Title
Machine Learning Applications to Robot Control

Author
Abdul-hadi, Omar

Publication Date
2018

Peer reviewed|Thesis/dissertation

Machine Learning Applications to Robot Control

by

Omar Abdul-hadi

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ruzena Bajcsy, Co-chair
Professor Masayoshi Tomizuka, Co-chair
Professor Andrew Packard
Professor Edward Lee

Spring 2018

Machine Learning Applications to Robot Control

Abstract

Machine Learning Applications to Robot Control

by

Omar Abdul-hadi

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Ruzena Bajcsy, Co-chair

Professor Masayoshi Tomizuka, Co-chair

Control of robot manipulators can be greatly improved with the use of velocity and torque feedforward control. However, the effectiveness of feedforward control greatly relies on the accuracy of the model. In this study, kinematics and dynamics analysis is performed on a six axis arm, a Delta2 robot, and a Delta3 robot. Velocity feedforward calculation is performed using the traditional means of using the kinematics solution for velocity. However, a neural network is used to model the torque feedforward equations. For each of these mechanisms, we first solve the forward and inverse kinematics transformations. We then derive a dynamic model. Later, unlike traditional methods of obtaining the dynamics parameters of the dynamics model, the dynamics model is used to infer dependencies between the input and output variables for neural network torque estimation. The neural network is trained with joint positions, velocities, and accelerations as inputs, and joint torques as outputs. After training is complete, the neural network is used to estimate the feedforward torque effort. Additionally, an investigation is done on the use of neural networks for deriving the inverse kinematics solution of a six axis arm. Although the neural network demonstrated outstanding ability to model complex mathematical equations, the inverse kinematics solution was not accurate enough for practical use.

To my family and friends

Contents

List of Figures

List of Tables

Acknowledgments

I would like to especially thank my research advisor, Professor Ruzena Bajcsy for all her support throughout the PhD program. Her commitment to her students and research has been a very inspiring experience for me. I would also like to thank my co-advisor, Professor Masayoshi Tomizuka for his continual support during my graduate study at UC Berkeley. Additionally, I would like to thank Professor Andrew Packard for his insights into control strategies and his availability throughout my study at UC Berkeley. I would also like to thank Professor Edward Lee for teaching the course that helped me obtain my current job at Yaskawa. I took Introduction to Embedded Systems which had the course number EE C 149 with Professor Lee in Fall 2012 and it directly helped me get prepared for my interview questions with Yaskawa as an embedded software engineer. Coming from a Mechanical Engineering background, which is the field of my Bachelors of Science degree, EE C 149 was of great help for me. I also owe many thanks to my employer Yaskawa America, Inc. for providing all the robots and equipment for the research. Theodore Phares and Dr. Edward Nicolson from Yaskawa America, Inc. were very supportive while I worked for the company and attended school. Mustapha Harb, who was a Civil Engineering PhD student at UC Berkeley, encouraged me to take the course on Machine Learning with him. While taking that course, Mustapha Harb, Mohamed Hariri Nokob and I worked on the class project together. The class project, which focused on six axis inverse kinematics solutions using neural networks, was a success and it has made its way into my thesis. The class project has also inspired the focus of my thesis on neural network for dynamics. Larry Huang at Yaskawa America, Inc. is an expert in the operation of Delta robot mechanisms. He has been of great help with commissioning the Delta robots and operating them to collect data for research. Finally, Dr. Fernando L. Garcia Bermudez at Yaskawa America, Inc. has been of great help, because he is the author of the Delta3 kinematics derivation found in appendix B. Additionally, he added support for Delta3 mechanisms in the controller firmware.

Chapter 1

Introduction

1.1 Background

Neural networks [18] are increasingly making their way into practical robotics applications, because of improvements in computing power [19]. A challenge with using neural networks is that they are often slow and impractical for high speed applications [9]. In our work, we investigate the use of neural networks in learning complex nonlinear functions in relation to various types of robots. The functions we attempt to learn are the inverse kinematics and dynamics equations.

Literature review

Much of the prior work on machine learning for kinematics has been on simpler mechanisms such as 3 DOF (Degree of Freedom) manipulators [14] and [31]. On the other hand, work done on 6 axis inverse kinematics such as [2] does not solve the most general 6 axis configurations that we attempt to solve.

The closest work that we were able to find on machine learning for dynamics was the use of a neural network to model a 2 DOF robotic arm [23]; however like other work in this area [22] and [3], it lacks emphasis on runtime performance limitations of neural networks.

Embedded robot controllers have limited CPU power, but are expected to keep up with high update rates (usually above 250 Hz). This means a neural network must run in less than 100 μs before it can even be considered for an industrial controller.

Although it may be easy to model a system using a large neural network, with multiple hidden layers, the challenges lie in training a small enough neural network such that it meets the performance limitations, yet is still capable of capturing the complexities of a nonlinear dynamic system.

Other work on neural networks for dynamics such as [10] and [13] only test the performance in simulation mode and have no testing on physical mechanisms. As for [16], only the performance of the neural network torque predictions are discussed.

The focus of our work is to not only predict the estimated torque efforts using a neural network, but to also improve the tracking performance on the actual mechanism. As we will see later on, these are two completely different problems.

1.2 Motivation

Neural Networks for Kinematics

In the case of serial manipulators, it is easy to obtain the forward kinematics solutions using Denavit Hartenberg parameters. However, the inverse kinematics solution is much harder to solve. In the case of 6 axis mechanisms, they may be of two types, either spherical wrist (i.e. last three axes intersect at a single point) or they can be of a general configuration. The spherical wrist robot has a trivial inverse kinematics solution, which we present in appendix A. However, the inverse kinematics solution of a non spherical wrist mechanism is far harder to solve, and is very computationally expensive. Thus an attempt is made at using the forward kinematics equations to generate data, and then to train a neural network to learn the inverse kinematics mapping. A challenge is that a six axis general manipulator has 16 possible solutions [17]. For an idea of what the different solutions look like please see A.2, which shows the 8 different possible solutions in the case of a spherical wrist manipulator. Given the inverse kinematics problem has different solutions, a different neural network must be trained for each solution. However, when we use the forward kinematics equations to randomly produce the data, we need to separate the data into the different solutions. Another type of machine learning is considered which is clustering. The type of clustering considered is K-means. This type of clustering works out for us, because we only need to specify the number of clusters, which is 16 in our case, and the learning algorithm will group the data automatically. K-means clustering is a type of non supervised learning, because we have no labels for any of the configurations. We only know that certain configurations are different from others. Given that different inverse kinematics solutions are situations in which the world position is the same, but joint values are very different, as can be seen in A.2, the clustering problem attempts to separate the data based on the joint and world position values. Thus, different data points that have very similar joint and world position values will belong to the same configuration, while those that have different world and joint position values will belong to different configurations.

Neural Networks for Dynamics

Robot control is usually divided into two categories, either MIMO (Multiple Input Multiple Output) control [27], or SISO (Single Input Single Output) control. Traditionally, in MIMO control, a single controller generates multiple control efforts (torque/force) to all the robot. On the other hand, SISO control involves separate controllers generating control efforts to each axis to set the output at the target value (usually position).

Yaskawa servos run at high internal update rates. Thus, the plan for this research is to rely on the Yaskawa servos for closing the position control loop (ie. SISO control), yet the controller will assist the servos by providing velocity and torque feedforward estimates.

Accurate velocity and torque feedforward estimates will improve the trajectory following performance of the servos.

A dynamics model is critical for accurate torque estimation. A dynamics model of a system can be expressed as $\tau = f(p, v, a)$, where τ, p, v and a are the torque, position, velocity and acceleration of the joints respectively. f is normally a nonlinear function, and each variable can be a vector of arbitrary length. For the purpose of this project, research will be done on 2 and 3 DOF Delta robots and a 6 axis robot which are all provided by Yaskawa.

The purpose of this project is to find the function f, not by dynamics modeling, which usually suffers from inaccuracies due to simplifications, but by training a neural network on data generated on the actual mechanism.

Once a neural network is trained, its accuracy can be evaluated by how closely it estimates joint torque efforts for a given position, velocity, and acceleration of the joints. This is done by comparing the neural network torque estimate values against actual torque feedback values.

SISO Control with Velocity and Torque Feedforward

Feedforward control can greatly improve the performance of a control system [28]. As such, Yaskawa servo drives which support feedforward control will be used to control each axis on the mechanisms. Each servo will implement the control loops seen in figure 1.1. The controller will be responsible for generating the values p_{ff}, v_{ff}, and t_{ff} in this diagram which are the quantities for the target position, and velocity and torque feedforward values respectively.

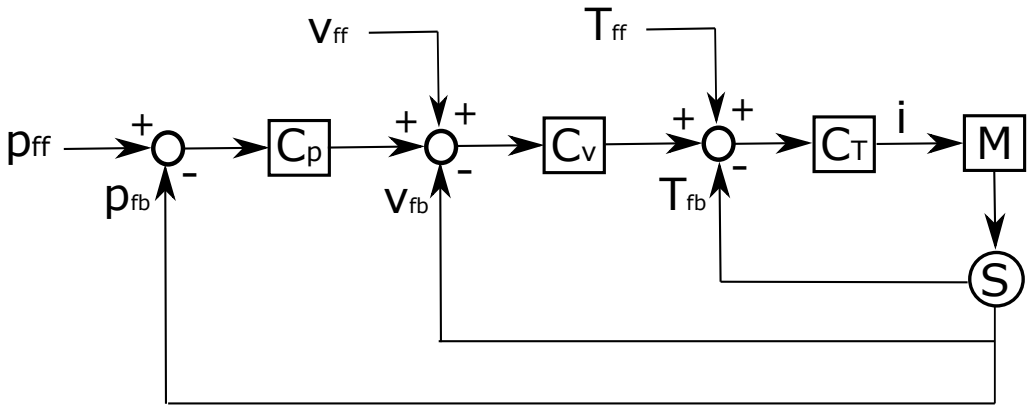


Figure 1.1: Control schematic

1.3 Neural Networks and the Back Propagation Algorithm

The neural network presented in figure 1.2 has only one hidden layer, and is based on using the sigmoid function on both matrices. However, this is just one example. Research will involve using various numbers of hidden layers with various dimensions. Additionally, research will explore other functions besides the sigmoid function such as the tanh function.

Details about the neural network and the back propagation algorithm [33] for updating the matrices during learning are given below:

Note that in our case the input variable x will contain the position, velocity, and acceleration of all the joints, while z will contain the predicted torque values. We also note that the cost function has this particular logarithmic form, because it is well suited for the sigmoid function. As we will see shortly, this cost function simplifies the derivation of the gradients.

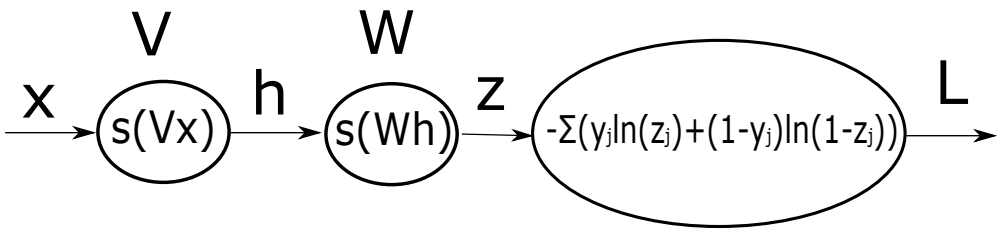


Figure 1.2: Neural network model

x is the input variable.
y is the actual output variable.
z is the predicted output variable.
d is the dimension of the input variable x.
k is the dimension of the output variable y.
n is the number of training samples.

X is d x n and is the input training data.
Y is k x n and is the output training data.
x is (d+1) x 1 and is the input layer.
h is (m+1) x 1 and is the hidden layer.
z is k x 1 and is the output layer.
L is 1 x 1 and is the cost function.
z is the predicted value for a given input x.
y is the actual value for a given input x.

V is (m+1) x (d+1) and maps the inputs to the hidden layer.
W is k x (m+1) and maps the hidden layer to the outputs.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_d \\ 1 \end{bmatrix}, h = \begin{bmatrix} h_1 \\ h_2 \\ . \\ . \\ h_m \\ 1 \end{bmatrix}, z = \begin{bmatrix} z_1 \\ z_2 \\ . \\ . \\ . \\ z_k \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_k \end{bmatrix}, V = \begin{bmatrix} V_1 \\ V_2 \\ . \\ . \\ V_m \\ V_{m+1} \end{bmatrix}, W = \begin{bmatrix} W_1 \\ W_2 \\ . \\ . \\ . \\ W_k \end{bmatrix}$$

$$s(\gamma) = \frac{1}{1+e^{-\gamma}}, \qquad s'(\gamma) = s(1-s), \qquad h = s(Vx), \qquad z = s(Wh)$$

$$\frac{\partial L}{\partial z_j} = -\left(\frac{y_j}{z_j} - \frac{1-y_j}{1-z_j}\right) = -\frac{y_j - y_j z_j - z_j + z_j y_j}{z_j(1-z_j)} = \frac{z_j - y_j}{z_j(1-z_j)}$$

$$\nabla_{W_j} L = \frac{\partial L}{\partial z_j} \nabla_{W_j} z_j = \frac{\partial L}{\partial z_j} z_j(1-z_j) h^T = (z_j - y_j) h^T$$

$$\nabla_W L = (z - y)h^T$$

$$\nabla_h L = \Sigma \frac{\partial L}{\partial z_j} \nabla_h z_j = \Sigma \frac{\partial L}{\partial z_j} z_j(1 - z_j) W_j^T = \Sigma W_j^T (z_j - y_j) = W^T(z - y)$$

$$\nabla_{V_i} L = \frac{\partial L}{\partial h_i} \nabla_{V_i} h_i = \frac{\partial L}{\partial h_i} h_i(1 - h_i) x^T$$

$$\nabla_V L = W^T(z - y) * h * (\mathbf{1} - h) x^T$$

Note that * represents element wise multiplication. Additionally, **1** is a vector of ones of appropriate size. Also note that canceling the term $z_j(1 - z_j)$ in the numerator and denominator of the derivation above is safe, because z_j is the output of a sigmoid function, and thus we have $z_j \neq 0$ and $z_j \neq 1$.

Finally, learning is performed using the update rule below for gradient descend:

$$W = W - \epsilon \nabla_W L$$

$$V = V - \epsilon \nabla_V L$$

where ϵ is the learning rate.

1.4 Six Axis Robot System

The six axis robot system was entirely composed of Yaskawa products. The diagram in figure 1.3 shows the connectivity between the various components. The control panel can be seen in figure 1.4. Table 1.1 contains all the joint specification and setup information. The complete system consisted of the following:

- Yaskawa MP3300iec PLC (Programmable Logic Controller)
- 6 Yaskawa Sigma 7 Mechatronlink Servos
- Yaskawa Motoman MHJ six axis arm
- Windows PC

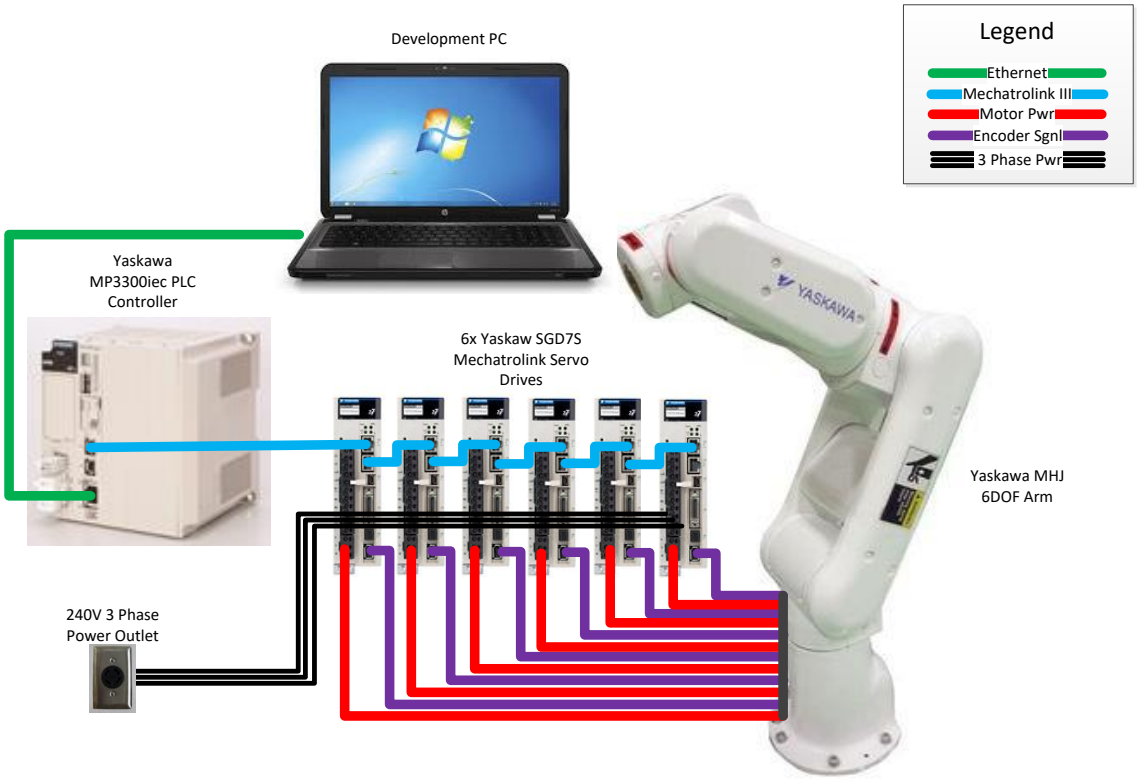


Figure 1.3: Six axis system diagram

Figure 1.4: Six axis robot control panel

	Joint 0	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5
Motor Rated Torque (N.m)	0.159	0.159	0.159	0.159	0.159	0.159
Motor Rated Speed (RPM)	3000	3000	3000	3000	3000	3000
Motor Power (Watts)	50	50	50	50	50	50
Encoder Resolution (tics/rev)	2^{17}	2^{17}	2^{17}	2^{17}	2^{17}	2^{17}
Gear Ratio	1:120	1:275	3:400	1:120	1:90	1:50
Polarity	Neg	Neg	Neg	Pos	Neg	Pos
Zero Position Offset (tics)	249536	74560	618464	-38176	73696	-520320
Min limit (Degrees)	-160	-90	-45	-180	-130	-180
Max limit (Degrees)	160	110	210	180	130	180

Table 1.1: MHJ robot joint specification and setup

Chapter 2

Six Axis Kinematics

2.1 Analytical Forward and Inverse Kinematics Derivation

The first requirement for robot control is to obtain the solutions for the forward and inverse kinematics problems. The six axis robot used for experiments in this study is the Motoman MHJ [20] seen in Figure 2.1 This robot is a spherical wrist robot, meaning that the last three axes of the robot intersect at a single point. This property makes the inverse kinematics solution much easier to solve. The complete forward and inverse kinematics derivation can be found in appendix A. This derivation closely follows the techniques used in [32]. The kinematics model used for this class of six axis robots can be seen in figure A.1. Additionally, table A.1 contains the Denavit Hartenberg parameters [8]. The Euler angle convention followed for three dimensional rotations is the commonly used Roll-Pitch-Yaw convention. This convention is defined by various texts [32] and [29] in the same manner.

2.2 Machine Learning Inverse Kinematics solution

As previously outlined in section 1.2, the proposed plan for solving the inverse kinematics solution for a general serial manipulator is as follows:

1. Generate a random mechanism by obtaining random values for the DH parameters α, a, and d for each of the six joints in table A.1
2. Solve forward kinematics equations

Figure 2.1: Yaskawa Motoman MHJ robot

3. Randomly generate joint and world position values using the forward kinematics equations

4. Use K-means clustering to separate the data into the different solutions

5. Train a different neural network on each clustered dataset to learn the inverse kinematics solution

2.3 Forward Kinematics for a Generalized Manipulator

The forward kinematics solution was obtained using the Denavit Hartenberg convention. Equations A.1 and A.11 work for any serial manipulator with six revolute joints even if it is not a spherical wrist mechanism. We note that for a general mechanism that is of non spherical wrist, none of the Denavit Hartenberg parameters in table

A.1 will be zero. However, table A.1 contains many zeros, because it is for the model in figure A.1 which is a spherical wrist robot.

2.4 K-means Clustering

K-means clustering [12] is a commonly used unsupervised learning algorithm. Its purpose is to separate a set of data points into a given number of clusters based on the distances between the data points. The method chosen for clustering is random initialization and iterative update using the Lloyd-Forgy algorithm [1]. The initial step is to assign the data points to random clusters given the number of clusters to be formed. The next step is to iteratively update the mean of each cluster and later update which cluster each data point belongs to. The sequence is below:

1. Calculate the centroid of each cluster using the mean of all the data points in that cluster

2. Reassign each data point to the cluster whose centroid has the least euclidean distance to that data point

The algorithm above is repeated until the solution converges. Convergence is reached when no more data points can be reassigned to different clusters.

2.5 Neural Network Training of Each Kinematic Solution

Given the complexity of the inverse kinematics solution, we have chosen neural networks for the regression algorithm. The input to the neural network in this case is the world position of the end effector of the robot, while the output is the joint positions of the robot. We note that a different neural network is trained for each inverse kinematics solution according to different clusters that were previously obtained. As usual, training is done on 80 percent of the data, while 20 percent of the data is reserved for validation after training is complete. The accuracy of the inverse kinematics solution is evaluated based on how close the joint position output values are to the real solution.

