ABSTRACT

Title of Document: KINEMATIC DETERMINATION OF AN UNMODELED SERIAL MANIPULATOR BY MEANS OF AN IMU

Constance Ciarleglio
Master of Science, 2013

Directed By: Associate Professor David L. Akin
Department of Aerospace Engineering

Kinematic determination for an unmodeled manipulator is usually done through a-priori knowledge of the manipulator physical characteristics or external sensor information. The mathematics of the kinematic estimation, often based on Denavit-Hartenberg convention, are complex and have high computation requirements, in addition to being unique to the manipulator for which the method is developed. Analytical methods that can compute kinematics on-the fly have the potential to be highly beneficial in dynamic environments where different configurations and variable manipulator types are often required. This thesis derives a new screw theory based method of kinematic determination, using a single inertial measurement unit (IMU), for use with any serial, revolute manipulator. The method allows the expansion of reconfigurable manipulator design and simplifies the kinematic process for existing manipulators. A simulation is presented where the theory of the method is verified and characterized with error. The method is then implemented on an existing manipulator as a verification of functionality.

Kinematic Determination of an Unmodeled Serial Manipulator by Means of an IMU

by

Constance A. Ciarleglio

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2013

Advisory Committee:
Dr. David L. Akin, Chair
Dr. Norman Wereley
Dr. Craig Carignan

Acknowledgments

I would first like to thank Dr. Akin for his support throughout the years and for the incredible projects I've had a chance to work on. His help and guidance have been invaluable through my time at the SSL and with this thesis endeavor. I would also like to thank Dr. Bowden for her support over the course of my experience and for giving me all the opportunities to excel that I could want. I owe much of my success to both these professors.

I would also like to thank my labmates for reading numerous drafts of this thesis and sitting through the presentation. Your input was extremely beneficial in getting both the paper and presentation up to snuff. I would especially like to thank Nick DAmore for his help in the past months. For getting me unstuck when the math seemed insurmountable in times of crisis; for patiently listening to me go on and on about software problems; and for providing a logical sounding board to my somewhat chaotic thought process. Your patience and time are very much appreciated.

I also want to thank my best friend and mentor. I couldn't have done it without you Dru. Last but certainly not least, I would like to thank my mom and dad for their support over the years and for being understanding through the last months of craziness-and the original six years of it as well.

Contents

List of Figures **v**

List of Tables **vii**

List of Abbreviations **viii**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Background . . . 3

2 Initial Research **5**
2.1 DH Parameter Kinematics . . . 5
2.2 Research . . . 8
2.2.1 Pre-Determined Information Based Systems . . . 8
2.2.2 External Sensor Based Methods . . . 12
2.2.3 DH Method Testing . . . 20
2.2.4 Screw Theory in Kinematic Determination . . . 23

3 Screw Theory Kinematics **26**

4 Method Development **34**
4.1 Importance of Method Development . . . 34
4.2 Kinematic Determination Method . . . 35
4.2.1 Error Mitigation . . . 40

5 Kinematic Determination Method Simulation **43**
5.1 Mathematical Simulation . . . 43

6 Hardware Experiment **55**
6.1 Experiment Description . . . 55
6.1.1 Error Analysis . . . 58
6.2 Calculation of "True" Position . . . 61
6.3 Position Error Analysis . . . 63
6.3.1 Error Correction . . . 68

7 Summary and Conclusions **70**
7.1 Summary and Conclusions . . . 70
7.2 Future Work . . . 71
7.2.1 Future Applications . . . 73

A **74**

B **80**

C **83**

D **84**

E **86**

F **116**

List of Figures

2.1 Example Joint as Defined by RMMS 9

3.1 Spherical Displacement of A Rigid Body 26
3.2 Rotation and Translation of a Rigid Body 28
3.3 Stanford Manipulator with Screw Axes 31

5.1 Simulated Manipulator 44
5.2 Comparison of IMU error with Screw Vector Error 46
5.3 Influence of measurement error on tip position 47
5.4 Influence of measurement error on tip position for N DOF 47
5.5 Comparison of Screw Vector Error with Tip Position Error 48
5.6 Effect of Actuated Angle Size of Tip Position Error 49
5.7 Effect of Actuated Angle Size on Tip Position Error NDOF 50
5.8 Effect of Arc Size on Screw Vector Error 51
5.9 Effect of Arc Size on Position Error 52
5.10 Effect of Arc Period on Screw Vector Error 52
5.11 Effect of Arc Period on Position Error 53
5.12 Effect of Arc Repeats on Screw Vector Error 54

6.1 IMU as connected to Manipulator 56
6.2 IMU as attached to Joint 6 Plate 56
6.3 IMU Error with running motors 59
6.4 Error in Joint 4 measurement 60
6.5 Manipulator Initial Pose with and without vectors 63
6.6 Joint Example Accelerations 64
6.7 Joint Example Angular Rotation 64
6.8 Joint Example Angular Rotation 65
6.9 X and Y Position Plotted for 21 Poses 66
6.10 X Position and Position Root Mean Error for 21 Poses 66
6.11 Position Error Contributed by each joint 67
6.12 X and Y Exact and Estimated Position Over Joint 2 Angles 68

A.1 Joint 5 XY Accelerations 74
A.2 Z Acceleration J5 74
A.3 Joint 5 XY Angular Rates 75
A.4 Angular Rate About Z J5 75
A.5 Joint 4 XY Accelerations 75
A.6 Joint 4 Z Accel and X Rotation 76
A.7 Joint 4 YZ Angular Rates 76
A.8 Joint 3 XY Accelerations 77
A.9 Joint 3 Z Accel and X Rotation 77

A.10 Joint 4 YZ Angular Rates 78
A.11 Joint 2 XY Accelerations 78
A.12 Joint 2 Z Accel and X Rotation 78
A.13 Joint 2 YZ Angular Rates 79

B.1 Calculation of Pose via Point to Line 80

C.1 Influence of measurement error on tip position for 2 DOF 83
C.2 Influence of measurement error on tip position for 5 DOF 83

D.1 Effect of Actuated Angle Size on Tip Position Error 2DOF 84
D.2 Effect of Actuated Angle Size on Tip Position Error 3DOF 84
D.3 Effect of Actuated Angle Size on Tip Position Error 4DOF 85
D.4 Effect of Actuated Angle Size on Tip Position Error 5DOF 85

List of Tables

2.1 Initial Values for 2 Link Manipulator 20
2.2 Estimated Values for 2nd Joint of a 2 Link Manipulator 21
2.3 Estimated Values for a 5 Link Manipulator Given 3 of 6 parameters . 22
2.4 DH Parameters for a 5 link Manipulator given 4 of 6 Parameters and Error 22

5.1 Simulated Arm Characteristics 45
5.2 Calculated Manipulator Parameters 45

6.1 IMU Manufacturer Error Tolerances 58
6.2 Joint Trajectory Period and Repeat Count 61
6.3 Denavit-Hartenberg parameters for Ranger Mark I. From [17]. 62
6.4 Initial Arm Pose Joint Angles 63
6.5 Calculated Screw Vectors for Ranger Mark I 65

List of Abbreviations

AIM	Assembly Incidence Matrix
DH Parameters	Denevit-Hartenberg Parameters
DOF	Degree of Freedom
IMU	Inertial Measurement Unit
NBV	Neutral Buoyancy Facility
PSD	Power Spectral Density
RMMS	Reconfigurable Modular Manipulator System
SSL	Space Systems Lab
SVD	Singular Value Decomposition

Chapter 1: Introduction

1.1 Motivation

Most robotic manipulator systems are limited in the scope of their possible applications. Many can perform just a select few tasks optimally. As such, manipulators are frequently designed for as broad a task base as possible, given mechanical design limitations for each desired application. [19] In exchange for increased task capability, performance optimization for each individual task is decreased. In some cases, different tools and advanced wrist structures [20] are used to expand the capability of the manipulator. One other solution to the tradeoff of performance and task range is the reconfigurable manipulator. A reconfigurable manipulator is hardware and software configurable to be optimal for a specific task,[21] while retaining task flexibity by the nature of its reconfigurable construction.

Reconfigurable manipulators are often preferred for highly dynamic environments, such as outer space and nuclear facilities. Underwater manipulators also operate in a variable environment where a reconfigurable system is potentially useful. [22] An additional requirement of the underwater manipulator is covering or water-proofing of electrical connections. Micro-gravity based manipulators are of particular interest to this thesis, due to the high range of tasks performed and flexibility required. Some micro-gravity manipulators are expecially unique, such as the

morphing or flexible manipulator. The more widely used serial revolute reconfigurable manipulator will be addressed under the scope of this thesis.

For each task, the kinematics of the reconfigurable manipulator are different. As such, kinematic determination for the manipulator can be difficult. Joint order, type, and link length change on a task-by-task basis and this variability adds complexity in determining the current kinematics. There are several current methods of reconfigurable manipulator kinematic determination. Most reconfigurable manipulators are based around a library of known components. Optimal configuration for a task is pre-calculated and the manipulator is assembled accordingly. A similar version of this method is done with an Assembly Incidence Matrix (AIM), or a matrix of 1s and 0s which relate joints and links. Both of these methods rely on prior knowledge of the manipulator and its configuration and are software and optimization intensive.

Other kinematic determination methods are based on sensor input. The most common types are visual and accelerometer data. Machine vision data gives a very accurate description of manipulator configuration, but requires high volume data and an extensive calibration setup. Use of accelerometer data is the most interesting method for the purposes of this thesis. It is a minimally sized sensor that can be incorporated into manipulator design or placed on the outside of a system. All current uses of the accelerometer in kinematic determination are based around measurement of the gravity vector. Since a mirco-gravity based manipulator is desired, these methods are unsuited for use in this thesis in their current form.

All current methods of reconfigurable manipulator kinematic determination

are based on previous knowledge of arm configuration, high complexity calibration or optimization, or use of the gravity vector. The purpose of this thesis is to develop a new method of kinematic determination that is independant of previous configuration knowledge, requires no optimization, calibration, or external setup, and is not dependant on knowledge of the gravity vector such that it can be used in a microgravity environment.

1.2 Background

Some understanding of current forward and inverse kinematic techniques is required following along with the robotics intensive aspects of this thesis. Two calculations of manipulator kinematics are generally accepted as commonplace in the robotics community. The Denavit-Hartenberg (DH) convention attaches a cartesian frame to each link of the manipulator. Following specific rules, and, using the inherent physical properties of the manipulator, a homogeneous transformation is derived between the frame of the one link and the next. With these transformations, the forward kinematics of the manipulator can be calculated. Four parameters are used to describe each joint and joint relationships. The four DH parameters of each joint are practical to use due to their association with link lengths and joint frame orientation.

Screw theory kinematics presents a different approach. Screw theory relates the position of the manipulator tip with a rotation and translation about some axis. By using a series of rotations and translations associated with each joint, the

position and orientation of the manipulator tip can be calculated with respect to a reference frame. While this kinematic method is utilized less in the manufacturing industry, it is the most geometrically based method of kinematic calculation, This geometric basis can simplify the mathematics of the system and add flexibility for use in a variety of applications. Since both DH and screw theory kinematics are necesary knowledge in the goal of this thesis and associated background research, a brief description of both is included for the reader.

Chapter 2: Initial Research

2.1 DH Parameter Kinematics

In DH convention, the defined relationship between two joints allows calculations of forward kinematics. The relationship between two neighboring joint axis is defined specifically, incorporating the rigid link in between which constrains motion of one joint relative to the other. For any two joint axes, the distance between them is well defined and measured along a line perpendicular to both axes. This quantity is called link length a_{i-1}. In order to quantify rotation of one link axis relative to the other, link twist is also defined. Link twist is the measure of the angle between the link axes as projected onto a plane coincident with the perpendicular line between the axes. This angle α_{i-1} is measured from axis i-1 to i. These two quantities, length and twist, define the relationship between any two lines, in this case two joint axes. Link offset d_i is additionally defined specifically for joints as the distance from one link to the next along a common axis. Rotation about this common axis from one link to the next is defined as joint offset θ_i. With these four parameters, $a_{i-1}, \alpha_{i-1}, d_i$ and θ_i, defined between joints, kinematics for any robot can be described.[10] Definition of kinematics using these parameters is called DH convention.

Joint frames are attached following a convention that holds to the link rela-

tionship parameters described above. In general, frames are affixed such that [10]:

- $\hat{a}_i$ = the distance from $\hat{Z}_i$ to $\hat{Z}_{i+1}$ along $\hat{X}_i$
- α_i = the angle from $\hat{Z}_i$ to $\hat{Z}_{i+1}$ measured about $\hat{X}_i$
- d_i = the distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along $\hat{Z}_i$
- θ_i = the angle from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured about $\hat{Z}_i$

With the DH parameters assigned to each joint which has a frame affixed in the above convention, a transformation from frame i to frame i-1 can be calculated. For the interested reader, the derivation of the exact transform may be found in [10]. The transformation matrix from frame i to frame i-1 can be written as

$$ {}^{i-1}_{i}T = \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & a_{i-1} \\ sin\theta_i cos\alpha_{i-1} & cos\theta_i cos\alpha_{i-1} & -sin\alpha_{i-1} & -sin\alpha_{i-1} d_i \\ sin\theta_i sin\alpha_{i-1} & cos\theta_i sin\alpha_{i-1} & cos\alpha_{i-1} & cos\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1} $$

Once the transformation has been established for each link, forward kinematics are straight forward. The link transformation matrices can be multipled linearly to find a single transformation matrix from frame N to frame 0

$$ {}^{0}_{N}T = {}^{0}_{1}T {}^{1}_{2}T \cdots {}^{N-1}_{N}T \tag{2.2} $$

Position in one frame can thus be calculated in any other frame of the manipulator.

The Jacobian matrix is a set of time-varying linear transformations. It is composed of the partial derivates of a system of functions. In kinematics the Jacobian

plays a wide variety of roles, including mapping joint angle velocites to the velocity of the end effector and as an idicator of singularities. The Jacobian can either be found by looking at the z components multipled by velocity in the equation that defines angular velocity

$$^{i+1}\omega_{i+1} = {}^{i+1}_{i}R^{i}\omega_i + \dot{\theta}_{i+1}{}^{i+1}\hat{Z}_{i+1} \tag{2.3}$$

or by direct differentiation of the kinematics equations. [10] One important application of the Jacobian is its use in inverse kinematics as presented in [18]. In this method it can be written that

$$\dot{\vec{s}} = J(\theta)\dot{\theta} \tag{2.4}$$

where $\dot{\vec{s}}$ represents the position of the end effector tip and $\dot{\theta}$ the array of joint angles. The change in end effector position as caused by change in joint angles is written as

$$\delta\vec{s} = J\delta\theta \tag{2.5}$$

where the change in end effector position should be equal to that of the difference in target and current end effector position $\vec{e}$. The values of the joint angles are updated iteratively until the value obtained is close to a solution. One approach to the angle update is to say that

$$\vec{e} = J\delta\theta \tag{2.6}$$

and then solve

$$\delta\theta = J^{-1}\vec{e} \tag{2.7}$$

In this way, the inverse kinematics can be solved for the manipulator.

2.2 Research

There are, generally, two classes of kinematic determination methods for reconfigurable manipulators. Kinematics can be determined by some type of previously known information about the manipulator. Pre-determined knowledge can include component, joint and link, libraries or a different method of representing the relationship and types of joints and links. Manipulator kinematics can also be found using external sensor information. The most common forms of external sensing associated with kinematic determination are machine vision and accelerometers. Sometimes external sensing techniques are combined with component libraries to create a more comprehensive kinematic estimation. This section details applicable previous works in the areas of kinematic determination using pre-knowledge and external sensing techniques.

2.2.1 Pre-Determined Information Based Systems

Micro-gravity based reconfigurable manipulators tend to use a library of components for kinematic estimation. [6] and [9] introduce two similar library techniques. Only the Reconfigurable Modular Manipulator System (RMMS) kinematic deter-

mination method will be discussed while the interested reader is referred to [9] for a detailed description of a similar method. Starting with a geometric description of the joints and links of the manipulator, as defined and created in [19], and their sequence, DH parameters can be generated which describe the forward kinematics of the manipulator.

Geometric desciptions for each component include a homogeneous transformation matrix for the link or joint module. In the case of a link, a transformation from the connector at one end of the link to the connector at the other completely specifies the geometry of the link. Because a joint has relative motion, two homogeneous transformation matrices are needed to incorporate both geometry and degree of freedom (DOF). A sample joint is shown in 2.1.

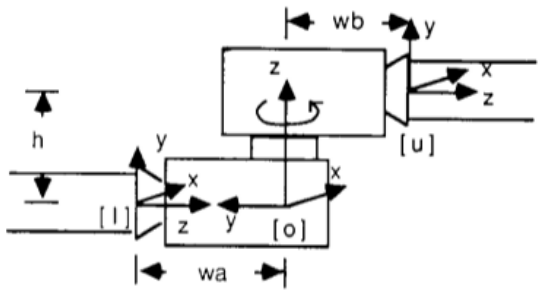


Figure 2.1: Example Joint as Defined by RMMS

Two transformation matrices represent a translation and rotation from the center of the lower right connector to the upper left. In order to facilite assembly in an undetermined sequence, frames are attached to the connector points in identical orientation so that the transformation between modules is the identity matrix. This is the case except for an offset angle, twist between modules, which is incorporated with an angle about the z axis in the connector. The transformations between links

and joints are now used to calculate DH parameters for the robot. However, in some cases, the calculated module transformation may not be possible in DH, such as a translation along a Y-axis. Therefore, the DH frames must necessarily be different from the module frame described above. Another transform is therefore needed from the module to DH frame. At this point, the forward kinematics for the manipulator can be calculated.

The difficulty in the library method is the optimization of the manipulator configuration. With the configuration decision, the kinematics are known. DH parameter generation requires relative position and orientation of successive DH frames, and is based on a line representing the joint axis and the direction cosines of that line. DH parameters are then determined based on succesive direction cosines. Inverse kinematics are as difficult in a reconfigurable manipulator as they are in a static configuration. Iterative methods are used to solve to inverse kinematic problem, specifically using the inverse Jacobian.

While a library based kinematic determination system works for most manipulators, it has less than ideal features as it is not applicable to a truly unknown manipulator. The use of DH parameters in this system also complicates the mathematics of the kinematic calculation. Mathmatical intensity is added as manipulator configuration is determined by an optimization of library components for a specific task, in addition to the extra transformations required to rotate the joint module frames to match DH convention. In this case, DH convention adds some complexity to the system through extra rotation and in the configuration selection. Specifics of the library method described above can be found in [6].

Another method of reconfigurable kinematic determination is based around the Assembly Incidence Matrix (AIM) as presented in [5]. This is a graph based representation used to indicate changing configurations and, like the library method, is based around a set of known joint and link modules. Mechanically, the manipulator modules are based around link modules with a multiport connector, such that the joint module can be attached in many orientations, producing different arm configurations. Like the library method, the AIM system cannot be used on an undetermined manipulator.

The AIM based kinematic model does not use typical DH convention, instead using a method based on the kinematic graphs of mechanisms.[24] In a modular robot, vertices represent link modules and edges represent joint modules in the kinematic graph. This graph is then expressed as a matrix of 1s and 0s, or a vertex-edge matrix,[25] in which a value is 1 if an edge is incident on a vertex and zero otherwise, and with an extra column for type of link and extra row for type of joint. The AIM is therefore an expression of joint and link types, order of assembly, and connection information. This is a complete description of the robot configuration.

In forward kinematics with the AIM method, two adjacent links υ_i and υ_j connected by joint e_k and with a joint displacement θ_k relative to the i frame, can be related by

$$T_{ij}(\theta_k) = e^{\xi_k \theta_k} T_{ij}(0) \tag{2.8}$$

where T_{ij} is a 4X4 transformation matrix and ξ_k describes the twist of the joint. Based on 2.8, an algorithm is generated for a general tree-structured robot which

can automatically derive forward kinematics given an AIM. A graph-traversing algorithm, or a searching algorithm, is used to find link order. In this method, a close-form inverse kinematic solution is generally not possible. Again, a numerical iterative method is used to solve for inverse kinematics. The interested reader may refer to [23] for a detailed calculation of forward and inverse kinematics for this method.

Like the library based method of kinematic determination, the AIM method is dependant on knowledge of manipulator configuration prior to forward kinematic calculation, but in the form of an modified vertex-edge matrix. Additionally, the AIM depends on highly complex mathemetics, even more so than a library based calculation. Calculations also required include optimization and searching algorithms for forward kinematics. Inverse kinematics require the usual numerical iteration method. High complexity, in addition to required knowledge of the manipulator, makes this method non-ideal for use in a system with only encoder counts per joint revolution known. A more simplistic, analytical method is desired for kinematic calculation of an arm that is rapidly changing to suit a task.

2.2.2 External Sensor Based Methods

Another strategy for kinematic estimation of an unmodeled manipulator is to use data derived from sources external to the system. Visual data from an external camera and accelerometer data are the most common types. Visual data can also be used in combination with a library based kinematic estimation system for a more

calibrated forward kinematics calculation. This section details the use of machine vision, and machine vision with component libraries, and accelerometer data in kinematic determination. A final determination is made, based on these sensors, on the best method of simple kinematic estimation for a micro-gravity manipulator.

A vision based kinematic parameter identification strategy was developed by [8] for the purpose of calculating industrial robot kinematics with higher accuracy than the parameters specified by the manufacturer, in order to account for differences in individual robots. The system begins with limited knowledge of the manipulator, including the relative position and orientation of the base and camera attached to the manipulator wrist. Also included is an extra transformation representing the orientation error between two successive parallel joints. This joint orientation error is highly dependant on errors in camera orientation.

The identification method for DH parameters needed in forward kinematics is based on the measurement of a target position and orientation for several configurations of the manipulator. The transformation matrix representing the position and orientation of the target with respect to the camera is calculated by analyzing the image of the target. Each manipulator configuration generates six independant equations, with more manipulator DOF increasing the number of equations required. The use of additional equations will increase the accuracy of the guessed parameters. The unknown parameters of the manipulator and the camera relative to the target are resolved with an iterative method to solve an overdetermined set of equations by means of least-squares. Convergence can be achieved by using the manufacturer specified kinematics as a starting point.

A complex vision system calibration and image processing scheme are required for successfull kinematic calibration. The target used is composed of an array of spheres, which appear as circles when projected onto a 2D image sensor. Using image processing techniques, pixel interpolation of the image, and a least-squares technique, the approximate centers of the circles in the image can be found, as well as the radius of each sphere. With the centers of the circles, the position and orientation of the target with respect to the camera frame can be determined. The coordinates of the circle centers are mapped into the physical coordinates of the camera first. A general change in rotation is written using the quaternion representation of the finite rotation formula. This yields an overdetermined set of equations which can be solved with a least-squares method. Each point on the target is linked to the focal center of the lens and the quarternion representation is expressed using the direction cosines of the target points projected onto the 2D camera plane. Kinematic parameters are calculated from the solutions of the set of equations.

There are many limitations to the method of kinematic determination with machine vision as it relates to the goals of this thesis. While accurate to less than a tenth of a millimeter, this system is mathematically complex and intensive. The majority of the method relies on vision based optimization, iterative techniques, and least-squares techniques. In addition to having the highest complexity of any method presented thus far, a vision system is dependant on external target parameters and calibration of the camera relative to these targets. If a component library is introduced as well as vision calculation as in [11], the system becomes even more complex. Because of these factors, a vision based kinematic calculation is non-ideal

for the goals of the kinematic determination method in this thesis.

Kinematic determination through use of an accelerometer is an interesting proposal due to the low size and complexity of the sensor. There are two methods which directly address the use of an accelerometer to find the kinematics of a system in an earth gravity environment. One method supports the use of DH parameters in forward kinematic calculation[1] while the other bases the estimation on screw theory convention.

The first step to analysis of a revolute manipulator using acceleration, as will be used in the subsequent previous works and research in this thesis, is the recognition of circular motion during single joint actuation. In circular motion, the linear acceleration of a point around the edge of the circle can be characterized by the angular rotation rate, angular acceleration, and radius of the circle around which the motion occurs. The derivation of the linear acceleration equation begins with the definition of angular velocity.

$$\omega = \frac{2\pi}{T} \tag{2.9}$$

Linear velocity can be defined in terms of the period of the motion and the distance from the center of motion.

$$v = \frac{2\pi r}{T} \tag{2.10}$$

substituting 2.9 into 2.10 yields

$$v = \omega r \tag{2.11}$$

which holds generally for non-periodic motion. The linear velocity vector can be

thought of as the right hand side of 2.11 multiplied with some unit vector which defines the direction of the product.

$$\vec{v} = \omega r \hat{u} \tag{2.12}$$

Since the angular rate and radius of the circle will be perpendicular in vector form, 2.12 can be written as

$$\vec{v} = \vec{\omega} \times \vec{r} \tag{2.13}$$

Differentiating 2.13 with respect to time and using the chain rule with the cross product yields

$$\frac{d\vec{v}}{dt} = \frac{d\vec{w}}{dt} \times \vec{r} + \vec{\omega} \times \frac{d\vec{r}}{dt} \tag{2.14}$$

where it can be seen that

$$\frac{d\vec{w}}{dt} = \vec{\alpha} \tag{2.15}$$

$$\frac{d\vec{v}}{dt} = \vec{a} \tag{2.16}$$

and

$$\frac{d\vec{r}}{dt} = \vec{v} \tag{2.17}$$

Substituing 2.15, 2.16, and 2.17 into 2.14 gives

$$\vec{a} = \vec{\alpha} \times \vec{r} + \vec{\omega} \times \vec{v} \tag{2.18}$$

Using 2.13 in equation 2.18 yields the final linear acceleration equation around a circle

$$\vec{a} = \vec{\alpha} \times \vec{r} + \vec{\omega} \times (\vec{\omega} \times \vec{r}) \tag{2.19}$$

which is based solely on the radius of the circle, angular rate, and angular acceleration. 2.19 is an important equation upon which the following methods of kinematic determination will be based.

A manipulator with an artificial sensing skin has access to a more complete data set than a standard manipulator. This allows the manipulator to localize and react to events, such as touch. [1] implements an artificial skin manipulator with which accelerometers are used to actively determine the relative location of every sensor type and actuator. The robot uses its own sensors, located on each joint, to model and calibrate itself, without any a-priori knowledge. This method is also contact free and open-loop, whereby the robot can quickly and unambiguously determine a kinematic model.

DH parameters for the method used by [1] are determined by actuation of one joint at a time. There are N accelerometers for an N DOF manipulator. A serial chain model is developed with DH parameters as input and acceleration and rotation matrices as output. This model is capable of taking in the DH parameters, outputting an acceleration, and optimizing the DH parameters such that the output accelerations are as close as possible to the measured values. The first step in the calculation is to place a measurement unit, i.e. accelerometer, on the z axis of each joint. The measurement unit frame will be called the SU frame. The transformation

between the joint and the SU frame is simply a rotation about and translation along the z axis of the joint. The transformations that must be calculated are for each joint frame, which are determined by the unknown kinematics of the manipulator. The transformation from a measurement unit frame to some inertial frame is given as

$$^{SU_N}T = {}^{SU_N}T_{DOF_{N+1}} \cdot (\prod_{d=1}^{N} {}^{DOF_d+1}T_{DOF_d})^{DOF_1}T_{RS} \tag{2.20}$$

Each transformation matrix is calculated as per the standard DH convention given in equation 2.1, with the addition of an initial pose angle and the angle as the joint as actuated added to the calculated θ parameter. The estimated kinematic model is now used to calculate position of the end of the manipulator using the joint transformation matrices. Position is differenced twice to calculate the estimated acceleration of the measurement unit in the reference frame.

$$^{RS_N}\vec{a}_N = \frac{(^{RS}\vec{p}_N(h) + {}^{RS}\vec{p}_N(-h) - 2 \cdot {}^{RS}\vec{p}_N(0))}{h^2} \tag{2.21}$$

However, the acceleration must be calculated in the SU frame to be accurate. Another rotation from the reference frame to the SU frame is needed as given by

$$^{SU_N}\vec{a}_N = {}^{SU_N}_{RS}R \cdot {}^{RS_N}\vec{a}_N \tag{2.22}$$

The acceleration calculated in equation 2.22 will be the same as the acceleration calculated in equation 2.19.

