The Inverse Jacobian Control Method Applied to a Four Degree of Freedom Confined Space Robot

Alexis Ehrlich

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

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

University of Washington

2016

Reading Committee:

Joseph Garbini, Chair

Santosh Devasia, Chair

James Buttrick

Program Authorized to Offer Degree:
Mechanical Engineering

University of Washington

Abstract

The Inverse Jacobian Control Method Applied to a Four Degree of Freedom Confined Space Robot

Alexis Ehrlich

Co-Chairs of the Supervisory Committee:
Professor Joseph Garbini
Mechanical Engineering

Professor Santosh Devasia
Mechanical Engineering

Robotics and automation of aerospace manufacturing and assembly in confined spaces such as a wing is challenging due to limited tooling access and geometric constraints set by the environment. The emphasis of this thesis is on the control of such robots using the Inverse Jacobian method. The approach is implemented on an experimental four degree of freedom serial linked confined space robot prototype. Simulation and experimental results are used to evaluate the Inverse Jacobian control. Results show that the approach achieves tracking of the prototype robot with errors on the order of 10^{-2} inches. This thesis concludes with lessons learned and suggestions for further improvements.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

This thesis would not have been possible if not for the Boeing Advanced Research Center. In particular James Buttrick, Craig Battles and Lance O McCann guided the project, while emphasizing practical engineering tactics. As well, Professor Santosh Devasia and Professor Joseph Garbini provided invaluable instruction and advice, showing great patience.

DEDICATION

I dedicate this to my parents, Ian and Deny Ehrlich, as well as my brother, David.

Chapter 1

INTRODUCTION

Often in aircraft assembly, it is possible to use large *open space* tooling that allows convenient access to all sides of the assembly, and for which the size of the machine is unconstrained. By contrast, this thesis addresses *confined space* assembly in which tooling access is limited, and the available space is geometrically bounded. Specifically, this work explores the use of a small dexterous robot, using the Inverse Jacobian control, for confined space applications.

In the manufacturing of aircraft wings, individual aluminum components are joined and assembled using discrete fasteners, such as rivets, bolts and swagged collars. These components are then assembled into a finished commercial aircraft wing. Many commercial aircraft wings are composed of three main components; the top skin, a rib or ladder assembly that provides stiffness, and a bottom skin. Each of these three components and their sub-assemblies can be manufactured and assembled with large, rigid, high precision machines in open space, with convenient access to both sides of the assemblies. Assembly of the three main wing components begins by attaching the top skin to the ladder assembly. This too can be done in open space. However, the task of fastening the bottom skin to the rest of the assembly must happen from inside the wing box. This is the point in assembly of the wing at which the confined space problem is first encountered.

To gain entry to the task space inside the wing box, special access holes have been designed into the bottom skin of aircraft wings. The conventional procedure requires that the technicians crawl fully or partially through the access hole, to manually perform the assembly tasks. Working through access holes inside the cramped wing box is difficult and time consuming.

The goal of the development described in this thesis is to reduce manual assembly in the confined space environment by designing a robot that can preform these in-wing assembly tasks. Fastening within the wing box requires robot motion largely parallel to the wing surface. For example, drilling a row of holes to fasten the wing skin to the spar chord. Only a small amount of motion is required perpendicular to this plane. The requirement that the robot perform assembly tasks inside this confined space is the principal constraint on the mechanical design, as well as the control scheme.

The robot tested in this thesis is designed to be placed through the small, human-sized access hole into the confined space of the wing box. To achieve mechanical rigidity, the base of the robotic arm has been designed to be placed within the wing, as close to the task as possible. An alternative method is to put the base of a large robot outside the confined space, from which the robot can reach through the access hole to the task space. Disadvantages of this approach include the required long length of the arm, the difficulty of its insertion, and the complexity of the design.

The geometry of the confined space, as well as the task motion requirements suggest that a suitable robot would consist of a four degree of freedom (DOF) serial linked robotic arm. The axes of rotation of the first three joints are parallel to each other and approximately perpendicular to the wing skin. The axis of rotation of the fourth joint is perpendicular to that of the first three joints, see figure 1.1. This allows for the tool tip of the robotic arm to reach all points of the task plane. The fourth joint gives access to points above and below this main plane of motion. This kinematic chain also allows for the robot to be folded up so it can fit through the access hole.

The version of the arm tested in this thesis is a 3D printed prototype using servo actuators equipped with harmonic drive gear boxes and incremental encoders. The first joint uses a 30:1 harmonic drive gear ratio with incremental encoding of 240,000 counts per revolution on the gearbox output shaft. The remaining three joints use a 100:1 gear ratio with 800,000 counts per revolution on the output shaft.

The confined space fastening tasks require continuous control of the tool tip trajectory. In

this context, a trajectory describes the three-dimensional path of the tool tip as a continuous function of time. From the tool tip trajectories, the required individual joint angle velocities are derived. For the serial link arm of figure 1.1, the relationship between a tool tip trajectory and the corresponding joint motion is not unique. This thesis addresses the difficulty of this ambiguity using the Inverse Jacobian method [1], described in the next chapter, to determine the appropriate joint velocities. The goal of this thesis is to evaluate the performance of the Inverse Jacobian method, for the described application, through a combination of analysis and experimentation.

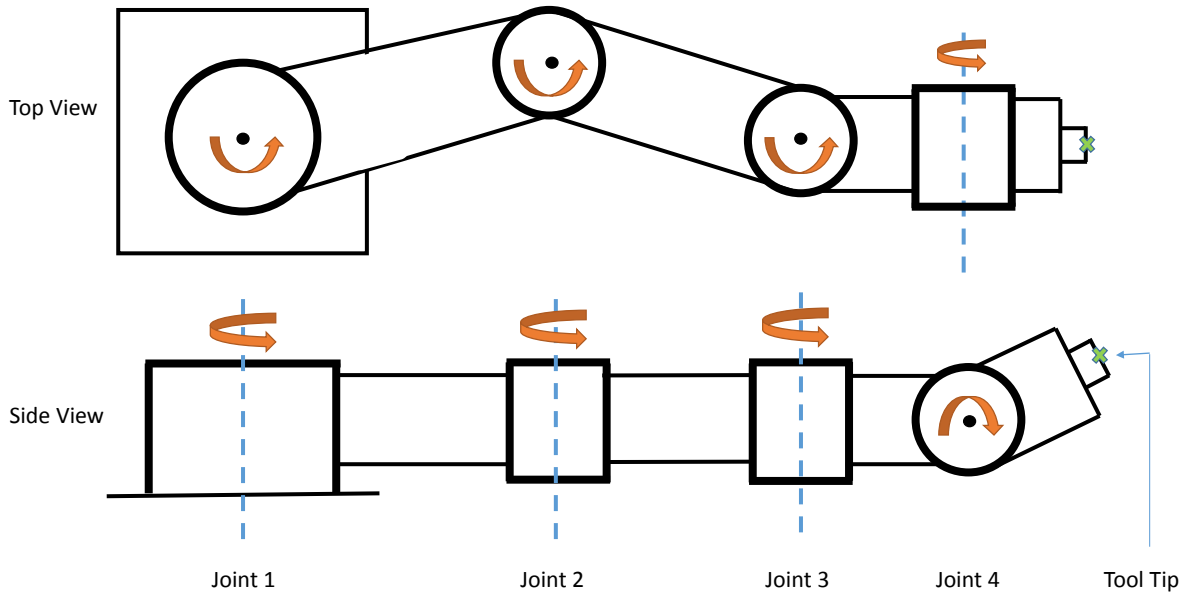

Figure 1.1: Schematic of robot arm. Above: Top view of the robot joints and links. Below: Side view of the robot joints and links.

Chapter 2

INVERSE JACOBIAN CONTROL SCHEME: DERIVATION AND ATTRIBUTES

2.1 Introduction and Background on the Jacobian in Kinematics

Recall that our task is to determine a description of the joint motions from the tool tip trajectory. To accomplish this goal for a four DOF robot, this thesis uses the Inverse Jacobian method. The Inverse Jacobian method is derived from the Jacobian equation, which will briefly be introduced here and then expounded on in the ensuing sections. The full Jacobian equation is a relation from angular velocities of the robot joints, to both Cartesian velocities, as well as angular velocity about the global frame's x, y and z axes, or $\omega_x, \omega_y, \omega_z$. For a derivation of the full Jacobian equation, see [2]. Here, we are concerned only with the control of the tool tip position, and not with its angular orientation. Therefore, the Jacobian equation referenced in this thesis will be a relationship between the Cartesian velocities of the tool tip $\dot{P} = (\dot{x}(t), \dot{y}(t), \dot{z}(t))$ and the vector of the joint angle velocities $\dot{\theta}(t)$ described by the ordinary differential equation:

$$\dot{P} = J(\theta)\dot{\theta} \tag{2.1}$$

where the Jacobian matrix, $J(\theta)$ is dependent on the time-varying joint angle vector. This chapter will review the kinematics of a serial link robot arm, show that for a given desired trajectory motion $\dot{P}(t)$, equation 2.1 can be inverted to find the joint velocities, $\dot{\theta}(t)$, and show how the Inverse Jacobian method is implemented in discrete time.

Equation 2.1 can be directly derived from the forward kinematics equations, [2]. The most important attribute of equation 2.1 is that this mapping from joint motion $\dot{\theta}$, to Cartesian motion $\dot{P}$, is a function of θ, derived from its dependency on the orientation of the robot's

joints. The tests implemented on the control system and robot described in the following chapters were designed to exploit characteristics of this mapping.

The following sections begin with the forward kinematics equation, which describes the geometry of the machine being controlled, and go on to derive the Inverse Jacobian control scheme.

2.2 A Brief Introduction to Forward Kinematics

In robotics, it is beneficial to have a mathematical description of the robot. The function describing the tool tip position, $P(t)$ as a function of the joint angles θ, is known as the forward kinematic function:

$$P = f(\theta) \tag{2.2}$$

For example, consider the single DOF robot that rotates about the origin of the global (x, y) frame, shown in figure 2.1. Frame 1 is rigidly attached to the end of the link with length L_1.

For this case, the global position of the tool tip is described by:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & L_1\cos(\theta) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & L_1\sin(\theta) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \tag{2.3}$$

where (x_1, y_1, z_1) are the coordinates of the tool tip in frame 1, and (x, y, z) are its corresponding coordinates in the global frame. For a given angle θ, the position of the tool tip is transformed from the local coordinate frame 1, into the global coordinate frame.

If a second link is added as shown in figure 2.2, a coordinate in the new tool tip frame can be transformed back to coordinate frame 1. This new coordinate in frame 1 can then be transformed into the global coordinate frame. This results in a transformation from the tool tip, through both links of the robot, back to the global frame.

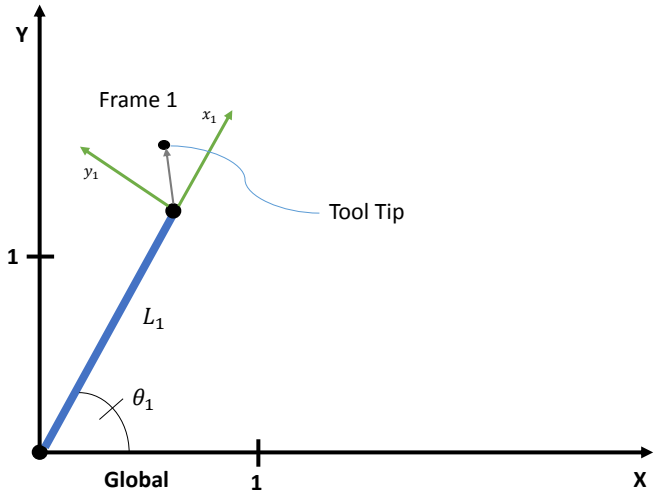

Figure 2.1: Forward kinematics illustration of a single DOF robot.

The overall transformation of the coordinates of the tool tip from the local frame to the global frame, as shown in figure 2.2, is the concatenation of the individual transformations. That is:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & L_1\cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & L_1\sin(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & L_2\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & L_2\sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} \tag{2.4}$$

By concatenating each link's respective matrix, this method can be extended to any number of serial links. The forward kinematic equations are a mathematical representation of the robot. However, as we will see in the following section, the inverse of the forward kinetic equations is often required.

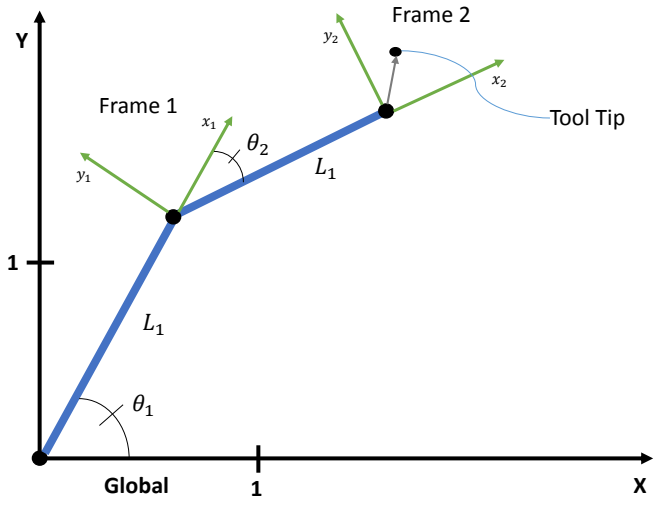

Figure 2.2: Forward kinematics illustration of a two DOF robot

2.3 Inverse Kinematics and the Problem in Robotics

The forward kinematics of equation 2.2 transform the robot joint angles into a coordinate in Cartesian space. However, for purposes of control, the opposite (or inverse) is needed. Frequently, we want to transform the Cartesian coordinates of the tool tip into the joint angles necessary to reach that position.

Consider a robot that drills holes. The robot will need to position its tool tip over the hole location before it can start drilling. Robot arm joint angles must be found that correspond to the hole location. For this reason, the inverse of the forward kinematics, known as Inverse Kinematics (IK), is often the key for robotic applications. Unfortunately, it is not easy to solve equation 2.2 for the joint angles. This is true algebraically, but it can also be understood physically.

If every angle of a robot's joints are specified, there is only one position that the tool

tip can be in. Conversely, if the tool tip position is specified (and the robot has more than 2 DOF), there are an infinite amount of joint angles that will satisfy the desired tool tip position. As illustrated in figure 2.3.

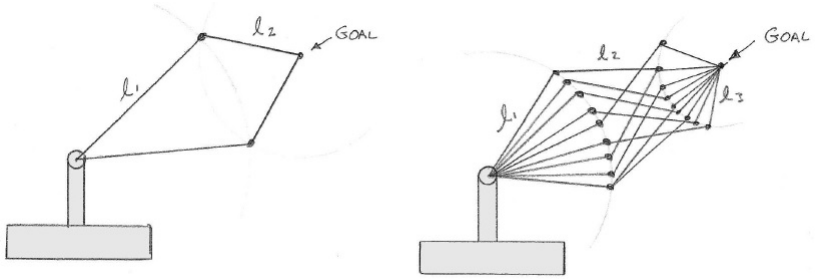

Figure 2.3: Planar manipulators with 2 link (left) and 3 links (right) pointing at the same goal point, [2].

This infinite set of joint angle possibilities manifests itself in not being able to solve for any joint angle configuration. This seemingly innocuous issue of not being able to solve for the Inverse Kinematics has proven to be one of the most persistent problems in robotics. One method to get around the IK problem, is to add constraint equations to the forward kinematic equations. This can be done, but it will result in constrained motion of the robot. In some applications this is an acceptable method. However in many applications the full range of the robot's motion is needed.

Alternatively, an iterative algorithm called the Inverse Jacobian incrementally guides the tool tip from its current location, towards the final desired location. This was the method applied to the confined space robot and is the focus of this this thesis.

2.4 Forward Kinematics to the Jacobain

We begin to formulate the Inverse Jacobian method by first deriving the Jacobian equation, 2.1 from the forward kinematic equation, 2.2 for the particular case of the two-link robot illustrated in figure 2.2.

To get the Cartesian velocities of the Jacobian equation, first the rotational terms of the forward kinematic equation should be zeroed out, isolating the translational components of the equation. Equation 2.5 shows the the forward kinematic equation describing the two-link robot which transforms the origin of frame two into the global frame.