2.6 Challenges and Simplifications

Given the joint position vector is a 6x1 vector and given the world position vector is also a 6x1 vector, the clustering problem presented here attempts to separate data in a 12 dimensional space. This is a very high order system, which means an extremely large amount of data is needed to yield accurate results. In addition, a large amount of data would mean a long runtime until the clustering of the data converges. Moreover, convergence is not guaranteed. In many occasions convergence is never achieved.

Given the analytical solution for the spherical wrist robot is already solved in appendix A, we thus do know how to separate the kinematics solutions into their different configurations, based on the joint values. Thus we choose to use the spherical wrist model for clustering, because we are able to validate the performance of our clustering solution. We note that in the spherical wrist problem, there are 8 solutions, thus each solution is of configuration 0 through 7.

2.7 Identifying Robot Configurations from Joint Positions

The inverse kinematics solutions given in appendix A easily allow us to choose the solution for the joint position given the configuration we want. However, the opposite problem is to identify the current robot configuration given a robot joint position. This problem can be solved by analyzing figure A.2 and the multiple solutions for θ_0, θ_2, and θ_4.

The equations for the solutions of θ_0 are A.23, A.24, and A.25. The multiple solutions in θ_0 are called the Front Reach and Back Reach configurations. Front Reach is when the difference between θ_0 and ϕ_0 is zero, while Back Reach is when the difference between θ_0 and ϕ_0 is π. Given that θ_0 is the angle of joint L relative to the origin, and given ϕ is the angle of joint B relative to the origin, we now know that if joints L and B are on the same side relative to the origin, then we are in Front Reach, otherwise we are in Back Reach configuration. The forward kinematics equations can give us the x and y coordinates of joints L and B as follows:

$$\begin{bmatrix} p_{Lx} \\ p_{Ly} \end{bmatrix} = \begin{bmatrix} a_0 cos\theta_0 \\ a_0 sin\theta_0 \end{bmatrix} \tag{2.1}$$

$$\begin{bmatrix} p_{Bx} \\ p_{By} \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} cos(\theta_0)(a_0 + a_1 cos(\theta_1) + a_2 cos(\theta_{2-1}) - d_3 sin(\theta_{2-1})) \\ sin(\theta_0)(a_0 + a_1 cos(\theta_1) + a_2 cos(\theta_{2-1}) - d_3 sin(\theta_{2-1})) \end{bmatrix} \tag{2.2}$$

By inspecting the equations for the forward kinematics in x and y coordinates for joints L and B above, and given the term a_0 is always positive, we can see that the term $(a_0+a_1cos(\theta_1)+a_2cos(\theta_{2-1})-d_3sin(\theta_{2-1})$ is responsible for making joint B have an angle with respect to the origin opposite from that of joint L's angle with respect to the origin. Thus, we conclude that if the term $(a_0 + a_1cos(\theta_1) + a_2cos(\theta_{2-1}) - d_3sin(\theta_{2-1})$ is positive then we are in front reach, otherwise we are in Back Reach.

As for the dual solution in θ_2, when the angle formed between joints L, U and B is smaller than 180 degrees then we are in Upper Arm configuration, otherwise we are in Lower Arm configuration. We note that when the angle of joints L, U and B is 180 degrees then the robot arm is positioned for maximum reach. We note that geometrically one can easily obtain that $\theta_{2MaxReach} = arctan(a_2/d_3) + \frac{\pi}{2}$. Thus if $\theta_2 < arctan(a_2/d_3) + \frac{\pi}{2}$ then we are in Upper Arm configuration, otherwise we are in Lower Arm configuration.

As for the dual solution in θ_4, which is given in equation A.66, when $\theta_4 < 0$ we are in No Flip configuration, otherwise we are in Flip configuration. This is the case, because the function cos^{-1} always returns a positive angle.

In summary, the configuration flags for a spherical wrist robot are given as:

- if $(a_0 + a_1cos(\theta_1) + a_2cos(\theta_{2-1}) - d_3sin(\theta_{2-1}) > 0$ then Front Reach, otherwise Back Reach
- if $\theta_2 < arctan(a_2/d_3) + \frac{\pi}{2}$ then Upper Arm, otherwise Lower Arm
- if $\theta_4 < 0$ then No Flip, otherwise Flip

Given we have 2 possibilities for each configuration, we obtain a total of $2*2*2 = 8$ solutions.

2.8 Machine Learning for Inverse Kinematics Results

One million data points were generated using the forward kinematics equations for the spherical wrist robot. The K-means clustering algorithm never converged on the dataset, and it wasn't possible to find a mapping between the clusters assigned at the end of the training and joint configurations obtained analytically. A workaround for this problem was that we collected data for only configuration 0, and trained various types of machine learning algorithms to learn the inverse kinematics solution. Two possible approaches to training were identified, the first was to train with only one machine learning algorithm to find all the joint values, and the second was to train a

Algorithm	All joints
Linear regression	0.13
10 Nearest neighbors	0.48
Neural network	0.56

Table 2.1: R^2 accuracy of regression: Training one machine learning algorithm for all 6 joints simultaneously

Algorithm	θ_0	θ_1	θ_2	θ_3	θ_4	θ_5
Linear regression	0.39	0.28	0	0	0	0
10 Nearest neighbors	0.73	0.33	0.70	0.50	0.42	0.20
Neural network	0.98	0.33	0.76	0.75	0.47	0.30

Table 2.2: R^2 accuracy of regression: Training separate machine learning algorithms without propagation

Algorithm	θ_0	θ_1	θ_2	θ_3	θ_4	θ_5
Linear regression	0.39	0.37	0	0	0	0
10 Nearest neighbors	0.73	0.64	0.70	0.65	0.98	0.59
Neural network	0.99	0.94	0.77	0.87	0.99	0.80

Table 2.3: R^2 accuracy of regression: Training separate machine learning algorithms with propagation

separate machine learning algorithm for each joint. The results are shown in tables 2.1, 2.2, and 2.3. Three types of machine learning algorithms were tested, which are linear regression, 10 nearest neighbors, and neural networks. We note that as demonstrated in [2] using elements of the end effector rotation matrix as inputs to the neural network outperforms using Euler angles as inputs. This however should come as no surprise given the analytical inverse kinematics solution is in fact a function of elements of the rotation matrix, and not the Euler angles directly.

We note that the results that used propagation are based on feeding in solutions for joint values into neural networks of subsequent joints. The dependencies of the joint solutions to one another were obtained from the analytical solution to the inverse kinematics problem in appendix A. The map of these dependencies can be seen in figure 2.2.

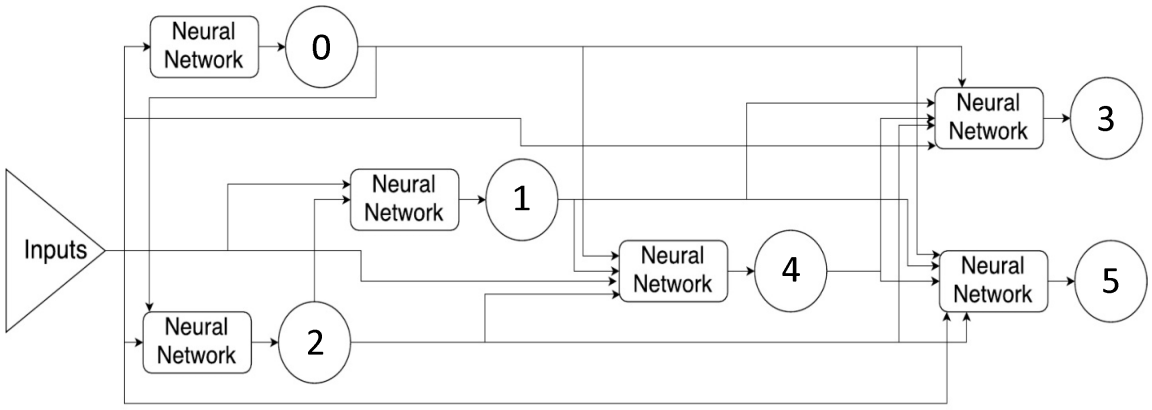


Figure 2.2: Neural network solution propagation

As evident from the results, the best performance was from the neural network, and in particular when solution propagation was used.

The parameters used for the neural networks were:

- Number of layers: 2-4
- Hidden layer size: 43
- Activation function: Logistic
- Step Size: 0.01
- Batch Size: 200-300
- Maximum Number of iterations: 1,000,000

Chapter 3

Six Axis Dynamics

3.1 Analytical Dynamics Derivation

In this section, we will go over the analytical derivation of a simplified six axis dynamics model. The model is reduced to a 3 DOF serial manipulator [21]. The fact that the robot is a spherical wrist mechanism does not only simplify the kinematics derivation, but it also simplifies the dynamics model. The dynamics model presented in figure 3.1 is for obtaining the torque estimates of the first three joints and makes the following assumptions:

- a_0 and a_2 in figure A.1 are zero, note that this is the case for an MHJ robot
- Rotations in T joint have negligible effects on inertia of upper limb. This is an accurate assumption given the T joint has negligible inertia.
- Rotations in R and B joints have negligible effects on the inertia of the upper limb. This is an accurate assumption given this robot is normally targeted for pick and place applications such that there are no rotations in the R joint, and rotations in the B joint are only done in order to keep the end effector oriented down.
- Robot links are approximated as uniform rods of negligible diameter
- Rated payload on an MHJ (1 Kg) is negligible relative to the weight of the MHJ (15 Kg)

An important note about figure 3.1 is that $-\theta_1$ is simply the negative of θ_1 which is from the DH parameters in table A.1. The dynamics equations can be obtained using Lagrange's equation of motion [15].

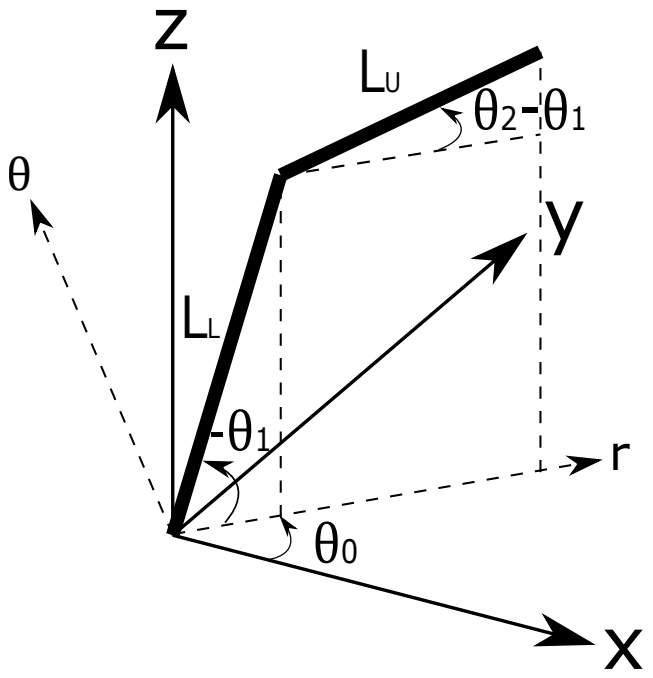


Figure 3.1: MHJ dynamics model

Lagrangian:

$$L(\theta, \dot{\theta}) = KE(\theta, \dot{\theta}) - PE(\theta) \tag{3.1}$$

Lagrangian says:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_i} - \frac{\partial L}{\partial \theta_i} = \tau_i - sign(\dot{\theta}_i)Fr_i \tag{3.2}$$

Kinetic energy:

$$KE(\theta, \dot{\theta}) = \frac{1}{2}\dot{\theta}^T M(\theta)\dot{\theta} = \Sigma(KE_{tran}) + \Sigma(KE_{rot}) \tag{3.3}$$

Potential energy:

$$PE(\theta) = m^T h(\theta) g \tag{3.4}$$

Dynamics equation:

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G(\theta) + sign(\dot{\theta})Fr = \tau \tag{3.5}$$

Below is a definition of terms:

- L_L : Length of Lower link
- L_U : Length of Upper link
- θ : 3x1 vector of joint angles $[\theta_0, \theta_1, \theta_2]$
- $\dot{\theta}$: 3x1 vector of joint velocities $[\dot{\theta_0}, \dot{\theta_1}, \dot{\theta_2}]^T$
- $\ddot{\theta}$: 3x1 vector of joint accelerations $[\ddot{\theta_0}, \ddot{\theta_1}, \ddot{\theta_2}]^T$
- $M(\theta)$: 3x3 Inertia matrix
- $C(\theta, \dot{\theta})$: 3x3 Coriolis matrix
- $G(\theta)$: 3x1 Gravitational forces
- Fr : 3x1 Dynamic friction forces $[Fr_0, Fr_1, Fr_2]^2$
- m : 2x1 vector of link masses $[m_L, m_U]^T$
- $h(\theta)$: 2x1 vector of height of links $[h_L, h_U]^T$
- g : Acceleration due to gravity
- $KE(\theta, \dot{\theta})$: Kinetic Energy
- $PE(\theta)$: Potential Energy
- $L(\theta, \dot{\theta})$: The Lagrange of the system
- KE_{tran} : Translational Kinetic Energy of the center of mass
- KE_{rot} : Rotational Kinetic Energy about the center of mass
- τ : 3x1 vector of joint torques $[\tau_0, \tau_1, \tau_2]$

$$\begin{aligned} PE(\theta) &= m_L h_L g + m_U h_U g \\ &= -m_L \frac{L_L}{2} s\theta_1 g + m_U(-L_L s\theta_1 + \frac{L_U}{2} s\theta_{2-1})g \end{aligned} \tag{3.6}$$

$$\begin{aligned} KE_{tranL}(\theta, \dot{\theta}) &= \frac{1}{2} m_L(\dot{x}_{cmL}^2 + \dot{y}_{cmL}^2 + \dot{z}_{cmL}^2) \\ &= \frac{m_L L_L^2}{8}(c^2\theta_1 \dot{\theta_0}^2 + \dot{\theta_1}^2) \end{aligned} \tag{3.7}$$

$$\begin{aligned} KE_{tranU}(\theta,\dot{\theta}) &= \frac{1}{2}m_U(\dot{x}_{cmU}^2+\dot{y}_{cmU}^2+\dot{z}_{cmU}^2) \\ &= \frac{m_U}{2}\Bigg(L_L^2\dot{\theta_1}^2+\frac{L_U^2}{4}(\dot{\theta_2}-\dot{\theta_1})^2-L_LL_Uc\theta_2(\dot{\theta_2}-\dot{\theta_1})\dot{\theta_1} \\ &\quad +\left(L_Lc\theta_1+\frac{L_U}{2}c\theta_{2-1}\right)^2\dot{\theta_0}^2\Bigg) \end{aligned} \tag{3.8}$$

$$KE_{rotL}(\theta,\dot{\theta}) = \frac{1}{2}\omega_L^TI_L\omega_L = \frac{m_LL_L^2}{24}(\dot{\theta_1}^2+c^2\theta_1\dot{\theta_0}^2) \tag{3.9}$$

$$KE_{rotU}(\theta,\dot{\theta}) = \frac{1}{2}\omega_U^TI_U\omega_U = \frac{m_UL_U^2}{24}((\dot{\theta_2}-\dot{\theta_1})^2+c^2\theta_{2-1}\dot{\theta_0}^2) \tag{3.10}$$

Substituting equations 3.6, 3.7, 3.8, 3.9, and 3.10 into 3.1 and 3.2 then combining with equation 3.5 yields the terms of the Matrices $M(\theta)$ and $C(\theta,\dot{\theta})$, and the vector $G(\theta)$ as follows:

$$M_{11} = \frac{m_LL_L^2}{3}c^2\theta_1+\frac{m_UL_U^2}{3}c^2\theta_{2-1}+m_u(L_L^2c^2\theta_1+L_LL_Uc\theta_1c\theta_{2-1})$$

$$M_{12} = M_{13} = M_{21} = M_{31} = 0$$

$$M_{22} = \frac{m_LL_L^2}{3}+\frac{m_UL_U^2}{3}+m_UL_L^2+m_UL_LL_Uc\theta_2$$

$$M_{23} = M_{32} = -\frac{m_UL_U^2}{3}-\frac{m_UL_LL_U}{2}c\theta_2$$

$$M_{33} = \frac{m_UL_U^2}{3}$$

Given that for our purposes the the individual terms of the matrix $C(\theta,\dot{\theta})$ are not important, we shall define $C^*(\theta,\dot{\theta}) = C(\theta,\dot{\theta})\theta$. Thus $C^*(\theta,\dot{\theta})$ is now a 3x1 vector:

$$C_1^* = -2\frac{m_L L_L^2}{3} c\theta_1 s\theta_1 \dot{\theta}_1 \dot{\theta}_0 - 2\frac{m_U L_U^2}{3} c\theta_{2-1} s\theta_{2-1} (\dot{\theta}_2 - \dot{\theta}_1)\dot{\theta}_0$$
$$- 2m_u \left(L_L^2 c\theta_1 s\theta_1 \dot{\theta}_1 + \frac{L_L L_U}{2} \left(c\theta_1 s\theta_{2-1} (\dot{\theta}_2 - \dot{\theta}_1) + s\theta_1 c\theta_{2-1} \dot{\theta}_1 \right) \right) \dot{\theta}_0$$

$$C_2^* = \frac{m_L L_L^2}{3} c\theta_1 s\theta_1 \dot{\theta_0}^2 - \frac{m_U L_U^2}{3} c\theta_{2-1} s\theta_{2-1} \dot{\theta_0}^2 + m_U L_L L_U s\theta_2 \dot{\theta}_2 \left(\frac{\dot{\theta}_2}{2} - \dot{\theta}_1 \right)$$
$$+ m_u \left(L_L^2 c\theta_1 s\theta_1 + \frac{L_L L_U}{2} (-c\theta_1 s\theta_{2-1} + s\theta_1 c\theta_{2-1}) \right) \dot{\theta_0}^2$$

$$C_3^* = \frac{m_U L_L L_U}{2} s\theta_2 \dot{\theta_1}^2 + \frac{m_U L_U^2}{3} c\theta_{2-1} s\theta_{2-1} \dot{\theta_0}^2 + \frac{m_U L_L L_U}{2} c\theta_1 s\theta_{2-1} \dot{\theta_0}^2$$

$$G_1 = 0$$

$$G_2 = -\left(m_L \frac{L_L}{2} + m_U L_L \right) c\theta_1 g - m_U \frac{L_U}{2} c\theta_{2-1} g$$

$$G_3 = m_U \frac{L_U}{2} c\theta_{2-1} g$$

3.2 Machine Learning Dynamics

As evident in the dynamics model derived in section 3.1, many simplifications had to be made in order to make the dynamics model solvable without excessive complexity. Specifically, parameters for the inertia of the links as well as the frictional terms can be difficult to obtain accurately. Thus in this section we propose a machine learning approach to obtaining the dynamics equations. The strategy is to move the actual robot through random movements in space, and to collect feedback data for positions, velocities, accelerations, and torques from each joint. After this data is collected, a neural network can be trained on the position, velocity, and acceleration data as inputs and the torque data as outputs. This is consistent with the dynamics equation given in 3.5. Given the data is generated on the actual mechanism, frictional terms will automatically be accounted for by the neural network.

3.3 Machine Learning Dynamics Offline Training

As a first step, the offline training approach was used. Specifically, we move the robot, collect the data, and then train the neural network. Given the neural network is targeted to run on an embedded real time controller, emphasis was made on the size of the neural network, such that it does not consume too much time to run on the controller. Thus, various sizes of neural networks were tested, and a trade off was seen between neural network runtime and neural network accuracy. Not surprisingly, a very large neural network can be very accurate, yet it will take a very long time to both train and/or run. On the other hand, a very small neural network can be very fast, yet it will be inaccurate. Figure 3.2 shows how the accuracy of the neural network eventually plateaus even if the size of the hidden layer keeps being increased. We note that we only used one hidden layer.

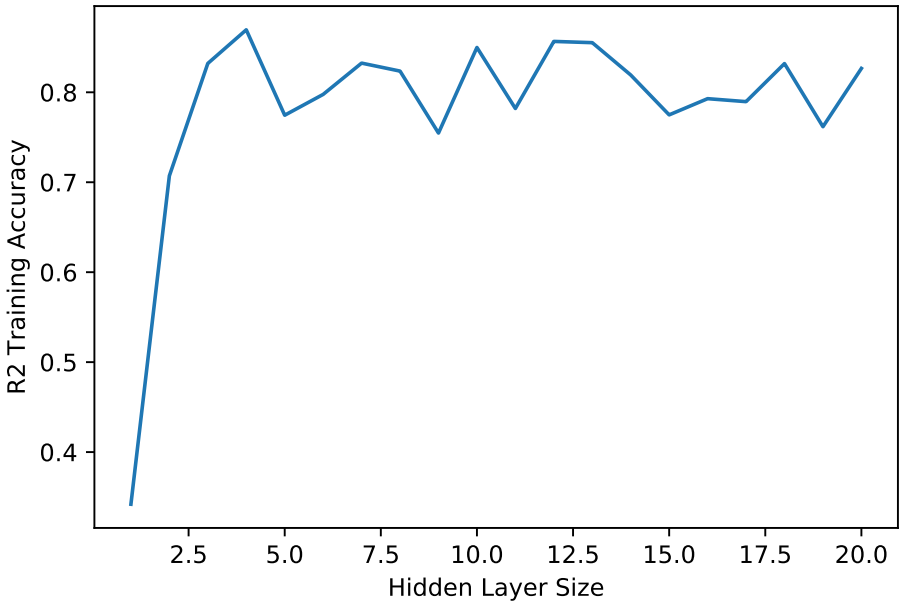


Figure 3.2: Neural network accuracy vs hidden layer size

Figure 3.3 shows the estimated computational runtime of the neural network on the 1.2 GHz processor being used in our case. As the figure shows, runtime is fairly linear with the size of the hidden layer. This is expected, because as figure 1.2 shows, x and z which are input and output prediction values respectively are constant in size, while h is varied. As h is varied each of the weight matrices V and W will

grow linearly in h, thus the amount of matrix multiplication operations as well as additions will also grow linearly.