With the set of generated accelerations, the measured accelerations can be

used to calculate error in the system. One part of the error calculation is based around the model estimated acceleration values and can be found by

$$e_2 = \sum_{p=1}^{P} \sum_{d=N-3}^{N} |{}^{SU_N}\vec{a}^{model} - {}^{SU_N}\vec{a}^{actual}|^2 \tag{2.23}$$

The other portion of the error must be calculated using the measured and estimated gravity vector. The gravity vector is sampled when the manipulator is in a static position for every pose used in the kinematic calculation. The calculated rotation matrices for the manipulator are used to rotate the gravity in an inertial frame to the gravity seen by each measurement unit. The difference in these two values is calculated as

$$e_1 = \sum_{p=1}^{P} |{}^{RS}\vec{g}_N - \frac{1}{P}\sum_{p=1}^{P} {}^{RS}\vec{g}_N|^2 \tag{2.24}$$

The sum of the acceleration and gravity error

$$e_T = e_1 + e_2 \tag{2.25}$$

is the output of a cost function which minimizes the error of the model output and measured values. Since the optimizer can only succesfully handle so many DH parameters to be estimated at a time, an optimization is performed for 2 joints per optimization sequence, with one joint overlapping for each calculation. This means only 10 variables are optimized at a time.

2.2.3 DH Method Testing

The DH parameter estimation based on an accelerometer for each joint of the robot presented in [1] has potential for use in a space environment, provided the error offset added by gravity in the cost function of the model optimization is not significant. An implementation of this method was tested with and without gravity in order to assess the feasibility of using the method in a micro-gravity environment.

A Matlab program was created to recreate and verify the method described in [1]. The program was tested with a 2DOF and 5DOF manipulator and functionality of the method verified to within the tolerances specified by [1]. Verification was achieved by looking at the DH parameters output of the optimization function without any error added to the system. Now removing the gravity error from optimization function, for a 2DOF manipulator, the DH parameters as estimated by the optimizer are compared to the exact values, for the first joint, and given in table 2.1

Type	α	a	θ	d	θ_{acc}	d_{acc}
Estimated First Joint	1.5710	0.3996	0.0446	1.0991	0.2007	0.0552
Exact First Joint	$\pi/2$	0.4	0	1.1	0.2	0.1

Table 2.1: Initial Values for 2 Link Manipulator

This is an estimation, without the use of gravity error, for the first joint DH parameters when the second joint parameters are given exactly to the optimizer. θ_{acc} and d_{acc} refer to the offset of the SU relative to the joint and, since joint 1

cannot be measured, are not capable of being accurately estimated for the first joint of any manipulator.

When the joint two variables are estimated instead of given, the optimizer returns the values in table 2.2 for the second joint of the 2DOF manipulator.

Type	α	a	θ	d	θ_{acc}	d_{acc}
Estimated Joint 2	-1.5713	0.1997	0.6906	0.5016	0.098	-0.4908
Exact Joint 2	$\frac{-\pi}{2}$	0.2	0	0.5	0.1	0.2

Table 2.2: Estimated Values for 2nd Joint of a 2 Link Manipulator

Without the use of gravity, the optimizer cannot estimate the DH parameters of joint 2 accurately. In the serial chain model, an innaccurate estimate of the second joint will yield an even more inaccurate guess of the parameters for the first joint. Based on this and other similar tests for different 2 link configurations, the use of this method for DH parameter determination without gravity is not possible.

Without gravity error, the only possible accurate use of this method is for pose determination of a manipulator with other DH parameters known. This was tested with a 5 DOF manipulator with no input error and gave the results in table 2.3.

i	a_{exact}	$a_{estimate}$	θ_{exact}	$\theta_{estimate}$	$d_{accexact}$	$d_{accestimate}$	Num of Iterations
1	0	0	0.3	1.0407	0.1	1.00	105
2	0	0.0002	0.9	0.9002	0.2	0.1995	104
3	0	0	0.2	0.1995	0.3	0.3003	268
4	0.4	0.4002	1.1	1.0998	0.1	0.1004	153
5	0.2	0.2002	0.5	0.5011	0.2	0.1996	13

Table 2.3: Estimated Values for a 5 Link Manipulator Given 3 of 6 parameters

This is a relatively accurate result compared to the error tolerance in [1]. However, when error is added to the system, DH parameter accuracy degrades to a point where additional parameters must be known by the system. Even when 4 of 6 DH parameters are known per joint, the optimizer produces a result outside the desired error tolerance, as shown in table 2.4.

i	a_{exact}	$a_{estimate}$	θ_{exact}	$\theta_{estimate}$	$d_{accexact}$	$d_{accestimate}$	Num of Iterations
1	0	0	0.3	3.1416	0.1	0.6281	74
2	0	0.0014	0.9	0.9000	0.2	0.1969	66
3	0	0.0023	0.2	0.1951	0.3	0.2969	86
4	0.4	0.3998	1.1	1.1055	0.1	0.1030	126
5	0.2	0.1977	0.5	0.4856	0.2	0.2006	13

Table 2.4: DH Parameters for a 5 link Manipulator given 4 of 6 Parameters and Error

Given the high error and required number of known variables for this system,

it can only be used for pose determination, and not to desirable error tolerances. It provides no useful information for the kinematic determination of an unmodeled micro-gravity manipulator and cannot be used for the purposes of this thesis.

2.2.4 Screw Theory in Kinematic Determination

There is one method which uses a different convention with acceleration in order to determine unmodeled kinematics. [4] presents a method based on the use of a single accelerometer and screw theory conevention in order to solve for the kinematics of an earth-based manipulator. The accelerometer is placed at the tip of the robot. There are only three defined frames for the manipulator; joint frame [w], accelerometer frame at a hard stop [a], and moving accelerometer frame [m]. The transformation from [a] to [m] is a simple rotation about the z axis. The transformation from the joint frame [w] to the accelerometer frame [a] is unknown. Gravity can be used to help define the unknown rotation. Gravity in the moving frame is written as

$$^{m}g = {}^{m}_{a}R^{a}g = {}^{m}_{a}R^{a}_{w}R^{w}g \tag{2.26}$$

If gravity is written as the portion only in the direction of the screw axis, equation 2.26 can be rewritten as

$$^{a}z^{T}{}_{w} \cdot {}^{m}g = ({}^{w}g)_{z} \tag{2.27}$$

It can be stated, from this equation, that the gravity in the screw axis direction is constant in the joint frame [w]. With this value, the orientation of the screw axis

for a joint can be determined and gravity in the moving frame calculated.

Since no acceleration occurs in the screw axis direction, a new linear acceleration equation can be written for the motion of the accelerometer as the joint actuates.

$$\int_0^t {}^w a_m(\tau)\mathrm{d}\tau = {}^w v(t) - {}^w v(0) = rwi_\theta - rw(0)i_{\theta 0} \tag{2.28}$$

where i_θ is in the direction of tangential velocity. Equation 2.28 can be integrated a second time to get a position in the joint frame.

$$\int_0^t \int_0^t {}^w a_m(\tau)\mathrm{d}\tau\mathrm{d}\tau = \int_0^t {}^w v(\tau)\mathrm{d}\tau = {}^w x(\tau) - {}^w x(\tau) \tag{2.29}$$

Using position data and a least-squares fit, the point about which the accelerometer turns can be calculated.

The radius of the arc is calculated based on the angular rate as derived from the joint encoders and the previous integration of acceleration.

$$r = \frac{|\int_0^t {}^w a_m(\tau)\mathrm{d}\tau}{\omega(t)} \tag{2.30}$$

Using geometric relationships, r can be used to find the vector to the screw axis as

$$b = \begin{bmatrix} rcos(90 + \theta - \phi) \\ rsin(90 + \theta - \phi) \end{bmatrix} \tag{2.31}$$

where θ refers to the angle between r and the perpendicular to tangential velocity.

Phi is calculated as

$$\phi = arctan2(v_y, v_z); \tag{2.32}$$

With the screw axis and vector to the screw axis calculated, the forward kinematics are determined. Due to limited technology of the time, other calculations were needed in this method to compensate for anomolies in the hardware which are not relevent with accelerometer technology today.

While the screw theory method in [4] is dependant on gravity and double integration techniques, it is a good example of the advantages of screw theory in kinematic estimation. Because the reference in screw theory can be defined as needed by the user, the mathematics of the system become simpler and geometrically based, rather than referring back to a pre-defined pose or joint to joint relationships. No optimization is needed because the zero pose is defined as is most convenient for the mathematics being calculated. Only an analytical solution is required.

Chapter 3: Screw Theory Kinematics

Screw theory kinematics are based on the screw axis representation, whereby the orientation of a rigid body is reperesentated by a rotation about some screw axis. This way of thinking about orientation leads to the representation of the spherical displacement of a rigid body as shown in figure 3.1.[1]

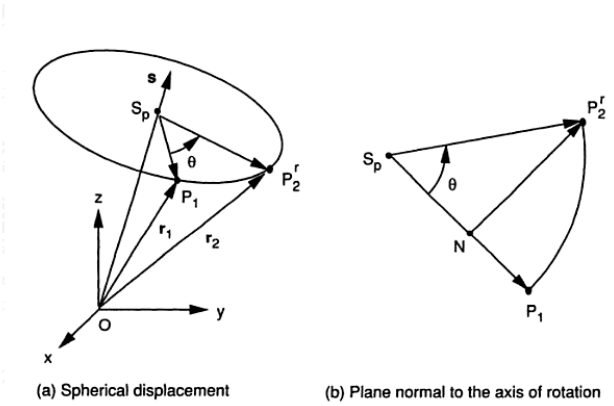


Figure 3.1: Spherical Displacement of A Rigid Body

The vector $\vec{r}_2$ represents the new position of the rigid body after rotation about the vector $\vec{s}$ and can be written as

$$\vec{r}_2 = \vec{r}_1 cos\theta + \vec{s} \times \vec{r}_1 sin\theta + \vec{s}(\vec{r}_1^T \vec{s})(1 - cos\theta) \tag{3.1}$$

where $\vec{r}_1$ is the initial position of the rigid body before rotation and θ is the angle rotated about $\vec{s}$. Equation 3.1 can be written simply as a pure rotation of the form

$$^A\vec{p} = {}^A_B R^B \vec{p} \tag{3.2}$$

[1] All figures presented in this section are examples given in [2]

where

$$^{B}\vec{p} = \vec{r}_1 \tag{3.3}$$
$$^{A}\vec{p} = \vec{r}_2 \tag{3.4}$$

The elements of the rotation matrix $^{A}_{B}R$ in equation 3.2 are given by

$$\begin{aligned}
a_{11} &= ({s_x}^2 - 1)(1 - cos\theta) + 1 \\
a_{12} &= s_x s_y(1 - cos\theta) - s_z sin\theta \\
a_{13} &= s_x s_z(1 - cos\theta) + s_y sin\theta \\
a_{21} &= s_y s_z(1 - cos\theta) + s_z sin\theta \\
a_{22} &= ({s_x}^2 - 1)(1 - cos\theta) + 1 \\
a_{23} &= s_y s_z(1 - cos\theta) - s_x sin\theta \\
a_{31} &= s_z s_x(1 - cos\theta) - s_y sin\theta \\
a_{32} &= s_z s_y(1 - cos\theta) + s_x sin\theta \\
a_{33} &= ({s_z}^2 - 1)(1 - cos\theta) + 1
\end{aligned} \tag{3.5}$$

where $\vec{s}$ is the screw axis of about which the rigid body is rotated by θ, and is made of components

$$\vec{s} = \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} \tag{3.6}$$

The theory of rotation about an axis can be further expanded, as in Chasles' theorum, to state that the displacement of a rigid body, regardless of how it is displaced, can be described by a rotation about and a translation along some axis.

Given a rigid body that is translated by t and rotated by θ about an axis $\vec{s}$ as is shown in figure 3.2, the new position of the rigid body can be written as

$$^{A}\vec{p} = \vec{s}_0 + t\vec{s} + (\vec{p}_1 - \vec{s}_0)cos\theta + \vec{s} \times (\vec{p}_1 - \vec{s}_0)sin\theta + [(\vec{p}_1 - \vec{s}_0)^T \vec{s}]\vec{s}(1 - cos\theta) \tag{3.7}$$

where $\hat{s}$ is the screw axis and $\vec{s}_0$ is a translation vector to the screw axis. $\vec{p}_1$ denotes the original position of the rigid body. This combination of rotation and

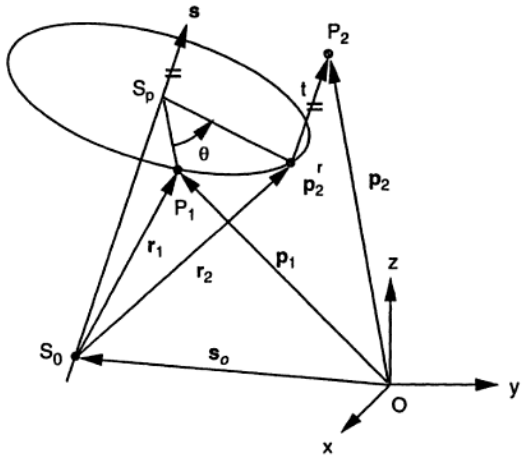


Figure 3.2: Rotation and Translation of a Rigid Body

translation is called a screw displacement. A homogeneous transformation can be written, based on the rotation and translation, associated with the screw displacment. Simplifying 3.7,

$$^{A}\vec{p} = A^{B}\vec{p} \tag{3.8}$$

where A is the homogeneous transformation matrix associated with that screw displacement. The components of a A are

$$
\begin{aligned}
a_{11} &= ({s_x}^2 - 1)(1 - cos\theta) + 1 \\
a_{12} &= s_x s_y(1 - cos\theta) - s_z sin\theta \\
a_{13} &= s_x s_z(1 - cos\theta) + s_y sin\theta \\
a_{21} &= s_y s_z(1 - cos\theta) + s_z sin\theta \\
a_{22} &= ({s_x}^2 - 1)(1 - cos\theta) + 1 \\
a_{23} &= s_y s_z(1 - cos\theta) - s_x sin\theta \\
a_{31} &= s_z s_x(1 - cos\theta) - s_y sin\theta \\
a_{32} &= s_z s_y(1 - cos\theta) + s_x sin\theta \\
a_{33} &= ({s_z}^2 - 1)(1 - cos\theta) + 1 \\
a_{14} &= ts_x - s_{0x}(a_{11} - 1) - s_{0y}a_{12} - s_{0z}a_{13} \\
a_{24} &= ts_y - s_{0x}a_{21} - s_{0y}(a_{22} - 1) - s_{0z}a_{23} \\
a_{34} &= ts_z - s_{0x}a_{31} - s_{0y}a_{32} - s_{0z}(a_{33} - 1) \\
a_{41} &= 0 \\
a_{42} &= 0 \\
a_{43} &= 0 \\
a_{44} &= 1
\end{aligned} \tag{3.9}
$$

where θ is the rotation angle and t is the translational distance.

In the consideration of a kinematic chain, screw displacements can be applied succesively to form a transformation from a moving frame to a reference frame. Consider a rigid body guided to a fixed frame by a dyad made of 2 kinematic pairs, called $\$_1$ and $\$_2$ respectively. The first pair connects the first moving link to the reference frame and the second pair connects the second moving link to the reference frame. The axis of the first pair is the fixed joint axis and the axis of the second pair

is the moving joint axis. As the rigid body is rotated and translated along these two joint axis, the total displacement can be considered a rotation about the moving joint axis, followed by a rotation about the fixed joint axis. In this manner, the initial location of the moving joint axis can be used for derivation of the transformation matrix. The resulting transformation matrix is obtained by premultiplication of two successive screw displacements as in,

$$A_r = A_1 A_2 \tag{3.10}$$

where the transformation matrices denote the screw displacement about the moving and fixed joint axis, respectively. This proposition can then be extended to an n link manipulator where a reference frame is chosen arbitrarily. Then, this method can be thought of as the displacement of the manipulator from a reference position to a target position by a series of finite screw displacments, each about a joint axis. By using these screw displacements and premultiplying, a total rotation and translation for the manipulator becomes

$$A_f = A_1 A_2 \cdots A_{n-1} A_n \tag{3.11}$$

It can be easily observed that, for a manipulator, in a revolute joint $t_i = 0$ and in a prismatic joint $\theta_i = 0$. The joint variables associated with the reference position are subtracted from those associated with the target position and, using these, the forward kinematics are computed directly as in

$$\vec{p} = A_1 A_2 \cdots A_n \vec{p}_0 \tag{3.12}$$

An example of the Stanford manipulator and associated screw vectors is shown in figure 3.3. All information needed to calculate forward kinematics in screw theory

has now been characterized.

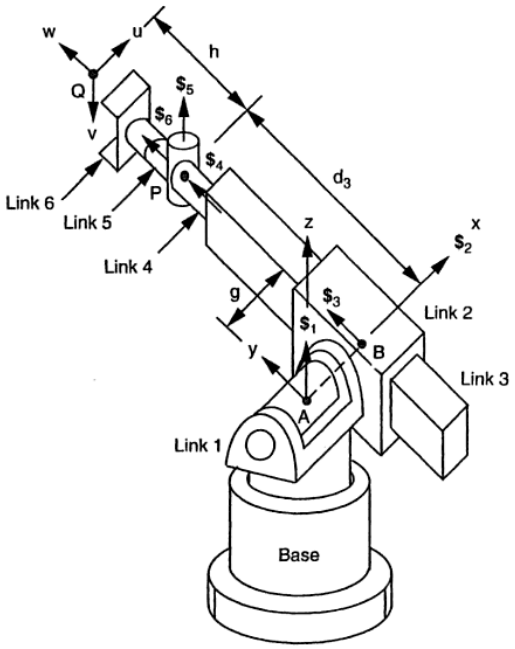


Figure 3.3: Stanford Manipulator with Screw Axes

Screw theory inverse kinematics can be derived in two ways. Due to the nature of the screw theory vectors, there is almost always an analytical solution based on these vectors to the inverse kinematic problem. This is because screw vectors are geometrically based on the physical characteristics of the arm. In most cases, this analytical method yields less potential solutions than an iterative solution.

The inverse kinematics can also be solved in the same manner as the DH convention, using a jacobian and an iterative method[18]. Like the analytical solution, the components of the screw based jacobian come directly from the screw vectors of the manipulator. In order to calculate the screw based jacobian, a screw based coordinate system must be defined. The coordinates of a unit screw are defined as

$$\hat{\$} = \begin{bmatrix} \hat{s} \\ \vec{s_0} \times \hat{s} + \lambda \hat{s} \end{bmatrix} \tag{3.13}$$

The vector $\vec{s_0} \times \hat{s}$ defines the moment of the screw axis about a static reference frame. For a revolute joint,

$$\hat{\$} = \begin{bmatrix} \hat{s} \\ \vec{s_0} \times \hat{s} \end{bmatrix} \tag{3.14}$$

However, displacement is not determined until the intensity of the screw axis is specified, leading to

$$\$ = \dot{q}\hat{\$} \tag{3.15}$$

This equation defines the twist, where $\dot{q}$ is the intensity of the twist. For a manipulator, the tip is being twisted simultaneously about the joint axis. These instantaneous twists can be added linearly to describe the motion of the end effector. Kinematics can then be written as

$$\$_n = \sum_{i=1}^{n} \dot{q}_i \hat{\$}_i \tag{3.16}$$

These screws and unit screws are then used to compose the Jacobian in the static reference frame. In screw theory the Jacobian is with respect to

$$\dot{x} = \begin{bmatrix} \omega_n \\ v_0 \end{bmatrix} \tag{3.17}$$

which is defined in terms of the angular velocity of the end effector and the linear velocity of a reference point which is instantaeously coincident with the origin of the reference frame in which the screws are expressed. The velocity of the end effector is then written as

$$\dot{x} = \begin{bmatrix} \omega_n \\ v_0 \end{bmatrix} = \sum_{i=1}^{n} \dot{q}_i \hat{\$}_i \tag{3.18}$$

which can be simplified to

$$^{j}\dot{x} = {}^{j}J\dot{q} \tag{3.19}$$

It is observed that, from this, the Jacobian is simply

$$J = \left[\hat{\$}_1, \hat{\$}_2, \cdots, \hat{\$}_n\right] \tag{3.20}$$

which is calculated directly from the scew axis $\hat{s}$ and the vector to the screw axis $\vec{s_0}$. The Jacobian is useful for a wide range of functions, including inverse kinematics and controls, and it is beneficial to be able to derive this directly from the manipulator kinematics.

In this section, screw theory kinematics and implementation have been discussed. The subject of this thesis is dependant on the functionality and use of these methods. The forward and inverse kinematics solutions provide usefull insight into the understanding of kinematic methods for solving reconfigurable and other unmodeled manipulators.

Chapter 4: Method Development

4.1 Importance of Method Development

Most of the previous work in kinematic determination for earth-based manipulators uses DH convention. This convention perscribes a specific method of joint frame attachment, thus defining the relationship between joints. Based on this specific frame assignment, a zero-pose for the manipulator is pre-defined upon which the kinematics must be based. For an unmodeled manipulator using DH convention, an optimization method is required to find this specific way of relating joints and the zero-pose of the robot. Simply by the way the DH convention is defined, a kinematic determination method has added complexity in order to attempt to search for pre-defined link relationships and a zero-pose of the manipulator. Screw theory, in this application, removes the need for pre-defined relationships. Because the zero-pose and reference frame can be defined arbitrarily as is most convenient for the mathematics, much of the complexity of the math and optimization needed is eliminated, while still producing fully defined kinematics for the arm. As proven in [4], this process can be done with a single accelerometer and gravity. A method was therefore developed for this thesis in order to determine the kinematics of a micro-gravity manipulator based on a single IMU, with acceleration and angular rate, and using screw theory for the kinematic determination. The goal of the method is to produce a simple mathematical calculation by which, only having knowledge of the encoder counts per joint revolution in the manipulator, the forward kinematics and Jacobian can be easily determined.

The following chapter details the mathematics of a novel application of screw theory in kinematic determination. This new method depends only on information

from a single IMU at the tip of the manipulator and encoder counts in the joints and is applicable specifically to serial revolute manipulators. A discussion of screw theory as an integral part of the mathematical system is presented first. The calculation of the screw vector and components of the measured angular velocity in the screw vector direction is derived directly from a unique application of the screw theory technique. Equation 2.19 is presented using measured and derived values and rearranged as system of linear equations. A solution to the system of linear equations using a truncated SVD produces the vector to the screw axis for each joint. Once the screw axis and the vector to the screw axis are calculated, these values are the complete the forward kinematics for the manipulator.

4.2 Kinematic Determination Method

Screw theory describes the orientation of a body in 3D space by a unit vector about which the body is rotated and an angle describing how far the body has been rotated about this axis. The description can be further expanded as a rotation about an axis and a translation along an axis to describe both the orientation and position of a body relative to a static frame. In a manipulator, a series of these rotation and translation transformations, or screw displacements, is used to describe the position and orientation of a moving frame, usually the manipulator tip, relative to a chosen static frame, in most cases the base of the robot. Each joint represents a screw displacement where the defining characteristics are the axis of rotation, or screw axis, and vector from the static frame to the screw axis of the joint. These two vectors completely define the manipulator kinematics. Thus, the first step in kinematic determination is to calculate the screw axis and vector to the screw axis for each joint.

In order to determine the unit vector screw axis and vector to the screw axis

for each joint, the placement of the static frame and the IMU, or the moving frame, is key. In order for the IMU to take measurements describing the motion of each joint, the IMU and coincident moving frame must be placed somewhere beyond the last manipulator joint frame. Conventional screw theory, as applied to robotics, places the static frame at the base of the manipulator. Since there is an unknown relationship between the joint screw axis and the base of the manipulator, this convention is no longer applicable. However, the relationship between the joint screw axis and a frame at the tip of the manipulator is always the same, when moving only a single joint at a time, and clearly shown in 2.19. Constant parameters through a joint motion can be described by

$$\vec{s}_0, \hat{s} = const \tag{4.1}$$

From the perspective of the frame moving in an arc, the screw axis and vector to the screw axis always look the same numerically as in equation 4.1. Because of this constant defined relationship between IMU and every joint, it is mathematically convenient to define the static frame as a frame placed coincident with the IMU frame in the initial manipulator pose. This is true for any random manipulator start pose. A position measurement will thus be taken from the origin of this static frame to the origin of the moving frame, or in other words, from the origin of the IMU frame in the initial robot pose to the origin of the current IMU frame.

To find the unit vector screw axis, only the angular velocity values as measured by the IMU are needed. In fact, due to the properties of circular motion

$$\hat{s} = \hat{w} \tag{4.2}$$

and can be taken directly from the angular velocity in an ideal case. From the previous discussion of circular motion, the acceleration of the IMU frame is given in

equation 2.19. This is solved for

$$\vec{r} = \vec{s_0} \tag{4.3}$$

A single differentiation of the angular velocity is used to find the angular acceleration at each point of the arc trajectory. At this point, all the values needed to solve for $\vec{s_0}$ in equation 2.19 have been generated.

In order to solve for the vector to screw axis $\vec{r}$, it is easiest to write 2.19 as a system of linear equations. To write 2.19 as a system of equations, each cross product in the equation can be written as a skew symmetric matrix. The individual components of 2.19 in skew symmetric form become:

$$\vec{\omega} \times \vec{r} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} \omega_2 r_3 - \omega_3 r_2 \\ \omega_3 r_1 - \omega_1 r_3 \\ \omega_1 r_2 - \omega_2 r_1 \end{bmatrix} \tag{4.4}$$

$$\vec{\omega} \times (\vec{\omega} \times \vec{r}) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \begin{bmatrix} \omega_2 r_3 - \omega_3 r_2 \\ \omega_3 r_1 - \omega_1 r_3 \\ \omega_1 r_2 - \omega_2 r_1 \end{bmatrix} \tag{4.5}$$

$$\vec{\alpha} \times \vec{r} = \begin{bmatrix} 0 & -\alpha_3 & \alpha_2 \\ \alpha_3 & 0 & -\alpha_1 \\ -\alpha_2 & \alpha_1 & 0 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} \alpha_2 r_3 - \alpha_3 r_2 \\ \alpha_3 r_1 - \alpha_1 r_3 \\ \alpha_1 r_2 - \alpha_2 r_1 \end{bmatrix} \tag{4.6}$$

When the skew symmetric forms are substituted back into 2.19, the new equation becomes

$$\vec{a} = \begin{bmatrix} \omega_3\omega_1 r_3 - {\omega_3}^2 r_1 - {\omega_2}^2 r_1 + \omega_2\omega_1 r_2 \\ -{\omega_3}^2 r_2 + \omega_2\omega_3 r_3 + \omega_1\omega_2 r_1 - {\omega_1}^2 r_2 \\ \omega_2\omega_3 r_2 - {\omega_2}^2 r_3 - {\omega_1}^2 r_3 + \omega_3\omega_1 r_1 \end{bmatrix} + \begin{bmatrix} \alpha_2 r_3 - \alpha_3 r_2 \\ \alpha_3 r_1 - \alpha_1 r_3 \\ \alpha_1 r_2 - \alpha_2 r_1 \end{bmatrix} \tag{4.7}$$

It can be seen that there is a vector $\vec{r}$ such that

$$A\vec{r} = \vec{a} \tag{4.8}$$

where A is a matrix of coefficients that when multiplied by $\vec{r}$ give the right-hand side of equation 4.7. A is of the form

$$A = \vec{\alpha} \times + \vec{\omega} \times (\vec{\omega} \times) = \begin{bmatrix} (-{\omega_3}^2 - {\omega_2}^2) & (-\alpha_3 + \omega_2\omega_1) & (\omega_3\omega_1 + \alpha_2) \\ (\omega_1\omega_2 + \alpha_3) & (-{\omega_3}^2 - {\omega_1}^2) & (-\alpha_1 + \omega_3\omega_2) \\ (-\alpha_2 + \omega_1\omega_3) & (\omega_2\omega_3 + \alpha_1) & (-{\omega_2}^2 - {\omega_1}^2) \end{bmatrix} \tag{4.9}$$

Substituting A back into 4.8 gives the final equation

$$\begin{bmatrix} (-{\omega_3}^2 - {\omega_2}^2) & (-\alpha_3 + \omega_2\omega_1) & (\omega_3\omega_1 + \alpha_2) \\ (\omega_1\omega_2 + \alpha_3) & (-{\omega_3}^2 - {\omega_1}^2) & (-\alpha_1 + \omega_3\omega_2) \\ (-\alpha_2 + \omega_1\omega_3) & (\omega_2\omega_3 + \alpha_1) & (-{\omega_2}^2 - {\omega_1}^2) \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{4.10}$$

which can be solved for $\vec{r}$ in most cases.