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 & L_2\cos(\theta_1+\theta_2)+L_1\cos(\theta_1) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 & L_2\sin(\theta_1+\theta_2)+L_1\sin(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.5}$$

By multiplying the origin of the local coordinate frame with a 1 concatenated on the bottom, the 3x3 rotation matrix within the forward kinematics equation is nullified and the translational components of the forward kinematics equation are left unchanged. The forward kinematics equation takes the form:

$$\begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} L_2\cos(\theta_1(t)+\theta_2(t))+L_1\cos(\theta_1(t)) \\ L_2\sin(\theta_1(t)+\theta_2(t))+L_1\sin(\theta_1(t)) \\ 0 \end{bmatrix} \tag{2.6}$$

where the fourth row of the forward kinematics can be left out, as it was only used to make inverting the forward kinematics matrix possible, and holds no information about the robot.

To get from the forward kinematics to the Jacobian, the derivative of equation 2.6 must be taken with respect to time. Each joint angle, theta is dependent on time, therefore angular velocity emerges on the right hand side of the equation. Each Cartesian coordinate is dependent on time and therefore Cartesian velocity emerges on the left hand side.

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} -L_2\sin(\theta_1(t)+\theta_2(t))-L_1\sin(\theta_1(t)) & -L_2\sin(\theta_1(t)+\theta_2(t)) \\ L_2\cos(\theta_1(t)+\theta_2(t))+L_1\cos(\theta_1(t)) & L_2\cos(\theta_1(t)+\theta_2(t)) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \end{bmatrix} \tag{2.7}$$

Equation 2.7 is the Jacobian equation for the double link robot illustrated in figure 2.2. The general form of this equation was introduced at the beginning of chapter 2 and is shown again below,

$$\dot{P} = J(\theta)\dot{\theta} \tag{2.8}$$

The matrix $J(\theta)$ for the two-link robot is dependent on both joint angles θ_1 and θ_2. This shows the dependency of the relationship between joint velocities and Cartesian tool tip velocity on the robot pose. This dependency will be a focal point of tests described in chapter 3.

The following sections will take the Jacobian equation and manipulate it into the Inverse Jacobian control used in the tests on the four DOF arm.

2.5 Inverse of the Jacobian

Section 2.3 explained how robot applications often require the joint angle motions for a corresponding tool tip trajectory. To solve for the joint angle velocities of the robot, we can take the inverse of the Jacobian matrix in equation 2.8.

If there are exactly three joints and the robot is working in x, y and z coordinates, the Jacobian matrix will be a 3x3. This falls out from collecting terms after taking the derivative of the forward kinematics equation. If the robot has more or less joints than it has Cartesian degrees of freedom, the Jacobian matrix will not be square. It then becomes impossible to take the inverse of this non-square Jacobian matrix.

Fortunately, it is still possible to solve for $\dot{\theta}$. The Pseudoinverse of the Jacobian matrix is one possible method for solving equation 2.8 for $\dot{\theta}$.

There are multiple methods that could be used, to name a few:

- Method of least squares (Pseudoinverse)
- Use of the Singular Value Decomposition (SVD)
- QR Decomposition
- LU Decomposition
- Take the transpose (not an approximation to the inverse but has some desirable characteristics for control)

The method of inversion directly affects the form of the robot tool tip trajectory. Therefore, it is crucial to understand the attributes of the tool tip trajectory for the chosen method of inversion. The Least Squares (Pseudoinvere) was chosen because, under the correct circumstances, it will produce a straight line tool tip trajectory. This predictable straight line trajectory could then be used to approximate more complex tool tip trajectories.

2.5.1 Least Squares Jacobian Inverse Derivation

An informative way to look at the least squares inverse of the Jacobian is to formulate it as a minimization problem, [3]. The least squares inverse is a by product of minimizing the the tool tip error as well as minimizing the change in joint angle. This can be formulated as the following cost function, F.

$$F = \frac{1}{2}\dot{\theta}^T\dot{\theta} + \lambda^T(\dot{P} - J(\theta)\dot{\theta}) \tag{2.9}$$

Where λ^T is a vector of Lagrange multipliers.

(1) $\frac{\partial F}{\partial \lambda} = 0 \Rightarrow \dot{P} = J(\theta)\dot{\theta}$

(2) $\frac{\partial F}{\partial \dot{\theta}} = 0 \Rightarrow \dot{\theta} = J(\theta)^T\lambda \Rightarrow J(\theta)\dot{\theta} = J(\theta)J(\theta)^T\lambda \Rightarrow \lambda = (J(\theta)J(\theta)^T)^{-1}J(\theta)\dot{\theta}$

Insert (1) into (2):

(3) $\lambda = (J(\theta)J(\theta)^T)^{-1}J(\theta)\dot{P}$

Finally, insert (3) into (2):

$$\dot{\theta} = J(\theta)^T(J(\theta)J(\theta)^T)^{-1}\dot{P} \tag{2.10}$$

In literature, the least squares method is often called the Pseudoinverse and is represented with the dagger symbol shown below:

$$\dot{\theta} = J(\theta)^{\dagger}\dot{P} \tag{2.11}$$

The cost function formulation provides intuition into the behavior of the robot manipulators motion. If the least squares inverse is used in the Inverse Jacobian control scheme, it is expected that the tool tip will track a straight line, simultaneously minimizing its change in joint angles. This is inherent in the cost function formulation, who's first term minimizes change in joint angles, and second term minimizes change in Cartesian position.

2.6 Formulation of the Inverse Jacobain Control Scheme

Section 2.5 derived the Inverse Jacobian equation and explained the attributes of the equation through a minimization formulation. This section will take that Inverse Jacobian equation and formulate it as a control scheme.

This control scheme will be implemented on a digital system, so equation 2.11 must be discretized. The discretization of the Inverse Jacobian equation takes the form,

$$\frac{\Delta\theta}{\Delta t} = J(\theta)^{\dagger}\frac{\Delta P}{\Delta t} \tag{2.12}$$

The goal of the control scheme is to be able to specify a desired location and have the robot tool tip move along a predictable trajectory to this position. The input to the control scheme is the 3x1 desired Cartesian vector. The output will be an nx1 vector of joint angle velocities, where n is the number of joints. The output of equation 2.11 are joint angle velocities. However, the input is Cartesian velocity but to solve the Inverse Kinematic problem, a single position vector is needed. This issue can be resolved by representing the change in position, ΔP as the difference between desired tool tip position and current tool tip position.

$$\frac{\Delta\theta}{\Delta t} = J(\theta)^{\dagger}(P_{desired} - P_{current})\frac{1}{\Delta t} \tag{2.13}$$

Equation 2.13 will be implemented as the Inverse Jacobian control scheme and is the focal point of this thesis.

2.7 Inverse Jacobain Control Block Diagram

To implement this control scheme on the four DOF confined space robot, this control scheme was written in matlab and Simulink. The code was uploaded on to a real time controller called a Speedgoat GmbH. The code was formulated in Simulink as a block diagram and can be seen below in figure 2.4.

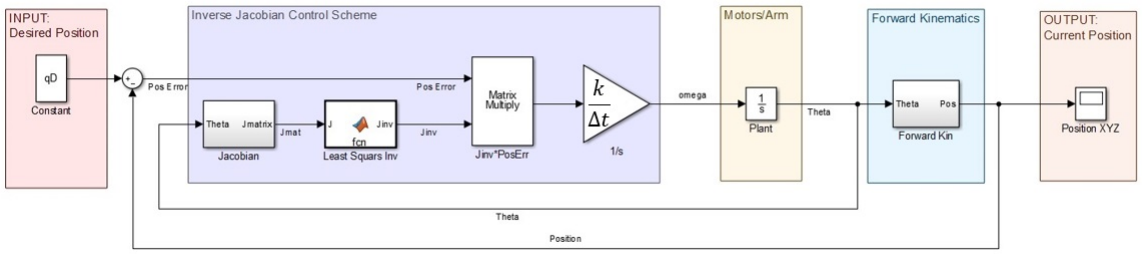

Figure 2.4: Block Diagram of Inverse Jacobian Control Scheme

Looking at figure 2.4, the current joint angles are fed through the forward kinematics. This produces the current Cartesian position. This current position is fed back and subtracted from the desired position, to produce ΔP. The current joint angles are also used to update the Jacobian matrix, before it is inverted. The Jacobian is then inverted according to the least squares method described in section 2.5.1 and multiplied by ΔP, as well as the gain, k and $\frac{1}{\Delta t}$. This produces joint angle velocities that will move the tool tip in a direction approximately pointing towards the desired location. These joint velocities are fed to the robot, where local PI controllers are responsible for tracking these angular velocity inputs. The motors move the links of the arm, which result in an updated tool tip position, as well as joint angle positions at every time step. These updated positions are fed back through the control scheme and the process repeats from there.

As the current position of the tool tip approaches the desired location, ΔP becomes smaller and the velocity commands subsequently become smaller. As well, the desired location becomes closer to the pose of the arm that is updating the Jacobian, resulting in

evermore accurate movement towards the desired location. The latter concept will be examined through experimentation in the following chapter.

It is with noting that the forward kinematics equation is used to updated the tool tip position, producing the $P_{current}$ in equation 2.13. This use of the forward kinematics could be replaced, and even account for mechanical backlash if there was a sensor tracking global position of the robot tool tip. However, this is unlikely in many applications. The use of forward kinematics is a cheap method that puts to use motor encoder positions to produce a tool tip location.

The Inverse Jacobian control derived in this chapter and shown in figure 2.4 accomplishes the original goal of determining the appropriate joint angle motions to move the tool tip of a robot to a desired location. Additionally, the tool tip motion is predictable, and when kinematically possible, follows a straight line. This control scheme gives the engineer or operator a predictable method for control of the robot. The following chapter will examine this Inverse Jacobian control through experimentation on a four DOF serial linked confined space robot.

Chapter 3

INVERSE JACOBIAN CONTROL SCHEME TESTS ON A FOUR DOF CONFINED SPACE ROBOT

3.1 Introduction to the Preformed Tests

This chapter introduces the test instrument and setup routine, reviews three different tests used to isolate errors, followed by two tests and a simulation to elucidate expectations of Inverse Jacobian control. The purpose of these tests is to validate expectations of the Inverse Jacobian control derived in chapter two, as well as to learn if there are any attributes of the Inverse Jacobian that may have been overlooked.

The approach is to, control the robot arm based on the Inverse Jacobian control method, independently measure the motional response, and to compare the modal behavior.

There were 11 different tests, simulations and procedures preformed on the four DOF confined space robotic arm:

- 5 procedures to set the coordinate frame, initialize the arm and measure kinematic parameters.

- 3 tests to isolate errors from the measurement machine, servos and robot arm.

- 2 tests and 1 simulation to elucidate the tool tip motion characteristics of the Inverse Jacobian control as well as how update rates affect tool tip tracking errors.

The machine used for measurements during these tests was a laser tracker. The following chapter will detail what a laser tracker is, how it works and how it was used.

3.2 Laser Tracker Setup and Procedures

Before the tests on the control system and robot could be carried out, the test setup had to be carefully executed to make sure inferences could be made from the data. This section will introduce the measurement tool used, as well as detail the test setup procedure.

3.2.1 Laser Tracker and software Description

The machine used for taking the measurements from each test was an API Tracker3 laser tracker. A laser tracker is a general purpose machine measurement tool for measuring distance and angles. It can be used to map points in space to approximate geometric shapes. It can also be used to dynamically track machine motion.

There are two main components, the Spherically Mounted Retro-reflectors or SMR, seen in figure 3.2, and the laser tracker machine, seen in figure 3.1. SMRs are magnetic hallowed out balls filled with mirrors for reflecting back a laser beam. They are used to reflect the laser back to the laser tracker and are the object that the laser tracker measures. The laser tracker machine has two axes of motion controlled by servos. It also has the laser generator and receiver located in the head of the machine. As the SMR ball is moved, the servos rotate the laser to follow it. The tracker reports the position of the SMR ball by monitoring its encoders on each axis and by measuring the distance to the SMR ball through interferometry.

The data aquisition is controlled by the Spatial Analyzer software. Spatial Analyzer is software used in conjunction with the Tracker3 laser tracker. The software controls how the laser tracker takes data (how often it takes points, how many points to take, etc.), it stores the data and it can manipulate the data (fit the data to geometric shapes).

To ensure the SMR is placed in a repeatable position relative the the object it is measuring, magnetic SMR nests were hot glued to the object. The SMR ball could then be placed in the nest, and to within a high tolerance, be in the same position with respect to the object it was mounted on. In some cases, such as in section 3.2.2, the SMR ball was manually held in place by an operator. This was only used in cases where placing the SMR ball in the same

position was not critical to collecting the data.

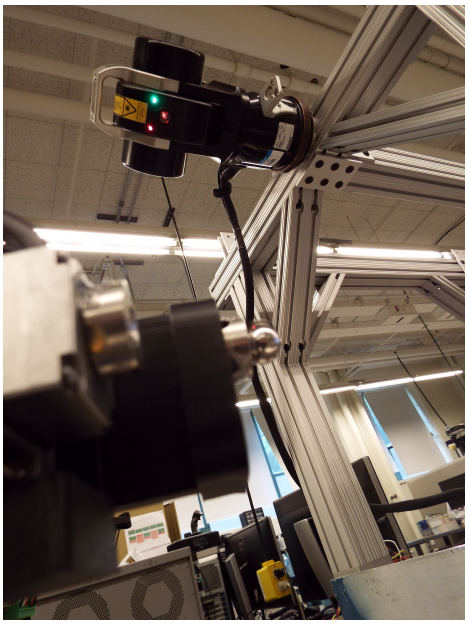

Figure 3.1: Laser Tracker pointing at an SMR. Viewed from below.

Figure 3.2: This SMR is larger than the one used in the experiments.

3.2.2 Setting up a Global Coordinate Frame

A global coordinate frame had to be established so that the data sets from one day to the next could be compared. To establish the global coordinate frame:

- The direction of the x-axis was measured from a machined edge on the base holding the robot. This was done by manually dragging the SMR ball along the machined edge and taking a point every 0.20 inches for 6 inches.
- The x-y plane was established by measuring points on the base plate of the robot. This was done by manually placing the SMR ball at different positions on the robot base plate. These points were then fit to a plane using Spatial Analyzer.
- The origin of the coordinate frame was established by putting an SMR in a nest on the tool tip of the robot, and commanding the robot's first joint to rotate. A circle was fit to the measured points using Spatial Analyzer. The origin of the circle was projected onto the x-y plane along the plane's normal and established as the coordinate frame origin.
- The spatial Analyzer software used these stored planes and lines to create a global coordinate frame.

The projection of this global coordinate frame onto the robot is show in figure 3.3.

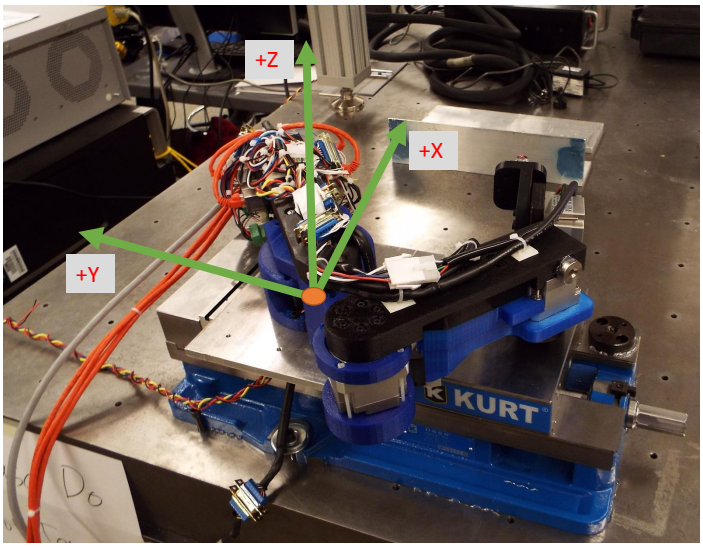

Figure 3.3: Working global coordinate frame that all data was taken in.

Once the original global coordinate frame was established, there had to be a method for the system to reestablish this global coordinate frame from day to day. To accomplish this, four magnetic SMR nests were laid out over the work space, seen in figure 3.4. These nests were placed in positions that would not be interfered with for the duration of the tests. On the first day the global coordinate frame was established, a point was taken with the SMR in each nest. These four points were stored and referenced for the duration of the tests.

To reestablish the global coordinate frame, rather than go through the entire procedure listed above, a point was measured with the SMR in each nest. The Spatial Analyzer software input the four original stored points and the software generated a best fit to the original coordinate frame. The Spatial Analyzer software also output discrepancies between the fit of the original points taken on the first day, and the fit to those points taken on each ensuing day. The discrepancies were on the order of 10^{-5} inches.