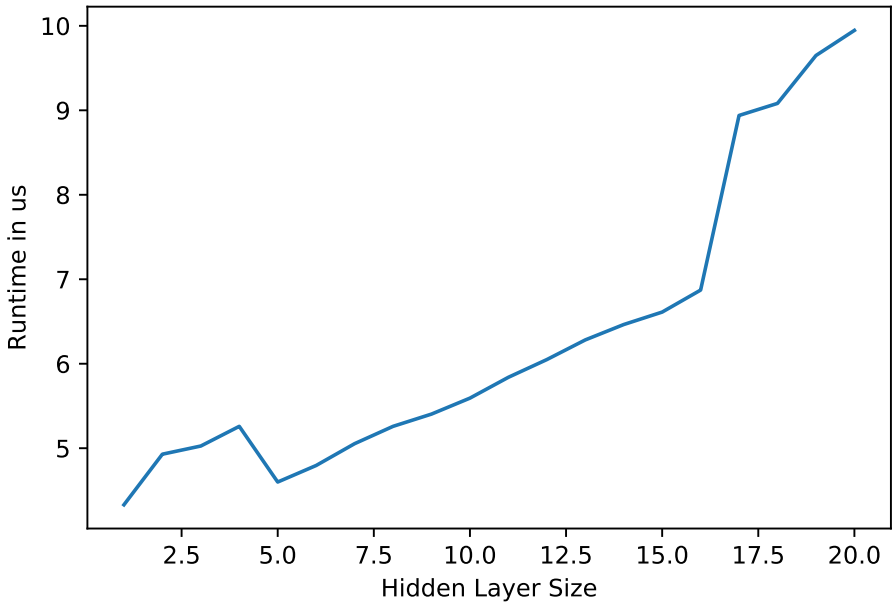


Figure 3.3: Neural network runtime vs hidden layer size

Data Acquisition

Training and validation data on the MHJ was collected by moving the robot along each edge of a prism back and forth, left and right, and up and down. Motion was done, while the end effector orientation was constantly pointing vertically downwards. We note that this movement results in motion in all the joints except joint 4 on the robot. This was done, because emphasis in this work is placed on pick and place applications where the end effector orientation will remain pointed vertically downwards. The dimensions of the prism are given in table 3.1. The data collected was only for the first three joints, because as previously discussed in section 3.1, these joints have the largest effects on the inertial, Coriolis, and gravitational forces on the system. Position, velocity, acceleration, and torque data was collected for each joint. Inputs for the neural network were the position, velocity, and acceleration of the three joints, and output for the neural network was the torque for each joint.

Training limits	x	y	z
min [mm]	270	-100	150
max [mm]	330	100	250

Table 3.1: MHJ Cartesian coordinate training limits

As usual, four fifth of the data was split for training and one fifth was reserved for validation. The size of the entire data set was about 10,000 points. An important step was that only the training data was shuffled. This makes testing of the validation data more representative of the true performance, because we will be verifying a complete movement profile as opposed to discrete random sample points from the entire motion profile.

Single Neural Network for all Joints Results

Plots of the actual and predicted torque values for the first three joints when the hidden layer is of size six can be seen in figure 3.4. In this scenario one neural network was used for all the joints. The R2 accuracy obtained was 0.75

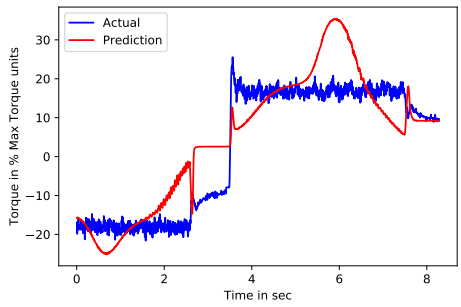
Separate Neural Networks with Independent Variable Elimination Results

In the previous section every single joint's position, velocity, and acceleration was used as an input, and all the torques were used as outputs. This however can confuse the neural network, because as previously derived in section 3.1, the torque output is not always a function of all the variables. Upon a closer inspection of the terms in the inertial, Coriolis, gravitational, and frictional forces obtained in section 3.1, we can infer the dependencies below:

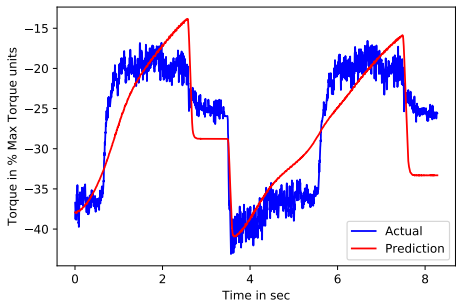
We note that although $\dot{\theta_2}$ does not appear in the inertial, Coriolis, or gravitational terms for τ_2, it does appear in the frictional term.

$$\tau_0 = f(\theta_1, \theta_2, \dot{\theta_0}, \dot{\theta_1}, \dot{\theta_2}, \ddot{\theta_0})$$

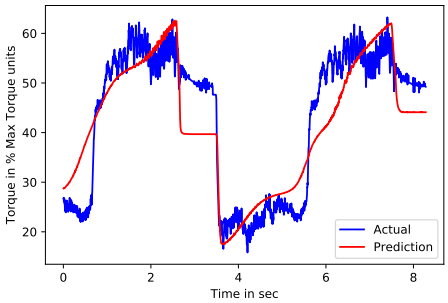
$$\tau_1 = f(\theta_1, \theta_2, \dot{\theta_0}, \dot{\theta_1}, \dot{\theta_2}, \ddot{\theta_1}, \ddot{\theta_2})$$



(a) Joint 0



(b) Joint 1



(c) Joint 2

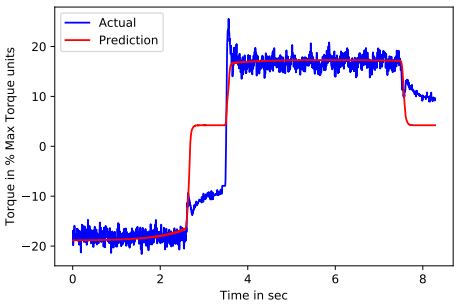
Figure 3.4: Actual and predicted torque values on MHJ using one neural network

$$\tau_2 = f(\theta_1, \theta_2, \dot{\theta_0}, \dot{\theta_1}, \dot{\theta_2}, \ddot{\theta_1}, \ddot{\theta_2})$$

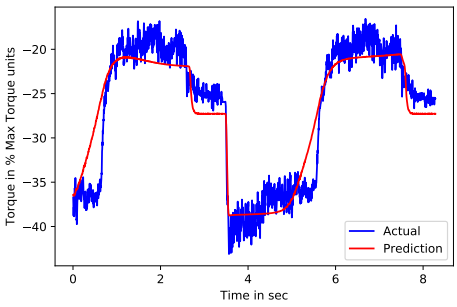
By eliminating independent variables and training a separate neural network for each joint, we obtain the actual and predicted torque values in figure 3.5. The R2 accuracies obtained for the first three joints were 0.90, .88, 0.87 respectively.

Neural Network Parameters

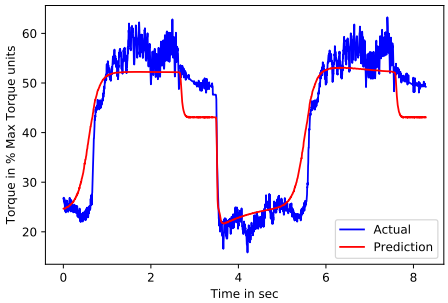
All neural networks used had a hidden layer of size 6, and used a sigmoid activation function. As demonstrated in [11], the inputs were scaled to a mean of zero and standard deviation of 1, and a decaying learning rate was used. The decay in the learning rate protects against over fitting.



(a) Joint 0



(b) Joint 1



(c) Joint 2

Figure 3.5: Actual and predicted torque values on MHJ using separate neural networks with independent variable elimination

Discussion

Clearly, using separate neural networks for each joint with independent variable elimination outperformed using only one neural network for all the joints. Additionally, runtime below 10 μs was achievable with neural networks that resulted in an accuracy of approximately .90 for each joint. These results are very suitable for an embedded PLC controller, and can greatly improve performance. Our assumption that movement in the last three joints has negligible effects on the torques of the first three joints appears to have been a reasonable assumption given the positive results we obtained without using them.

Chapter 4

Delta3

4.1 Delta3 Used for Experiments

The Codian D4-650 was used for experiments [26]. It can be seen in figure 4.1. The control panel can be seen in figure 4.2. Table 4.1 contains all the joint specification and setup information. This mechanism has 3 axes for 3 DOF in x, y, and z Cartesian space. Additionally, it has a fourth axis for rotating the end effector. This class of robots was invented by Reymond Clavel [4], and is normally called a Delta robot.

	Joint 0	Joint 1	Joint 2	Joint 3
Motor Rated Torque (N.m)	4.9	4.9	4.9	1.27
Motor Rated Speed (RPM)	3000	3000	3000	3000
Motor Power (Watts)	1500	1500	1500	400
Encoder Resolution (tics/rev)	2^{24}	2^{24}	2^{24}	2^{24}
Gear Ratio	2:77	2:77	2:77	1:10
Polarity	Neg	Neg	Neg	Pos
Zero Position Offset (tics)	249536	74560	618464	-38176
Min limit (Degrees)	-30	-30	-30	$-\infty$
Max limit (Degrees)	88	88	88	∞

Table 4.1: D4-650 robot joint specification and setup

Figure 4.1: Codian D4-650

4.2 Analytical Kinematics

A model and an analytical solution for the position, velocity, and acceleration forward and inverse kinematics solutions can be found in appendix B. The kinematics derivation uses the loop closure equations in vector form. Note that the derivation ignores the fourth axis on the D4-650, because it is merely a trivial rotation of the end effector.

4.3 Machine Learning Dynamics

In our dynamics analysis, we will be ignoring the fourth axis on the system used for rotating the end effector, because its dynamics are decoupled from the rest of the

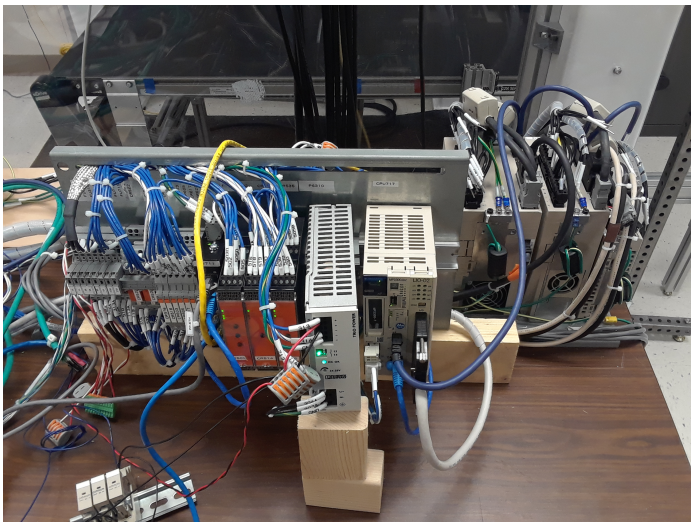


Figure 4.2: Codian D4-650 Control Panel

Training limits	x	y	z
min [mm]	-400	-400	900
max [mm]	400	400	1100

Table 4.2: Delta3 Cartesian coordinate training limits

system.

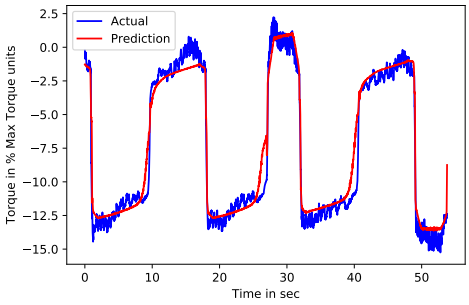
Given a Delta3 mechanism is a parallel manipulator and dynamics of the first three joints are highly coupled, it is safe to a assume that unlike the case with the six axis arm, torques of the first three joints are dependent on the position, velocity, and acceleration of each of the first three joints. Thus no Delta3 analytical dynamics derivation is presented in this work.

Data Acquisition

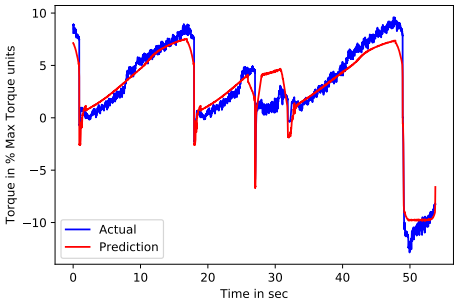
Similar to what was done with the six axis robot, the Delta3 robot was moved within a prism of dimensions found in table 4.2. Additionally, the fourth axis which rotates the end effector was not used. The number of data points was approximately 67,000.

Single Neural Network for all Joints Results

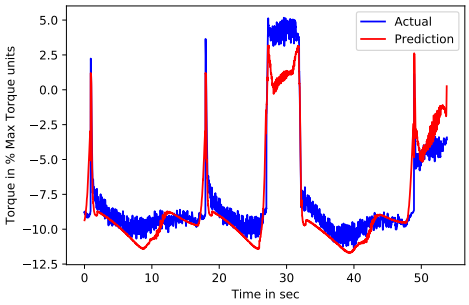
Plots of the actual and predicted torque values for the first three joint when the hidden layer is of size six can be seen in figure 4.3. In this scenario one neural network was used for all the joints. The R2 accuracy obtained was 0.93



(a) Joint 0



(b) Joint 1



(c) Joint 2

Figure 4.3: Actual and predicted torque values on D4-650 using one neural network

Discussion

The performance of the neural network is clearly very good. The predicted values followed the actual values very closely. We would also like to note that just as in the case of the six axis arm, the validation data was not shuffled, and no training was done on it. This is very good confirmation that the neural network is able to predict the required torques on movements it has not trained on. We would also like to mention that movement profiles were run using a limited range of velocities and accelerations, thus our expectation is that the neural network will have better

performance within that range. Nevertheless, this is acceptable, because under usual circumstances, the motion being performed for pick and place applications is highly repetitive and will only span a limited range of joint positions, velocities, and accelerations. Additionally, given we obtained such strong prediction results using one neural network for the whole system, this places confidence in our earlier assumption that all position, velocity, and acceleration values affect all the torques.

Chapter 5

Delta2

5.1 Delta2 used for Experiments

The Codian D2-800-Rz was used for experiments [25]. It can be seen in figure 5.1. The control panel can be seen in figure 5.2. Table 5.1 contains the joint specification and setup information. This mechanism has two axes for 2 DOF in x and z Cartesian space. Additionally, it has a third axis for rotating the end effector. Ultimately, a Delta2 mechanism is a simplified version of a Delta3 mechanism. Given that the dynamics equations will be of less complexity, we thought it would be worthwhile to include the Delta2 mechanism to see if the neural network can have even better performance.

	Joint 0	Joint 1	Joint 3
Motor Rated Torque (N.m)	3.18	3.18	1.27
Motor Rated Speed (RPM)	3000	3000	3000
Motor Power (Watts)	1000	1000	400
Encoder Resolution (tics/rev)	2^{24}	2^{24}	2^{24}
Gear Ratio	1:31	1:31	1:10
Polarity	Neg	Pos	Pos
Zero Position Offset (Degrees)	31.1314	14.2819	0
Min limit (Degrees)	-30	-30	$-\infty$
Max limit (Degrees)	85	85	∞

Table 5.1: D2-800-Rz robot joint specification and setup

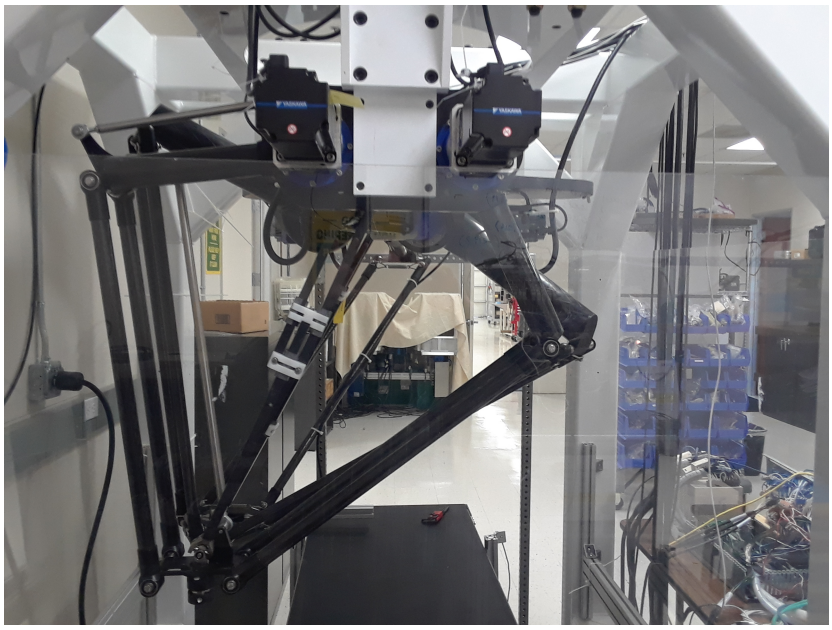


Figure 5.1: Codian D2-800-Rz

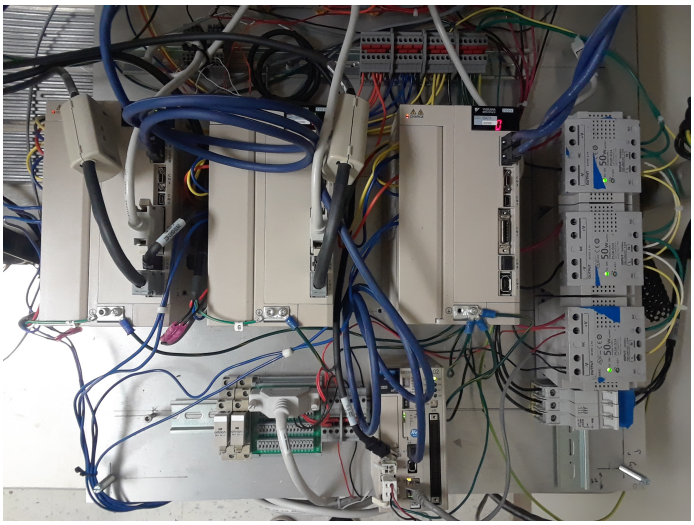


Figure 5.2: Codian D2-800-Rz Panel

5.2 Analytical Kinematics

A model and an analytical solution for the position, velocity, and acceleration forward and inverse kinematics solutions can be found in appendix C. Again, the kinematics

Training limits	x	z
min [mm]	-300	510
max [mm]	300	700

Table 5.2: Delta2 Cartesian coordinate training limits

derivation uses the loop closure equations in vector form. Note that the derivation ignores the third axis on the D2-800-Rz, because it is merely a trivial rotation of the end effector.

5.3 Machine Learning Dynamics

In our dynamics analysis, we will again be ignoring the third axis on the system used for rotating the end effector, because its dynamics are decoupled from the rest of the system.

As in the case of the Delta3 mechanism, we will be assuming that position, velocity, and acceleration of each of the first two joints will have effects on the torques of the first two joints. This is because again, the mechanism is a parallel manipulator.

Data Acquisition

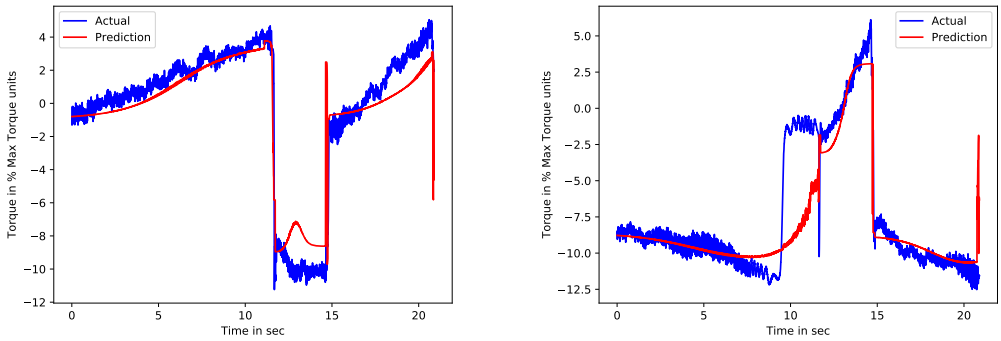
Similar to what was done with the six axis robot, the Delta2 robot was moved within a square of dimensions found in table 5.2. Additionally, the third axis which rotates the end effector was not used. The number of data points was approximately 26,000.

Single Neural Network for all Joints Results

Plots of the actual and predicted torque values for the first three joints when the hidden layer is of size three can be seen in figure 5.3. In this case, one neural network was used for all the joints. The R2 accuracy obtained was 0.89

Discussion

As in the case of the six axis arm and the Delta3 mechanism, the neural network was capable of capturing the dynamics of the system and was able to closely predict



(a) Joint 0 (b) Joint 1

Figure 5.3: Actual and predicted torque values on D2-800-Rz using one neural network

the torque outputs. The reason the accuracy was slightly less than that of Delta3, is because more data was used during training for Delta3.

