,i+1}I_{i+1}$, for each link $i+1$ must be calculated with respect to the center of mass of that link. (Hollerbach 410-412)

Having both the Jacobian and inertial matrices solved for, there are several methods available to solve a robot's dynamic equations regarding motion, forces, and torques. As Nagurka notes, these methods "include the Newton-Euler (N-E) method, the Lagrange-Euler(L-E) method, Kane's method, bond graph modeling, as well as recursive formulations for both Newton-Euler and Lagrange-Euler methods" (4-2). The recursive, or iterative, Newton-Euler method has the advantage that it "can be applied to the robot links from one end of the arm to the other providing an efficient means to determine the necessary forces and torques..." (Nagurka 4-2). Therefore, this method will be utilized in this paper. There are two major iterations: an outward iteration from the first link to the last link to determine the velocities and accelerations and a second inward iteration from the last link to the first link to compute the forces and torques acting on the center of each link. The equations for each iteration (Craig 176) are shown below:

Outward iterations: $i: 0 \rightarrow N-1$

$${}^{i+1}\omega_{i+1} = {}^iR^{i+1} \omega_i + \dot{\theta}_{i+1} {}^{i+1}\hat{Z}_{i+1}, \quad (2.4)$$

$${}^{i+1}\dot{\omega}_{i+1} = {}^iR^{i+1} \dot{\omega}_i + {}^iR^{i+1} \omega_i \times \dot{\theta}_{i+1} {}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1} {}^{i+1}\hat{Z}_{i+1}, \quad (2.5)$$

$${}^{i+1}\dot{v}_{i+1} = {}^iR^{i+1} (\dot{\omega}_i + {}^iP_{i+1} \omega_i \times ({}^i\omega_i \times {}^iP_{i+1}) + {}^i\dot{v}_i), \quad (2.6)$$

$${}^{i+1}\dot{v}_{C,i+1} = {}^iR^{i+1} \dot{\omega}_i \times {}^{i+1}P_{C,i+1} + {}^{i+1}\omega_{i+1} \times ({}^{i+1}\omega_{i+1} \times {}^{i+1}P_{C,i+1}) + {}^{i+1}\dot{v}_{i+1}, \quad (2.7)$$

$${}^{i+1}F_{i+1} = m_{i+1} {}^{i+1}\dot{v}_{C,i+1}, \quad (2.8)$$

$${}^{i+1}N_{i+1} = {}^{C,i+1}I_{i+1} {}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{C,i+1}I_{i+1} {}^{i+1}\omega_{i+1}, \quad (2.9)$$

Inward iterations: $i: N \rightarrow 1$

$${}^i f_i = {}^i R^{i+1} f_{i+1} + {}^i F_i, \quad (2.10)$$

$${}^i n_i = {}^i N_i + {}^i R^{i+1} n_{i+1} + {}^i P_{C,i} \times {}^i F_i + {}^i P_{i+1} \times {}^i R^{i+1} f_{i+1}, \quad (2.11)$$

$$\tau_i = {}^i n_i^T {}^i \hat{Z}_i. \quad (2.12)$$

The following notation is used:

$\dot{\theta}_i$: joint angle velocity,

$\ddot{\theta}_i$: joint angle acceleration,

m_i : mass of link i ,

${}^{i+1} \omega_{i+1}$: angular velocity of link $i+1$ with respect to frame $i+1$,

${}^{i+1} \dot{\omega}_{i+1}$: angular acceleration of link $i+1$ with respect to frame $i+1$,

${}^{i+1} \dot{v}_{i+1}$: linear acceleration of link $i+1$ with respect to frame $i+1$,

${}^{i+1} \dot{v}_{C,i+1}$: linear acceleration of the center of mass of link $i+1$,

${}^{i+1} F_{i+1}$: force acting at the center of mass of link $i+1$,

${}^{i+1} N_{i+1}$: torque acting at the center of mass of link $i+1$,

${}^i f_i$: force exerted on link i by link $i-1$,

${}^i n_i$: torque exerted on link i by link $i-1$,

τ_i : joint torque on link i .

To include gravity in the calculations, the magnitude of gravity can be placed in the initial linear acceleration pointing in the opposite direction:

$${}^0 \dot{v}_0 = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}. \quad (2.13)$$

Craig explains: “This fictitious upward acceleration causes exactly the same effect on the links as gravity would. So, with no extra computational expense, the gravity is calculated” (176). The gravitational constant g is equal to 9.8 m/s^2 .

After completing the Newton-Euler iterations, the joint torque τ_i will have been calculated for all joints. The dynamic equation of motion can then be written in the form:

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) \quad (2.14)$$

where $M(\Theta)$ is the $N \times N$ mass matrix of the manipulator, $V(\Theta, \dot{\Theta})$ is an $N \times 1$ vector of centrifugal and Coriolis terms, and $G(\Theta)$ is an $N \times 1$ vector of gravity terms. Each element is a complex function of Θ , the position of all the joints of the manipulator, which will be discussed in further detail in the following section.

2.4 Path Planning

Path planning, also referred to as trajectory generation, is the act of describing the desired motion of a manipulator in multidimensional space over a period of time. Typically, a user will simply define a beginning point, $P_1 = (x_1, y_1, z_1)$, and an ending point, $P_2 = (x_2, y_2, z_2)$, in Cartesian space. Then, a controller or computer program will determine the set of consecutive via points to connect the two. With the knowledge of the next Cartesian via point χ , the first and second time derivatives will respectively result in Cartesian velocity, $\dot{\chi}$, and acceleration, $\ddot{\chi}$, of the tip. Of course, the via points must then be converted to the joint space positions, Θ , with the use of inverse kinematics. Joint angle velocity, $\dot{\Theta}$, and joint angle acceleration, $\ddot{\Theta}$, can be calculated using the following expressions:

$$\dot{\Theta} = J^{-1} \dot{\chi} \quad (2.15)$$

$$\ddot{\Theta} = J^{-1} \ddot{\chi} - J^{-1} \dot{J} J^{-1} \dot{\chi} \quad (2.16)$$

where J^{-1} is the inverse of the Jacobian matrix and \dot{J} is the time derivative of the Jacobian (Diken 789). Please notice that since Θ , $\dot{\Theta}$, and $\ddot{\Theta}$ are now known, the equation (2.14) for joint torque can now be utilized.

For decades, researchers have studied various methods for calculating the via points, including using cubic polynomials or splines, high order polynomials, linear functions with parabolic blends, and other parametric equations (Craig 203-219). As Craig explains, there are multiple problems that need to be addressed when generating paths, including intermediate points that are unreachable due to the physical constraints of the robot or due to paths that require a joint to flip when it reaches its joint limits, resulting in an impossibly high joint rate (220-221).

Since there is infinite number of paths from one point to another, further research has been done to find optimal paths composed of via points according to a specific objective. For example, much research has been done in minimizing the total travel distance or minimizing the total travel time of a path. In one of the frequently cited publications written by Bobrow, a good argument is presented on why minimum-distance (or arc length) is not always the best criterion:

In many cases this [minimum-distance] solution is not practical, since minimum-distance paths are straight lines with sharp corners near the obstacle intrusions.

Robot motion along such paths would require the velocity of the arm tip to stop or slow down considerably at each corner. (443)

In addition, most practical applications in the manufacturing industry include minimizing the time required for a product to be manufactured or handled to maximize productivity.

Hence, Bobrow and many others explore various approaches to the minimum-time criterion problem in their research. As summarized by Field and Stepanenko, the “problem is essentially a one dimensional optimization which seeks to maximize the speed along the path subject to manipulator constraints which may include joint velocity, joint acceleration, and joint torque” (2755). Some researchers, such as Constantinescu and Croft, even use other constraints such as joint jerk, which is found by taking the time derivative of acceleration.

In the words of Field and Stepanenko, “The number of publications dealing with minimum energy performance criteria is significantly less than that for the minimum time problem” (2755). Furthermore, Field and Stepanenko note that many authors, including themselves, Vukobratovic and Kircanski, Shin and McKay, and Singh and Leu, all use dynamic programming approaches to solve the minimum-energy problem (2755). Regardless of the optimization criterion, in many of the above cases, authors have expressed concern over the computational time required to solve optimization problems of even simple robots of one or two joints.

Instead of solving a complex and computationally intense optimization problem, one approach taken by Diken is to simulate many paths and then analyze and compare the energy consumed. Diken uses sinusoidal paths with varying amplitudes to describe the manipulator path over a simple harmonic time function, where a point χ between the first and last point (P_1 and P_2) can be expressed with the following formula:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + \frac{1}{2} \left(1 - \cos\left(\pi \frac{t}{t_f}\right) \right) \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{bmatrix}. \quad (\text{Diken 789}) \quad (2.17)$$

Since the paths are described as functions of time, time t changes from 0 to t_f , with T number of finite time intervals according to:

$$t_{i+1} = t_i + \frac{t_f}{T} \quad \text{for } i=0 \text{ to } T. \quad (2.18)$$

The power requirement for a given joint i during a specific time interval is given by:

$$W_i = |\tau_i \dot{\theta}_i| \quad \text{for all } i = 1 \text{ to } N \text{ joints for each time interval.} \quad (2.19)$$

The total power requirements for the manipulator at a given time interval is calculated as:

$$W(t) = \sum_{i=1}^N W_i \quad \text{for all } t = 0 \text{ to } t_f. \quad (2.20)$$

Finally, the energy consumption for the path is the integral of the total power over all time intervals:

$$E = \int_0^{t_f} W(t) dt. \quad (\text{Diken 790}) \quad (2.21)$$

In words, the total energy consumed is the time integral of the sum of the absolute value of the control torques.

Chapter 3

METHODOLOGY

3.1 Introduction

A large portion of this chapter is mainly devoted to a direct application of the standard robot modeling methodologies as described in Chapter 2 to an ABB IRB 140, a conventional articulated arm robot. It is necessary to derive the various kinematic and dynamic equations in order to mathematically describe this particular robot before any calculations can be performed.

Since many of the calculations can become quite complex very quickly, a key assumption is made in Section 3.2.2 regarding which joints will be analyzed in this paper. This assumption greatly simplifies the calculations in the subsequent Sections 3.2.3 through 3.2.7, involving topics such as forward and inverse kinematics, the Jacobian matrix, and the iterative Newton-Euler equations for finding joint torque.

Section 3.3 greatly defines the problem by planning paths of five various geometries, which will each have two cases: a horizontal path and a vertical path. These paths consist of several points, which will be programmed into the robot controller using ABB's robotic language called RAPID in Section 3.4.1. Using ABB's RobotStudio as described in Section 3.4.2, the RAPID program can be simulated to find the total process time and verify joint angles.

Finally, Section 3.5 summarizes the MATLAB program written which will use the equations of the robot derived earlier in the chapter, as well as the simulation results, to apply Diken's method of calculating the energy consumption under ideal conditions.

3.2 Modeling the ABB IRB 140 Robot

The ABB IRB 140 is a 6-axis industrial robot that is intended to be highly adaptable to many environments and applications. As noted in the product specification document, there are many optional features, such as Foundry protection for harsh, high temperature environments, Clean Room classification for a high degree of corrosion protection and the ability to wash, and many other features for applications such as gluing and arc welding. The IRB 140 is the most compact robot manufactured by ABB and has a handling capacity of 5 kg.

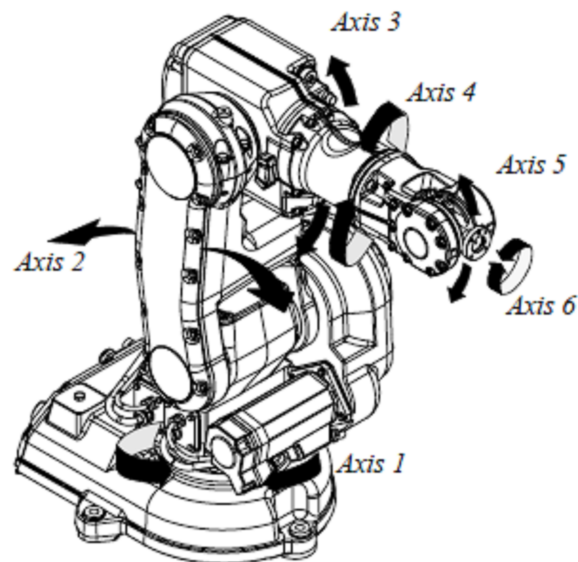


Figure 3-1: The IRB 140 has 6 axes. (Image Source: IRB 140 Product Specifications)

3.2.1 Notation and Modified Denavit-Hartenberg Parameters

As mentioned in Chapter 2, a common method to describe the frame location and the robot joints and links is to use Denavit-Hartenberg (DH) parameters. The modified version as described by Craig will be used in this paper. Figure 3-1 shows the frame assignment locations and robot dimensions. Table 3-1 displays the four parameters for each link: a_i , α_i , d_i , and θ_i .

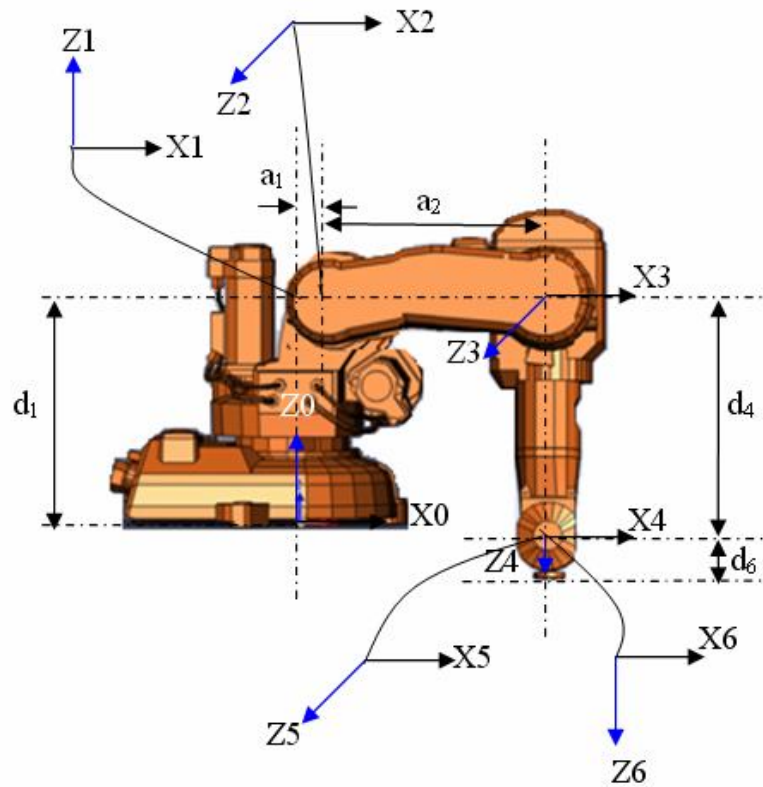


Figure 3-1: Kinematic parameters and frame assignments of the ABB IRB 140 manipulator.

Table 3-1: Modified Denavit-Hartenburg Parameters for the ABB IRB 140 manipulator.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	d_1	θ_1
2	90	a_1	0	θ_2
3	0	a_2	0	θ_3
4	90	0	d_4	θ_4
5	-90	0	0	θ_5
6	90	0	0	θ_6

$$d_1 = 352 \text{ mm}; a_1 = 70 \text{ mm}; a_2 = 360 \text{ mm}; d_4 = 380 \text{ mm}; d_6 = 65 \text{ mm}$$

3.2.2 Assumptions

As shown in Figure 3-1, joints 4, 5, and 6 all intersect at one point. This point will be referred to as the wrist center, since it has the ability to orient the manipulator tip in a wrist-like fashion with roll, pitch and yaw. It should be pointed out that the position of the wrist center (p_{xw}, p_{yw}, p_{zw}) can be calculated having only the link lengths and the joint angles of links 1, 2 and 3. The position of the tip of the manipulator (p_x, p_y, p_z) can be calculated very simply, and will be discussed in Section 3.2.3. Using the wrist center simplifies the forward and inverse kinematics (Sections 3.2.3 and 3.2.4), but more importantly, greatly reduces the calculations needed for the robot dynamics, especially the iterative process required for joint torque (Section 3.2.6) Joint torque is then used to calculate the energy consumption (Section 3.5). Therefore, this paper will only analyze the first three links of the robot. This simplification is justified by there not being much moving mass past the wrist center compared to the first three links, as well as the fact that

the remaining mass past the wrist center will be included in the link 3 center of mass and inertial calculations (Section 3.2.5).

3.2.3 Forward Kinematics

Forward kinematics is used to find the position of the manipulator given the joint positions, which is a relatively simple problem once the DH parameters are known.

Using Equation (2.2) for each row of Table 3-1, one can find the multiple transformation matrices shown in Appendix A. Next, Equation (2.3) can be used to multiply all of the matrices in order to calculate 0_6T , or the transformation matrix that describes the translation and rotation from the base of the robot to the wrist center:

$${}^0_6T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_{xw} \\ r_{21} & r_{22} & r_{23} & p_{yw} \\ r_{31} & r_{32} & r_{33} & p_{zw} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

$$\begin{aligned} r_{11} &= c_1[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6] + s_1(s_4c_5c_6 + c_4s_6) \\ r_{12} &= c_1[-c_{23}c_4c_5c_6 + s_{23}s_5s_6] - s_1s_4c_5s_6 \\ r_{13} &= c_1[c_{23}c_4s_5 + s_{23}c_5] + s_1s_4s_5 \\ r_{21} &= s_1[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6] - c_1(s_4c_5c_6 + c_4s_6) \\ r_{22} &= s_1[c_{23}c_4c_5c_6 + s_{23}s_5s_6] + c_1s_4c_5s_6 \\ r_{23} &= s_1[c_{23}c_4s_5 + s_{23}c_5] - c_1s_4s_5 \\ r_{31} &= s_{23}(c_4c_5c_6 - s_4s_6) + c_{23}s_5c_6 \\ r_{32} &= -s_{23}c_4c_5c_6 - c_{23}s_5s_6 \\ r_{33} &= s_{23}c_4s_5 - c_{23}c_5 \\ p_{xw} &= c_1[s_{23}d_4 + c_2a_2 + a_1] \\ p_{yw} &= s_1[s_{23}d_4 + c_2a_2 + a_1] \\ p_{zw} &= -c_{23}d_4 + s_2a_2 + d_1. \end{aligned} \tag{3.1}$$

To clarify notation, $c_1 = \cos(\theta_1)$ and $s_{23} = \sin(\theta_2 + \theta_3)$, etc. Please notice once again, that the equation describing the position of the wrist center (p_{xw}, p_{yw}, p_{zw}) only includes joint positions from joints 1, 2, and 3. If one wished to determine the absolute tip of the manipulator, it is simply a translation of length d_6 along the z axis of frame 6 with respect to the base frame 0:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} p_{xw} + d_6 r_{13} \\ p_{yw} + d_6 r_{23} \\ p_{zw} + d_6 r_{33} \end{bmatrix}. \quad (3.2)$$

3.2.4 Inverse Kinematics

Inverse kinematics is the process of calculating the joint positions, given the manipulator position. Unlike forward kinematics, inverse kinematics is much more challenging due to the fact that multiple solutions may exist. If all six joints were considered, up to 16 solutions could exist for any given position. For robots with 6 degrees of freedom with a wrist, it is common to break the inverse kinematics problem into two. The first part is a geometrical solution to find the joint angles corresponding to the position of the wrist center, while the second is an analytical solution to find the angles corresponding to the wrist orientation. A geometrical solution for the first three angles of the IRB 140 as derived by Vicente (57-60) will be summarized here, while the remaining analytical solution can be found in Appendix B since it will not be needed in this paper.

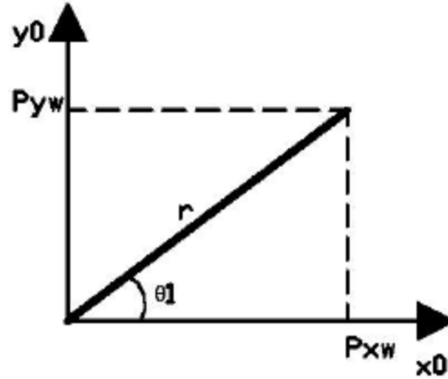


Figure 3-2: Projection of the wrist center onto the xy plane. (Image Source: Vicente 58)

To find the first joint angle (θ_1), we can examine a top down look at the robot in Figure 3-2. As the first joint angle changes, the whole manipulator moves and changes the x and y positions of the wrist. Therefore, using the arctangent function, we can derive two solutions for θ_1 (one of the solutions being when both wrist positions are negative):

$$\theta_1 = \tan^{-1}\left(\frac{p_{yw}}{p_{xw}}\right) = a \tan 2(p_{yw}, p_{xw}) \quad (3.3)$$

$$\text{OR } \theta_1 = \pi + \tan^{-1}\left(\frac{p_{yw}}{p_{xw}}\right) = a \tan 2(-p_{yw}, -p_{xw}). \quad (3.4)$$

To find the second and third joint angles (θ_2 and θ_3), we consider the plane formed by links 2 and 3 (Figure 3-3). Using the Law of Cosines while taking into account θ_1 :

$$\cos\theta_3 = \frac{(p_{xw} + a_1 \cos(\theta_1))^2 + (p_{yw} + a_1 \sin(\theta_1))^2 + (p_{zw} - d_1)^2 - a_2^2 - d_4^2}{2a_2d_4} = D. \quad (3.5)$$

We could solve for θ_3 with $\theta_3 = \cos^{-1}(D)$, however, to account for the two “elbow-up and elbow-down” solutions:

$$\theta_3 = \tan^{-1}\left(\frac{\pm\sqrt{1-D^2}}{D}\right) = a \tan 2(\pm\sqrt{1-D^2}, D) \quad (3.6)$$

Similarly,

$$\theta_2 = a \tan 2\left(p_{zw} - d_1, \sqrt{(p_{xw} + a_1 \cos(\theta_1))^2 + (p_{yw} + a_1 \sin(\theta_1))^2}\right) - a \tan 2(d_4 \sin(\theta_3), a_2 + d_4 \cos(\theta_3)). \quad (3.7)$$

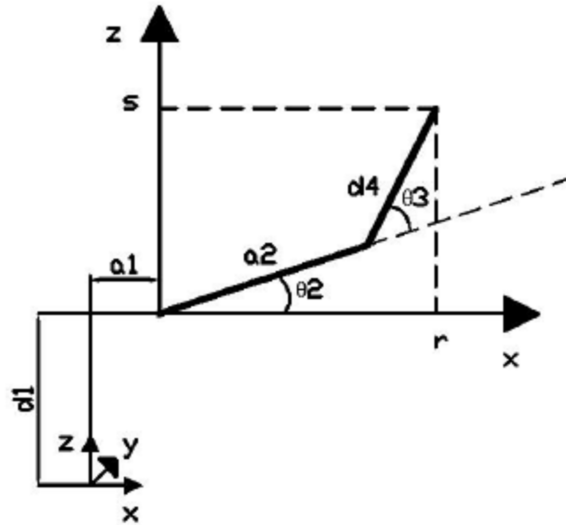


Figure 3-3: Projection onto the plane formed by links 2 and 3. (Image Source: Vicente 59)

The remaining joint angles can be solved analytically using closed-form equations which are derived in Appendix B. Since wrist position is determined by the first three joint angles, the remaining joint angles will not be used in this paper.

3.2.5 Inertia Tensors

As discussed previously, the mass distribution is needed to calculate the robot equations of motion. The method used to obtain the inertial data for this paper is via 3D CAD models which provide an estimation using factors such as geometry. The 3D modeling software used to analyze CAD models of the links will be SolidWorks. The CAD files of the ABB IRB 140 can be downloaded from the manufacturer's website. Once the assembly of all the IRB 140 links is loaded, coordinate frames are added which correspond to the orientation and location of the frames shown in Figure 3-1. Next, the particular link of interest is selected (highlighted) and the "Mass Properties" function of SolidWorks is run. An output window, as shown in the screenshots in Appendix C, displays the location of the center of mass with respect to the frame requested and the inertia tensor matrices ${}^{C,i+1}I_{i+1}$ for each link $i+1$ with respect to the center of mass of that link.

Due to the assumption that this paper will only analyze the first three links, the third link is unique in that it accounts for the remaining mass past the wrist center. In other words, links 3, 4, 5, and 6 are all selected and combined into link 3. They are all considered to move as one link while θ_3 changes. It should be stressed that the calculation of the inertia tensor using CAD models is an estimation.

3.2.6 The Jacobian Matrix

The Jacobian matrix is needed in order to relate joint velocities and Cartesian velocities. In Section 3.2.7, we will discuss how Cartesian points are selected on a path, which must be related to joint velocities so the joint torques and other values can be calculated. One method to derive the Jacobian matrix is to directly find the partial derivatives with respect to time of equations that describe the Cartesian position as functions of joint angles (Equations 3.1):

$$\frac{\delta p_{xw}}{\delta t} = \begin{bmatrix} -s_1(s_{23}d_4 + c_2a_2 + a_1) & c_1(c_{23}d_4 - s_2a_2) & c_1c_{23}d_4 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \quad (3.8)$$

$$\frac{\delta p_{yw}}{\delta t} = \begin{bmatrix} c_1(s_{23}d_4 + c_2a_2 + a_1) & s_1(c_{23}d_4 - s_2a_2) & s_1c_{23}d_4 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \quad (3.9)$$

$$\frac{\delta p_{zw}}{\delta t} = \begin{bmatrix} 0 & s_{23}d_4 + c_2a_2 & s_{23}d_4 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \quad (3.10)$$

Rewriting these equations into matrix form results in the Jacobian for linear velocity:

$$J_{3 \times 3} = \begin{bmatrix} -s_1(s_{23}d_4 + c_2a_2 + a_1) & c_1(c_{23}d_4 - s_2a_2) & c_1c_{23}d_4 \\ c_1(s_{23}d_4 + c_2a_2 + a_1) & s_1(c_{23}d_4 - s_2a_2) & s_1c_{23}d_4 \\ 0 & s_{23}d_4 + c_2a_2 & s_{23}d_4 \end{bmatrix}. \quad (3.11)$$

In a similar fashion, by completing another set of partial derivatives one can determine the derivative of J , written as \dot{J} .

3.2.7 Robot Dynamics

Using the iterative Newton-Euler method (Equations 2.4-2.12) for robots with many degrees of freedom becomes complex very quickly. The resulting closed-form equations for torque have been derived using MATLAB, but can literally be pages long, and for this reason they are not included. Instead of expressing the torque equations in closed-form, MATLAB code (see Appendix F) has been written to solve for torque numerically and is much more computationally efficient. The iterative Newton-Euler method must be performed numerous times per path, as it must be performed at each time interval. This loop is evident in a flowchart of MATLAB code shown in Section 3.5.

3.3 Path Planning

Path planning is a critical step in this paper since it defines which points the robot manipulator will pass through when moving between the start and end points. Often times this process will be calculated automatically based on some minimization criteria while using inverse kinematics and the Jacobian matrix to convert Cartesian path points to joint angles. The program must make sure that all points on the path are reachable and do not require unobtainable velocities or accelerations. For each point it must also have a method to choose one of the many inverse kinematic solutions. In this case of this paper, we would like to force the robot to move along certain paths and, therefore, path planning will be performed manually and each point will be inspected to ensure that all of the above mentioned is considered.

3.3.1 Various Path Geometries

Five path geometries are examined in this paper: triangular, square, rectangular, semi-circular, and a 4th degree polynomial. For Case 2, the same path geometries are explored but will be vertical instead of horizontal. The path distances, which are simply calculated using arc length formulas or basic perimeter methods, are the same for both cases of each geometry. The start and stop points also coincide. In the semi-circular and polynomial geometries, the path is defined by a function of the y position. The semi-circle equation is derived from the equation of a circle with radius r and centered at (h, k) ,

$$(x - h)^2 + (y - k)^2 = r^2, \quad (3.12)$$

which can be rewritten to be

$$x = \pm\sqrt{r^2 - (y - k)^2} + h. \quad (3.13)$$

Using either the + or – only results in a semi-circle. Likewise, the polynomial of four degrees can be written in a standard form such as:

$$x = ay^4 + by^3 + cy^2 + dy + e, \quad (3.14)$$

where a , b , c , d , and e are constants that can be solved for with 5 known coordinates.

In the remaining geometries, you may notice that the path could be defined in far fewer points since they are composed of linear movements. The reason for the extra points is to allow equal time intervals so that at each point, the position of the joint angles can easily be calculated, allowing the joint velocity and subsequent calculations to be performed for each interval. All five path geometries, as well as the two cases for each, are shown in Figure 3-4 through Figure 3-8 below.

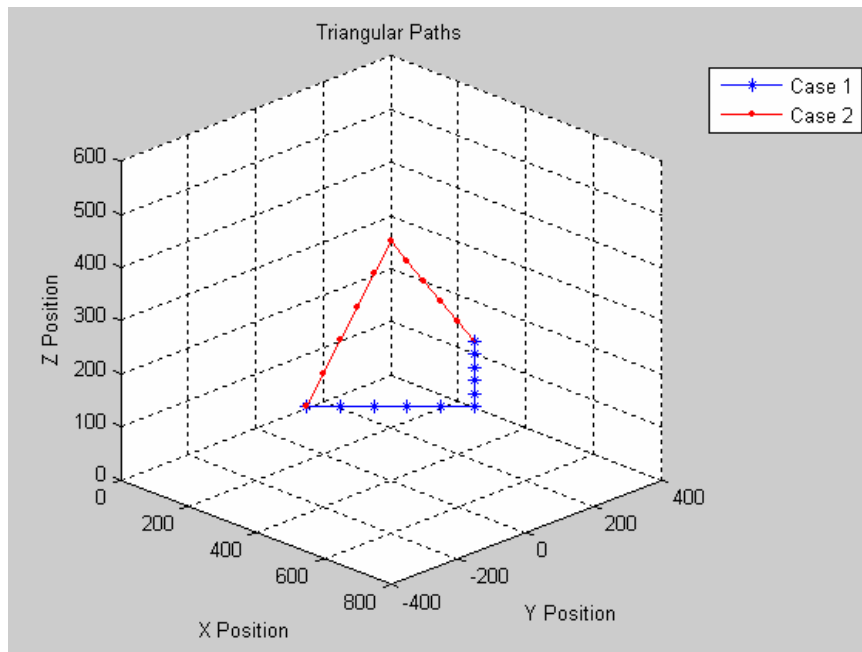


Figure 3-4: Triangular paths for Case 1 (horizontal path) and Case 2 (vertical path).