Usually, a system of equations like 4.10 can be solved by taking the inverse of A and calculating

$$A^{-1}\vec{a} = \vec{r} \tag{4.11}$$

However, in this case, the A matrix is always singular, which means that the inverse of A cannot be used in the solution for $\vec{r}$. The A matrix is singular because a vector to the screw axis can be from the static frame to any point on the screw axis, leading to non-unique, i.e. infinately many, solutions to equation set 4.10.

In order to resolve the issue of a non-invertable A matrix, a truncated version of the singular value decomposition can be used. In this case, the application of the truncated SVD solves the problem of a non-invertable matrix and presents a minimum norm solution. The SVD is a mathematical rearrangement which decomposes

a matrix into 2 rotation matrices, U and V^*, and one scaling transform matrix Σ along the rotated axis. Equation 4.8 can be re-written as

$$U\Sigma V^T\vec{r} = \vec{a} \tag{4.12}$$

where A is decomposed into 3 matrices. Equation 4.12 can be solved for $\vec{r}$ by rearranging. The equation then becomes

$$\vec{r} = V\Sigma^{-1}U^T\vec{a} \tag{4.13}$$

By definition of a truncated SVD [3], 4.13 can be rewritten as

$$\vec{r} = \sum_{i=1}^{n} \frac{\vec{u_i}^T\vec{a}}{\sigma_i}\vec{v_i} \tag{4.14}$$

where n is the number of columns in U and V, u_i is the ith column of U, v_i is the ith column of V, and σ_i refers to the diagonal value on the ith column of Σ. For zero values of σ_i it is obvious that equation 4.14 still gives a singular result. Therefore, the 4.14 expansion can be truncated by discarding values of σ_i which are zero and limiting the SVD to some number of columns $p < n$. The resulting $\vec{r}$ is the minimum norm solution, or least squares solution, and is the shortest distance from the origin of the static frame to the joint screw vector. This vector is $\vec{s}_{0,i}$, or the vector to the screw axis. A $\vec{s}_{0,i}$ vector is calculated for each point around a joint arc and will be identical at every point. Now both the screw axis $\hat{s}_i$ and vector to the screw axis $\vec{s}_{0,i}$ have been calculated and the forward kinematics for the manipulator can be determined.

Based on the $\hat{s}_i$ and $\vec{s}_{0,i}$ vectors, the screw displacement matrices for each joint can be calculated. The transformation matrix A is composed as per [2] and detailed in section 3. By using these matrices in series, a displacement of the tip from one

position to another can be calculated. If p_0 refers to a position vector in the static frame, then the new position p is calculated by

$$p = A_1 A_2 A_3 A_4 ... A_N p_0 \tag{4.15}$$

Thus, the kinematic solver can solve for any tip position in the static reference frame. It is worth noting that in a constant gravity environment, the accelerometer will have different values. A second acutation of the joint arc at a much slower trajectory can be used to measure these values at every joint angle. Once subtracted from the high speed values, the mathematics procedure above applies. Thus, the kinematic estimator can be applied in both micro-gravity and constant gravity environments.

4.2.1 Error Mitigation

Unlike an ideal case, real world systems have measurement error, which must be dealt with and replicated in any simulation of the kinematic determination model. This will also give an indication of how much measurement error the system can tolerate and the best methods of filtering out noise.

When the $\hat{s}$ vector is calculated in a system with error, it produces a wide range of answers due to the error in angular velocity ω. In order to remove error in the calculation injected by angular velocity measurements, a singular value decomposition (SVD) is used once again. In this application, the SVD will be used to indicate the strongest direction of the angular velocity. The SVD technique is used with the matrix of ω. Generally written,

$$\omega = B \Sigma W^* \tag{4.16}$$

W^* contains the orthonormalized eigenvectors of $\omega^T \omega$. The first column of

W^* corresponds to the highest non-zero singular value[1] of the ω matrix. The highest eigenvalue gives an indication of the dominant direction of ω, or a vector with magnitude in the direction of the joint screw axis. This "best fit" vector can be transformed into a unit vector and will be the solution to the screw axis. For an eigenvector and eigenvalue in W^* that solve for $\omega^T\omega$, there is a different eigenvalue and unit vector that also solve for $\omega^T\omega$. Since the unit vector solution is desired, using the built in Matlab functions provide the unit vector solution to the SVD automatically. The produced eigenvector is the unit vector screw axis. Thus, the screw axis for a joint is provided directly by an application of singular value decomposition to the angular rates for an arc motion.

Since a differentiation of ω is needed to calculate the angular acceleration $\vec{\alpha}$, the angular acceleration can be inaccurate due to other noise in ω. One way to improve the angular acceleration values, and thus the estimation of $\vec{r}$, is to limit the values of angular velocity $\vec{\omega}$ to only those components that point in the screw axis $\hat{s}$ direction. This is done by taking the dot product of the angular velocity and screw axis.

$$\vec{\omega}_{new} = (\vec{\omega} \cdot \hat{s})\hat{s} \tag{4.17}$$

With the calculation of angular acceleration, and the measurement of linear acceleration and angular velocity by the IMU, the system can be solved for the vector to the screw axis $\vec{r}$ as described in section 4.2. Additionally, for a system with error, these $\vec{r}$ vectors are averaged together over the whole arc motion in order to calculate the final $\vec{s}_{0,i}$.

In this chapter, a new method for definition of unmodeled forward kinematics is defined. A screw theory based solution is presented, using data from a single IMU placed on the tip of a manipulator to solve for the screw axis vectors of the joints and the vectors from the static frame origin to the joint screw axis. The method

[1] A singular value is defined as the square root of an eigenvalue

of using these calculated vectors in forward kinematics is then shown. The method is then considered when there is error in the IMU values and steps are taken to minimize influence. A full 6 DOF definition of the tip of the robot can then be calculated based solely on accelerometer and gyroscope data from the IMU.

Chapter 5: Kinematic Determination Method Simulation

5.1 Mathematical Simulation

A Matlab simulation was created in order to test the mathematial theory of the kinematic analysis method presented in section 4.2. A brief description of the simulation is included in this section. Variation of user defined parameters is used to collect data on the relationship of tip position error and screw vector error to simulated IMU measurement errors. An analysis of the simulation data is used to determine feasibility for use of this method and the initial parameters for a hardware based test.

The mathematical model presented in 4.2 is used as the basis for the generation of simulated linear acceleration and angular rotation values. Given a set of arm parameters, or a screw axis and vector to the screw axis, for each joint, the model is capable of generating the position of the manipulator tip in the static frame. These positions are then differenced twice to generate simulated accelerations. The simulated angular rate is the derivative of a user-input joint angle function. After these values are generated, to simulate a real world system, zero-mean error can be added to the system. At this point, just as with real IMU data, the simulated accelerations and angular rates are input back into the model as described in 4.2 and the program outputs a set of estimated screw axis vectors and vectors to the screw axis for each joint.

The advantage of a simulated mathematical model is the capability of the user to change the manipulator and error parameters of the simulation to reflect different scenarios. The user configurable variables in this simulation are:

- $\hat{s}_i$ and $\vec{s}_{0,i}$

- Number of joints
- Joint type
- Input angle function to the joint
- Simulated IMU sampling interval
- Joint angle arc period
- Joint arc sweep angle
- Error in acceleration (in Gs)
- Error in angular rate (in rad/s)

A generic sinusoidal joint angle input function, containing both angular velocity and angular acceleration at all points, was used to generate simulation data.

In order to validate the kinematic model position and vector estimation, a simulation analysis was performed using a 5 DOF manipulator with 0% error in the generated IMU measurements. The example manipulator constructed is of the form found in figure 5.1

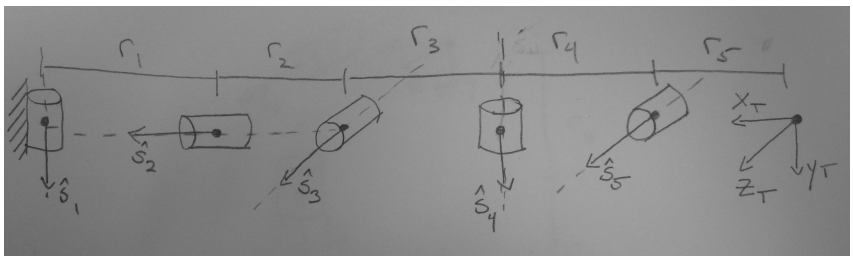


Figure 5.1: Simulated Manipulator

This manipulator has joint screw axis $\hat{s}$ and vectors to the screw axis $\vec{s}_0$ shown in tables 5.1a and 5.1b. These vectors can be seen with visual inspection of the manipulator diagram in figure 5.1.

$\hat{s}_1$	$\hat{s}_2$	$\hat{s}_3$	$\hat{s}_4$	$\hat{s}_5$
0	1	0	0	0
1	0	0	1	0
0	0	1	0	1

(a) Simulated Arm Screw Vectors

$s_{0,1}$	$s_{0,2}$	$s_{0,3}$	$s_{0,4}$	$s_{0,5}$
1.2	0	0.7	0.6	0.3
0	0	0	0	0
0	0	0	0	0

(b) Simulated Arm Vectors to Screw Axis

Table 5.1: Simulated Arm Characteristics

With 0% error added to the generated IMU measurements in the system, the manipulator vectors, as calculated by the kinematic estimator, are shown in table 5.2.

$\hat{s}_1$	$\hat{s}_2$	$\hat{s}_3$	$\hat{s}_4$	$\hat{s}_5$
0	1	0	0	0
1	0	0	1	0
0	0	1	0	1

(a) Ish Screw Vectors

$s_{0,1}$	$s_{0,2}$	$s_{0,3}$	$s_{0,4}$	$s_{0,5}$
1.20007398	0	0.7000432	0.60003699	0.3000185
0	0	-0.0000021398	0	-0.000000918
0.000003669	0	0	0.000001835	0

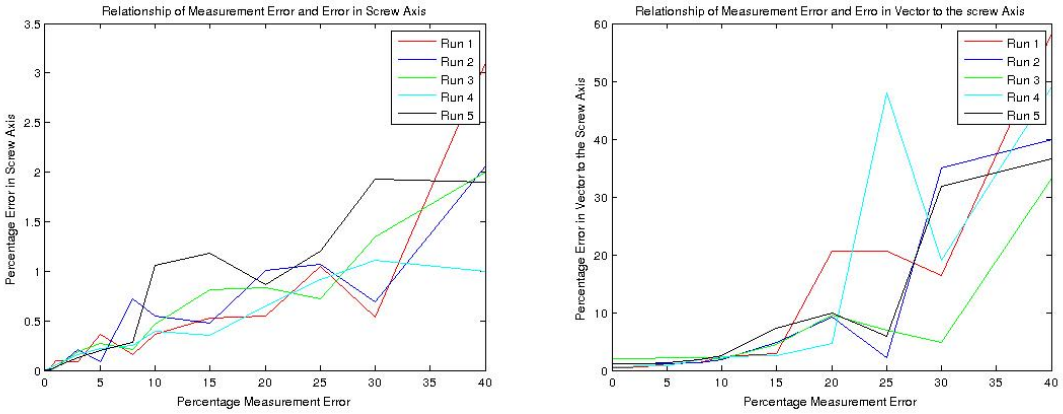
(b) Calculated Vectors to Screw Axis

Table 5.2: Calculated Manipulator Parameters

There are minor differences in the exact and calculated $\vec{s_0}$ vectors, all of which are less than $\frac{1}{10}$ of a millimeter in magnitude. These slight differences are likely caused by rounding in the mathematical operations of the kinematic estimator and are considered negligible for the purposes of this analysis. As demonstrated by a 5 DOF, 0% error manipulator analysis, the simulation succesfully verifies the operation of the kinematic calculation method and implementation in software.

One of the most important influences on the kinematic model estimation is the effect of error in the generated IMU values on the screw vectors and position of the manipulator tip. Error is applied to the simulated acceleration and angular rate values evenly, i.e. 5% to both acceleration and angular rate at the same time. The

error added is some percentage of the true maximum acceleration and angular rate generated by the first part of the simulation. It should be noted that, for example, 5% error refers to a psuedo-randomly generated error array where the maximum values on either side of zero are bounded by 5% of the true acceleration or angular rate values. Like a real IMU measurement, some error added will be less than the maximum/minimum possible value. Screw vectors are calculated for a range of error percentages in the measured values and compared to the exact screw vectors. The results of this comparison are shown in figures 5.2b and 5.2a.



(a) Error in the Screw Axis (b) Error in Vector to the Screw Axis

Figure 5.2: Comparison of IMU error with Screw Vector Error

As expected, error in the screw vectors increases with measurement value error. However, an increase in measurement value error has a much smaller effect on the screw axis value than the $\vec{s}$ value. The relationship between measurement error and $\vec{s}$ could be exponential, exhibiting an increase in slope at about 30% measurement error. In order to keep the error in $\vec{s_0}$ less than 10%, the measurement error must be below $20 - 25\%$. The relationship of error with $\hat{s}$ is relatively linear, with a 30% measurement causing 1% error in the screw axis values.

With the result of the effect of measurement error on screw vectors, the total effect of this error on tip position can be calculated. Figure 5.3 shows the cumulative effect of measurement error on tip position.

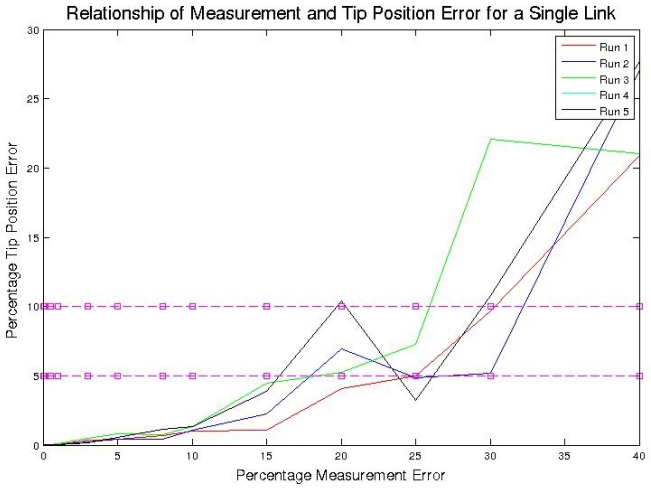


Figure 5.3: Influence of measurement error on tip position

In order to have less than 10% error in tip position, a measurement error below $25 - 30\%$ is desirable for a single link. However, these measurement errors become cumulative as the number of DOF in the manipulator increases. Five runs were taken for each of 2-5 DOF. The average for each DOF is shown in figure 5.4. The individual trial runs for each DOF can be found in Appendix C.

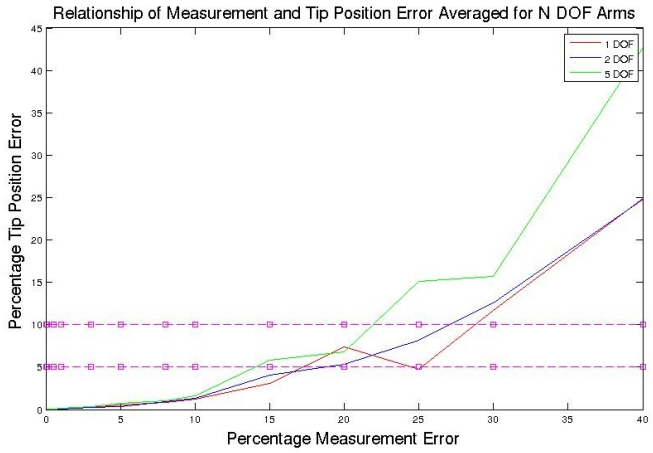
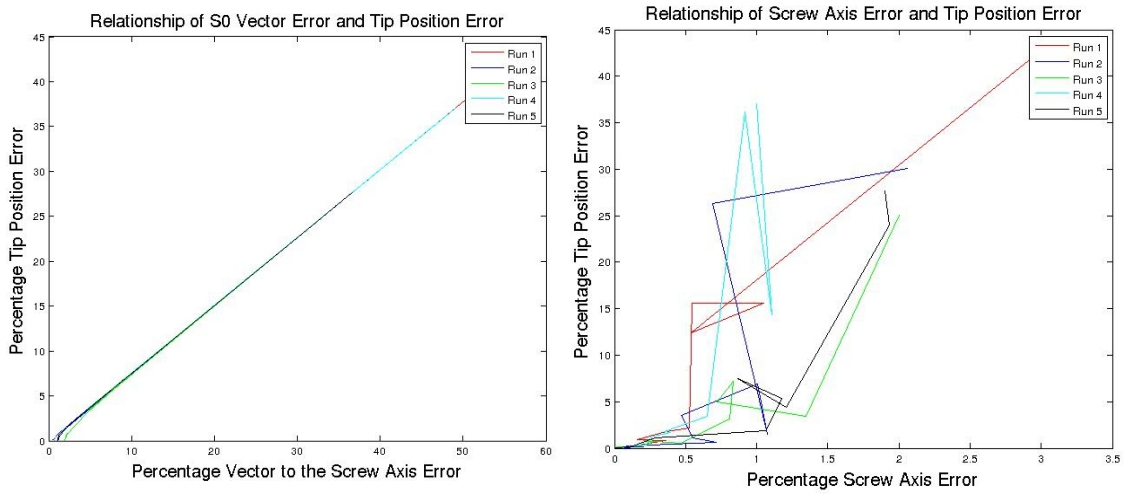


Figure 5.4: Influence of measurement error on tip position for N DOF

For a 1-2 DOF arm, $25 - 30\%$ measurement error is acceptable for a less than 10% position error. In a 5 DOF manipulator, this number is reduced to about 20% measurement error. This is also, as seen in figure 5.2b, the maximum percentage

measurement error to keep $\vec{s}_0$ with less than 10% error.

The relationship of measurement error to both the screw vector and tip position error can be used to show the effect of screw vector error directly on tip position error. Figures 5.5a and 5.5b show the effect of error in $\vec{s}_0$ and $\hat{s}$ on error in the final tip position.



(a) Effect of $\vec{s}_0$ Error

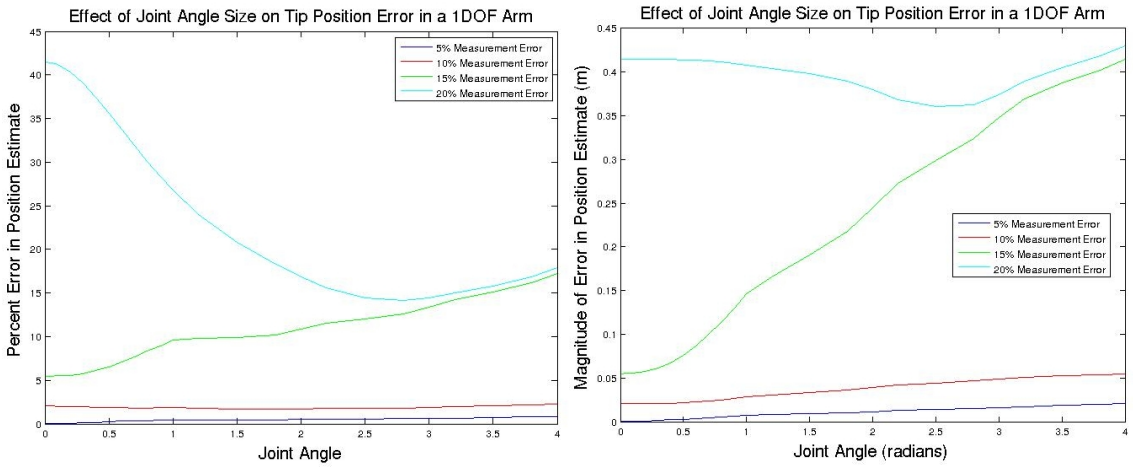
(b) Effect of $\hat{s}$ Error

Figure 5.5: Comparison of Screw Vector Error with Tip Position Error

The relationship of $\vec{s}_0$ error with tip position error is almost perfectly linear in all cases. With $\hat{s}$, very small error causes large changes in tip position. In general, to remain below the 10% position error, less than 20% error in $\vec{s}_0$ and less than 1.5% error in $\hat{s}$ is desirable. This is a confirmation of the information given by the influence of IMU measurement error on these values and tip position error.

With error in $\vec{s}_0$ and $\hat{s}$, it is hypothesized that larger joint angles would emphasize the error in these vectors as shown in the tip position. Further, more joints actuated at larger angles would increase the error in tip position by propogating the screw vector error through the kinematic calculation and multiplying it by these increasingly larger angles. In order to test this hypothesis, a joint with constant 10% IMU measurement error, and thus constant screw vector error, was used to

compare size of the actuated angle and the resulting tip position error. The percent error and magnitude of the error in position estimate are shown in figure 5.6.



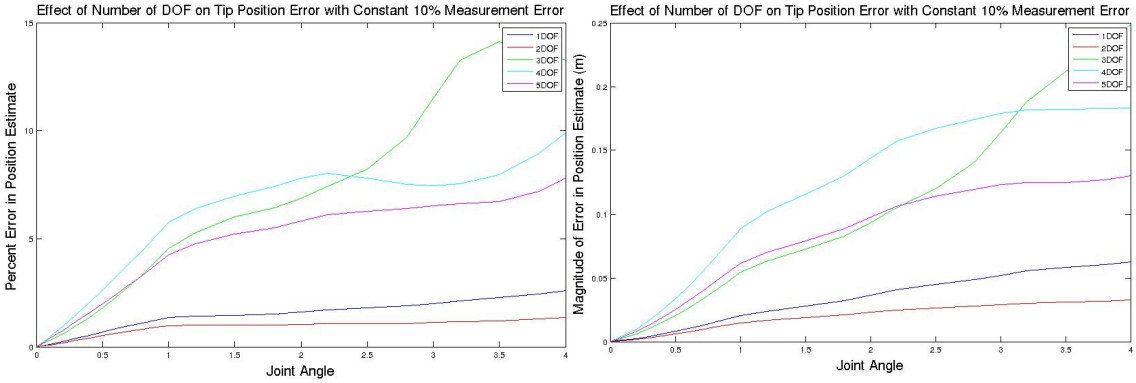
(a) Percentage Position Error with Angle Size (b) Magnitude Position Error with Angle Size

Figure 5.6: Effect of Actuated Angle Size of Tip Position Error

In most cases, the percentage position error for a small angle is larger than the percentage position error after a large actuation. This is due to the constant error in a small versus large motion. The difference in the slopes of the 5%/10% and 15%/20% error lines in the magnitude of the position error is also significant. 15%/20% measurement error begins with a much higher magnitude of position error and increases at a faster rate. 5%/10% measurement error yields a much lower initial position error magnitude and increases more slowly, almost not at all in fact, as the joint angle increases. Based on figure 5.6b, 5% or 10% error in IMU measurement values is ideal as the joint angles increase. This is a smaller percentage than the percentage of measurement error determined by looking at measurement error and tip position for a single pose found previously.

As before, it is hypothesized that as the number of DOF increase, so too the percent error in position will increase as joint angles increase. 2-5 DOF values were taken and averaged over several runs in order to determine an exact relationship for joint angle and percent position error. Individual runs can be found in Appendix

D with varying measurement error for each set of DOF. A summary of the average values for a 2-5 DOF arm is shown in figure 5.7 with a constant 10% measurement error.



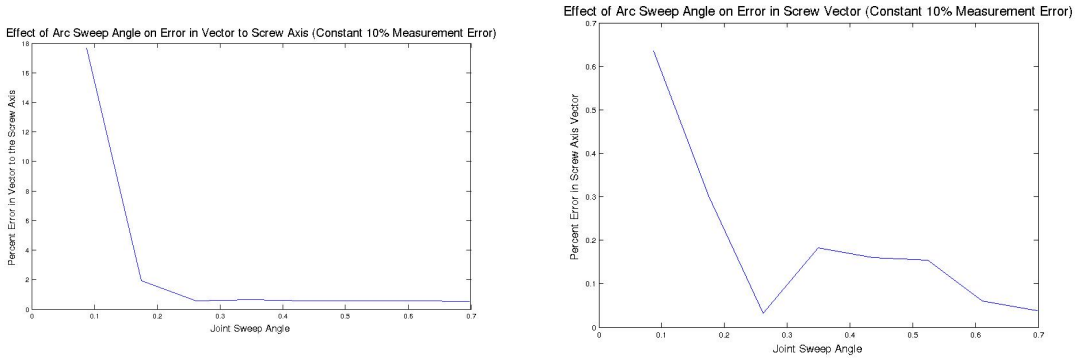
(a) Percentage Position Error with Angle Size(b) Magnitude Position Error with Angle Size

Figure 5.7: Effect of Actuated Angle Size on Tip Position Error NDOF

Different percentages of measurement error over 1-5 DOF exhibit the same trends. It is interesting that as number of DOF increases to 3, the percentage and magnitude of the error in position increases. However, for 4 and 5 DOF manipulators, the percentage and magnitude of the position error decreases for high joint angles. With percentages, this trend might be expected as the size of the motion is greater than the constant measurement error input. However, the slope of the error magnitude also changes and at high DOF, the magnitude of the position error is less than what it was for 3 DOF. 5 DOF position error is also less then 4 DOF position error for high angles. This is an interesting and unexpected trend that may be related to the configuration of the simulated manipulator. However, for large joint angles, the results suggest that a high DOF manipulator may be best in order to minmize tip position error.

Variation in how the input parameters, such as screw vectors, are measured also changes the accuracy of the results. Each joint is actuated in an arc to generate acceleration and angular rate data. The defining characteristics of the arc are the

sweep angle and period. Changes to each of these parameters changes the accurary of the estimated $\vec{s}_0$ and $\hat{s}$. Changes in the sweep angle of the arc as related to the error in $\vec{s}_0$ and $\hat{s}$ are shown in figure 5.8.



(a) $\vec{s}_0$ Error with Arc Size

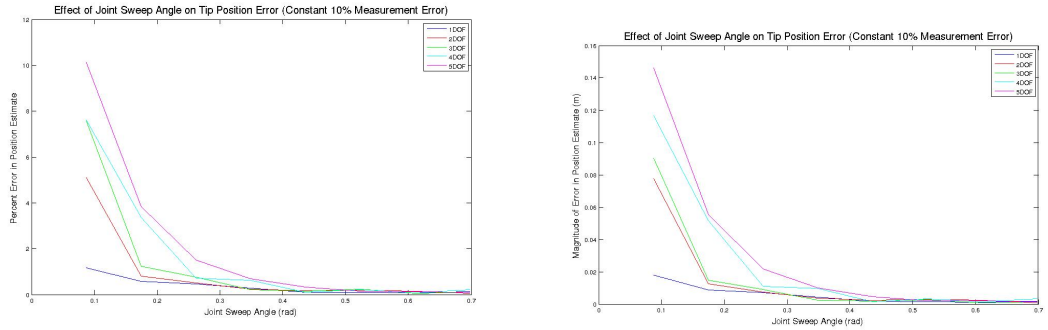
(b) $\hat{s}$ Error with Arc Size

Figure 5.8: Effect of Arc Size on Screw Vector Error

There is a drastic decrease in the $\vec{s}_0$ and $\hat{s}$ error as the angle is increased from 5 to 10 degrees. Further increases do not decrease the $\vec{s}_0$ error significantly. Error in $\hat{s}$ remains below 0.6% regardless of changes in sweep angle, and after an increase to 10 degrees, remains below 0.2% error in $\hat{s}$. In order to keep error in the screw vectors low, a minimum sweep angle of 10 ° should be used in the joint actuation to generate acceleration and angular rate values.

Since screw vector error is related to tip position error, a comparison of sweep angle size and tip position error is also useful. Again, cumulative error over multiple DOF affects tip position error. Figure 5.9 shows a comparison over multiple DOF of tip position error and the sweep angle used to evaluate each of the DOF.