Chapter 6

Delta3 On-line Neural Network Torque Estimation

6.1 PLC Controller Architecture

The Delta3 system was composed similarly to the six axis system in figure 1.3. The PLC controller is responsible for generating a world position, velocity and acceleration trajectory profile given a target position and a profile velocity, and acceleration. The controller then translates the world trajectory profile into a joint trajectory profile using the inverse kinematics equations given in appendix B. Finally, the neural network that was trained off line using feedback data in chapter 4 is run on the controller using joint position, velocity, and acceleration values as inputs in order to generate torque feedforward estimates as outputs. The controller architecture can be seen in figure 6.1. The PLC controller generates joint position, velocity, and torque feedforward commands, which are then sent to the servo drives controlling the motors. The servo drives use both feedforward and feedback control loops to control the torque, velocity, and position of the motors as shown in figure 1.1.

6.2 System Overview and PLC Controller and Servo Drive Update Rates

Given the PLC controller is designed to run PLC code in real time, it is required to perform a great deal of processing. PLC motion controllers are normally expected to conform to IEC-61131-3 [5] and PLCOpen Part 4 [24]. This is a high level programming language and requires a lot of computational power to be run. Thus PLC

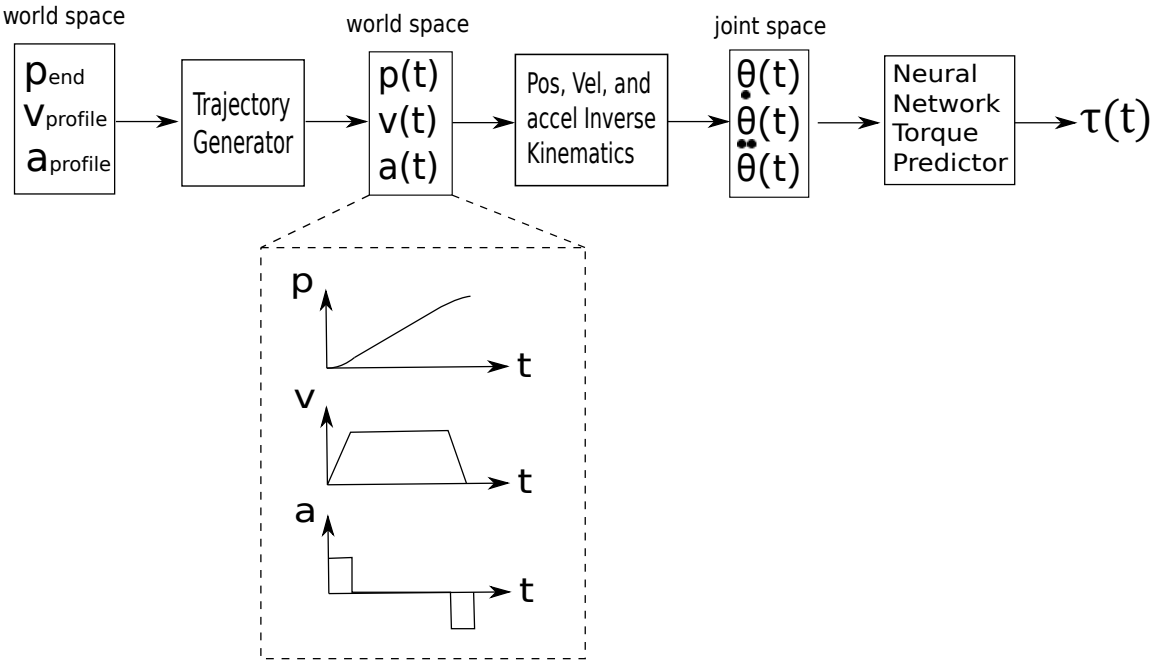


Figure 6.1: PLC controller architecture

controller update rates are typically slower than those of the servos. On the other hand, the servo drives, which are mostly running feedforward and feedback control loops, are capable of running at much higher update rates. In our system the PLC controller is running at 1 ms, and the servo control loops in figure 1.1 are running at faster update rates.

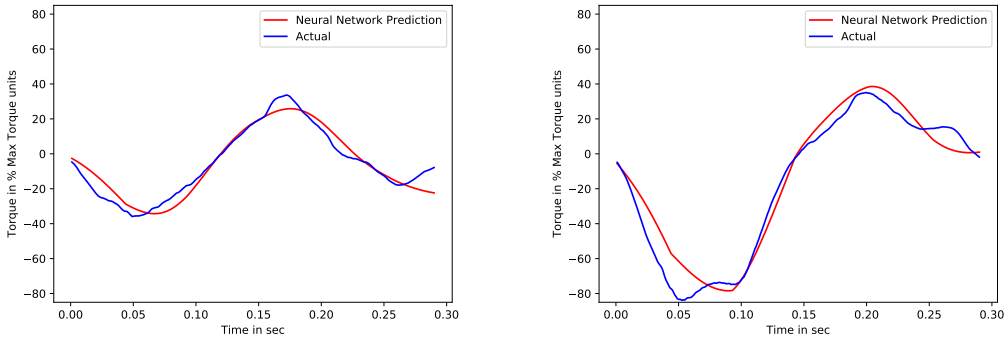
6.3 Experimentation Results

In order to reduce the impact of the inaccuracies in the neural network, a highly optimized neural network was trained on feedback data generated by seven back and forth moves between the two world position in table 6.1. The size of the data set was approximately 4000 points. The on-line torque feedforward prediction had excellent tracking of the feedback torque. This can be seen in figure 6.2. The R2 accuracy was 0.96. Motion trajectory was a trapezoidal velocity profile, and the profile velocity was 5 m/s, while the acceleration was 5 g's.

The same test move was run with and without the neural network torque feedforward, and position errors reported by each servo drive were plotted. Figure 6.3 contains the position error plots.

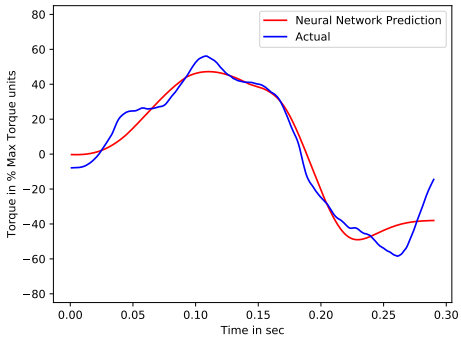
	x	y	z
Start [mm]	0	400	1000
End [mm]	0	-400	1000

Table 6.1: Delta3 world move limits



(a) Joint 0

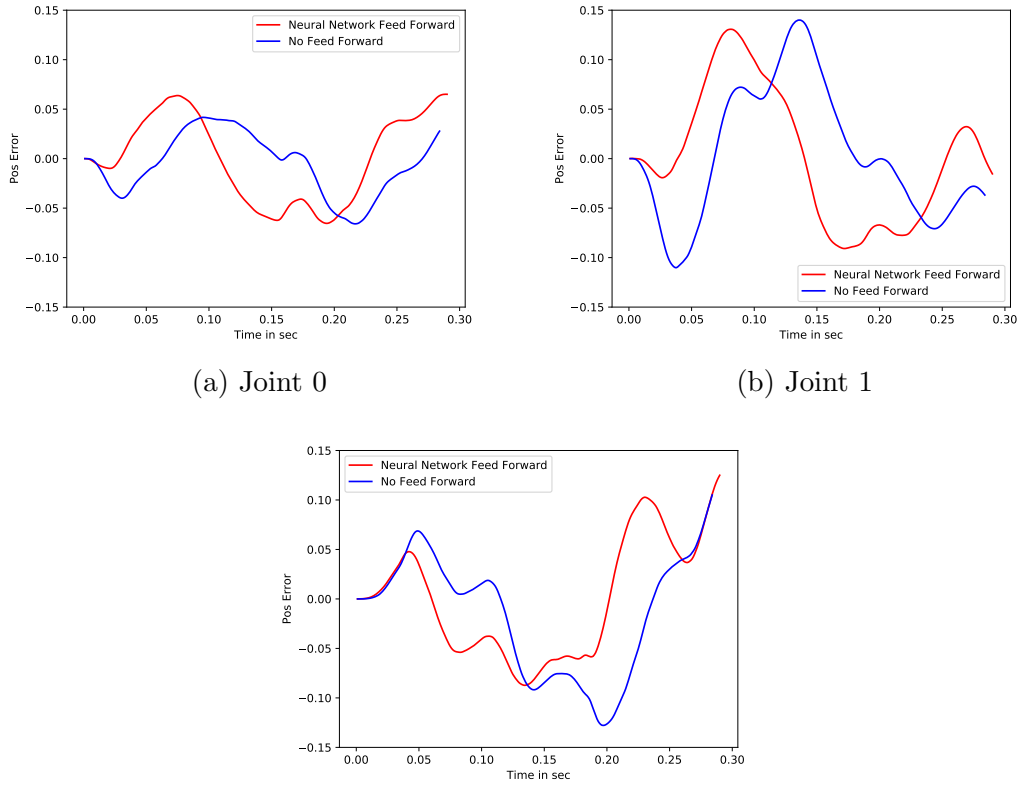
(b) Joint 1



(c) Joint 2

Figure 6.2: Actual and predicted torque values on D4-650 using optimized neural network

Given Yaskawa currently employs a far simpler solution for torque feedforward estimation called Pass Through, we have also included results while running the same moves using the Pass Through torque feedforward method in Figures 6.5 and 6.4 for comparison purposes. We note that the Pass Through method uses the equation $\tau_{ff} = C\ddot{\theta}$, where C is the Pass Through coefficient and is manually tunned such



(a) Joint 0

(b) Joint 1

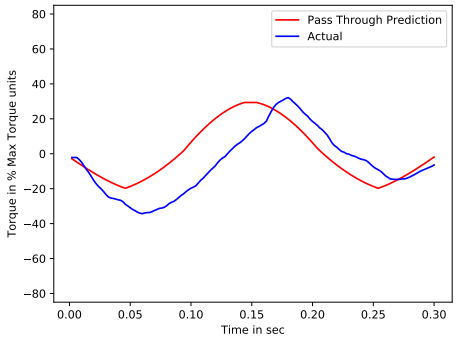
(c) Joint 2

Figure 6.3: Joint position errors on D4-650 using optimized neural network

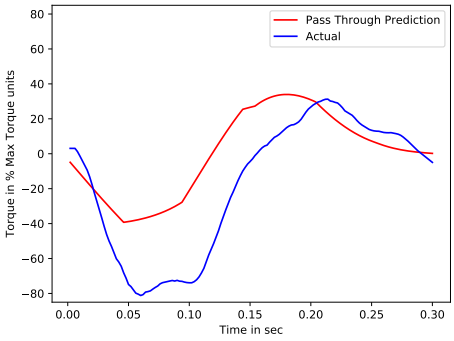
that position errors are minimized during motion. In the case of the delta3 robot that we used C was 0.00755 for each of the three joints. We note that the Pass Through method assumes there are no Coriolis, gravitational, or frictional terms in the dynamics. Additionally, it assumes that the inertia of the system is constant.

6.4 Discussion

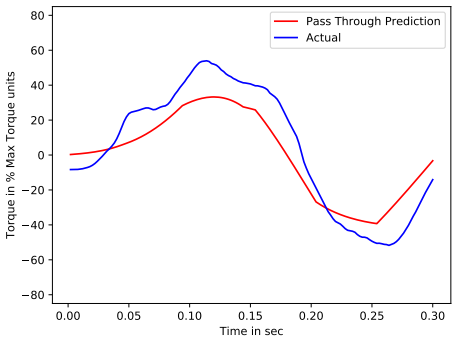
Although the neural network clearly predicted the torque values far better than the Pass Through method, the overall improvement in position errors was clearly better with the use of the Pass Through method.



(a) Joint 0

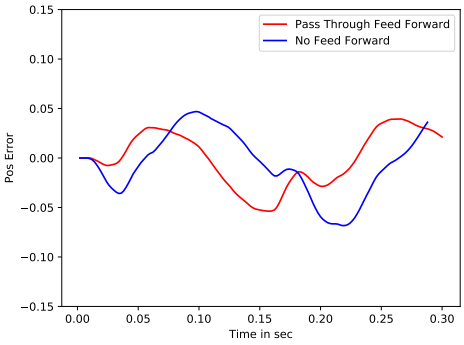


(b) Joint 1

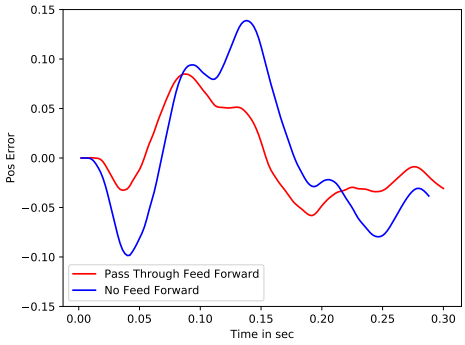


(c) Joint 2

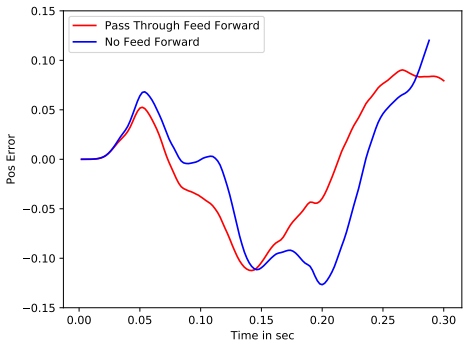
Figure 6.4: Actual and predicted torque values on D4-650 using optimized pass through coefficients



(a) Joint 0



(b) Joint 1



(c) Joint 2

Figure 6.5: Joint position errors on D4-650 using optimized pass through coefficients

Chapter 7

Simulation Analysis

7.1 Discrete Time Simulation of Simple Mass System

In order to better interpret the results seen with neural network torque feedforward, a simple test simulating the system in diagram 7.1 was run. The system was discretized using zero order hold. Simulation was done using python. The filters in the diagram are low pass filters for filtering out high frequency noise. Model error used was 10%.

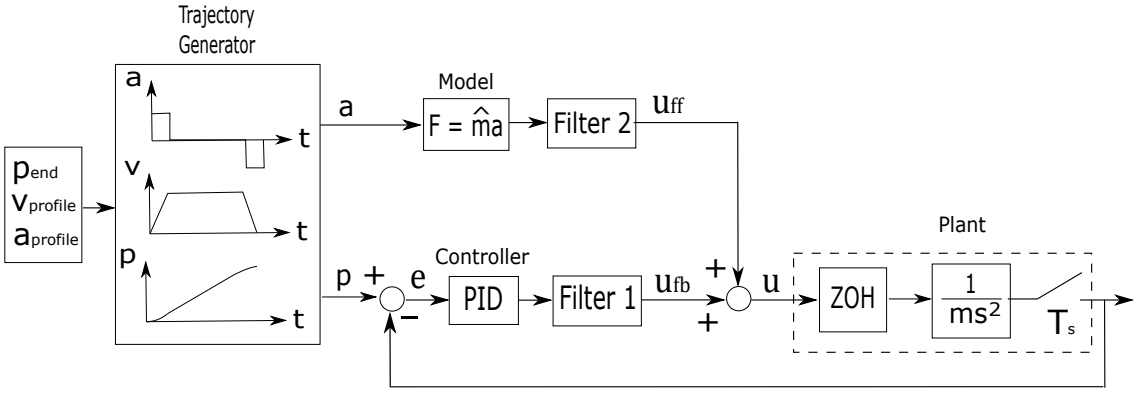
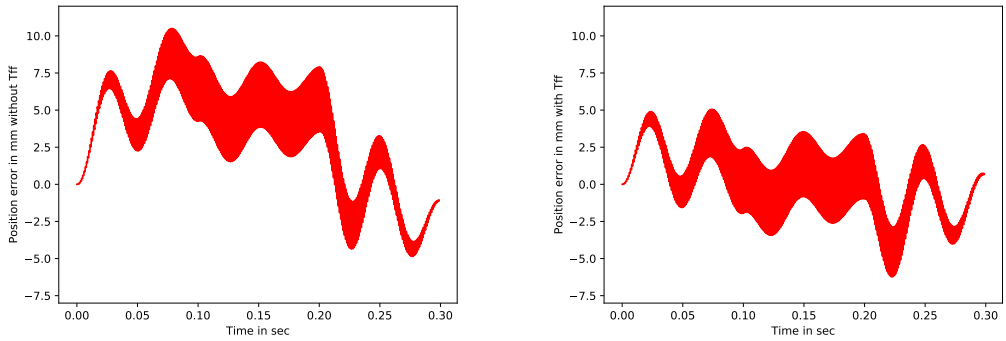


Figure 7.1: Simple mass system with feedforward control

Simulation results for position errors vs time without and with feedforward can be seen in figure 7.2. We note that the position error signals are wide due to noise.



(a) Without feedforward (b) With feedforward

Figure 7.2: Joint position errors vs time without and with feedforward

7.2 Continuous Time Simulation of a MIMO System

Given the use of a neural network for torque feedforward is essentially equivalent to plant inversion, simulation was performed using matlab in continuous time of a 2 input 2 output system. Tests were run with and without feedforward and with and without modeling error. Additionally tests were run with and without the M matrix being ill-conditioned. The plant constituted of the M matrix and an integrator. The system can be seen in figure 7.3. We note that K is a diagonal matrix, while M is randomly generated. When M is ill-conditioned, the condition number used was 900. The condition number was achieved using singular value decomposition in matlab and using eigenvalues of 30 and $\frac{1}{30}$. When modeling error was present, 10% was used. Diagonal elements of K were found iteratively such that the closed loop system was stable without feedforward control.

Simulation results for a step response with and without feedforward, plant uncertainty, and the plant being ill-conditioned, can be seen in figures 7.4, 7.5, 7.6, and 7.7. The ill-conditioned matrix used for the plant is given in equation 7.1. A summary of whether feedforward or no feedforward had better performance for all the scenarios can be seen in table 7.1.

$$M = \begin{bmatrix} 10.1345 & 13.7001 \\ 14.6362 & 19.8843 \end{bmatrix} \tag{7.1}$$

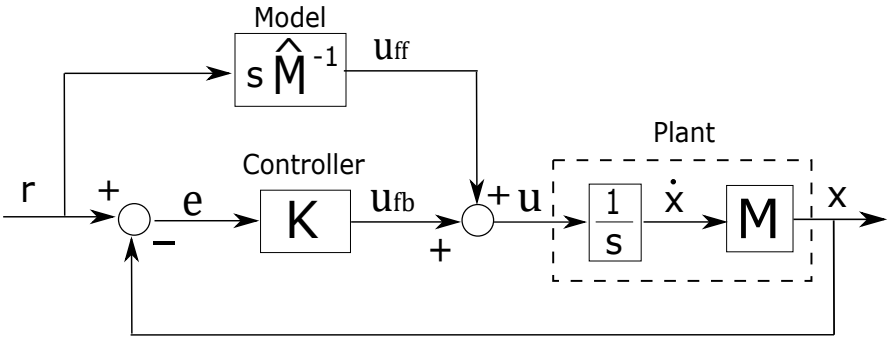


Figure 7.3: Continuous time MIMO system with plant inversion

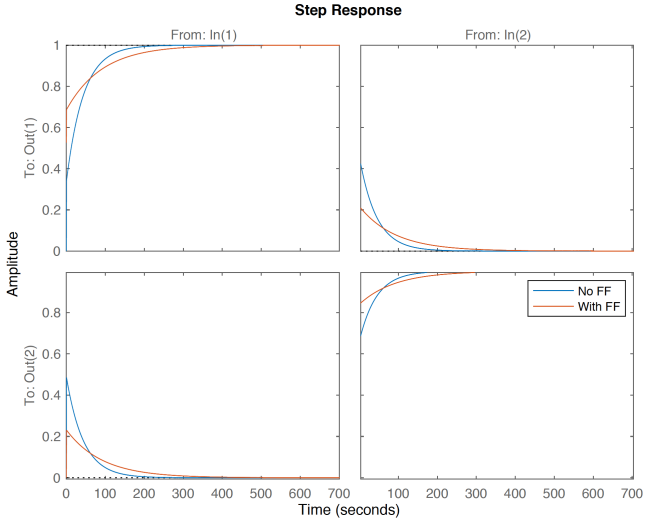


Figure 7.4: Ill-conditioned MIMO system step response with model uncertainty and without and with feedforward

7.3 Discussion

The simple mass system clearly demonstrated improved performance with the use of feedforward control. This is not surprising, because the effectiveness of feedforward control is already well established [28]. However, simulation results of the ill-conditioned plant with feedforward showed degraded performance. Settling time of both the inputs and the outputs of the system increased considerably when feedforward control was added, despite modeling error only being at 10%. This is the case, because according to [30], inverting an ill-conditioned plant yields bad performance if there is model uncertainty. Not surprisingly, the only situation where no

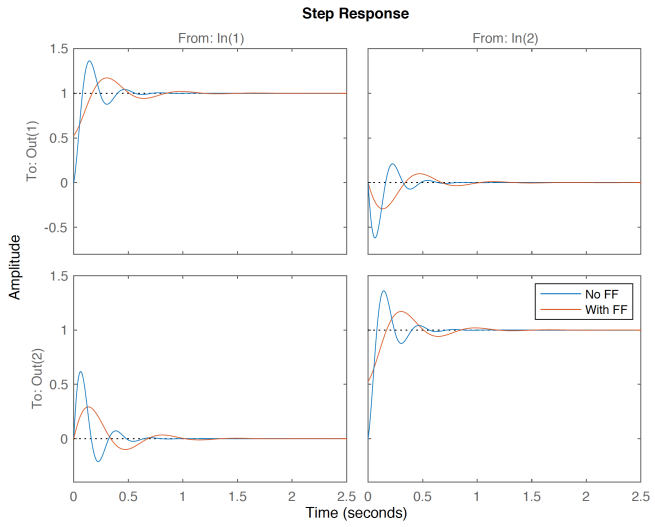


Figure 7.5: Non ill-conditioned MIMO system step response with model uncertainty and without and with feedforward

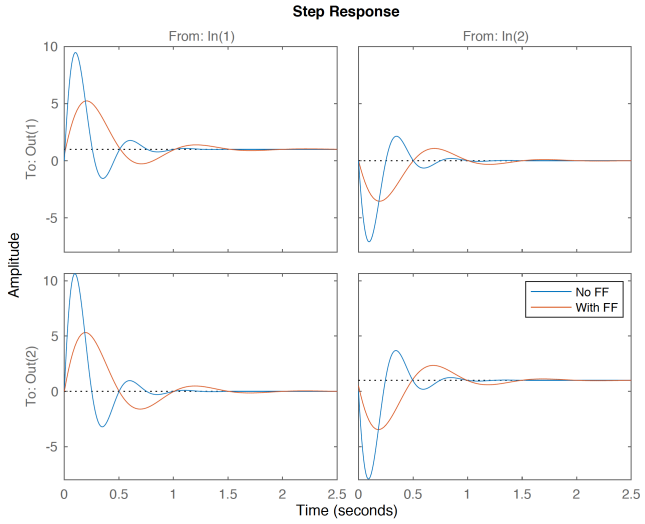


Figure 7.6: Ill-conditioned MIMO system step response without model uncertainty and without and with feedforward

feedforward outperformed feedforward is when the plant is ill-conditioned, and modeling error is present. In all other scenarios, adding feedforward control improved the overall performance of the system.