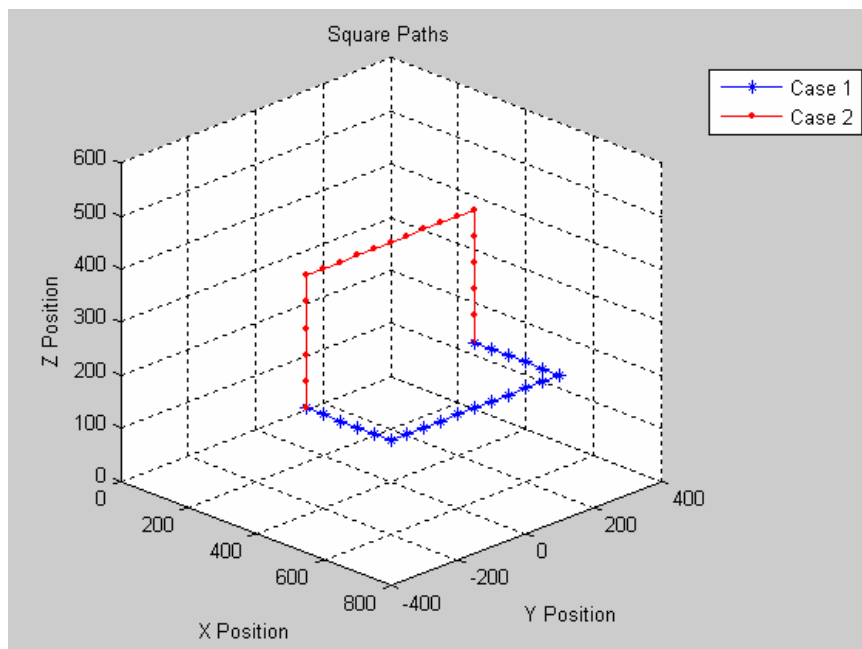


Figure 3-5: Square paths for Case 1 (horizontal path) and Case 2 (vertical path).

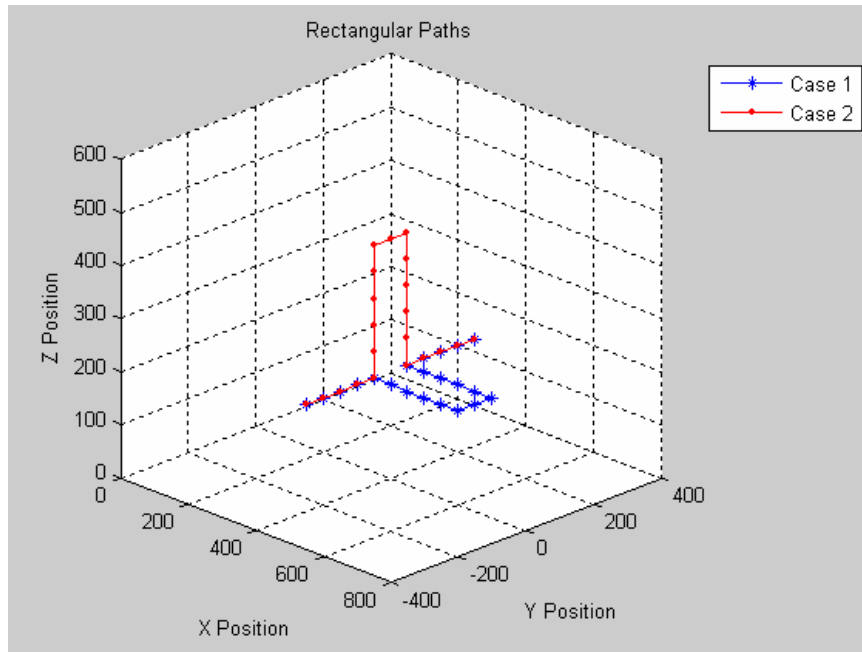


Figure 3-6: Rectangular paths for Case 1 (horizontal path) and Case 2 (vertical path).

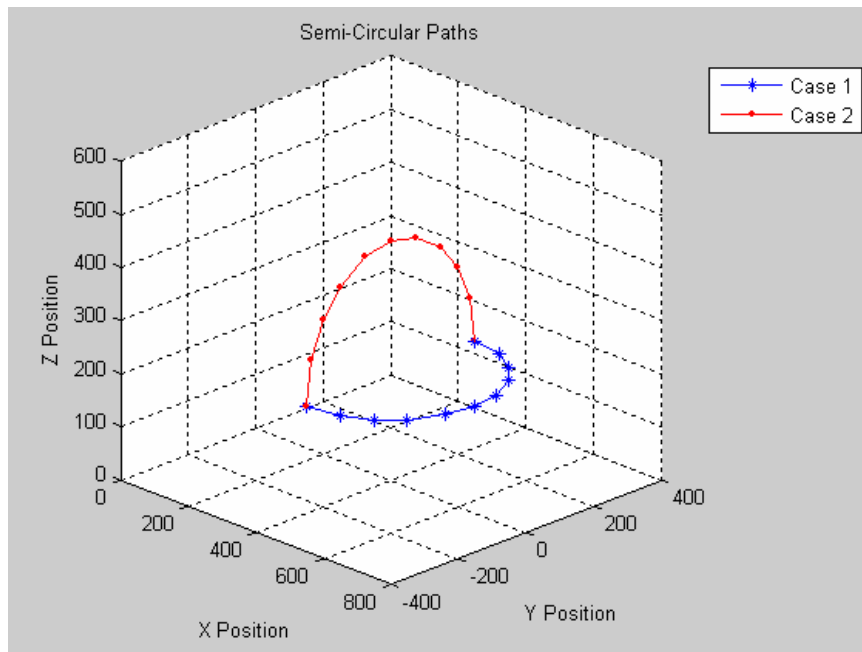


Figure 3-7: Semi-circular paths for Case 1 (horizontal path) and Case 2 (vertical path).

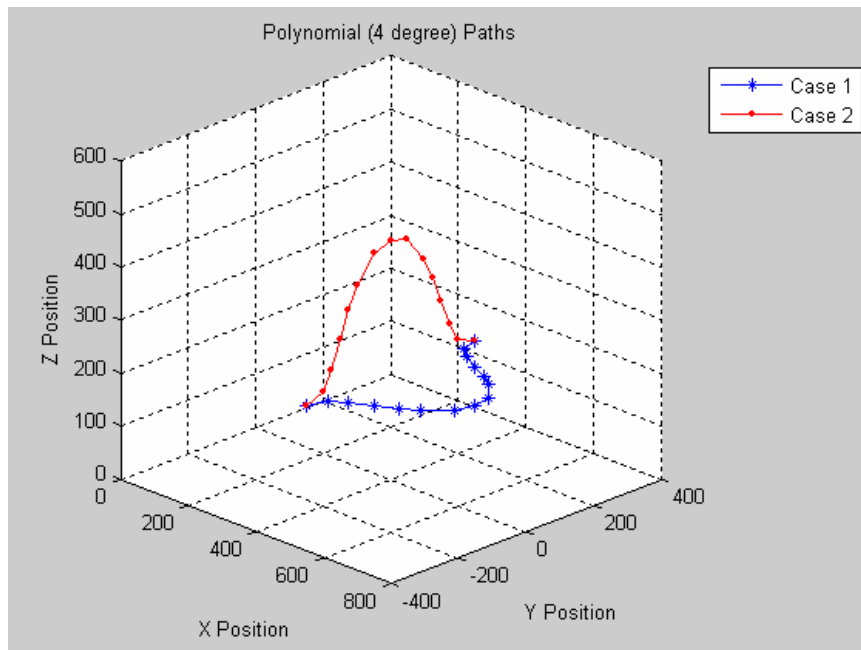


Figure 3-8: Polynomial paths for Case 1 (horizontal path) and Case 2 (vertical path).

3.4 Programming and Simulating the ABB IRB 140 Robot

RobotStudio is an offline programming software made by ABB, the same company which manufactures the IRB 140 robot. The intent of the software is to allow robot users to program and simulate the robot on a computer, rather than having to program and test run on the actual robot, which often means robot downtime or lost productivity. According to ABB, RobotStudio is built on an ABB Virtual Controller that is an exact copy of the real software that runs the ABB robots.

3.4.1 Programming Using RAPID

Both the ABB Virtual Controller and the actual robot controller operate with the programming language, known as RAPID. RAPID is a high-level programming language which, like many other languages, uses many English words as commands. Although it is capable of most typical high level functionalities, such as FOR loops and IF/THEN statements, they will not be needed in this paper. Instead, RAPID will be used as a method to control the motions of the robot by programming specific path points, velocity, and movement options. Without delving into too much detail, there are two main commands that are used in the RAPID programs for this paper: the data type *robtarget* and the function *MoveL()*. Using *robtarget* allows a specific point, or target, to be programmed into the controller's memory and can be recalled by name, such as *p20*. This data type stores x, y, z coordinates, orientation, and other axes angles not needed here. *MoveL()* can then be used to move the robot in a linear motion from target to target. The syntax is:

```
MoveL ToPoint Speed Zone Tool;
```

where *ToPoint* is a specific *robtarget*, *Speed* is one of the predefined velocities, *Zone* defines how close the manipulator needs to get to the target, and *Tool* describes which tool, if any, is on the manipulator. A typical *MoveL()* command might look like the following:

```
MoveL p20, v1000, fine, tool0;
```

The argument 'v1000' is a predefined maximum speed of 1000 mm/s, 'fine' means that the manipulator must exactly pass through the point (as opposed to an argument such as 'z10' which allows the manipulator to miss the target by 10 mm), and 'tool0' is the

default argument meaning that no tool is attached. Note that a zone of 'z0' can also be used in the place of 'fine,' which can help result in smoother transitions between targets. For more detailed information on these functions or RAPID, refer to the [Introduction to RAPID](#) manual (ABB Robotics AB, "Operating Manual"). Sample RAPID code used in this paper can be viewed in Appendix E.

3.4.2 Simulating Using RobotStudio

After writing RAPID code, mainly consisting of the above mentioned commands numerous times (calling each once for every path point), a simulation can be run using RobotStudio. This simulation gives a graphical validation that the RAPID code was programmed correctly to follow the desired path. In addition, RobotStudio has an option to calculate the process time of specific events during the simulation. This option was enabled to find an estimate of the total traveling time of the robot from the start to end point along the specified path. Note that this traveling time is not necessarily the minimum travel time for the path since velocity is limited in the RAPID code. Having the total process time, or t_f of Equation 2.18, gives a good way to compare and evaluate the performances of various path geometries. More importantly, it is needed to calculate the length of the equal time intervals (Equation 2.18) since there is a known number of points composing a path and the points are chosen to approximately meet this assumption of equal time intervals. Knowing the length of the intervals makes it possible to calculate variables, such as joint velocity, acceleration, torque, and energy consumed per time interval.

3.5 Calculation of Consumed Energy

Calculating the exact amount of consumed energy is a very difficult task, since there are many additional variables that would need to be modeled, including friction and wear or fatigue in the motors, electrical properties, and other anomalies. Therefore, calculations of consumed energy for ideal conditions are performed using Diken's approach as outlined by Equations 2.18 through 2.21. These equations are implemented in the MATLAB code which is available in Appendix F and summarized in the flowchart below.

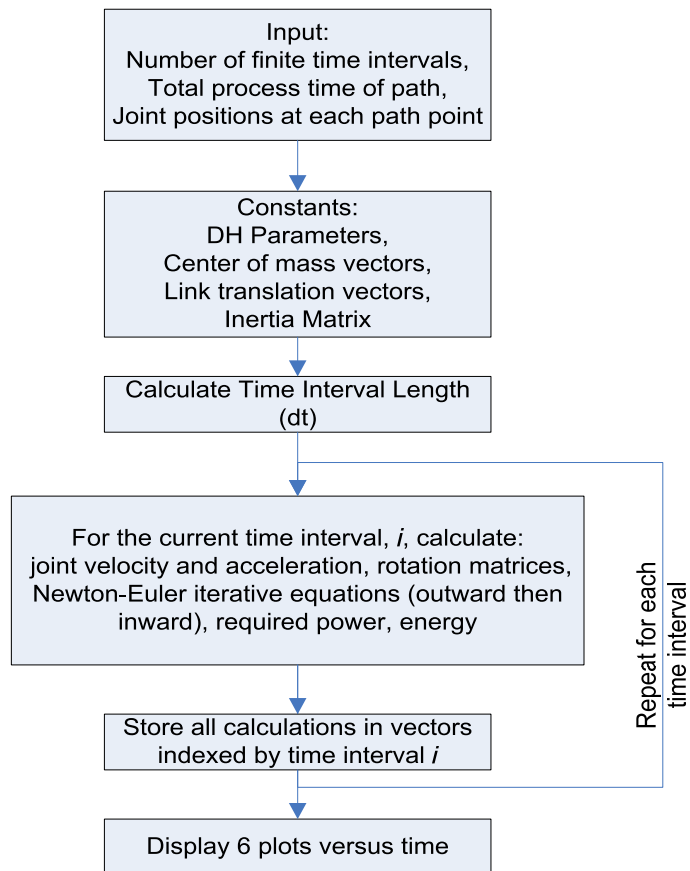


Figure 3-9: MATLAB flowchart.

Chapter 4

RESULTS

4.1 Introduction

This chapter discusses the outcomes of the simulations and calculations performed in this paper. In Section 4.2, an overview is given of the MATLAB graphical results and a table summarizes the numerical results for all paths. Sections 4.3 and 4.4 compare and analyze specific path geometries within Case 1 and 2, respectively. Since the travel times for each path were determined by simulation, Section 4.5 validates and performs a sensitivity analysis on the travel time for a particular path. Section 4.6 makes path recommendations based on certain process efficiency metrics and Section 4.7 follows with general observations that can be used as a starting point when planning a path.

4.2 Results

The MATLAB Code, as described in the previous chapter, has been run for all paths analyzed. The results are shown in Table 4-1 and are compared to the horizontal triangular path in Table 4-2. Each time the code is run, it displays six graphs for the following variables versus time: joint angle position, joint velocity, joint acceleration, joint torque, joint power, and cumulative energy consumption. Due to the size and number of graphs, all are shown in Appendix G with only selected graphs (typically joint torque or power) shown within this chapter. The graphs of joint angle position, joint

velocity, and joint acceleration are primarily for verification purposes to make sure erroneous data points are not evident and to make sure that the robot is not moving or accelerating faster than the specification limits. For example, the joints considered are not capable of a velocity of more than $200^\circ/\text{s}$, or 3.5 radians/s. The remaining graphs are much more interesting! The joint torque graphs show which joints are more affected by gravitational forces, especially when in positions extended far away from the robot's center of gravity. It is also usually very evident when there is a sudden change in the direction of the robot by the spike(s) in torque. The joint power graph is also interesting, since it shows which parts of the paths require the most power. Much of the power graphs are predictable, however, there are some points where torque is high but power is much lower than expected. This is typically due to the joint moving very slowly and the fact that power is calculated as the product of the two (Equation 2.19), resulting in low power. Finally, the cumulative energy consumption graph gives a good perspective of which segments of the path demand the most energy. Since the energy consumption is

Table 4-1: Summary of travel time, travel distance, and energy consumed for each path.

Path Geometry	Case Type	Travel Time, t_f (sec)	Travel Distance(mm)	Total Energy Consumed (J)
Triangular	Case 1 - Horizontal	2.4	707	70.49
Square	Case 1 - Horizontal	4.2	1000	124.83
Rectangular	Case 1 - Horizontal	4.3	1000	93.81
Semi-Circular	Case 1 - Horizontal	2.5	785	72.13
Polynomial - 4	Case 1 - Horizontal	3.0	750	76.77
Triangular	Case 2 - Vertical	2.4	707	83.94
Square	Case 2 - Vertical	4.2	1000	84.23
Rectangular	Case 2 - Vertical	4.3	1000	75.78
Semi-Circular	Case 2 - Vertical	2.5	785	78.80
Polynomial - 4	Case 2 - Vertical	3.0	750	82.09

Table 4-2: Percent differences compared to the horizontal triangular path geometry.

Path Geometry	Case Type	% Difference Travel Time tf	% Difference Travel Distance	% Difference Total Energy Consumed
Triangular	Case 1 - Horizontal	-	-	-
Square	Case 1 - Horizontal	75.0	41.4	77.1
Rectangular	Case 1 - Horizontal	79.2	41.4	33.1
Semi-Circular	Case 1 - Horizontal	4.2	11.0	2.3
Polynomial - 4	Case 1 - Horizontal	25.0	6.1	8.9
Triangular	Case 2 - Vertical	0.0	0.0	19.1
Square	Case 2 - Vertical	75.0	41.4	19.5
Rectangular	Case 2 - Vertical	79.2	41.4	7.5
Semi-Circular	Case 2 - Vertical	4.2	11.0	11.8
Polynomial - 4	Case 2 - Vertical	25.0	6.1	16.5

the integral, or area under the curve, with respect to time, another way one can get a rough estimate of the areas where a lot of energy is consumed by looking at peaks in the power requirement graphs.

The analysis of the remaining process efficiency metrics is much less complex. As noted previously, the travel distance was determined by calculating arc lengths for the paths defined by functions and by basic perimeter calculations for the remaining paths. The process travel time was determined by simulation of the path in RobotStudio. All of these results are also shown in Table 4-1 and compared to the horizontal triangular path in Table 4-2. It appears as though there are some correlations between some metrics, but there are certainly some exceptions. More detail will be given in the following discussions.

4.3 Case 1 Discussion

One may notice that for all of the horizontal paths in Case 1, Joint 2 experiences much more torque than the other joints. This is logical since Joint 2 is responsible for most of the reaching of the robot, and as the robot reaches further out away from the center of gravity, there is a torque created by gravity and the mass of the remaining links. If the joint has a high velocity at those same point of high torque, a large amount of power is required. It should also be pointed out that both the highest and lowest energy consumption of all paths examined are found within Case 1, suggesting that horizontal moves are not always less energy consuming than efficient vertical paths.

Regarding the relationship between travel distance or travel time and energy consumed, there appears to be a correlation—especially in Case 1—but there are exceptions. For example, the triangular and semi-circular paths are among the paths with the shortest traveling times and distances, while consuming the least amount of energy. However, the semi-circular path had a longer travel distance than the polynomial, but took less travel time and less energy. Nonetheless, this relationship leads to a general observation expressed further in the paper (Section 4.7).

4.3.1 Triangular Path Geometry

Having the shortest travel distance of 707 mm, it is not a big surprise that the travel time is also the shortest at 2.4 seconds. In the horizontal case, the triangular path also consumes the least energy of all paths examined at 70.49 J. In examining the torque

graph, there are spikes of torque in the beginning of the path and right after the peak of the triangle, which makes sense because of the sudden changes in direction. It is worth noting that the total power level for all joints (Figure 4-1) also has a somewhat triangular curve (mainly due to Joint 2), suggesting that the robot consumes the most energy when the arm is reached out the furthest.

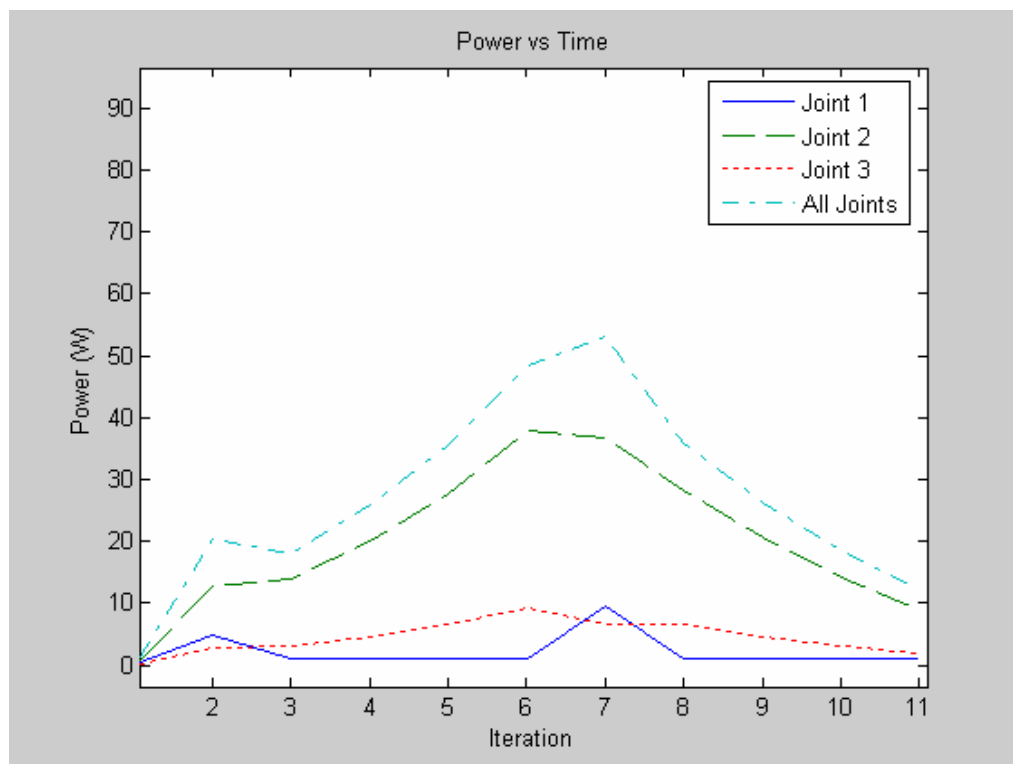


Figure 4-1: Power requirements of the horizontal triangular path also follow a somewhat triangular curve.

4.3.2 Square Path Geometry

With the square path geometry, it is very clear in almost all of the graphs where the robot changes direction. In variables such as joint acceleration and joint torque

(Figure 4-2), there are spikes when first starting the path, and then 2 additional spikes when the path changes at right angles. Interestingly, during the path between the two right angles, Joint 2 experiences high torque but has close to zero velocity, resulting in little power required in the middle of the path. It is when Joint 2 moves out and then back in that requires high peaks in power since there is a high torque and a non-zero velocity, which results in the path that is the highest in energy consumption at 124.83 J. Not only is the square path highest in energy consumption, but it also requires the longest travel distance at 1000 mm and one of the longest travel times at 4.2 seconds. Hence, this path is not very desirable in any efficiency metric.

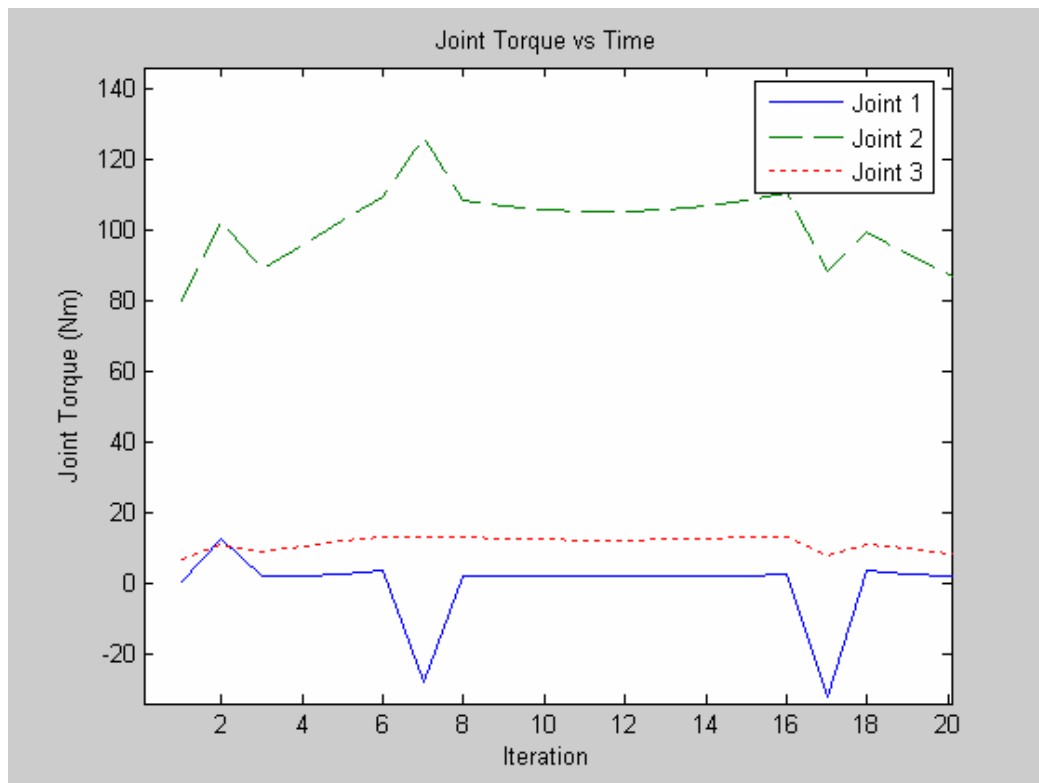


Figure 4-2: Joint torque of the square path (Case 1) with spikes at changes in direction.

4.3.3 Rectangular Path Geometry

In regards to spikes in variables, the rectangular path is even worse than the square in that it has five spikes due to the initial start and four right angle moves. The power requirements, as expected, are not terribly different than the horizontal square path except that the peak power spikes are moved closer together. After all, the robot does not move away from the base until the middle of the path. Like the square path, there is a period of very little power demanded (and therefore little additional energy consumption) when at the furthest points away, where the highly torqued Joint 2 does not move and

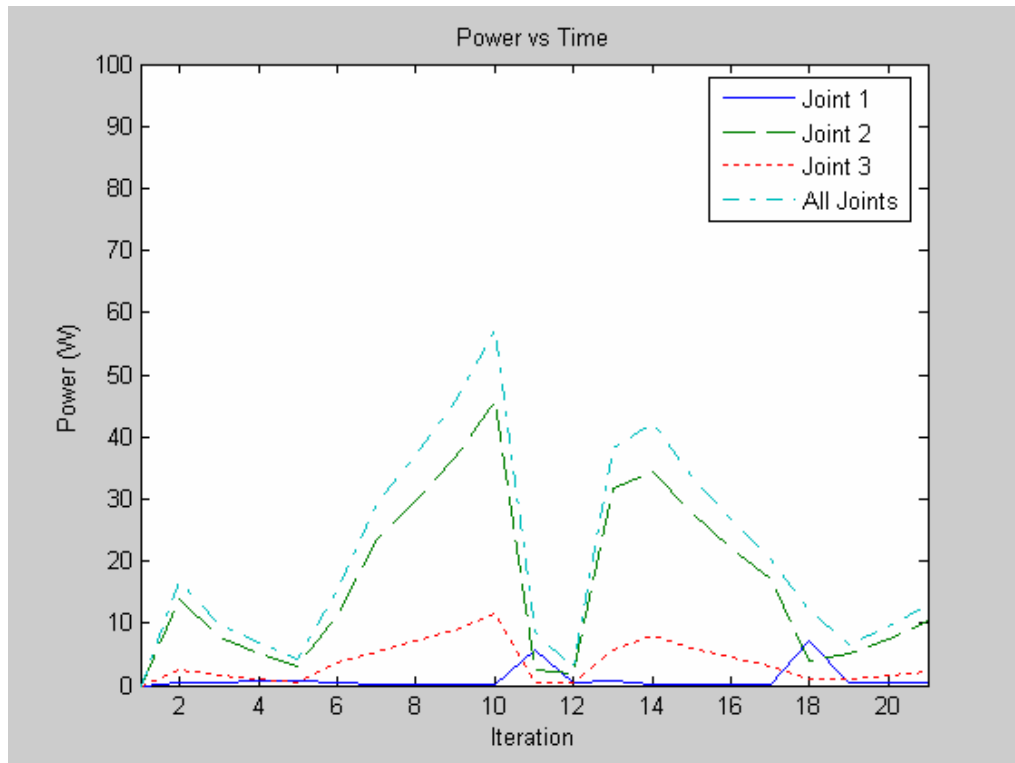


Figure 4-3: Joint power requirements of the horizontal (Case 1) rectangular path with peaks of power when the robot uses Joint 2 to move away and towards the base.

Joint 1 simply rotates the robot. This is shown in Figure 4-3 between iteration 11 and 12. Also like the square path, the travel distance is tied at the largest length of 1000 mm and requires the longest travel time of 4.3 seconds to complete.

4.3.4 Semi-Circular Path Geometry

The semi-circular geometry generates a path that is very smooth, not only in position but in many other graphs, and has only one minor spike in the graphs for when the robot begins moving. However, since there is a spike in velocity and a spike in torque, the result is a high power requirement in the first few points. Regardless, the travel time, distance traveled, and energy consumed are all very comparable to that of the best path (triangular Case 1)—only 4.2%, 11.0%, and 2.3% larger than the triangular Case 1 path. Therefore, the semi-circular path lends itself to being a good alternative to the triangular path.

4.3.5 Polynomial Path Geometry

The polynomial geometries provided smooth joint positions, but progressively get choppy when taking first and second derivatives for joint velocity and acceleration. The power curve looks very similar to that of the rectangular Case 1 (Figure 4-3), but of slightly less energy consumed compared to the rectangular path. It is probable that parametric equations of position would help the smoothness of the velocity and acceleration curves. In terms of energy consumption and travel distance, the polynomial

compares respectably to the triangular Case 1 path, but the travel time is 25% larger at 3.0 seconds. Although the travel distance is slightly shorter than the semi-circular path, the polynomial path still requires slightly more travel time and energy.

4.4 Case 2 Discussion

In Case 2, where vertical paths are examined, three of the five vertical paths required more energy than the same horizontal path in Case 1, showing that overcoming the effects of gravity may require more power. The remaining two geometries are square and rectangular paths, but in Case 2, the paths move along much closer to the center of gravity and not in an extended position as in Case 1.

In general, paths in Case 2 have high peaks of torque for Joint 2 in the first and last few points and very little torque in the middle of the path. This is unlike most paths in Case 1, where the Joint 2 torque was somewhat high over the entire path. The reason for this outcome is due to the position and movement of Joint 2 in Case 2: first, there is an upward movement against gravity, then the robot is in an upright position in the middle of the path, and finally the robot is moving downwards with gravity and must slow down to remain on the path. When in the middle of the path, the robot is closer to being above the center of gravity and thus less torque is exerted on the joint. The high torque in the first and last few points makes the power curves peak closer to the edges of the graphs compared to most of Case 1 paths. Nonetheless, the power curves look similar in that they drop to near zero in the middle, as there is not much joint velocity or torque at that point in the path.

