2014

Reconfigurable Validation Model for Identifying Kinematic Singularities and Reach Conditions for Articulated Robots and Machine Tools

Luv Aggarwal
University of Windsor

Reconfigurable Validation Model for Identifying Kinematic Singularities and Reach Conditions for Articulated Robots and Machine Tools

By

Luv Aggarwal

A Thesis

Submitted to the Faculty of Graduate Studies
through the Department of Mechanical, Automotive and Materials Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science
at the University of Windsor

Windsor, Ontario, Canada

2014

Reconfigurable Validation Model for Identifying Kinematic Singularities and Reach Conditions for Articulated Robots and Machine Tools

By

Luv Aggarwal

APPROVED BY:

_____
Dr. Z. Pasek
Industrial and Manufacturing Systems Engineering

_____
Dr. J. Johrendt
Mechanical, Automotive and Materials Engineering

_____
Dr. R. J. Urbanic, Advisor
Mechanical, Automotive and Materials Engineering

June 23, 2014

DECLARATION OF CO-AUTHORSHIP / PREVIOUS PUBLICATION

I. Co-Authorship Declaration

I hereby declare that this thesis incorporates material that is result of joint research, as follows:

This thesis incorporates the outcome of a joint research undertaken in collaboration with Kush Aggarwal under the supervision of Dr. Jill Urbanic. The collaboration is covered in Chapter 9 of the thesis. In all cases, the key ideas, primary contributions, experimental designs, data analysis and interpretation, were performed by the author, and the contribution of co-authors was primarily through the provision of comments, thoughts, and suggestions for improvement to the quality of work.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Declaration of Previous Publication

This thesis includes concepts, methodology, and excerpt(s) from 2 original papers that have been previously published in peer reviewed journals, as follows:

Thesis Chapter	Publication title/full citation	Publication status*
Chapter(s) 1,2,5,8	▪ Aggarwal, L., Urbanic, R., and Aggarwal, K., "A Reconfigurable Algorithm for Identifying and Validating Functional Workspace of Industrial Manipulators," SAE Technical Paper 2014-01-0734, 2014, doi:10.4271/2014-01-0734.	Published

Chapter(s) 2,6,8,10	▪ Aggarwal, L., Aggarwal, K., and Urbanic, R. J. (2014). Use of artificial neural networks for the development of an inverse kinematic solution and visual identification of singularity zone(s). 47th CIRP Conference on Manufacturing Systems. Windsor: Elsevier Ltd. doi: 10.1016/j.procir.2014.01.107	Published

I certify that I have obtained a written permission (Appendices E, F) from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as graduate student at the University of Windsor.

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Automation has led to industrial robots facilitating a wide array of high speed, endurance, and precision operations undertaken in the manufacturing industry today. An acceptable level of functioning and control is therefore vital to the efficacy and successful implementation of such manipulators. This research presents a comprehensive analytical tool for downstream optimization of manipulator design, functionality, and performance. The proposed model is reconfigurable and allows for modelling and validation of different industrial robots. Unique 3D visual models for a manipulator workspace and kinematic singularities are developed to gain an understanding into the task space and reach conditions of the manipulator's end-effector. The developed algorithm also presents a non-conventional and computationally inexpensive solution to the inverse kinematics problem through the use Artificial Neural Networks. Application of the proposed technique is further extended to aid in development of path planning models for a uniform, continuous, and singularity free motion.

To my parents,

for their unconditional love, endless support, encouragement

and

for their sacrifices in providing me with a better future.

Thank you for giving so selflessly.

ACKNOWLEDGEMENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS / SYMBOLS

Abbreviations

ANN	Artificial neural network
CAD	Computer-aided design
CNC	Computer numerical control
D-H	Denavit-Hartenberg parameters
DOF	Degrees of freedom
FANUC	Fuji Automatic Numeric Controls
LM	Levenberg Marquardt training algorithm
MLP	Mulit-layer perceptron
MSE	Mean squared error
NER	Newton-Euler recursive method
PUMA	Programmable universal machine for assembly
SCARA	Selective compliance articulated robot arm

Symbols for Mathematical Modelling of Manipulators

(n, b, t)	Orientation vectors for defining the orientation of link *n*
(X,Y,Z)	Cartesian coordinates
$\dot{P}_i^{i-1}$	Joint Rate Vector for prismatic joints
$\dot{\theta}_i^{i-1}$	Joint rate vector for revolute joints
A_i^0	Forward kinematics solution
A_i^{i-1}	Homogeneous transformation matrix of frame *i* with reference to frame *i-1*
C_{ji}	Matrix of cofactors of the Jacobian
P_i^{i-1}	Position matrix of object in frame *i* with reference to frame *i-1*
R_i^{i-1}	Rotation matrix of object in frame *i* with reference to frame *i-1*