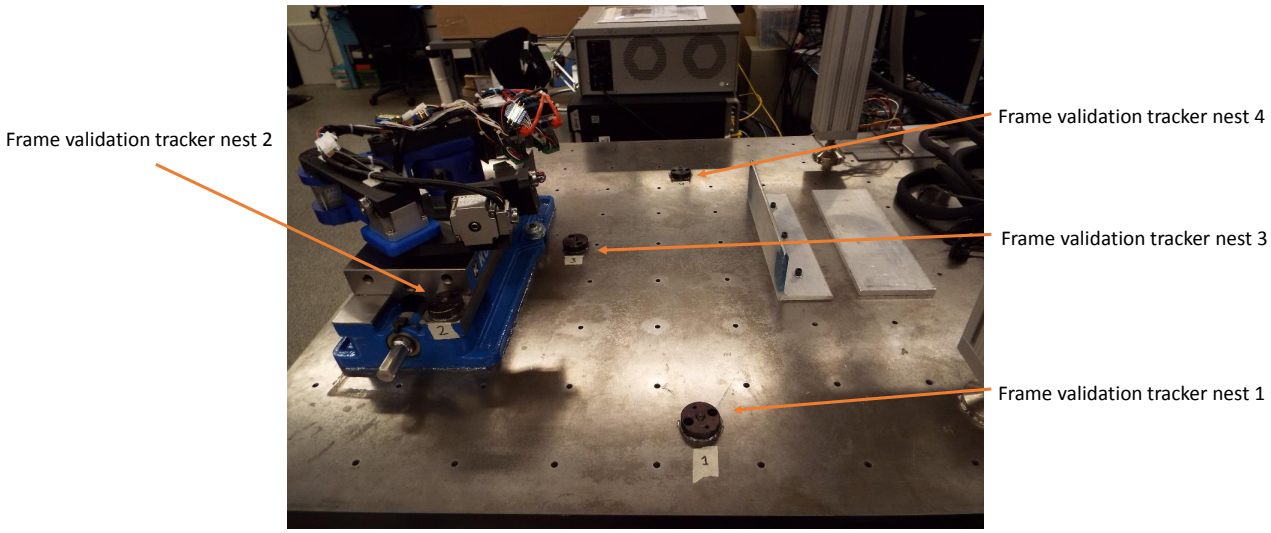

Figure 3.4: Tracker nests for global coordinate frame validation.

3.2.3 Robot Encoder Initialization

To establish the forward kinematics and a mathematical description of the robot, coordinate frames were assigned at specific positions and orientations on the robotic arm. To have the measurement of the encoders match the mathematical joint angles, the joints of the robot were moved as close to the kinematic zero angle position as possible, where the encoder counts were zeroed. This would ensure the difference between the true angular position of each joint and the calculated angular position of each joint, to be as small as possible.

To put the first three joint angles in their theoretical zero angle position, the first three links of the arm were set parallel to the x-axis. To accomplish this, the face of the robot tool tip needed to be as far from the origin of the robot as mechanically possible, while simultaneously having the face of the robot tool tip perpendicular to the x-axis. This would establish the links parallel to the x-axis with the first three joints in their zero position. As well, if the face of the robot tool tip was perpendicular to the x-axis, the forth joint would be in its zero angle position.

To get the robot into this pose, an aluminum angle bar was fixed to the same optical table the robot was fixed to. The face of the angle bar was measured to be as perpendicular as possible to the global x-axis. This was done by measuring points on the surface of the angle bar and fitting a plane to the measured points. Then the angle between the plane normal and the x-axis was calculated. The angle bar was tapped into place according to that angle measurement. The process was repeated until the angle was approximately less than 0.1 degrees. The face of the angle bar was set so that it was at the end of the robot's fully extended reach. At this point, the robot was constrained to be parallel to the x-axis. While the robot was in this position, the encoder w counts were zeroed.

The motors used for this robot were incremental and there were no homing switches integrated into the robot. Therefore, the motor encoders had to be zeroed in this same configuration each time the robot was turned on. The process of using the angle bar to get the arm straight along the x-axis was only used once.

To zero the encoder counts in the correct position each time the robot was turned on, four small tracker nests were glued to each link of the robot arm, including the tool tip, seen in figure 3.6. Immediately after the motor encoder counts had been zeroed using the angle bar method, an SMR was placed in the tracker nest on each link. A point was taken in each of the four nests and stored. These four points would be used to get the arm back into the same initialization position before zeroing out the encoders.

Every time the robot was turned on, an SMR would be placed in the nest on the first link. The first joint would then be driven to within .001 inches of the stored initialization point for link one. Then the SMR would be put into the tracker nest on the second link. The second joint would then be driven to within .001 inches of the stored initialization point for link two. This same procedure was followed until link four (or the tool tip) had been driven into place. At that point, the motor encoder counts were all be zeroed and the arm had been initialized.

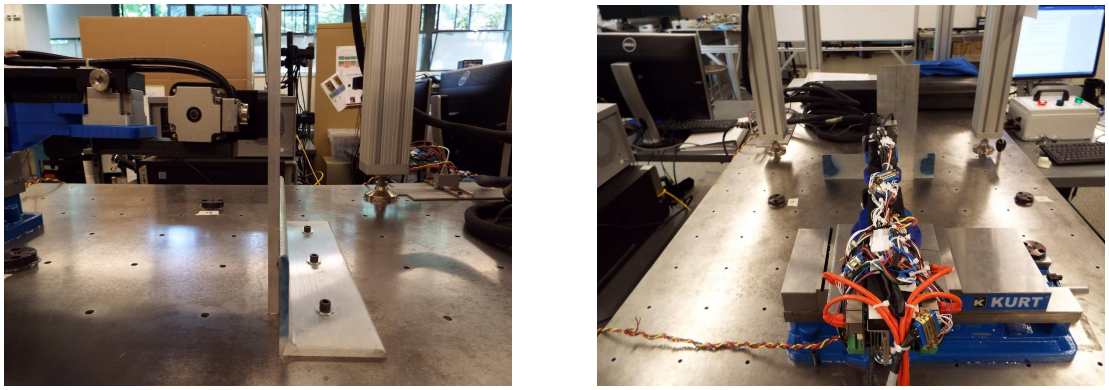

Figure 3.5: Left: Side view of the tool tip co-planar to the calibrated face. Right: Top view of the arm stretching out along the global X axis.

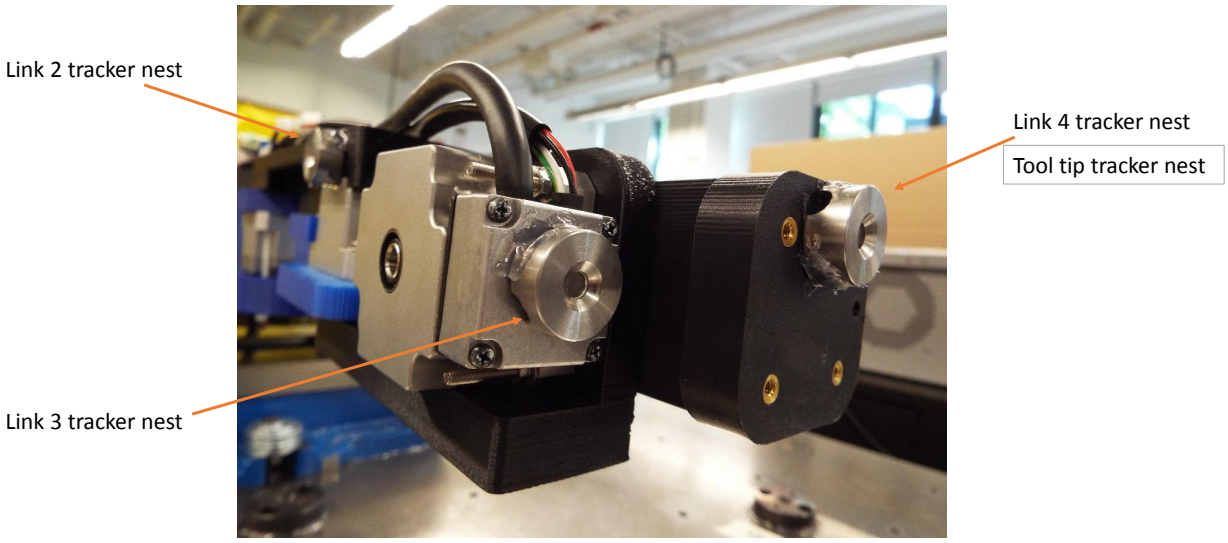

Figure 3.6: Three of the four tracker nests attached to each link of the robot arm

3.2.4 Measurement of Kinematic Parameters

One of the most important tasks was to measure the kinematic parameters. If we recall the setup of the forward kinematic equation in section 2.2, any errors in the kinematic parameters will cascade through math and show up as error between the commanded tool tip position and actual tool tip position. Consider for example, the forward kinematic equation developed for the single link robot shown in figure 2.1. If the true length of the link is 6.5 inches but the value for the link put in the forward kinematic equation is 6 inches, the reported position will always be off by at least .5 inches from the true position.

Kinematic parameters could have been measured from the CAD model, but this would be an idealized measurement ignorant of small deviations. Instead, the laser tracker was used to measure these kinematic parameters.

The kinematic parameters that had to be measured were:

- Distance between joint 1 and joint 2's center of rotation.
- Distance between joint 2 and joint 3's center of rotation.
- Distance between joint 3 and joint 4's center of rotation.
- Distance from the top of the base plate to the SMR tool tip nest, along the z axis.
- Offset from the tool tip nest to the z-x plane.

To measure the respective distances between each joint's center of rotation, an SMR was placed in the tool tip nest. Joint 1 was then commanded to rotate while the laser tracker stored points. Using the Spatial Analyzer software, a circle was fit to the data. The center of the circle and the normal to the circle were stored. This was done 5 time for joint 1. The average of the circle's center and normal were stored. This same procedure of commanding the joint to rotate, and finding the average of the circle's center and normal, was done for all four joints.

To find the distance between centers of joint rotation, the difference was calculated between the center of each circle. To find the height from the base plate to the tool tip nest, the difference between the plane of the circle fit to the data from joint 1 and the x-y plane was used. To find the offset from the tool tip nest to the z-x plane, the difference between the plane of the 4th axis of rotation and the z-x plane was used.

These measurements were made to within the accuracy of the laser tracker, approximately 10^{-5} inches. However, due to the relative flexibility of the mechanical components of the robot, these parameters would change with the pose of the arm. This would be one of the major contributors to tool tip error.

3.3 Tests to Help Isolate Sources of Error

To isolate the different sources of error, causing the measurements from the laser track to deviate from the predicted values, three different tests were preformed. First, the standard deviation of the laser tracker itself was measured. This was predicted to be the smallest

contributor of error. Second, the standard deviation of the robot in steady state servoing was measured. This was predicted to be only slightly larger than the tracker deviation. Finally, the baseline accuracy and repeatability of the robot arm as a whole was measured. This was predicted to show an appreciable contribution to error.

3.3.1 Standard Deviation of the Laser Tracker

To measure the standard deviation of the laser tracker, an SMR was placed in one of the stationary tracker nests. With the SMR sitting untouched in the tracker nest, 1000 points were taken. The SMR sitting in the stationary tracker nest is seen in figure 3.7. A standard deviation on the order of 10^{-5} inches was measured in x, y and z, seen in table 3.1.

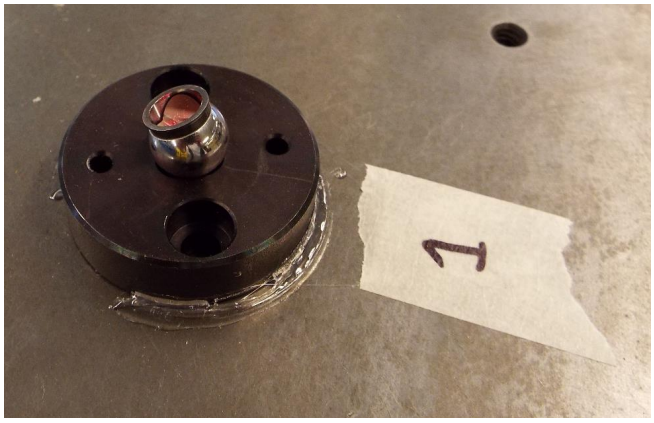

Figure 3.7: SMR sitting in tracker nest to determine standard deviation of the laser tracker

Standard Deviation in X (in)	Standard Deviation in Y (in)	Standard Deviation in Z (in)
8.8317E-05	6.12302E-05	3.1469E-05

Table 3.1: Table of the standard deviation of the laser tracker in x, y and z.

As predicted, the error from the laser tracker is small at around 10^{-5} inches. Further tests in this section will show that the tracker error is the smallest contributing error.

3.3.2 Standard Deviation of the Robot in Steady State Servoing

To get a measurement of the standard deviation of the servos in steady state using the Inverse Jacobian control scheme, the robot was set to drive to a desired position. The robot was allowed to move towards this position for over an hour, ensuring it was close to steady state. An SMR ball was then placed in the tool tip tracker nest. With the robot controller operating, the laser tracker then recorded a point every 2 seconds. It recorded 513 points over 1026 seconds. The standard deviations were on the order of 10^{-4} inches in x, y and z, seen in table 3.2.

Standard Deviation in X (in)	Standard Deviation in Y (in)	Standard Deviation in Z (in)
0.000339	0.0001837	0.0001187

Table 3.2: Table of the standard deviation of the the servos in steady state.

The steady state robot servo error had a standard deviation one order of magnitude larger than the laser tracker. The following section will show that this error from the motors servoing is still small with respect to the baseline accuracy and repeatability tests.

3.3.3 Baseline Accuracy and Repeatability of the Robot

The purpose of this test was to establish the best possible accuracy and repeatability with this particular electro-mechanical system. The best way to do this was to use the PI loop on the motor drives to command positions to the robotic arm. This would create a baseline for the accuracy and repeatabliltiy of the system with out the Inverse Jacobian control.

The commands in this test were taken care of by the PI control scheme on Elmo Motion Control's motor drives, called Gold Twitters. These drivers could hold each motor to within a couple counts. Joint 1 had a motor with an incremental encoder with 240,000 counts/rev. The other 3 joints had a motor equipped with an incremental encoder with 800,000 counts. That means if the motor could consistently hold its position to within ± 5 counts from its

commanded position:

- Joint 1 could hold the tool tip to $\pm.0026$ inches over the 20 inch span to the tool tip.
- Joint 2 could hold the tool tip to $\pm 5.12 \times 10^{-4}$ inches over the 13 inch span to the tip.
- Joint 3 could hold the tool tip to $\pm 2.75 \times 10^{-4}$ inches over the 7 inch span to the tip.
- Joint 4 could hold the tool tip to $\pm 1.26 \times 10^{-4}$ inches over the 3.2 inch span to the tip.

For this test, an SMR was placed in the tool tip. The robotic arm had been initialized according to the process outlined in subsection 3.2.3. An angle of +22.5 degrees was converted into encoder counts. Joint 1 was commanded to move to this encoder position. Once in position, a point was taken with SMR on the tool tip. Then, joint 1 was commanded to move to 0 degrees, where a point was taken. Followed by a command to -22.5 degrees, where a point was taken. The arm was then commanded to go back to 0 degrees and the entire test was repeated. This was done until there were 12 point taken at +22.5, 0 and -22.5 degrees.

This same test was preformed on joints 2 through 4. The data for joint 2 can be seen in in figure 3.8. The rest of the charts on the baseline tests can be found in appendix A. It is important to note that the commanded position seen in figure 3.8 was calculated by inputting the joint angles in the forward kinematics equation for this robot. This means the commanded position contains the error from the measurements of the kinematic parameters.

Figure 3.8: Top Left: Measured and nominal command to +45 degrees. Top Middle: Measured and nominal command to 0 degrees. Top Right: Measured and nominal command to -45 degrees. Bottom: Measured X-Y error from commanded position.

The error from the commanded position in X and Y was on the order of 10^{-2} to 10^{-3} inches. The standard deviation was on the order of 10^{-3} to 10^{-4} inches. Due to the relative magnitude of the error observed in the laser tracker and the servos, the majority of the error observed in this baseline test can likely be attributed to hysteresis in the mechanical system as well as inaccuracies in the measurements of the kinematic parameters.

3.4 Tests to Elucidate Inverse Jacobian Control Attributes

To study the characteristics of the Inverse Jacobian control on the four DOF confined space robot, two different tests and one simulation were designed tested on the robots. The two different tests and the simulation are:

- Transient motion and steady state error through the robot's field of motion.
- Simulation of position error from a single inverse Jacobian calculation.
- Variable spatial update rate of the Inverse Jacobain.

The following subsections describe the tests and present the data from each test.