Regarding the relationship between travel distance or travel time and energy consumed, it is much more difficult than it was for Case 1 to say that there is a correlation. It is likely that since all of the vertical paths operate at close proximity to the robot base (as opposed to reaching far out for some of the horizontal paths), the change in energy among the vertical paths are all less than 11%. Perhaps for vertical paths, the location of the path may have more of an impact than the travel time or distance of the path.

4.4.1 Triangular Path Geometry

Although the triangular path consumed the least energy of all the paths in Case 1, it was one of the higher energy consuming paths in Case 2. The high initial and final torque on Joint 2 creates high power requirements, as discussed previously. The power curve is shown in Figure 4-4 and is quite the opposite of the triangular Case 1 power curve (Figure 4-1). Also quite the opposite is how the triangular path ranked among the energy consumed for the vertical paths. Although the path still takes only 2.4 seconds over 707 mm, the energy consumed is 83.94 J, which is the second highest among Case 2.

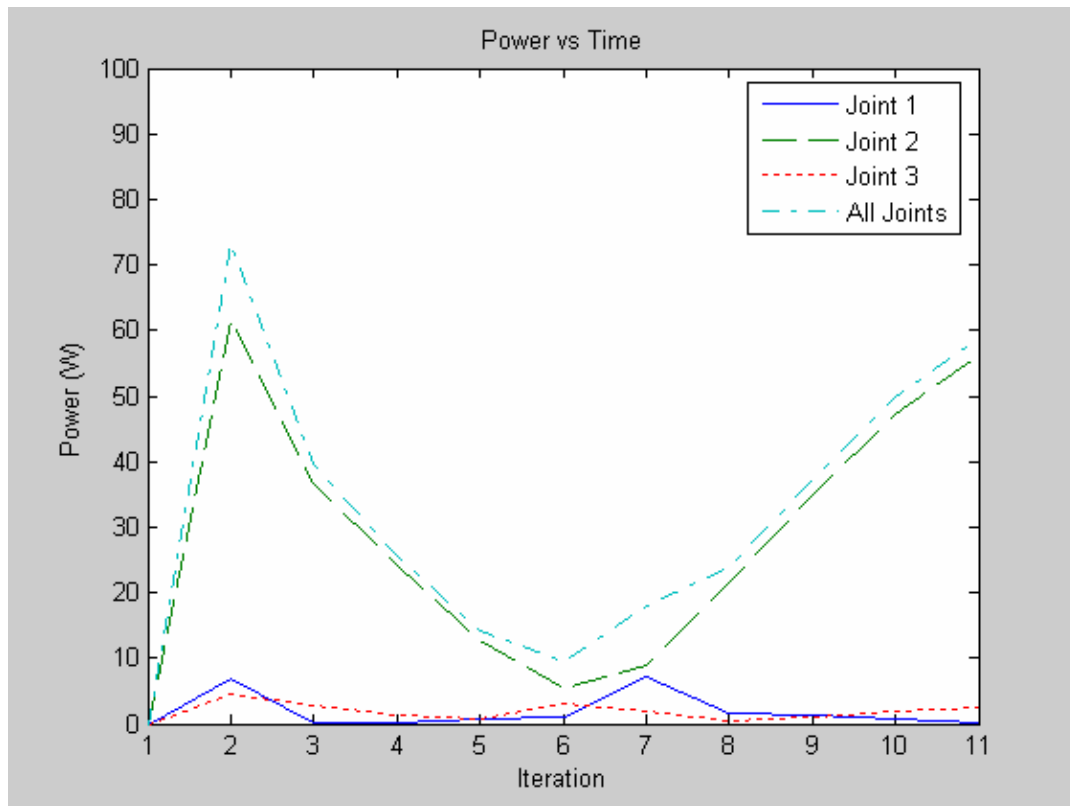


Figure 4-4: Power requirements of the vertical (Case 2) triangular path with peaks at the initial and final points.

4.4.2 Square Path Geometry

The square path geometry in Case 2 is very similar to Case 1 in that almost all of the graphs clearly show where the robot changes direction. It is also similar to Case 1 in that during the path between the two right angles, Joint 2 experiences high torque (Figure 4-5) but has close to zero velocity, resulting in little power required in the middle of the path. It is when Joint 2 moves up and down that requires high peaks in power since there is a high torque and a velocity. This is shown in the first and last few points, since this is where the robot is affected by gravity most. Throughout the path, Case 2 has much less

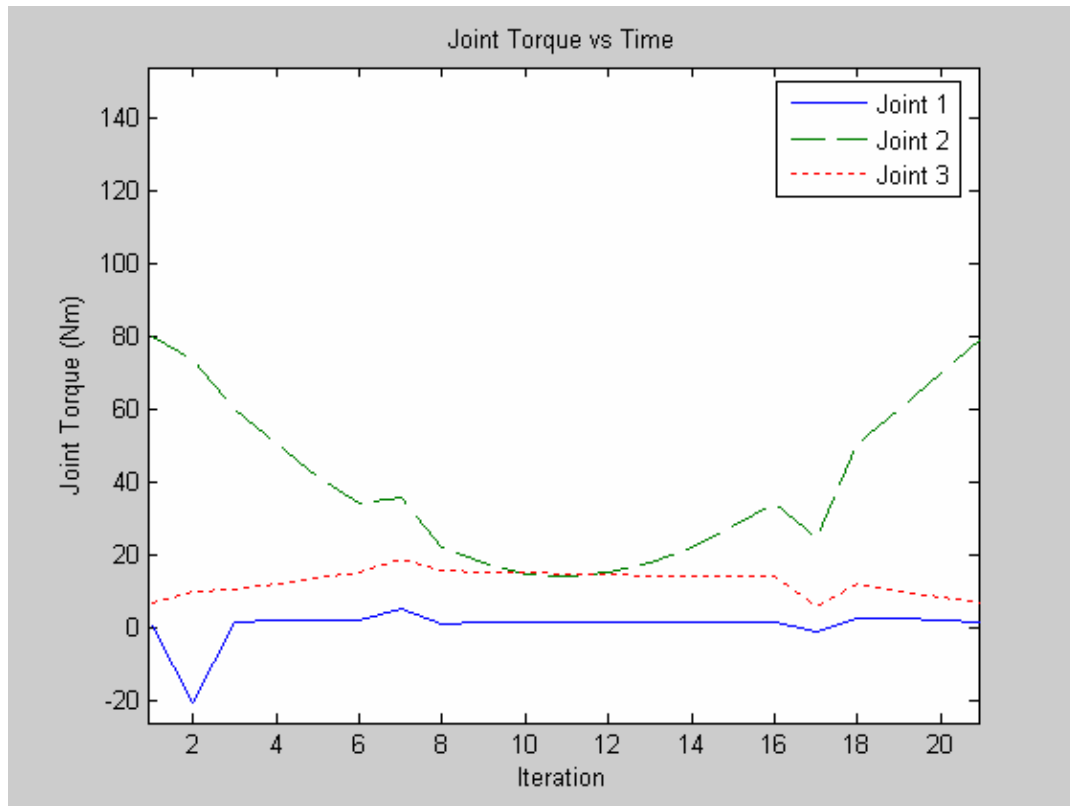


Figure 4-5: Joint torque of the vertical square path with peaks at the beginning and end of the path.

torque than Case 1 due to the fact that the arm operates much closer to the robot base.

Like in Case 1, the square path in Case 2 has the highest energy consumption of the vertical paths, the longest travel distance at 1000 mm and one of the longest travel times at 4.2 seconds. Hence, this path is also not very desirable in any efficiency metric.

4.4.3 Rectangular Path Geometry

As mentioned previously, in paths with many direction changes, there are very evident spikes in the various graphs. While the power graph looks similar to the horizontal rectangular path, the torque graph (Figure 4-6) shows similar characteristics as

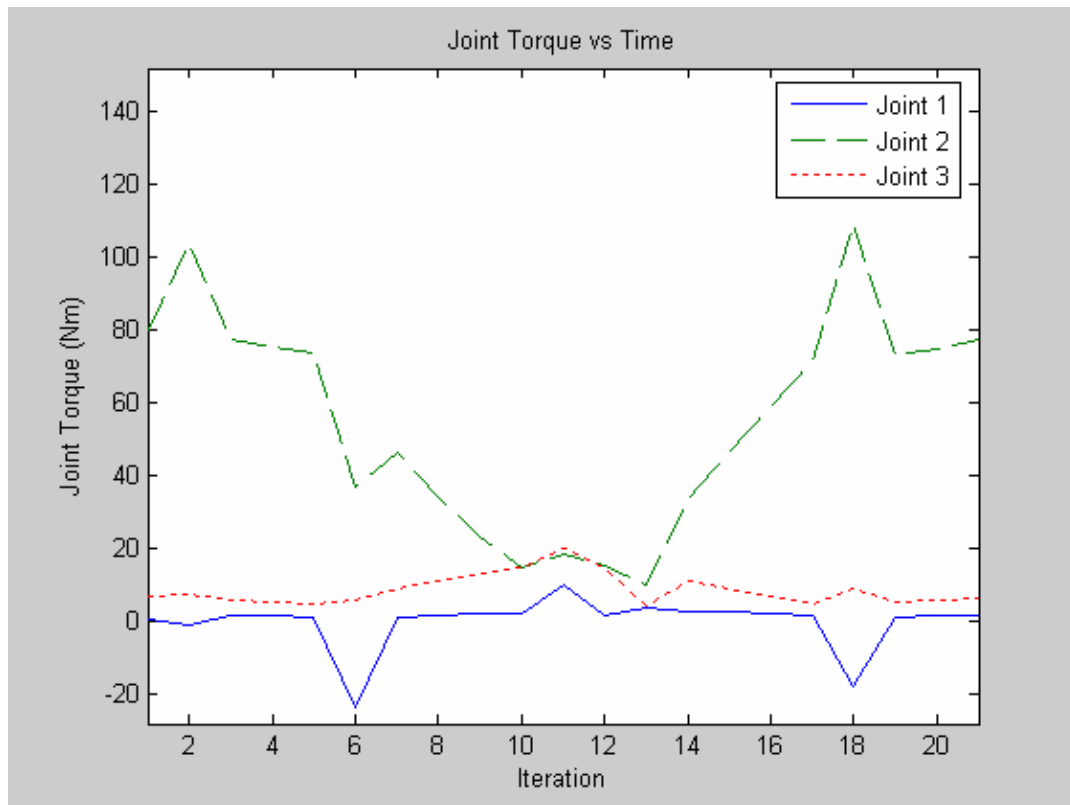


Figure 4-6: Joint torque of the vertical (Case 2) rectangular path with spikes when the robot changes direction and larger peaks of torque during the first and last few points.

most other Case 2 paths do with Joint 2 dipping in the middle of the path as well as evidence of spikes due to changes in direction. Additionally, due to the path geometry, Joint 2 does not move during the first and last few points. This is the same area where Joint 2 is high in torque, yet since power is the product of the velocity and torque, the power is actually minimal at these stages. For this reason, the consumed energy for this path is the lowest of all the Case 2 (vertical) paths at 75.78 J. The downfall with this path is that it has the longest travel time of 4.3 seconds and the largest travel distance of 1000 mm. Unless the manufacturer is only concerned with minimizing energy, this path is not a likely choice.

4.4.4 Semi-Circular Path Geometry

The semi-circular geometry, once again, generates a path that is very smooth (Figure 4-7), not only in position but in many other graphs. Due to the high torque in the first and last few points, the power graph has peaks that are spread apart from each other. Depending on the goals and costs of the manufacturer, the semi-circular path may be more favorable over the rectangular Case 2, despite the latter being the vertical path of lowest consumed energy. For example, if travel time, distance and changes in direction were more important than energy consumption, the vertical semi-circular path would

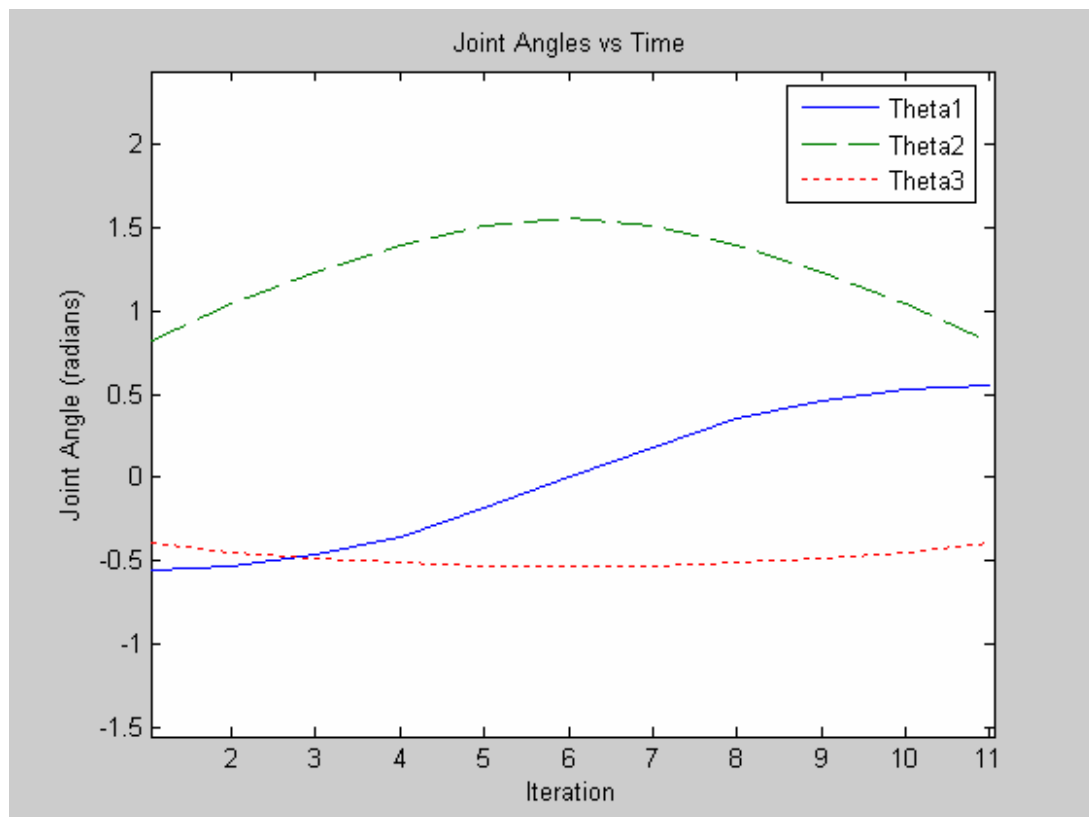


Figure 4-7: Joint position of the vertical (Case 2) semi-circular path, which is very smooth since there are no major direction changes like other paths examined.

probably be preferred over the vertical rectangular path since the travel time is significantly lower, the distance is shorter, there are no sudden changes in direction, and the energy consumption is only 4% larger than that of the rectangular path.

4.4.5 Polynomial Path Geometry

As in Case 1, the polynomial geometry provided smooth joint positions, but progressively get choppy when taking first and second derivatives for joint velocity and acceleration. Even so, the power curve looks very similar to most of those in Case 2.

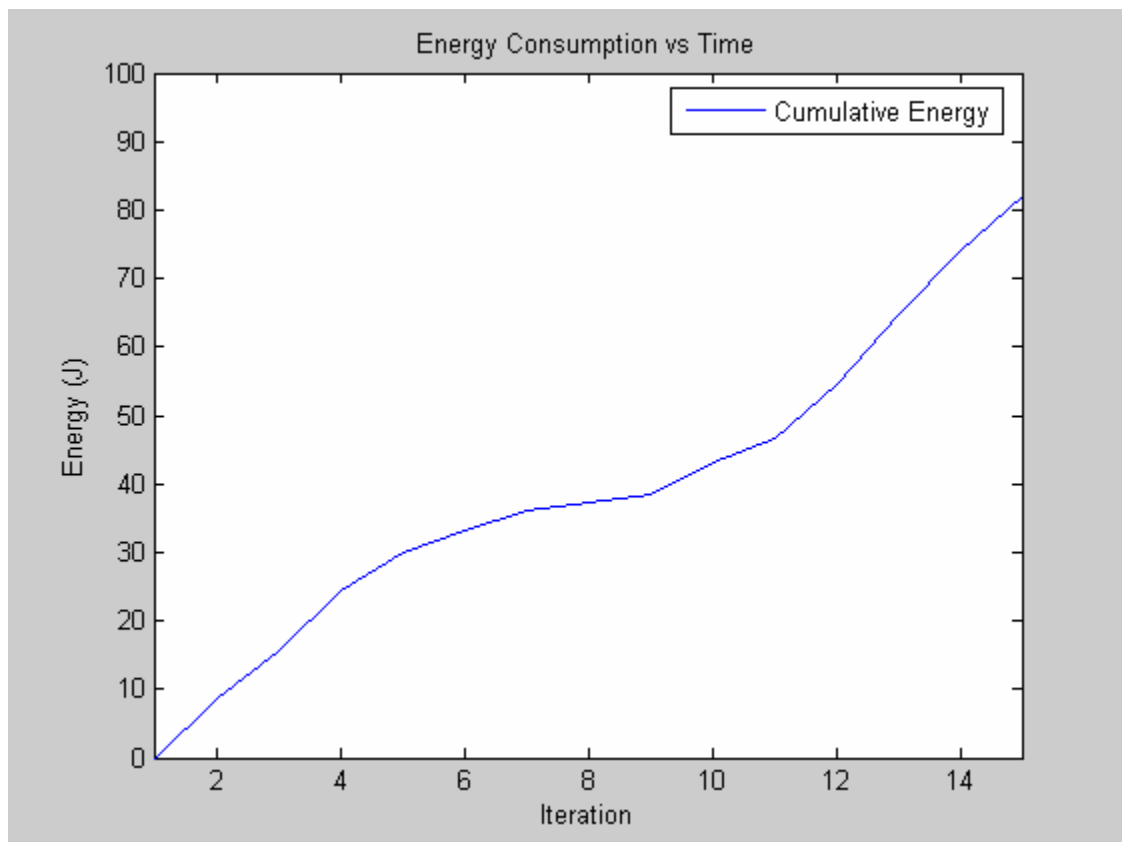


Figure 4-8: Cumulative energy consumption of the vertical (Case 2) polynomial path, which is very typical for all of the vertical paths examined.

The cumulative energy curve shown in Figure 4-8, therefore, looks very typical for the vertical paths, with large energy consumption in the same areas as peak power requirements and little energy consumption in the middle of the path. Although the performance of the polynomial path is good in that it has no sudden changes in direction, the performance of the path is only mediocre in terms of travel time, travel distance, and energy consumption.

4.5 Travel Time Validation and Sensitivity Analysis

The travel times above were found from running a simulation for all of the paths in RobotStudio. Due to the fact that simulations do not always account for real-world scenarios, RAPID programs were loaded and ran on an actual ABB IRB 140 robot to validate the results. Further analysis was performed on the horizontal triangular path since it was the fastest and least energy consuming path. When running the simulation, the travel time is said to be 2.4 seconds, whereas when timed on the actual robot with a stopwatch, the travel time was approximately 4.0 seconds. The actual time was much higher than anticipated, so a sensitivity analysis was performed to see the effects of such an error on the MATLAB results. Interestingly, the energy change between travel times of 2.4 and 4.0 is about 4 joules less, or a reduction of only 5%. Increasing the travel time in the MATLAB code changes the length of the time interval, but would keep the changes in joint position the same. This would therefore result in a lower joint velocity and acceleration in the graphs. At a certain point, joint torque and total energy consumption are impacted less and less. Figure 4-9 shows that after 3 seconds, the

energy curve begins to approach what appears to be a horizontal asymptote. Conversely, as the travel time approaches zero, the energy calculation rises enormously, as it should, since a very fast path would require the joints to move and accelerate much more than the motor capabilities. Of course, there is some minimum travel time based on physical constraints that will prevent the travel time from being extremely fast. Caution should be used when using the RobotStudio travel time, since it tends to underestimate the actual time and, if too low, could cause the energy results to be much higher. In this particular path, the underestimate does not have a significant impact on the energy calculation.

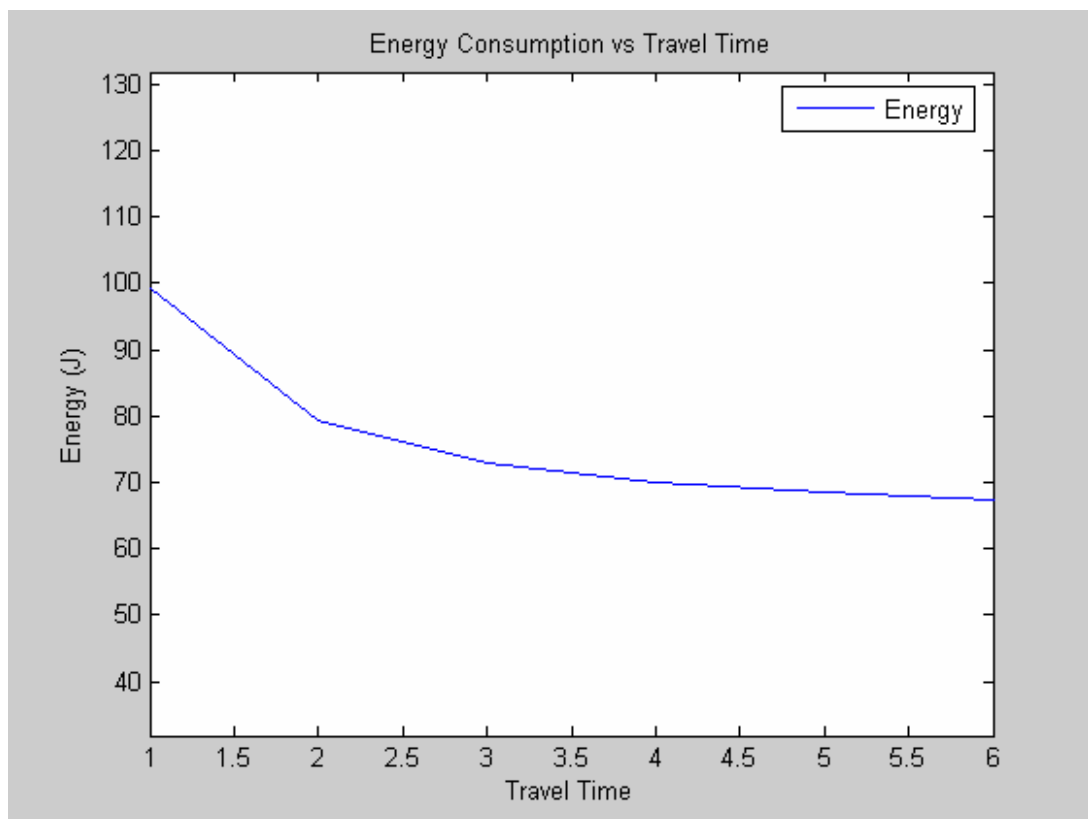


Figure 4-9: Energy consumption of the triangular (Case 1) path as travel time changes.

4.6 Recommendations

If travel time has a significant impact on the production rate or the process needs to be completed as quickly as possible, it makes sense to choose the path with the fastest process time. The two paths of lowest travel times are the triangular and semi-circular paths.

If motor life and replacement is a major concern or cost, the semi-circular paths are recommended since they have no sudden change in direction like most other paths (with the exception of the polynomial) and do not travel unnecessarily far. The semi-circular paths have near minimum travel time and distances, and consistently rank at the second least energy consumed for both the horizontal and vertical cases.

If the total energy consumed is the largest cost or initiative, the horizontal triangular case is recommended. Although it has one sudden change in direction, it still has the minimum travel time, travel distance, and energy consumed of all the paths examined in this paper.

4.7 General Observations

While it is difficult to describe every single anomaly in the results of this paper, there are several general observations that could be argued using physics and the trends of the results. As expected, paths of the shortest distance will generally result in paths that take the shortest amount of time to travel. This makes sense because there is an assigned and limited linear velocity that the robot will try to obtain whenever possible,

and for the same velocity, shorter distances will take less time than longer distances. The shortest distance between two points is a straight line, which is shown by the triangular paths composed of two straight lines. The result is the path which has the shortest traveling distance and the shortest traveling time.

It can also be argued that lower travel times will generally result in lower amounts of energy consumed. The results of Case 1 seem to demonstrate this: as travel times increase, so does the energy consumption. This can be explained by the fact that energy is the time integral of power and if robot motors output a somewhat constant amount of power (or maximum power), shorter travel times will result in less energy consumed. Of course, the motors are not always at a constant or maximum power and a long travel time of low power could result in less energy compared to a short travel time of maximum power.

Paths of numerous sudden changes in motion, such as the rectangular paths, should be avoided if possible to reduce the wear of motors. It is very evident in the MATLAB graphs where the changes occur by the spikes in variables such as acceleration and torque. On the other hand, paths such as the semi-circular geometry have no spikes except for the initial start-up movement.

Chapter 5

CONCLUSIONS AND FUTURE RESEARCH

5.1 Conclusions

Companies are always looking to save money and are being pressured to become environmentally conscious. One such method to reaching these goals is to increase process efficiency of manufacturing machines, such as industrial robots, while not preventing them from accomplishing what they were designed to do. This study has examined various path geometries that a robot manipulator can move along, from a start position to a goal position while circumventing an obstacle along the way. For each of the five path geometries, two cases were analyzed: Case 1 was a path that moved strictly horizontally while Case 2 was the same path in a vertical motion. Each path was then evaluated and compared by the amount of energy consumed, the travel time and the distance over the entire path.

Of all paths examined, a triangular path moving horizontally showed to consume the least amount of energy while also requiring the least amount of travel time and travel distance. Among only the vertical cases, the rectangular path consumed the least amount of energy, but has the longest travel time and distance. However, depending on the goals and costs of the manufacturer, other paths may be chosen to minimize certain criterion. For example, if travel time, distance and changes in direction were considered more

important than energy consumption, the semi-circular path would probably be preferred over the rectangular path in the vertical case. This is because semi-circular travel time is significantly lower, the distance is shorter, there are no sudden changes in direction, and the energy consumption is only 4% larger than that of the rectangular path.

The travel time, which was predicted by simulation and validated by timing the actual robot, was analyzed to determine the effect the error had on energy consumption. It was suggested that caution should be exercised when using the RobotStudio predicted travel time, since it tends to underestimate the actual time. If the travel time is analyzed at too low of a number, it could result in a very erroneous energy calculation. However, reasonable or slow travel times result in an energy calculation that is not very sensitive. In the case of the recommended horizontal triangular path, the underestimate did not have a significant impact on the energy calculation.

While it is difficult to describe every single anomaly in the results of this paper, there were several general observations that could be argued using physics and the trends evident in the results:

- 1) Paths of the shortest distance will generally result in paths that take the shortest amount of time to travel;
- 2) Paths of shorter travel times will generally result in lower amounts of energy consumed;
- 3) Paths of numerous sudden changes in motion, such as the rectangular paths, should be avoided if possible to reduce the wear of motors.

These general observations do not always hold true, but can be used as a starting point in the path planning stage of programming a robotic arm.

5.2 Future Research

The work related to this thesis leads to numerous directions for further research. One obvious extension would be to validate the power requirements or energy consumption with an electrical device, such as a power or current probe, and log the data over time as the robot moves along various paths. This was attempted in this research but proved to be quite technical in determining which wires in the robot controller would need to be measured. Therefore, an experienced technician and wiring diagrams for the controller would be needed.

Another extension would be to further examine some of the paths in this paper by replacing the sharp changes in direction with arcs of various radii. This may help reduce the spikes in the various graphs seen in this paper and may have significant impacts on all of the efficiency metrics.

Of course, as discussed briefly in Chapter 2, many sophisticated path planning techniques exist that minimize certain criterion such as travel time while using various methods such as parametric equations or cubic polynomials. To extend this research, one could first apply some of these advanced optimization techniques and then, using a similar approach to this paper, determine which paths are most efficient in a multi-criteria optimization problem.

REFERENCES

- ABB Robotics AB. IRB 140 M2000 Product Specification. Article 3HAC 9041-1,
Rev 2. Sweden: ABB Robotics AB.
- ABB Robotics AB. Operating Manual: Introduction to RAPID.
Article 3HAC029364-001. Sweden: ABB Robotics AB.
- Bobrow, James E. "Optimal Robot Path Planning Using the Minimum-Time Criterion."
IEEE Journal of Robotics and Automation 4.4 (1988): 443-50.
- Constantinescu, D. and E.A. Croft. "Smooth and Time-Optimal Trajectory Planning for
Industrial Manipulators along Specified Paths." Journal of Robotic Systems 17.5
(2000): 233-249.
- Craig, John J. Introduction to Robotics: Mechanics and Control. 3rd ed.
New Jersey: Pearson Prentice, 2005.
- Diken, Hamza. "Energy Efficient Sinusoidal Path Planning of Robot Manipulators."
Mech. Mach. Theory 29.6 (1994): 785-92.
- Field, Glen and Yury Stepanenko. "Iterative Dynamic Programming: An Approach to
Minimum Energy Trajectory Planning for Robotic Manipulators." Proceedings of
the 1996 IEEE International Conference on Robotics and Automation.
Minneapolis, 1996. 2755-2760.
- Hollerbach, John M. "Kinematics and Dynamics for Control." Robotics Science.
Ed. Michael Brady. Cambridge: MIT Press, 1989. 378-431.
- Manseur, Rachid. Robot Modeling and Kinematics. Boston: Da Vinci, 2006.

Nagurka, Mark L. "Newton-Euler Dynamics of Robots." Robotics and Automation

Handbook. Ed. Thomas R. Kurfess. New York: CRC Press, 2005. 4.1 – 4.10.

Vicente, David Barrio. "Modeling and Balancing of Spherical Pendulum using a Parallel

Kinematic Manipulator." Master Thesis, Lund Institute of Technology.

Sweeden: 2007.