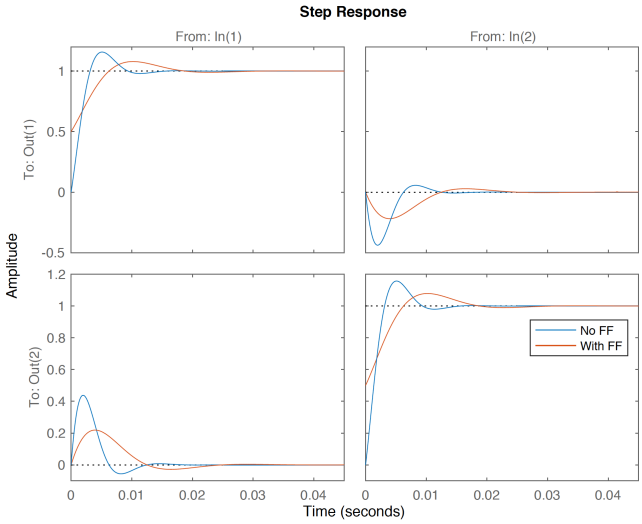


Figure 7.7: Non ill-conditioned MIMO system step response without model uncertainty and without and with feedforward

	Model error	No Model error
Plant ill-conditioned	No feedforward	Feedforward
Plant not ill-conditioned	Feedforward	Feedforward

Table 7.1: Summary of whether feedforward or no feedforward had improved step response

Chapter 8

Concluding Remarks and Future Work

8.1 Concluding Remarks

As seen in this work, neural networks were capable of learning robot inverse kinematics solutions as well as robot dynamics equations. However, they were highly dependent on training with large amounts of data. Additionally, removal of independent variables from the input training data highly improved performance as seen with the six axis robot dynamics. In the case of robot inverse kinematics, the accuracies obtained with the neural networks are not sufficient for accurate robot positioning. However, in the case of robot dynamics modeling, the accuracies obtained with neural networks were sufficient for improving robot tracking performance. Neural networks were able to produce good estimates of torque efforts on multiple types of parallel and serial manipulators. Training on actual feedback data from the mechanism eliminates the need for traditional methods of modeling and estimation that suffer from many inaccuracies. Traditional methods of dynamics modeling require many simplifications in the model to make the equations practical for analytical derivation.

We conclude that small but highly optimized neural networks can still accurately capture the dynamics of nonlinear systems. The neural networks used for dynamics in this work only had one hidden layer of length 6. These neural networks ran in approximately 5 μs. However, as we saw while looking at the position error plots, the neural network had mediocre improvement in performance. We attribute this to the plant dynamics equations being ill-conditioned. This was clearly the case with simulation results. Plant inversion degraded the performance when the plant model was ill-conditioned.

On the other hand, the simple Pass Through method for torque feedforward estimation had better performance improvement in position errors although it in fact had worse prediction of the torque values. This is because the Pass Through method is a much milder attempt at inverting the plant, and it thus did not suffer from the plant being ill-conditioned.

8.2 Future Work

The performance of clustering robot configurations may be increased by either using a simpler mechanism such as a SCARA (Selective Compliance Assembly Robot Arm) [7] robot which only has two configurations and 3 DOF, or by drastically increasing the number of data points. This is still a very interesting problem to solve, and can make the neural network inverse kinematics solution far more practical in the case of a generalized manipulator.

Testing the neural network torque feedforward strategy on a different system with faster response times such as a SCARA mechanism would be highly desirable. Since SCARA mechanisms are normally built with direct drive motors [6], they do not have gear boxes but have very fast response times. Direct drive motors are motors that are large enough to provide necessary torques for motion without the use of a gear box. As such, little torque effort goes to friction, since gear boxes normally consume the majority of the friction. Such systems may be far more suited for neural network torque feedforward control, because of their higher bandwidth.

Another highly desirable benefit of neural networks for torque estimation is the potential to have the robot perform the training online. For example, as the robot moves around, the neural network weights could be updated based on the feedback data it receives from the motors. Thus the mechanism could continuously improve its performance. This would require that the neural network training algorithm be implemented on the PLC controller firmware.

Bibliography

[1] "Abstracts". In: *Biometrics* 21.3 (1965), pp. 761–777. ISSN: 0006341X, 15410420. URL: http://www.jstor.org/stable/2528559.

[2] Z. Bingul, H. M. Ertunc, and C. Oysu. "Comparison of inverse kinematics solutions using neural network for 6R robot manipulator with offset". In: *2005 ICSC Congress on Computational Intelligence Methods and Applications*. 2005. DOI: 10.1109/CIMA.2005.1662342.

[3] S. Chen, S. A. Billings, and P. M. Grant. "Non-linear system identification using neural networks". In: *International Journal of Control* 51.6 (1990), pp. 1191–1214. DOI: 10.1080/00207179008934126. eprint: https://doi.org/10.1080/00207179008934126. URL: https://doi.org/10.1080/00207179008934126.

[4] R. Clavel. *Device for the movement and positioning of an element in space*. US Patent 4,976,582. Dec. 1990. URL: https://encrypted.google.com/patents/US4976582.

[5] International Electrotechnical Commission. *IEC 61131-3*. Feb. 2013. URL: https://webstore.iec.ch/publication/4552.

[6] Wikipedia contributors. *Direct drive mechanism — Wikipedia, The Free Encyclopedia*. [Online; accessed 1-April-2018]. 2018. URL: https://en.wikipedia.org/w/index.php?title=Direct_drive_mechanism&oldid=822010497.

[7] Wikipedia contributors. *SCARA — Wikipedia, The Free Encyclopedia*. [Online; accessed 1-April-2018]. 2018. URL: https://en.wikipedia.org/w/index.php?title=SCARA&oldid=830935139.

[8] J. Denavit and R. S. Hartenberg. "A kinematic notation for lower-pair mechanisms based on matrices". In: *Trans. ASME E, Journal of Applied Mechanics* 22 (June 1955), pp. 215–221.

[9] Chris Edwards. "Growing Pains for Deep Learning". In: *Commun. ACM* 58.7 (June 2015), pp. 14–16. ISSN: 0001-0782. DOI: 10.1145/2771283. URL: http://doi.acm.org/10.1145/2771283.

[10] A. Goedtel et al. "Load torque estimation in induction motors using artificial neural networks". In: *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*. Vol. 2. 2002, pp. 1379–1384. DOI: 10.1109/IJCNN.2002.1007717.

[11] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer New York, 2013. ISBN: 9780387216065. URL: https://books.google.com/books?id=yPfZBwAAQBAJ.

[12] G. James et al. *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer New York, 2014. ISBN: 9781461471370. URL: https://books.google.com/books?id=at1bmAEACAAJ.

[13] Seul Jung and T. C. Hsia. "Neural network impedance force control of robot manipulator". In: *IEEE Transactions on Industrial Electronics* 45.3 (June 1998), pp. 451–461. ISSN: 0278-0046. DOI: 10.1109/41.679003.

[14] Rasit Koker et al. "A study of neural network based inverse kinematics solution for a three-joint robot". In: *Robotics and Autonomous Systems* 49.3 (2004). Patterns and Autonomous Control, pp. 227–234. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2004.09.010. URL: http://www.sciencedirect.com/science/article/pii/S0921889004001666.

[15] J.L. Lagrange. *Mécanique analytique*. Mécanique analytique v. 1. Ve Courcier, 1811. URL: https://books.google.co.uk/books?id=Q8MKAAAAYAAJ.

[16] Jer-Junn Luh et al. "Isokinetic elbow joint torques estimation from surface EMG and joint kinematic data: using an artificial neural network model". In: *Journal of Electromyography and Kinesiology* 9.3 (1999), pp. 173–183. ISSN: 1050-6411. DOI: https://doi.org/10.1016/S1050-6411(98)00030-3. URL: http://www.sciencedirect.com/science/article/pii/S1050641198000303.

[17] D. Manocha and J. F. Canny. "Efficient inverse kinematics for general 6R manipulators". In: *IEEE Transactions on Robotics and Automation* 10.5 (Oct. 1994), pp. 648–657. ISSN: 1042-296X. DOI: 10.1109/70.326569.

[18] Warren S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259.

[19] E. Mollick. "Establishing Moore's Law". In: *IEEE Annals of the History of Computing* 28.3 (July 2006), pp. 62–75. ISSN: 1058-6180. DOI: 10.1109/MAHC.2006.45.

[20] Yaskawa Motoman. *Motoman MHJ Robot*. URL: https://www.motoman.com/industrial-robots/mhjf.

[21] R.M. Murray et al. *A Mathematical Introduction to Robotic Manipulation*. Taylor & Francis, 1994. ISBN: 9780849379819. URL: https://books.google.com/books?id=D%5C_PqGKRo7oIC.

[22] K. S. Narendra and K. Parthasarathy. "Identification and control of dynamical systems using neural networks". In: *IEEE Transactions on Neural Networks* 1.1 (Mar. 1990), pp. 4–27. ISSN: 1045-9227. DOI: 10.1109/72.80202.

[23] H. C. Nho and P. Meckl. "Intelligent feedforward control and payload estimation for a two-link robotic manipulator". In: *IEEE/ASME Transactions on Mechatronics* 8.2 (June 2003), pp. 277–282. ISSN: 1083-4435. DOI: 10.1109/TMECH.2003.812847.

[24] PLC Open. *PLC Open Part4*. Dec. 2008. URL: http://www.plcopen.org/pages/tc2_motion_control/downloads/part4_coordinatedmotion_v10.pdf.

[25] Codian Robotics. *Codian D2 Robots*. URL: http://www.codian-robotics.com/en/d2-robots/.

[26] Codian Robotics. *Codian D4 Robots*. URL: http://www.codian-robotics.com/en/d4-robots/.

[27] María M. Seron, Julio H. Braslavsky, and Graham C. Goodwin. "MIMO Control". In: *Fundamental Limitations in Filtering and Control*. London: Springer London, 1997, pp. 85–117. ISBN: 978-1-4471-0965-5. DOI: 10.1007/978-1-4471-0965-5_4. URL: https://doi.org/10.1007/978-1-4471-0965-5_4.

[28] B. Siciliano and O. Khatib. *Springer Handbook of Robotics*. Springer Handbook of Robotics. Springer Berlin Heidelberg, 2008. ISBN: 9783540239574. URL: https://books.google.com/books?id=Xpgi5gSuBxsC.

[29] B. Siciliano et al. *Robotics: Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer London, 2010. ISBN: 9781849966344. URL: https://books.google.com/books?id=vNpdewAACAAJ.

[30] S. Skogestad, M. Morari, and J. C. Doyle. "Robust control of ill-conditioned plants: high-purity distillation". In: *IEEE Transactions on Automatic Control* 33.12 (Dec. 1988), pp. 1092–1105. ISSN: 0018-9286. DOI: 10.1109/9.14431.

[31] Sreenivas Tejomurtula and Subhash Kak. "Inverse kinematics in robotics using neural networks". In: *Information Sciences* 116.2 (1999), pp. 147–164. ISSN: 0020-0255. DOI: https://doi.org/10.1016/S0020-0255(98)10098-1. URL: http://www.sciencedirect.com/science/article/pii/S0020025598100981.

[32] L.W. Tsai. *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*. A Wiley-Interscience publication. Wiley, 1999. ISBN: 9780471325932. URL: https://books.google.com/books?id=PK%5C_N9aFZ3ccC.

[33] P.J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University, 1975. URL: https://books.google.com/books?id=z81XmgEACAAJ.

Appendix A

Six Axis Kinematics Derivation

A.1 Kinematics Model

This appendix goes over the forward and inverse kinematics derivation for a 6 DOF Yaskawa Motoman robot. Figure A.1 has assigned frames of references that match Yaskawa joint movement conventions. The frames of reference are also assigned to match Denavit Hartenberg conventions.

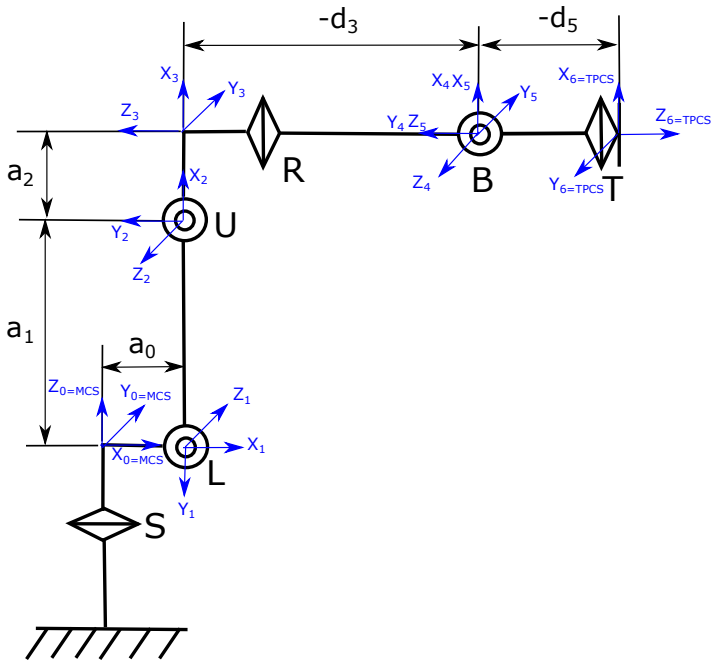


Figure A.1: 6 DOF arm kinematics model

Joint	α	a	d	θ
0	$-\frac{\pi}{2}$	a_0	0	θ_0
1	π	a_1	0	θ_1
2	$-\frac{\pi}{2}$	a_2	0	θ_2
3	$\frac{\pi}{2}$	0	d_3	θ_3
4	$-\frac{\pi}{2}$	0	0	θ_4
5	π	0	d_5	θ_5

Table A.1: DH parametrization

Given the defined model, the Denavit Hartenberg parameters are presented in table A.1

The transformation from joint n-1 to n using the Denavit Hartenberg parameters is done using the transformation matrix in equation A.1

$$ {}^{n-1}A_n = \begin{bmatrix} cos\theta_n & -sin\theta_n cos\alpha_n & sin\theta_n sin\alpha_n & a_n cos\theta_n \\ sin\theta_n & cos\theta_n cos\alpha_n & -cos\theta_n sin\alpha_n & a_n sin\theta_n \\ 0 & sin\alpha_n & cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.1} $$

A.2 Position Forward Kinematics

To simplify the equations the following convention shall be used:

$cos\theta = c\theta$

$sin\theta = s\theta$

$cos(\theta_0 + \theta_1) = c\theta_{0+1}$

$cos(\theta_0 - \theta_1) = c\theta_{0-1}$

Evaluating equation A.1 for joints 1 through 6 yields:

$$ {}^{0}A_1 = \begin{bmatrix} c\theta_0 & 0 & -s\theta_0 & a_0 c\theta_0 \\ s\theta_0 & 0 & c\theta_0 & a_0 s\theta_0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{A.2} $$

$$ {}^{1}A_{2} = \begin{bmatrix} c\theta_1 & s\theta_1 & 0 & a_1 c\theta_1 \\ s\theta_1 & -c\theta_1 & 0 & a_1 s\theta_1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{A.3} $$

$$ {}^{2}A_{3} = \begin{bmatrix} c\theta_2 & 0 & -s\theta_2 & a_2 c\theta_2 \\ s\theta_2 & 0 & c\theta_2 & a_2 s\theta_2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{A.4} $$

$$ {}^{3}A_{4} = \begin{bmatrix} c\theta_3 & 0 & s\theta_3 & 0 \\ s\theta_3 & 0 & -c\theta_3 & 0 \\ 0 & 1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{A.5} $$

$$ {}^{4}A_{5} = \begin{bmatrix} c\theta_4 & 0 & -s\theta_4 & 0 \\ s\theta_4 & 0 & c\theta_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{A.6} $$

$$ {}^{5}A_{6} = \begin{bmatrix} c\theta_5 & s\theta_5 & 0 & 0 \\ s\theta_5 & -c\theta_5 & 0 & 0 \\ 0 & 0 & -1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{A.7} $$

$$ {}^{0}A_{6} = \begin{bmatrix} u_x & v_x & w_x & q_x \\ u_y & v_y & w_y & q_y \\ u_z & v_z & w_z & q_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{A.8} $$

It follows that:

$$
\begin{aligned}
{}^0A_3 &= {}^0A_1^1A_2^2A_3 \\
&= \begin{bmatrix} c\theta_0 & 0 & -s\theta_0 & a_0c\theta_0 \\ s\theta_0 & 0 & c\theta_0 & a_0s\theta_0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_1 & s\theta_1 & 0 & a_1c\theta_1 \\ s\theta_1 & -c\theta_1 & 0 & a_1s\theta_1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_2 & 0 & -s\theta_2 & a_2c\theta_2 \\ s\theta_2 & 0 & c\theta_2 & a_2s\theta_2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c\theta_0 & 0 & -s\theta_0 & a_0c\theta_0 \\ s\theta_0 & 0 & c\theta_0 & a_0s\theta_0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_{2-1} & 0 & -s\theta_{2-1} & a_1c\theta_1 + a_2c\theta_{2-1} \\ -s\theta_{2-1} & 0 & -c\theta_{2-1} & a_1s\theta_1 - a_2s\theta_{2-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c\theta_0c\theta_{2-1} & -s\theta_0 & -c\theta_0s\theta_{2-1} & c\theta_0(a_0 + a_1c\theta_1 + a_2c\theta_{2-1}) \\ s\theta_0c\theta_{2-1} & c\theta_0 & -s\theta_0s\theta_{2-1} & s\theta_0(a_0 + a_1c\theta_1 + a_2c\theta_{2-1}) \\ s\theta_{2-1} & 0 & c\theta_{2-1} & -a_1s\theta_1 + a_2s\theta_{2-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{A.9}
$$

$$
\begin{aligned}
{}^3A_6 &= {}^3A_4^4A_5^5A_6 \\
&= \begin{bmatrix} c\theta_3 & 0 & s\theta_3 & 0 \\ s\theta_3 & 0 & -c\theta_3 & 0 \\ 0 & 1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_4 & 0 & -s\theta_4 & 0 \\ s\theta_4 & 0 & c\theta_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_5 & s\theta_5 & 0 & 0 \\ s\theta_5 & -c\theta_5 & 0 & 0 \\ 0 & 0 & -1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c\theta_3c\theta_4 & -s\theta_3 & -c\theta_3s\theta_4 & 0 \\ s\theta_3c\theta_4 & c\theta_3 & -s\theta_3s\theta_4 & 0 \\ s\theta_4 & 0 & c\theta_4 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_5 & s\theta_5 & 0 & 0 \\ s\theta_5 & -c\theta_5 & 0 & 0 \\ 0 & 0 & -1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c\theta_3c\theta_4c\theta_5 - s\theta_3s\theta_5 & c\theta_3c\theta_4s\theta_5 + s\theta_3c\theta_5 & c\theta_3s\theta_4 & -d_5c\theta_3s\theta_4 \\ s\theta_3c\theta_4c\theta_5 + c\theta_3s\theta_5 & s\theta_3c\theta_4s\theta_5 - c\theta_3c\theta_5 & s\theta_3s\theta_4 & -d_5s\theta_3s\theta_4 \\ s\theta_4c\theta_5 & s\theta_4s\theta_5 & -c\theta_4 & d_5c\theta_4 + d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{A.10}
$$