There is, matching the trend of sweep angle size and screw vector error, a drastic decrease in position error as the sweep angle is increased from 5 to 10 degrees for lower DOF, and 5 to about 20 degrees for higher DOF. Because large sweep angles gather more data, a better average of the screw vector can be calculated over shorter sweep angles and thus a higher sweep angle will give more accurate position results. Higher DOF, as expected, have a higher position error for a larger range

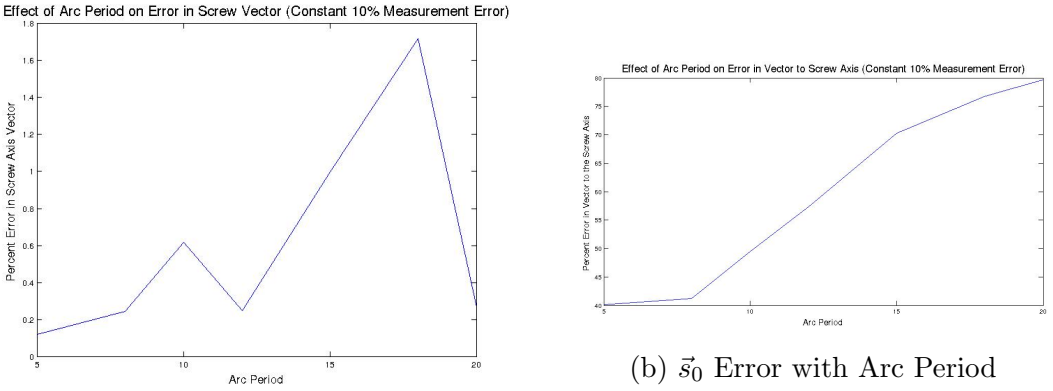


(a) Magnitude of Position Error with Arc Size (b) Percent Position Error with Arc Size

Figure 5.9: Effect of Arc Size on Position Error

of small sweep angles. After about 20 degrees, further increases in sweep angle size do not have significant reduction of tip position error. For a 1-3 DOF manipulator, a minimum sweep angle of 10° is ideal. For a 4-5 DOF manipulator, a minimum sweep angle of 20° is ideal.

Sweep arc period also has an impact on the accuracy of the screw vectors, and thus the error in tip position. Figure 5.10 shows the effect of increasing the arc period, with a constant arc sweep angle of 10°, on the accuracy of the screw vectors.



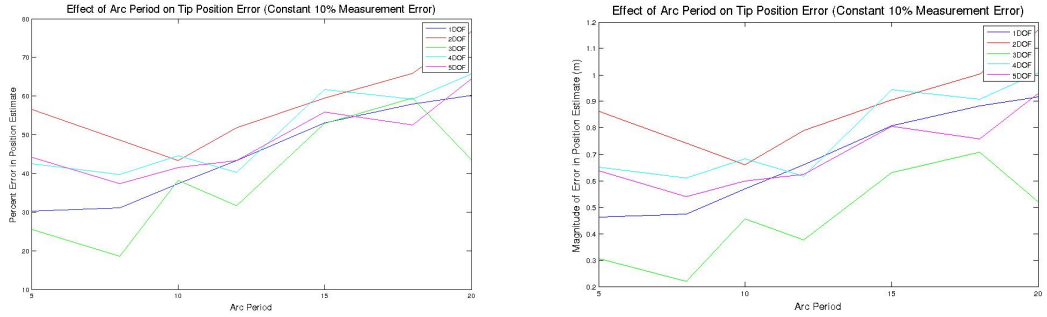
(a) $\hat{s}$ Error with Arc Period (b) $\vec{s}_0$ Error with Arc Period

Figure 5.10: Effect of Arc Period on Screw Vector Error

An arc period of greater than approximately 7 seconds increases the error in the $\vec{s}_0$ significantly. This result shows that an arc should have a constant speed approxmation of greater than 1.4°/s for a system with a constant measurement error

of 10%. If the measurement error increases, the constant speed requirement for the arc period will also increase.

As before, the error in tip position can be related to the arc period through the screw vectors. Figure 5.11 shows the relationship of arc period and tip position error for multiple DOF manipulators.

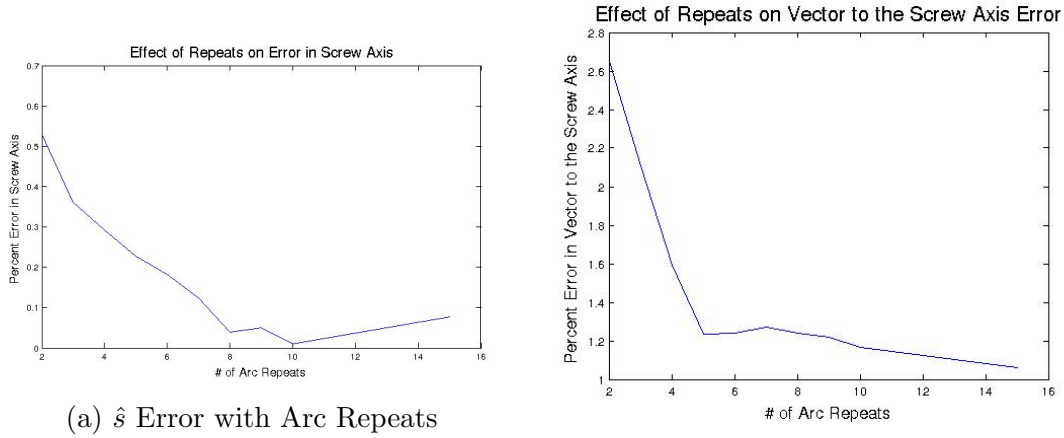


(a) Position Error with Arc Period (b) Position Error with Arc Period

Figure 5.11: Effect of Arc Period on Position Error

Surprisingly, changing the period of the arc does not significantly affect position error for a given large angle pose. This may be due to the unique configuration of the simulation manipulators or other factors. Manipulators with DOF 1-5 all also shows about the same effect in position error. There may be other factors in the manipulator, such as measurement error or arc sweep angle, which affect the final position much more than arc period and overwhelm the effects of arc period in the tip position graphs.

The last method increasing accuracy in the screw vectors and position estimates is to increase the number of joint arcs over which acceleration and angular rate data is taken. This method is similar to an increase in joint sweep angle in that more arcs, and thus more data points, will increase the screw vector accuracy by averaging in additional data to mitigate zero-mean error. Figure 5.12 shows the effect of increasing and averaging multiple arcs on the accuracy of the calculated screw vectors.



(a) $\hat{s}$ Error with Arc Repeats

(b) $\vec{s_0}$ Error with Arc Repeats

Figure 5.12: Effect of Arc Repeats on Screw Vector Error

There is a drastic decrease in $\vec{s_0}$ and $\hat{s}$ error as repeats are increased to 8-10 arcs averaged toghether. Increasing number of arcs further has diminishing returns in decreasing screw vector error. However, the total magnitude of error decreases in both vectors is small. $\hat{s}$ error remains below 0.6% in all cases and 4 arc repeats brings error below 0.3%. $\vec{s_0}$ error remains below 2.8% in all cases and 4 arc repeats bring error below 1.4%. Based on these results, 8-10 repeats per arc is ideal, but 4-6 repeats will be adequate for significant reduction of vector error compared to a single arc.

The kinematic estimation simulation provides results that define the relationship between tip position error and measurement error, screw vector error, and input variables which define how measurement data is collected. These results define how well a system will function given constant error and variable values, as might be expected in a real world scenario with an N DOF manipulator. Based on the simulation, a hardware test would be best suited to a 4-5 DOF manipulator collecting joint data with a wide and fast arc actuation. 8-10 arc repeats are ideal but 4-6 arc repeats will reduce screw vector error significantly. Ideally, for a high DOF manipulator, measurement error should be less than 20% of the maximum true values in order to keep tip position error to less than 10%.

Chapter 6: Hardware Experiment

6.1 Experiment Description

A hardware implementation of the kinematic determination method described in section 4.2 was also created in order to test the capability of the model in a real world scenario. The goal of the hardware experiment is to calculate both screw vectors and implement the forward kinematics, comparing "true" position values of the manipulator tip to those positions generated based on the kinematic determination method. A description of the hardware and experiment procedures follows in this chapter, along with expected results and method of results analysis. Error and error mitigation specific to the hardware system is also incorporated into the analysis of the data. Data from the hardware experiment is analyzed and used to investigate the capability of the method and improvements to the experiment which could increase the fidelity of the model. Analysis of the system is completed as compared to the values and expectations generated in the simulation.

In order to test kinematic determination capability, a serial revolute manipulator with several DOF, 4-5 as per the simulation determination, is ideal. The Ranger Mark I manipulator originally specified for the Underwater Neutral Buoyancy Vehicle (NBV) at the Space Systems Lab (SSL) is a 6 DOF arm with RPPRPR joints in that order from base to tip. It is useful in this experiment due its high DOF, easy existing software interface[16], and user-friendly joint angle control. The IMU is placed, as needed by the screw theory kinematics for this method, beyond joint 6 of the manipulator. Joint 6 is not used as an actuated joint since the moment arm from the IMU to joint six is negligible in terms of acceleration generation. Figures 6.1 and 6.2 show the Ranger manipulator with IMU attached.

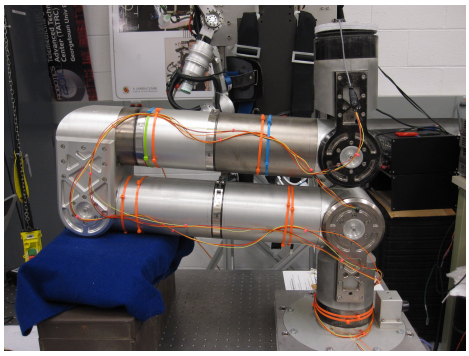


Figure 6.1: IMU as connected to Manipulator

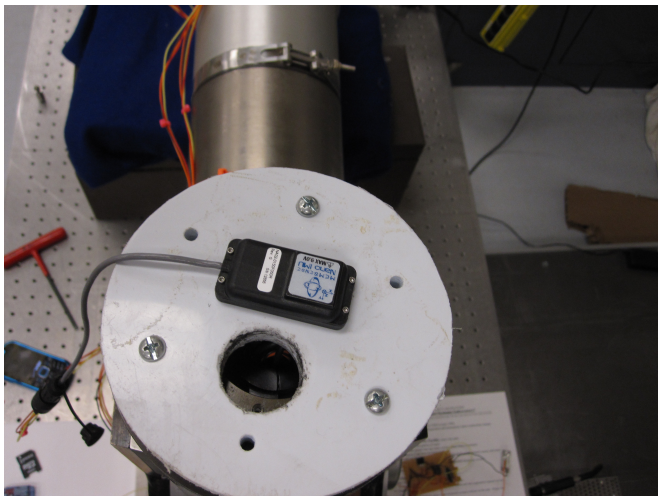


Figure 6.2: IMU as attached to Joint 6 Plate

At this point, the manipulator is set up for joint actuation, which will measure acceleration and angular rate values for each joint. The user can select a manipulator pose that is semi-random from which to perform the joint actuation. Again as per the simulation, an arc sweep angle of 20° is ideal, so a pose where this can be accomplished is necesary without interference from hard stops. A trapazoidal joint angle trajectory between a series of points is used as the input function to the joint. As the joints perform the trajectory, IMU data is taken. A second joint angle trajectory is then performed, identical to the first, but at a much slower rate. This is done in order to measure gravity at each point on the joint trajectory. Once data is taken for each joint at both high and slow speed, the gravity for each joint angle can be removed from the high speed measurements, leaving an acceleration reading that is due only to the angular velocity and acceleration around the arc trajectory. Once this acceleration and angular rate data is measured for each joint, it is used to calculate the screw axis $\hat{s}_i$ and vector to the screw axis $\vec{s_i}$ for each joint of the manipulator. These two vectors are then used to complete the forward kinematics model of the manipulator. As the manipulator is actuated, the new forward kinematics model produces position estimates which can be compared to the measured "true" position at the same points. The error between the estimated and true position can then be used to analyze the performance of the developed kinematic determination system as compared with results found in the kinematic simulation. A brief summary of the experiment procedure is given below.

- Select manipulator
- Place IMU beyond the last actuated joint of the manipulator
- Select manipulator initial pose
- Input joint angle trajectory for each joint
- Measure acceleration and angular rate for each point on the joint angle trajectory

- Measure gravity for each point on the joint angle trajectory
- Subtract gravity from measured acceleration values for acceleration due to circular motion
- Use angular rate and acceleration to calculate $\hat{s}_i$ and $\vec{s}_i$ for each joint
- Implement forward kinematics calculation for a variety of poses covering the robot workspace
- Measure or calculate "true" position of the IMU
- Analyze difference in position calculations to quantify fidelity of the new model

6.1.1 Error Analysis

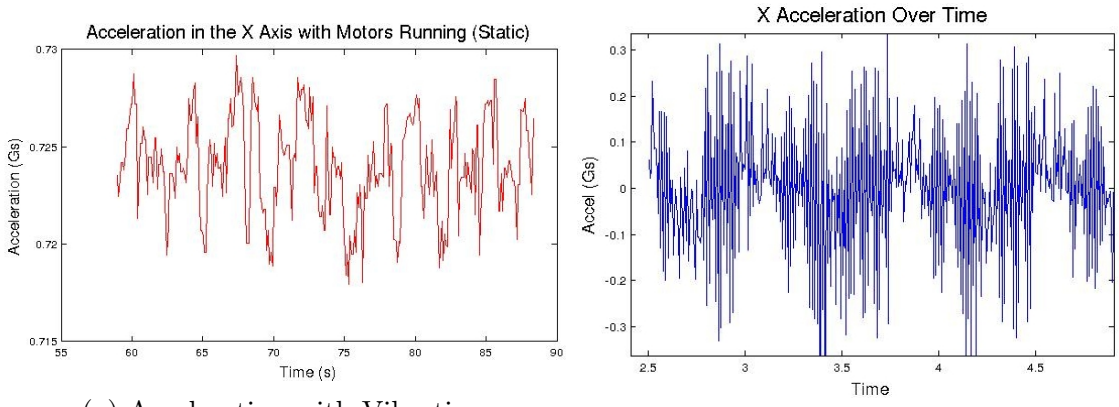
Based solely on the Matlab simulation of the forward kinematics, it is expected that the results of the hardware implementation will give a position estimate that is closely related to measurement error and how the measurement data is taken. Some of these variables can be modified by the user, until hardware limitations are exceeded, and some are built into the system, such as manufacturer IMU zero-mean noise and drift.

The minimum error in the system is based on the error values built into the selected IMU. A commercial mems IMU, the Memsense nIMU, was chosen for this experiment due to accuracy and availability of the unit. The minimum error indicated by the manufacturer can be seen in table 6.1. A control test of the specific IMU used for the experiment verified the manufacturer numbers and indicated an approximate $1.5°/s$ offset in the X axis and $0.5°/s$ offset in the Y axis of the gyroscope.

Gyro Offset	Gyro Noise	Accelerometer Offset	Accelerometer Noise
$\pm 1.5°$	$\pm 0.35 - 0.95°$	± 30 mG	$\pm 6 - 8$ mG

Table 6.1: IMU Manufacturer Error Tolerances

If the IMU produced the only error in the system, accelerations and angular rates of magnitude $\pm 24 - 32$ mG and $\pm 1.4 - 3.8°/s$ would be required to produce an accurate estimate of the manipulator kinematics. However, there is additional error produced by the vibration of the joint motors, increasing in magnitude as the speed of the joint increases. An example of small and large error caused by motor vibration is shown in figure 6.3



(a) Acceleration with Vibration

(b) X Axis in Motion

Figure 6.3: IMU Error with running motors

The joint with the most error will be joint 4, since it has the smallest moment arm. Based on a least-squares fit, shown in figure 6.4, the average difference between the true acceleration value and the measured acceleration value was approximately 1.8 $\frac{m}{s^2}$, using a good test run. For a system where the measured values should, at a maximum, be 2 $\frac{m}{s^2}$, based on the least-squares fit, this constitues a 90% error in the measured acceleration. Comparing with values from other joints, this is the worst case error that will be seen in the system.

As concluded from the simulation results, an increase in the speed of the joint angle trajectory may somewhat increase the magnitude of measured values compared to the magnitude of the error. This speed increase is limited by the physical characteristics of the manipulator, especially one in an earth gravity environment, such as motor maximum speed and current, loading on specific joints due to high

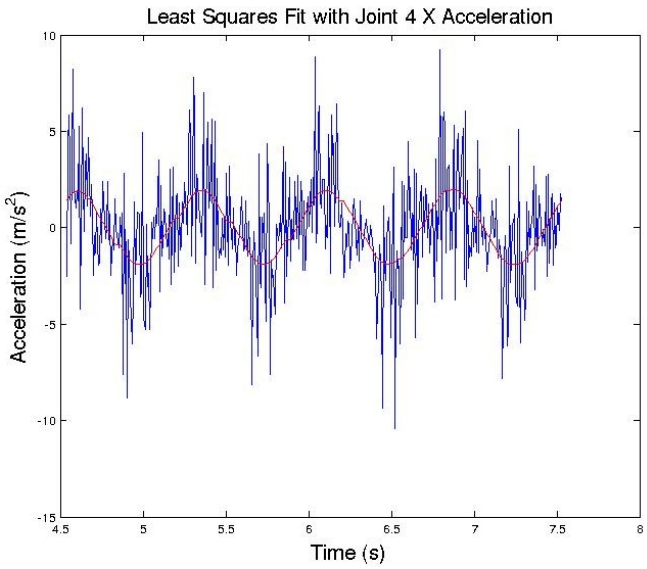


Figure 6.4: Error in Joint 4 measurement

inertia, and loading of specifc joints due to gravity. Repetitions of the joint angle trajectory for each joint will also reduce error by providing more points which can be averaged for a more repeatable, accurate measurement. Using these techniques, error in the IMU measurement was reduced by increasing the speed of the joint angle trajectory and increasing the number of arc repetitions using the values in table 6.2. The arc trajectory sweeps out a 10° angle. This trajectory angle, less than ideal, was chosen due to limitations in the hardware, based on both speed and sweep of the arc. 4-6 arc repeats were chosen based on reducing the error in the screw vectors, and are also somewhat less than ideal numbers due to hardware limitations. These numbers represent close to the physical limitations of the manipulator in a gravity environment.

Joint	Arc Trajectory Period (s)	Number of Repeats
Joint 2	1.0	6
Joint 3	1.0	6
Joint 4	0.368	4
Joint 5	0.368	4

Table 6.2: Joint Trajectory Period and Repeat Count

Based on the measurement error seen in the worst and best joint actuations, and using simulation data as a basis, without considering some of the extra filters added to the hardware data processing, $25 - 30\%$ error can be expected in the position estimation.

6.2 Calculation of "True" Position

In order to compare results from the new kinematic determination method with a "true" position of the IMU, the truth position must be calculated or measured. While direct measurement is ideal, the position can be calculated using accurately measured DH parameters for the manipulator in an equivalent forward kinematics scheme. This calculation is also more complex due to the relative placement of the DH frames with the static frame of the screw based method. For the comparison in this thesis, the position of the IMU was calculated using the DH method rather than measured directly.

In order to calculate the true IMU position in the perspective of the screw theory static frame, the DH parameters as measured by Ellsberry [17] were used in the forward kinematics, as given in table 6.3.

i	α_{i-1} (deg)	a_{i-1} (m)	d_i (m)	θ_i (deg)
1	0	0	0.2491	θ_1
2	90	0	0	θ_2
3	0	0.5589	0	θ_3
4	-90	0.1514	0.5388	θ_4
5	90	0	0	θ_5
6	90	0	0	θ_6
T	0	0	0.2666	0

Table 6.3: Denavit-Hartenberg parameters for Ranger Mark I. From [17].

As placed in the screw theory based method, the static frame in which position measurements are taken is located coincident with the IMU frame in the initial pose of the robot. The transformation from the inertial world frame to this frame is calculated by

$$ {}^{A}_{W}T_{z} = {}^{A}_{T}T^{T}_{6}T^{6}_{5}T^{5}_{4}T^{4}_{3}T^{3}_{2}T^{2}_{1}T^{1}_{0}T^{0}_{W}T \tag{6.1} $$

The transformation from the world frame [W] to the base frame [0] and from the tip frame [T] to the IMU frame [A] is a Euler-Angle rotation and translation. The total transformation matrix ${}^{A}_{W}T_{z}$ is unique to the initial pose of the robot. To calculate the new position of the IMU in the static reference frame, and thus calculate the same position as the screw theory method, the position of the origin of the IMU frame must be transformed into the world frame, and then into the static reference frame.

$$ p = {}^{A}_{W}T_{z}{}^{W}_{0}T^{0}_{1}T^{1}_{2}T^{2}_{3}T^{3}_{4}T^{4}_{5}T^{5}_{6}T^{6}_{T}T^{T}_{A}T^{A}p_{o,A} \tag{6.2} $$

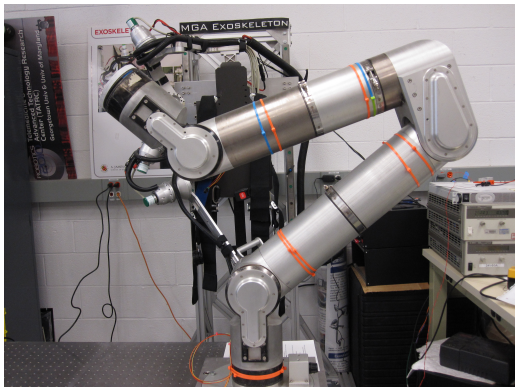
This calculation will produce a position measurement in the same frame as the equivalent position measurement in the screw theory based method.

There is also additional error added into the system by using the DH trans-

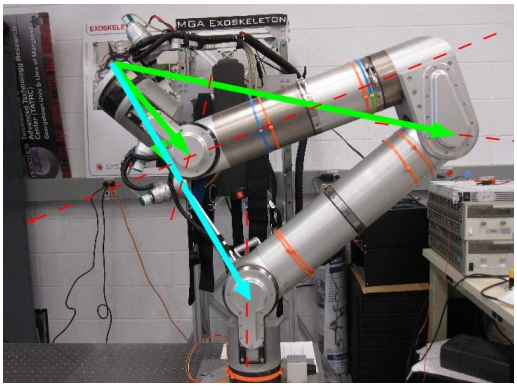
formation instead of direct measurement. Error in the measurement of the DH parameters contributes to error in the "true" position estimation. Also, the transformation between the IMU frame [A] and tip frame [T] cannot be measured exactly and, as such, is visually estimated. This will also inject position error on the order of 1-2 cm into the "true" position estimation.

6.3 Position Error Analysis

An initial pose was selected for the manipulator as shown in figure 6.5a and the exact joint angles, as represented from a pre-determined "home" position, are listed in table 6.4. A visual representation of the screw vectors is also shown in figure 6.5b.



(a) "zero" arm pose



(b) Arm pose with vectors

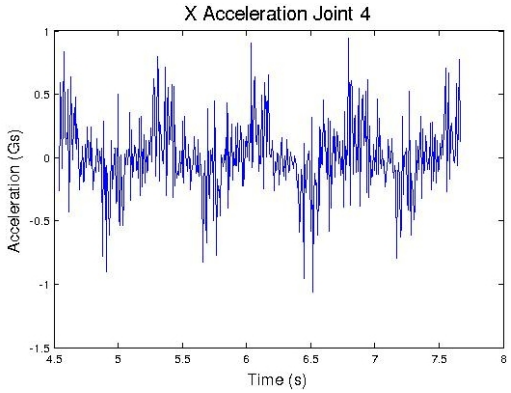
Figure 6.5: Manipulator Initial Pose with and without vectors

Joint Angles (Radians)						
Joint	1	2	3	4	5	6
Angle	0	0.63	1.2109	0	2.2	0

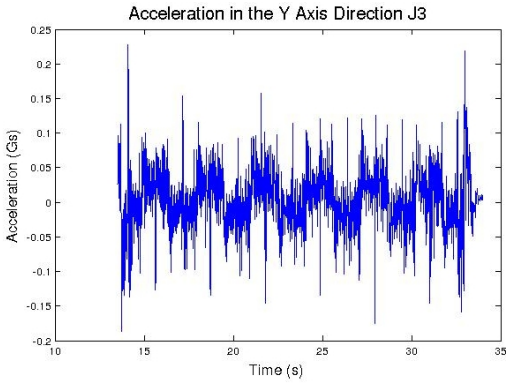
Table 6.4: Initial Arm Pose Joint Angles

IMU data for joints 2-4 was collected and processed for screw vector calculation. An example of the acceleration and angular rate is shown in figures 6.6 and

6.7. The full set of joint measurement data can be found Appendix A.

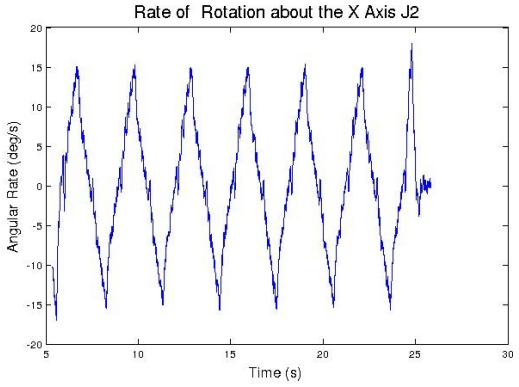


(a) X Acceleration Joint 4

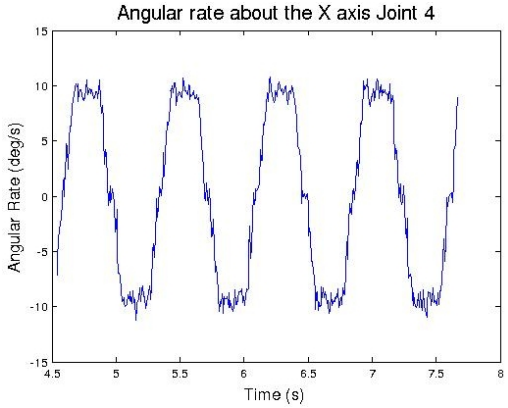


(b) Y Acceleration Joint 3

Figure 6.6: Joint Example Accelerations



(a) Angular Rate about X Joint 2

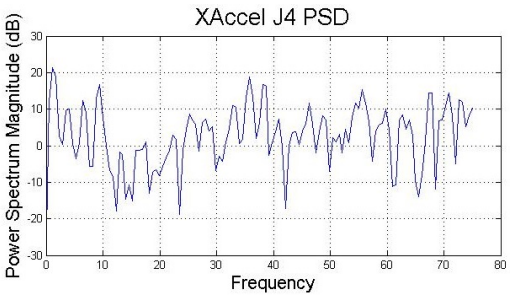


(b) Angular Rate about X Joint 4

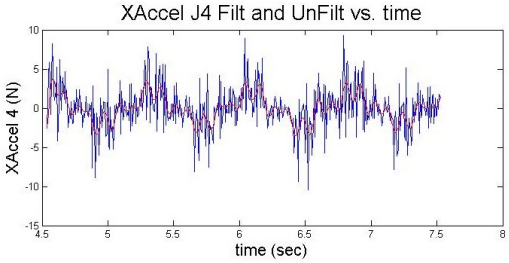
Figure 6.7: Joint Example Angular Rotation

Due to noisy data, a filter was required. Each filter was tuned based on the power spectral density (PSD) of each axis of each joint. The tuned filter was then applied to the data. An example of one utilized PSD plot and a comparison of unfiltered and filtered data is shown in 6.8.

As per the mathematical model and simulation described previously in secion 4.2, the screw axis of the joint and the vector to the screw axis was calculated for joints 2-4. Table 6.5 shows the calculated values.



(a) PSD Plot of Joint 4 X Axis



(b) Comparison of filtered and unfiltered data

Figure 6.8: Joint Example Angular Rotation

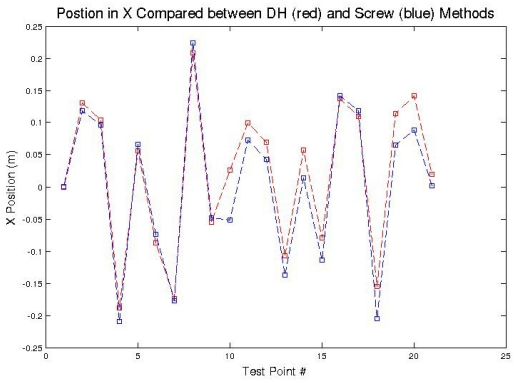
$\hat{s}_2$	$\hat{s}_3$	$\hat{s}_4$	$\hat{s}_5$	$\vec{s}_{0,2}$	$\vec{s}_{0,3}$	$\vec{s}_{0,4}$	$\vec{s}_{0,5}$
0.9709	0.9718	0.1964	0.9697	0.04555	-0.0887	0.0158	0.01944
-0.2390	-0.2358	0.8307	-0.2433	0.1546	-0.3851	0.1367	0.05255
-0.0154	-0.0068	-0.5209	-0.0213	0.4945	0.6781	0.2239	0.2843

Table 6.5: Calculated Screw Vectors for Ranger Mark I

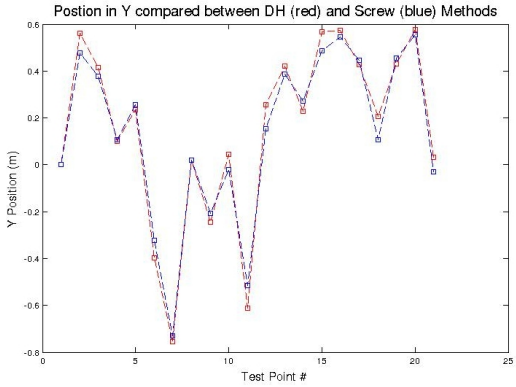
These vectors to the joint screw axis are visually accurate. The consistency of the screw axis of joints 2,3, and 5 is also a positive sign that the system has come up with a reasonable answer, as the screw axis for those three joints should be identical. Now, with these vectors, the position estimates of the DH kinematics can be compared to the screw theory kinematics position estimates as calculated by the values in 6.5.