Appendix A

Transformation Matrices

$${}^0_1T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^3_4T = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ 0 & 0 & -1 & -d_4 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^1_2T = \begin{bmatrix} c_2 & -s_2 & 0 & a_1 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^4_5T = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^2_3T = \begin{bmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^5_6T = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The following are needed for deriving inverse kinematics and Jacobian equations:

$${}^0_2T = {}^0_1T {}^1_2T = \begin{bmatrix} c_1c_2 & -c_1s_2 & s_1 & c_1a_1 \\ s_1c_2 & -s_1s_2 & -c_1 & s_1a_1 \\ s_2 & c_2 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0_3T = {}^0_2T {}^2_3T = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & c_1(c_2a_2 + a_1) \\ s_1c_{23} & -s_1s_{23} & -c_1 & s_1(c_2a_2 + a_1) \\ s_{23} & c_{23} & 0 & s_2a_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0_4T = {}^0_3T {}^3_4T = \begin{bmatrix} c_1c_{23}c_4 + s_1s_4 & -c_1c_{23}s_4 + s_1c_4 & c_1s_{23} & c_1[s_{23}d_4 + c_2a_2 + a_1] \\ s_1c_{23}c_4 - c_1s_4 & -s_1c_{23}s_4 - c_1c_4 & s_1s_{23} & s_1[s_{23}d_4 + c_2a_2 + a_1] \\ s_{23}c_4 & -s_{23}s_4 & -c_{23} & -c_{23}d_4 + s_2a_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0_5T = {}^0_4T {}^4_5T = \begin{bmatrix} c_5(c_1c_{23}c_4 + s_1s_4) - s_5c_1s_{23} & -s_5(c_1c_{23}c_4 + s_1s_4) - c_1s_{23}c_5 & -c_1c_{23}s_4 + s_1c_4 & c_1[s_{23}d_4 + c_2a_2 + a_1] \\ c_5(s_1c_{23}c_4 - c_1s_4) - s_5s_1s_{23} & -s_5(s_1c_{23}c_4 - c_1s_4) - s_1s_{23}c_5 & -s_1c_{23}s_4 - c_1c_4 & s_1[s_{23}d_4 + c_2a_2 + a_1] \\ c_5(s_{23}c_4) + c_{23}c_5 & -s_{23}c_4s_5 + c_{23}c_5 & -s_{23}s_4 & -c_{23}d_4 + s_2a_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^3_6T = {}^3_4T {}^4_5T {}^5_6T = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 & c_4s_5 & 0 \\ s_5c_6 & -s_5s_6 & -c_5 & -d_4 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 & s_4s_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0_6T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_{xw} \\ r_{21} & r_{22} & r_{23} & p_{yw} \\ r_{31} & r_{32} & r_{33} & p_{zw} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

$$r_{11} = c_1c_{23}(c_4c_5c_6 - s_4s_6) - c_1s_{23}s_5c_6 + s_1(s_4c_5c_6 + c_4s_6)$$

$$r_{12} = -c_1c_{23}c_4c_5c_6 + c_1s_{23}s_5s_6 - s_1s_4c_5s_6$$

$$r_{13} = c_1c_{23}c_4s_5 + c_1s_{23}c_5 + s_1s_4s_5$$

$$r_{21} = s_1c_{23}(c_4c_5c_6 - s_4s_6) - s_1s_{23}s_5c_6 - c_1(s_4c_5c_6 + c_4s_6)$$

$$r_{22} = -s_1c_{23}c_4c_5c_6 + s_1s_{23}s_5s_6 + c_1s_4c_5s_6$$

$$r_{23} = s_1c_{23}c_4s_5 + s_1s_{23}c_5 - c_1s_4s_5$$

$$r_{31} = s_{23}(c_4c_5c_6 - s_4s_6) + c_{23}s_5c_6$$

$$r_{32} = -c_6s_{23}c_4c_5 - c_{23}s_5s_6$$

$$r_{33} = s_{23}c_4s_5 - c_{23}c_5$$

$$p_{xw} = c_1s_{23}d_4 + c_1c_2a_2 + c_1a_1$$

$$p_{yw} = s_1s_{23}d_4 + s_1c_2a_2 + s_1a_1$$

$$p_{zw} = -c_{23}d_4 + s_2a_2 + d_1$$

Appendix B

Inverse Kinematics Derivations

$$\theta_1 = \tan^{-1}\left(\frac{p_{yw}}{p_{xw}}\right) = a \tan 2(p_{yw}, p_{xw}) \quad \text{OR} \quad \theta_1 = \pi + \tan^{-1}\left(\frac{p_{yw}}{p_{xw}}\right) = a \tan 2(-p_{yw}, -p_{xw})$$

Using the Law of Cosines:

$$\cos \theta_3 = \frac{(p_{xw} + a_1 \cos(\theta_1))^2 + (p_{yw} + a_1 \sin(\theta_1))^2 + (p_{zw} - d_1)^2 - a_2^2 - d_4^2}{2a_2d_4} = D$$

We could solve for θ_3 with $\theta_3 = \cos^{-1}(D)$, however, to account for the “elbow-up and elbow-down” solutions:

$$\theta_3 = \tan^{-1}\left(\frac{\pm \sqrt{1 - D^2}}{D}\right) = a \tan 2(\pm \sqrt{1 - D^2}, D)$$

Similarly,

$$\theta_2 = a \tan 2\left(p_{zw} - d_1, \sqrt{(p_{xw} + a_1 \cos(\theta_1))^2 + (p_{yw} + a_1 \sin(\theta_1))^2}\right) - a \tan 2(d_4 \sin(\theta_3), a_2 + d_4 \cos(\theta_3))$$

$${}^0_3T^{-1} {}^0_6T = {}^3_4T {}^4_5T {}^5_6T$$

$$\begin{bmatrix} c_1 c_{23} & s_1 c_{23} & s_{23} & 0 \\ -c_1 s_{23} & -s_1 s_{23} & c_{23} & 0 \\ s_1 & -c_1 & 0 & 0 \\ c_1(c_2 a_2 + a_1) & s_1(c_2 a_2 + a_1) & s_2 a_2 + d_1 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_{xw} \\ r_{21} & r_{22} & r_{23} & p_{yw} \\ r_{31} & r_{32} & r_{33} & p_{zw} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_4 c_5 c_6 - s_4 s_6 & -c_4 c_5 s_6 & c_4 s_5 & 0 \\ s_5 c_6 & -s_5 s_6 & -c_5 & -d_4 \\ s_4 c_5 c_6 + c_4 s_6 & -s_4 c_5 s_6 & s_4 s_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equate (1,3) and (3,3):

$$c_1 c_{23} r_{13} + s_1 c_{23} r_{23} + s_{23} r_{33} = c_4 s_5$$

$$s_1 r_{13} - c_1 r_{23} = s_4 s_5$$

$$\theta_4 = a \tan 2(s_1 r_{13} - c_1 r_{23}, c_1 c_{23} r_{13} + s_1 c_{23} r_{23} + s_{23} r_{33})$$

$$({}_4^0 T)^{-1} {}_6^0 T = {}_5^4 T {}_6^5 T =$$

$$\begin{bmatrix} c_1 c_{23} c_4 + s_1 s_4 & s_1 c_{23} c_4 - c_1 s_4 & s_{23} c_4 & 0 \\ -c_1 c_{23} s_4 + s_1 c_4 & -s_1 c_{23} s_4 - c_1 c_4 & -s_{23} s_4 & 0 \\ c_1 s_{23} & s_1 s_{23} & -c_{23} & 0 \\ c_1 [s_{23} d_4 + c_2 a_2 + a_1] & s_1 [s_{23} d_4 + c_2 a_2 + a_1] & -c_{23} d_4 + s_2 a_2 + d_1 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_{xw} \\ r_{21} & r_{22} & r_{23} & p_{yw} \\ r_{31} & r_{32} & r_{33} & p_{zw} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_5 c_6 & -c_5 s_6 & s_5 & 0 \\ s_6 & c_6 & 0 & 0 \\ -s_5 c_6 & s_5 s_6 & c_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equate (1,3) and (3,3):

$$(c_1 c_{23} c_4 + s_1 s_4) r_{13} + (s_1 c_{23} c_4 - c_1 s_4) r_{23} + s_{23} c_4 r_{33} = s_5$$

$$(c_1 s_{23}) r_{13} + (s_1 s_{23}) r_{23} - c_{23} r_{33} = c_5$$

$$\theta_5 = a \tan((c_1 c_{23} c_4 + s_1 s_4) r_{13} + (s_1 c_{23} c_4 - c_1 s_4) r_{23} + s_{23} c_4 r_{33}, (c_1 s_{23}) r_{13} + (s_1 s_{23}) r_{23} - c_{23} r_{33})$$

Equate (2,1) and (2,2):

$$(-c_1 c_{23} s_4 + s_1 c_4) r_{11} - (s_1 c_{23} s_4 + c_1 c_4) r_{21} - s_{23} s_4 r_{31} = s_6$$

$$(-c_1 c_{23} s_4 + s_1 c_4) r_{12} - (s_1 c_{23} s_4 + c_1 c_4) r_{22} - s_{23} s_4 r_{32} = c_6$$

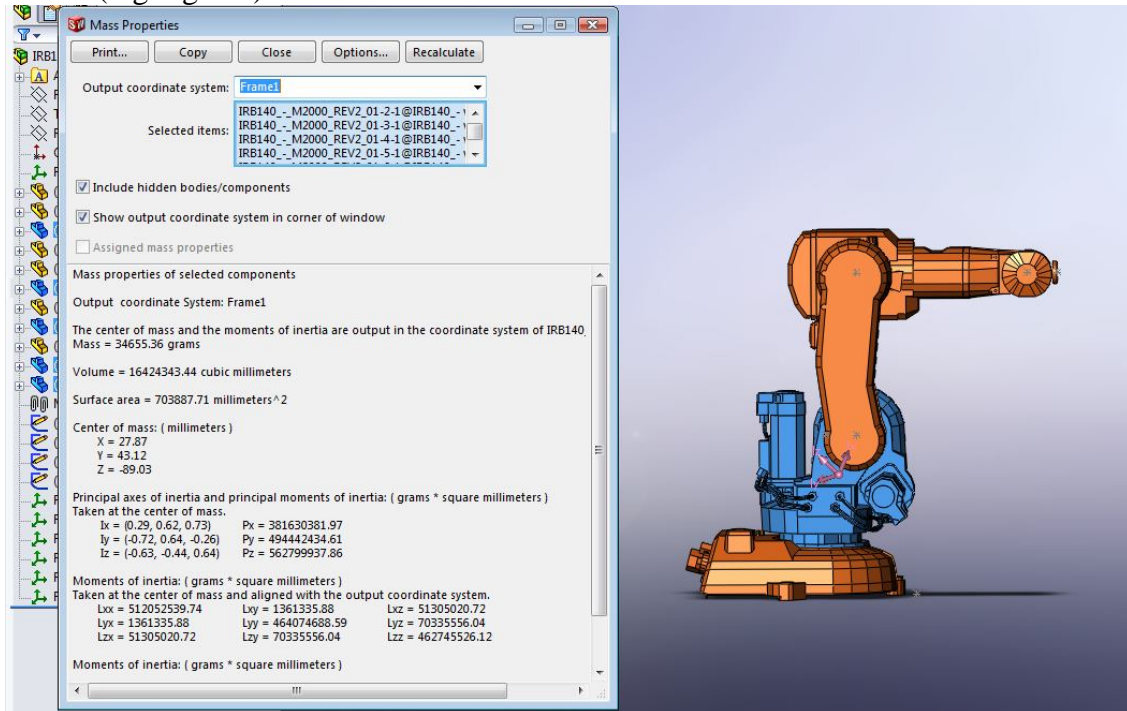
$$\theta_6 = a \tan((-c_1 c_{23} s_4 + s_1 c_4) r_{11} - (s_1 c_{23} s_4 + c_1 c_4) r_{21} - s_{23} s_4 r_{31}, (-c_1 c_{23} s_4 + s_1 c_4) r_{12} - (s_1 c_{23} s_4 + c_1 c_4) r_{22} - s_{23} s_4 r_{32})$$

Appendix C

Calculation of Inertia Matrices and Center of Mass Vectors

Using the “Mass Properties” function of the 3D CAD software SolidWorks, along with CAD files of the IRB 140 available for download from the ABB website, the inertia tensor matrices ${}^{C,i+1}I_{i+1}$, for each link $i+1$, were calculated with respect to the center of mass of that link.

Link 1 (highlighted):



The center of mass and the moments of inertia are output in the coordinate system of Frame1:

Mass = 34655.36 grams

Volume = 16424343.44 cubic millimeters

Surface area = 703887.71 millimeters²

Center of mass: (millimeters)

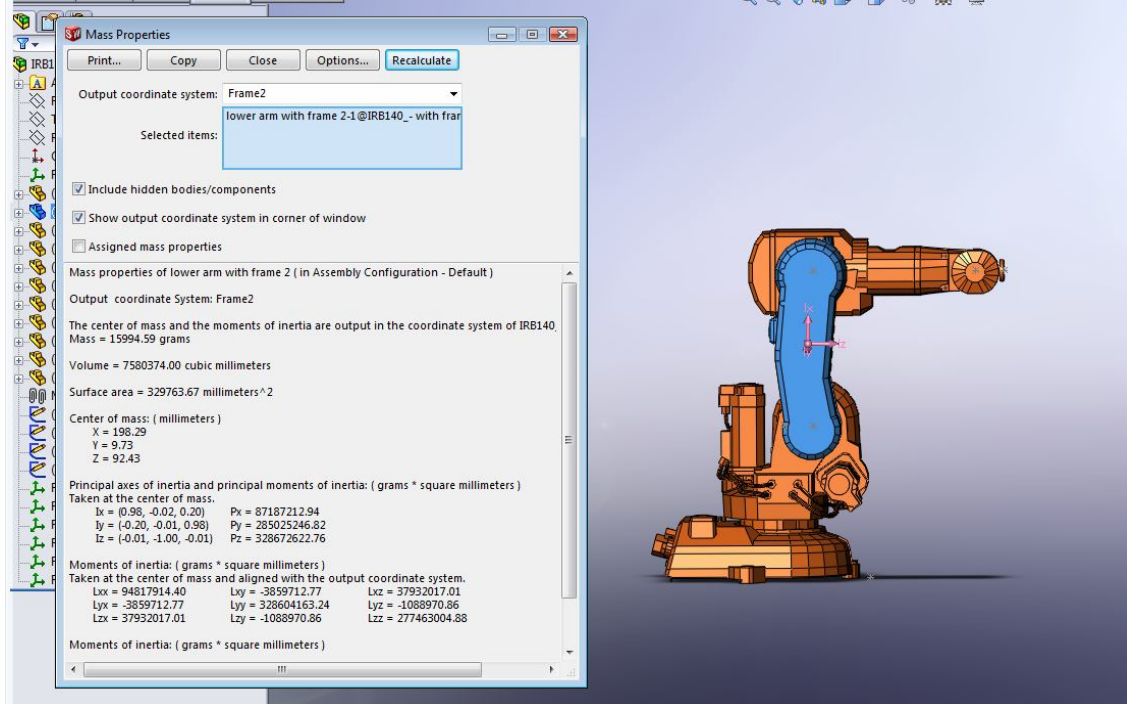
X = 27.87 Y = 43.12 Z = -89.03

Moments of inertia: (grams * square millimeters)

Taken at the center of mass and aligned with the output coordinate system.

Ixx = 512052539.74 Ixy = 1361335.88 Ixz = 51305020.72
 Iyx = 1361335.88 Iyy = 464074688.59 Iyz = 70335556.04
 Izx = 51305020.72 Izy = 70335556.04 Izz = 462745526.12

Link 2 (highlighted):



The center of mass and the moments of inertia are output in the coordinate system of Frame 2:

Mass = 15994.59 grams

Volume = 7580374.00 cubic millimeters

Surface area = 329763.67 millimeters²

Center of mass: (millimeters)

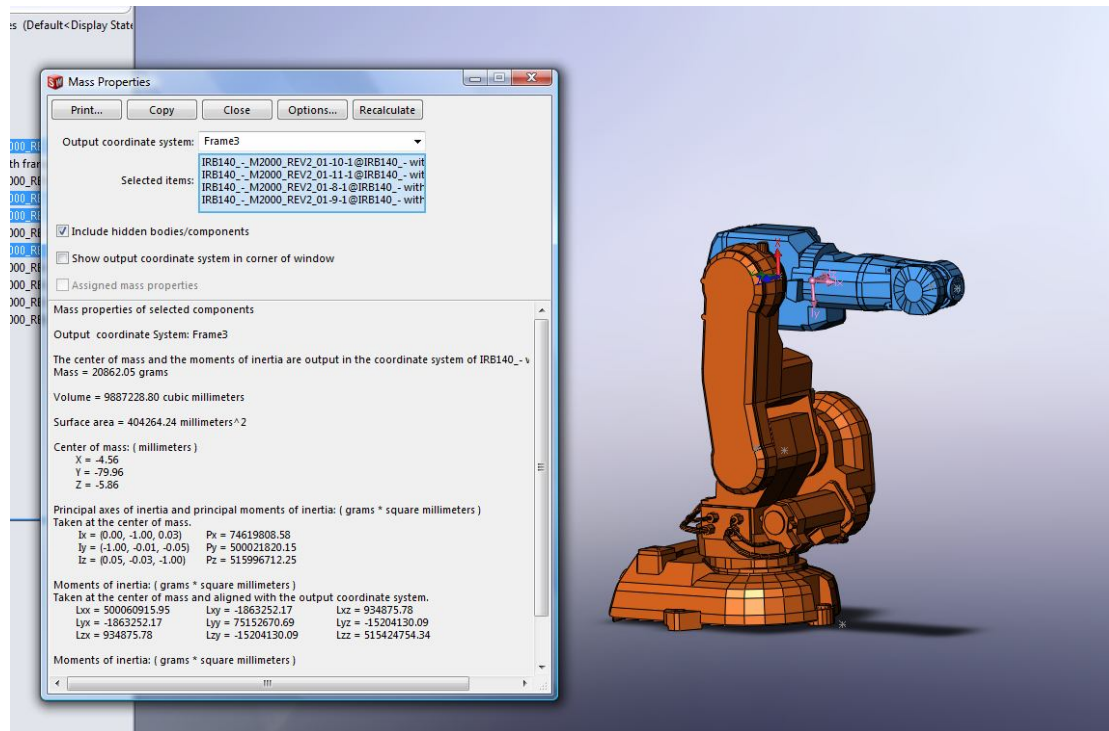
X = 198.29 Y = 9.73 Z = 92.43

Moments of inertia: (grams * square millimeters)

Taken at the center of mass and aligned with the output coordinate system.

Ixx = 94817914.40 Ixy = -3859712.77 Ixz = 37932017.01
 Iyx = -3859712.77 Iyy = 328604163.24 Iyz = -1088970.86
 Izx = 37932017.01 Izy = -1088970.86 Izz = 277463004.88

Link 3 (highlighted, including remaining links to be assumed as part of link 3):



The center of mass and the moments of inertia are output in the coordinate system of Frame 3:

Mass = 20862.05 grams

Volume = 9887228.80 cubic millimeters

Surface area = 404264.24 millimeters²

Center of mass: (millimeters)

X = -4.56 Y = -79.96 Z = -5.86

Moments of inertia: (grams * square millimeters)

Taken at the center of mass and aligned with the output coordinate system.

Ixx = 500060915.95 Ixy = -1863252.17 Ixz = 934875.78
 Iyx = -1863252.17 Iyy = 75152670.69 Iyz = -15204130.09
 Izx = 934875.78 Izy = -15204130.09 Izz = 515424754.34

Appendix D

Jacobian Matrix Derivations

(Using another method to validate, Assuming looking at only first 3 joints)

$$J_{6 \times 3} = \begin{bmatrix} J_{L1} & J_{L2} & J_{L3} \\ J_{A1} & J_{A2} & J_{A3} \end{bmatrix}$$

Calculating the J_{Ai} :

$$J_{A1} = \hat{b}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J_{A2} = \hat{b}_2 = \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix}$$

$$J_{A3} = \hat{b}_3 = \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix}$$

Calculating the r vectors (to be used to calculate J_{Li})

$$\bar{r}_{0,3} - \bar{r}_{0,i} = \bar{r}_{i,3}$$

$$\bar{r}_{1,3} = \begin{bmatrix} p_{xw} \\ p_{yw} \\ p_{zw} - d_1 \end{bmatrix} = \begin{bmatrix} c_1[s_{23}d_4 + c_2a_2 + a_1] \\ s_1[s_{23}d_4 + c_2a_2 + a_1] \\ -c_{23}d_4 + s_2a_2 \end{bmatrix}$$

$$\bar{r}_{2,3} = \begin{bmatrix} p_{xw} - c_1a_1 \\ p_{yw} - s_1a_1 \\ p_{zw} - d_1 \end{bmatrix} = \begin{bmatrix} c_1[s_{23}d_4 + c_2a_2] \\ s_1[s_{23}d_4 + c_2a_2] \\ -c_{23}d_4 + s_2a_2 \end{bmatrix}$$

Calculating the J_{Li} :

$$\begin{aligned}
 J_{L1} = \hat{b}_1 \times \bar{r}_{1,3} &= \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} c_1[s_{23}d_4 + c_2a_2 + a_1] \\ s_1[s_{23}d_4 + c_2a_2 + a_1] \\ -c_{23}d_4 + s_2a_2 \end{bmatrix} = \begin{bmatrix} -s_1[s_{23}d_4 + c_2a_2 + a_1] \\ c_1[s_{23}d_4 + c_2a_2 + a_1] \\ 0 \end{bmatrix} \\
 J_{L2} = \hat{b}_2 \times \bar{r}_{2,3} &= \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix} \times \begin{bmatrix} c_1[s_{23}d_4 + c_2a_2] \\ s_1[s_{23}d_4 + c_2a_2] \\ -c_{23}d_4 + s_2a_2 \end{bmatrix} = \begin{bmatrix} -c_1(-c_{23}d_4 + s_2a_2) \\ -s_1(-c_{23}d_4 + s_2a_2) \\ s_1^2[s_{23}d_4 + c_2a_2] + c_1^2[s_{23}d_4 + c_2a_2] \end{bmatrix} \\
 &= \begin{bmatrix} -c_1(-c_{23}d_4 + s_2a_2) \\ -s_1(-c_{23}d_4 + s_2a_2) \\ s_{23}d_4 + c_2a_2 \end{bmatrix} \\
 J_{L3} &= \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix} \times \begin{bmatrix} c_1s_{23}d_4 \\ s_1s_{23}d_4 \\ -c_{23}d_4 \end{bmatrix} = \begin{bmatrix} c_1c_{23}d_4 \\ s_1c_{23}d_4 \\ s_{23}d_4 \end{bmatrix}
 \end{aligned}$$

Only considering linear velocity (J_{Li}) to relate joint rates to end-effector velocity:

$$J_{3 \times 3} = \begin{bmatrix} -s_1(s_{23}d_4 + c_2a_2 + a_1) & c_1(c_{23}d_4 - s_2a_2) & c_1c_{23}d_4 \\ c_1(s_{23}d_4 + c_2a_2 + a_1) & s_1(c_{23}d_4 - s_2a_2) & s_1c_{23}d_4 \\ 0 & s_{23}d_4 + c_2a_2 & s_{23}d_4 \end{bmatrix}$$

Appendix E

Sample RAPID Code

MODULE Module1

```

CONST robtarget Target_10:=[[ 506.291651245987,1.29490239543512E-
12,679.5],[0.4999999999999998,0,0.866025403784439,0],[0,0,0,0],[9E
9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Target_20:=[[ 400,-250,200],[-6.12303176911189E-
17,1,0,0],[-1,0,-3,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Target_30:=[[ 415,-200,200],[-6.12303176911189E-
17,1,0,0],[-1,0,-3,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Target_40:=[[ 450,-175,200],[-6.12303176911189E-
17,1,0,0],[-1,0,-3,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Target_50:=[[ 500,-150,200],[-6.12303176911189E-
17,1,0,0],[-1,0,-3,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Target_60:=[[ 550,-125,200],[-6.12303176911189E-
17,1,0,0],[-1,0,-3,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Target_70:=[[ 590,-100,200],[-6.12303176911189E-
17,1,0,0],[-1,0,-3,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Target_80:=[[ 640,-50,200],[-6.12303176911189E-
17,1,0,0],[-1,0,-3,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Target_90:=[[ 650,0,200],[-6.12303176911189E-
17,1,0,0],[0,0,-3,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Target_100:=[[ 640,50,200],[-6.12303176911189E-
17,1,0,0],[0,0,-2,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Target_110:=[[ 590,100,200],[-6.12303176911189E-
17,1,0,0],[0,0,-2,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Target_120:=[[ 550,125,200],[-6.12303176911189E-
17,1,0,0],[0,0,-2,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Target_130:=[[ 500,150,200],[-6.12303176911189E-
17,1,0,0],[0,0,-2,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Target_140:=[[ 450,175,200],[-6.12303176911189E-
17,1,0,0],[0,0,-2,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Target_150:=[[ 415,200,200],[-6.12303176911189E-
17,1,0,0],[0,0,-2,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Target_160:=[[ 400,250,200],[-6.12303176911189E-
17,1,0,0],[0,0,-2,0],[9E9,9E9,9E9,9E9,9E9,9E9]];

```

PROC Path_40()

```

MoveL Target_20,v1000,fine,tool0\WObj:=wobj0;
MoveL Target_30,v1000,fine,tool0\WObj:=wobj0;
MoveL Target_40,v1000,fine,tool0\WObj:=wobj0;
MoveL Target_50,v1000,fine,tool0\WObj:=wobj0;
MoveL Target_60,v1000,fine,tool0\WObj:=wobj0;
MoveL Target_70,v1000,fine,tool0\WObj:=wobj0;
MoveL Target_80,v1000,fine,tool0\WObj:=wobj0;
MoveL Target_90,v1000,fine,tool0\WObj:=wobj0;

```

```
MoveL Target_100,v1000,fine,tool0\WObj:=wobj0;  
MoveL Target_110,v1000,fine,tool0\WObj:=wobj0;  
MoveL Target_120,v1000,fine,tool0\WObj:=wobj0;  
MoveL Target_130,v1000,fine,tool0\WObj:=wobj0;  
MoveL Target_140,v1000,fine,tool0\WObj:=wobj0;  
MoveL Target_150,v1000,fine,tool0\WObj:=wobj0;  
MoveL Target_160,v1000,fine,tool0\WObj:=wobj0;
```

ENDPROC

ENDMODULE

Appendix F

Sample MATLAB Code

```

N = 10; %Number of finite time intervals
tf=2.4; %Time of movement in seconds
dt = tf/N; %Time of interval in seconds

%DH Parameters & Constants
alpha_degrees = [0,90,0,90,-90,90];
alpha = alpha_degrees * pi/180;
a1=.070; %m
a2=.360;
d1=.352;
d4=.380;
m1 = 34.65536; %kg
m2 = 15.99459;
m3 = 20.86205;
g= 9.8; %m/s^2
P_1_C1 = (1/1000)*[27.87;43.12;-89.03];
P_2_C2 = (1/1000)*[198.29;9.73;92.43];
P_3_C3 = (1/1000)*[-4.56;-79.96;-5.86];
P_0_1=(1/1000)*[0;0;352];
P_1_2=(1/1000)*[70;0;0];
P_2_3=(1/1000)*[360;0;0];

%Inertia matrix
I1 = (1/1000000000)*[512052539.74,1361335.88,51305020.72;
    1361335.88,464074688.59,70335556.04;
    51305020.72,-62720521.14,554113843.58];
I2 = (1/1000000000)*[94817914.40,-3859712.77,37932017.01;
    -3859712.77,328604163.24,-1088970.86;
    37932017.01,-1088970.86,277463004.88];
I3 = (1/1000000000)*[500060915.95,-1863252.17,934875.78;
    -1863252.17,75152670.69,-15204130.09;
    934875.78,-15204130.09,515424754.34];

%array of joint angles from RobotStudio q(theta #, iteration #)
q(1:3,1)=[-32.01;90-43.61;-22.58]* pi/180;
q(1:3,2)=[-23.96;90-45.10;-18.76]* pi/180;
q(1:3,3)=[-16.70;90-47.38;-13.12]* pi/180;
q(1:3,4)=[-10.30;90-50.51;-5.73]* pi/180;
q(1:3,5)=[-4.76;90-54.58;3.49]* pi/180;
q(1:3,6)=[0;90-59.75;14.82]* pi/180;
q(1:3,7)=[4.76;90-54.58;3.49]* pi/180;
q(1:3,8)=[10.30;90-50.51;-5.73]* pi/180;
q(1:3,9)=[16.70;90-47.38;-13.12]* pi/180;

```

```

q(1:3,10)=[23.96;90-45.10;-18.76]* pi/180;
q(1:3,11)=[32.01;90-43.61;-22.58]* pi/180;

%Iteration 1 = time 0
for i=1:(N+1)
    if i==1
        qdot(1:3,i)=[0;0;0];
        qdotdot(1:3,i)=[0;0;0];
    else
        qdot(1:3,i)= (q(:,i) - q(:,i-1))/dt;
        qdotdot(1:3,i)= (qdot(:,i) - qdot(:,i-1))/dt;
    end

    %Calculate torques using iterative process for each time interval
    R_0_1=[cos(q(1,i)), -sin(q(1,i)), 0 ;...
           sin(q(1,i))*cos(alpha(1)),cos(q(1,i))*cos(alpha(1)),...
           -sin(alpha(1));...
           sin(q(1,i))*sin(alpha(1)),cos(q(1,i))*sin(alpha(1)),...
           cos(alpha(1))];
    R_1_0=R_0_1';

    R_1_2= [cos(q(2,i)), -sin(q(2,i)), 0 ;...
           0,0, -sin(alpha(2));...
           sin(q(2,i))*sin(alpha(2)),cos(q(2,i))*sin(alpha(2)), 0];
    R_2_1=R_1_2';

    R_2_3= [cos(q(3,i)), -sin(q(3,i)), 0 ;...
           sin(q(3,i))*cos(alpha(3)),cos(q(3,i))*cos(alpha(3)),...
           -sin(alpha(3));...
           sin(q(3,i))*sin(alpha(3)),cos(q(3,i))*sin(alpha(3)),...
           cos(alpha(3))];
    R_3_2=R_2_3';

    thdot1Z=[0;0;qdot(1,i)];
    thdot2Z=[0;0;qdot(2,i)];
    thdot3Z=[0;0;qdot(3,i)];
    thdotdot1Z=[0;0;qdotdot(1,i)];
    thdotdot2Z=[0;0;qdotdot(2,i)];
    thdotdot3Z=[0;0;qdotdot(3,i)];

    w0=[0;0;0];
    wdot0=[0;0;0];
    vdot0=[0;0;g];

    %Outward Iterations
    %i=0
    w1 = R_1_0*w0 + thdot1Z;
    wdot1 = R_1_0*wdot0 + cross(R_1_0*w0,thdot1Z) + thdotdot1Z;
    vdot1 = R_1_0*(wdot0 + P_0_1 + cross(w0,cross(w0,P_0_1)) +
vdot0);
    vdotC1 = cross(wdot1,P_1_C1)+ cross(w1,cross(w1,P_1_C1))+
vdot1;
    F1 = m1*vdotC1;