3.4.1 Transient Motion and Steady State Error Through the Robot's Field of Motion

The main goal of this study was to examine how well the robot would track a line through different areas in the robots field of motion, as well as examining the difference between final tool tip position and desired position. The motivation for looking at the line tracking and position error in different regions of the robot work space comes from the Inverse Jacobian control scheme dependency on the pose of the arm. This field of motion test focuses on two different areas of the robot work space, which are illustrated in figure 3.9. The two sets of trajectories seen in figure 3.9 are very close to the motion that the confined space robot may be required to perform. These trajectories cover the scenario of the robot in a folded pose (near trajectory), as well as in a stretched out pose (far trajectory). It also looks at trajectories close to the edge of the work space (which is close to a kinematic singularity), as well as trajectories that drive the tool tip close to hitting it's own links. A description of the test is written below.

The robot was moved in to the initial pose, depicted in figure 3.9. The corresponding encoder values were written down and stored. Storing the encoder values from this initial pose allowed the test engineer to drive the robot back to this same pose to within the baseline accuracy. Simple joint level commands (excluding the Inverse Jacobian control) were used

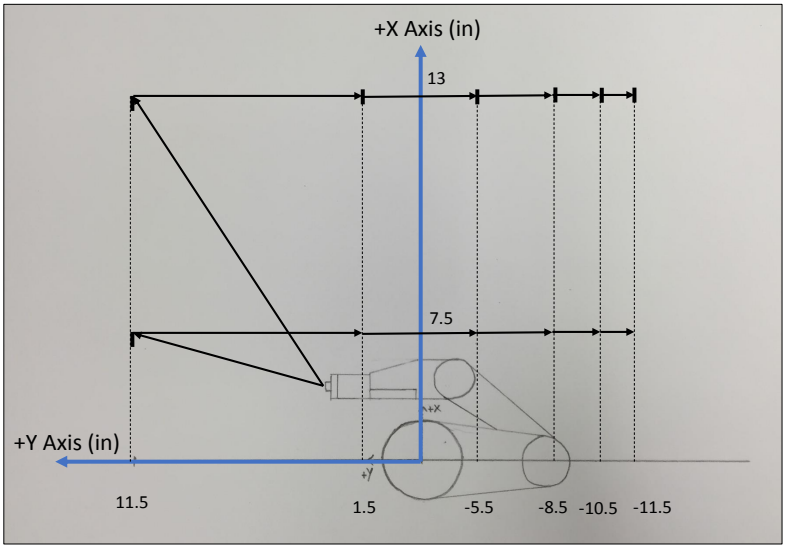

Figure 3.9: Field of motion, path schematic

for returning to this pose.

Once in this starting pose, an SMR ball was placed in the tool tip tracker nest. The laser tracker was set to output a position every 0.01 inches, as well as a corresponding time stamp. This allowed to track the transient motion of the system, as well as record the steady state errors.

The inverse Jacobian control was initialized and the robot was commanded to move to the top left point seen in figure 3.9. The robot's Inverse Jacobian control scheme was allowed to run until a point was not output by the laser tracker within 10 seconds. This meant that the tool tip had moved less than .01 inches in 10 seconds. If a data point was not output within 10 seconds, the the robot was commanded to move to the next point down the line, with the same stipulation for run time. This was done until the robot reached the final point seen in the top right of figure 3.9. Finally, the robot was commanded from a joint level (not

using the inverse Jacobian control) back into its starting pose. This process was repeated until each point was driven to eight times. The same procedure was also done for the second set of trajectories closer to the robot.

Figure 3.10 shows the measured and simulated relative x-position vs. time. This relative x-position is the commanded x-position minus the current x-position. The data represented by the blue dots in figure 3.10 was manipulated from the raw data. Each point on the chart is an average of the x-position from each of the 8 trials. To be able to average the x-position of each point, the points had to be at the same time intervals. The raw data did not have each point taken at the same time interval. So, the time stamps of the data set with the least amount of data points was taken. Then the remaining 7 data sets had their x-positions interpolated to the same time values of the base data set. This allowed us to use the average of the x-positions.

Figure 3.10 follows a similar layout to the rest of the charts in the field of motion test. The top row of figure 3.10 corresponds to the far field trajectories seen in figure 3.9. The bottom row corresponds to the near field trajectories seen in figure 3.9. It is important to note the scale on the x-axis of figure 3.10 is on the order of 10^{-2} inches for each chart, except for the two graphs in the first column. That is because the change in x was large for the first move and then close to zero for the rest of the trajectories.

The red line is the simulated line, or what the robot should theoretically do given the same inputs. We might assume that the simulated lines in figure 3.9, should start at zero inches in x and end at zero inches in x, since there should be no change in the commanded x-position after the first move. This would be true, but as noted above, the robot did not have infinite time to reach its goal. Therefore, the start of its new trajectory was the unfinished end of its last trajectory. The simulated line on each chart takes this unfinished trajectory into account by looking at the base data set's (data set that the other 7 data sets were interpolated against) elapsed time before the control scheme was cut off. The simulation was then run for the same amount of time and the final position of the simulation was used as the initial position for the next simulation.

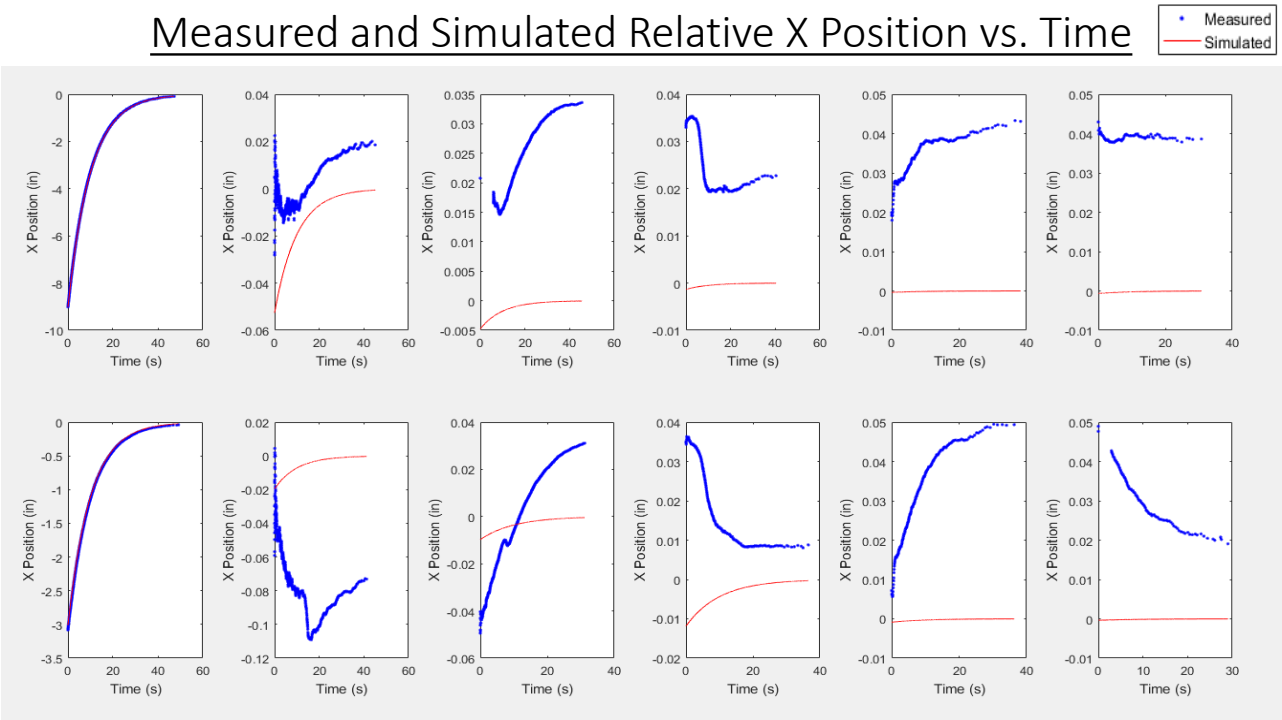

Figure 3.10: Top Row: x position vs time for each of the far field trajectories. Bottom Row: x position vs time for each of the near field trajectories. The individual charts corresponding final position matches up with the points in figure 3.9

The deviations from the simulated trajectory likely stem from the discrepancies in the measurements of the kinematic parameters, as well as hysteresis in the mechanical system. The discrepancies in the kinematic parameters would explain the difference in the shape of the measured trajectory and the simulated trajectory. Although both systems were given the same kinematic parameters, the real system has variations from the prescribed parameters. These would lead to the tool tip following a different path. The deviation from the trajectories are on the order of 10^{-2} inches, which is the baseline errors reported in section 3.3.3.

Figure 3.11 is the same as figure 3.10, except that it is reporting the measured and simulated relative y position vs time, rather than the x position vs time. The same description explained for figure 3.10, is true for figure 3.11. The most important difference between the two charts is the scale of the y-axis is much larger. This is due to the relative command in

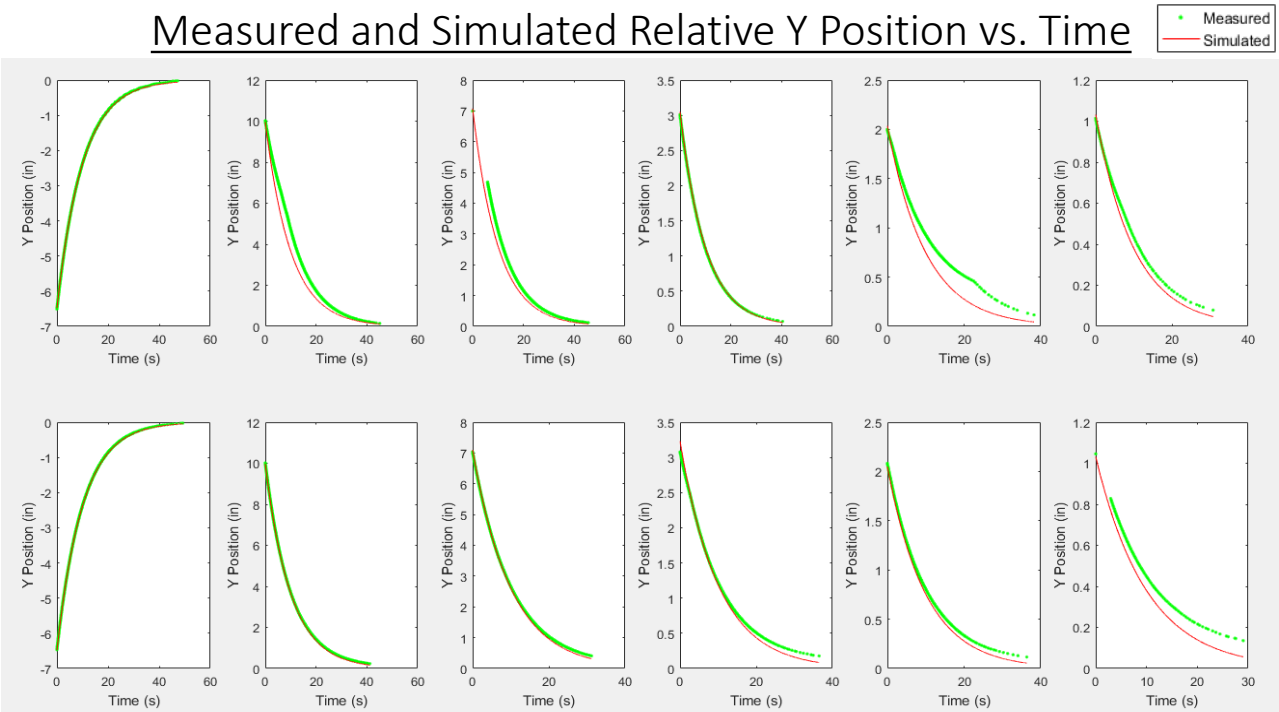

Figure 3.11: Top Row: y position vs time for each for far field trajectories. Bottom Row: y position vs time for each for near field trajectories. The individual charts corresponding final positoin matches up with the points in figure 3.9

y being much larger. Deviations from the simulated trajectory were on the same order as that for the x-axis, 10^{-2} inches. The same sources of error apply to the y values as the x values. The data shows a small tendency for y errors to be larger than x errors. This may have to do with the offset of the tool nest from the x-z plane along the y axis, as being the most difficult kinematic parameter to measure.

Figure 3.12 shows the average of the error between the commanded point and the measured final point. It should be noted that in figure 2.4 the forward kinematics are responsible for creating what the control scheme sees as the current position. This means any errors in the kinematic parameters will affect the final tool tip position error. That being said, the errors of the steady state positions, shown in figure 3.12 are on the same order as the baseline errors reported in subsection 3.3.3, that is 10^{-2} inches.

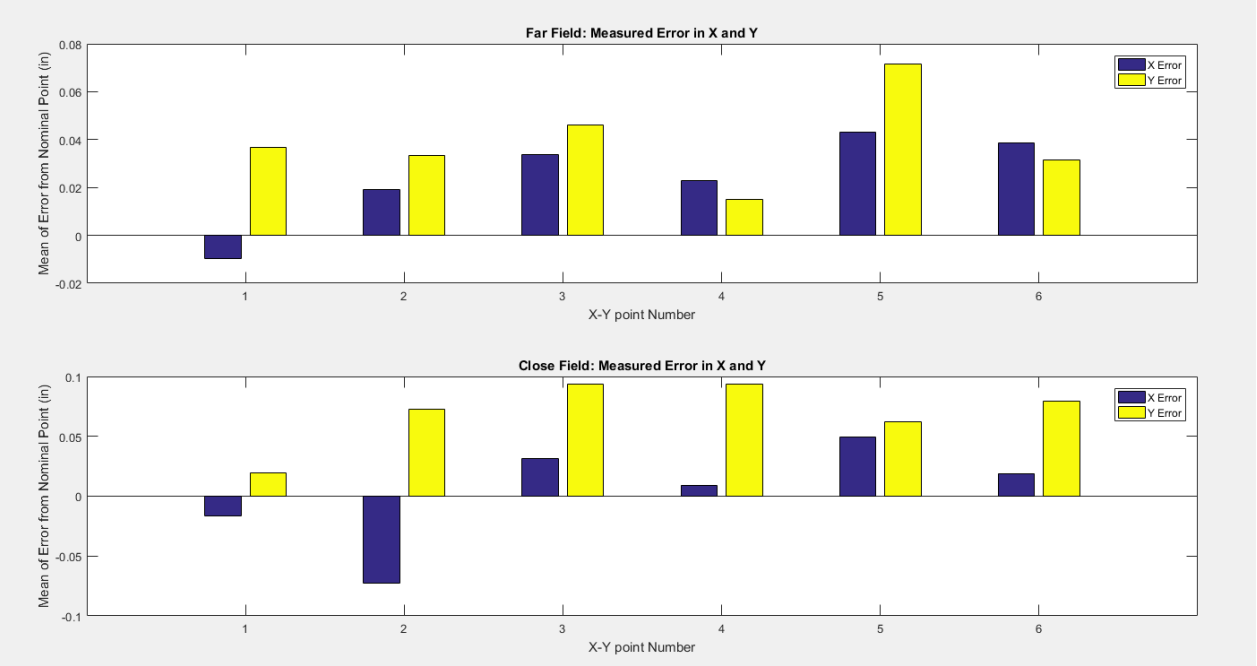

Figure 3.12: Top: Average of the steady state error of the far field trajectories in x and y. Bottom: Average of the steady state errors of the near field trajectories in x and y.

The field of motion steady state errors are on average a little larger than the baseline errors. The source of the error likely has to do with the the control scheme itself and how it was implemented. Joint angle velocity commands are output from the control scheme based on position errors. This means the current position will only reach the desired position after an infinite amount of time. The control scheme had to be cut off at some point, since there was not an infinite amount of time to run these tests. This stipulation was mentioned above as the control scheme being cut off if after 10 seconds a point was not output by the laser tracker. Cutting the control scheme off doesn't allow the tool tip to reach the desired location and will result in an error between desired position and current position.

3.4.2 Simulation of Position Error from a Single Inverse Jacobian Calculation

The goal of this simulation is to examine the error si between the calculated tool tip position and the desired position, when populating the Inverse Jacobian matrix using only the set of joint angles corresponding to the initial position. That is, we determine how close a single calculation of the inverse Jacobian will get the tool tip to the desired position. The calculated position should become more accurate as the desired position is moved closer to the initial tool tip position. This stems from the spatial dependency of the Jacobian matrix mapping joint velocities to Cartesian velocities.