The complete forward transformation matrix is given by:

$$
{}^0A_6 = {}^0A_3^3A_6 \tag{A.11}
$$

Substituting equations A.9 and A.10 into A.11 and equating to A.8 gives:

$$\begin{bmatrix} u_x & v_x & w_x & q_x \\ u_y & v_y & w_y & q_y \\ u_z & v_z & w_z & q_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c\theta_0 c\theta_{2-1} & -s\theta_0 & -c\theta_0 s\theta_{2-1} & c\theta_0(a_0 + a_1 c\theta_1 + a_2 c\theta_{2-1}) \\ s\theta_0 c\theta_{2-1} & c\theta_0 & -s\theta_0 s\theta_{2-1} & s\theta_0(a_0 + a_1 c\theta_1 + a_2 c\theta_{2-1}) \\ s\theta_{2-1} & 0 & c\theta_{2-1} & -a_1 s\theta_1 + a_2 s\theta_{2-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$\begin{bmatrix} c\theta_3 c\theta_4 c\theta_5 - s\theta_3 s\theta_5 & c\theta_3 c\theta_4 s\theta_5 + s\theta_3 c\theta_5 & c\theta_3 s\theta_4 & -d_5 c\theta_3 s\theta_4 \\ s\theta_3 c\theta_4 c\theta_5 + c\theta_3 s\theta_5 & s\theta_3 c\theta_4 s\theta_5 - c\theta_3 c\theta_5 & s\theta_3 s\theta_4 & -d_5 s\theta_3 s\theta_4 \\ s\theta_4 c\theta_5 & s\theta_4 s\theta_5 & -c\theta_4 & d_5 c\theta_4 + d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(A.12)}$$

Which yields the forward transformation equations below:

$$u_x = c\theta_0 c\theta_{2-1}(c\theta_3 c\theta_4 c\theta_5 - s\theta_3 s\theta_5) - s\theta_0(s\theta_3 c\theta_4 c\theta_5 + c\theta_3 s\theta_5) - c\theta_0 s\theta_{2-1} s\theta_4 c\theta_5$$

$$u_y = s\theta_0 c\theta_{2-1}(c\theta_3 c\theta_4 c\theta_5 - s\theta_3 s\theta_5) + c\theta_0(s\theta_3 c\theta_4 c\theta_5 + c\theta_3 s\theta_5) - s\theta_0 s\theta_{2-1} s\theta_4 c\theta_5$$

$$u_z = s\theta_{2-1}(c\theta_3 c\theta_4 c\theta_5 - s\theta_3 s\theta_5) + c\theta_{2-1} s\theta_4 c\theta_5$$

$$v_x = c\theta_0 c\theta_{2-1}(c\theta_3 c\theta_4 s\theta_5 + s\theta_3 c\theta_5) - s\theta_0(s\theta_3 c\theta_4 s\theta_5 - c\theta_3 c\theta_5) - c\theta_0 s\theta_{2-1} s\theta_4 s\theta_5$$

$$v_y = s\theta_0 c\theta_{2-1}(c\theta_3 c\theta_4 s\theta_5 + s\theta_3 c\theta_5) + c\theta_0(s\theta_3 c\theta_4 s\theta_5 - c\theta_3 c\theta_5) - s\theta_0 s\theta_{2-1} s\theta_4 s\theta_5$$

$$v_z = s\theta_{2-1}(c\theta_3 c\theta_4 s\theta_5 + s\theta_3 c\theta_5) + c\theta_{2-1} s\theta_4 s\theta_5$$

$$w_x = c\theta_0 c\theta_{2-1} c\theta_3 s\theta_4 - s\theta_0 s\theta_3 s\theta_4 + c\theta_0 s\theta_{2-1} c\theta_4$$

$$w_y = s\theta_0 c\theta_{2-1} c\theta_3 s\theta_4 + c\theta_0 s\theta_3 s\theta_4 + s\theta_0 s\theta_{2-1} c\theta_4$$

$$w_z = s\theta_{2-1}c\theta_3 s\theta_4 - c\theta_{2-1}c\theta_4$$

$$q_x = -d_5 c\theta_0 c\theta_{2-1}c\theta_3 s\theta_4 + d_5 s\theta_0 s\theta_3 s\theta_4 - c\theta_0 s\theta_{2-1}(d_5 c\theta_4 + d_3) + c\theta_0(a_0 + a_1 c\theta_1 + a_2 c\theta_{2-1})$$

$$q_y = -d_5 s\theta_0 c\theta_{2-1}c\theta_3 s\theta_4 - d_5 c\theta_0 s\theta_3 s\theta_4 - s\theta_0 s\theta_{2-1}(d_5 c\theta_4 + d_3) + s\theta_0(a_0 + a_1 c\theta_1 + a_2 c\theta_{2-1})$$

$$q_z = -d_5 s\theta_{2-1}c\theta_3 s\theta_4 + c\theta_{2-1}(d_5 c\theta_4 + d_3) - a_1 s\theta_1 + a_2 s\theta_{2-1}$$

A.3 Position Inverse Kinematics

Solution for S, U, and L Joint Angles

Let p denote the position of the wrist center, where the wrist center is the intersection point of Axis 4, 5 and 6.

Using figure A.1, it follows that the position of point p in coordinate frame 6 is:

$${}^6p = \begin{bmatrix} 0 \\ 0 \\ d_5 \\ 1 \end{bmatrix} \tag{A.13}$$

Thus:

$${}^0p = \begin{bmatrix} u_x & v_x & w_x & q_x \\ u_y & v_y & w_y & q_y \\ u_z & v_z & w_z & q_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ d_5 \\ 1 \end{bmatrix} = \begin{bmatrix} q_x + d_5 w_x \\ q_y + d_5 w_y \\ q_z + d_5 w_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \tag{A.14}$$

Also using figure A.1, it follows that the position of point p in coordinate frame 3 is:

$$ {}^{3}p = \begin{bmatrix} 0 \\ 0 \\ d_3 \\ 1 \end{bmatrix} \tag{A.15} $$

Using the matrix transformation, we have:

$$ {}^{0}p = {}^{0}A_3 {}^{3}p \tag{A.16} $$

Left multiplying both sides of equation A.16 by the inverse of ${}^{0}A_1$ yields:

$$ \left({}^{0}A_1\right)^{-1} {}^{0}p = {}^{1}A_2 {}^{2}A_3 {}^{3}p = {}^{1}A_3 {}^{3}p \tag{A.17} $$

Note that for a general transformation matrix T with a 3 x 3 rotation R and a 3 x 1 translation vector V:

$$ T = \begin{bmatrix} R & V \\ 0 & 1 \end{bmatrix} $$

The inverse is given by:

$$ T = \begin{bmatrix} R^T & -R^T V \\ 0 & 1 \end{bmatrix} $$

Given ${}^{0}A_1$ is of the form:

$$ {}^{0}A_1 = \begin{bmatrix} {}^{0}R_1 & {}^{0}V_1 \\ 0 & 1 \end{bmatrix} $$

Then ${}^{0}A_1$ inverse can be computed using:

$$ \left({}^{0}A_1\right)^{-1} = \begin{bmatrix} \left({}^{0}R_1\right)^T & -\left({}^{0}R_1\right)^T {}^{0}V_1 \\ 0 & 1 \end{bmatrix} $$

where:

$$\left({}^0R_1\right)^T = \begin{bmatrix} c\theta_0 & s\theta_0 & 0 \\ 0 & 0 & -1 \\ -s\theta_0 & c\theta_0 & 0 \end{bmatrix}$$

and:

$$-\left({}^0R_1\right)^T {}^0V_1 = -\begin{bmatrix} c\theta_0 & s\theta_0 & 0 \\ 0 & 0 & -1 \\ -s\theta_0 & c\theta_0 & 0 \end{bmatrix} * \begin{bmatrix} a_0 c\theta_0 \\ a_0 s\theta_0 \\ 0 \end{bmatrix} = \begin{bmatrix} -a_0 \\ 0 \\ 0 \end{bmatrix}$$

Thus the inverse of ${}^0A_1$ is:

$$\left({}^0A_1\right)^{-1} = \begin{bmatrix} c\theta_0 & s\theta_0 & 0 & -a_0 \\ 0 & 0 & -1 & 0 \\ -s\theta_0 & c\theta_0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.18}$$

Substituting equations A.18, A.14, A.3, A.4, and A.15 into equation A.17 yields:

$$\begin{aligned}
\begin{bmatrix} c\theta_0 & s\theta_0 & 0 & -a_0 \\ 0 & 0 & -1 & 0 \\ -s\theta_0 & c\theta_0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} &= \begin{bmatrix} c\theta_1 & s\theta_1 & 0 & a_1 c\theta_1 \\ s\theta_1 & -c\theta_1 & 0 & a_1 s\theta_1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_2 & 0 & -s\theta_2 & a_2 c\theta_2 \\ s\theta_2 & 0 & c\theta_2 & a_2 s\theta_2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ d_3 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} c\theta_1 & s\theta_1 & 0 & a_1 c\theta_1 \\ s\theta_1 & -c\theta_1 & 0 & a_1 s\theta_1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -d_3 s\theta_2 + a_2 c\theta_2 \\ d_3 c\theta_2 + a_2 s\theta_2 \\ 0 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} -d_3(s\theta_2 c\theta_1 - c\theta_2 s\theta_1) + a_2(c\theta_2 c\theta_1 + s\theta_2 s\theta_1) + a_1 c\theta_1 \\ -d_3(c\theta_2 c\theta_1 + s\theta_2 s\theta_1) - a_2(s\theta_2 c\theta_1 - c\theta_2 s\theta_1) + a_1 s\theta_1 \\ 0 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} -d_3 s\theta_{2-1} + a_2 c\theta_{2-1} + a_1 c\theta_1 \\ -d_3 c\theta_{2-1} - a_2 s\theta_{2-1} + a_1 s\theta_1 \\ 0 \\ 1 \end{bmatrix}
\end{aligned} \tag{A.19}$$

Which yields:

$$p_x c\theta_0 + p_y s\theta_0 - a_0 = a_1 c\theta_1 + a_2 c\theta_{2-1} - d_3 s\theta_{2-1} \tag{A.20}$$

$$-p_z = a_1 s\theta_1 - a_2 s\theta_{2-1} - d_3 c\theta_{2-1} \tag{A.21}$$

$$-p_x s\theta_0 + p_y c\theta_0 = 0 \tag{A.22}$$

Equation A.22 yields the solutions:

$$\theta_0 = \phi_0 \tag{A.23}$$

or:

$$\theta_0 = \phi_0 + \pi \tag{A.24}$$

where:

$$\phi_0 = Atan2(p_y, p_x) \tag{A.25}$$

The first solution shall be denoted as the "Front" configuration, while the second solution shall be denoted as the "Back" configuration. This is illustrated in Figure A.2a.

Summing the squares of equations A.20, A.21, and A.22 yields:

$$\begin{aligned}(p_x c\theta_0 + p_y s\theta_0 - a_0)^2 + (-p_z)^2 + (-p_x s\theta_0 + p_y c\theta_0)^2 = \\ (a_1 c\theta_1 + a_2 c\theta_{2-1} - d_3 s\theta_{2-1})^2 + (a_1 s\theta_1 - a_2 s\theta_{2-1} - d_3 c\theta_{2-1})^2\end{aligned} \tag{A.26}$$

Expanding and simplifying equation A.26 yields:

$$p_x^2 + p_y^2 + p_z^2 - 2p_x a_0 c\theta_0 - 2p_y a_0 s\theta_0 + a_0^2 = a_1^2 + a_2^2 + d_3^2 + m_1 + m_2 + m_3 \tag{A.27}$$

where:

$$\begin{aligned} m_1 &= 2a_1a_2(c\theta_1c\theta_{2-1} - s\theta_1s\theta_{2-1}) \\ &= 2a_1a_2(c\theta_{(2+(3-2))}) \\ &= 2a_1a_2c\theta_2 \end{aligned} \tag{A.28}$$

$$\begin{aligned} m_2 &= 2a_1d_3(-s\theta_1c\theta_{2-1} - c\theta_1s\theta_{2-1}) \\ &= -2a_1d_3(s\theta_1c\theta_{2-1} + c\theta_1s\theta_{2-1}) \\ &= -2a_1d_3(s\theta_{(2+(3-2))}) \\ &= -2a_1d_3s\theta_2 \end{aligned} \tag{A.29}$$

$$\begin{aligned} m_3 &= 2a_2d_3(-c\theta_{2-1}s\theta_{2-1} + s\theta_{2-1}c\theta_{2-1}) \\ &= 0 \end{aligned} \tag{A.30}$$

Substituting equations A.28, A.29 and A.30 into equation A.27 yields:

$$p_x^2 + p_y^2 + p_z^2 - 2p_xa_0c\theta_0 - 2p_ya_0s\theta_0 + a_0^2 = a_1^2 + a_2^2 + d_3^2 + 2a_1a_2c\theta_2 - 2a_1d_3s\theta_2 \tag{A.31}$$

Rearranging equation A.31 yields:

$$k_1s\theta_2 + k_2c\theta_2 = k_3 \tag{A.32}$$

where:

$$k_1 = -2a_1d_3 \tag{A.33}$$

$$k_2 = 2a_1a_2 \tag{A.34}$$

$$k_3 = p_x^2 + p_y^2 + p_z^2 - 2p_xa_0c\theta_0 - 2p_ya_0s\theta_0 + a_0^2 - a_1^2 - a_2^2 - d_3^2 \tag{A.35}$$

Dividing both sides of equation A.32 by $\sqrt{k_1^2 + k_2^2}$ yields:

$$s\theta_2 \frac{k_1}{\sqrt{k_1^2+k_2^2}} - c\theta_2 \frac{-k_2}{\sqrt{k_1^2+k_2^2}} = \frac{k_3}{\sqrt{k_1^2+k_2^2}} \tag{A.36}$$

which is equivalent to:

$$s\theta_2 c\phi_2 - c\theta_2 s\phi_2 = s\omega_2 \tag{A.37}$$

where:

$$\phi_2 = Atan2(-k_2, k_1) = tan^{-1}\left(\frac{-k_2}{k_1}\right) = tan^{-1}\left(\frac{-2a_1a_2}{-2a_1d_3}\right) = tan^{-1}\left(\frac{a_2}{d_3}\right) \tag{A.38}$$

$$\omega_2 = sin^{-1}\left(\frac{k_3}{\sqrt{k_1^2+k_2^2}}\right) \tag{A.39}$$

This is equivalent to solving:

$$s(\theta_2 - \phi_2) = s\omega_2 \tag{A.40}$$

which yields:

$$\theta_2 = \phi_2 + \omega_2 \tag{A.41}$$

or:

$$\theta_2 = \phi_2 + \pi - \omega_2 \tag{A.42}$$

If $\left|\frac{k_3}{\sqrt{k_1^2+k_2^2}}\right| > 1$, then the position is unreachable. If $\left|\frac{k_3}{\sqrt{k_1^2+k_2^2}}\right| = 1$, then there is one solution, which corresponds to the robot arm extended for maximum reach. If $\left|\frac{k_3}{\sqrt{k_1^2+k_2^2}}\right| < 1$, then there are two unique solutions. The solution $\theta_2 = \phi_2 + \omega_2$ shall be denoted as the "Upper Arm" configuration, while the solution $\theta_2 = \phi_2 + \pi - \omega_2$ shall be denoted as the "Lower Arm" configuration. This is illustrated in Figure A.2b.

The simplification done in equation A.38 is only accurate because the x component of the Atan2(y,x) function is positive. Specifically, $k_1 > 0$ since $a_1 > 0$ and $d_3 < 0$.

We also note that:

$$\sqrt{k_1^2 + k_2^2} = \sqrt{(-2a_1d_3)^2 + (2a_1a_2)^2} = 2a_1\sqrt{d_3^2 + a_2^2} \tag{A.43}$$

Now that θ_0 and θ_2 are known, θ_1 can be solved for by expanding and rearranging equations A.20 and A.21, which yields:

$$a_1c\theta_1 + a_2c\theta_2c\theta_1 + a_2s\theta_2s\theta_1 - d_3s\theta_2c\theta_1 + d_3c\theta_1s\theta_2 = p_xc\theta_0 + p_ys\theta_0 - a_0 \tag{A.44}$$

$$a_1s\theta_1 - a_2s\theta_2c\theta_1 + a_2c\theta_2s\theta_1 - d_3c\theta_2c\theta_1 - d_3s\theta_2s\theta_1 = -p_z \tag{A.45}$$

Rearranging gives:

$$\mu_0c\theta_1 + \nu_0s\theta_1 = \gamma_0 \tag{A.46}$$

$$\mu_1c\theta_1 + \nu_1s\theta_1 = \gamma_1 \tag{A.47}$$

where:

$$\mu_0 = a_1 + a_2c\theta_2 - d_3s\theta_2 \tag{A.48}$$

$$\nu_0 = a_2s\theta_2 + d_3c\theta_2 \tag{A.49}$$

$$\gamma_0 = p_xc\theta_0 + p_ys\theta_0 - a_0 \tag{A.50}$$

$$\mu_1 = -a_2s\theta_2 - d_3c\theta_2 \tag{A.51}$$

$$\nu_1 = a_1 + a_2 c\theta_2 - d_3 s\theta_2 \tag{A.52}$$

$$\gamma_1 = -p_z \tag{A.53}$$

Solving equations A.46 and A.47 for $c\theta_1$ and $s\theta_1$ yields:

$$c\theta_1 = \frac{\nu_1\gamma_0 - \nu_0\gamma_1}{\nu_1\mu_0 - \nu_0\mu_1} \tag{A.54}$$

$$s\theta_1 = \frac{\mu_0\gamma_1 - \mu_1\gamma_0}{\nu_1\mu_0 - \nu_0\mu_1} \tag{A.55}$$

We note that:

$$\nu_1\mu_0 - \nu_0\mu_1 = (a_1 + a_2 c\theta_2 - d_3 s\theta_2)^2 + (a_2 s\theta_2 + d_3 c\theta_2)^2 \tag{A.56}$$

Thus the quantity $\nu_1\mu_0 - \nu_0\mu_1$ is greater or equal to zero for all values of θ_2. Additionally, with the use of some simple algebraic manipulation, it can be shown that the quantity $\nu_1\mu_0 - \nu_0\mu_1$ is zero if and only if the robot dimensions satisfy the following equation $a_1^2 = a_2^2 + d_3^2$ and joint B is coincident with joint L. This condition is mechanically not possible.

Thus we obtain a unique solution:

$$\theta_1 = Atan2(s\theta_1, c\theta_1) = Atan2((\mu_0\gamma_1 - \mu_1\gamma_0), (\nu_1\gamma_0 - \nu_0\gamma_1)) \tag{A.57}$$

Solution for R, B, and T Joint Angles

Once θ_0, θ_1, and θ_2 are solved for, ${}^0A_3$ is completely known. The remaining joint angles can be found by left multiplying both sides of equation A.11 by $({}^0A_3)^{-1}$:

$${}^3A_6 = ({}^0A_3)^{-1}\,{}^0A_6 \tag{A.58}$$

Following the same procedure for obtaining $({}^0A_1)^{-1}$ in equation A.18, $({}^0A_3)^{-1}$ can be obtained as such:

$$ {}^0A_3 = \begin{bmatrix} {}^0R_3 & {}^0V_3 \\ 0 & 1 \end{bmatrix} $$

$$ ({}^0A_3)^{-1} = \begin{bmatrix} ({}^0R_3)^T & -({}^0R_3)^T\,{}^0V_3 \\ 0 & 1 \end{bmatrix} $$

Where:

$$ \left({}^0R_3\right)^T = \begin{bmatrix} c\theta_0 c\theta_{2-1} & s\theta_0 c\theta_{2-1} & s\theta_{2-1} \\ -s\theta_0 & c\theta_0 & 0 \\ -c\theta_0 s\theta_{2-1} & -s\theta_0 s\theta_{2-1} & c\theta_{2-1} \end{bmatrix} $$

and:

$$ \begin{aligned} -\left({}^0R_3\right)^T {}^0V_3 &= -\begin{bmatrix} c\theta_0 c\theta_{2-1} & s\theta_0 c\theta_{2-1} & s\theta_{2-1} \\ -s\theta_0 & c\theta_0 & 0 \\ -c\theta_0 s\theta_{2-1} & -s\theta_0 s\theta_{2-1} & c\theta_{2-1} \end{bmatrix} \begin{bmatrix} c\theta_0(a_0 + a_1 c\theta_1 + a_2 c\theta_{2-1}) \\ s\theta_0(a_0 + a_1 c\theta_1 + a_2 c\theta_{2-1}) \\ -a_1 s\theta_1 + a_2 s\theta_{2-1} \end{bmatrix} \\ &= -\begin{bmatrix} c\theta_{2-1}(c\theta_0^2 + s\theta_0^2)(a_0 + a_1 c\theta_1 + a_2 c\theta_{2-1}) + s\theta_{2-1}(-a_1 s\theta_1 + a_2 s\theta_{2-1}) \\ 0 \\ -s\theta_{2-1}(c\theta_0^2 + s\theta_0^2)(a_0 + a_1 c\theta_1 + a_2 c\theta_{2-1}) + c\theta_{2-1}(-a_1 s\theta_1 + a_2 s\theta_{2-1}) \end{bmatrix} \\ &= -\begin{bmatrix} a_0 c\theta_{2-1} + a_1 c\theta_1 c\theta_{2-1} + a_2 c\theta_{2-1}^2 - a_1 s\theta_1 s\theta_{2-1} + a_2 s\theta_{2-1}^2 \\ 0 \\ -a_0 s\theta_{2-1} - a_1 c\theta_1 s\theta_{2-1} - a_2 s\theta_{2-1} c\theta_{2-1} - a_1 s\theta_1 c\theta_{2-1} + a_2 s\theta_{2-1} c\theta_{2-1} \end{bmatrix} \\ &= -\begin{bmatrix} a_0 c\theta_{2-1} + a_1 c\theta_{2+(3-2)} + a_2 \\ 0 \\ -a_0 s\theta_{2-1} - a_1 s\theta_{2+(3-2)} \end{bmatrix} \\ &= \begin{bmatrix} -a_0 c\theta_{2-1} - a_1 c\theta_3 - a_2 \\ 0 \\ a_0 s\theta_{2-1} + a_1 s\theta_3 \end{bmatrix} \end{aligned} $$