```

```

N1= I1*wdot1 + cross(w1,I1*w1);

%i=1
w2 = R_2_1*w1 + thdot2Z;
wdot2 = R_2_1*wdot1 + cross(R_2_1*w1,thdot2Z) + thdotdot2Z;
vdot2 = R_2_1*(wdot1 + P_1_2 + cross(w1,cross(w1,P_1_2)) +
vdot1);
vdotC2 = cross(wdot2,P_2_C2)+ cross(w2,cross(w2,P_2_C2))+
vdot2;
F2 = m2*vdotC2;
N2= I2*wdot2 + cross(w2,I2*w2);

%i=2
w3 = R_3_2*w2 + thdot3Z;
wdot3 = R_3_2*wdot2 + cross(R_3_2*w2,thdot3Z) + thdotdot3Z;
vdot3 = R_3_2*(wdot2 + P_2_3 + cross(w2,cross(w2,P_2_3)) +
vdot2);
vdotC3 = cross(wdot3,P_3_C3)+ cross(w3,cross(w3,P_3_C3))+
vdot3;
F3 = m3*vdotC3;
N3= I3*wdot3 + cross(w3,I3*w3);

%Inward Iterations
%i=3
ff3 = F3;
nn3 = N3 + cross(P_3_C3, F3);
tao(3,i) = nn3'*[0;0;1];

%i=2
ff2 = R_2_3*ff3 + F2;
nn2 = N2 + R_2_3*nn3 + cross(P_2_C2, F2)+ cross(P_2_3,
R_2_3*ff3);
tao(2,i) = nn2'*[0;0;1];

%i=1
ff1 = R_1_2*ff2 + F1;
nn1 = N1 + R_1_2*nn2 + cross(P_1_C1, F1)+ cross(P_1_2,
R_1_2*ff2);
tao(1,i) = nn1'*[0;0;1];

%Calculate power for each joint for each time interval i
W(1,i)=abs(tao(1,i)*qdot(1,i));
W(2,i)=abs(tao(2,i)*qdot(2,i));
W(3,i)=abs(tao(3,i)*qdot(3,i));
%Calculate total power for each time interval i
Power(i)=W(1,i)+W(2,i)+W(3,i);
%Calculate energy consumed
if i>1
    RunSumEnergy(i) = RunSumEnergy(i-1) + Power(i)*dt;
else
    RunSumEnergy(i)= Power(i)*dt;
end

```

```

end

%Display Consumed Energy
RunSumEnergy(N+1)

%Display various graphs
figure(2)
x_plot=[1:N+1];
th1=q(1,1:N+1);
th2=q(2,1:N+1);
th3=q(3,1:N+1);
plot(x_plot, th1, '-', x_plot, th2, '--', x_plot, th3, ':');
legend('Theta1', 'Theta2', 'Theta3')
axis([1 N+1 -2 2])
xlabel('Iteration')
ylabel('Joint Angle (radians)')
title('Joint Angles vs Time')

figure(3)
x_plot=[1:N+1];
thdot1=qdot(1,1:N+1);
thdot2=qdot(2,1:N+1);
thdot3=qdot(3,1:N+1);
plot(x_plot, thdot1, '-', x_plot, thdot2, '--', x_plot, thdot3, ':');
legend('ThetaDot1', 'ThetaDot2', 'ThetaDot3')
axis([1 N+1 -4 4])
xlabel('Iteration')
ylabel('Joint Angle Velocity (radians/sec)')
title('Joint Angle Velocites vs Time')

figure(4)
x_plot=[1:N+1];
thdotdot1=qdotdot(1,1:N+1);
thdotdot2=qdotdot(2,1:N+1);
thdotdot3=qdotdot(3,1:N+1);
plot(x_plot, thdotdot1, '-', x_plot, thdotdot2, '--', x_plot,
thdotdot3, ':');
legend('ThetaDotDot1', 'ThetaDotDot2', 'ThetaDotDot3')
axis([1 N+1 -10 10])
xlabel('Iteration')
ylabel('Joint Angle Acceleration (radians/sec^2)')
title('Joint Angle Acceleration vs Time')

figure(5)
x_plot=[1:N+1];
T1=tao(1,1:N+1);
T2=tao(2,1:N+1);
T3=tao(3,1:N+1);
plot(x_plot, T1, '-', x_plot, T2, '--', x_plot, T3, ':');
legend('Joint 1', 'Joint 2', 'Joint 3')
axis([1 N+1 -20 160])
xlabel('Iteration')
ylabel('Joint Torque (Nm)')

```



```
title('Joint Torque vs Time')

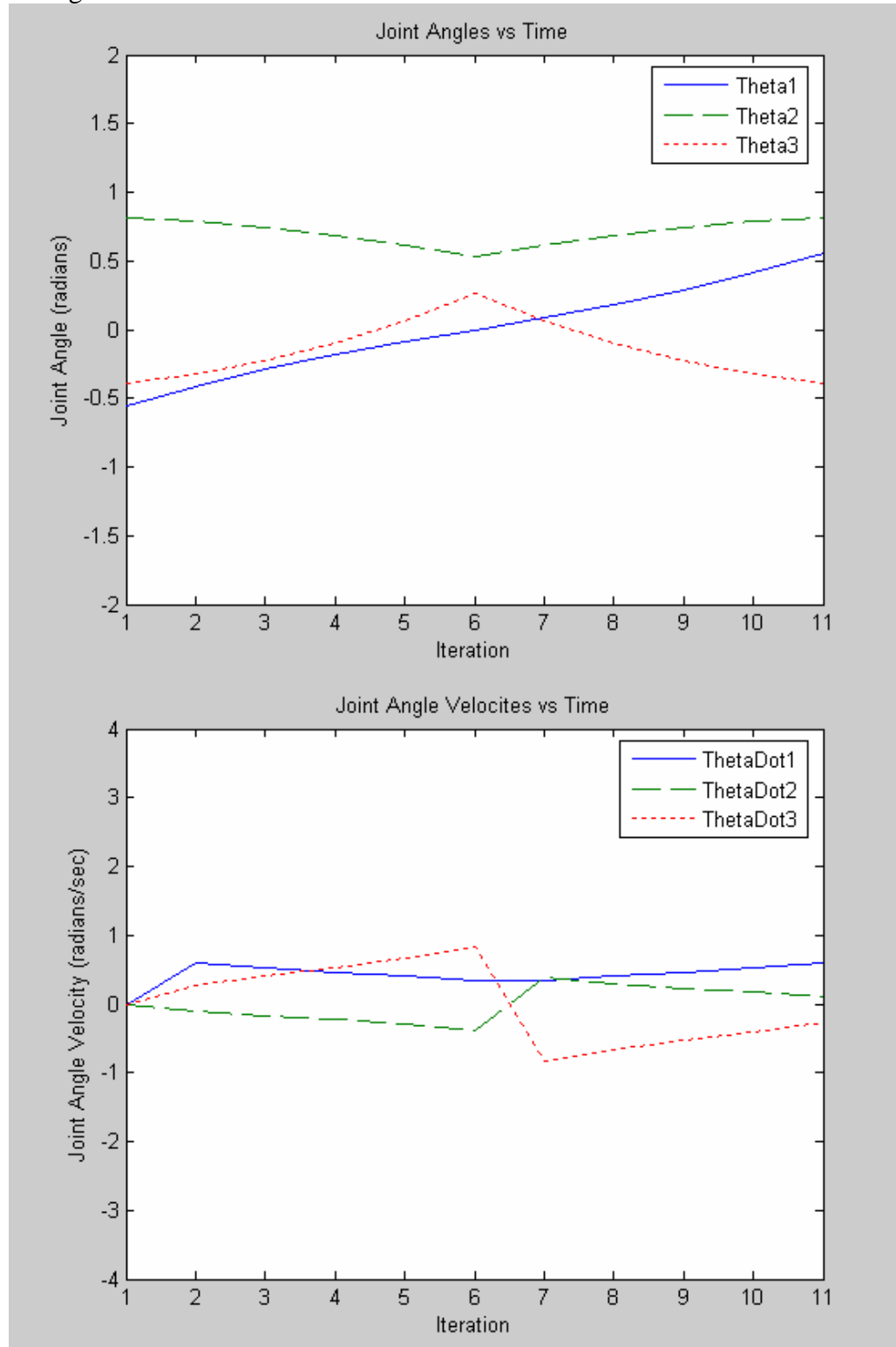
figure(6)
x_plot=[1:N+1];
Power1=W(1,1:N+1);
Power2=W(2,1:N+1);
Power3=W(3,1:N+1);
TotPower=Power(1:N+1);
plot(x_plot, Power1, '-', x_plot, Power2, '--', x_plot, Power3, ':',
x_plot, TotPower, '-.');
legend('Joint 1', 'Joint 2', 'Joint 3', 'All Joints')
axis([1 N+1 0 100])
xlabel('Iteration')
ylabel('Power (W)')
title('Power vs Time')

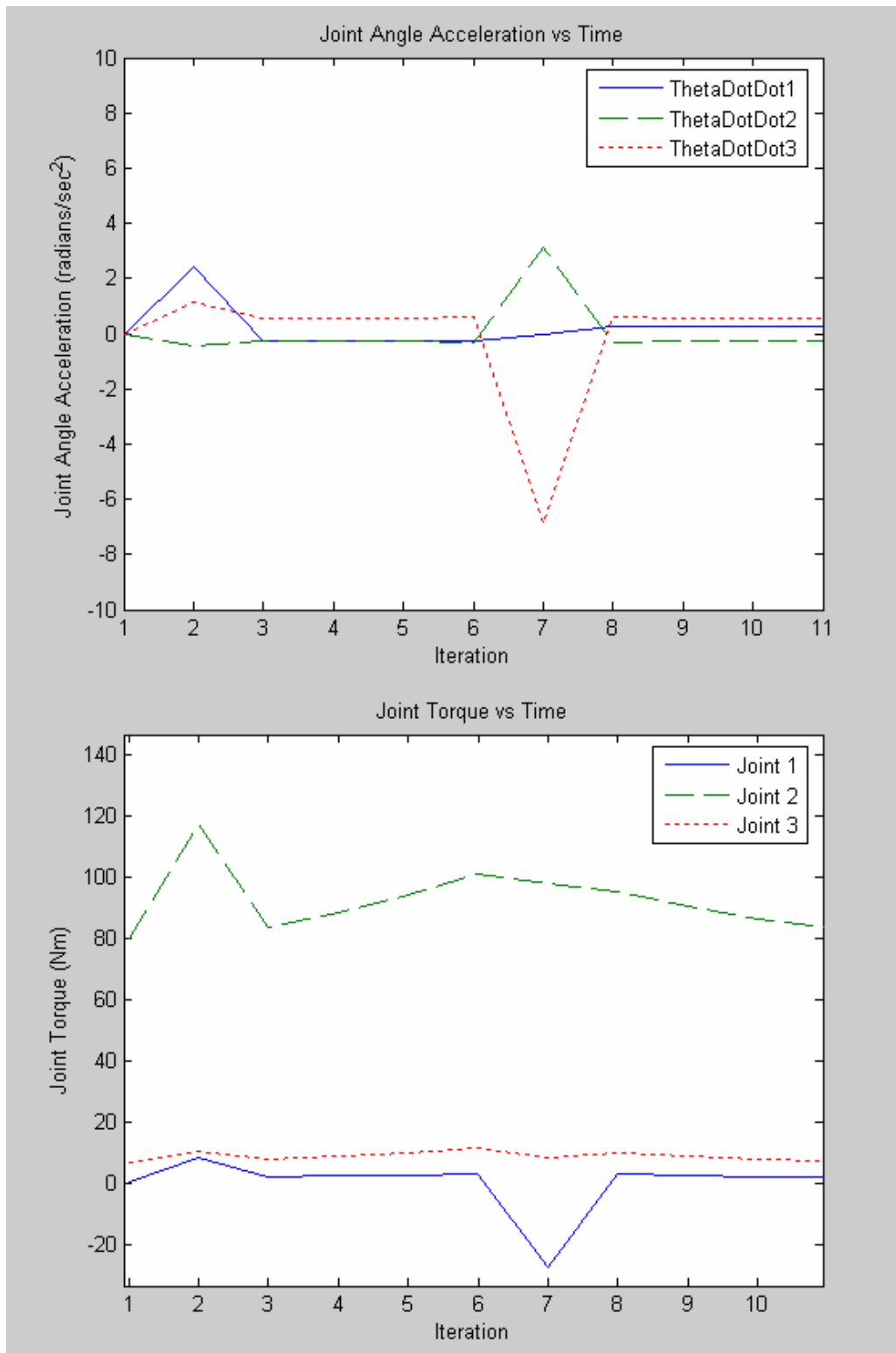
figure(7)
x_plot=[1:N+1];
Energy=RunSumEnergy(1:N+1);
plot(x_plot, Energy);
legend('Cumulative Energy')
axis([1 N+1 0 100])
xlabel('Iteration')
ylabel('Energy (J)')
title('Energy Consumption vs Time')
```