This simulation focused on the mathematics of the Jacobian itself, rather than the Inverse Jacobian control scheme. Starting with equation 2.12, which is restated below, Δt was multiplied on both sides. As well, ΔP was broken out into the difference between the desired position and the initial position, seen in equation 3.2.

$$\frac{\Delta\theta}{\Delta t} = J(\theta)^{\dagger}\frac{\Delta P}{\Delta t} \tag{3.1}$$

$$\Delta\theta = J(\theta)^{\dagger}(P_{desired} - P_{initial}) \tag{3.2}$$

The Inverse Jacobian was populated with one static set of joint angles corresponding to the initial position and pose of the arm.

$$\theta_{Initial} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix} \tag{3.3}$$

Along with the initial joint angles, $P_{desired}$ was also specified. $P_{initial}$ could be calculated by inputting $\theta_{Initial}$ into the forward kinematics. $\Delta\theta$ was calculated from equation 3.2, using $P_{desired}$ and $P_{initial}$. $\Delta\theta$ was then added to the initial vector of joint angles, shown below.

$$\theta_{Calulated} = \begin{bmatrix} \theta_1 + \Delta\theta_1 \\ \theta_2 + \Delta\theta_2 \\ \theta_3 + \Delta\theta_3 \\ \theta_4 + \Delta\theta_4 \end{bmatrix} \tag{3.4}$$

$\theta_{Calculated}$ was fed through the forward kinematics to find $P_{Calculated}$. $P_{desired}$ was subtracted from $P_{Calculated}$. This difference between $P_{desired}$ and $P_{Calculated}$ is the target of the simulation.

The first desired position was input into the simulation as, [10, 8, 2.46] inches. The code executed as discussed above 100 times to find the calculated position based on closer desired positions. The error between $P_{desired}$ and $P_{Calculated}$ was recorded. A new desired position was determined following the equation below:

$$P_{desired} = P_{desired} - ((P_{desired} - P_{Initial})/10) \tag{3.5}$$

Where $P_{desired}$ on the left side of the equation is the new desired position and $P_{desired}$ on the right side of the equation is the prior desired position. $P_{Initial}$ is the tool tip position from the initial pose (it is the same on every iteration of the simulation). This would bring $P_{desired}$ closer to $P_{Initial}$, effectively reducing the magnitude of ΔP, but maintaining the direction.

Figure 3.13 shows the desired points in blue and the corresponding calculated points in red. Figure 3.14 plots the error between the desired position and the calculated position in inches, against the distance between the desired position and the initial position in inches.

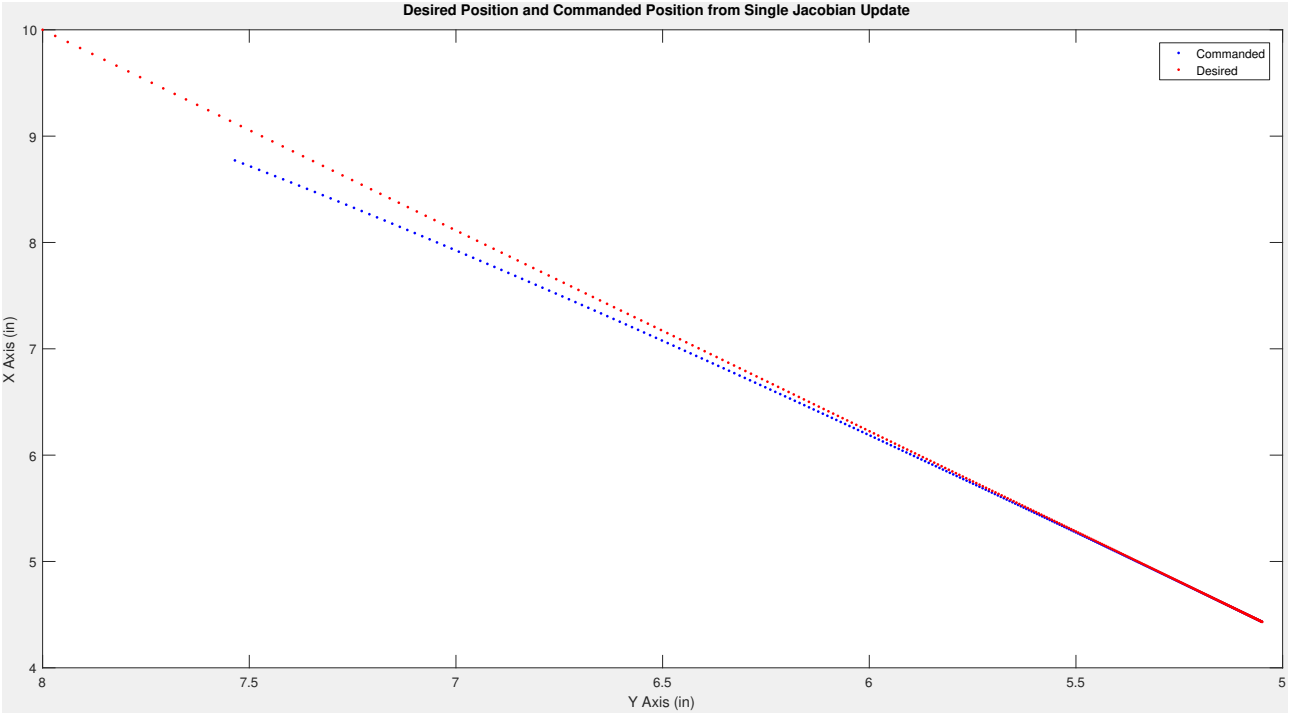

Figure 3.13: Plot of the desired positions and corresponding calculated positions. 100 iteratations are shown in the plot.

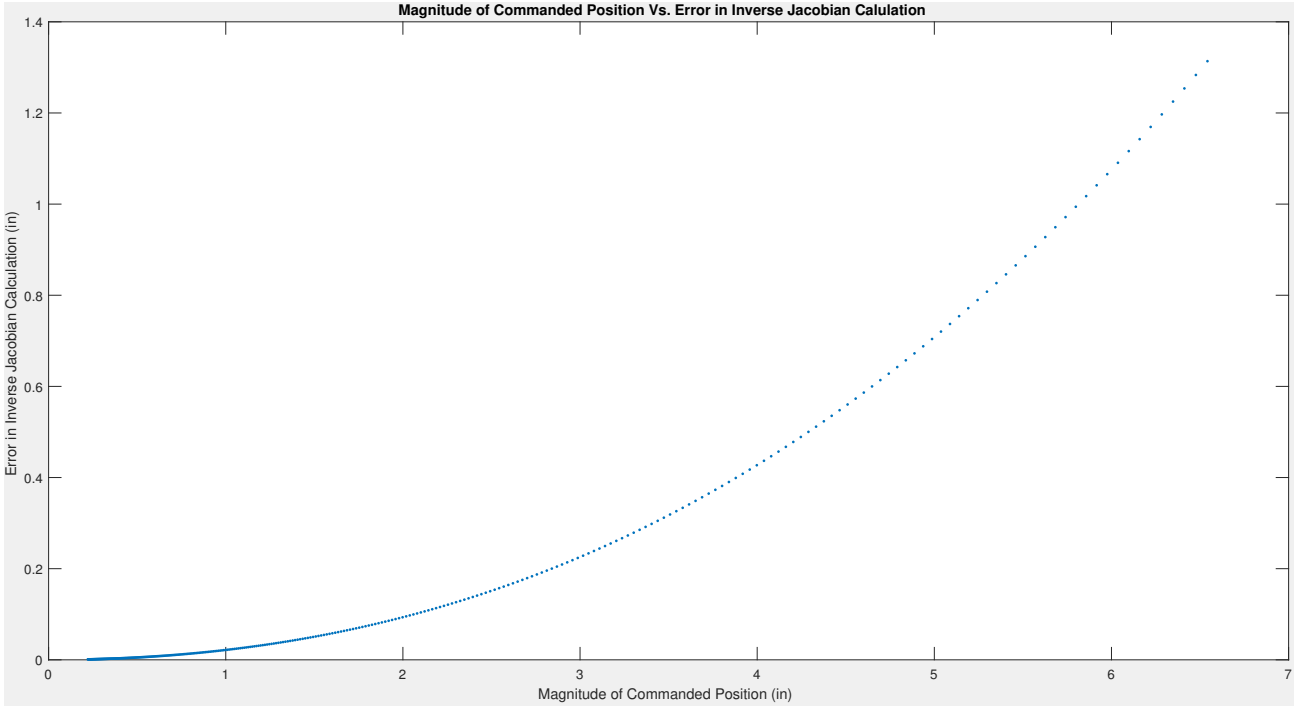

Figure 3.14: Magnitude of the delta position vector vs the error in the Inverse Jacobian calculation.

We can see that the error decreases as the the desired position gets closer to the initial position. This follows from the definition of the Jacobian matrix being an instantaneous map. What is surprising is how small the calculated error is as the desired position gets further from the initial position.

3.4.3 Variable Spatial Update Rate of the Inverse Jacobain

One of the main goals of this evaluation is to determine the nature of how the Inverse Jacobian tracks a line if the inverse Jacobian matrix is not continuously updated. To examine how the Inverse Jacobian line tracking depends on the rate at which the inverse Jacobian matrix is updated, the inverse Jacobian matrix update rate was varied while monitoring tool tip position.

At first we considered changing the update rate of the inverse Jacobian matrix based on time. However, since the change in the inverse Jacobian matrix values are dependent on space rather than time, a spatial update was used. The spatial update rate was based on displacement of the tip in Cartesian space. Alternatively, the Jacobian might be updated based on changes in the joint angles.

For the Field of Motion tests described in section 3.4.1, the inverse Jacobian matix was updated at the the frequency of the outer control loop, 100 Hz. The control scheme for this simulation was altered so that the update rate was based on the change in the Cartesian tool tip position. This was done by continuing to monitor the joint angles at the rate of the outer control loop, 100 Hz. Then, feeding those angles through the forward kinematics to find the new tool tip position. That tool tip position was subtracted from the tool tip position at the last update. If the distance from the current tool tip position to the previous tool tip position at the last Inverse Jacobian update was larger than the chosen spatial interval, the Inverse Jacobian was updated with the current joint angles. In this way, we could update the inverse Jacobian matrix based on the distance the tool tip traveled.

The largest Cartesian velocity was calculated given the initial and final points. This was then used to find out what the maximum spatial update rate was at 100 Hz. This turned out to be a 0.006 inches spatial update rate. This was the floor for how small the spatial update rate could be, given given the frequency of the outer control loop. The spatial update was then incremented by distances of .05 inches, then .1 inches and finally 1 inch, with a range from $0.006 - 6.0$ inches. The charts from three of these tests with spatial update rates of

0.006 inches, .50 inches and 6 inches are be shown below. The rest of the charts for this test can be found in appendix B.

This was one of the more interesting tests and really struck at the core of what the evaluation was about. First, it was surprising to see how large the spatial update rate could get before a sizable deviation from tracking the line was noticed. Even when the inverse Jacobian matrix was only updated every .5 inches, the deviations from the line were minimal.

Second, was that the Inverse Jacobian update had a larger impact on the steady state error than it did on tracking a line. The Inverse Jacobian may get close to the desired position but even if it passes over the desired position, the control scheme has no way of adapting the velocity of the tool tip until the scheme is updated. The tool tip was unable to converge on the desired position, continually hovering around it. This led to a "bouncing" effect around the desired location that was easier to observe as the spatial update got larger, see figure 3.17.

Another interesting observation was that the form of the first trajectory, before a spatial update occurred, followed the form seen in the simulation from section 3.4.2. This is exemplified in figure 3.17, where just before the first update, the trajectory followed a very similar path to the one seen in the simulation figure 3.13. The "bouncing" effect is at its worst when the inverse Jacobian matrix is updated every 6 inches, however the initial line tracking does surprisingly well.

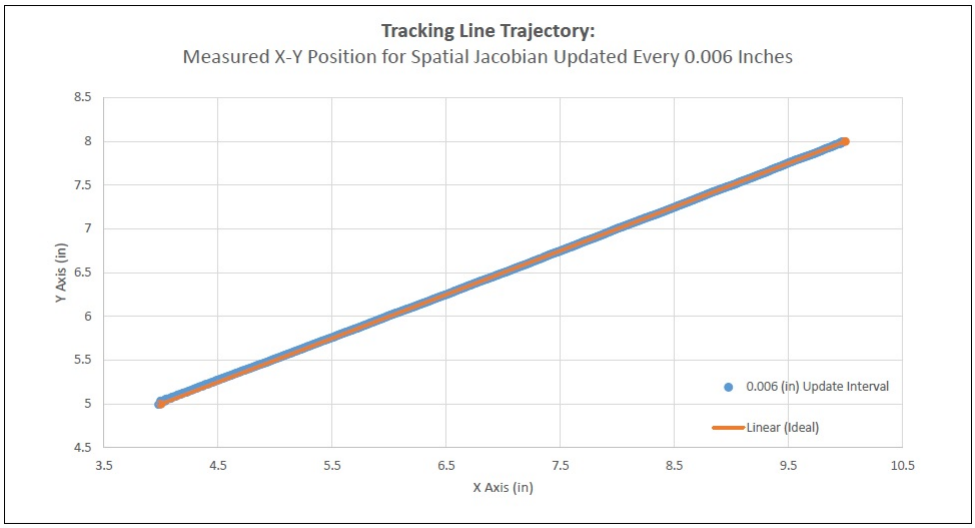

Figure 3.15: Chart of the smallest spatial update occurring every .006 inches

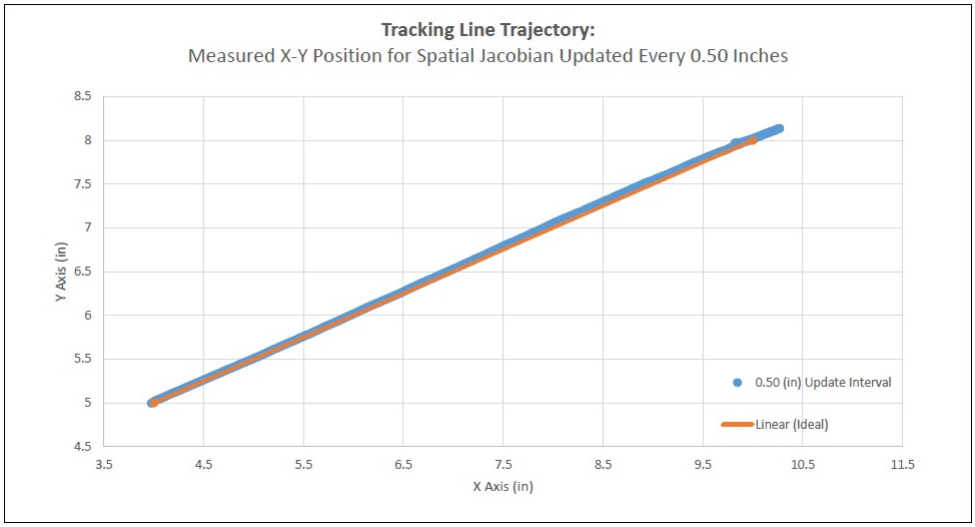

Figure 3.16: Chart of an intermediate spatial update occurring every .50 inches

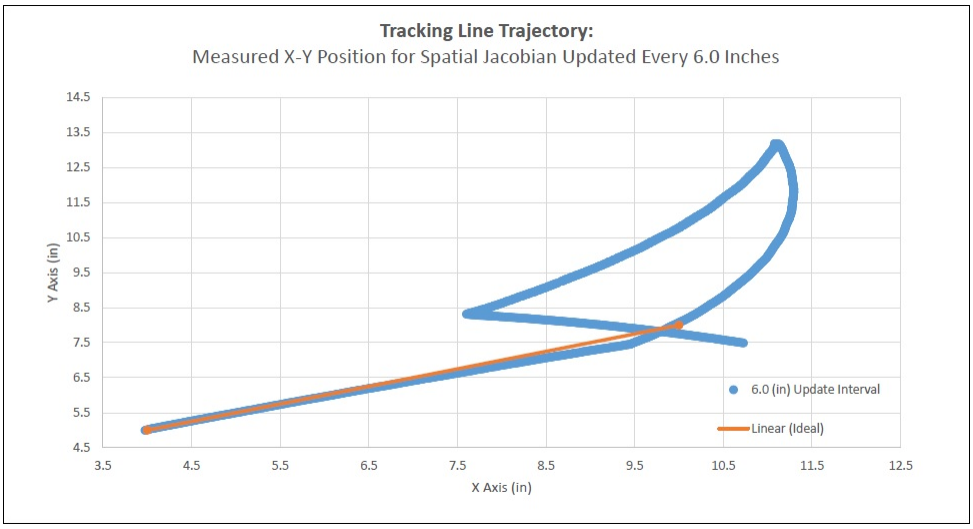

Figure 3.17: Chart of the one of the largest spatial update, occurring every 6.0 inches

Chapter 4

CONCLUSION

4.1 Conclusion: Test Setup and the Machine

The test setup was one of the most difficult parts of this evaluation. The robotic arm used for this study was a prototype that became the focal point of the Inverse Jacobian tests. For future robot designs, there are some areas of improvement that could have isolated errors, and made the data more meaningful. For instance, the three joints that rotated about their z-axis were cantilevered from the face of their respective motors. If the links captured the joints on the back of the motors, as well as the face of the motors, the system would have been stiffer.