Thus the inverse of ${}^0A_3$ is:

$$ \left({}^0A_3\right)^{-1} = \begin{bmatrix} c\theta_0 c\theta_{2-1} & s\theta_0 c\theta_{2-1} & s\theta_{2-1} & -a_0 c\theta_{2-1} - a_1 c\theta_3 - a_2 \\ -s\theta_0 & c\theta_0 & 0 & 0 \\ -c\theta_0 s\theta_{2-1} & -s\theta_0 s\theta_{2-1} & c\theta_{2-1} & a_0 s\theta_{2-1} + a_1 s\theta_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(A.59)} $$

Substituting equations A.10, A.59, and A.8 into equation A.58 yields:

$$\begin{bmatrix} c\theta_3c\theta_4c\theta_5 - s\theta_3s\theta_5 & c\theta_3c\theta_4s\theta_5 + s\theta_3c\theta_5 & c\theta_3s\theta_4 & -d_5c\theta_3s\theta_4 \\ s\theta_3c\theta_4c\theta_5 + c\theta_3s\theta_5 & s\theta_3c\theta_4s\theta_5 - c\theta_3c\theta_5 & s\theta_3s\theta_4 & -d_5s\theta_3s\theta_4 \\ s\theta_4c\theta_5 & s\theta_4s\theta_5 & -c\theta_4 & d_5c\theta_4 + d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$
$$\begin{bmatrix} c\theta_0c\theta_{2-1} & s\theta_0c\theta_{2-1} & s\theta_{2-1} & -a_0c\theta_{2-1} - a_1c\theta_3 - a_2 \\ -s\theta_0 & c\theta_0 & 0 & 0 \\ -c\theta_0s\theta_{2-1} & -s\theta_0s\theta_{2-1} & c\theta_{2-1} & a_0s\theta_{2-1} + a_1s\theta_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & w_x & q_x \\ u_y & v_y & w_y & q_y \\ u_z & v_z & w_z & q_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.60}$$

Using equation A.60, if we equate the left and right hand side of the (3,3) (3,1) (3,2) (1,3) and (1,2) elements of the matrices, we obtain:

$$-c\theta_4 = -w_xc\theta_0s\theta_{2-1} - w_ys\theta_0s\theta_{2-1} + w_zc\theta_{2-1} \tag{A.61}$$

$$c\theta_3s\theta_4 = w_xc\theta_0c\theta_{2-1} + w_ys\theta_0c\theta_{2-1} + w_zs\theta_{2-1} \tag{A.62}$$

$$s\theta_3s\theta_4 = -w_xs\theta_0 + w_yc\theta_0 \tag{A.63}$$

$$s\theta_4c\theta_5 = -u_xc\theta_0s\theta_{2-1} - u_ys\theta_0s\theta_{2-1} + u_zc\theta_{2-1} \tag{A.64}$$

$$s\theta_4s\theta_5 = -v_xc\theta_0s\theta_{2-1} - v_ys\theta_0s\theta_{2-1} + v_zc\theta_{2-1} \tag{A.65}$$

Which yields:

$$\theta_4 = \pm cos^{-1}(w_xc\theta_0s\theta_{2-1} + w_ys\theta_0s\theta_{2-1} - w_zc\theta_{2-1}) \tag{A.66}$$

We note that equation A.66 has a positive and a negative solution. The positive solution shall be denoted as the "Flip" configuration, while the negative solution shall be denoted as the "No Flip" configuration. This is illustrated in Figure A.2c.

Unless θ_4 is 0 or π which is a singularity position, the last 2 angles can be computed as follows:

$$c\theta_3 = \frac{w_x c\theta_0 c\theta_{2-1} + w_y s\theta_0 c\theta_{2-1} + w_z s\theta_{2-1}}{s\theta_4} \tag{A.67}$$

$$s\theta_3 = \frac{-w_x s\theta_0 + w_y c\theta_0}{s\theta_4} \tag{A.68}$$

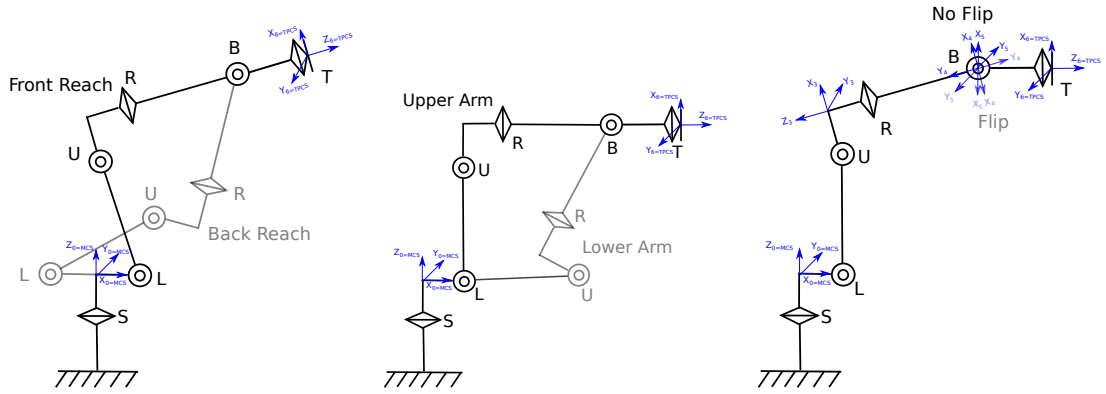
$$\theta_3 = Atan2(s\theta_3, c\theta_3) \tag{A.69}$$

$$c\theta_5 = \frac{-u_x c\theta_0 s\theta_{2-1} - u_y s\theta_0 s\theta_{2-1} + u_z c\theta_{2-1}}{s\theta_4} \tag{A.70}$$

$$s\theta_5 = \frac{-v_x c\theta_0 s\theta_{2-1} - v_y s\theta_0 s\theta_{2-1} + v_z c\theta_{2-1}}{s\theta_4} \tag{A.71}$$

$$\theta_5 = Atan2(s\theta_5, c\theta_5) \tag{A.72}$$

We conclude that there is a total of 8 possible solutions corresponding to the "Front"/"Back", "Upper Arm"/"Lower Arm", and "Flip"/"No Flip" configurations as can be seen in Figure A.2 below.



(a) Front/Back Reach (b) Upper/Lower Arm (c) No Flip/Flip

Figure A.2: Robot configurations

Appendix B

Delta3 Kinematics Derivation[1]

B.1 Definitions and Kinematics Model

$\mathbf{o}$	origin
$\mathbf{h}_i$	hip
$\mathbf{k}_i$	knee
$\mathbf{a}_i$	angle
$\mathbf{p}$	tool position
ϕ_i	leg angle
θ_i	motor angle
r_b	base radius
o_b	base offset
l_b	base leg length
l_p	platform leg length
o_p	platform offset
r_p	platform radius
$\hat{\imath}$	x-direction
$\hat{\jmath}$	y-direction
$\hat{k}$	z-direction

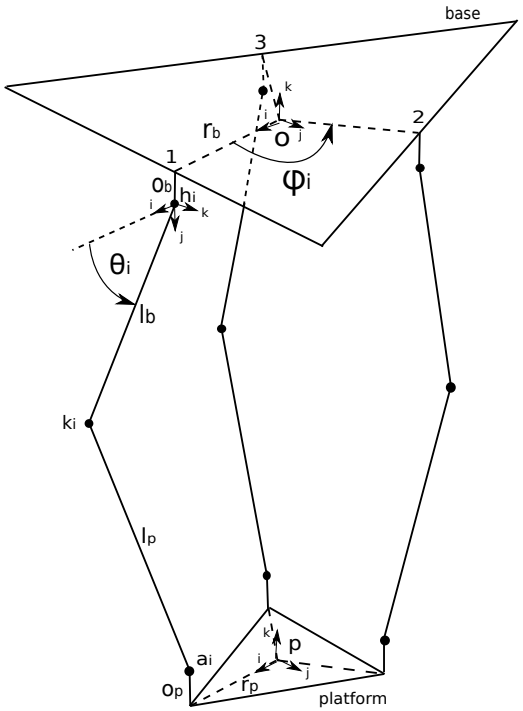


Figure B.1: Delta3 kinematics model

[1]Authored by Dr. Fernando L. Garcia Bermudez at Yaskawa America, Inc.

B.2 Equations of Motion

$$
\begin{aligned}
l_p &= \|\mathbf{a}_i - \mathbf{k}_i\|_2 \\
&= \left\| \mathbf{p} + \begin{bmatrix} r_p \cos\phi_i \\ r_p \sin\phi_i \\ o_p \end{bmatrix} - \mathbf{h}_i - R_{\hat{k}}(-\phi_i) R_{\hat{i}}(90°) \begin{bmatrix} l_b \cos\theta_i \\ l_b \sin\theta_i \\ 0 \end{bmatrix} \right\|_2 \\
&= \left\| \mathbf{p} + \begin{bmatrix} r_p \cos\phi_i \\ r_p \sin\phi_i \\ o_p \end{bmatrix} - \begin{bmatrix} r_b \cos\phi_i \\ r_b \sin\phi_i \\ -o_b \end{bmatrix} - \begin{bmatrix} \cos\phi_i & -\sin\phi_i & 0 \\ \sin\phi_i & \cos\phi_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} l_b \cos\theta_i \\ l_b \sin\theta_i \\ 0 \end{bmatrix} \right\|_2 \\
&= \left\| \begin{bmatrix} p_{\hat{i}} + (r_p - r_b)\cos\phi_i - l_b \cos\phi_i \cos\theta_i \\ p_{\hat{j}} + (r_p - r_b)\sin\phi_i - l_b \sin\phi_i \cos\theta_i \\ p_{\hat{k}} + o_p + o_b + l_b \sin\theta_i \end{bmatrix} \right\|_2
\end{aligned}
$$

Defining $r = r_p - r_b$ and $o = o_p + o_b$ and writing out the ℓ^2-norm we get:

$$
\begin{aligned}
l_p^2 &= (p_{\hat{i}} + r\cos\phi_i - l_b \cos\phi_i \cos\theta_i)^2 + (p_{\hat{j}} + r\sin\phi_i - l_b \sin\phi_i \cos\theta_i)^2 + (p_{\hat{k}} + o + l_b \sin\theta_i)^2 \\
&= p_{\hat{i}}^2 + 2r\cos\phi_i p_{\hat{i}} - 2l_b \cos\phi_i \cos\theta_i p_{\hat{i}} + r^2 \cancel{\cos^2\phi_i} - 2l_b r \cancel{\cos^2\phi_i} \cos\theta_i + l_b^2 \cancel{\cos^2\phi_i \cos^2\theta_i} + \\
&\quad p_{\hat{j}}^2 + 2r\sin\phi_i p_{\hat{j}} - 2l_b \sin\phi_i \cos\theta_i p_{\hat{j}} + r^2 \cancel{\sin^2\phi_i} - 2l_b r \cancel{\sin^2\phi_i} \cos\theta_i + l_b^2 \cancel{\sin^2\phi_i \cos^2\theta_i} + \\
&\quad p_{\hat{k}}^2 + 2o p_{\hat{k}} + 2l_b \sin\theta_i p_{\hat{k}} + o^2 + 2l_b o \sin\theta_i + l_b^2 \cancel{\sin^2\theta_i} \\
0 &= p_{\hat{i}}^2 + p_{\hat{j}}^2 + p_{\hat{k}}^2 + r^2 + o^2 + l_b^2 - l_p^2 + \\
&\quad 2\left(r\left(\cos\phi_i p_{\hat{i}} + \sin\phi_i p_{\hat{j}}\right) + o p_{\hat{k}} - l_b\left(\left(\cos\phi_i p_{\hat{i}} + \sin\phi_i p_{\hat{j}} + r\right)\cos\theta_i - \left(p_{\hat{k}} + o\right)\sin\theta_i\right)\right)
\end{aligned}
$$

Further simplifying by defining $q = r^2 + o^2 + l_b^2 - l_p^2$ we get:

$$
\begin{aligned}
0 = {} & p_{\hat{i}}^2 + p_{\hat{j}}^2 + p_{\hat{k}}^2 + q \\
& + 2(r(\cos\phi_i p_{\hat{i}} + \sin\phi_i p_{\hat{j}}) + o p_{\hat{k}} \\
& - l_b((\cos\phi_i p_{\hat{i}} + \sin\phi_i p_{\hat{j}} + r)\cos\theta_i - (p_{\hat{k}} + o)\sin\theta_i))
\end{aligned}
\tag{B.1}
$$

B.3 Position

Inverse

Defining $u = \cos\phi_i p_{\hat{i}} + \sin\phi_i p_{\hat{j}}$, $v = p_{\hat{i}}^2 + p_{\hat{j}}^2 + p_{\hat{k}}^2 + q + 2(ru + o p_{\hat{k}})$, and $w = 2l_b(u + r)$ for brevity:

$$
0 = v - w\cos\theta_i + 2l_b(p_{\hat{k}} + o)\sin\theta_i
$$

We then apply the tangent half-angle formula where $\cos\theta_i = \frac{1-\tan^2\frac{\theta_i}{2}}{1+\tan^2\frac{\theta_i}{2}}$ and $\sin\theta_i = \frac{2\tan\frac{\theta_i}{2}}{1+\tan^2\frac{\theta_i}{2}}$ and multiply the equation by the common denominator:

$$\begin{aligned} 0 &= v\left(1+\tan^2\frac{\theta_i}{2}\right) - w\left(1-\tan^2\frac{\theta_i}{2}\right) + 4l_b\left(p_{\hat{k}}+o\right)\tan\frac{\theta_i}{2} \\ &= (v+w)\tan^2\frac{\theta_i}{2} + 4l_b\left(p_{\hat{k}}+o\right)\tan\frac{\theta_i}{2} + (v-w) \end{aligned}$$

The above quadratic equation with $a = v+w$, $b = 4l_b\left(p_{\hat{k}}+o\right)$, and $c = v-w$ yields:

$$\theta_i = 2\tan^{-1}\left(\frac{-b \pm \sqrt{b^2-4ac}}{2a}\right) \tag{B.2}$$

Forward

Starting from (B.1), we define $k = p_{\hat{i}}^2 + p_{\hat{j}}^2 + p_{\hat{k}}^2 + q$ and rearrange it as:

$$0 = k+2\left(\left(r-l_b\cos\theta_i\right)\left(\cos\phi_i p_{\hat{i}} + \sin\phi_i p_{\hat{j}}\right) + \left(o+l_b\sin\theta_i\right)p_{\hat{k}} - l_b\left(r\cos\theta_i - o\sin\theta_i\right)\right)$$

Writing it out for all values of i:

$$0 = k+2\left(\left(r-l_b\cos\theta_1\right)\left(\cos\phi_1 p_{\hat{i}} + \sin\phi_1 p_{\hat{j}}\right) + \left(o+l_b\sin\theta_1\right)p_{\hat{k}} - l_b\left(r\cos\theta_1 - o\sin\theta_1\right)\right) \tag{B.3}$$

$$0 = k+2\left(\left(r-l_b\cos\theta_2\right)\left(\cos\phi_2 p_{\hat{i}} + \sin\phi_2 p_{\hat{j}}\right) + \left(o+l_b\sin\theta_2\right)p_{\hat{k}} - l_b\left(r\cos\theta_2 - o\sin\theta_2\right)\right) \tag{B.4}$$

$$0 = k+2\left(\left(r-l_b\cos\theta_3\right)\left(\cos\phi_3 p_{\hat{i}} + \sin\phi_3 p_{\hat{j}}\right) + \left(o+l_b\sin\theta_3\right)p_{\hat{k}} - l_b\left(r\cos\theta_3 - o\sin\theta_3\right)\right) \tag{B.5}$$

Combining (B.3) and (B.4):

$$\begin{aligned} &\left(\left(r-l_b\cos\theta_2\right)\cos\phi_2 - \left(r-l_b\cos\theta_1\right)\cos\phi_1\right)p_{\hat{i}} + \left(\left(r-l_b\cos\theta_2\right)\sin\phi_2 - \left(r-l_b\cos\theta_1\right)\sin\phi_1\right)p_{\hat{j}} = \\ &\qquad l_b\left(r\left(\cos\theta_2-\cos\theta_1\right) - o\left(\sin\theta_2-\sin\theta_1\right)\right) - \left(o+l_b\sin\theta_2 - o - l_b\sin\theta_1\right)p_{\hat{k}} \end{aligned} \tag{B.6}$$

Combining (B.3) and (B.5):

$$\begin{aligned} &\left(\left(r-l_b\cos\theta_3\right)\cos\phi_3 - \left(r-l_b\cos\theta_1\right)\cos\phi_1\right)p_{\hat{i}} + \left(\left(r-l_b\cos\theta_3\right)\sin\phi_3 - \left(r-l_b\cos\theta_1\right)\sin\phi_1\right)p_{\hat{j}} = \\ &\qquad l_b\left(r\left(\cos\theta_3-\cos\theta_1\right) - o\left(\sin\theta_3-\sin\theta_1\right)\right) - \left(o+l_b\sin\theta_3 - o - l_b\sin\theta_1\right)p_{\hat{k}} \end{aligned} \tag{B.7}$$

We can rewrite (B.6) and (B.7) in matrix form by defining:

$$A = \begin{bmatrix} (r - l_b\cos\theta_2)\cos\phi_2 - (r - l_b\cos\theta_1)\cos\phi_1 & (r - l_b\cos\theta_2)\sin\phi_2 - (r - l_b\cos\theta_1)\sin\phi_1 \\ (r - l_b\cos\theta_3)\cos\phi_3 - (r - l_b\cos\theta_1)\cos\phi_1 & (r - l_b\cos\theta_3)\sin\phi_3 - (r - l_b\cos\theta_1)\sin\phi_1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

Where its inverse is defined as:

$$A^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}$$

Thus,

$$\begin{bmatrix} p_{\hat{i}} \\ p_{\hat{j}} \end{bmatrix} = l_b A^{-1} \left(r \begin{bmatrix} \cos\theta_2 - \cos\theta_1 \\ \cos\theta_3 - \cos\theta_1 \end{bmatrix} - o \begin{bmatrix} \sin\theta_2 - \sin\theta_1 \\ \sin\theta_3 - \sin\theta_1 \end{bmatrix} \right) - l_b A^{-1} \begin{bmatrix} \sin\theta_2 - \sin\theta_1 \\ \sin\theta_3 - \sin\theta_1 \end{bmatrix} p_{\hat{k}} \tag{B.8}$$

Defining $\mathbf{m}$ and $\mathbf{n}$ such that:

$$\begin{bmatrix} p_{\hat{i}} \\ p_{\hat{j}} \end{bmatrix} = \mathbf{m} - \mathbf{n}p_{\hat{k}} = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} - \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} p_{\hat{k}}$$

and substituting these into (B.3):

$$\begin{aligned} 0 = {} & (m_1 - n_1 p_{\hat{k}})^2 + (m_2 - n_2 p_{\hat{k}})^2 + p_{\hat{k}}^2 + q + \\ & 2\left((r - l_b\cos\theta_1)\left(\cos\phi_1\left(m_1 - n_1 p_{\hat{k}}\right) + \sin\phi_1\left(m_2 - n_2 p_{\hat{k}}\right)\right) + \right. \\ & \left. \left(o + l_b\sin\theta_1\right) p_{\hat{k}} - l_b\left(r\cos\theta_1 - o\sin\theta_1\right)\right) \\ = {} & m_1^2 - 2m_1 n_1 p_{\hat{k}} + n_1^2 p_{\hat{k}}^2 + m_2^2 - 2m_2 n_2 p_{\hat{k}} + n_2^2 p_{\hat{k}}^2 + p_{\hat{k}}^2 + q \ - 2l_b\left(r\cos\theta_1 - o\sin\theta_1\right) + \\ & 2\left((r - l_b\cos\theta_1)\left(\cos\phi_1 m_1 - \cos\phi_1 n_1 p_{\hat{k}} + \sin\phi_1 m_2 - \sin\phi_1 n_2 p_{\hat{k}}\right) + \left(o + l_b\sin\theta_1\right) p_{\hat{k}}\right) \\ = {} & \left(1 + n_1^2 + n_2^2\right) p_{\hat{k}}^2 - 2\left(m_1 n_1 + m_2 n_2 + (r - l_b\cos\theta_1)\left(\cos\phi_1 n_1 + \sin\phi_1 n_2\right) - o - l_b\sin\phi_1\right) p_{\hat{k}} + \\ & m_1^2 + m_2^2 + q + 2\left((r - l_b\cos\theta_1)\left(\cos\phi_1 m_1 + \sin\phi_1 m_2\right) - l_b\left(r\cos\theta_1 - o\sin\theta_1\right)\right) \end{aligned}$$

With quadratic coefficients defined as such:

$$\begin{aligned} a &= 1 + n_1^2 + n_2^2 \\ b &= -2\left(m_1 n_1 + m_2 n_2 + (r - l_b\cos\theta_1)\left(\cos\phi_1 n_1 + \sin\phi_1 n_2\right) - o - l_b\sin\phi_1\right) \\ c &= m_1^2 + m_2^2 + q + 2\left((r - l_b\cos\theta_1)\left(\cos\phi_1 m_1 + \sin\phi_1 m_2\right) - l_b\left(r\cos\theta_1 - o\sin\theta_1\right)\right) \end{aligned}$$

We can solve for $p_{\hat{k}}$ using:

$$p_{\hat{k}} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \text{ where we choose the } p_{\hat{k}} < 0 \tag{B.9}$$