21 positions representative of a wide variety of joint angle combinations were used for the initial position estimates. Each position is calculated as though the manipulator were traveling from the initial pose to the new position. The first eight test points represent positive and negative single joint actuations. Figures 6.9a, 6.9b, and 6.10a show the difference in position on the three axis of the static reference frame. Figure 6.10b shows the magnitude root mean square error of the position.

In general, the position as determined by the kinematic model presented in

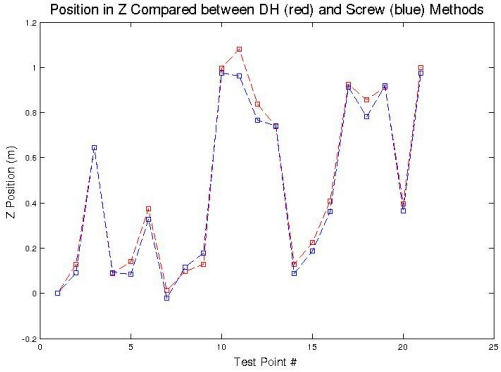


(a) X Position Error

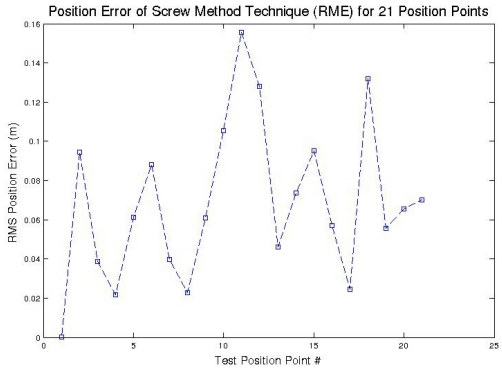


(b) Y Position Error

Figure 6.9: X and Y Position Plotted for 21 Poses



(a) Z Position Error



(b) Root Mean Error of Position

Figure 6.10: X Position and Position Root Mean Error for 21 Poses

this thesis follows closely with the position as calculated by measured values. The average percentage error in position estimation is 17%. Given the high measurement error in the system, this is a better result than expected based on the simulation data. Error in the data points is caused by error in position contributed by each joint. A summary of how much joint contributes over a set of angles is shown in figure 6.11.

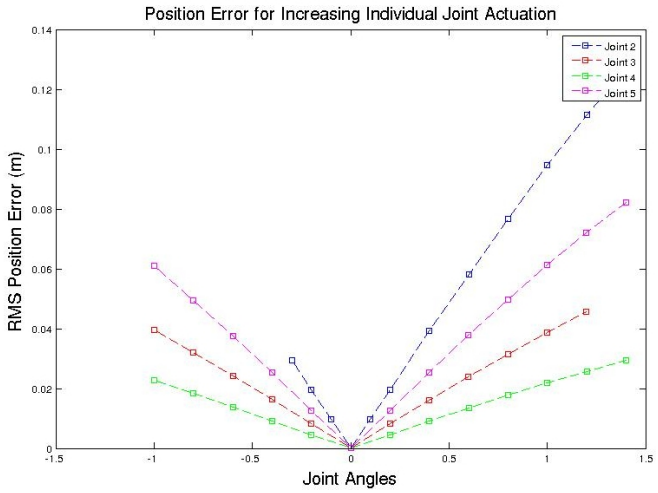
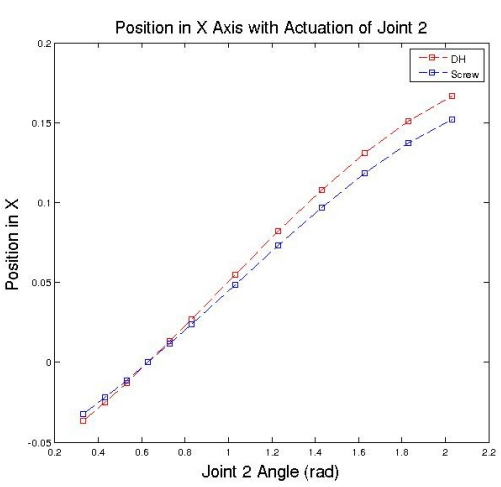


Figure 6.11: Position Error Contributed by each joint

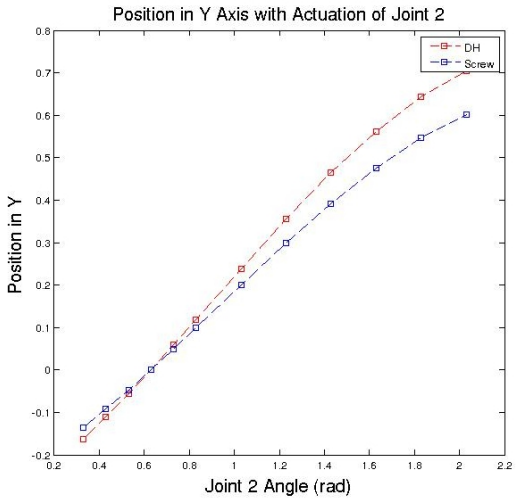
Based on the results in figure 6.11, joints 2 and 5 contribute the most position error to the system. The contribution of joint 2 to error is expected, as that joint was actuated slowly and had a smaller moment arm than joint 3. The second highest error contributor, joint 5, is somewhat unexpected as it has a larger moment arm than joint 4 and was actuated at the same speed. This could be due to bad runs or other unexpected contributing factors.

As an example, the actual position versus the estimated position for a range of angles of joint 2 is shown in figure 6.12.

As expected from the simulation results, larger joint angles will contribute more to the position error. Over the range of results from the hardware versus the simulation, the hardware performed better than expected given high measurement



(a) Estimated and Actual X position



(b) Estimated and Actual Y position

Figure 6.12: X and Y Exact and Estimated Position Over Joint 2 Angles

error. Some of the trends seen in the simulation, such as error and joint angle, were also verified with the hardware implementation.

6.3.1 Error Correction

There are several corrections that can be made to this type of hardware system which will improve the accuracy of the guessed manipulator tip positions. A more accurate system for measurement of actual position will be discussed, along with suggestions for improvements in the IMU and manipulator selection. Location of the manipulator, either in gravity or microgravity, also makes a difference in the accuracy of the forward kinematics.

As mentioned in section 6.2, the final transformation in the DH calculation from frame T to frame A, or from tip to IMU, was innacurate due to errors in measurement. The final values used were measured and approximated as best as possible from visual inspection and crude measurement tools. This adds approximately 1-2 cm of error into the "true" position calculation. Ideally, a tool such as a Faro arm can measure this transformation, and in fact the position vector, very

accurately to sub-mm levels. In order to perform a true comparison, a high accuracy measurement of the actual position vector should be taken and compared to the guessed position vector. In order to use the DH comparison, an accurate final transformation should be measured.

A change in manipulator and IMU selection will also improve the accuracy of the kinematic estimator. In comparison to IMUs currently being used in space applications, the IMU used for this experiment was of relatively low accuracy and high error. A higher end IMU will give higher precision measurements and less zero-mean error inherent to the device. An IMU with a higher tolerance for vibration would also be beneficial. It is also worth noting that the Ranger Mark I arm has been in heavy use for the past 15+ years. Because the original mechanical systems are still in place on the arm, extra vibration and error is likely visible compared to a newer system. In addition, the Mark I was not built to take high gravity loads and high loads at top speed, unlike some of the other versions of the Ranger system. A manipulator built to take high loading in gravity would be ideal due to the ability to move all joints at higher speed without risking damage to the system. Another way of mitigating gravity load is to test the manipulator in a microgravity environment, as intended for the purposes of this thesis. The manipulator will not have gravity loads and will be able to actuate joints with more speed at less cost to the arm, which will again improve accuracy in the kinematic estimator.

7.1 Summary and Conclusions

The research detailed in this thesis developed a new method of kinematic determination for unmodeled manipulators based around the use of screw theory and a single IMU. This method can be used with any serial revolute manipulator and requires only knowledge of encoder counts per revolution for each joint. The mathematics of the method were presented, with and without error correction, as a low-complexity, mathematically convenient way of calculating a full 6 DOF position and orientation for a manipulator. A simulation of the method was analyzed, looking at the relationship of measurement error and input variables to position error to assess the feasibility of use of the method. The results of the simulation were then verified with a real-world manipulator and a comparison of the measurement error to position error discussed.

An analysis was completed of previous works using different methods to determine unmodeled manipulator kinematics. Most previous work is based on prior knowledge of the manipulator or other external sensor information. The majority of these methods use DH convention. It was concluded, based on previous methods, that the use of DH parameters adds unnecessary complexity to the kinematic determination. Screw theory is a simpler method for this application and used in the current research. Accelerometers also provide an advantage in being a small, simple system that can be used with any manipulator, not just one where the joints and links are known. As such, a method using a single IMU and screw theory was developed for kinematic estimation of a micro-gravity manipulator, based on successes and limitations of previous research.

A simulation was developed in order to test the performace and fidelity of the kinematic estimation method under different conditions. A relationship between the variables of the method and error of the system was determined in order to characterize how changes in one part of the calculation effect the outcome. In general, a measurement error of less than 30% in both acceleration and angular rate is needed in order to calculate the position of the end effector with less than 10% error. Other variables, such as arc sweep and period, number of arc repeats, and number of DOF of the manipulator can improve or worsen this ratio. The simulation determines that the kinematic estimator is capable to a degree of accuracy directly related to measurement error in the system.

A hardware implementation was also developed to test the kinematic estimator and verified the trends seen in the simulation with measurement error and position error. Screw vectors and vectors to the screw axis were determined on the manipulator and those values were used to calculate the forward kinematics. A comparison of true and estimated positions was used to evaluate the performance of the estimator for a real-world system. Given a $40 - 90\%$ measurement error in each joint, a position error of $15 - 25\%$ was calculated, with magnitude depending heavily on the size of the joint angles. Based on the simulation results, this ratio falls within expected values and is actually slightly better than expected for this manipulator in its current configuration.

7.2 Future Work

As detailed in section 6.3.1, there are several improvements possible to improve the results of a real-world use of the kinematic estimator. Each manipulator will have error inherent in the design and hardware and different manipulators may be more or less accurate. A higher accuracy IMU will also help the calculation. One of

the most important changes is the use of a highly accurate system to measure the position vector as the arm moves, instead of the less accurate measurement used in this thesis due to hardware availability. Measurement in micro-gravity wil also help remove error by improving the capabilities of the arm and removing the need to subtract out gravity, and add other measurement error, from the calculation. Different tests implementing some or all of these methods will help additionally in evaluating the performance of the kinematic estimator for future research.

An analysis of the reaction of an IMU to different levels and frequencies of vibration would also be beneficial. It may be that there are vibration modes in the arm motors which excite vibration modes in the IMU which cause additional measurement error. This leads very naturally into a dynamics analysis for the manipulator. An understanding of the dynamic modes of the manipulator, as most likely measured by a very similar method to the kinematic estimator, will help to quantify error in measurements and the performance of the manipulator itself. The combination of a kinematic estimation and dynamic estimation would be a much more complete picture of the manipulator characteristics for refined control.

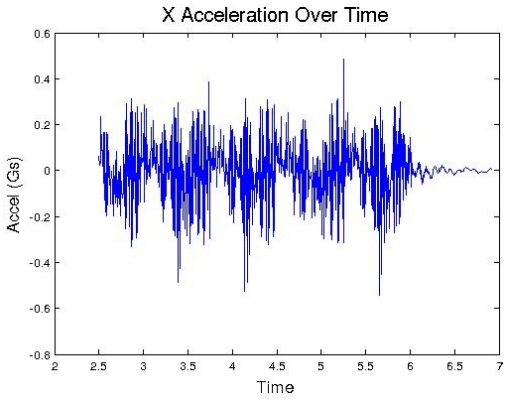
There are several manipulator types not addressed that would be interesting to develop using the identified kinematic estimator. Parallel manipulators and prismatic link manipulators could potentially benefit from or have simplified calculation of forward kinematics. The ability to determine kinematics for a generic manipulator, without restrictions, would be extremely advantageous. Since screw theory is applied to parallel manipulator forward and inverse kinematics, because of how much it simplifies the calculation, use of this kinematic estimator may fit in well with those already developed methods.

7.2.1 Future Applications

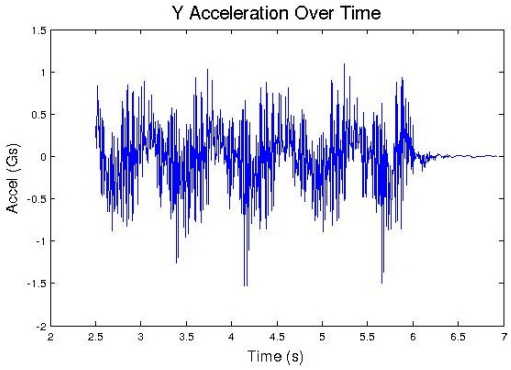
With the use of the developed kinematic estimator, the way in which reconfigurable manipulators can be designed is expanded significantly. No geometric knowledge of joints or connectors is required, allowing the use of any shapes, connectors, links, or other designs as benefits the manipulator task. A system can be developed by which any manipulator can determine its own kinematics, in whatever configuration is desirable for the system. The potential for a system like this, especially if manipulators are modified or changed out on a regular basis, where no software updates or changes are required, and no complex calculations are needed, is very interesting.

Another interesting application is the used of this method in underwater robotics. Currently, there are very few reconfigurable underwater manipulators. Using the kinematic estimator developed in this research, no electrical or other connections are needed between joints, allowing a completely mechanical interface. A mechanical interface with no other connections is far easier to change out underwater and could allow the development of a useful underwater reconfigurable manipulator.

Chapter A:



(a) X Acceleration J5



(b) Y Acceleration J5

Figure A.1: Joint 5 XY Accelerations

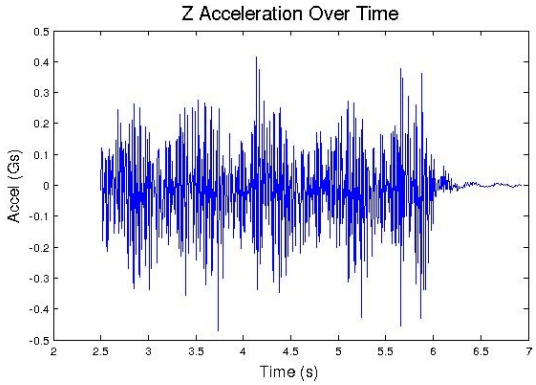
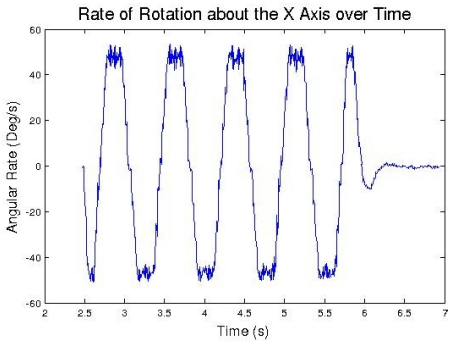


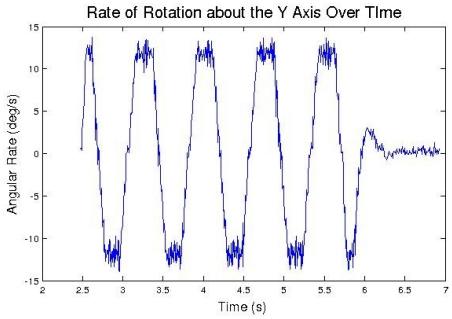
Figure A.2: Z Acceleration J5

The arcs in the angular rate data are much clearer than those of acceleration, as seen in figures A.3a, A.3b, and A.4. Angular rate graphs also show a "glide" period of constant angular rate. This is typical of a trapazoidal trajectory as controlled by the joint motor.

The majority of the rotation occurs about the X and Y axis. The same procedure was performed for joint 4, at identical arc period and sweep angle.



(a) Angular Rate About X J5



(b) Angular Rate About Y J5

Figure A.3: Joint 5 XY Angular Rates

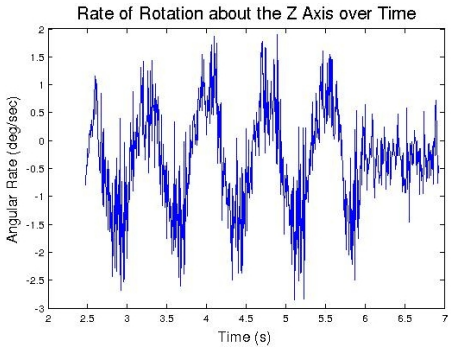
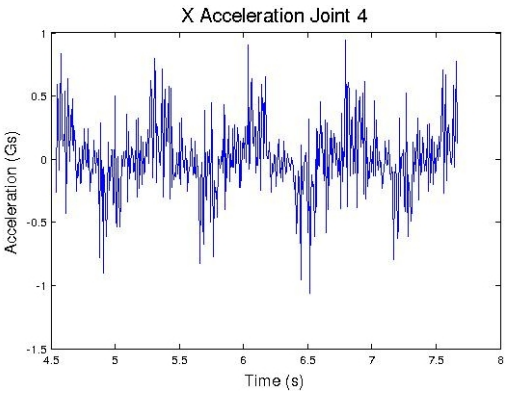
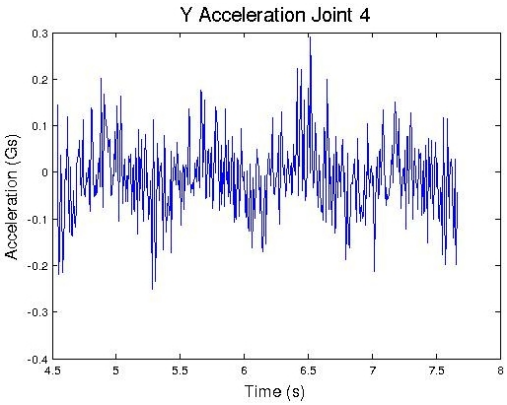


Figure A.4: Angular Rate About Z J5

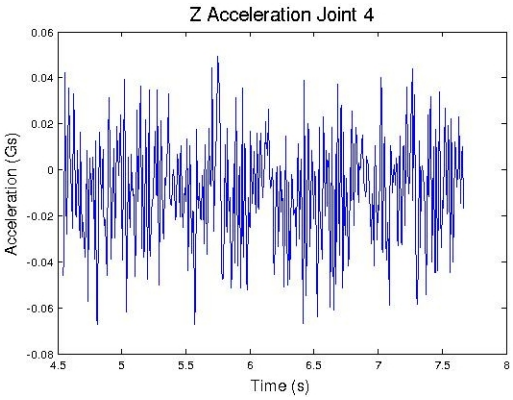


(a) X Acceleration J4

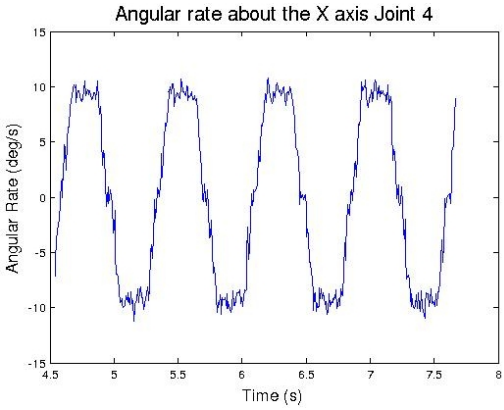


(b) Y Acceleration J4

Figure A.5: Joint 4 XY Accelerations

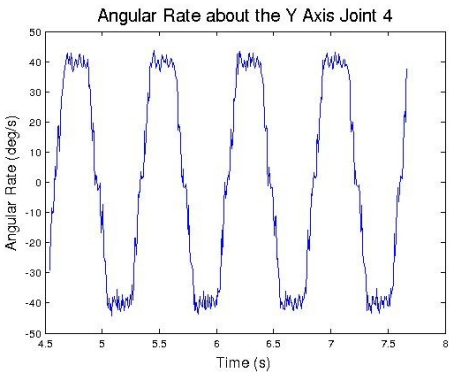


(a) Z Acceleration J4

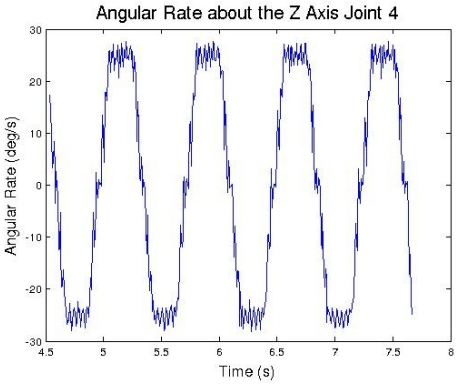


(b) Angular Rate about X J4

Figure A.6: Joint 4 Z Accel and X Rotation



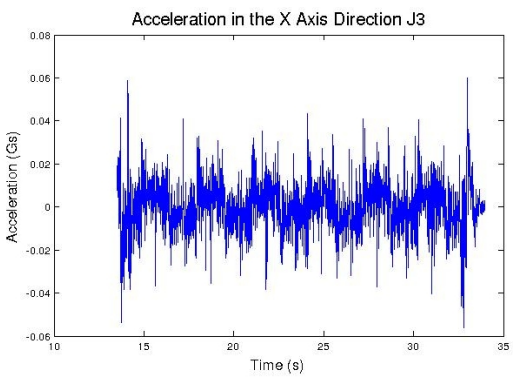
(a) Angular Rate About Y J4



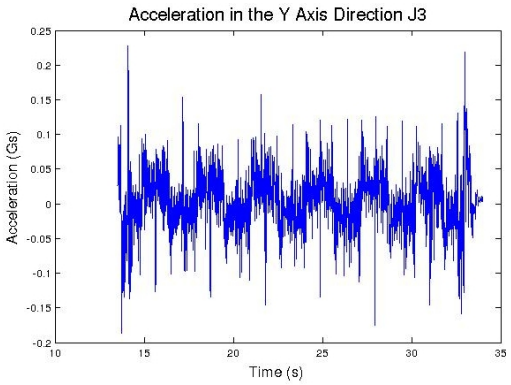
(b) Angular Rate About Z J4

Figure A.7: Joint 4 YZ Angular Rates

The graphs of joints 5 and 4 are very similar. Joint 4 is a noisier data set, seen in A.1a, A.1b, and A.2, due to the fact that the IMU was closer to the joint 4 axis of rotation. Joints 2 and 3 behaved somewhat differently as they were further from the measurement point and had increased arc period. The extra arc repeats as described in 6.1.1can be seen in the graphs of the data for joint 3.

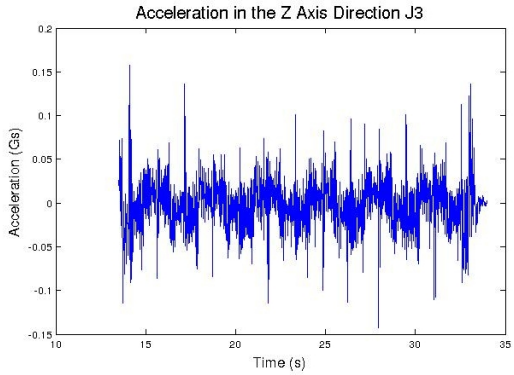


(a) X Acceleration J3

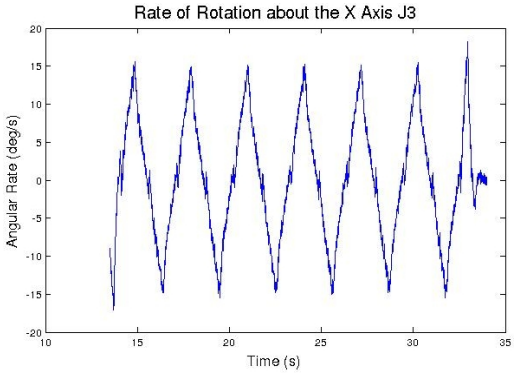


(b) Y Acceleration J3

Figure A.8: Joint 3 XY Accelerations



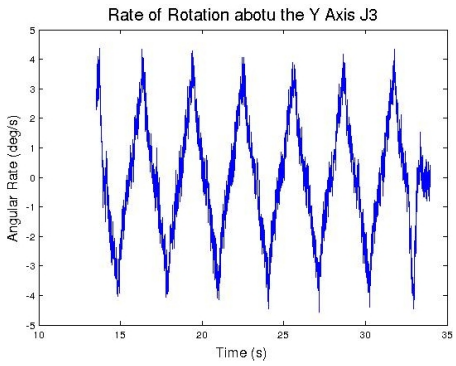
(a) Z Acceleration J3



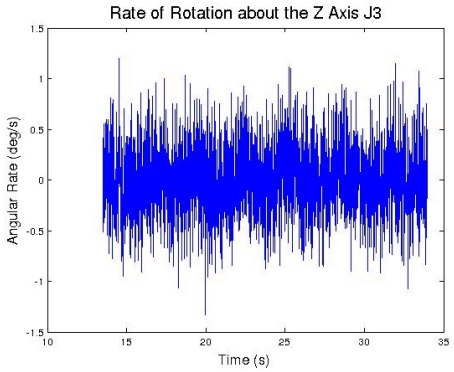
(b) Angular Rate about X J3

Figure A.9: Joint 3 Z Accel and X Rotation

There is no period of constant angular velocity in the recorded measurements for joint 3. Angular acceleration and angular velocity are present throughout the arc time, in theory leading to a stronger total acceleration reading. There is, however, still significant noise. The IMU measurements of joint 2 are similar to those of joint 3, as seen in figures A.11a through A.13b.