Overall stiffer links and connections would allow for a more repeatable robot. This would lead to more more accurate kinematic parameters over the field of motion of the robot. If the robot can not be made stiffer, then it would be beneficial to use a laser tracker to build up a large compensation table of the kinematic parameters throughout the robot's field of motion.

The use of physical datum's on the robot for measuring kinematic parameters would have been useful. This would have given a baseline for more accurate kinematic parameters. Datums would also have been very useful in setting up the initialization routine of the encoder counts. A machined apparatus could have been used to contact the datums and zero the encoder counts closer to the mathematical zero angles.

Using a larger SMR ball with a greater field of view would have allowed for more flexibility in the tests. If possible, use of the largest SMR ball for the greatest field of view would have been valuable.

The standard deviation of the laser tracker was approximately two orders of magnitude

smaller than the standard deviation of the objects being measured. This was sufficient for the tests outlined in this thesis. The laser tracker was used for temporal tests, as well as spatial tests. However, the tracker and its software are primarily designed for spatial measurements. It preformed well for the temporal tests, but it proved difficult to make the laser tracker output accurate temporal information. For temporal tests with required sampling greater than 10 Hz, a different measurement system should be considered.

The harmonic drive FHA motors proved to be very reliable and contribute errors on the order of 10^{-4} inches for the 100:1 geared motors and 10^{-3} inches for the 30:1 geared motors. The standard deviation of the steady state robot servo error was very small with respect to the baseline errors, at approximately 10^{-4} inches.

A final consideration would be to fully automate as much of the test as possible. Running the tests manually became time consuming and tedious to get the necessary amount of data.

As mentioned previously the greatest contributor to error was the lack of stiffness and hysteresis in the robotic arm. This led to a lack in ability to measure the kinematic parameters, as well as a greater dynamic change in the kinematic parameters throughout the robots field of motion.

4.2 Conclusion: Inverse Jacobian Control

The Inverse Jacobian control as implemented on this four DOF robotic arm was an adequate method for solving the Inverse Kinematics problem for the four OF confined space robot. This method allowed for the robot to move in a predictable trajectory towards a desired tool tip position. The characteristics of the Inverse Jacobian control scheme remained consistent over the area of the robot work space tested in the field of motion tests. However, this particular four DOF arm has only one type of singular pose, which is anywhere the arm is stretched out straight. The Inverse Jacobian control's most blatant issue is the break down in control around kinematic singularities. A more complex robot may have more kinematic singularities and greater deviation in the control output over it's field of motion.

The final error of the tool tip with respect to the desired location was on the order of

10^{-2} inches. This would have been smaller if the joint velocity limits had been set higher, allowing higher Jacobian control gains, leading to shorter settling times.

The simulation in section 3.4.2 showed that even if the desired position is 6 inches away from the current tool tip position, one calculation of the Inverse Jacobian will get the tool tip to within approximately 1.25 inches from the desired position. This error in final simulated position from the desired position, decreases approximately exponentially with respect to the distance from the initial tool tip to the desired location. This simulation was only done in one area of the four DOF robot's field of motion. This correlation may change in the presence of a singularity, or over the field of a more complex robot.

The Inverse Jacobian control tracked a line better than expected for large spatial update rates, on the order of tenths of an inch. However, the bouncing effect around the desired position, for update rates larger than 10^{-2} inches, was observable to the naked eye and shed light on an attribute of the Inverse Jacobian control that should be researched more.

The Inverse Jacobian control for a serial linked confined space robot provided desirable control characteristics. The tool tip settled at the desired position to within 10^{-2} inches and could have been faster and more accurate given a larger gain. As well, the tool tip tracked a line, even for larger update rates on the order of 10^{-1} inches. The control scheme has areas that require more research, such as the bouncing effect, which will be discussed in the following section.

4.3 Conclusion: Suggested Future Work

There is a great deal of work that was untouched due to lack of time. The rotational aspect of the Inverse Jacobian was left out. Rotations are less intuitive and more involved, mathematically. Insights on the rotational component of the Inverse Jacobian control scheme could be gained by running similar tests to those run on the linear portion. This would highlight how the nonlinear aspects of the Jacobian interact with the attributes of the rotational Inverse Jacobian control.

More research should be done into how this "bouncing" effect shows up on a smaller

scale. This effect may be correlated with the gain on the Inverse Jacobian control scheme. If high tolerances are required for the steady state error of the tool tip, the bouncing effect may disrupt the accuracy.

The sensitivity of the Jacobian for this particular robot was relatively benign. These tests could be re-run on a robot with more degrees of freedom, or a set of joints more susceptible to the nonlinear effects of the Jacobian.

A finding that was left out of this document, due to it being an initial mistake in a simulation, was that the Inverse Jacobian does not track a straight line if dynamics are involved. Since any real machine will have dynamics, it could be interesting to see if these dynamics could be accounted for to correct for a straight line trajectory.

Finally, the Inverse Jacobian using a least squares solution is just the base foundation of a class of control schemes. One notable variation of the Inverse Jacobian is the Extended Jacobian method [3], which allows the user to specify a desired final pose, as well as final tool tip position. This could be very useful if the pose of the arm is of any concern to the application. As well, there are methods that attempt to minimize energies, rather than simply kinematic qualities. One method, minimizes the kinematic energy to get more natural trajectories.

BIBLIOGRAPHY

[1] Samuel R. Sr Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *University of California, San Diego, Typeset manuscript ...*, 132(4):1–19, 2004.

[2] Jacob Hannaford, Blake Rosen. *Robot Manipulators.* 2013.

[3] Stefan Schaal. Jacobian methods for inverse kinematics and planning Slides from Stefan Schaal The Inverse Kinematics Problem, 2006.

Appendix A

CHARTS FROM THE BASELINE ACCURACY AND REPEATABILITY OF THE ROBOT

Below are the remaining charts for each joint in the baseline accuracy and repeatability tests. See subsection 3.3.3 for more information.

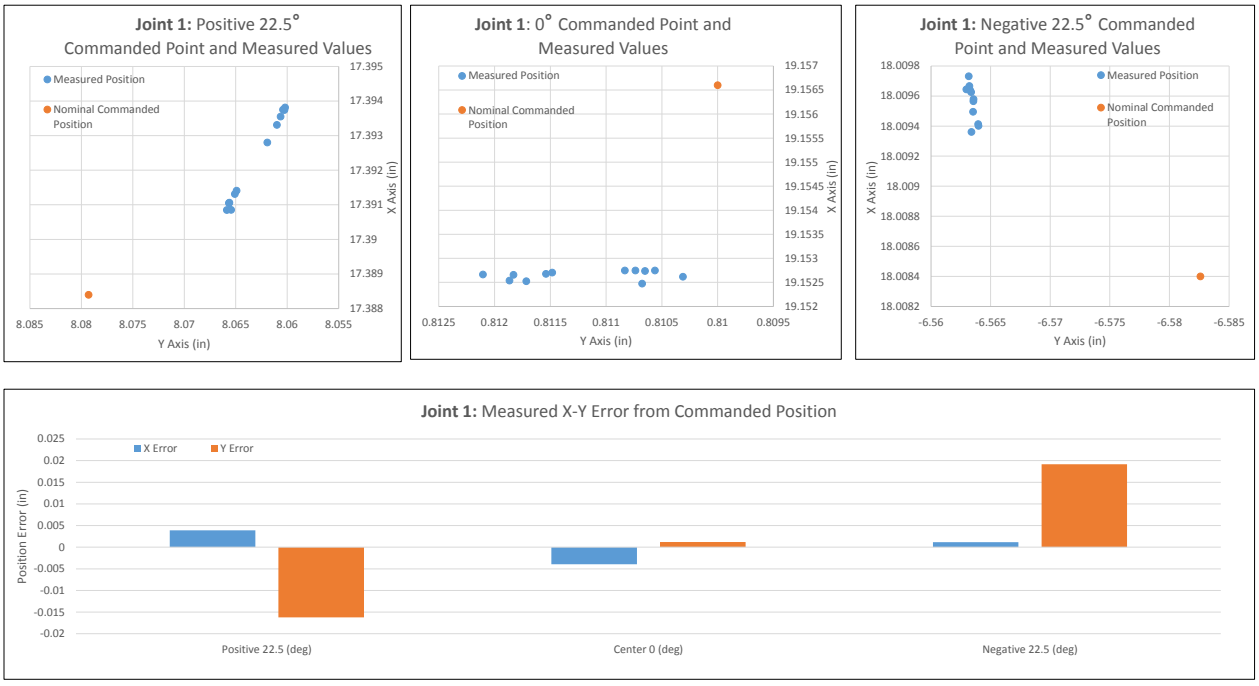

Figure A.1: Top Left: Measured and nominal command to +22.5 degrees. Top Middle: Measured and nominal command to 0 degrees. Top Right: Measured and nominal command to -22.5 degrees. Bottom: Measured x-y error from commanded position.

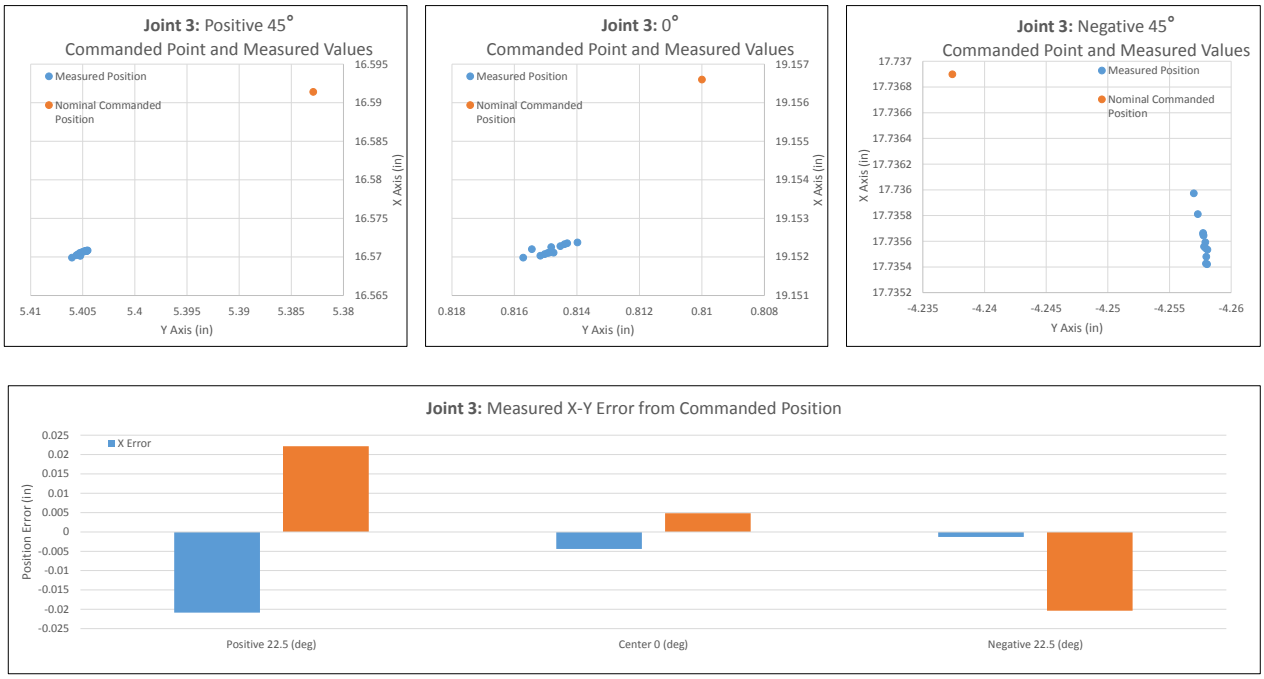

Figure A.2: Top Left: Measured and nominal command to +45 degrees. Top Middle: Measured and nominal command to 0 degrees. Top Right: Measured and nominal command to -45 degrees. Bottom: Measured x-y error from commanded position.

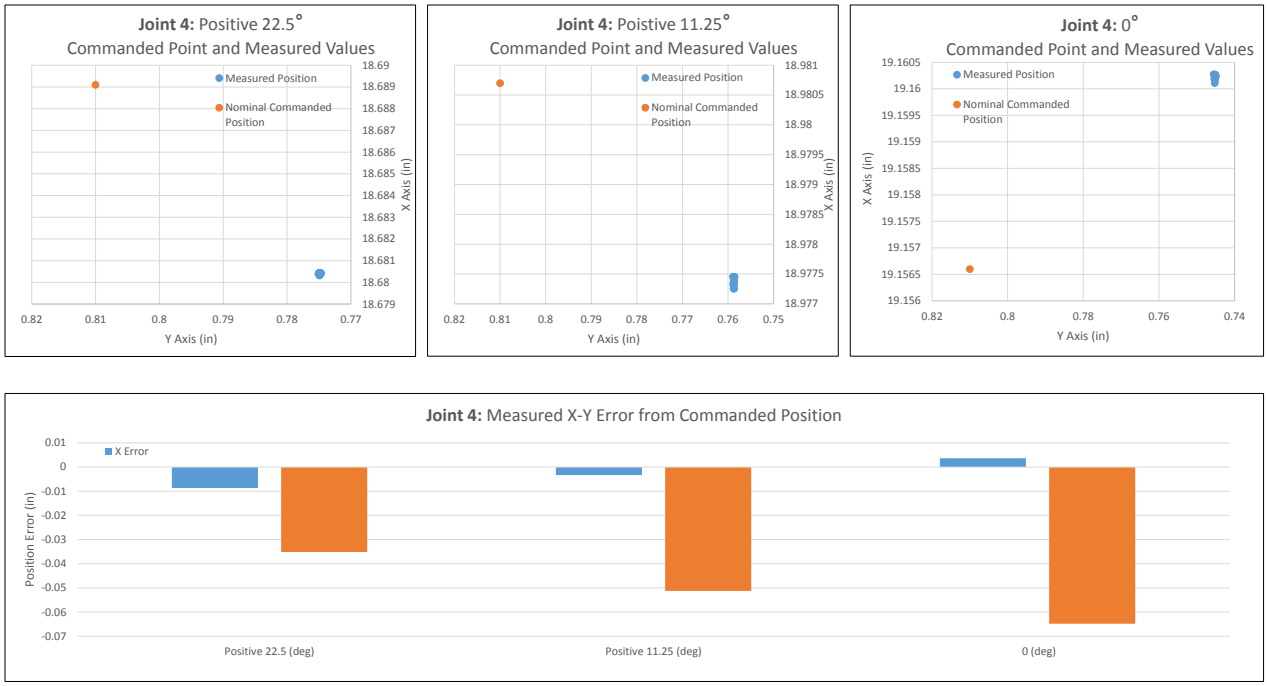

Figure A.3: Top Left: Measured and nominal command to +22.5 degrees. Top Middle: Measured and nominal command to 11.25 degrees. Top Right: Measured and nominal command to 0 degrees. Bottom: Measured x-y error from commanded position.

Appendix B

CHARTS FROM THE VARIABLE SPATIAL UPDATE RATE OF THE INVERSE JACOBAIN

Below are the remaining charts from the variable spatial update rate tests of the Inverse Jacobain. For more information, see subsection 3.4.3.