B.4 Velocity

$$
\begin{aligned}
0 &= p_{\hat{i}}^2 + p_{\hat{j}}^2 + p_{\hat{k}}^2 + q + 2\left(r\left(\cos\phi_i p_{\hat{i}} + \sin\phi_i p_{\hat{j}}\right) + o p_{\hat{k}}\right) \\
&\quad - 2l_b\left(\left(\cos\phi_i p_{\hat{i}} + \sin\phi_i p_{\hat{j}} + r\right)\cos\theta_i - \left(p_{\hat{k}} + o\right)\sin\theta_i\right) \\
&= 2p_{\hat{i}}\dot{p}_{\hat{i}} + 2p_{\hat{j}}\dot{p}_{\hat{j}} + 2p_{\hat{k}}\dot{p}_{\hat{k}} + 2\left(r\left(\cos\phi_i\dot{p}_{\hat{i}} + \sin\phi_i\dot{p}_{\hat{j}}\right) + o\dot{p}_{\hat{k}}\right) \\
&\quad - 2l_b\left(\left(\cos\phi_i\dot{p}_{\hat{i}} + \sin\phi_i\dot{p}_{\hat{j}}\right)\cos\theta_i - \left(\cos\phi_i p_{\hat{i}} + \sin\phi_i p_{\hat{j}} + r\right)\sin\theta_i\dot{\theta}_i - \dot{p}_{\hat{k}}\sin\theta_i - \left(p_{\hat{k}} + o\right)\cos\theta_i\dot{\theta}_i\right) \\
&= \left(p_{\hat{i}} + r\cos\phi_i - l_b\cos\phi_i\cos\theta_i\right)\dot{p}_{\hat{i}} + \left(p_{\hat{j}} + r\sin\phi_i - l_b\sin\phi_i\cos\theta_i\right)\dot{p}_{\hat{j}} + \left(p_{\hat{k}} + o + l_b\sin\theta_i\right)\dot{p}_{\hat{k}} \\
&\quad - l_b\left(-\left(p_{\hat{k}} + o\right)\cos\theta_i - \left(\cos\phi_i p_{\hat{i}} + \sin\phi_i p_{\hat{j}} + r\right)\sin\theta_i\right)\dot{\theta}_i \\
&= J_p\dot{\mathbf{p}} - J_\theta\dot{\boldsymbol{\theta}}
\end{aligned}
$$

Inverse

$$\dot{\boldsymbol{\theta}} = J_\theta^{-1} J_p\dot{\mathbf{p}} \tag{B.10}$$

Forward

$$\dot{\mathbf{p}} = J_p^{-1} J_\theta\dot{\boldsymbol{\theta}} \tag{B.11}$$

B.5 Acceleration

$$
\begin{aligned}
0 = &\left(\dot{p}_{\hat{i}} + l_b\cos\phi_i\sin\theta_i\dot{\theta}_i\right)\dot{p}_{\hat{i}} + \left(\dot{p}_{\hat{j}} + l_b\sin\phi_i\sin\theta_i\dot{\theta}_i\right)\dot{p}_{\hat{j}} + \left(\dot{p}_{\hat{k}} + l_b\cos\theta_i\dot{\theta}_i\right)\dot{p}_{\hat{k}} \\
&+ \left(p_{\hat{i}} + r\cos\phi_i - l_b\cos\phi_i\cos\theta_i\right)\ddot{p}_{\hat{i}} + \left(p_{\hat{j}} + r\sin\phi_i - l_b\sin\phi_i\cos\theta_i\right)\ddot{p}_{\hat{j}} + \left(p_{\hat{k}} + o + l_b\sin\theta_i\right)\ddot{p}_{\hat{k}} \\
&- l_b\left(-\dot{p}_{\hat{k}}\cos\theta_i + \left(p_{\hat{k}} + o\right)\sin\theta_i\dot{\theta}_i - \left(\cos\phi_i\dot{p}_{\hat{i}} + \sin\phi_i\dot{p}_{\hat{j}}\right)\sin\theta_i - \left(\cos\phi_i p_{\hat{i}} + \sin\phi_i p_{\hat{j}} + r\right)\cos\theta_i\dot{\theta}_i\right)\dot{\theta}_i \\
&- l_b\left(-\left(p_{\hat{k}} + o\right)\cos\theta_i - \left(\cos\phi_i p_{\hat{i}} + \sin\phi_i p_{\hat{j}} + r\right)\sin\theta_i\right)\ddot{\theta}_i \\
&= \dot{J}_p\dot{\mathbf{p}} + J_p\ddot{\mathbf{p}} - \dot{J}_\theta\dot{\boldsymbol{\theta}} - J_\theta\ddot{\boldsymbol{\theta}}
\end{aligned}
$$

Inverse

$$\ddot{\boldsymbol{\theta}} = J_\theta^{-1}\left(\dot{J}_p\dot{\mathbf{p}} + J_p\ddot{\mathbf{p}} - \dot{J}_\theta\dot{\boldsymbol{\theta}}\right) \tag{B.12}$$

Forward

$$\ddot{\mathbf{p}} = J_p^{-1}\left(\dot{J}_\theta\dot{\boldsymbol{\theta}} + J_\theta\ddot{\boldsymbol{\theta}} - \dot{J}_p\dot{\mathbf{p}}\right) \tag{B.13}$$

Appendix C

Delta2 Kinematics Derivation

C.1 Kinematics Model

This document goes over the forward and inverse kinematics derivation of a Delta2 robot. Figure C.1 below has assigned frames of references that match Yaskawa conventions. Additionally, joint angle measurements match Yaskawa conventions:

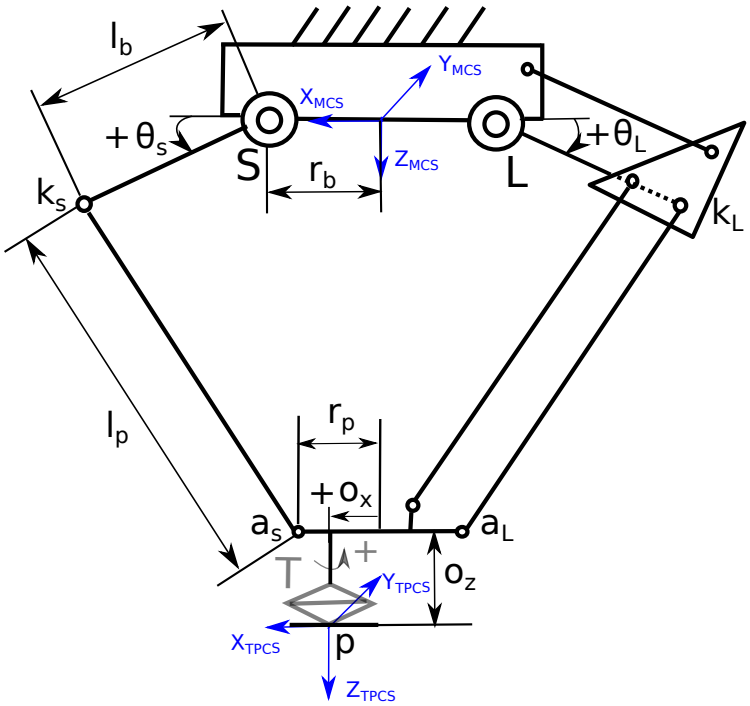


Figure C.1: Delta2 kinematic model

C.2 Equations of Motion

We denote p as the position of the TPCS (Tool Plate Coordinate System) relative to the MCS (Machine Coordinate System). We therefore have:

$$p = \begin{bmatrix} p_x \\ p_z \end{bmatrix}$$

Given that the TPCS remains horizontal due to mechanical constraints, and given that link l_p has fixed length, we obtain the equations:

$$\begin{aligned} l_p &= \|\mathbf{a}_S - \mathbf{k}_S\|_2 \\ &= \left\| \begin{bmatrix} p_x - o_x + r_p \\ p_z - o_z \end{bmatrix} - \begin{bmatrix} r_b + l_b cos(\theta_s) \\ l_b sin(\theta_s) \end{bmatrix} \right\|_2 \\ &= \left\| \begin{bmatrix} p_x - o_x + (r_p - r_b) - l_b \cos(\theta_S) \\ p_z - o_z - l_b \sin(\theta_S) \end{bmatrix} \right\|_2 \end{aligned}$$

$$\begin{aligned} l_p &= \|\mathbf{a}_L - \mathbf{k}_L\|_2 \\ &= \left\| \begin{bmatrix} p_x - o_x - r_p \\ p_z - o_z \end{bmatrix} - \begin{bmatrix} -r_b - l_b cos(\theta_L) \\ l_b sin(\theta_L) \end{bmatrix} \right\|_2 \\ &= \left\| \begin{bmatrix} p_x - o_x - (r_p - r_b) + l_b \cos(\theta_L) \\ p_z - o_z - l_b \sin(\theta_L) \end{bmatrix} \right\|_2 \end{aligned}$$

Defining $x = p_x - o_x$, $z = p_z - o_z$, and $r = r_p - r_b$ and writing out the ℓ^2-norm we obtain:

$$l_p^2 = ((x + r) - l_b \cos(\theta_S))^2 + (z - l_b \sin(\theta_S))^2 \tag{C.1}$$

$$l_p^2 = ((x - r) + l_b \cos(\theta_L))^2 + (z - l_b \sin(\theta_L))^2 \tag{C.2}$$

C.3 Position

Inverse

Expanding equations C.1 and C.2 we obtain:

$$l_p^2 = (x+r)^2 - 2(x+r)l_b\cos(\theta_S) + l_b^2\cos(\theta_S)^2 + z^2 - 2zl_b\sin(\theta_S) + l_b^2\sin(\theta_S)^2 \quad \text{(C.3)}$$

$$l_p^2 = (x-r)^2 + 2(x-r)l_b\cos(\theta_L) + l_b^2\cos(\theta_L)^2 + z^2 - 2zl_b\sin(\theta_L) + l_b^2\sin(\theta_L)^2 \quad \text{(C.4)}$$

Simplifying and rearranging we obtain:

$$z\sin(\theta_S) + (r+x)\cos(\theta_S) = \frac{(r+x)^2 + z^2 + l_b^2 - l_p^2}{2l_b} \quad \text{(C.5)}$$

$$z\sin(\theta_L) + (r-x)\cos(\theta_L) = \frac{(r-x)^2 + z^2 + l_b^2 - l_p^2}{2l_b} \quad \text{(C.6)}$$

Equations C.5 and C.6 are of the form:

$$k_1 s\theta_{S,L} + k_2 c\theta_{S,L} = k_3 \quad \text{(C.7)}$$

where:

$$k_1 = z; \qquad k_2 = r \pm x; \qquad k_3 = \frac{(r \pm x)^2 + z^2 + l_b^2 - l_p^2}{2l_b}$$

We now scale the quantities k_1 and k_2 in equation C.7 by $\frac{1}{\sqrt{k_1^2+k_2^2}}$ such that they represent a sin and a cos of a given angle, this yields:

$$sin(\theta_2)\frac{k_1}{\sqrt{k_1^2+k_2^2}} + cos(\theta_2)\frac{k_2}{\sqrt{k_1^2+k_2^2}} = \frac{k_3}{\sqrt{k_1^2+k_2^2}}$$

which is equivalent to:

$$sin(\theta_{S,L})cos(\phi_{S,L}) + cos(\theta_{S,L})sin(\phi_{S,L}) = sin(\omega_{S,L})$$

where:

$$\phi_{S,L} = Atan2(k_2, k_1) = Atan2(r \pm x, z)$$

$$\omega_{S,L} = sin^{-1}\left(\frac{k_3}{\sqrt{k_1^2 + k_2^2}}\right)$$

This is equivalent to solving:

$$sin(\theta_{S,L} + \phi_{S,L}) = sin(\omega_{S,L})$$

which yields:

$$\theta_{S,L} = -\phi_{S,L} + \omega_{S,L}$$

or:

$$\theta_{S,L} = -\phi_{S,L} + \pi - \omega_{S,L}$$

If $\left|\frac{k_3}{\sqrt{k_1^2+k_2^2}}\right| > 1$, then the position is unreachable. If $\left|\frac{k_3}{\sqrt{k_1^2+k_2^2}}\right| = 1$, then there is one solution, which corresponds to the robot leg extended for maximum reach. If $\left|\frac{k_3}{\sqrt{k_1^2+k_2^2}}\right| < 1$, then there are two unique solutions. The solution $\theta_{S,L} = -\phi_{S,L} + \omega_{S,L}$ shall be denoted as the "Knee Out" configuration, while the solution $\theta_{S,L} = \phi_{S,L} + \pi - \omega_{S,L}$ shall be denoted as the "Knee In" configuration. This is illustrated in Figure C.2. We note that Knee In configuration for leg L is mechanically not reachable. Additionally, delta2 mechanisms are normally operated while both legs are in Knee Out configuration.

Forward

Taking the difference of equations C.5 and C.6 we obtain:

$$z(sin(\theta_S) - sin(\theta_L)) + r(cos(\theta_S) - cos(\theta_L)) + x(cos(\theta_S) + cos(\theta_L)) = \frac{2rx}{l_b}$$

which yields:

$$x = mz + n \tag{C.8}$$

where:

$$m = \frac{l_b(sin(\theta_S) - sin(\theta_L))}{2r - l_b(cos(\theta_S) + cos(\theta_L))}; \qquad n = \frac{l_b(cos(\theta_S) - cos(\theta_L))}{2r - l_b(cos(\theta_S) + cos(\theta_L))}r$$

Substituting (C.8) into (C.5) yields:

$$z\sin(\theta_S) + (r + mz + n)\cos(\theta_S) = \frac{(r + mz + n)^2 + z^2 + l_b^2 - l_p^2}{2l_b}$$

which yields:

$$az^2 + bz + c = 0$$

such that:

$$a = 1 + m^2$$

$$b = 2m(r + n) - 2l_b(mcos(\theta_S) + sin(\theta_S))$$

$$c = (r + n)^2 + l_b^2 - l_p^2 - 2l_b(r + n)cos(\theta_S)$$

which yields:

$$z = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

We note that although we obtain two solutions for z, we will only accept the one using the positive sign. The reason the second solution was introduced is because the original equations of motion were squared to yield equations C.1 and C.2. Squaring the equations was necessary for eliminating the square root, but lead to introducing a second solution for z which is always negative. As seen from figure C.1, z is never negative.

C.4 Velocity

Differentiating equations C.1 and C.2, and rearranging into matrix form, we obtain:

$$0 = 2((x+r) - l_b cos(\theta_S))(\dot{x} + l_b sin(\theta_S)\dot{\theta_S}) + 2(z - l_b sin(\theta_S))(\dot{z} - l_b cos(\theta_S)\dot{\theta_S})$$

$$0 = 2((x-r) + l_b cos(\theta_L))(\dot{x} - l_b sin(\theta_L)\dot{\theta_L}) + 2(z - l_b sin(\theta_L))(\dot{z} - l_b cos(\theta_L)\dot{\theta_L})$$

which can be rearranged into:

$$0 = J_p\dot{p} - J_\theta\dot{\theta} \tag{C.9}$$

such that:

$$\dot{p} = \begin{bmatrix}\dot{x}\\ \dot{z}\end{bmatrix}, \dot{\theta} = \begin{bmatrix}\dot{\theta_S}\\ \dot{\theta_L}\end{bmatrix}$$

$$J_p = \begin{bmatrix}x + r - l_b cos(\theta_S) & z - l_b sin(\theta_S)\\ x - r + l_b cos(\theta_L) & z - l_b sin(\theta_L)\end{bmatrix}$$

$$J_\theta = \begin{bmatrix}J_{\theta 1} & J_{\theta 2}\end{bmatrix}$$

$$J_{\theta 1} = \begin{bmatrix}-(x + r - l_b cos(\theta_S))l_b sin(\theta_S) + (z - l_b sin(\theta_S))l_b cos(\theta_S)\\ 0\end{bmatrix}$$

$$J_{\theta 2} = \begin{bmatrix}0\\ (x - r + l_b cos(\theta_L))l_b sin(\theta_L) + (z - l_b sin(\theta_L))l_b cos(\theta_L)\end{bmatrix}$$

Forward

$$\dot{p} = J_p^{-1} J_\theta \dot{\theta} \tag{C.10}$$

We note that the Jacobian matrix $J = J_p^{-1} J_\theta$ is by definition the matrix that takes us from joint velocities to world velocities.

Inverse

$$\dot{\theta} = J_\theta^{-1} J_p \dot{p} \tag{C.11}$$

C.5 Acceleration

Differentiating equation C.9 once more yields:

$$0 = \dot{J}_p \dot{p} + J_p \ddot{p} - \dot{J}_\theta \dot{\theta} - J_\theta \ddot{\theta} \tag{C.12}$$

$$\dot{J}_p = \begin{bmatrix} \dot{x} + l_b sin(\theta_S)\dot{\theta_S} & \dot{z} - l_b cos(\theta_S)\dot{\theta_S} \\ \dot{x} - l_b sin(\theta_L)\dot{\theta_L} & \dot{z} - l_b cos(\theta_L)\dot{\theta_L} \end{bmatrix}$$

$$\dot{J}_\theta = \begin{bmatrix} \dot{J_{\theta 1}} & \dot{J_{\theta 2}} \end{bmatrix}$$

$$\begin{aligned} \dot{J_{\theta 1}} &= \begin{bmatrix} -(x + r - l_b cos(\theta_S)) l_b cos(\theta_S)\dot{\theta_S} - (\dot{x} + l_b sin(\theta_S)\dot{\theta_S}) l_b sin(\theta_S) \\ -(z - l_b sin(\theta_S)) l_b sin(\theta_S)\dot{\theta_S} + (\dot{z} - l_b cos(\theta_S)\dot{\theta_S}) l_b cos(\theta_S) \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} -(x + r) l_b cos(\theta_S)\dot{\theta_S} - \dot{x} l_b sin(\theta_S) - z l_b sin(\theta_S)\dot{\theta_S} + \dot{z} l_b cos(\theta_S) \\ 0 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \dot{J_{\theta 2}} &= \begin{bmatrix} 0 \\ (x - r + l_b cos(\theta_L)) l_b cos(\theta_L)\dot{\theta_L} + (\dot{x} - l_b sin(\theta_L)\dot{\theta_L}) l_b sin(\theta_L) \\ -(z - l_b sin(\theta_L)) l_b sin(\theta_L)\dot{\theta_L} + (\dot{z} - l_b cos(\theta_L)\dot{\theta_L}) l_b cos(\theta_L) \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ (x - r) l_b cos(\theta_L)\dot{\theta_L} + \dot{x} l_b sin(\theta_L) - z l_b sin(\theta_L)\dot{\theta_L} + \dot{z} l_b cos(\theta_L) \end{bmatrix} \end{aligned}$$

Forward

$$\ddot{p} = J_p^{-1}(-\dot{J}_p\dot{p} + \dot{J}_\theta\dot{\theta} + J_\theta\ddot{\theta}) \tag{C.13}$$

Inverse

$$\ddot{\theta} = J_\theta^{-1}(\dot{J}_p\dot{p} + J_p\ddot{p} - \dot{J}_\theta\dot{\theta}) \tag{C.14}$$

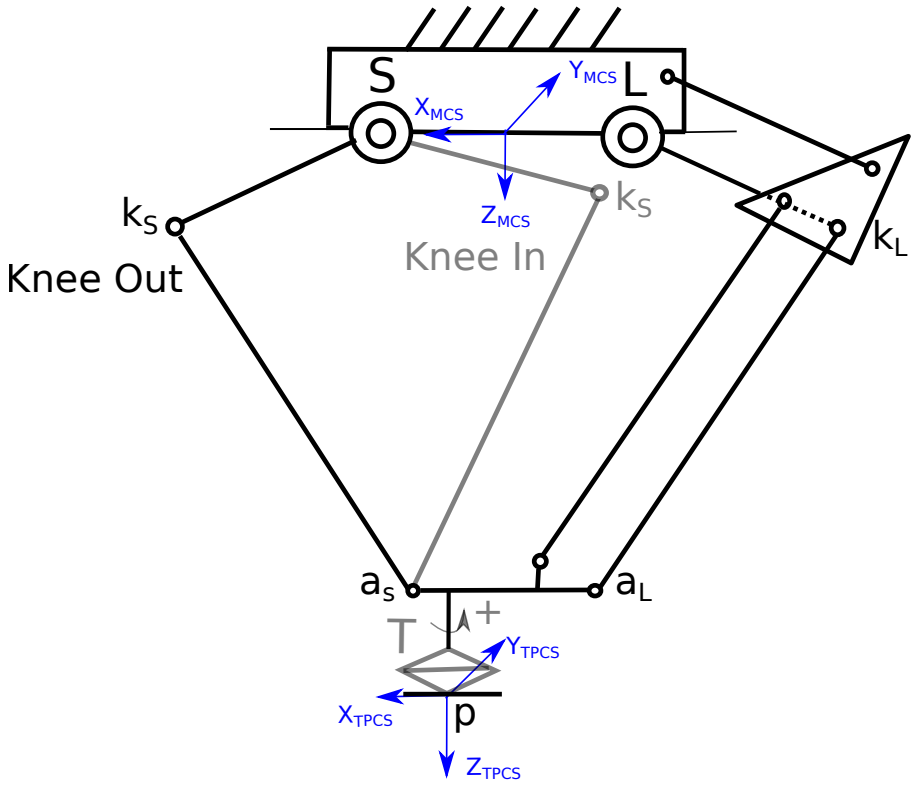


Figure C.2: Knee Out vs. Knee In configurations