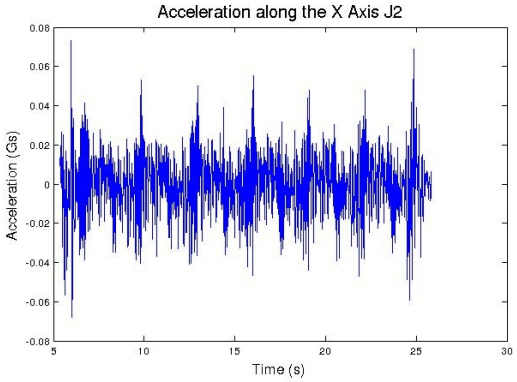


(a) Angular Rate About Y J3

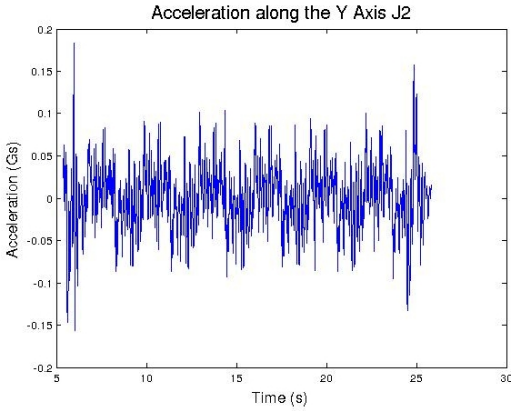


(b) Angular Rate About Z J3

Figure A.10: Joint 4 YZ Angular Rates

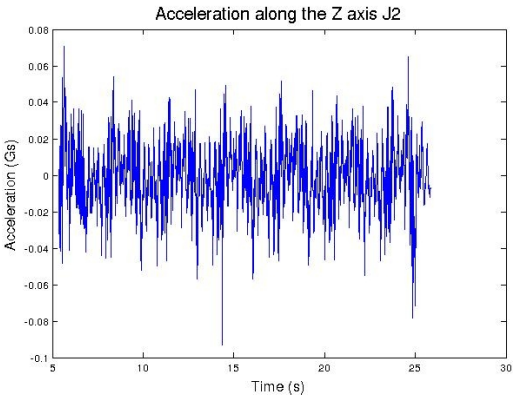


(a) X Acceleration J2

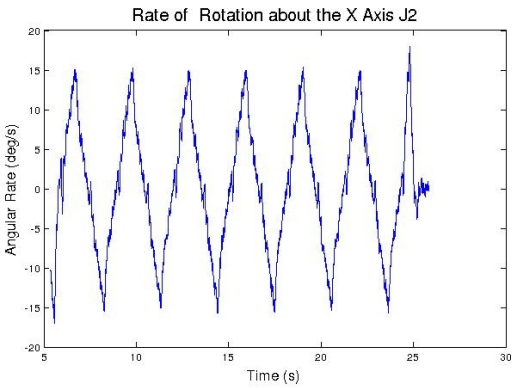


(b) Y Acceleration J2

Figure A.11: Joint 2 XY Accelerations

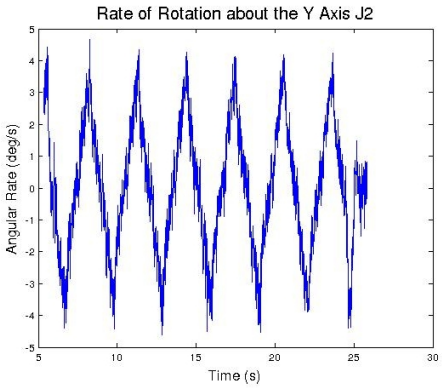


(a) Z Acceleration J2

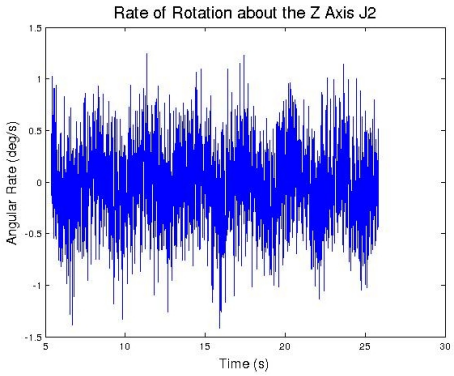


(b) Angular Rate about X J2

Figure A.12: Joint 2 Z Accel and X Rotation



(a) Angular Rate About Y J2



(b) Angular Rate About Z J2

Figure A.13: Joint 2 YZ Angular Rates

Chapter B:

While not necesary for kinematic calculation by the method presented in this thesis, it may be desirable for the user to have a visual representation of the manipulator. Specifically, the pose of the arm, or at least an approximation, may be beneficial. Given the screw axis and vectors to the screw axis, it is possible to calculate a visual representation of the manipulator, not including joint offsets. Given a point, the origin of a frame, a line that represents the screw axis, and a perpendicular line between the point and screw axis, a second vector to the screw axis that matches the physical representation of the manipulator can be determined. This concept is illustrated in B.1.

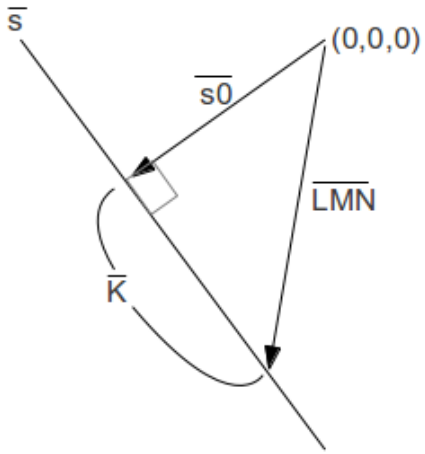


Figure B.1: Calculation of Pose via Point to Line

The LMN vector $L\vec{M}N$ represents the physical link between two joints. K represents some distance along the screw axis between the intersection of the perpendicular vector to the screw axis and the physical vector. Traveling K along the screw axis results in a vector $K\hat{s}$. Figure B.1 shows that the addition of the vectors

$K\hat{s}$ and $\vec{s_0}$ results in the vector $L\vec{M}N$, or rearranged:

$$K\hat{s} = \vec{s_0} - L\vec{M}N \tag{B.1}$$

which is also applicable in 3D space. Considering the point at which the physical vector $L\vec{M}N$ intersects the screw axis, this points is labeled (L,M,N) in cartesian space. Thus, equation B.1 can be divided into three seperate equations, each representing an axis of the cartesian reference frame.

$$s_{0,x} + K\hat{s}_x = L \tag{B.2}$$

$$s_{0,y} + K\hat{s}_y = M \tag{B.3}$$

$$s_{0,Z} + K\hat{s}_Z = N \tag{B.4}$$

There are now three equations and four unknowns. Figure B.1 also shows that lines $\hat{s}$ and $K\hat{s}$ will be parallel, which means their dot product will be 0. This can be used to produce the fourth equation. Equation B.1 can be rewritten as a dot product with $\hat{s}$

$$0 = (\vec{s_0} - L\vec{M}N) \cdot \hat{s} \tag{B.5}$$

Using the algeabraic definition of a dot product, B.5 can now be written as

$$0 = (s_{0,x} - L)s_x + (s_{0,y} - M)s_y + (s_{0,z} - N)s_z \tag{B.6}$$

Simplifying the expanded form of B.6 by using A and B as coefficents of the form

$$A = s_{0,i-1,x}s_x + s_{0,i-1,y}s_y + s_{0,i-1,z}s_z \tag{B.7}$$

$$B = s_{0,i,x}s_x + s_{0,i,y}s_y + s_{0,i,z}s_z \tag{B.8}$$

and solving for K by substituing B.2, B.3, and B.4 into B.6

$$K = \frac{A - B}{({s_x}^2 + {s_y}^2 + {s_z}^2)} \tag{B.9}$$

K is substituted back into B.2, B.3, and B.4, and the $L\vec{M}N$ vector is solvable. The endpoint of the $L\vec{M}N$ vector is used as the starting point for the next joint. Thus, by graphing the LMN vectors in 3D space, a rough picture of the pose of the manipulator can be generated in addition to the kinematic solution.

Chapter C:

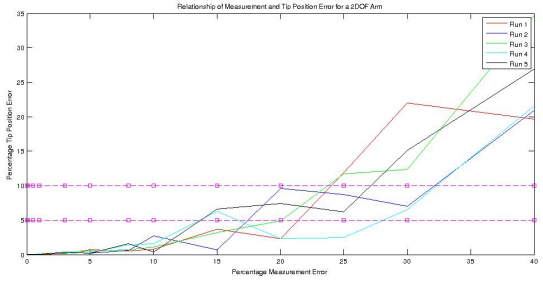


Figure C.1: Influence of measurement error on tip position for 2 DOF

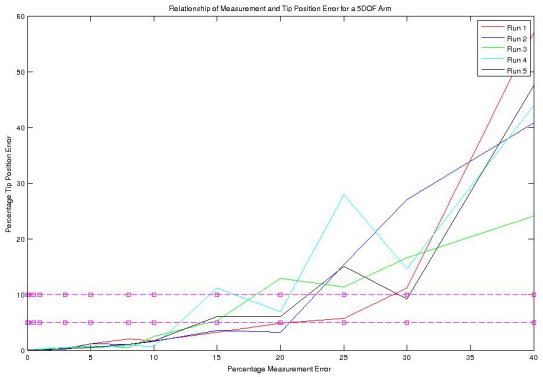
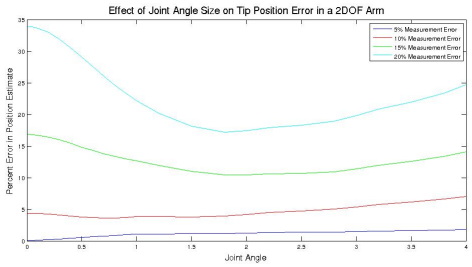
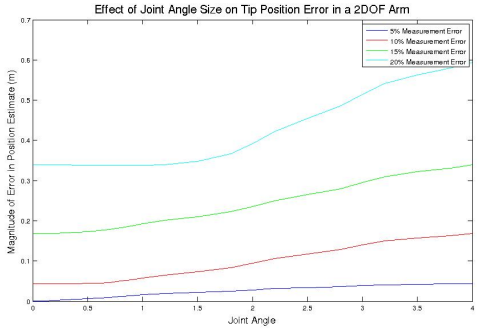


Figure C.2: Influence of measurement error on tip position for 5 DOF

Chapter D:

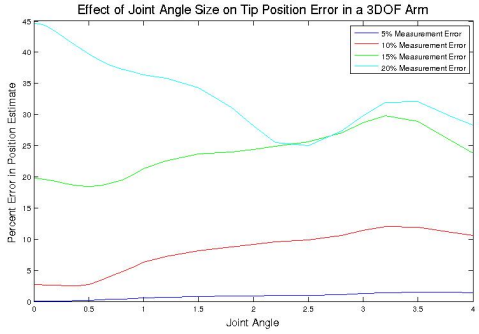


(a) Percentage Position Error with Angle Size

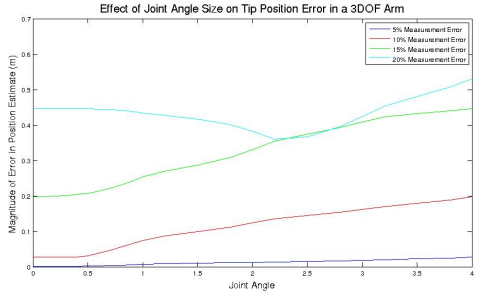


(b) Magnitude Position Error with Angle Size

Figure D.1: Effect of Actuated Angle Size on Tip Position Error 2DOF

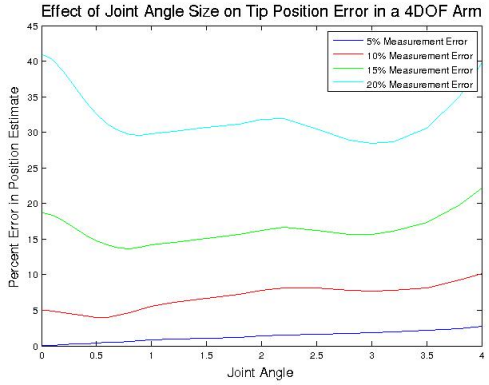


(a) Percentage Position Error with Angle Size

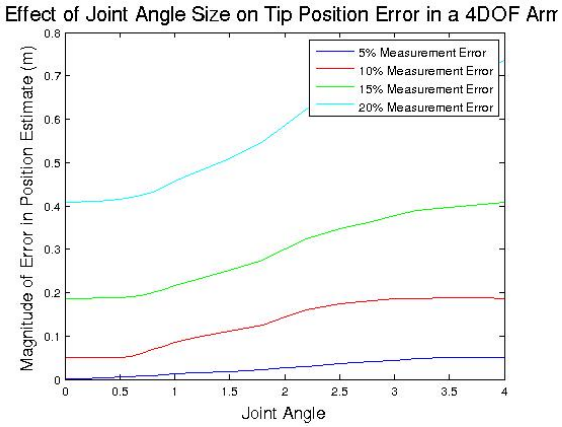


(b) Magnitude Position Error with Angle Size

Figure D.2: Effect of Actuated Angle Size on Tip Position Error 3DOF

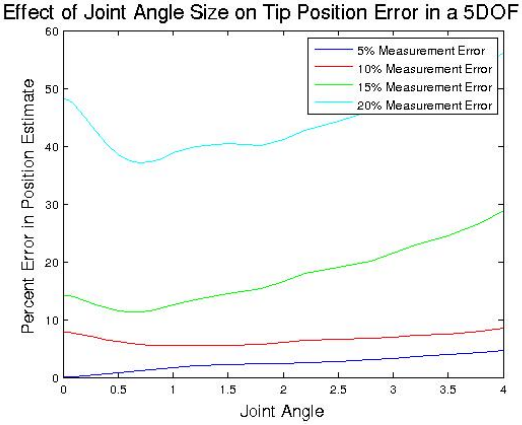


(a) Percentage Position Error with Angle Size

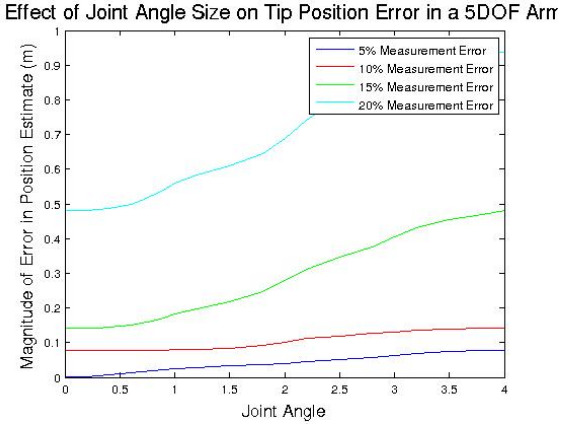


(b) Magnitude Position Error with Angle Size

Figure D.3: Effect of Actuated Angle Size on Tip Position Error 4DOF



(a) Percentage Position Error with Angle Size



(b) Magnitude Position Error with Angle Size

Figure D.4: Effect of Actuated Angle Size on Tip Position Error 5DOF

Chapter E:

The following code is an example, using variable measurement error, of the kinematic determination simulation.

```
%This is the arc calculation
%Set the link lengths (the r magnitudes)
r = [.2 .3 .1 .3 .3];
%Generate a pose
deg = [30 0 0 0 0];
radians = [deg(1)*pi/180,deg(2)*pi/180, deg(3)*pi/180, deg(4)*pi/180,
    deg(5)*pi/180];
%Prismatic joints
t = 0;
%Number of joints
N = 5;
%Set the s hat vectors (from the final reference frame perspective)(in the
%"flat" configuration
s = [0;1;0];
s(:,2) = [1;0;0];
s(:,3) = [0;0;1];
s(:,4) = [0;1;0];
s(:,5) = [0;0;1];
%Set the s0 vectors (from the final reference frame in the "flat"
%configuration)
s0 = [1*(r(1)+r(2)+r(3)+r(4)+r(5));0;0];
s0(:,2) = [0;0;0];
s0(:,3) = [1*(r(3)+r(4)+r(5));0;0];
```

```
s0(:,4) = [1*(r(4)+r(5));0;0];

s0(:,5) = [1*(r(5));0;0];

%Set the orientation of the final frame and q relative to the final

%reference frame-and in the flat config

q0 = [0;0;0;1];

%For the arc calcs

%Set the angle we want to sweep through

B = 0.6981317;

%Set the period of the sin function of theta

T = 5;

%Grab a time step-does not have to be this but its convenient for

%simulation purposes

h = 0.01;

points = T/h;

w = zeros(3,N);

Asave = eye(4);

ROT = eye(3);

g = 1;

k = 1;

Accel_cell = cell(1,N);

w_cell = cell(1,N);

%g is incremented every loop no matter what

%k begins to increment on the 3rd loop-or when accelerations can first be

%calculated

%Generation of values: Simulated "measurements"

%Create the new array of A and R

figure(15)

for i=1:N

%For each joint go through the sweep
```

```
for t0=(-h*9):h:T
%Generate the pattern
theta_patt = B*sin((2*pi*t0)/T);
plot(t0,theta_patt);
hold on;
w_patt = ((2*pi*B)/T)*cos((2*pi*t0)/T);
alpha_patt = -1*((4*pi^2*B)/(T^2))*sin((2*pi*t0)/T);
theta = theta_patt;
%Calculate the screw matrix for the individual joint-for subsequent
%matrices, use this matrix cumulatively with the previous 0
%matrices
8/3/2013 4:08 PM
ArcCalcErrVariable2
2 of 16
file:///C:/Users/connie/Desktop/octave/html/ArcCalcErrVariable2.html
a11
a12
a13
a21
a22
a23
a31
a32
a33
a14
a24
a34
a41
a42
```

```
a43
a44
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
(s(1,i)^2-1)*(1-cos(theta))+1;
s(1,i)*s(2,i)*(1-cos(theta))-s(3,i)*sin(theta);
s(1,i)*s(3,i)*(1-cos(theta))+s(2,i)*sin(theta);
s(2,i)*s(1,i)*(1-cos(theta))+s(3,i)*sin(theta);
(s(2,i)^2-1)*(1-cos(theta))+1;
s(2,i)*s(3,i)*(1-cos(theta))-s(1,i)*sin(theta);
s(3,i)*s(1,i)*(1-cos(theta))-s(2,i)*sin(theta);
s(3,i)*s(2,i)*(1-cos(theta))+s(1,i)*sin(theta);
(s(3,i)^2-1)*(1-cos(theta))+1;
t*s(1,i)-s0(1,i)*(a11-1)-s0(2,i)*a12-s0(3,i)*a13;
t*s(2,i)-s0(1,i)*a21-s0(2,i)*(a22-1)-s0(3,i)*a23;
```

```
t*s(3,i)-s0(1,i)*a31-s0(2,i)*a32-s0(3,i)*(a33-1);
0;
0;
0;
1;
%Asave is the A matrix of the previous loop recorded when theta=0
Aold = Asave;
A = [a11 a12 a13 a14; a21 a22 a23 a24; a31 a32 a33 a34; a41 a42 a43 a44];
%Calculate the rotation matrix so that we can rotate the
%accelerations into the IMU frame. We are calculating the
%accelerations in the reference frame so they must be rotated
ROT11 = s(1,i)*s(1,i)*(1-cos(theta))+cos(theta);
ROT12 = s(1,i)*s(2,i)*(1-cos(theta))-s(3,i)*sin(theta);
ROT13 = s(1,i)*s(3,i)*(1-cos(theta))+s(2,i)*sin(theta);
ROT21 = s(1,i)*s(2,i)*(1-cos(theta))+s(3,i)*sin(theta);
ROT22 = s(2,i)*s(2,i)*(1-cos(theta))+cos(theta);
ROT23 = s(2,i)*s(3,i)*(1-cos(theta))-s(1,i)*sin(theta);
ROT31 = s(1,i)*s(3,i)*(1-cos(theta))-s(2,i)*sin(theta);
ROT32 = s(2,i)*s(3,i)*(1-cos(theta))+s(1,i)*sin(theta);
ROT33 = s(3,i)*s(3,i)*(1-cos(theta))+cos(theta);
ROTsave = ROT;
ROTprev = ROT;
ROT = [ROT11 ROT12 ROT13;ROT21 ROT22 ROT23;ROT31 ROT32 ROT33];
%Multiply the current screw matrix by the previous one
A = A*Aold;
%Use the succesive screw displacements to calculate the new
%position vector which is 0 in the original reference frame
qnew = A*q0;
%Keep track of the analytical values
```

```
theta_deg(g) = theta_patt;

alpha_patt2(g) = alpha_patt;

w_patt2(g) = w_patt;

if(g == 2)

%If time is 0, save the current A

%Asave = A;

%Create an array to calculate the 0 acceleration and the

%accelerations in general

%qstore(:,2) = qnew;

tstore(:,2) = qnew;

%Calculate teh angular velocity in the IMU frame-w refers to

%the array of 0 only, w_e refers to the larger array of all

%values

%w(:,i) = s(:,i)*w_patt;

w_e(:,g) = s(:,i)*w_patt;

%Make an array of the position vectors-just for fun I guess

pos(:,g) = qnew;

g = g+1;

elseif(g == 3)

%qstore(:,3) = qnew;

tstore(:,3) = qnew;

%Calculate w vector in the IMU frame

w_e(:,g) = s(:,i)*w_patt;

%Double differentiate to get the 0 acceleration. Since we need

%both the before and after values, we need to calculate the

%zero values 1 loop after 0

%General case array-this array begins when k = 1;

accel_e(:,k) = (tstore(:,3)-2*tstore(:,2)+tstore(:,1))/(h^2);

accel_e(1:3,k) = transpose(ROTsave)*accel_e(1:3,k);
```

```
pos(:,g) = qnew;

g = g+1;

k = k+1;

elseif(g == 1 )

%Same thing as zero, but just store these values, they will be

%used in the first double differentation

%qstore(:,1) = qnew;

tstore(:,1) = qnew;

w_e(:,g) = s(:,i)*w_patt;

pos(:,g) = qnew;

g = g+1;

elseif(g == 10)

%If time is 0, save the current A

Asave = A;

%Create an array to calculate the 0 acceleration and the

%accelerations in general

qstore(:,2) = qnew;

tstore(:,1) = tstore(:,2);

tstore(:,2) = tstore(:,3);

tstore(:,3) = qnew;

accel_e(:,k) = (tstore(:,3)-2*tstore(:,2)+tstore(:,1))/(h^2);

%Rotate to IMU frame

accel_e(1:3,k) = transpose(ROTsave)*accel_e(1:3,k);

%Calculate teh angular velocity in the IMU frame-w refers to

%the array of 0 only, w_e refers to the larger array of all

%values

w(:,i) = s(:,i)*w_patt;

w_e(:,g) = s(:,i)*w_patt;

%Make an array of the position vectors-just for fun I guess
```

```
pos(:,g) = qnew;
g = g+1;
k = k+1;
elseif(g == 11)
qstore(:,3)
tstore(:,1)
tstore(:,2)
tstore(:,3)
=
=
=
=
qnew;
tstore(:,2);
tstore(:,3);
qnew;
%Calculate w vector in the IMU frame
w_e(:,g) = s(:,i)*w_patt;
%Double differentiate to get the 0 acceleration. Since we need
%both the before and after values, we need to calculate the
%zero values 1 loop after 0
accel(:,i) = (qstore(:,3)-2*qstore(:,2)+qstore(:,1))/(h^2);
%Rotate the value back into the IMU frame
accel(1:3,i) = transpose(ROTsave)*accel(1:3,i);
%General case array-this array begins when k = 1;
accel_e(:,k) = (tstore(:,3)-2*tstore(:,2)+tstore(:,1))/(h^2);
accel_e(1:3,k) = transpose(ROTsave)*accel_e(1:3,k);
pos(:,g) = qnew;
g = g+1;
```

```
k = k+1;
elseif(g == 9 )
%Same thing as zero, but just store these values, they will be
%used in the first double differentation
qstore(:,1) = qnew;
tstore(:,1) = tstore(:,2);
tstore(:,2) = tstore(:,3);
tstore(:,3) = qnew;
accel_e(:,k) = (tstore(:,3)-2*tstore(:,2)+tstore(:,1))/(h^2);
%Rotate to IMU frame
accel_e(1:3,k) = transpose(ROTsave)*accel_e(1:3,k);
w_e(:,g) = s(:,i)*w_patt;
pos(:,g) = qnew;
g = g+1;
k = k+1;
else
%Only for general case now-we already calculated our 0 value
tstore(:,1) = tstore(:,2);
tstore(:,2) = tstore(:,3);
tstore(:,3) = qnew;
%Calculate all the accelerations for the pattern based on
%numerical double differentiation
%Cuts off t = -h and the last valeu, t = 20
accel_e(:,k) = (tstore(:,3)-2*tstore(:,2)+tstore(:,1))/(h^2);
%Rotate to IMU frame
accel_e(1:3,k) = transpose(ROTsave)*accel_e(1:3,k);
w_e(:,g) = s(:,i)*w_patt;
pos(:,g) = qnew;
k = k +1;
```

```
g = g +1;

end

end

Accel_cell(1,i) = {accel_e};

w_cell(1,i) = {w_e};

g = 1;

k = 1;

end

hold off;

%Now for some error addition:

%I need 2000 points of random error for each

%Reduce the error-these are approximately 1% of the maximum value I see in

%the readings

Accel_save = Accel_cell;

w_save = w_cell;

maxa = max(max(cell2mat(Accel_save)));

maxg = max(max(cell2mat(w_save)));

a = [0.1 0.1];

c = [0.1 0.1];

b = [-0.1 -0.1];

d = [-0.1 -0.1];

a

b

c

d

=

=

=

=
```

```
a.*maxa;
b.*maxa;
c.*maxg;
d.*maxg;
for e=1:length(a)
permeaserrora(e) = a(e)/max(max(cell2mat(Accel_save)))*100;
permeaserrorg(e) = c(e)/max(max(cell2mat(w_save)))*100;
end
%Generate some random error for each joint
for n=1:(length(a))
for i=1:N
Randa_error(n,i) = {(a(n)-b(n))*rand(3,points+9)+b(n)};
Randg_error(n,i) = {(c(n)-d(n))*rand(3,points+9)+d(n)};
accel1 = cell2mat(Accel_cell(1,i));
error = cell2mat(Randa_error(n,i));
error(4,:) = 0;
for u=1:points
accel1(:,u) = accel1(:,u) + error(:,u);
end
Accel_cell(n,i)
= {accel1};
gyro1 = cell2mat(w_cell(1,i));
error = cell2mat(Randg_error(n,i));
for u=1:(points+2)
gyro1(:,u) = gyro1(:,u) + error(:,u);
end
w_cell(n,i) = {gyro1};
end
end
```

```
%Now, we think, things can actually be calculated analytically
%INPUTS:
%Acceleration array for the arc from accelerometer
%Angular velocity array fron the gyro
%
As a side note, the velocity array can be compared to the analytical
%
solution for a real robot to see where the differences are and possibly
%
use this to compensate for the derived angular acceleration as well
%Theta array as commanded to the motor
%Associated time array and steps
%OUTPUTS:
%Take the derivative of the gyro readings (compare to theoretical solution
%based on motor input pattern for a real arm)
%Calculate the s unit vector
%Grab the shat first-we can take an average for a real robot, so lets
%average here just to be safe
%So the error makes things really really sucky-try a central moving
   average
%filter to make the data better-8 points for the average to start
%Start with just the rotations to see if it works
ma_cell = cell(length(a),N);
maa_cell = cell(length(a),N);
LPFcell = cell(length(a),N);
for n=1:length(a)
for b=1:N
wvalues = cell2mat(w_cell(n,b));
avalues = cell2mat(Accel_cell(n,b));
```

```
for i=1:1:(points+2)
%Average of point 5 (i+4) (t = -0.25 s ) is the recorded first point of
    the moving
%average
ma(1,i) = (wvalues(1,i)+wvalues(1,i+1)+wvalues(1,i+2)+
wvalues(1,i+3)+wvalues(1,i+4)+wvalues(1,i+5)+wv
ma(2,i) = (wvalues(2,i)+wvalues(2,i+1)+wvalues(2,i+2)+
wvalues(2,i+3)+wvalues(2,i+4)+wvalues(2,i+5)+wv
ma(3,i) = (wvalues(3,i)+wvalues(3,i+1)+wvalues(3,i+2)+
wvalues(3,i+3)+wvalues(3,i+4)+wvalues(3,i+5)+wv
ma_cell(n,b) = {ma};
end
for i=1:1:points
%Average of point 5 (i+4) is the recorded first point of the moving
%(t = -0.2 s)
%average
%Also losing the last point (or point 409)
maa(1,i) = (avalues(1,i)+avalues(1,i+1)+avalues(1,i+2)+
avalues(1,i+3)+avalues(1,i+4)+avalues(1,i+5)+a
maa(2,i) = (avalues(2,i)+avalues(2,i+1)+avalues(2,i+2)+
avalues(2,i+3)+avalues(2,i+4)+avalues(2,i+5)+a
maa(3,i) = (avalues(3,i)+avalues(3,i+1)+avalues(3,i+2)+
avalues(3,i+3)+avalues(3,i+4)+avalues(3,i+5)+a
maa_cell(n,b) = {maa};
end
%Try filtering the CMA version first (also possibly the filter
%before the CMA would work better)
%Filter our some of the high speed error
passed = 1/T+5;
```

```
fNorm = passed / ((1/h)/2);
[l,d] = butter(10, fNorm, 'low');
LPFcma_a(1,:) = filtfilt(l, d, maa(1,:) );
LPFcma_a(2,:) = filtfilt(l, d, maa(2,:) );
LPFcma_a(3,:) = filtfilt(l, d, maa(3,:) );
LPFcell(n,b) = {LPFcma_a};
end
end
vals = cell2mat(maa_cell(1,1));
vals2 = cell2mat(Accel_cell(1,1));
vals3 = cell2mat(LPFcell(1,1));
vals4 = cell2mat(ma_cell(1,1));
figure(1);
plot3(vals(1,:),vals(2,:),vals(3,:),'b');
hold on;
plot3(vals2(1,:),vals2(2,:),vals2(3,:),'r');
%plot3(vals3(1,:),vals3(2,:),vals3(3,:),'g');
hold off;
axis equal;
figure(10)
for i=1:(points)
hold on;
plot(i,vals3(3,i),'r');
end
hold off
figure(14);
plot3(vals4(1,:),vals4(2,:),vals4(3,:),'b');
%So the central moving average filter helps, but it isnt enough
%As it turns out, SVD can be used to find the solution to the screw axis
```

```
%The matrix V defines the singular values, such the the first column is
    the
%direction in which my cloud of points tends the strongest (this is a
%mathematical proof that I need to do/understand as to why this works)
length(a)
shatguesscell = cell(length(a),1);
for n=1:length(a)
for i=1:N
wvals1 = cell2mat(ma_cell(n,i));
wvals2 = transpose(wvals1);
[U,S,V] = svd(wvals2);
wsave = wvals1(:,1);
if( dot(wsave,V(:,1)) > 0)
shat_guess_ave(:,i) = 1*V(:,1);
else
shat_guess_ave(:,i) = -1*V(:,1);
end
%for my own analysis
diff(:,i) = abs(s(:,i)-shat_guess_ave(:,i));
end
shatguesscell(n,1) = {shat_guess_ave};
end
%Better! We'll see if thats enough
%Now get the portion of the measured w that is only in the shat direction
%to help make the ws better so we can differentiate later
for n=1:length(a)
for b=1:N
wvalues = cell2mat(ma_cell(n,b));
for i=1:1:(points+2)
```

```
wnew = (dot(wvalues(:,i),shat_guess_ave(:,b)))*shat_guess_ave(:,b);
wvalues(:,i) = wnew;
end
ma_cell(n,b) = {wvalues};
end
end
vals = cell2mat(ma_cell(1,1));
vals2 = cell2mat(w_cell(1,1));
figure(2);
plot3(vals(1,:),vals(2,:),vals(3,:),'b');
hold on;
plot3(vals2(1,:),vals2(2,:),vals2(3,:),'r');
hold off;
axis equal;
figure(3);
for i=1:(points+2)
plot(i,norm(vals(:,i)),'b');
hold on;
plot(i,norm(vals2(:,i+4)),'r');
end
%Okay, now we need to take the derivative of our angular velocities and
%compare them to the analytical angular acceleration magnitude
8/3/2013 4:08 PM
ArcCalcErrVariable2
7 of 16
file:///C:/Users/connie/Desktop/octave/html/ArcCalcErrVariable2.html
%This means the first angular acceleration will be for t=-0.2 and the
    last for
%t = 19.8 (since the w loop goes from -0.25 to 19.8 now)
```

```
%Also restrict the w arrays to these values (chop off the first and last
%values...)
g = 1;
w_cell_new = cell(length(a),N);
accel_ang_cell = cell(length(a),N);
for n=1:length(a)
for i=1:N
w_v = cell2mat(ma_cell(n,i));
for f=-5:1:(T/h)-4
t0 = h*f;
%Store angular acceleration for checking purposes
alpha(g) = -1*((4*pi^2*B)/(T^2))*sin((2*pi*t0)/T);
g = g + 1;
if(f == -5 )
nstore(:,1) = w_v(:,f+6);
elseif( f == -4)
nstore(:,2) = w_v(:,f+6);
%Create the array of w that does not have the end 2 values
w_new(:,f+5) = w_v(:,f+6);
elseif( f == -3)
nstore(:,3) = w_v(:,f+6);
w_new(:,f+5) = w_v(:,f+6);
%Create the first acceleration value
accel_ang(:,f+4) = (nstore(:,3)-nstore(:,1))/(2*h);
else
nstore(:,1) = nstore(:,2);
nstore(:,2) = nstore(:,3);
nstore(:,3) = w_v(:,f+6);
accel_ang(:,f+4) = (nstore(:,3)-nstore(:,1))/(2*h);
```

```
%Skip the last value for the new array of w
if( f~=(points-4) )
w_new(:,f+5) = w_v(:,f+6);
end
end
end
%Maybe Instead of CMA, I can do a low-pass filter? Seems tp help a
%little bit...
%I think it is true to assume that all passed frequencies will be lower
%than the sampling frequency-in fact I KNOW what the sinusoidal
%frequency should be and can pass only things close to that
passed = 1/T+5;
fNorm = passed / ((1/h)/2);
[b,d] = butter(10, fNorm, 'low');
LPFangular(1,:) = filtfilt(b, d, accel_ang(1,:) );
LPFangular(2,:) = filtfilt(b, d, accel_ang(2,:) );
LPFangular(3,:) = filtfilt(b, d, accel_ang(3,:) );
%filtering w doesn't seem to help all that much
if ( i==1)
LPFw(1,:) = filtfilt(b, d, w_new(1,:) );
LPFw(2,:) = filtfilt(b, d, w_new(2,:) );
LPFw(3,:) = filtfilt(b, d, w_new(3,:) );
end
w_cell_new(n,i) = {w_new};
%accel_ang_cell = {accel_ang};
accel_ang_cell(n,i) = {LPFangular};
end
end
for i=1:(points)
```

```
plot(i,norm(LPFw(:,i)),'g');
end
hold off;
figure(4);
angularaccel = cell2mat(accel_ang_cell(1,1));
plot3(angularaccel(1,:),angularaccel(2,:),angularaccel(3,:),'r');
hold off;
figure(5)
for i=1:points
y = norm(angularaccel(:,i));
plot(i,y,'b');
hold on;
end
hold off;
%Have the arrays, guess the rs, both the magnitudes and the zero vector r
%(which will be our s0hat!)
%Skew symetric method of writing a cross product yields:
%It is a good note that a measured theta will be required here, either
    from
%the encoders or the IMU
s0alphacell = cell(length(a),1);
for n=1:length(a)
for i=1:N
%Grab the values for each joint
angular = cell2mat(accel_ang_cell(n,i));
w_new_2 = cell2mat(w_cell_new(n,i));
%accel_new2 = cell2mat(maa_cell(1,i));
accel_new2 = cell2mat(LPFcell(n,i));
for f=-5:1:((T/h)-6)
```

```
%We only want 1:3 elements of acceleration anyway
accel_new(:,f+6) = accel_new2(1:3,f+6);
t0 = f*0.05;
thetan = B*sin((2*pi*t0)/T);
%This is the matrix that results from doing the cross products in
%the acceleration equation in the skew symetric method
A11 = -1*w_new_2(3,f+6)^2-w_new_2(2,f+6)^2;
A12 = -1*angular(3,f+6)+w_new_2(2,f+6)*w_new_2(1,f+6);
A13 = w_new_2(3,f+6)*w_new_2(1,f+6)+angular(2,f+6);
A21 = w_new_2(1,f+6)*w_new_2(2,f+6)+angular(3,f+6);
A22 = -1*w_new_2(3,f+6)^2-w_new_2(1,f+6)^2;
A23 = -1*angular(1,f+6)+w_new_2(3,f+6)*w_new_2(2,f+6);
A31 = -1*angular(2,f+6)+w_new_2(1,f+6)*w_new_2(3,f+6);
A32 = w_new_2(2,f+6)*w_new_2(3,f+6)+angular(1,f+6);
A33 = -1*w_new_2(2,f+6)^2-w_new_2(1,f+6)^2;
A_new = [A11 A12 A13;A21 A22 A23; A31 A32 A33];
%2 methods of doing this-Ill do a comparitive analysis at some
%point, but the SVD method is more numerically dependable
A_beta = pinv(A_new);
R_beta = A_beta*accel_new(:,f+6);
%Truncated singular value decomposition for the answer with the
%least mean norm value
[U,S,V] = svd(A_new);
sig = diag(S);
sum = 0;
columns = size(U(1,:));
weight = 0;
for r=1:columns(2)
if( abs(sig(r)) > 0.0001 )
```

```
sum = sum +
    ((transpose(U(:,r))*accel_new(:,f+6))./sig(r,1))*transpose(V(:,r));
weight = abs(sig(r)) + weight;
end
end
R = transpose(sum);
Rmag_beta(f+6) = norm(R_beta);
Rmag(f+6) = norm(R);
weights(f+6) = weight;
R_mult(:,f+6) = -1*R;
R_mult2(:,f+6) = R_beta;
end
%Calculate the weighting for the average

TotalW = 0;
for b=1:2000
TotalW = weights(b) + TotalW;
end
Rsum_x = 0;
Rsum_y = 0;
Rsum_z = 0;
TP = 0;
for b=1:2000
Percent(b) = weights(b)/TotalW;
TP = TP + Percent(b);
R_mult(:,b) = (Percent(b))*R_mult(:,b);
Rsum_x = Rsum_x + R_mult(1,b);
Rsum_y = Rsum_y + R_mult(2,b);
Rsum_z = Rsum_z + R_mult(3,b);
```

```
end
%Average the values to get a more accurate response
%Use the weighted average
R_ave = [mean(R_mult(1,:));mean(R_mult(2,:));mean(R_mult(3,:))];
%R_ave = [Rsum_x;Rsum_y;Rsum_z];
R_ave_2 = [mean(R_mult2(1,:));mean(R_mult2(2,:));mean(R_mult2(3,:))];
s0_alpha(:,i) = R_ave;
s0_beta(:,i) = R_ave_2;
end
s0alphacell(n,1) = {s0_alpha};
end
for n=1:length(a)
s0_alpha = cell2mat(s0alphacell(n,1));
shat_guess_ave = cell2mat(shatguesscell(n,1));
for i=1:N
s0diff(:,i) = s0(:,i) - s0_alpha(:,i);
shatdiff(:,i) = s(:,i)-shat_guess_ave(:,i);
s0diffmag(n,i) = norm(s0diff(:,i));
percenterrs0(n,i)= (s0diffmag(n,i)/norm(s0(:,i)))*100;
shatdiffmag(n,i) = norm(shatdiff(:,i));
percenterrshat(n,i) = (shatdiffmag(n,i)/norm(s(:,i)))*100;
end
end
%Output values will be shat_guess ave and s0_alpha
%These can be used to calculate the screw rotations for any thetas
    starting
%from the reference pose
%Since we know the vectors (in the reference pose) to each screw axis,
%minimized such that they point to the joint in question, we can actually
```

```
%calculate the initial pose of the robot based on these (and get
    individual
%link lengths in our revolute serial manipulator)
%actually not quite
%Lets run a test here to see if the result is the same with both methods,
%both the shortest distance to the screw axis found by our calculations,
%and the joint to joint used to calculate the accelerations
%Lets give it some thetas
%I think we need to make an array of theta to make some calculation a
%little easier
theta_test = [0 0 0 0 0;
0.05 0.05 0.05 0.05 0.05;
0.1 0.1 0.1 0.1 0.1;
0.2 0.2 0.2 0.2 0.2;
0.3 0.3 0.3 0.3 0.3;
0.4 0.4 0.4 0.4 0.4;
0.5 0.5 0.5 0.5 0.5;
0.6 0.6 0.6 0.6 0.6;
0.7 0.7 0.7 0.7 0.7;
0.8 0.8 0.8 0.8 0.8;
0.9 0.9 0.9 0.9 0.9;
1 1 1 1 1;
1.2 1.2 1.2 1.2 1.2;
1.5 1.5 1.5 1.5 1.5;
1.8 1.8 1.8 1.8 1.8;
2 2 2 2 2;
2.2 2.2 2.2 2.2 2.2;
2.5 2.5 2.5 2.5 2.5;
2.8 2.8 2.8 2.8 2.8;
```

```
3 3 3 3 3;
3.2 3.2 3.2 3.2 3.2;
3.5 3.5 3.5 3.5 3.5;
3.8 3.8 3.8 3.8 3.8;
4 4 4 4 4];
%Calculate both versions of As
Afirst_cell = cell(1,N);
Aafter_cell = cell(1,N);
cross_test = cell(2,N);
Poses = length(theta_test);
posdiffcell = cell(length(a),1);
pererrcell = cell(length(a),1);
posdmagcell = cell(length(a),1);
for v=1:length(a)
s0_alpha = cell2mat(s0alphacell(v,1));
shat_guess_ave = cell2mat(shatguesscell(v,1));
for n=1:Poses
for i=1:N
a11 = (shat_guess_ave(1,i)^2-1)*(1-cos(theta_test(n,i)))+1;
a12 = shat_guess_ave(1,i)*shat_guess_ave(2,i)*(1-cos(theta_test(n,i)))
-shat_guess_ave(3,i)*sin(t
a13 = shat_guess_ave(1,i)*shat_guess_ave(3,i)*(1-cos(theta_test(n,i)))
+shat_guess_ave(2,i)*sin(t
a21 = shat_guess_ave(2,i)*shat_guess_ave(1,i)*(1-cos(theta_test(n,i)))
+shat_guess_ave(3,i)*sin(t
a22 = (shat_guess_ave(2,i)^2-1)*(1-cos(theta_test(n,i)))+1;
a23 = shat_guess_ave(2,i)*shat_guess_ave(3,i)*(1-cos(theta_test(n,i)))
-shat_guess_ave(1,i)*sin(t
a31 = shat_guess_ave(3,i)*shat_guess_ave(1,i)*(1-cos(theta_test(n,i)))
```

```
-shat_guess_ave(2,i)*sin(t
a32 = shat_guess_ave(3,i)*shat_guess_ave(2,i)*(1-cos(theta_test(n,i)))
+shat_guess_ave(1,i)*sin(t
a33 = (shat_guess_ave(3,i)^2-1)*(1-cos(theta_test(n,i)))+1;
a14 = t*shat_guess_ave(1,i)-s0_alpha(1,i)*(a11-1)-s0_alpha(2,i)*
a12-s0_alpha(3,i)*a13;
a24 = t*shat_guess_ave(2,i)-s0_alpha(1,i)*a21-s0_alpha(2,i)*
(a22-1)-s0_alpha(3,i)*a23;
a34 = t*shat_guess_ave(3,i)-s0_alpha(1,i)*a31-s0_alpha(2,i)*
a32-s0_alpha(3,i)*(a33-1);
a41 = 0;
a42 = 0;
a43 = 0;
a44 = 1;
A = [a11 a12 a13 a14; a21 a22 a23 a24; a31 a32 a33 a34; a41 a42 a43 a44];
Aafter_cell(1,i) = {A};
a11 = (s(1,i)^2-1)*(1-cos(theta_test(n,i)))+1;
a12 = s(1,i)*s(2,i)*(1-cos(theta_test(n,i)))-s(3,i)*sin(theta_test(n,i));
a13 = s(1,i)*s(3,i)*(1-cos(theta_test(n,i)))+s(2,i)*sin(theta_test(n,i));
a21 = s(2,i)*s(1,i)*(1-cos(theta_test(n,i)))+s(3,i)*sin(theta_test(n,i));
a22 = (s(2,i)^2-1)*(1-cos(theta_test(n,i)))+1;
a23 = s(2,i)*s(3,i)*(1-cos(theta_test(n,i)))-s(1,i)*sin(theta_test(n,i));
a31 = s(3,i)*s(1,i)*(1-cos(theta_test(n,i)))-s(2,i)*sin(theta_test(n,i));
a32 = s(3,i)*s(2,i)*(1-cos(theta_test(n,i)))+s(1,i)*sin(theta_test(n,i));
a33 = (s(3,i)^2-1)*(1-cos(theta_test(n,i)))+1;
a14 = t*s(1,i)-s0(1,i)*(a11-1)-s0(2,i)*a12-s0(3,i)*a13;
a24 = t*s(2,i)-s0(1,i)*a21-s0(2,i)*(a22-1)-s0(3,i)*a23;
a34 = t*s(3,i)-s0(1,i)*a31-s0(2,i)*a32-s0(3,i)*(a33-1);
a41 = 0;
```

```
a42 = 0;

a43 = 0;

a44 = 1;

A = [a11 a12 a13 a14; a21 a22 a23 a24; a31 a32 a33 a34; a41 a42 a43 a44];

Afirst_cell(1,i) = {A};

cross_test(1,i) = {cross(s(:,i),s0(:,i))};

cross_test(2,i) = {cross(shat_guess_ave(:,i),s0_alpha(:,i))};

end

q_new1 = cell2mat(Afirst_cell(1,1))*q0;

q_new2 = cell2mat(Aafter_cell(1,1))*q0;

if( N > 1)

q_new1 =

q_new2 =

end

if( N > 2 )

q_new1 =

q_new2 =

end

if( N > 3 )

q_new1 =

q_new2 =

end

if( N > 4 )

q_new1 =

q_new2 =

end

cell2mat(Afirst_cell(1,1))*cell2mat(Afirst_cell(1,2))*q0;

cell2mat(Aafter_cell(1,1))*cell2mat(Aafter_cell(1,2))*q0;

cell2mat(Afirst_cell(1,1))*cell2mat(Afirst_cell(1,2))*
```

```
cell2mat(Afirst_cell(1,3))*q0;
cell2mat(Aafter_cell(1,1))*cell2mat(Aafter_cell(1,2))*
cell2mat(Aafter_cell(1,3))*q0;
cell2mat(Afirst_cell(1,1))*cell2mat(Afirst_cell(1,2))*
cell2mat(Afirst_cell(1,3))*cell2mat(A
cell2mat(Aafter_cell(1,1))*cell2mat(Aafter_cell(1,2))*
cell2mat(Aafter_cell(1,3))*cell2mat(A
cell2mat(Afirst_cell(1,1))*cell2mat(Afirst_cell(1,2))*
cell2mat(Afirst_cell(1,3))*cell2mat(A
cell2mat(Aafter_cell(1,1))*cell2mat(Aafter_cell(1,2))*
cell2mat(Aafter_cell(1,3))*cell2mat(A
pos_diff(:,n) = (q_new1-q_new2);
8/3/2013 4:08 PM
ArcCalcErrVariable2
11 of 16
file:///C:/Users/connie/Desktop/octave/html/ArcCalcErrVariable2.html
posd_mag(:,n) = norm(pos_diff);
percenterr(:,n) = (posd_mag(:,n)/norm(q_new1))*100;
%Root mean square error in position
posRMS(:,n) = sqrt((q_new1(1,1)-q_new2(1,1))^2+(q_new1(2,1)-
q_new2(2,1))^2+(q_new1(3,1)-q_new2(3,1))^2);
q1save(:,n) = q_new1;
q2save(:,n) = q_new2;
end
pos_diff = pos_diff;
posdiffcell(v,1) = {pos_diff};
pererrcell(v,1) = {percenterr};
posdmagcell(v,1) = {posd_mag};
end
```

```
figure(1)
plot(theta_test(:,1),cell2mat(pererrcell(1,1)),'b');
% hold on;
% plot(theta_test(:,1),cell2mat(pererrcell(2,1)),'r');
% plot(theta_test(:,1),cell2mat(pererrcell(3,1)),'g');
% plot(theta_test(:,1),cell2mat(pererrcell(4,1)),'c');
xlabel('Joint Angle','FontSize',14);
ylabel('Percent Error in Position Estimate','FontSize',14);
title('Effect of Joint Angle Size on Tip Position Error in a 5DOF
    Arm','FontSize',16);
%legend('5% Measurement Error','10% Measurement Error','15% Measurement
    Error','20% Measurement Error');
figure(2)
plot(theta_test(:,1),cell2mat(posdmagcell(1,1)),'b');
% hold on;
% plot(theta_test(:,1),cell2mat(posdmagcell(2,1)),'r');
% plot(theta_test(:,1),cell2mat(posdmagcell(3,1)),'g');
% plot(theta_test(:,1),cell2mat(posdmagcell(4,1)),'c');
xlabel('Joint Angle','FontSize',14);
ylabel('Magnitude of Error in Position Estimate (m)','FontSize',14);
title('Effect of Joint Angle Size on Tip Position Error in a 5DOF
    Arm','FontSize',16);
%legend('5% Measurement Error','10% Measurement Error','15% Measurement
    Error','20% Measurement Error');
test = transpose(cell2mat(pererrcell(1,1)))
test2 = transpose(cell2mat(posdmagcell(1,1)))
ans =
2
test =
```

```
0

0.1223

0.2702

0.5260

0.8071

1.0883

1.3643

1.6363

1.9076

2.1809

2.4575

2.7367

3.0170

3.2811

3.5603

3.9678

4.4777

5.1528

6.0458

7.0243

7.9812

8.8294

9.6391

10.3027

test2 =

0

0.0012

0.0028

0.0056
```

0.0092

0.0132

0.0174

0.0218

0.0263

0.0307

0.0351

0.0395

0.0441

0.0496

0.0571

0.0664

0.0779

0.0937

0.1129

0.1323

0.1505

0.1637

0.1694

0.1716

Chapter F:

The following code was used to process data for each joint of the Ranger video arm.

```
Arc calculation-for a real robot

1 of 10

file:///C:/Users/connie/Desktop/octave/html/ArcCalcRealData.html

Arc calculation-for a real robot

Program which analyzes, one joint at a time, based on acceleration and

   rotation data, the screw vectors

%Read in the file I generated for Joints 1-5

OofJ = input('Enter the joint order as a an array, no commas: ');

RofJ = input('Enter the number of repeats for each joint, in order: ');

N = 0;

for i=1:length(OofJ)

N = N + RofJ(i);

end

%We will have repeats-to average for each joint now

JOINTS = cell(1,N);

w_cell = cell(1,N);

Accel_cell = cell(1,N);

t_cell = cell(1,N);

j_cell = cell(1,N);

%Now I need to read multiple values and sort the values into cells

%Each cell should be 1 X N, where N is the number of joints, going from

%first to last joint-and parsing out gyro and accel values

num = 1;

for b=1:length(OofJ)
```

```
for i=1:RofJ(i)
str = num2str(OofJ(b));
mystring = 'Joint';
endstr = '.dat';
dash = '-';
rptcnt = num2str(i);
mystr = strcat(mystring,str,dash,rptcnt,endstr);
matrix = csvread(mystr);
matrixa = matrix(:,3:5)*9.81;
matrixg = matrix(:,6:8)*pi/180;
matrixj = matrix(:,2);
matrixt = matrix(:,1);
w_cell(1,num) = {matrixg};
Accel_cell(1,num) = {matrixa};
t_cell(1,num) = {matrixt};
JOINTS(1,num) = {matrix};
j_cell(1,num) = {matrixj};
num = num+1;
end
end
%Now, we think, things can actually be calculated analytically-real robot!
%INPUTS:
%Acceleration array for the arc from accelerometer
%Angular velocity array fron the gyro
%
As a side note, the velocity array can be compared to the analytical
%
solution for a real robot to see where the differences are and possibly
%
```

```
use this to compensate for the derived angular acceleration as well
%Theta array as commanded to the motor
%Associated time array and steps
%OUTPUTS:
%Take the derivative of the gyro readings (compare to theoretical solution
%based on motor input pattern for a real arm)
%Calculate the s unit vector
%Grab the shat first-we can take an average for a real robot, so lets
%average here just to be safe
%So the error makes things really really sucky-try a central moving
   average
8/3/2013 4:10 PM
Arc calculation-for a real robot
2 of 10
file:///C:/Users/connie/Desktop/octave/html/ArcCalcRealData.html
%filter to make the data better-8 points for the average to start
%Start with just the rotations to see if it works
%length defines how many points we have
ma_cell = cell(1,N);
maa_cell = cell(1,N);
LPFcell = cell(1,N);
for b=1:N
wvalues = transpose(cell2mat(w_cell(1,b)));
avalues = transpose(cell2mat(Accel_cell(1,b)));
tvalues = transpose(cell2mat(t_cell(1,b)));
T = tvalues(end,1);
lengthy(b) = length(wvalues(1,:))-10;
%h for me is somewhat variable-but it dowsn't matter too much
h = mean(diff(tvalues));
```

```
for i=1:1:(lengthy(b)+2)
%Average of point 5 (i+4) (t = -0.25 s ) is the recorded first point of
    the moving
%average
ma(1,i) = (wvalues(1,i)+wvalues(1,i+1)+wvalues(1,i+2)+
wvalues(1,i+3)+wvalues(1,i+4)+wvalues
ma(2,i) = (wvalues(2,i)+wvalues(2,i+1)+wvalues(2,i+2)+
wvalues(2,i+3)+wvalues(2,i+4)+wvalues
ma(3,i) = (wvalues(3,i)+wvalues(3,i+1)+wvalues(3,i+2)+
wvalues(3,i+3)+wvalues(3,i+4)+wvalues
ma_cell(1,b) = {ma};
end
for i=1:1:lengthy(b)
%Average of point 5 (i+4) is the recorded first point of the moving
%(t = -0.2 s)
%average
%Also losing the last point (or point 409)
maa(1,i) = (avalues(1,i)+avalues(1,i+1)+avalues(1,i+2)+
avalues(1,i+3)+avalues(1,i+4)+avalue
maa(2,i) = (avalues(2,i)+avalues(2,i+1)+avalues(2,i+2)+
avalues(2,i+3)+avalues(2,i+4)+avalue
maa(3,i) = (avalues(3,i)+avalues(3,i+1)+avalues(3,i+2)+
avalues(3,i+3)+avalues(3,i+4)+avalue
maa_cell(1,b) = {maa};
end
%Try filtering the CMA version first (also possibly the filter
%before the CMA would work better)
%Filter our some of the high speed error
passed = 1/T+5;
```

```
fNorm = passed / ((1/h)/2);
[l,a] = butter(10, fNorm, 'low');
%s=fdesign.lowpass('Fp,Fst,Ap,Ast',0.01,1,1,.5);
%d=design(s,'equiripple'); %Lowpass FIR filter
LPFcma_a(1,:) = filtfilt(l,a, maa(1,:) );
LPFcma_a(2,:) = filtfilt(l,a, maa(2,:) );
LPFcma_a(3,:) = filtfilt(l,a, maa(3,:) );
LPFcell(1,b) = {LPFcma_a};
maa = [];
ma = [];
a = [];
LPFcma_a = [];
end
vals = cell2mat(maa_cell(1,1));
vals2 = cell2mat(Accel_cell(1,1));
vals3 = cell2mat(LPFcell(1,1));
vals4 = cell2mat(ma_cell(1,1));
vals5 = cell2mat(w_cell(1,1));
figure(1);
plot(1:length(vals(1,:)),vals(1,:),'r');
8/3/2013 4:10 PM
Arc calculation-for a real robot
3 of 10
file:///C:/Users/connie/Desktop/octave/html/ArcCalcRealData.html
hold on;
plot(1:length(vals3(1,:)),vals3(1,:),'b');
hold off;
figure(10)
for i=1:(lengthy(b))
```

```
hold on;
plot(i,vals3(3,b),'r');
end
hold off
figure(14);
plot3(vals4(1,:),vals4(2,:),vals4(3,:),'b');
hold on;
plot3(vals5(:,1),vals5(:,2),vals5(:,3),'r');
hold off;
%
%
%
%
figure(15)
plot(1:104,vals4(1,:),'b');
hold on;
plot(1:104,vals5(5:108,1),'r');
%So the central moving average filter helps, but it isnt enough
%As it turns out, SVD can be used to find the solution to the screw axis
%The matrix V defines the singular values, such the the first column is
    the
%direction in which my cloud of points tends the strongest (this is a
%mathematical proof that I need to do/understand as to why this works)
for i=1:N
wvals1 = cell2mat(ma_cell(1,i));
%wvals2 = cell2mat(w_cell(1,i));
wvals2 = transpose(wvals1);
[U,S,V] = svd(wvals2);
V
```

```
wsave = wvals2(1,:);
%if( dot(wsave,V(:,1)) > 0)
shat_guess_ave(:,i) = 1*V(:,1);
%else
%shat_guess_ave(:,i) = -1*V(:,1);
%end
%for my own analysis
%diff(:,i) = abs(s(:,i)-shat_guess_ave(:,i));
end
%
%Better! We'll see if thats enough
%Now get the portion of the measured w that is only in the shat direction
%to help make the ws better so we can differentiate later
for b=1:N
wvalues = cell2mat(ma_cell(1,b));
%wvalues = transpose(cell2mat(w_cell(1,b)));
for i=1:1:(lengthy(b)+2)
wnew = (dot(wvalues(:,i),shat_guess_ave(:,b)))*shat_guess_ave(:,b);
wvalues(:,i) = wnew;
end
ma_cell(1,b) = {wvalues};
end
vals = cell2mat(ma_cell(1,1));
vals2 = cell2mat(w_cell(1,1));
figure(2);
plot(1:lengthy(1)+2,vals(1,:),'b');
hold on;
%Okay, now we need to take the derivative of our angular velocities and
%compare them to the analytical angular acceleration magnitude
```

```
%This means the first angular acceleration will be for t=-0.2 and the
    last for
%t = 19.8 (since the w loop goes from -0.25 to 19.8 now)
%Also restrict the w arrays to these values (chop off the first and last
8/3/2013 4:10 PM
Arc calculation-for a real robot
4 of 10
file:///C:/Users/connie/Desktop/octave/html/ArcCalcRealData.html
%values...)
g = 1;
w_cell_new = cell(1,N);
accel_ang_cell = cell(1,N);
accle_angraw = cell(1,N);
for i=1:N
w_v = cell2mat(ma_cell(1,i));
for f=-5:1:lengthy(i)-4
%t0 = h*f;
%Store angular acceleration for checking purposes
%alpha(g) = -1*((4*pi^2*B)/(T^2))*sin((2*pi*t0)/T);
g = g + 1;
if(f == -5 )
nstore(:,1) = w_v(:,f+6);
elseif( f == -4)
nstore(:,2) = w_v(:,f+6);
%Create the array of w that does not have the end 2 values
w_new(:,f+5) = w_v(:,f+6);
elseif( f == -3)
nstore(:,3) = w_v(:,f+6);
w_new(:,f+5) = w_v(:,f+6);
```

```
%Create the first acceleration value
accel_ang(:,f+4) = (nstore(:,3)-nstore(:,1))/(2*h);
else
nstore(:,1) = nstore(:,2);
nstore(:,2) = nstore(:,3);
nstore(:,3) = w_v(:,f+6);
accel_ang(:,f+4) = (nstore(:,3)-nstore(:,1))/(2*h);
%Skip the last value for the new array of w
if( f~=(lengthy(i)-4) )
w_new(:,f+5) = w_v(:,f+6);
end
end
end
%Maybe Instead of CMA, I can do a low-pass filter? Seems tp help a
%little bit...
%I think it is true to assume that all passed frequencies will be lower
%than the sampling frequency-in fact I KNOW what the sinusoidal
%frequency should be and can pass only things close to that
passed = 1/T+5;
fNorm = passed / ((1/h)/2);
[b,a] = butter(10, fNorm, 'low');
LPFangular(1,:) = filtfilt(b, a, accel_ang(1,:) );
LPFangular(2,:) = filtfilt(b, a, accel_ang(2,:) );
LPFangular(3,:) = filtfilt(b, a, accel_ang(3,:) );
%filtering w doesn't seem to help all that much
if ( i==1)
LPFw(1,:) = filtfilt(b, a, w_new(1,:) );
LPFw(2,:) = filtfilt(b, a, w_new(2,:) );
LPFw(3,:) = filtfilt(b, a, w_new(3,:) );
```

```
end
w_cell_new(1,i) = {w_new};
%accel_ang_cell = {accel_ang};
accel_ang_cell(1,i) = {LPFangular};
accel_angraw(1,i) = {accel_ang};
LPFangular = [];
accel_ang = [];
8/3/2013 4:10 PM
Arc calculation-for a real robot
5 of 10
file:///C:/Users/connie/Desktop/octave/html/ArcCalcRealData.html
w_new = [];
end
plot(1:lengthy(1),LPFw(1,:),'g');
hold off;
figure(4);
angularaccel = cell2mat(accel_ang_cell(1,1));
accel_test = cell2mat(accel_angraw);
plot3(angularaccel(1,:),angularaccel(2,:),angularaccel(3,:),'r');
hold off;
figure(5)
for i=1:lengthy(1)
y = norm(angularaccel(:,i));
f = norm(accel_test(:,i));
plot(i,y,'--bs');
plot(i,f,'--rs');
hold on;
end
hold off;
```

```
%what about a CMA for angular acceleration?
%CMA seems to make it WORSE for some reason
% ang_cma_cell = cell(1,N);
% for i=1:N
%
angularaccel = cell2mat(accel_ang_cell(1,i));
%
ang_cma(:,1) = angularaccel(:,1);
%
ang_cma(1,2) = (angularaccel(1,1) + angularaccel(1,2) +
angularaccel(1,3))/3;
%
ang_cma(2,2) = (angularaccel(2,1) + angularaccel(2,2) +
angularaccel(2,3))/3;
%
ang_cma(3,2) = (angularaccel(3,1) + angularaccel(3,2) +
angularaccel(3,3))/3;
%
ang_cma(1,3) = (angularaccel(1,1) + angularaccel(1,2) +
angularaccel(1,3) + angularaccel(1,4
%
ang_cma(2,3) = (angularaccel(2,1) + angularaccel(2,2) +
angularaccel(2,3) + angularaccel(2,4
%
ang_cma(3,3) = (angularaccel(3,1) + angularaccel(3,2) +
angularaccel(3,3) + angularaccel(3,4
%
ang_cma(1,4) = (angularaccel(1,1) + angularaccel(1,2) +
angularaccel(1,3) + angularaccel(1,4
```

```
%
ang_cma(2,4) = (angularaccel(2,1) + angularaccel(2,2) +
angularaccel(2,3) + angularaccel(2,4
%
ang_cma(3,4) = (angularaccel(3,1) + angularaccel(3,2) +
angularaccel(3,3) + angularaccel(3,4
%
%
for b=1:(length-8)
%
ang_cma(1,b+4) = (angularaccel(1,b)+angularaccel(1,b+1)+
angularaccel(1,b+2)+angularac
%
ang_cma(2,b+4) = (angularaccel(2,b)+angularaccel(2,b+1)+
angularaccel(2,b+2)+angularac
%
ang_cma(3,b+4) = (angularaccel(3,b)+angularaccel(3,b+1)+
angularaccel(3,b+2)+angularac
%
end
%
%
ang_cma(:,length) = angularaccel(:,length);
%
ang_cma(1,(length-1)) = (angularaccel(1,(length-2)) +
angularaccel(1,(length-1)) + angularac
%
ang_cma(2,(length-1)) = (angularaccel(2,(length-2)) +
angularaccel(2,(length-1)) + angularac
```

```
%
ang_cma(3,(length-1)) = (angularaccel(3,(length-2)) +
angularaccel(3,(length-1)) + angularac
%
ang_cma(1,(length-2)) = (angularaccel(1,(length-4)) +
angularaccel(1,(length-3)) + angularac
%
ang_cma(2,(length-2)) = (angularaccel(2,(length-4)) +
angularaccel(2,(length-3)) + angularac
%
ang_cma(3,(length-2)) = (angularaccel(3,(length-4)) +
angularaccel(3,(length-3)) + angularac
%
ang_cma(1,(length-3)) = (angularaccel(1,(length-6)) +
angularaccel(1,(length-5)) + angularac
%
ang_cma(2,(length-3)) = (angularaccel(2,(length-6)) +
angularaccel(2,(length-5)) + angularac
%
ang_cma(3,(length-3)) = (angularaccel(3,(length-6)) +
angularaccel(3,(length-5)) + angularac
%
ang_cma_cell(1,i) = {ang_cma};
% end
%
% angularaccelcma = cell2mat(ang_cma_cell(1,1));
% for i=1:length
%
y = norm(angularaccelcma(:,i));
```

```
%
plot(i,y,'r');
% end
%Have the arrays, guess the rs, both the magnitudes and the zero vector r
%(which will be our s0hat!)
%Skew symetric method of writing a cross product yields:
%It is a good note that a measured theta will be required here, either
    from
%the encoders or the IMU
for i=1:N
```



```
%Grab the values for each joint
angular = cell2mat(accel_ang_cell(1,i));
%angular = cell2mat(accel_angraw(1,i));
w_new_2 = cell2mat(w_cell_new(1,i));
%accel_new2 = cell2mat(maa_cell(1,i));
accel_new2 = cell2mat(LPFcell(1,i));
for f=-5:1:(lengthy(i)-6)
%We only want 1:3 elements of acceleration anyway
accel_new(:,f+6) = accel_new2(1:3,f+6);
t0 = f*0.05;
%thetan = B*sin((2*pi*t0)/T);
%This is the matrix that results from doing the cross products in
%the acceleration equation in the skew symetric method
A11 = -1*w_new_2(3,f+6)^2-w_new_2(2,f+6)^2;
A12 = -1*angular(3,f+6)+w_new_2(2,f+6)*w_new_2(1,f+6);
```

```
A13 = w_new_2(3,f+6)*w_new_2(1,f+6)+angular(2,f+6);
A21 = w_new_2(1,f+6)*w_new_2(2,f+6)+angular(3,f+6);
A22 = -1*w_new_2(3,f+6)^2-w_new_2(1,f+6)^2;
A23 = -1*angular(1,f+6)+w_new_2(3,f+6)*w_new_2(2,f+6);
A31 = -1*angular(2,f+6)+w_new_2(1,f+6)*w_new_2(3,f+6);
A32 = w_new_2(2,f+6)*w_new_2(3,f+6)+angular(1,f+6);
A33 = -1*w_new_2(2,f+6)^2-w_new_2(1,f+6)^2;
A_new = [A11 A12 A13;A21 A22 A23; A31 A32 A33];
%2 methods of doing this-Ill do a comparitive analysis at some
%point, but the SVD method is more numerically dependable
A_beta = pinv(A_new);
R_beta = A_beta*accel_new(:,f+6);
%Truncated singular value decomposition for the answer with the
%least mean norm value
[U,S,V] = svd(A_new);
sig = diag(S);
sum = 0;
columns = size(U(1,:));
weight = 0;
for r=1:columns(2)
if( abs(sig(r)) > 0.0001 )
sum = sum + ((transpose(U(:,r))*accel_new(:,f+6)).
/sig(r,1))*transpose(V(:,r));
weight = abs(sig(r)) + weight;
end
end
R = transpose(sum);
Rmag_beta(f+6) = norm(R_beta);
Rmag(f+6) = norm(R);
```

```
weights(f+6) = weight;
R_mult(:,f+6) = R;
R_mult2(:,f+6) = R_beta;
%
%
%
%
%
%
%
%
%
%
end
%Calculate the weighting for the average
TotalW = 0;
for b=1:2000
TotalW = weights(b) + TotalW;
end
Rsum_x = 0;
Rsum_y = 0;
Rsum_z = 0;
TP = 0;
for b=1:2000
Percent(b) = weights(b)/TotalW;
8/3/2013 4:10 PM
Arc calculation-for a real robot
7 of 10
%
```

```
%
%
%
%
%
file:///C:/Users/connie/Desktop/octave/html/ArcCalcRealData.html
TP = TP + Percent(b);
R_mult(:,b) = (Percent(b))*R_mult(:,b);
Rsum_x = Rsum_x + R_mult(1,b);
Rsum_y = Rsum_y + R_mult(2,b);
Rsum_z = Rsum_z + R_mult(3,b);
end
%Average the values to get a more accurate response
%Use the weighted average
R_ave = [mean(R_mult(1,:));mean(R_mult(2,:));mean(R_mult(3,:))];
%R_ave = [Rsum_x;Rsum_y;Rsum_z];
R_ave_2 = [mean(R_mult2(1,:));mean(R_mult2(2,:));mean(R_mult2(3,:))];
s0_alpha(:,i) = R_ave;
s0_beta(:,i) = R_ave_2;
end
s0_alpha
shat_guess_ave
%Output values will be shat_guess ave and s0_alpha
%These can be used to calculate the screw rotations for any thetas
    starting
%from the reference pose
%For these real repeated values, now we need to average our answers
shat_guess_avef = zeros(3,length(OofJ));
s0_alphaf = zeros(3,length(OofJ));
```

```
for i=1:length(OofJ)
Repeats = RofJ(1);
if(i > 1)
Spot = RofJ(i-1);
for b=Spot+1:Spot+RofJ(i)
shat_guess_avef(:,i) = shat_guess_avef(:,i) + shat_guess_ave(:,b);
s0_alphaf(:,i) = s0_alphaf(:,i) + s0_alpha(:,b);
end
else
for b=1:Repeats
shat_guess_avef(:,i) = shat_guess_avef(:,i) + shat_guess_ave(:,b);
s0_alphaf(:,i) = s0_alphaf(:,i) + s0_alpha(:,b);
end
end
shat_guess_ave_fin(:,i) = shat_guess_avef(:,i)/RofJ(i);
s0_alpha_fin(:,i) = s0_alphaf(:,i)/RofJ(i);
end
%Since we know the vectors (in the reference pose) to each screw axis,
%minimized such that they point to the joint in question, we can actually
%calculate the initial pose of the robot based on these (and get
    individual
%link lengths in our revolute serial manipulator)
%actually not quite
%Lets run a test here to see if the result is the same with both methods,
%both the shortest distance to the screw axis found by our calculations,
%and the joint to joint used to calculate the accelerations
%Lets give it some thetas
% theta_test = [0.2 0.4 0 0 0];
% %Calculate both versions of As
```

```
% Afirst_cell = cell(1,N);
% Aafter_cell = cell(1,N);
% cross_test = cell(2,N);
%
% for i=1:N
%
a11 = (shat_guess_ave(1,i)^2-1)*(1-cos(theta_test(i)))+1;
%
a12 = shat_guess_ave(1,i)*shat_guess_ave(2,i)*
(1-cos(theta_test(i)))-shat_guess_ave(3,i)
%
a13 = shat_guess_ave(1,i)*shat_guess_ave(3,i)*
(1-cos(theta_test(i)))+shat_guess_ave(2,i)
%
a21 = shat_guess_ave(2,i)*shat_guess_ave(1,i)*
(1-cos(theta_test(i)))+shat_guess_ave(3,i)
%
a22 = (shat_guess_ave(2,i)^2-1)*(1-cos(theta_test(i)))+1;
%
a23 = shat_guess_ave(2,i)*shat_guess_ave(3,i)*
(1-cos(theta_test(i)))-shat_guess_ave(1,i)
%
a31 = shat_guess_ave(3,i)*shat_guess_ave(1,i)*
(1-cos(theta_test(i)))-shat_guess_ave(2,i)
%
a32 = shat_guess_ave(3,i)*shat_guess_ave(2,i)*
(1-cos(theta_test(i)))+shat_guess_ave(1,i)
%
a33 = (shat_guess_ave(3,i)^2-1)*(1-cos(theta_test(i)))+1;
```

```
%
a14 = t*shat_guess_ave(1,i)-s0_alpha(1,i)*(a11-1)-
s0_alpha(2,i)*a12-s0_alpha(3,i)*a13;
```

```
%
%
%
%
%
%
%
%
%
%
%
%
%
%
%
%
%
%
%
%
%
%
%
%
```

```
%
%
%
%
%
%
%
%
%
%
%
file:///C:/Users/connie/Desktop/octave/html/ArcCalcRealData.html
a24 = t*shat_guess_ave(2,i)-s0_alpha(1,i)*a21-s0_alpha(2,i)*
(a22-1)-s0_alpha(3,i)*a23;
a34 = t*shat_guess_ave(3,i)-s0_alpha(1,i)*a31-s0_alpha(2,i)*
a32-s0_alpha(3,i)*(a33-1);
a41 = 0;
a42 = 0;
a43 = 0;
a44 = 1;
A = [a11 a12 a13 a14; a21 a22 a23 a24; a31 a32 a33 a34; a41 a42 a43 a44];
Aafter_cell(1,i) = {A};
a11 = (s(1,i)^2-1)*(1-cos(theta_test(i)))+1;
a12 = s(1,i)*s(2,i)*(1-cos(theta_test(i)))-s(3,i)*sin(theta_test(i));
a13 = s(1,i)*s(3,i)*(1-cos(theta_test(i)))+s(2,i)*sin(theta_test(i));
a21 = s(2,i)*s(1,i)*(1-cos(theta_test(i)))+s(3,i)*sin(theta_test(i));
a22 = (s(2,i)^2-1)*(1-cos(theta_test(i)))+1;
a23 = s(2,i)*s(3,i)*(1-cos(theta_test(i)))-s(1,i)*sin(theta_test(i));
```

```
a31 = s(3,i)*s(1,i)*(1-cos(theta_test(i)))-s(2,i)*sin(theta_test(i));
a32 = s(3,i)*s(2,i)*(1-cos(theta_test(i)))+s(1,i)*sin(theta_test(i));
a33 = (s(3,i)^2-1)*(1-cos(theta_test(i)))+1;
a14 = t*s(1,i)-s0(1,i)*(a11-1)-s0(2,i)*a12-s0(3,i)*a13;
a24 = t*s(2,i)-s0(1,i)*a21-s0(2,i)*(a22-1)-s0(3,i)*a23;
a34 = t*s(3,i)-s0(1,i)*a31-s0(2,i)*a32-s0(3,i)*(a33-1);
a41 = 0;
a42 = 0;
a43 = 0;
a44 = 1;
A = [a11 a12 a13 a14; a21 a22 a23 a24; a31 a32 a33 a34; a41 a42 a43 a44];
Afirst_cell(1,i) = {A};
cross_test(1,i) = {cross(s(:,i),s0(:,i))};
cross_test(2,i) = {cross(shat_guess_ave(:,i),s0_alpha(:,i))};
end
q_new1 =
q_new2 =
pos_diff
posd_mag
cell2mat(Afirst_cell(1,1))*cell2mat(Afirst_cell(1,2))*
cell2mat(Afirst_cell(1,3))*cell2m
cell2mat(Aafter_cell(1,1))*cell2mat(Aafter_cell(1,2))*
cell2mat(Aafter_cell(1,3))*cell2m
= q_new1-q_new2;
= norm(pos_diff);
%The new position is the same with both methods of calculation-proven
%kinematically
%This does throw a wrench in the pose-how can I calculate it?
%I think it can be done by theory of perpendicular distances, point to
```

```
    line
%shortest distance, and a dot product = 0
%Calculate the first R12 we want
LMN(:,1) = s0_alpha_fin(:,1);
% for i=2:N
%
D = LMN(1,i-1)*shat_guess_ave(1,i)+LMN(2,i-1)*shat_guess_ave(2,i)
+LMN(3,i-1)*shat_guess_ave(
%
F = s0_alpha(1,i)*shat_guess_ave(1,i)+s0_alpha(2,i)*shat_guess_ave
(2,i)+s0_alpha(3,i)*shat_g
%
K = (D-F)/(shat_guess_ave(1,i)^2+shat_guess_ave(2,i)^2+
shat_guess_ave(3,i)^2);
%
%
L = s0_alpha(1,i)+K*shat_guess_ave(1,i);
%
M = s0_alpha(2,i)+K*shat_guess_ave(2,i);
%
N = s0_alpha(3,i)+K*shat_guess_ave(3,i);
%
%
LMN(:,i) = [L;M;N];
% end
%A few more calculations here and a plot of the calculated manipulator in
%"0" pose in 3D space would be good, maybe with the shat and s0 vectors
    displayed
R1 = LMN(:,1);
```

```
% R2 = LMN(:,2);
% R3 = LMN(:,3);
% R4 = LMN(:,4);
% R5 = LMN(:,5);
%
% R12 = R2 - R1;
% R23 = R3 - R2;
% R34 = R4 - R3;
% R45 = R5 - R4;
figure(6);
8/3/2013 4:10 PM
Arc calculation-for a real robot
9 of 10
file:///C:/Users/connie/Desktop/octave/html/ArcCalcRealData.html
vec = quiver3(0,0,0,R1(1),R1(2),R1(3),'b');
hold on;
% vec2 = quiver3(R1(1),R1(2),R1(3),R12(1),R12(2),R12(3),'r');
% hold on;
% R = R1 + R12;
% if( abs(norm(R) - norm(R23) < 0.0001) )
%
R23 = zeros(3,1);
% end
% vec3 = quiver3(R(1),R(2),R(3),R23(1),R23(2),R23(3), 'g');
% R = R + R23;
% if( abs(norm(R) - norm(R34)) < 0.0001 )
%
R34 = zeros(3,1);
% end
```

```
% vec4 = quiver3(R(1),R(2),R(3),R34(1),R34(2),R34(3), 'b');
% R = R + R34;
% if( abs(norm(R) - norm(R45)) < 0.0001 )
%
R45 = zeros(3,1);
% end
% vec5 = quiver3(R(1),R(2),R(3),R45(1),R45(2),R45(3), 'r');
% axis([-2 2 -1 1 -1 1]);
hold off;
accel1
accel2
accel3
accel4
=
=
=
=
cell2mat(Accel_cell(1,1));
cell2mat(Accel_cell(1,2));
cell2mat(Accel_cell(1,3));
cell2mat(Accel_cell(1,4));
t1 = cell2mat(t_cell(1,1));
t2 = cell2mat(t_cell(1,2));
t3 = cell2mat(t_cell(1,3));
t4 = cell2mat(t_cell(1,4));
j1 = cell2mat(j_cell(1,1));
j2 = cell2mat(j_cell(1,2));
j3 = cell2mat(j_cell(1,3));
j4 = cell2mat(j_cell(1,4));
```

```
global matrixax tcell jcell;
matrixax = [accel1(:,1);accel2(:,1);accel3(:,1);accel4(:,1)];
matrixay = [accel1(:,2);accel2(:,2);accel3(:,2);accel4(:,2)];
matrixaz = [accel1(:,3);accel2(:,3);accel3(:,3);accel4(:,3)];
tcell = [t1;t2;t3;t4];
jcell = [j1;j2;j3;j4];
hold off;
figure(2)
plot(tcell,matrixay(:,1));
figure(3)
plot(tcell,matrixaz(:,1));
% CALCULATE THE FILTER COEFFICIENTS
SamplePeriod = mean(diff(tcell));
SampleFrequency = 1/SamplePeriod;
%
SET THE FILTER BANDWIDTH (HZ)
FilterBandwidth = 10.0 ;
display(FilterBandwidth) ;
[rows,cols] = size(matrixax);
NumberSamples = rows ;
column = 1;
F = (0:NumberSamples-1) ;
F(:) = matrixay(:,column) ;
8/3/2013 4:10 PM
Arc calculation-for a real robot
10 of 10
file:///C:/Users/connie/Desktop/octave/html/ArcCalcRealData.html
NFFT = min(256,NumberSamples) ;
Xave = mean(matrixay(:,column)) ;
```

```
figure(4)
psd(matrixay(:,column)-Xave,NFFT,SampleFrequency,[]) ;
title('YAccel J2 PSD') ;
% CALCULATE THE FILTER COEFFICIENTS
SamplePeriod = mean(diff(tcell));
SampleFrequency = 1/SamplePeriod;
FilterOrder = 10 ;
wn = FilterBandwidth/(0.5*SampleFrequency) ;
[b,a] = butter(FilterOrder,wn) ;
% RUN THE FORCE DATA THROUGH THE FILTER AND PLOT
global F_filt;
F_filt = filtfilt(b,a,F(:)) ;
figure(5);
plot(tcell(:),F(:),'b',tcell(:),F_filt(:),'r') ;
xlabel('time (sec)') ;
ylabel('YAccel 2 (N)') ;
title('YAccel J2 Filt and UnFilt vs. time') ;
figure(1)
plot(tcell,matrixay(:,1));
hold on;
options = optimset('Display', 'ITER', 'MaxIter',1000,'MaxFunEvals',10000,
    'TolX',1*10^-8);
cost = fminsearch('myfunc4', [-4 8],options);
Amp = cost(1);
Offset = cost(2);
for i=1:length(jcell)
Y(i) = Amp*sin(jcell(i)+Offset);
end
plot(tcell,Y,'r');
```

```
testdiff = abs(matrixax - transpose(Y));

testdiffnorm = mean(testdiff);

max(Y)

max(matrixax)

hold off;

Error using ==> input

Cannot call INPUT from EVALC.

Error in ==> ArcCalcRealData at 6

OofJ = input('Enter the joint order as a an array, no commas: ');

Published with MATLAB 7.6

8/3/2013 4:10 PM
```

Bibliography

[1] Mittendorfer, Philipp, and Gordon Cheng. "Open-loop Self-Calibration of Articulated Robots with Artificial Skins." 2012 IEEE International Conference on Robotics and Automation. RiverCentre,Saint Paul, Minnesota, USA: May 14-18, 2012.

[2] Tsai, Lung-wen. "Robot Analysis: The Mechanics of Serial and Parallel manipulators." New York,NY: John Wiley and Sons, Inc, 1999.

[3] O'Leary, Diane P. "Scientific Computing with Case Studies." Philadelphia, PA: Society for Industrial and Applied Mathematics, 2009.

[4] Canepa, Gaetano and Hollerbach, John and Boelen, Alexander. "Kinematic Calibration by Means of a Triaxial Accelerometer." 1994 IEEE, Biorobotics Laboratory, McGill University, 3775 University St., Montreal, PQ, Canada, H3A 2B4, 1994.

[5] Chen, I-Ming and Yang, Guilin. "Configuration Independant Kinematics for Modular Robots." 1996 IEEE International Conference on Robotics and Automation. Minneapolis, Minnesota: April, 1996.

[6] Kelmar, Laura and Khosla, Pradeep. "Automatic Generation of Kinematics for a Reconfigurable Modular Manipulator System." 1988 IEEE International Conference on Robotics and Automation. Philidelpha, PA: 24-29 April, 1988.

[7] Aghili, Farhad and Su, Chun-yi. "Reconfigurable Space Manipulators for Inorbit Servicing and Space Exploration." 2012 International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS). 2012.

[8] Rousseau, Patrick and Desrochers, Alain and Krouglicof, Nicholas. "Machine Vision System for Automatic Identification of Robot Kinematic Parameters." IEEE Transactions of Robotics and Automation, Vol. 17, NO. 6, December 2001.

[9] Akin, David and Roberts, Brian and Roderick, Stephen and Smith, Walter and Henriette, Jean-marc. "MORPHbots: Lightweight Modular Self-Reconfigurable Robotics for Space Assembly, Inspectionm and Servicing." Space 2006. San Jose, California: September 19-21, 2006.

[10] Craig, John. "Introduction to Robotics, Mechanics and Control." Upper Saddle River, New Jersey: Pearson Education, Inc., 2005.

[11] Quigley, Morgan and Brewer, Reuben and Soundararaj, Sai and Pradeep, Vijay and Le, Quoc and Ng, Andrew. "Low-cost Accelerometers for Robotic Manipulator Perception." 2010 IEEE International Conference on Intelligent Robots and Systems. Taipei, Taiwan: October 18-22, 2010.

[12] Roan, Phiilip and Deshpande, Nikhil and Wang, Yizhou and Pitzer, Benjamin. "Manipulator State Estimation with Low Cost Accelerometers and Gyroscopes." 2012 IEEE International Conference on Intellifent Robots and Systems. Vilamoura, Algarve, Portugal: October 7-12, 2012.

[13] Wu, Zhoncheng and Meng, Ming and Shen, Fei. "Interaction Force Measurement of Robotic Manipulator Based on 12DOF Force Sensor." 2004 IEEE International Conference on Information Aquisition. 2004.

[14] Aldridge, Hal and Juang, Jer-Nan. "Joint Position sensor fault tolerance in robot systems using Cartesian accelerometers." 1996 AIAA Meeting Papers, AIAA Paper 96-3898. 1996.

[15] Li, Y.F. and Chen, X.B. "End-Point Sensing and State Observation of a Flexible-Link Robot." IEEE/ASME Transaction of Mechatronics, Vol. 6, NO. 3: September, 2001.

[16] D'Amore, Nicholas. "Development of a Reusable Top-Level Control Architecture for a Robotic Manipulator." University of Maryland, College Park, ProQuest, UMI Dissertations Publishing, 2010. 1482497.

[17] Ellsberry, Andrew. Development and evaluation of a flexible distributed control architecture. unpublished draft, M.S. thesis, University of Maryland, College Park, MD, 2010.

[18] Buss, Samuel. "Introduction to Inverse Kinematics with Jacobian Transpose, Psuedoinverse and Damped Least Squares methods." 2004 IEEE Journal of Robotics and Automation. Depeartment of Mathematics, University of California, San Diego, April 2004.

[19] Paredis, Christiaan, Brown, Benjamin, and Khosla, Pradeep. "A Rapidly Deployable Manipulator System." 1996 International Conference on Robotics and Automation. Minneapolis, Minnesota, April, 1996.

[20] Akin, David, Lane, J., Roberts, Brians, and Weisman, Steve. "Robotic Capabilities for Complex Space Operations." AIAA Space 2001-Conference and Exposition. Albuquerque, NM, Aug. 28-31, 2001.

[21] Valsamos, Harry and Aspragathos, Nick. "Determination of Anatomy and Configuration of a Reconfigurable Manipulator for the Optimal Manipulability." Mechanical engineering and Aerunautics Department, University of Patras Rio, Achaia, Greece, 26500.

[22] Prats, Mario et. al. "Reconfigurable AUV for intervention missions: a case study on underwater object recovery." Intel Serv Robotics 2012 5:19-31, Special Issue of Marine Robotics, Part 2. 2012

[23] Chen, I-Ming and Yang, Guilin. "Inverse Kinematics for Modular Reconfigurable Robots." IEEE International Conference on Robotics and Automation, Leuven, Belgium, May, 1998.

[24] Chen, I-Ming. "Theory and Applications of Modular Reconfigurable Robotic Systems." PhD Thesis, California Institute of Technology, CA, 1994.

[25] Deo, N. "Graph Theory with Applications to Engineering and Computer Science." Prentice-Hall, 1974.