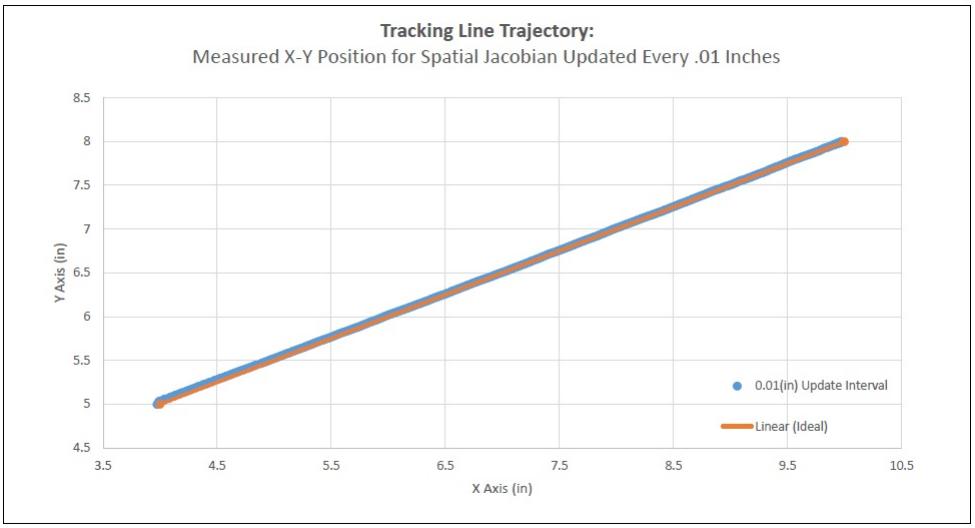

Figure B.1: Chart of the spatial update occurring every .01 inches

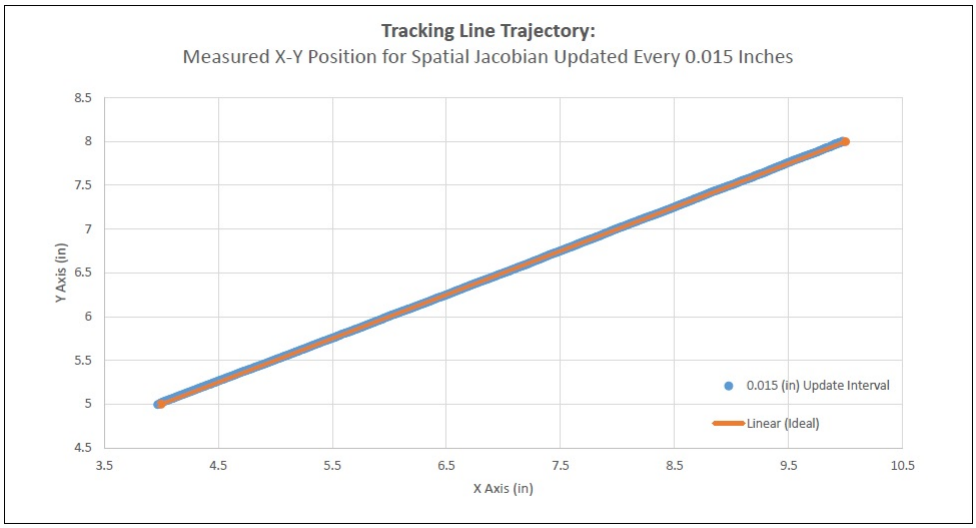

Figure B.2: Chart of the spatial update occurring every .015 inches

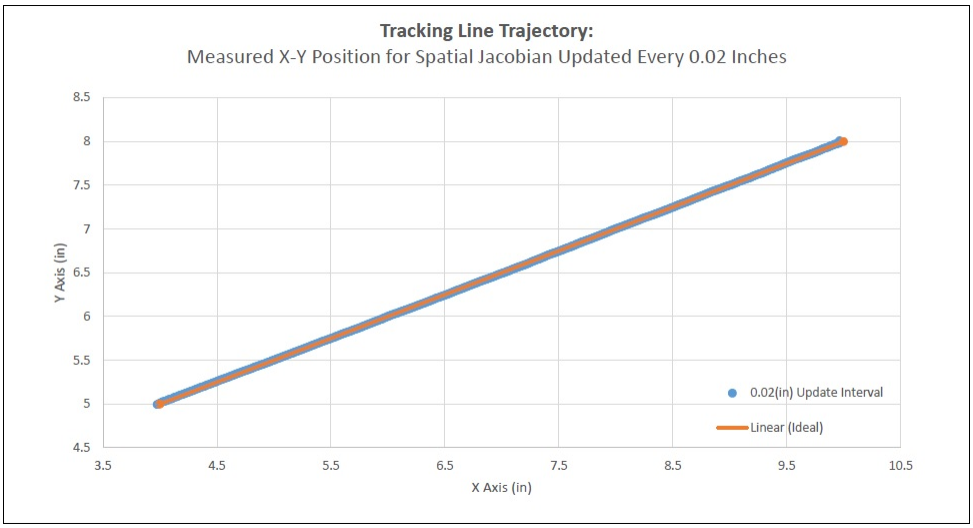

Figure B.3: Chart of the spatial update occurring every .02 inches

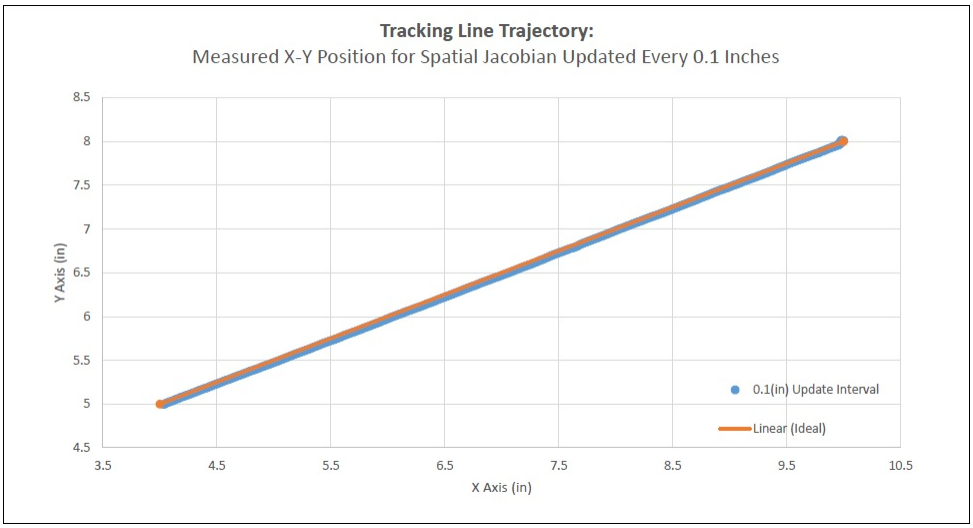

Figure B.4: Chart of the spatial update occurring every .10 inches

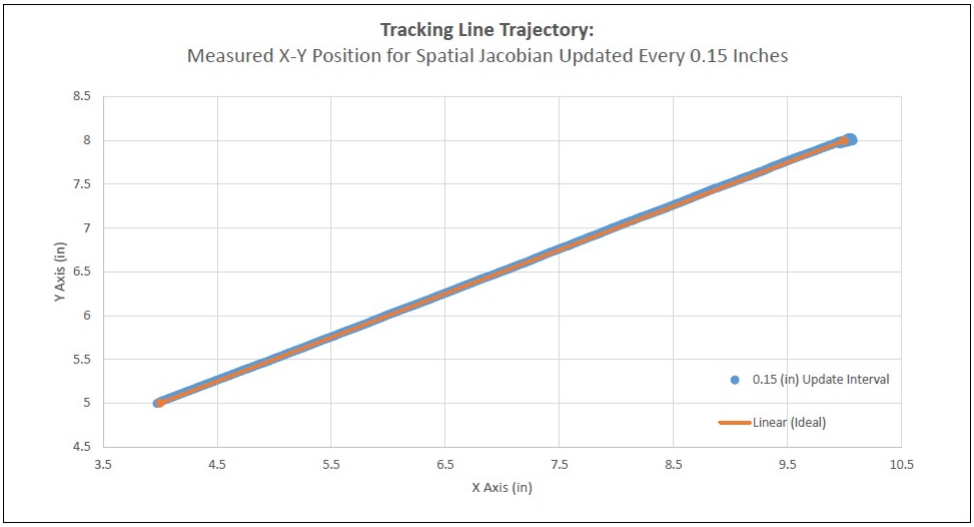

Figure B.5: Chart of the spatial update occurring every .15 inches

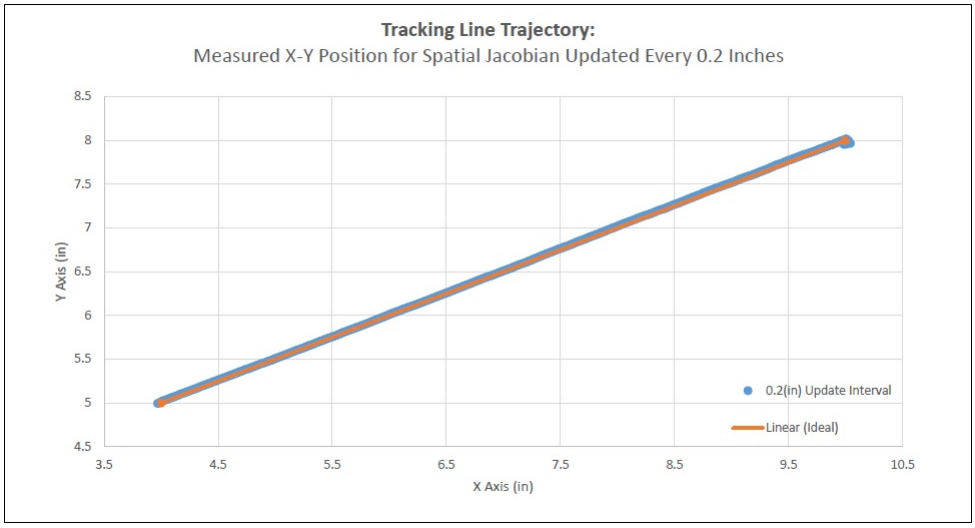

Figure B.6: Chart of the spatial update occurring every .20 inches

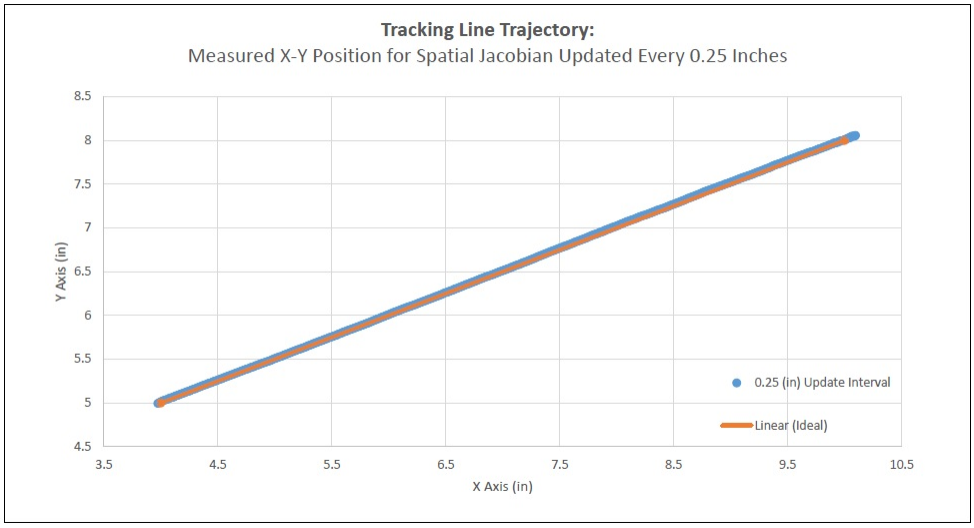

Figure B.7: Chart of the spatial update occurring every .25 inches

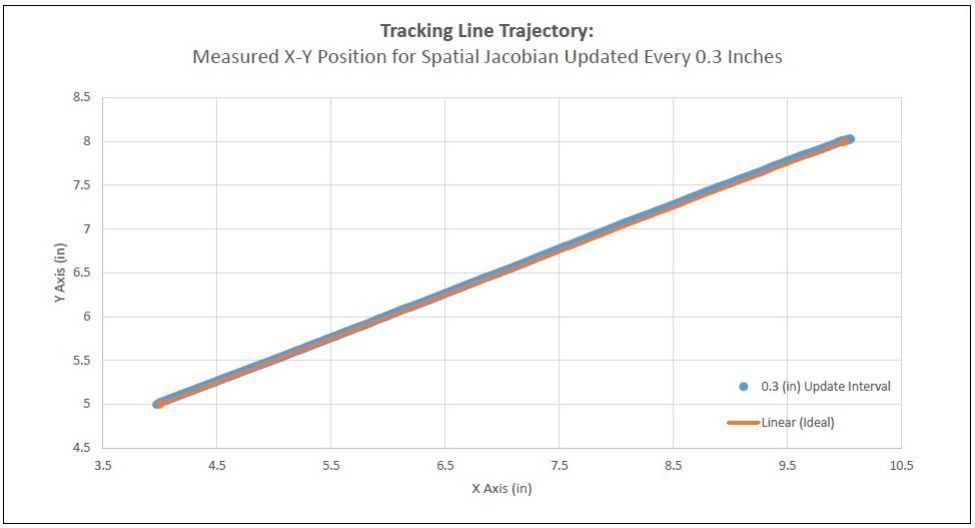

Figure B.8: Chart of the spatial update occurring every .30 inches

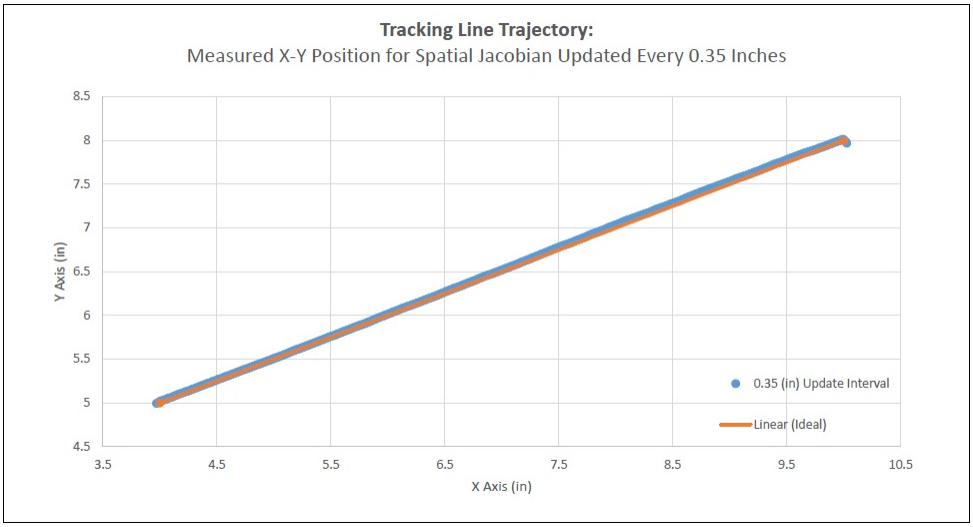

Figure B.9: Chart of the spatial update occurring every .35 inches

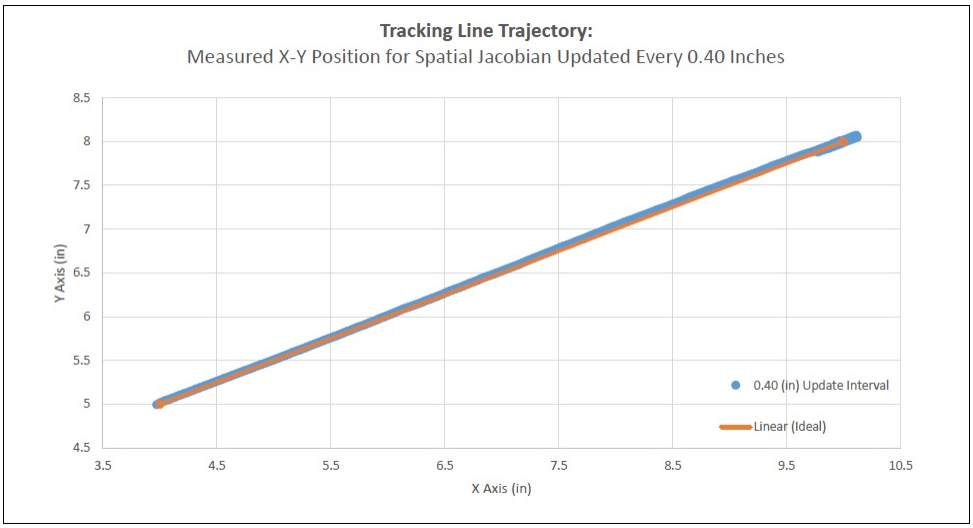

Figure B.10: Chart of the spatial update occurring every .40 inches

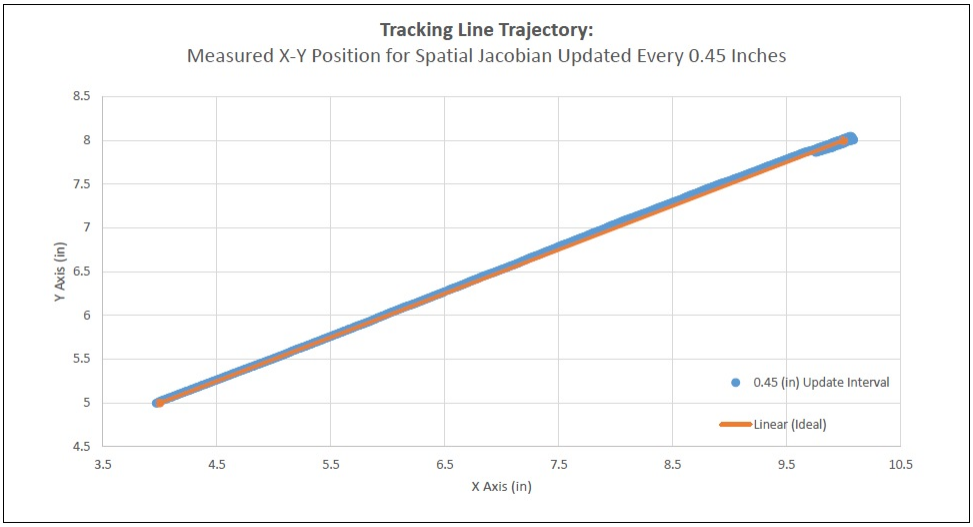

Figure B.11: Chart of the spatial update occurring every .45 inches

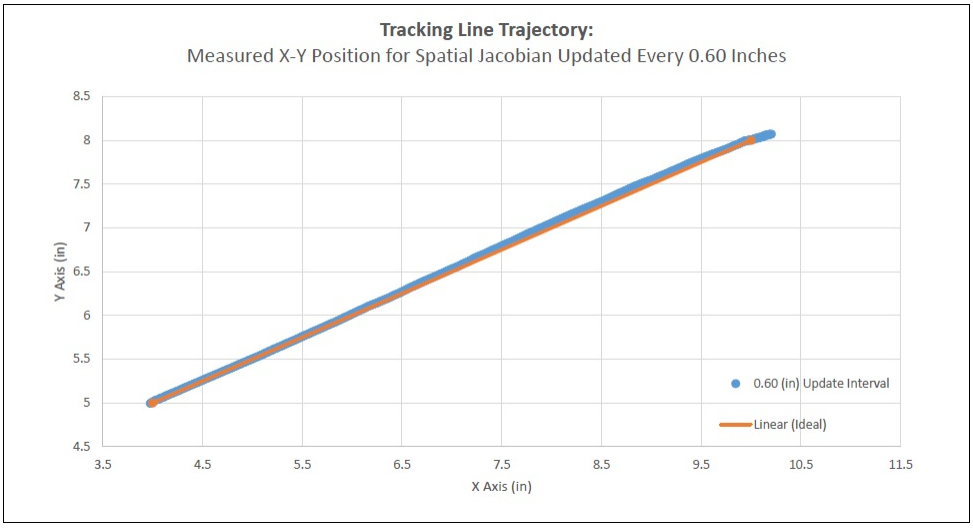

Figure B.12: Chart of the spatial update occurring every .60 inches