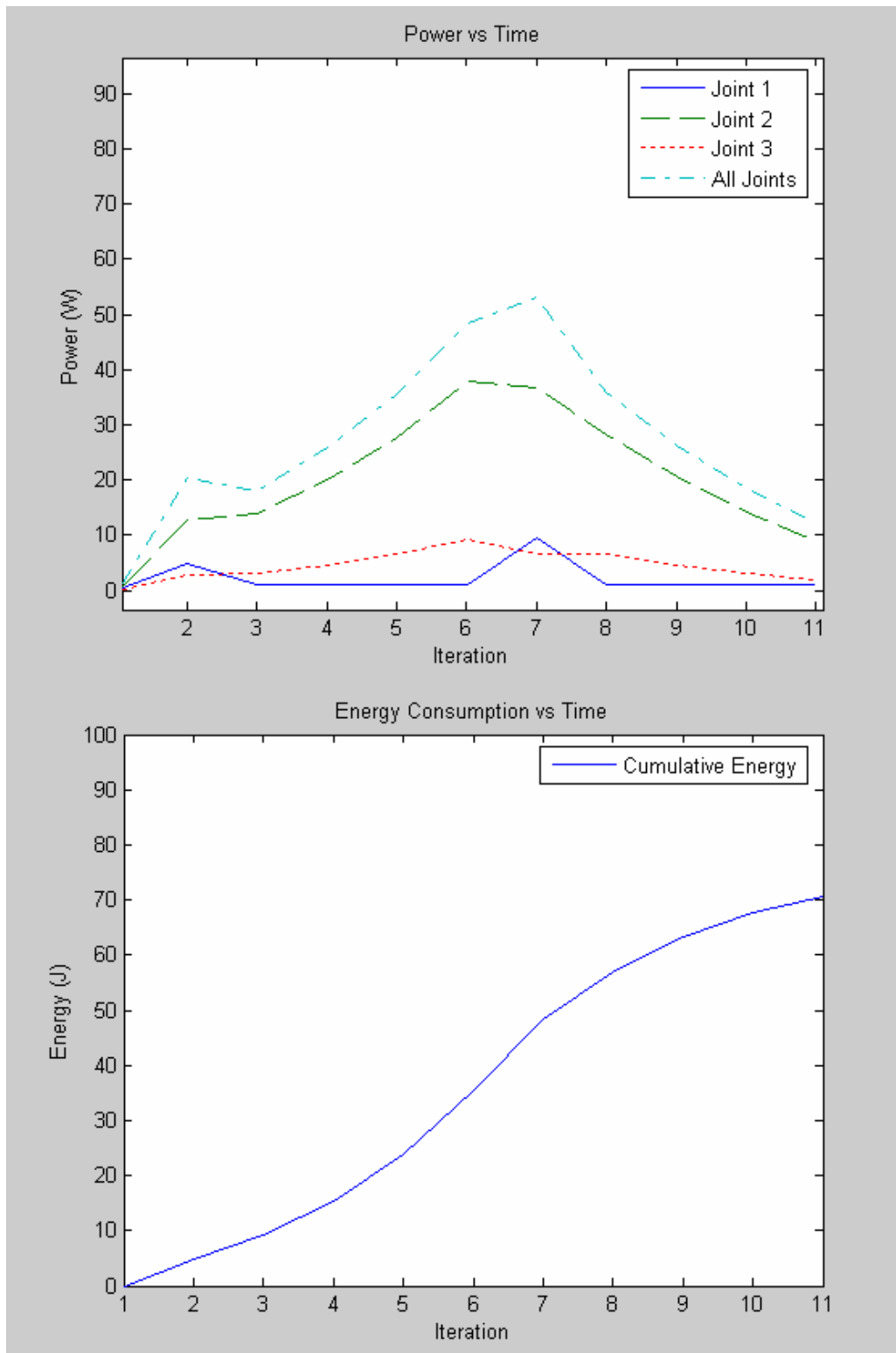
Appendix G

MATLAB Results

Triangular

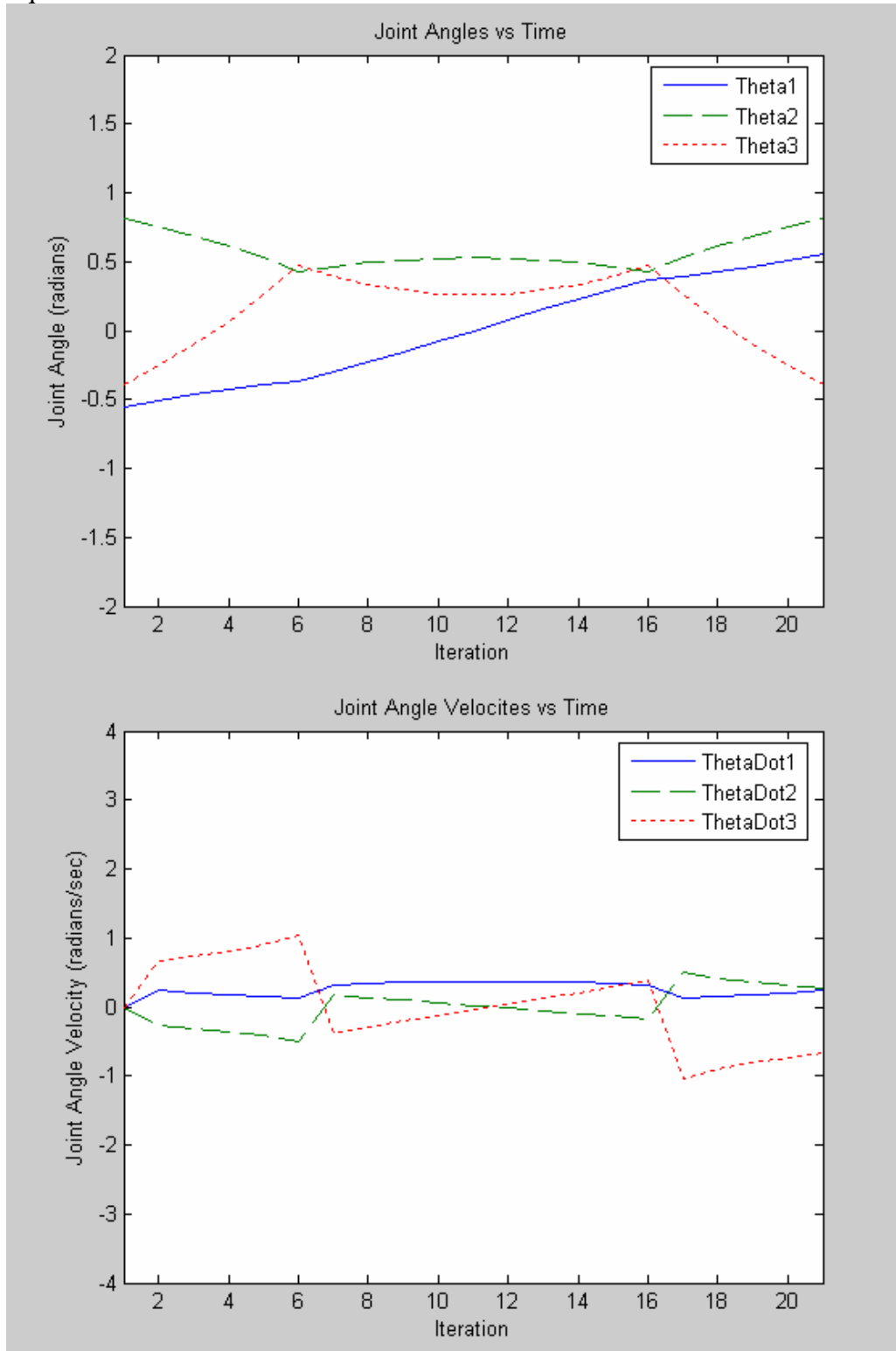


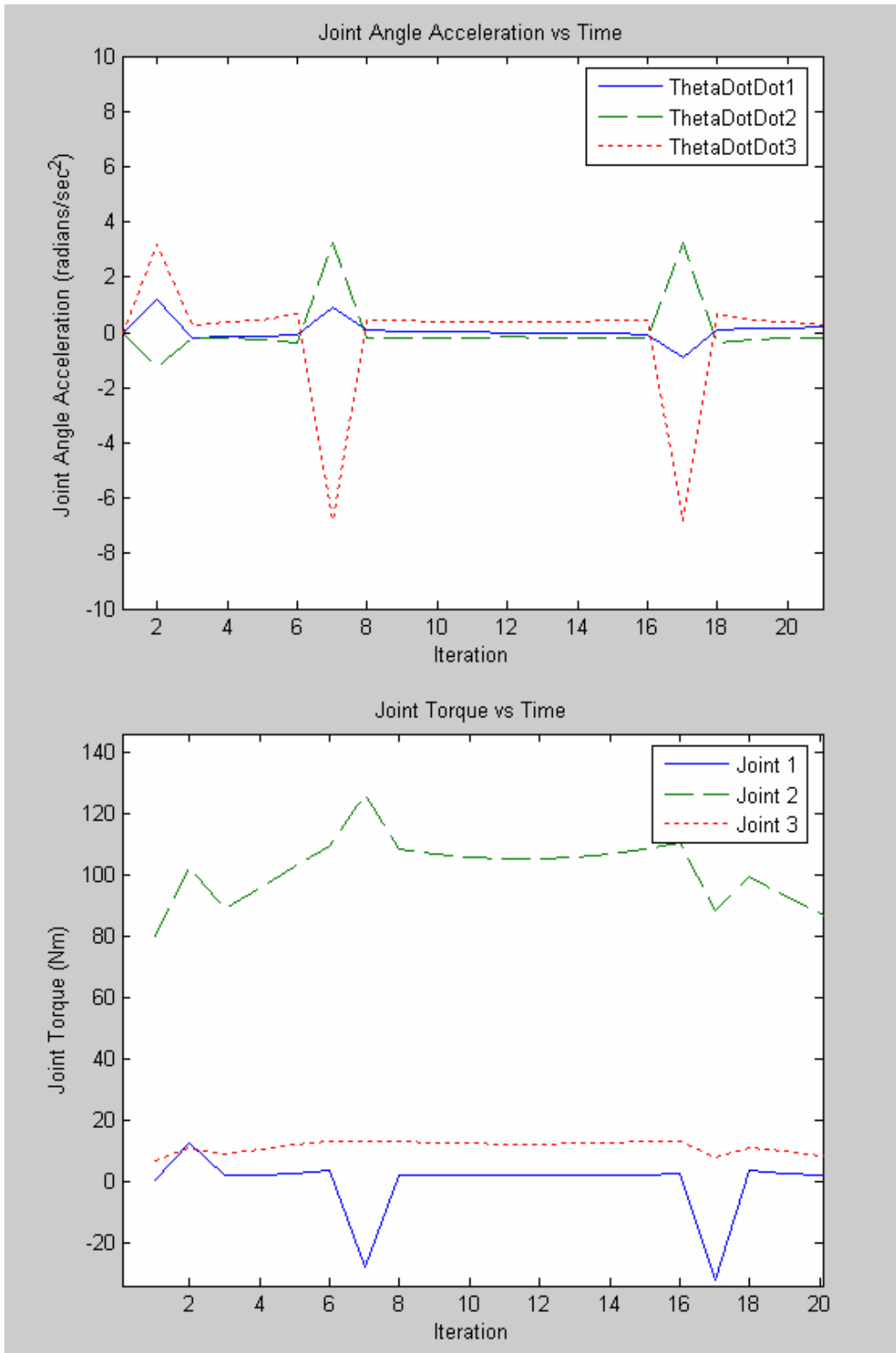


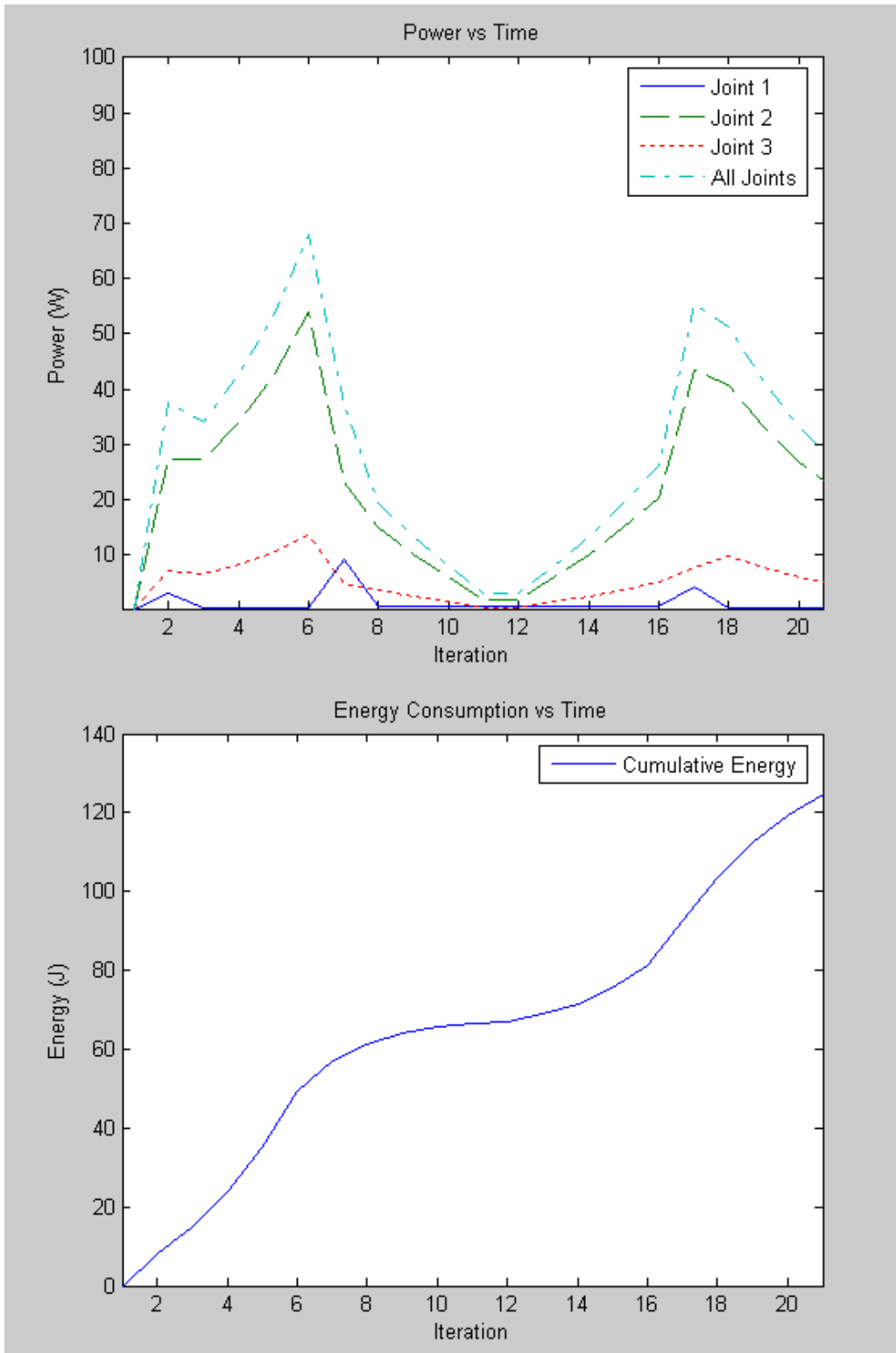


Energy consumed: 70.49 J

Square

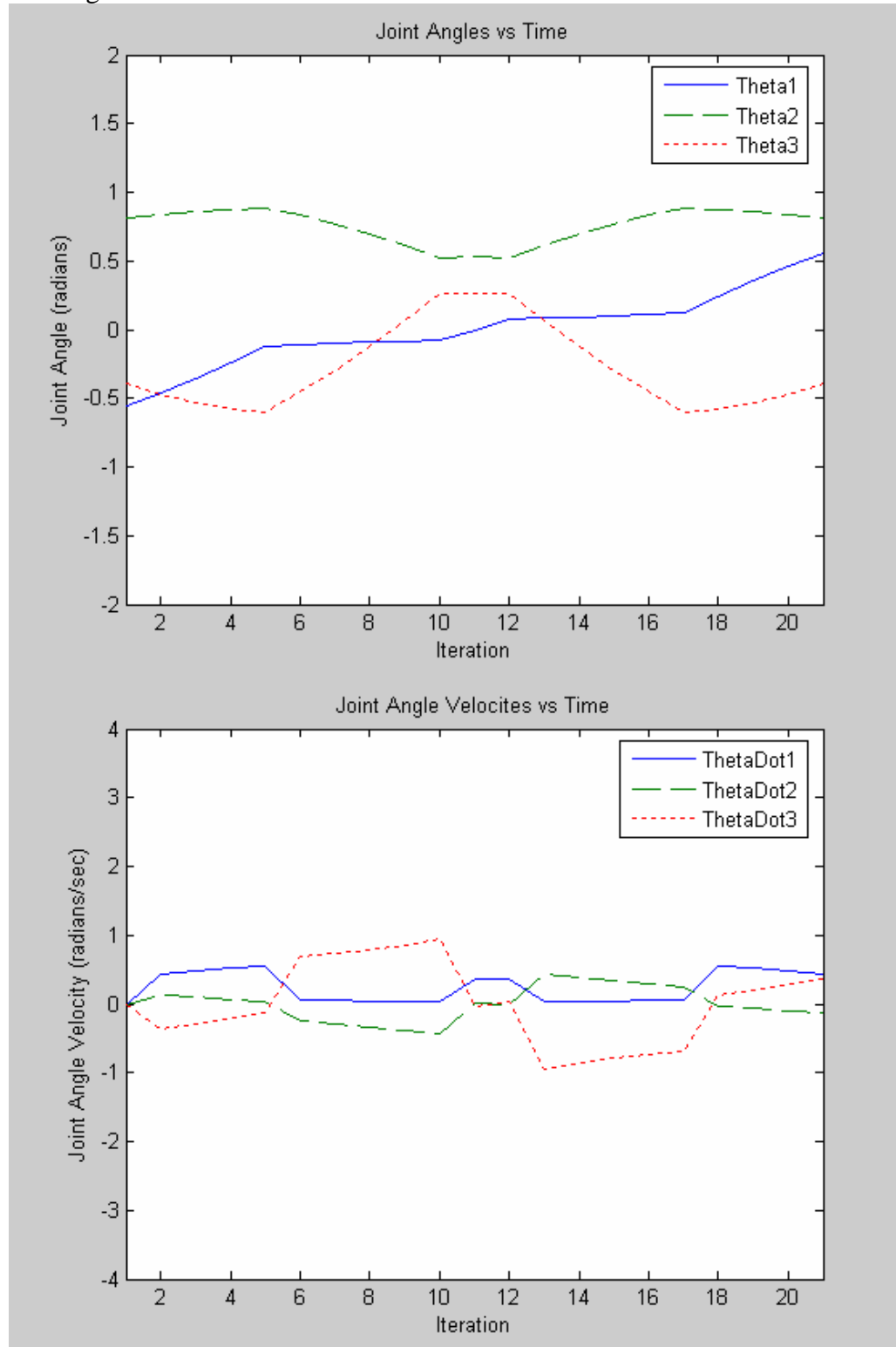


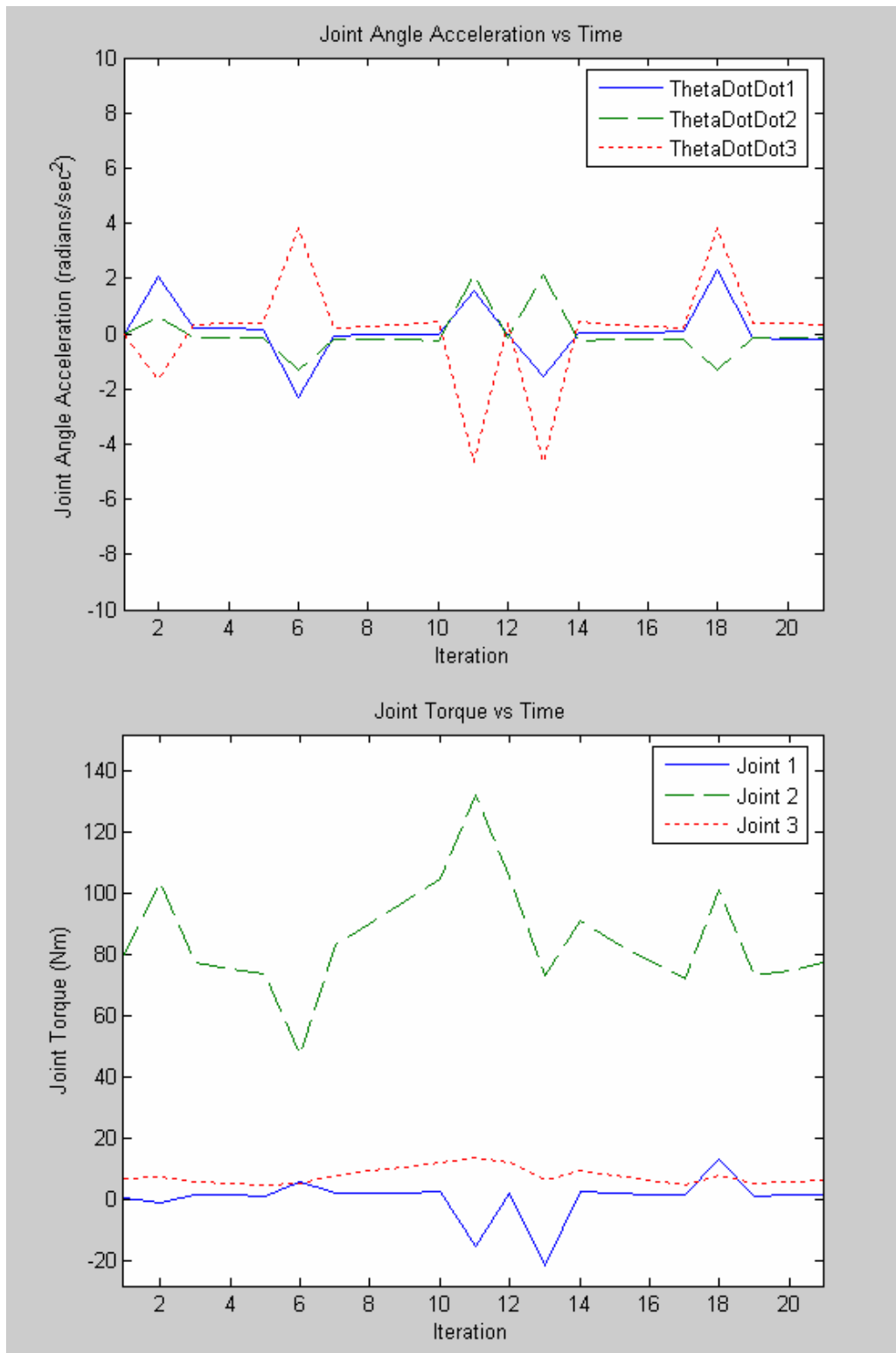


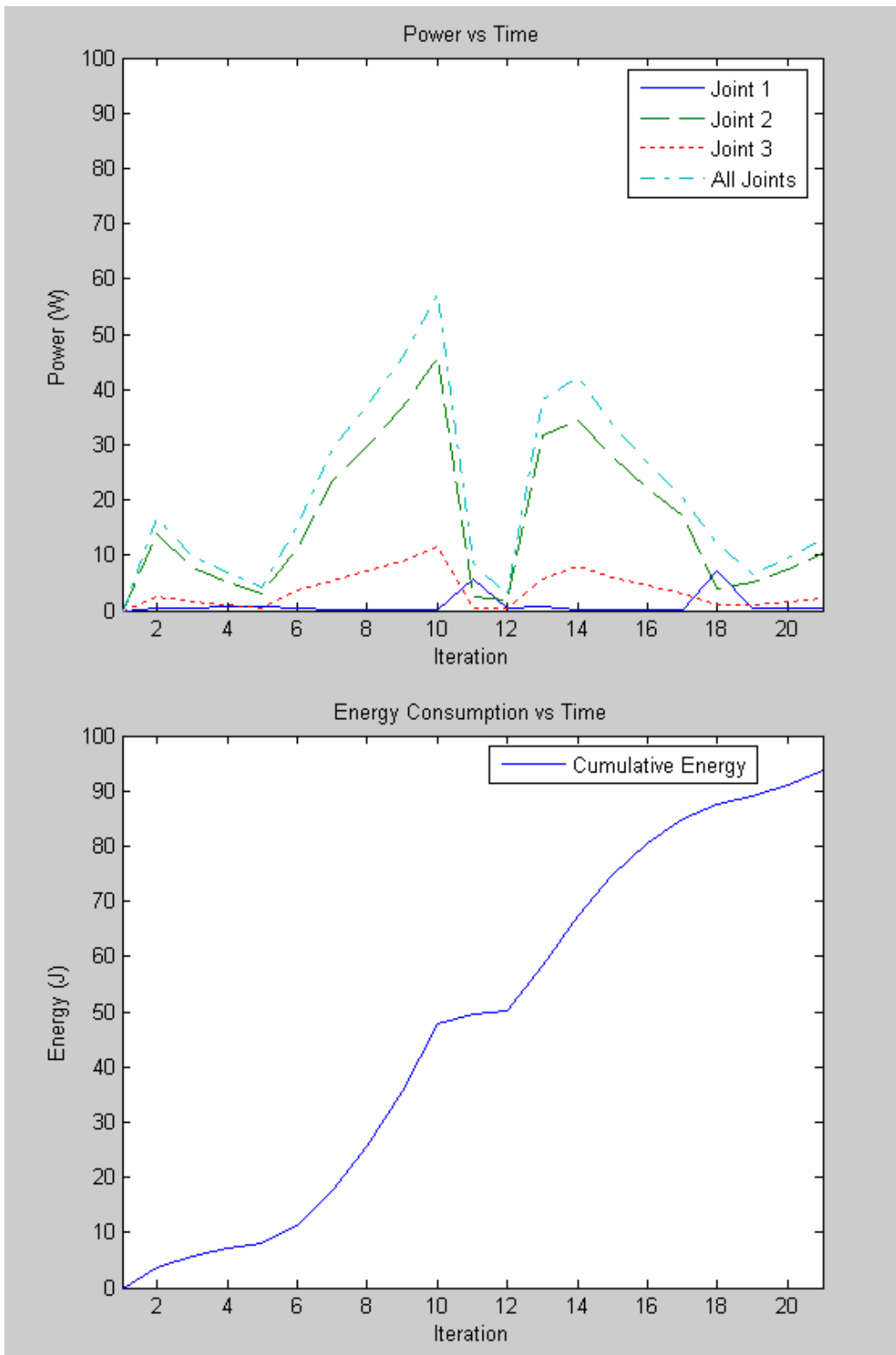


Energy consumed: 124.83 J

Rectangular

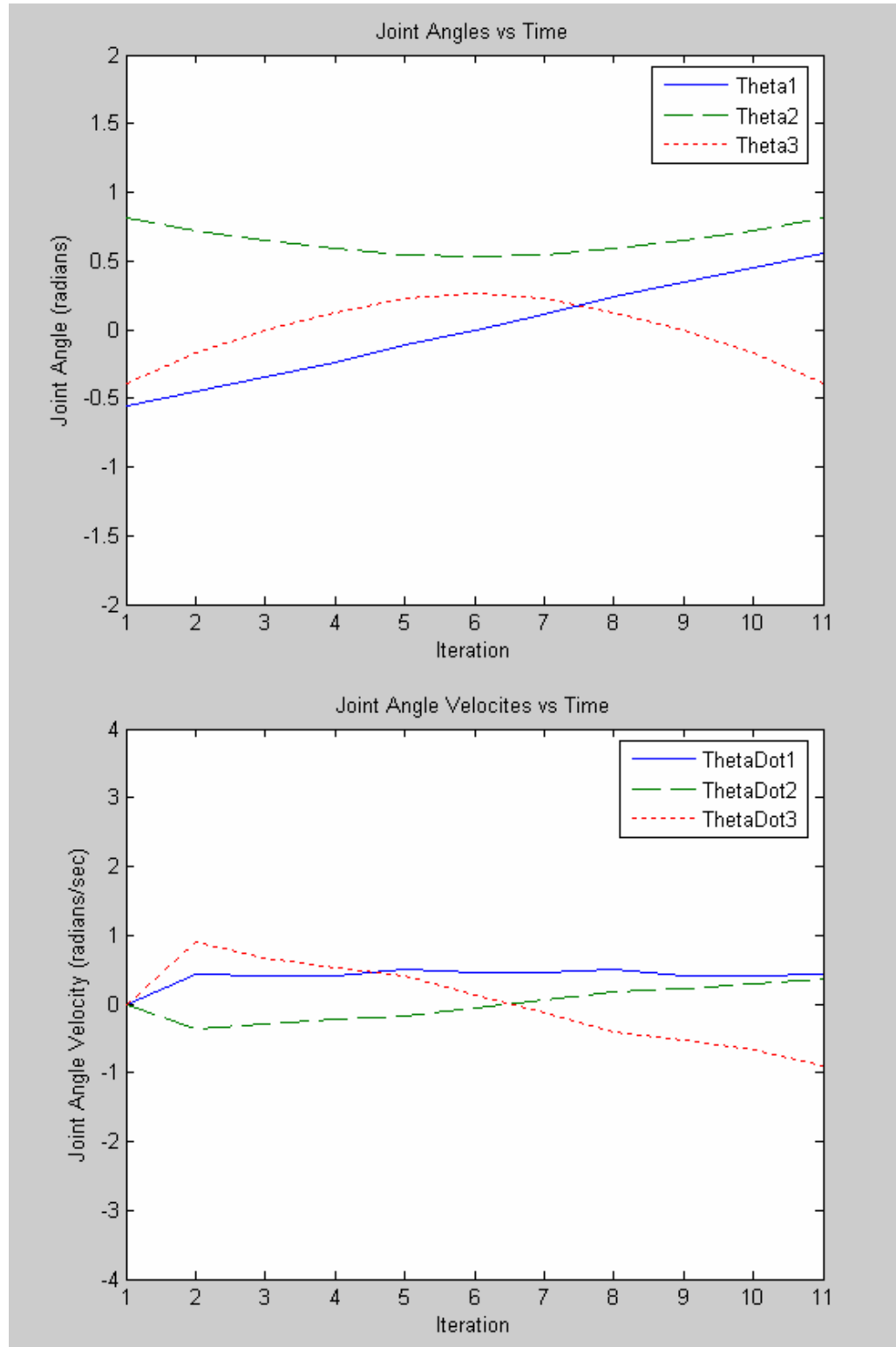


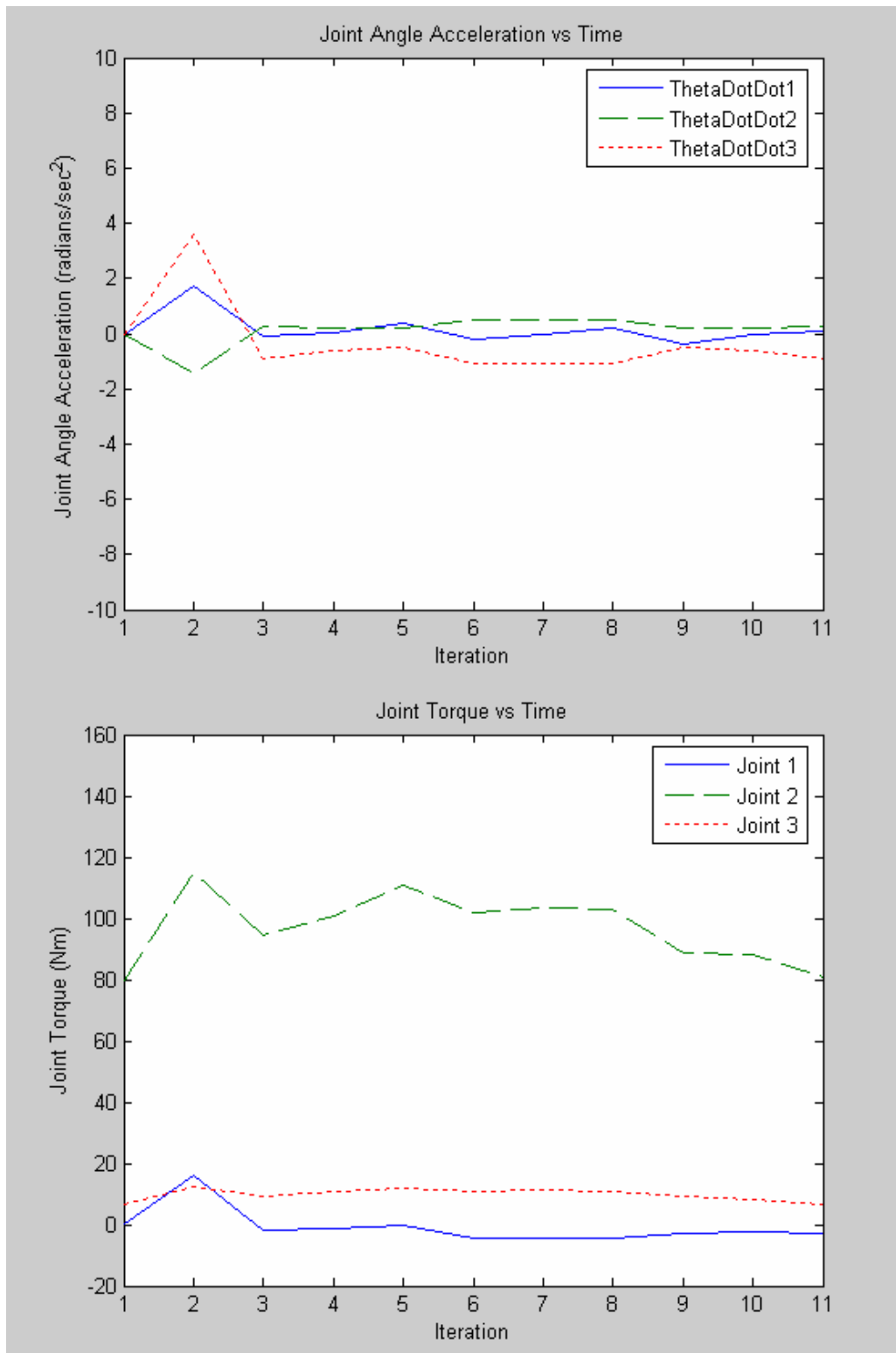


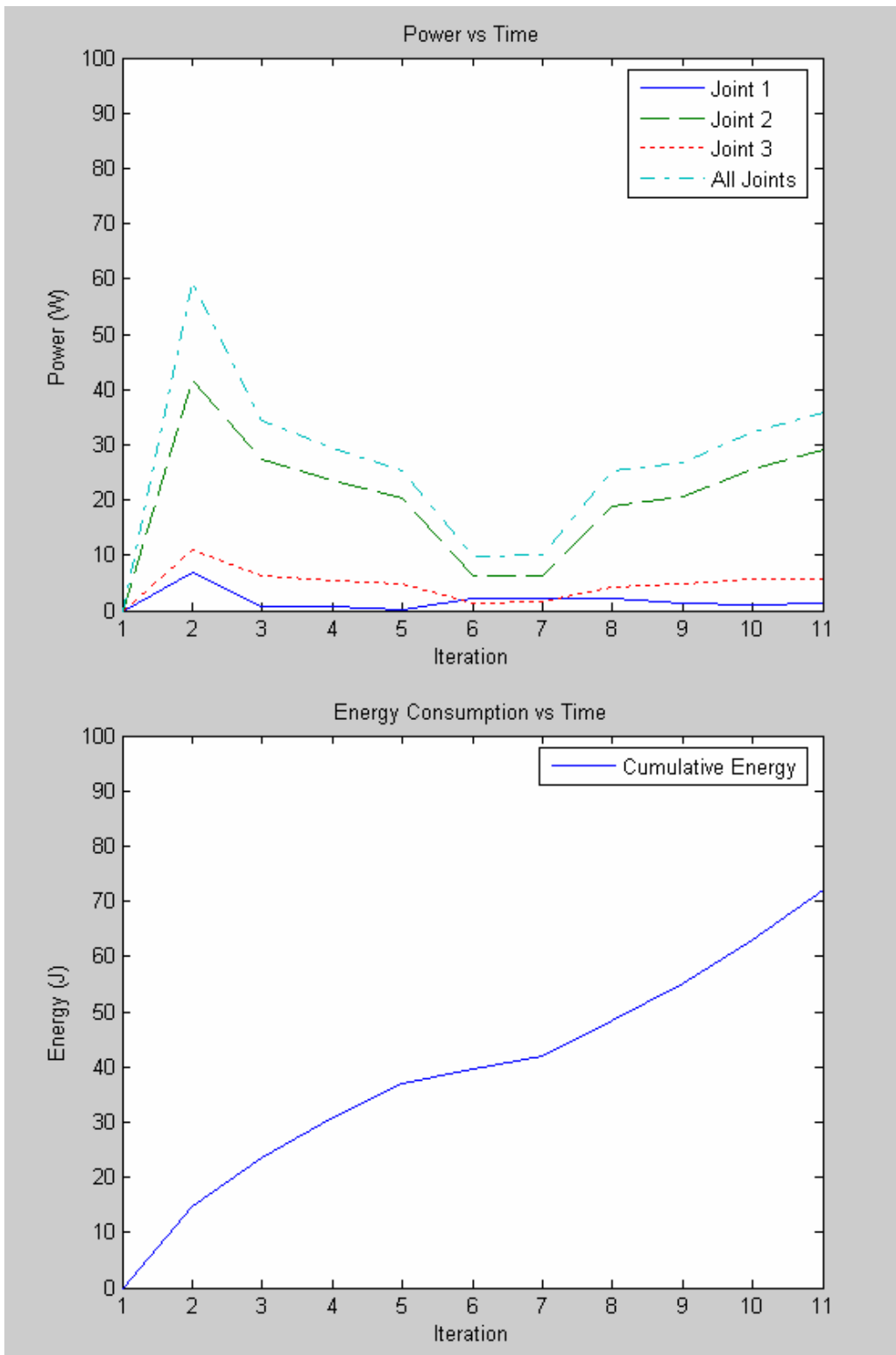


Energy consumed: 93.81 J