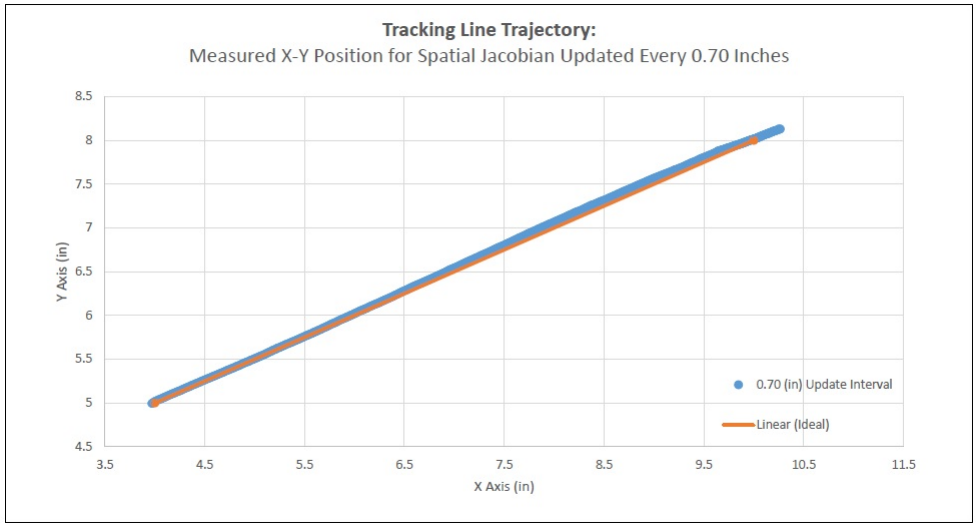

Figure B.13: Chart of the spatial update occurring every .70 inches

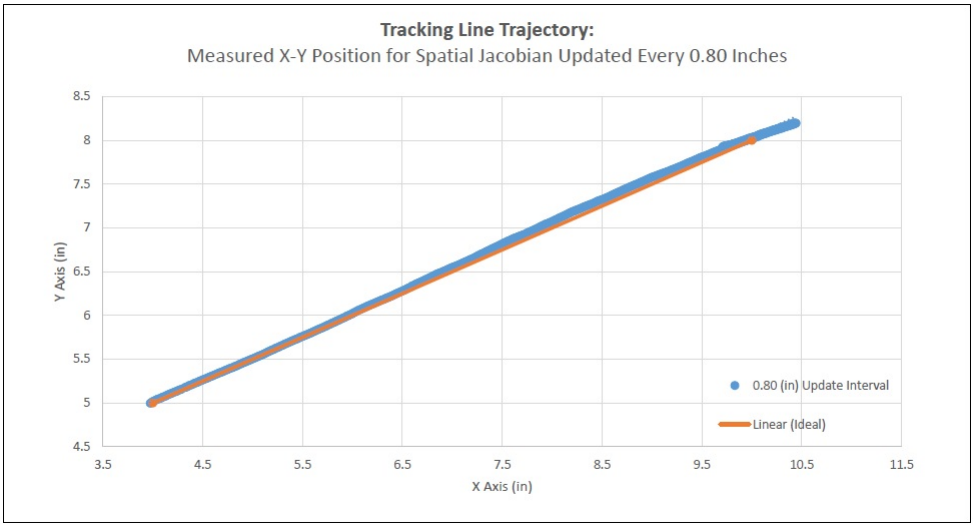

Figure B.14: Chart of the spatial update occurring every .80 inches

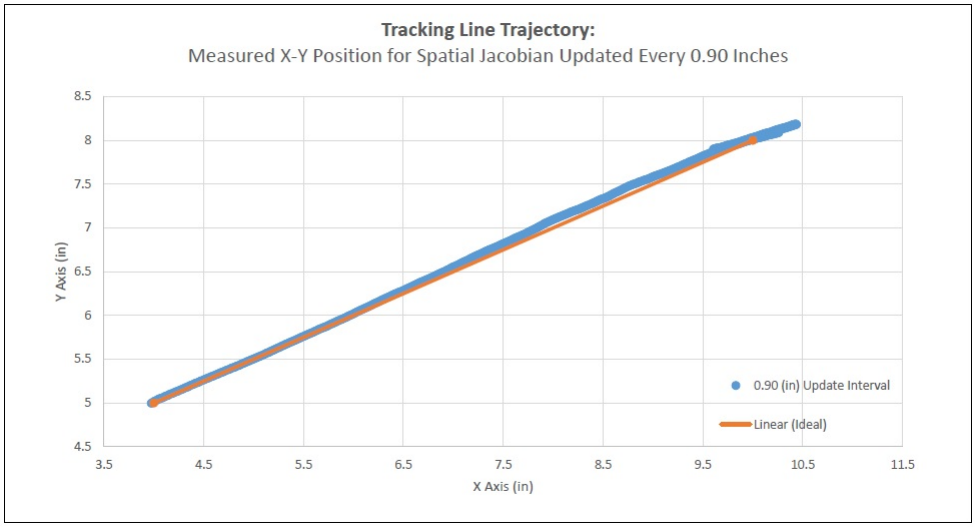

Figure B.15: Chart of the spatial update occurring every .90 inches

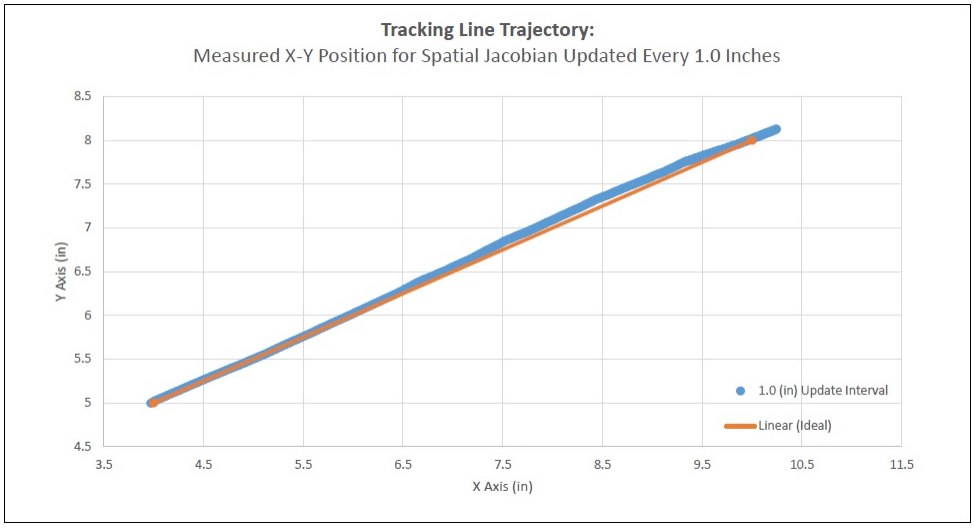

Figure B.16: Chart of the spatial update occurring every 1.0 inches

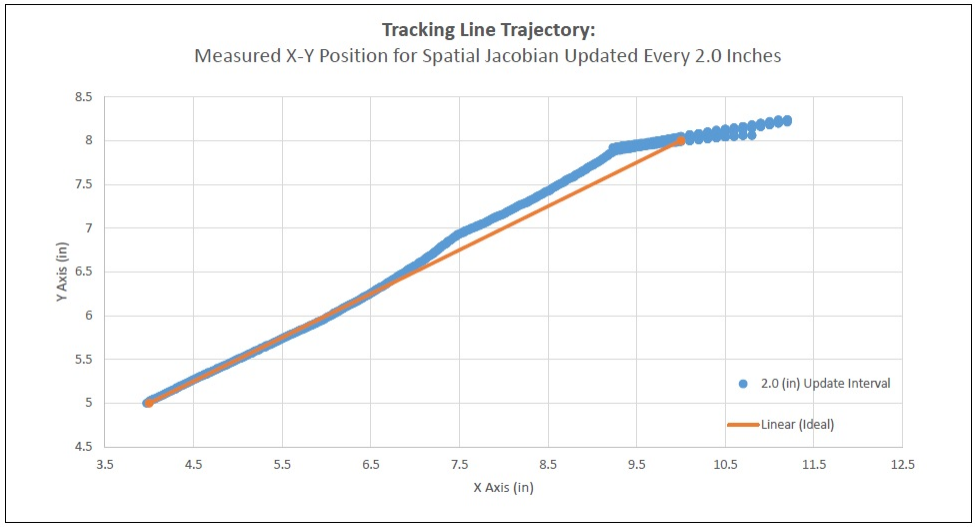

Figure B.17: Chart of the spatial update occurring every 2.0 inches

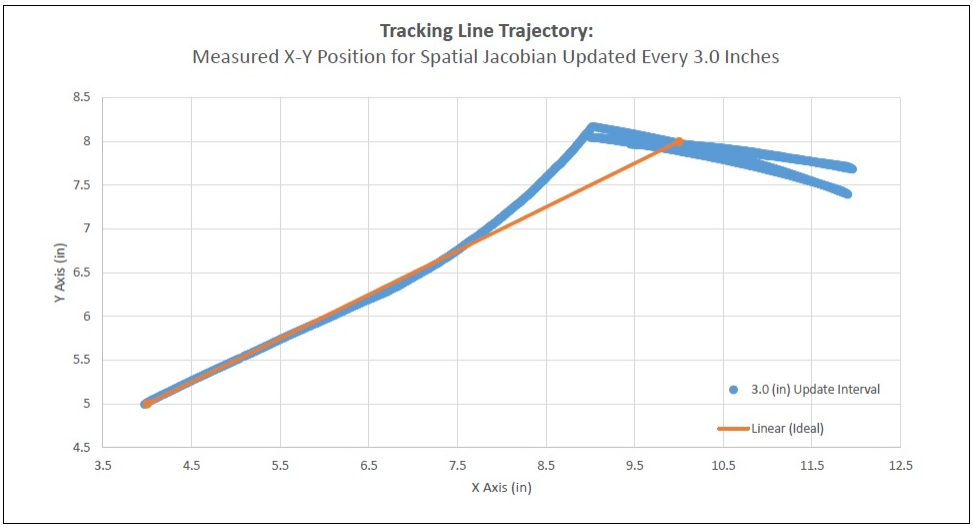

Figure B.18: Chart of the spatial update occurring every 3.0 inches

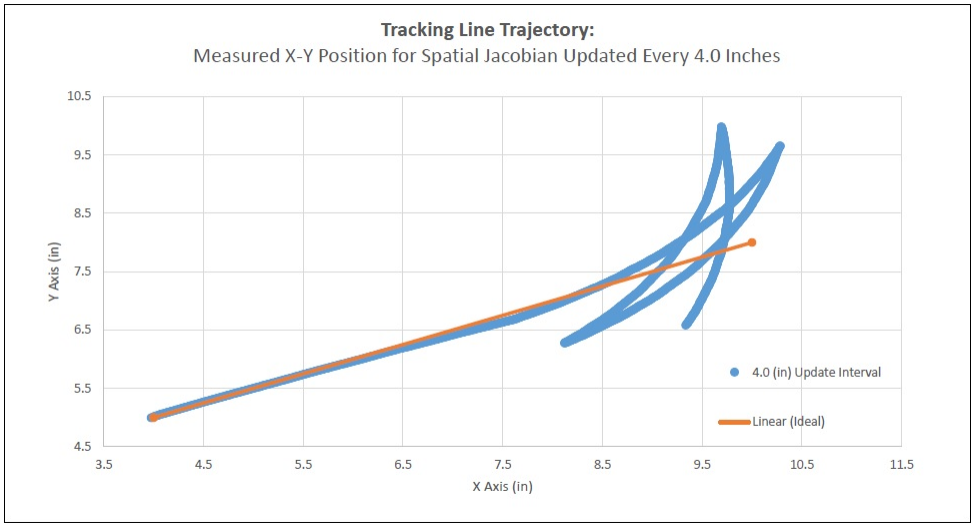

Figure B.19: Chart of the spatial update occurring every 4.0 inches

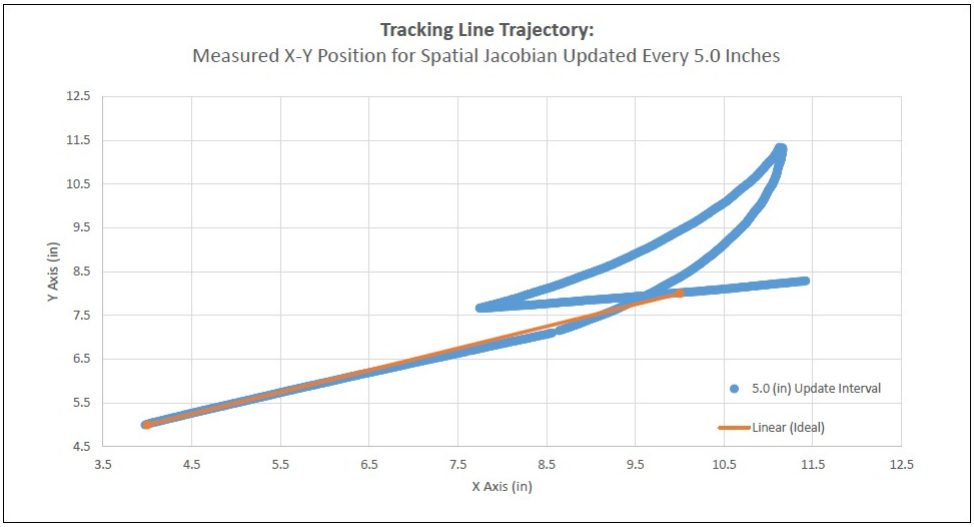

Figure B.20: Chart of the spatial update occurring every 5.0 inches

VITA

Alexis Ehrlich received his Bachelor of Science in Mechanical Engineering in 2012 from the University of Washington. He then briefly lived and worked ChongQing, China before returning to the United States, where he worked at Octopus Robotics. Alexis is currently a research assistant for the Boeing Advanced Research Center at the University of Washington where he developed this thesis. He will begin working as a controls engineer at Electroimpact in the summer of 2016.