Semi-circle

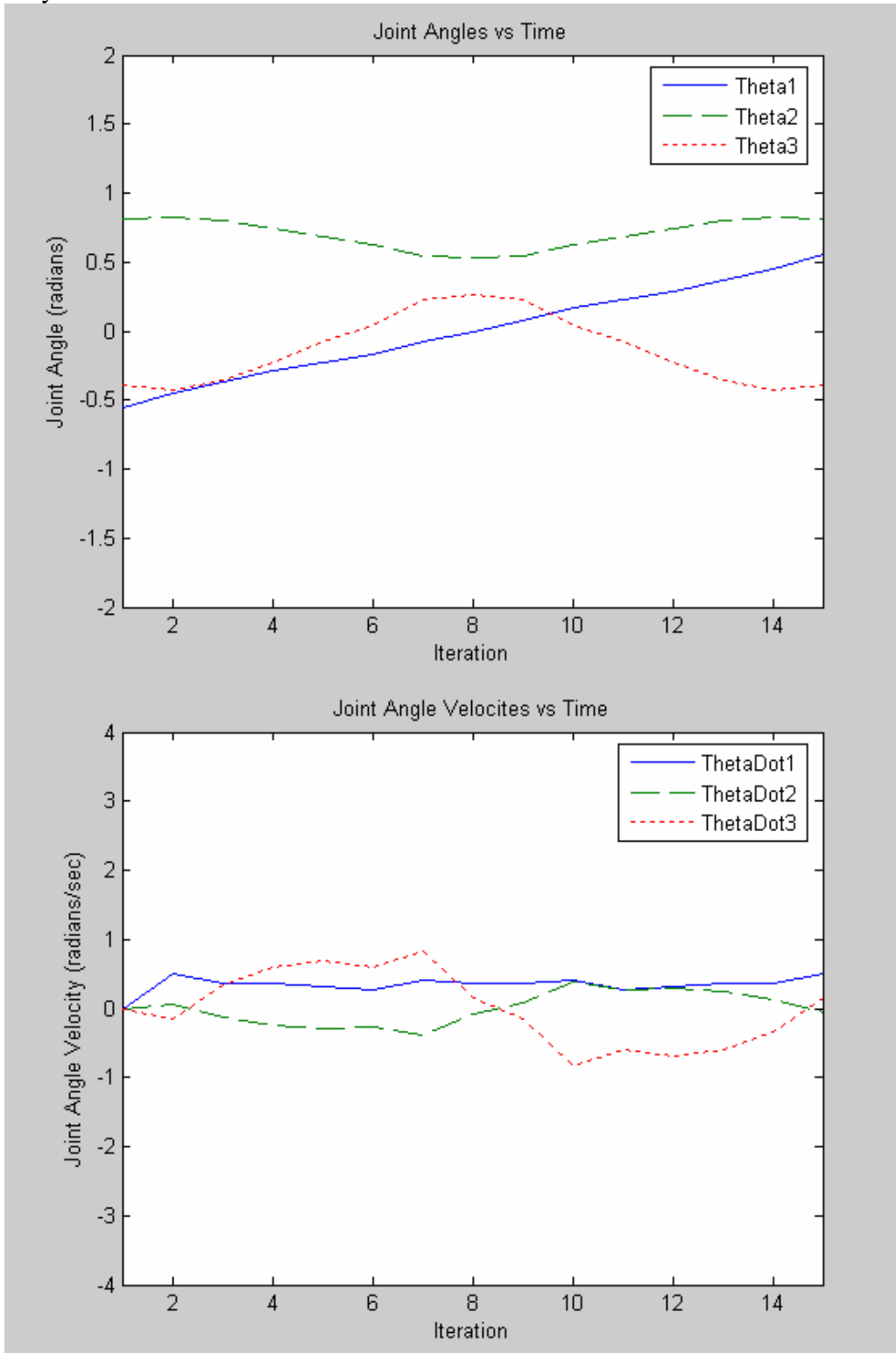


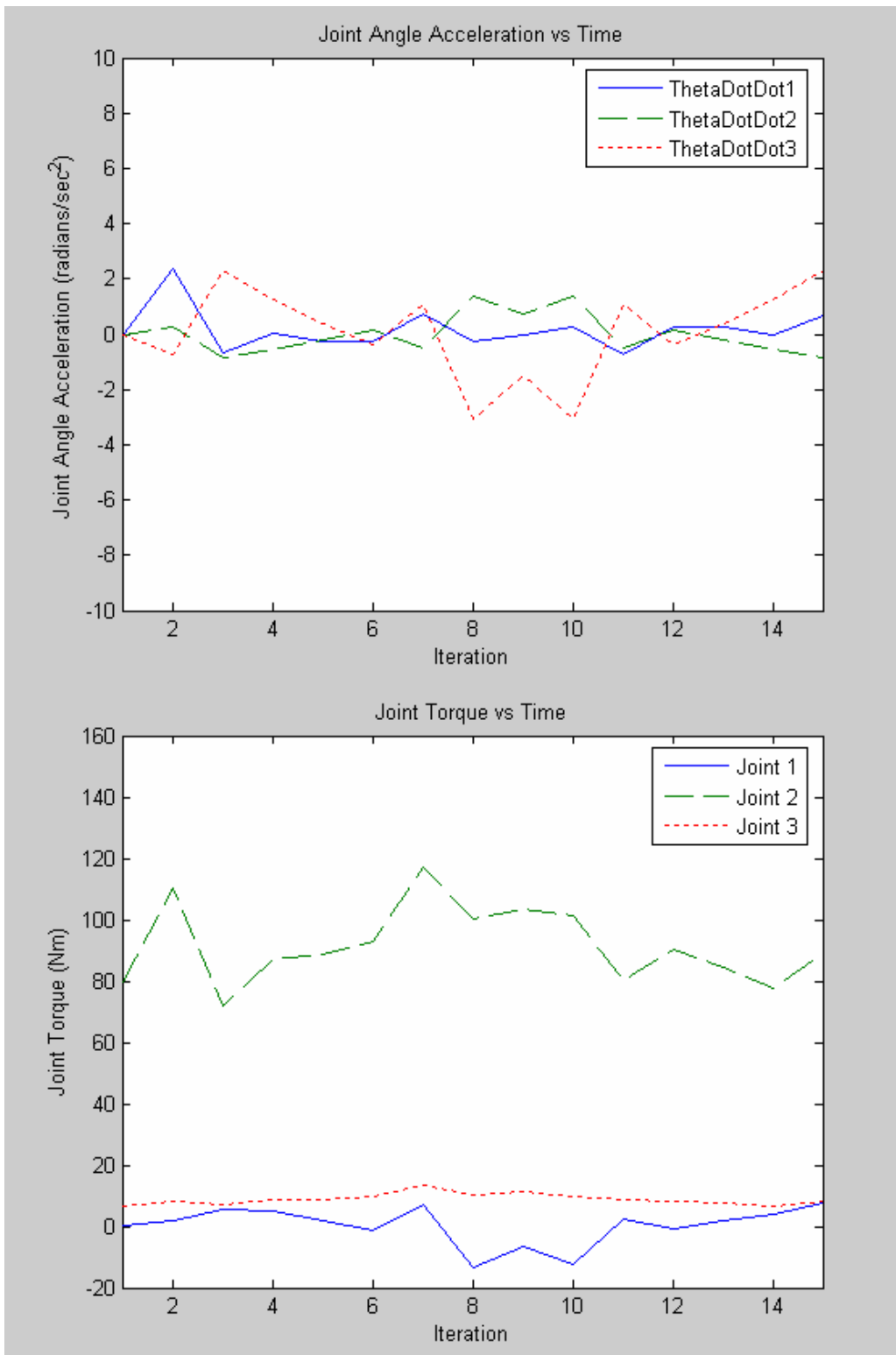


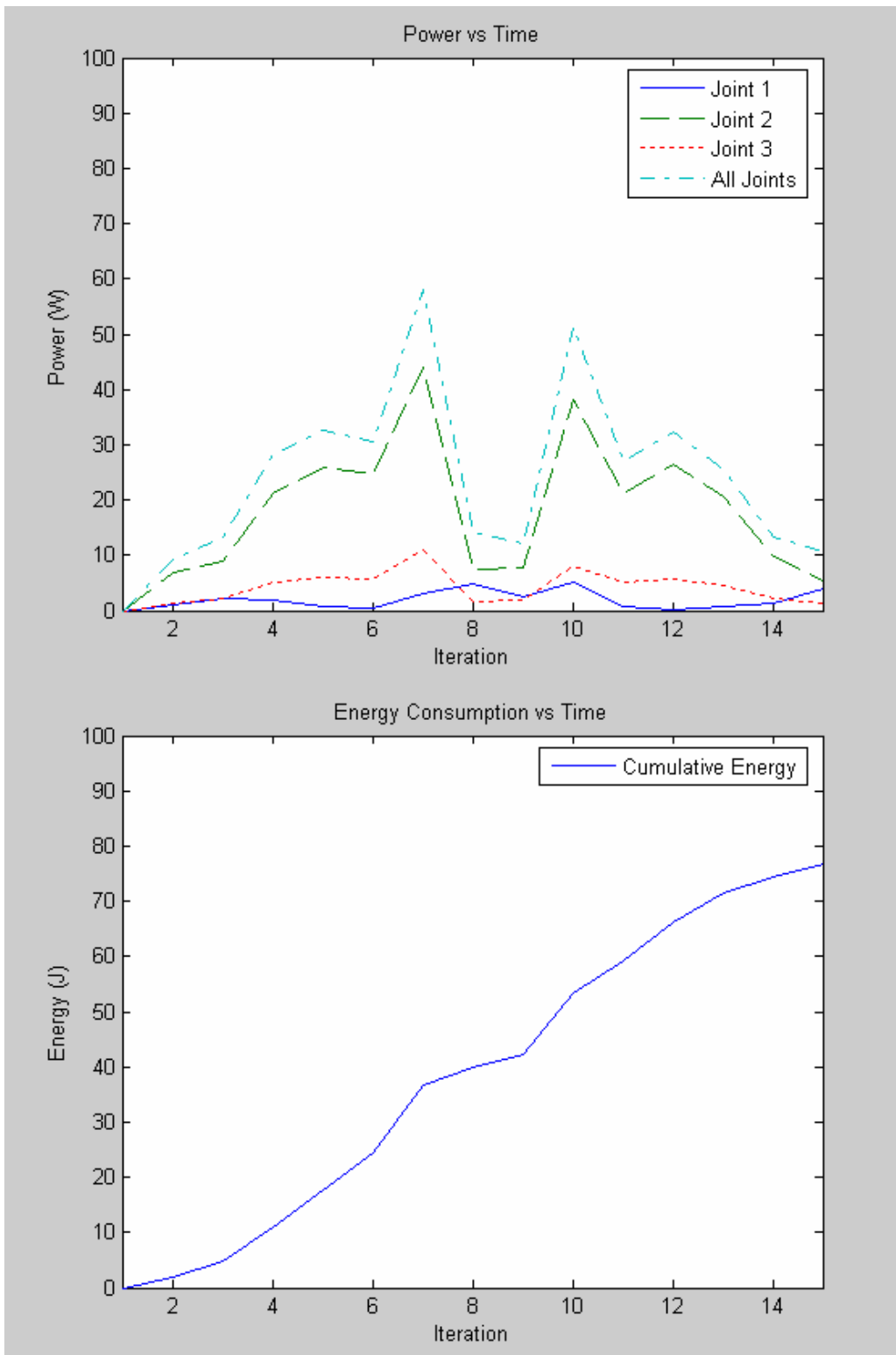


Energy consumed: 72.13 J

Polynomial – 4 DOF

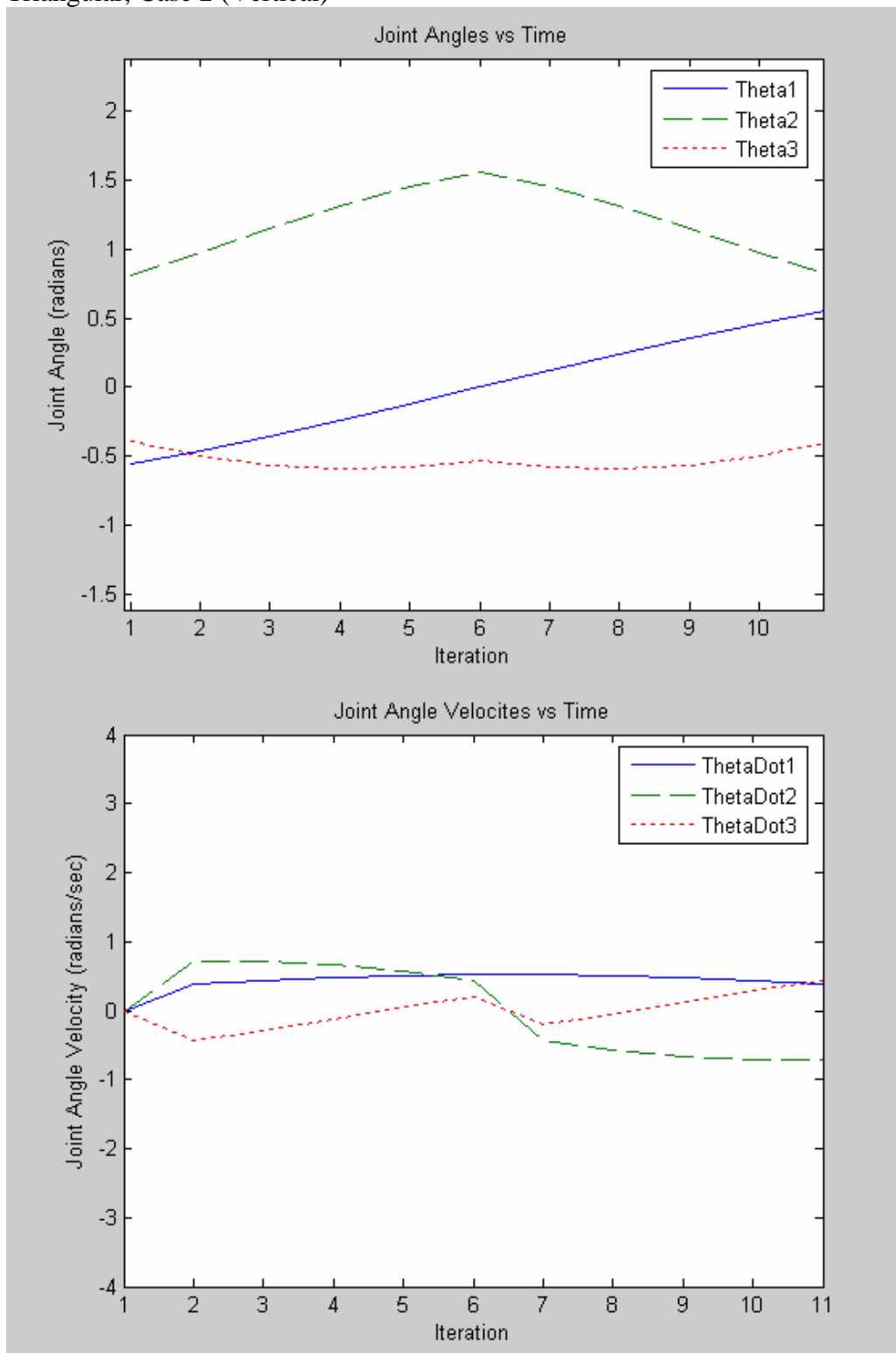


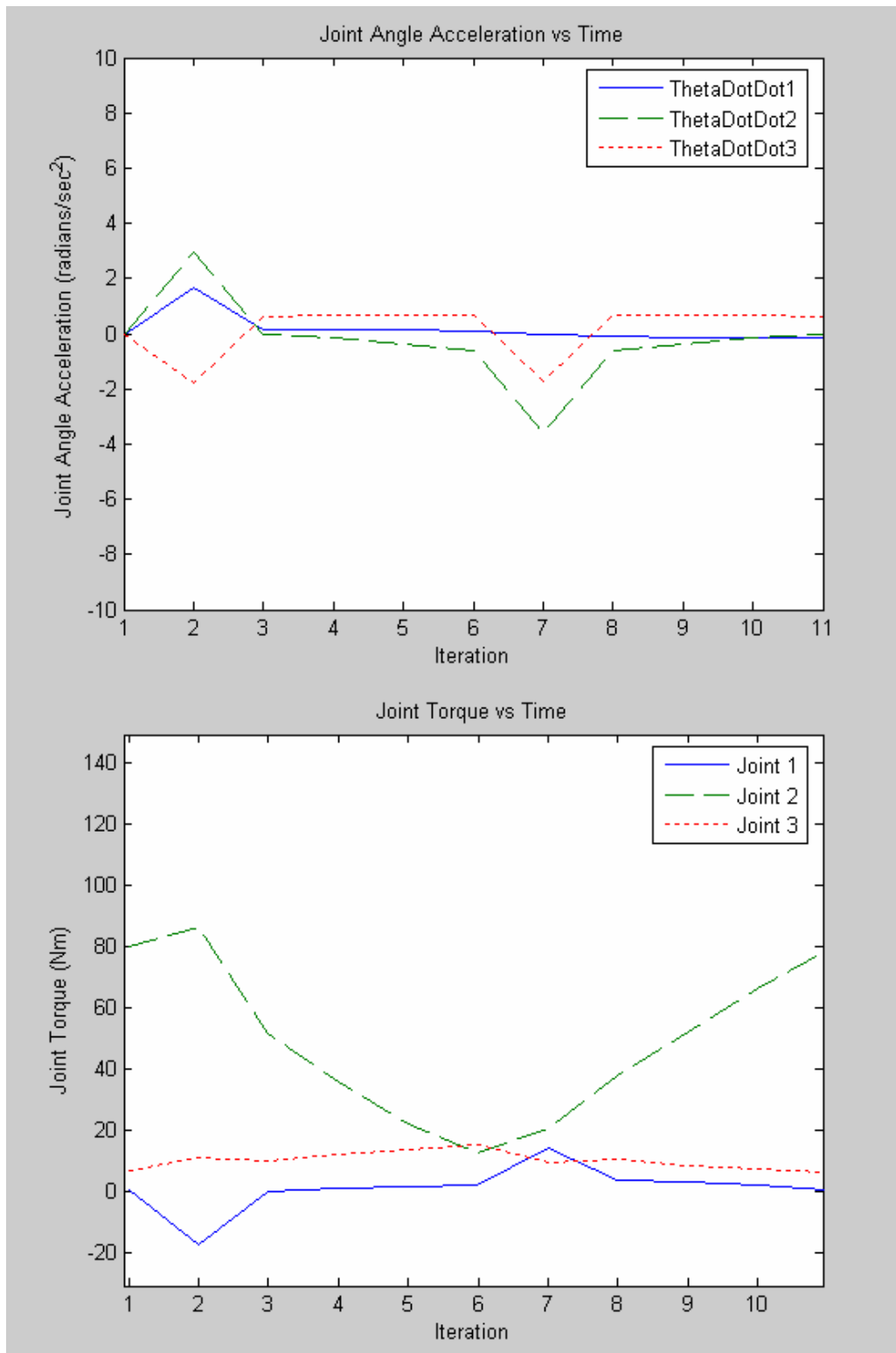


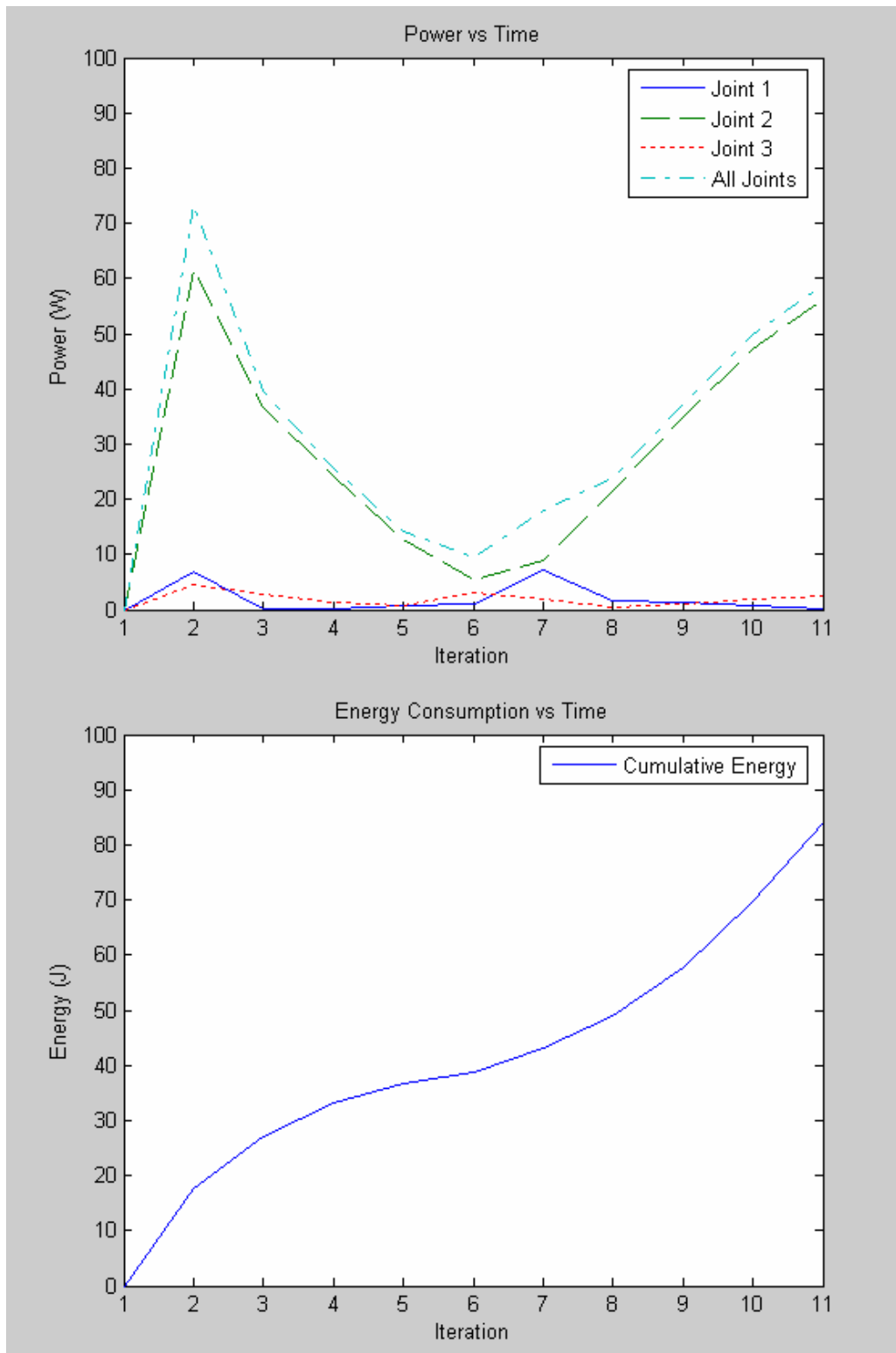


Energy consumed: 76.77 J

Triangular, Case 2 (Vertical)

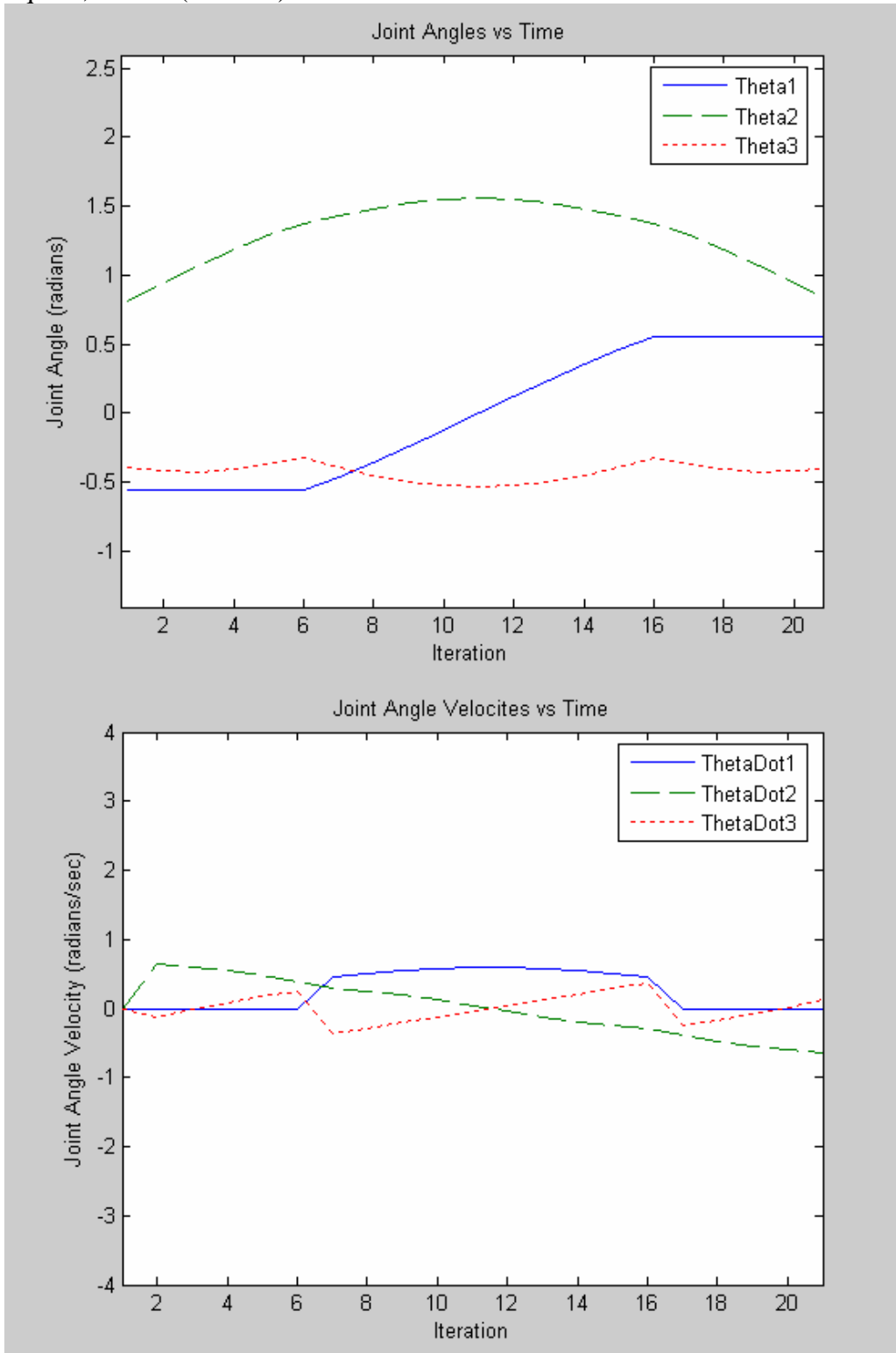


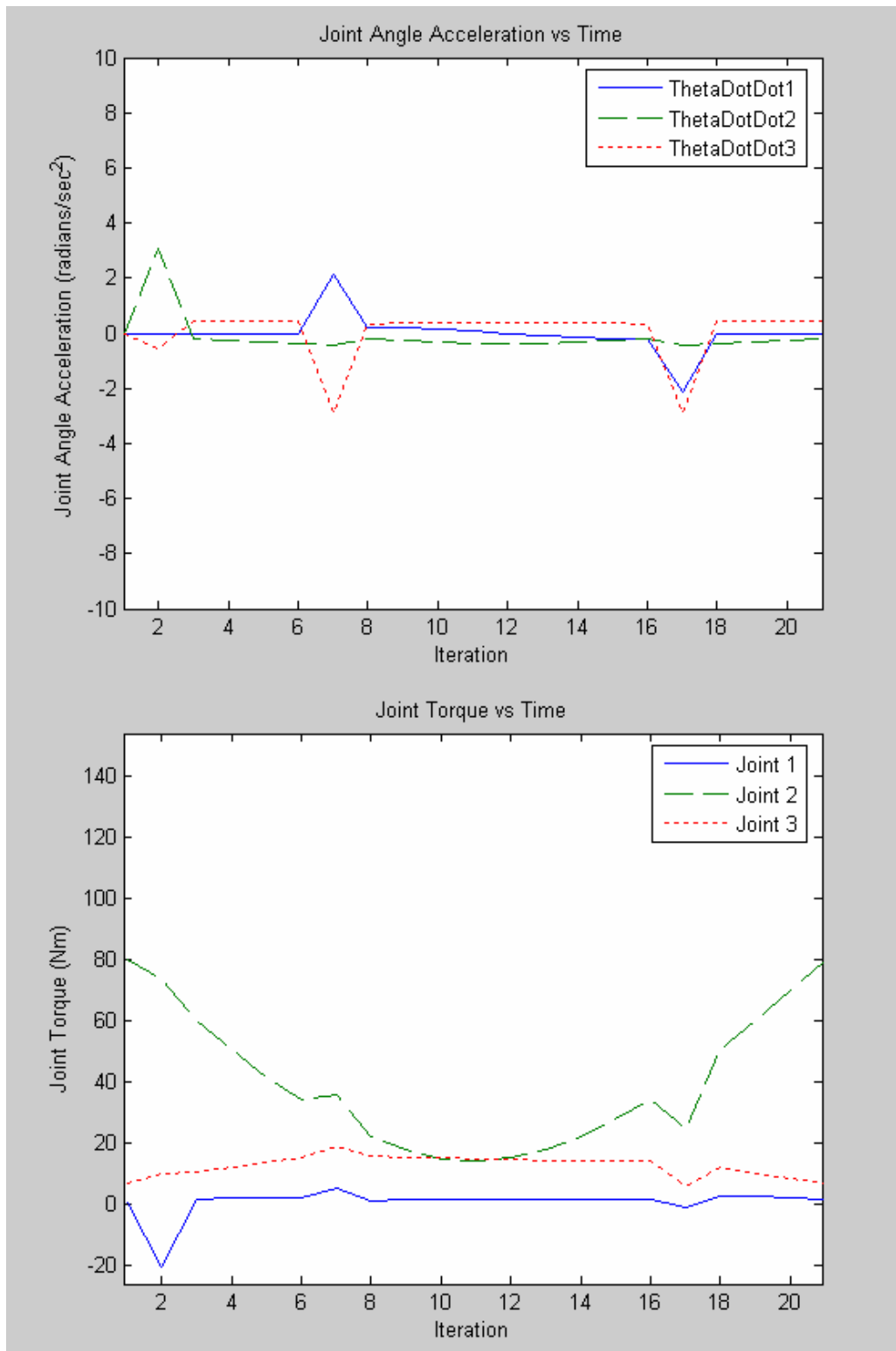


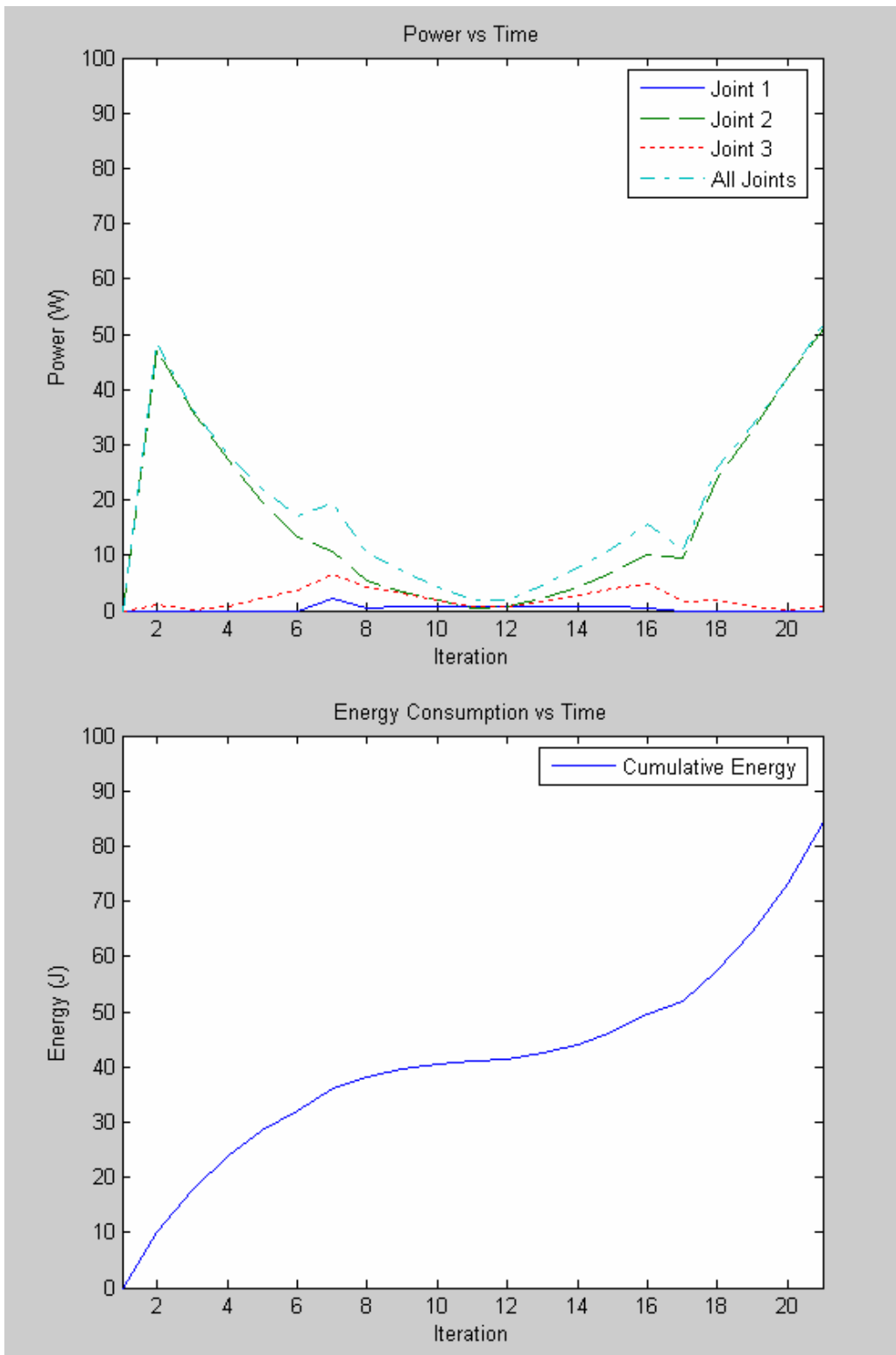


Energy consumed: 83.94 J

Square, Case 2 (Vertical)

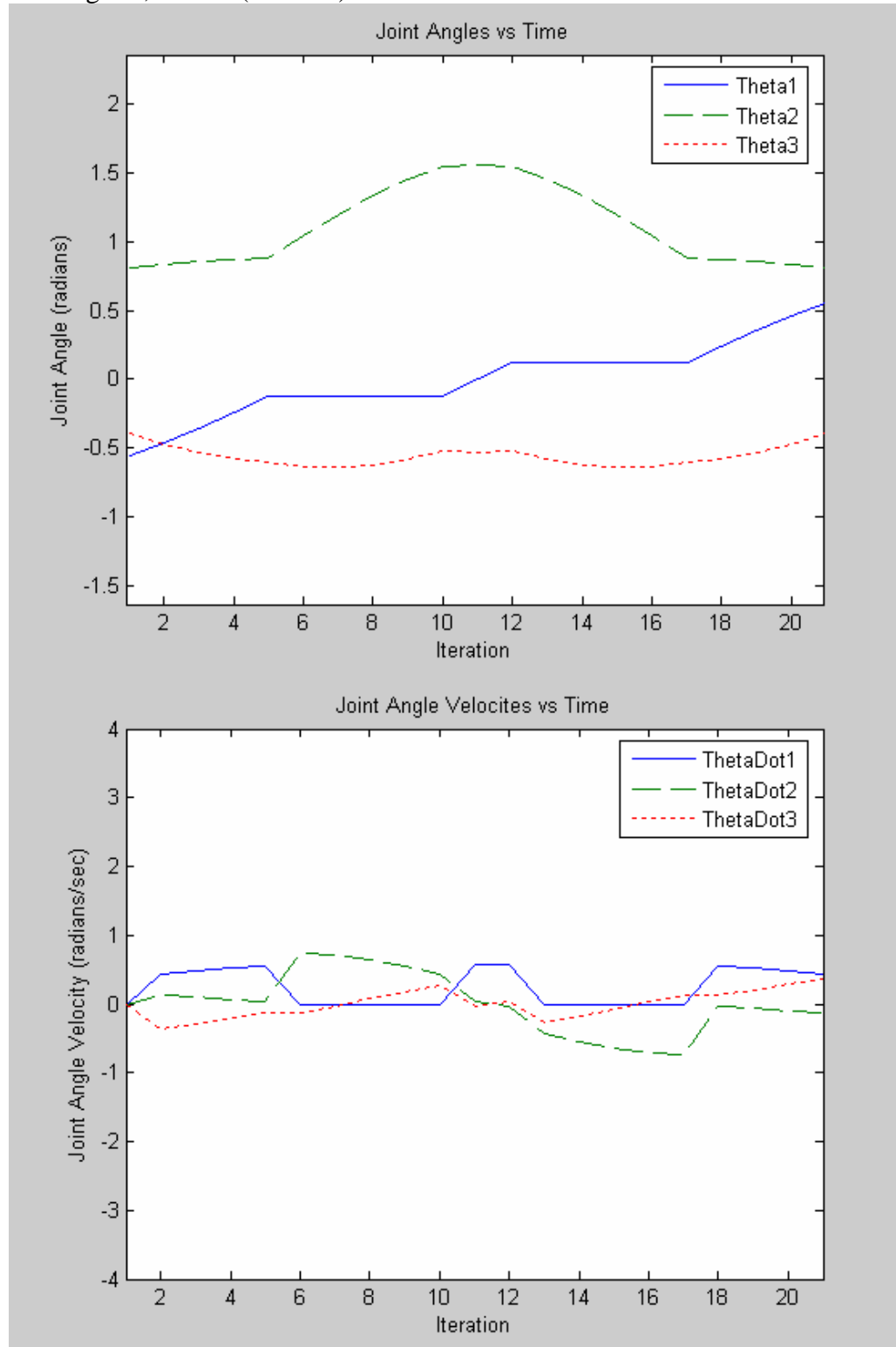


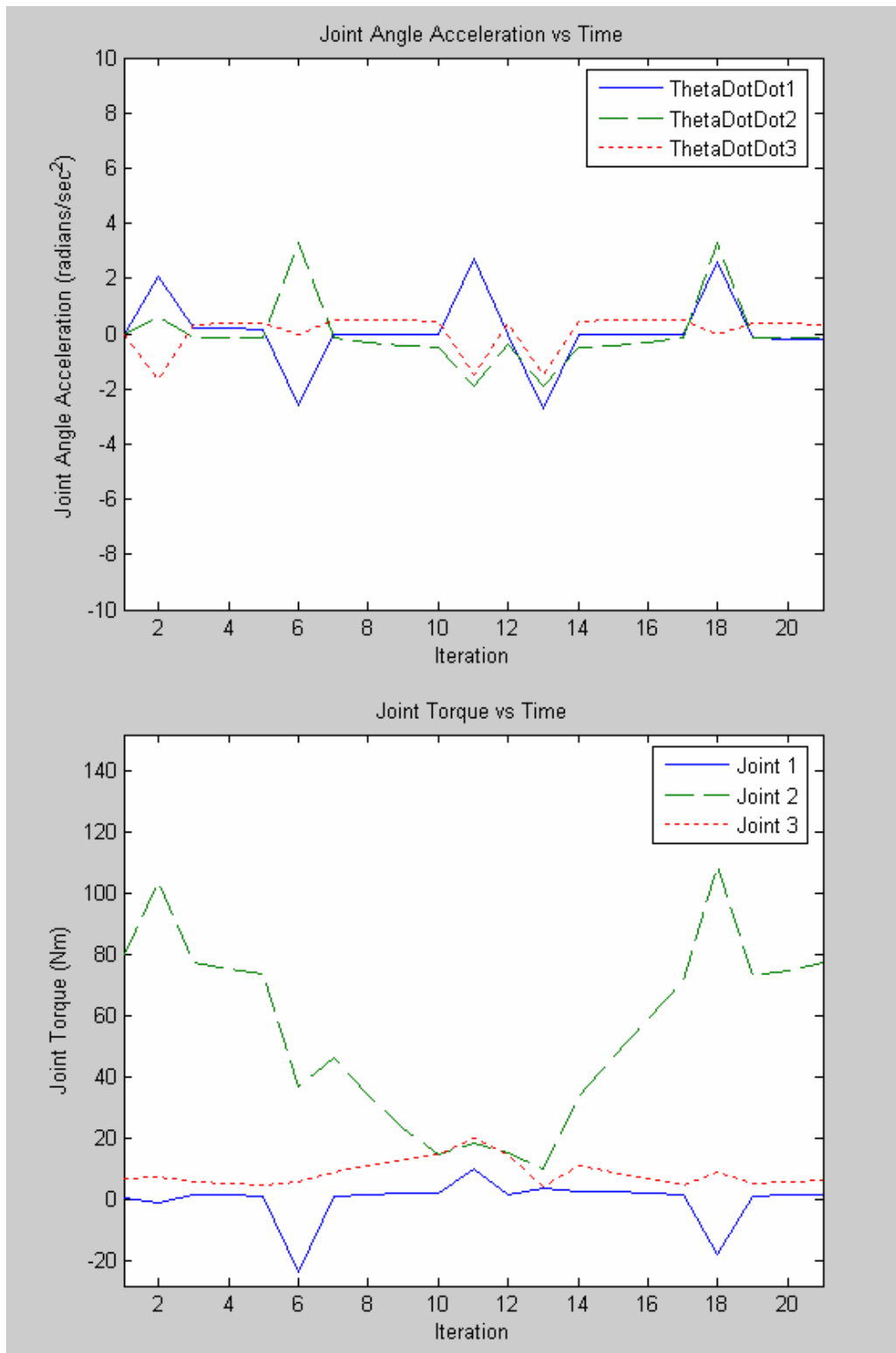


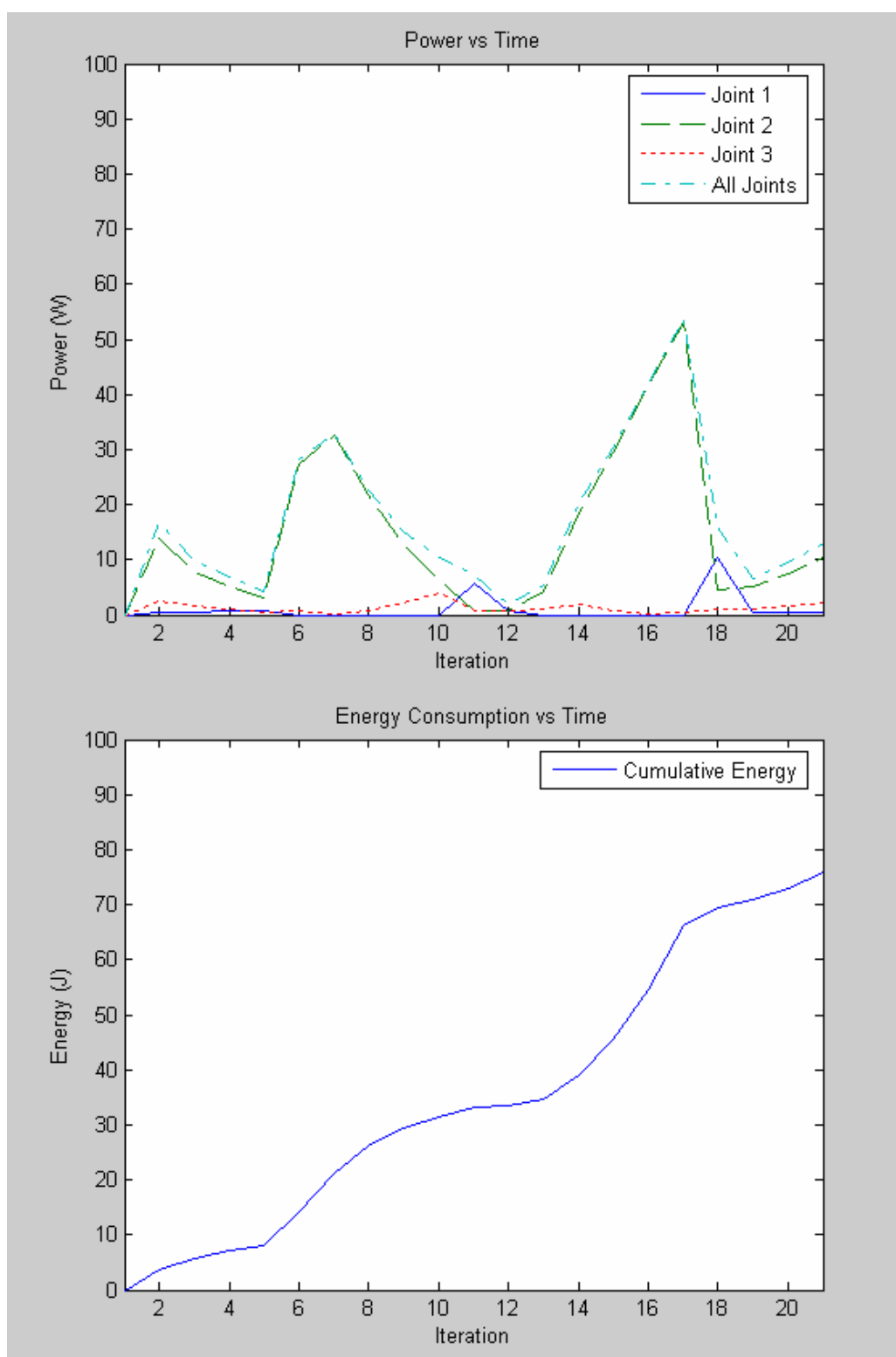


Energy consumed: 84.23 J

Rectangular, Case 2 (Vertical)

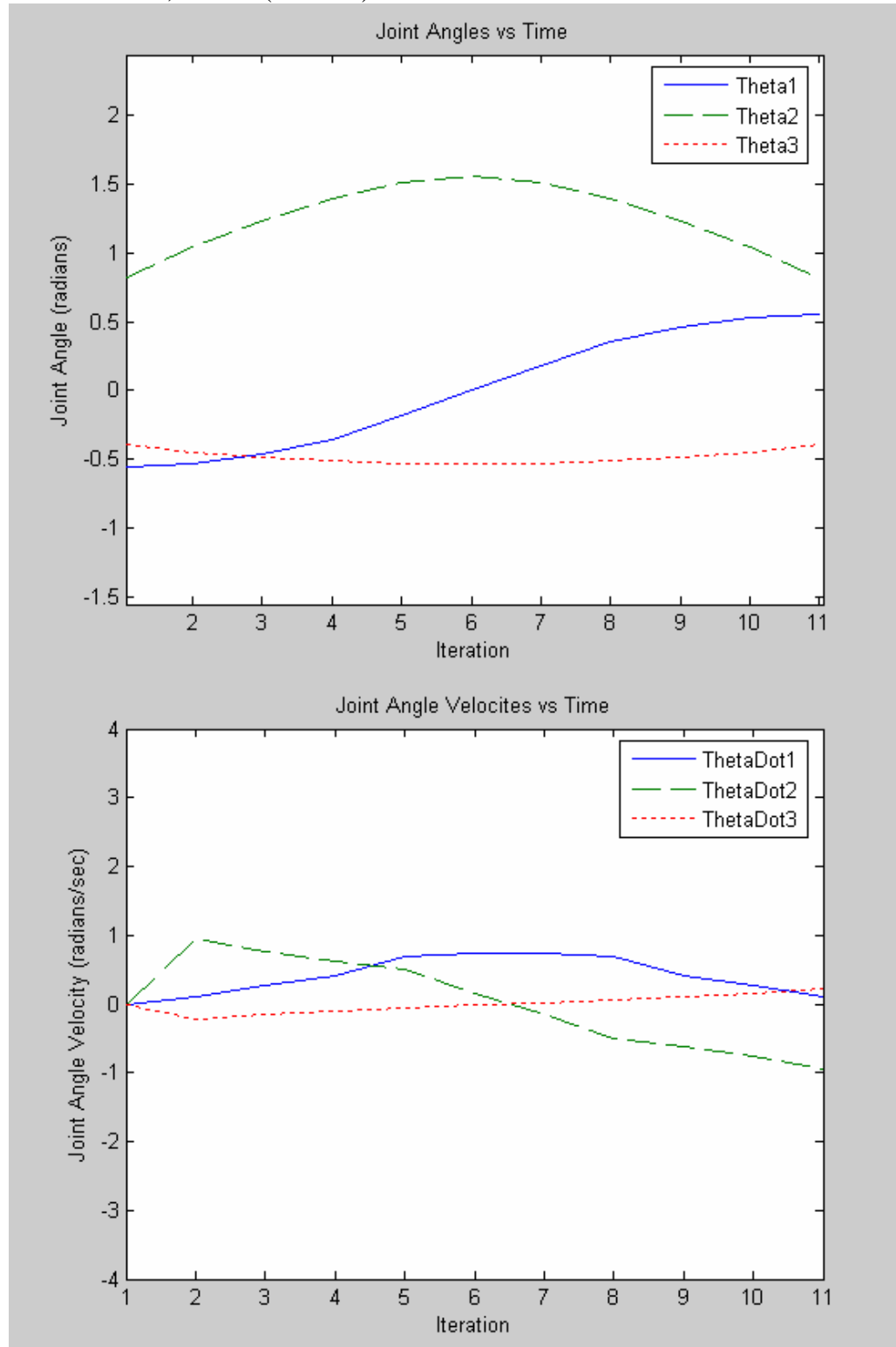


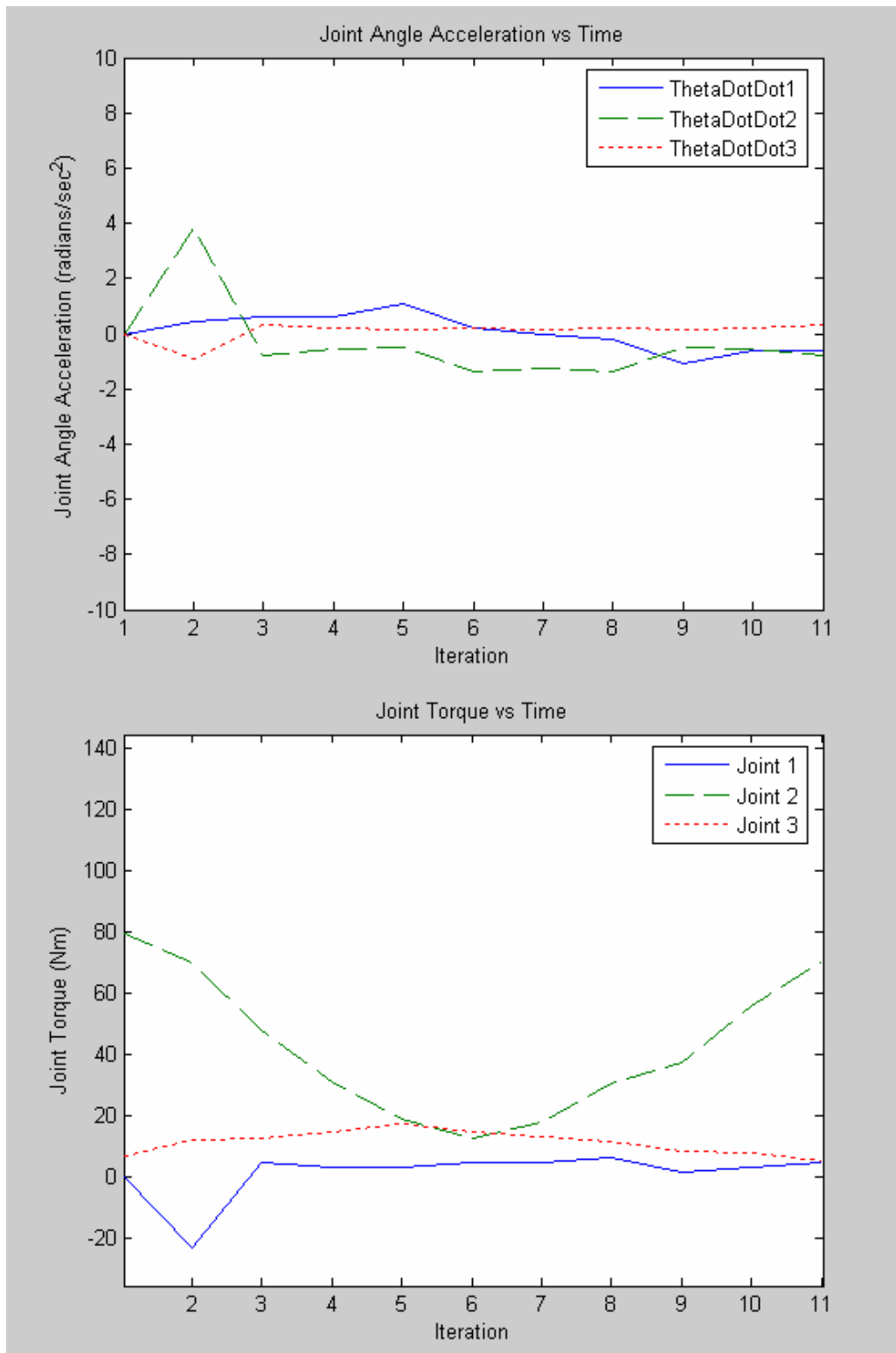


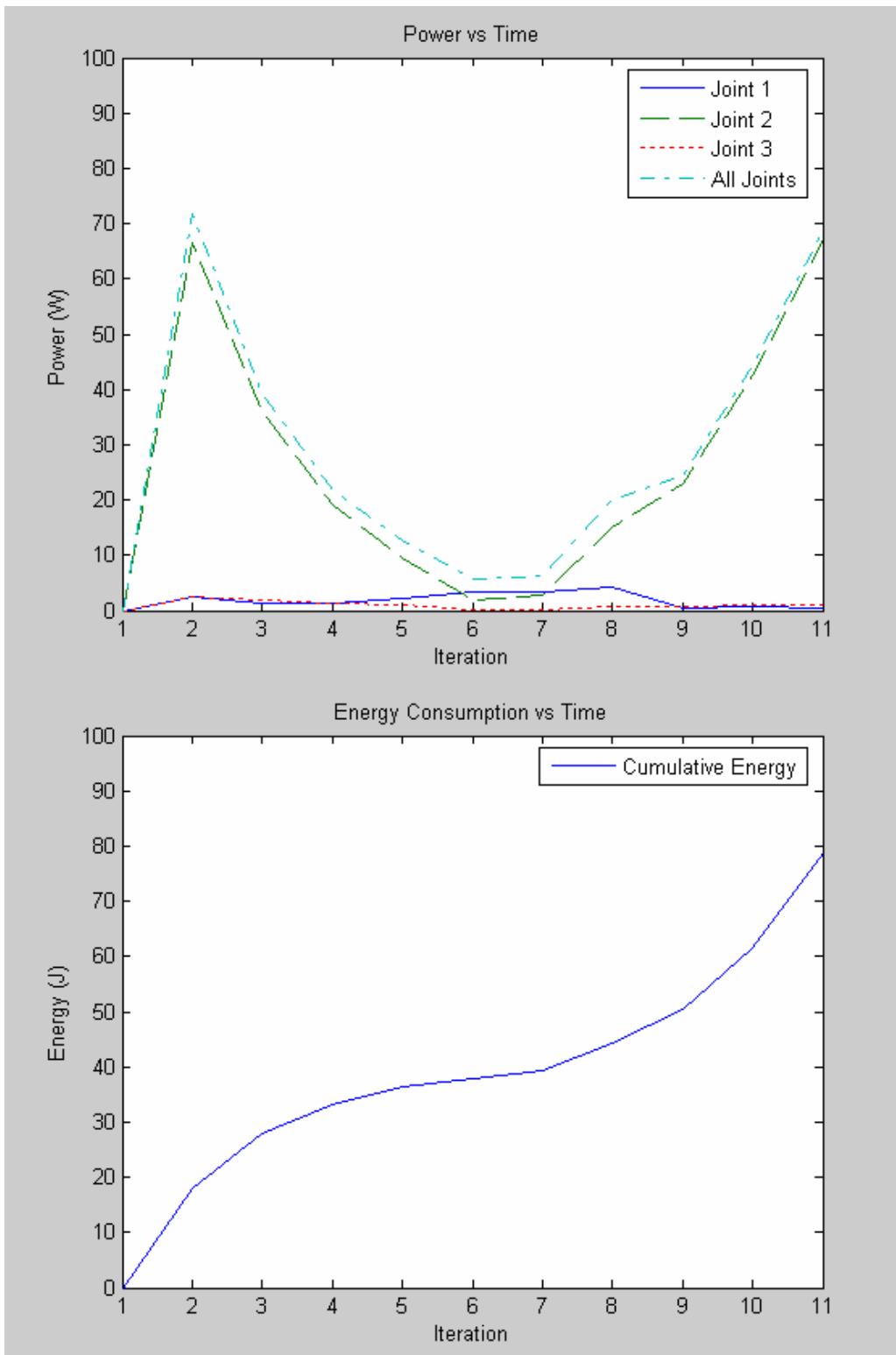


Energy consumed: 75.78 J

Semi-Circular, Case 2 (Vertical)

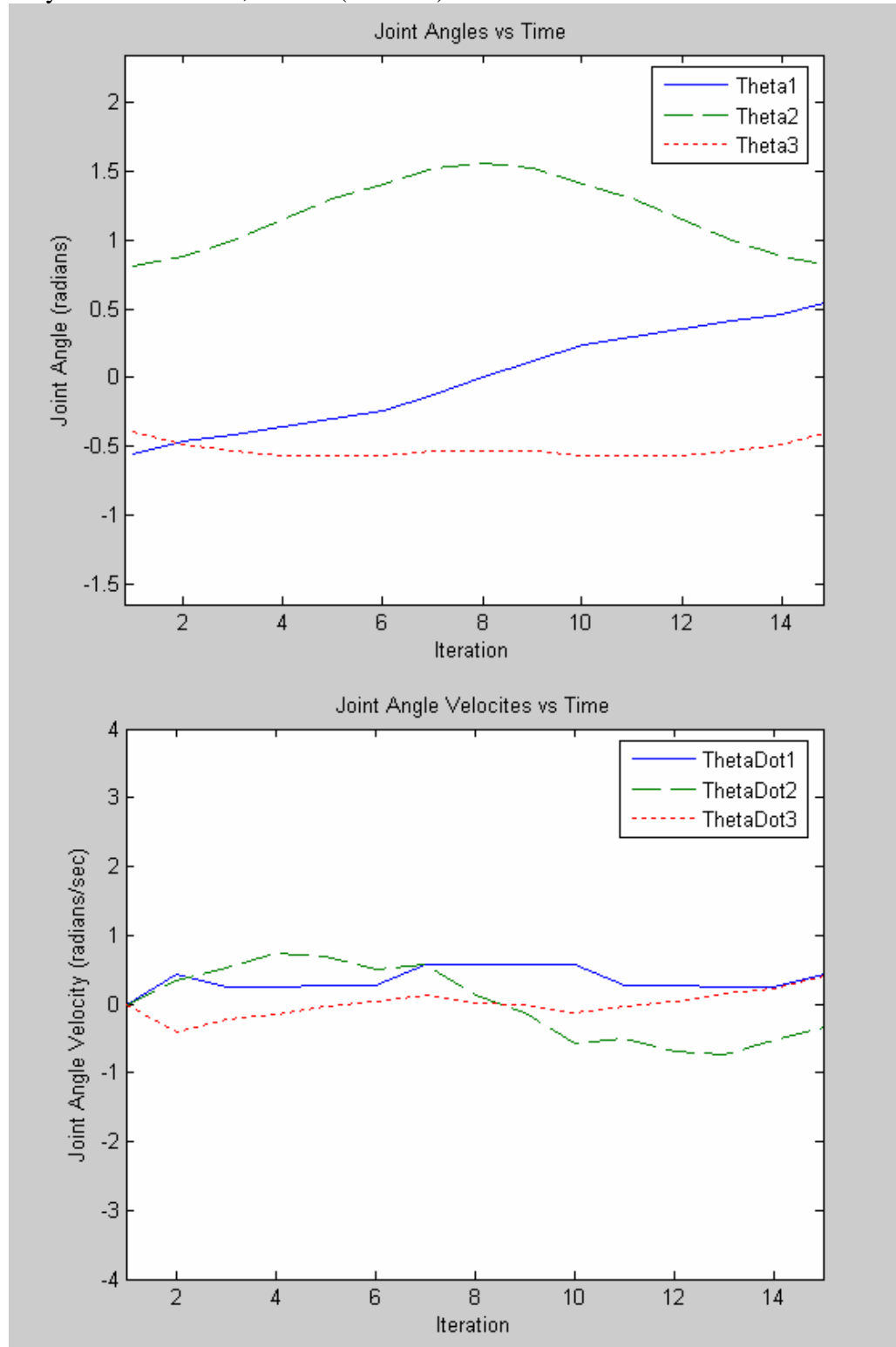


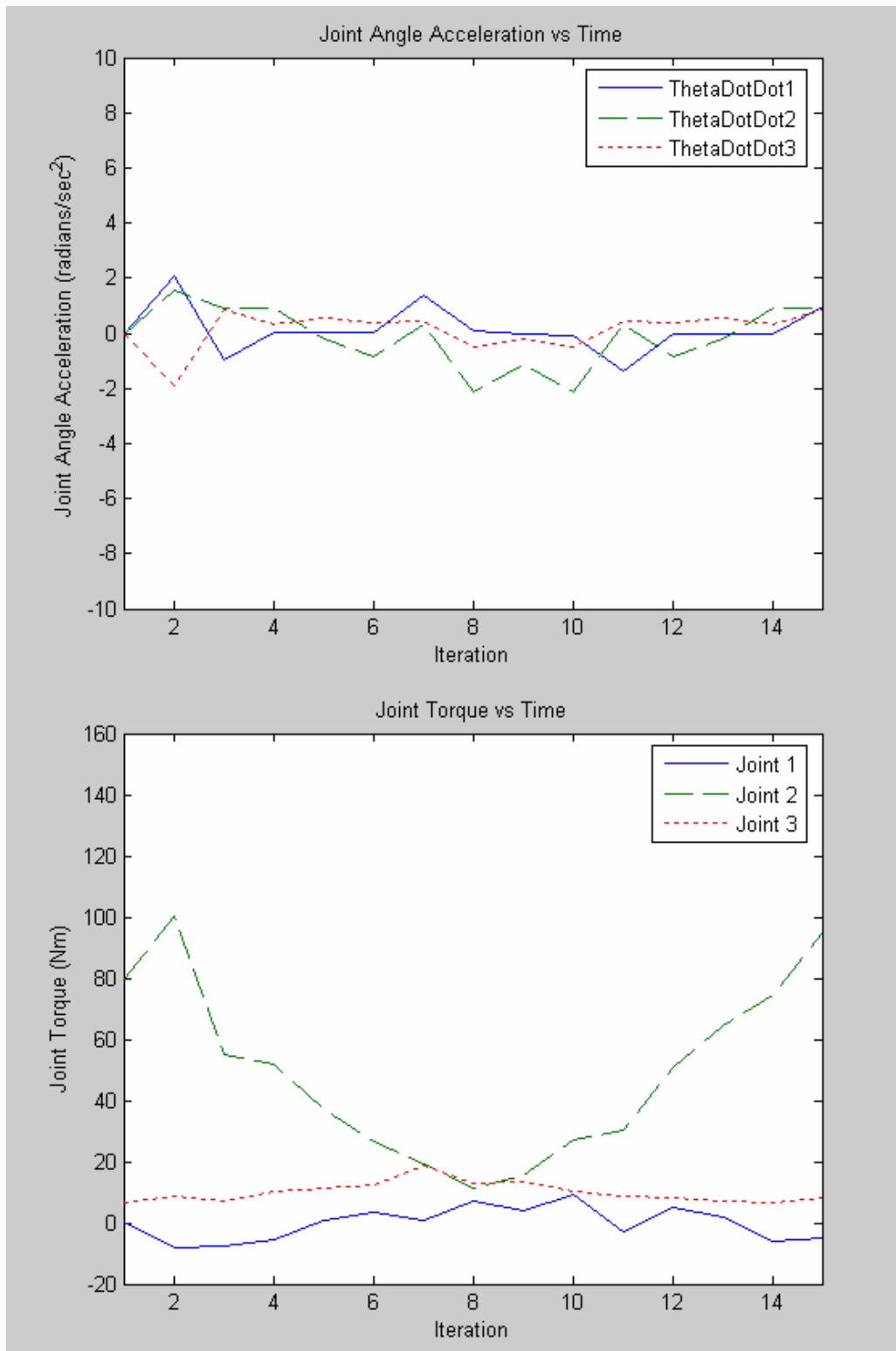


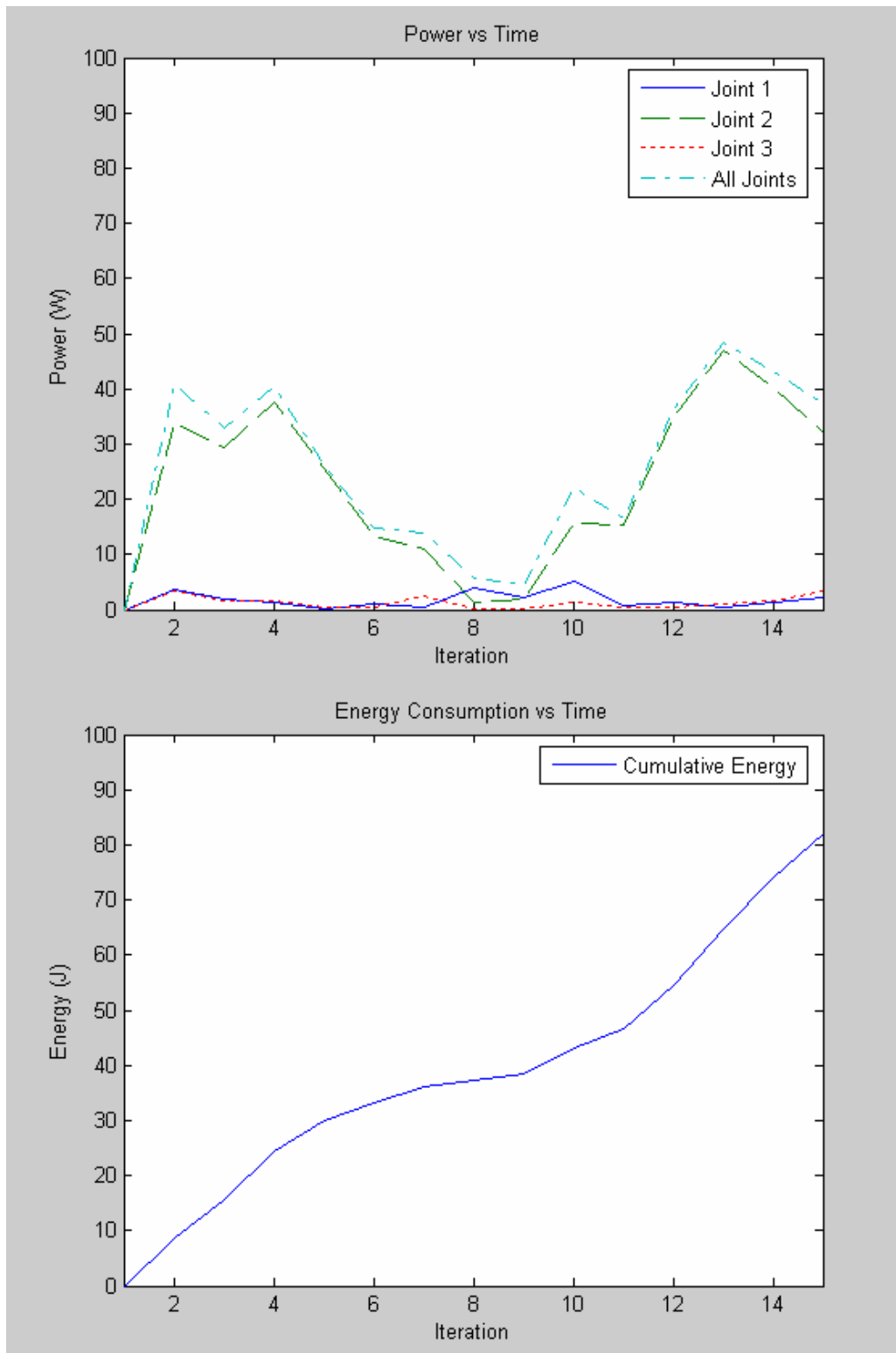


Energy consumed: 78.80 J

Polynomial – 4 DOF, Case 2 (Vertical)







Energy consumed: 82.09 J