T_i^{i-1}	Transpose matrix of frame i with respect to frame i-1
$^i v_i^0$	Linear velocity of link i with respect to base frame
$^i \omega_i^0$	Angular velocity of link i with respect to base frame
q_n	Joint space vector
a_i	Link length
d	Joint variable for prismatic joints
d_i	Link offset
F_i	Object frame
$J(q)$	Jacobian matrix
n	General variable
q	Common manipulator joint variable
R	Revolute joint
T	Prismatic joint
v	Cartesian space vector
V	Generalized velocity vector
α	Rotation angle about x-axis
α_i	Link twist angle
β	Rotation angle about y-axis
γ	Rotation angle about z-axis
θ	Joint variable for revolute joints
θ_i	Link angle

Symbols for Artificial Neural Network Model

$\bar{X}$	Mean value of a given variable X
d_m^s	Second derivative of error
d_m	First derivative of error
E	Network MSE value
E_A	Absolute error
E_p	Relative percentage error
m	Number of neurons in hidden layer

n	General variable
N	Total number of training patterns
R	Regression value
t_i	Target for the ith training pattern
w_{ij}	Weight associated with j^{th} input and i^{th} independent variable
X'	Normalized value of X
x_i	Input variable
Z_i	Network output
β	Slope parameter
λ	Damping factor

NOMENCLATURE

Degrees of Freedom   A direction allowing independent motion

D-H Parameters   Modelling convention for attaching reference frames to links in a kinematic chain

End-Effector   The last link of a manipulator

Error   Difference between the predicted and theoretical values

Jacobian   Matrix consisting of the first derivatives of the pose function

Joint   Location of contact for two or more parts of a robotic arm

Joint Limit   Constraining limit on the operating range of a joint

Joint Space   Joint angles or configurations of a manipulator

Kinematic Chain   Mathematical representation of rigid bodies connected by joints

Kinematics   Branch of mechanics dealing with motion without considering forces or mass

Manipulator   Industrial kinematic structure used for performing automated tasks in a manufacturing environment

Neural Network   Artificial intelligent machine learning and pattern recognition algorithm inspired from biological nervous system

Path Planning   Planning for maneuvering the end-effector of a manipulator through its workspace

Prismatic   A joint that allows linear motion between two consecutive links

Reconfigurable   A virtual tool capable of altering its computational capabilities for various manipulator structures

Redundant   A manipulator containing more than the required number of degrees of freedom

Revolute   A joint that allows single axis rotation between two links

Robot   A manipulator used in industrial applications

Singularity   A region defined by the position and orientation of a manipulator which causes loss of mobility

Workspace   Finite bound 3-D space defining the possible reach conditions of a manipulator

CHAPTER 1

INTRODUCTION

1.1 Background

Automation has led to industrial manipulators facilitating a wide array of operations such as assembly, inspection, material handling, processing etc. undertaken in the manufacturing industry today. A comprehensive set of robot structures have since been designed and built to fulfill the industry needs. These multi-Degrees of Freedom (DOF) structures are highly complex in their form and control. Most manipulators used in the industry today are articulated with six or more rotational joints. This structural form provides the manipulators with a great deal of flexibility, dexterity, and an ability to reach every specific coordinate of their workspace in more than one configuration. An acceptable level of functioning and control is therefore vital to the efficacy and successful implementation of industrial manipulators since the aforementioned tasks are highly repetitive and in many cases not apt for humans.

The initial steps in integrating manipulators, when planning for automation, includes their placement in an industrial setup based on the tasks they are required to perform. This is in direct correlation to the work envelope of each individual manipulator which dictates the working boundary of that manipulator. The total workspace of any multi-DOF manipulator is a finitely bounded 3-D space which is topologically complex and extremely challenging to visualize. In this total workspace, the true reachable workspace of a manipulator is a combination of various 3-D subset(s) that may or may not be mutually exclusive but are always collectively exhaustive. Each of these subset(s) is representative of range of joint configurations of such a manipulator. It is therefore important to assess and analyze the work envelope that defines the reachability and functionality of a manipulator. This assessment subsequently helps to identify and map user requirements to specific needs for automation.

A manipulator interacts with its environment (work envelope) through control of its joint space. The joint space of a manipulator entails all possible joint configurations of that manipulator. A 3-D work envelope is mapped in Cartesian space by the position and

orientation (pose) of a manipulator's tool (end-effector) for every configuration in its joint space. Industrial tasks and processes are seldom built to be accessible within a pre-positioned manipulator's work envelope. On the contrary, the positioning of a manipulator in a work cell is determined to ensure accessibility of tasks and processes it is intended to serve. An inverse mapping from the Cartesian space to a manipulator's joint space is thus required and is a challenging aspect of robot control.

The inverse mapping helps devise a control algorithm for a set of tasks to be accomplished by a manipulator. Numerous techniques such as use of teach pendants, robot simulation software, manual trial and error etc. are currently used in the industry for determining joint configurations that may produce a required tool pose for a task. All these techniques utilize conventional geometric, iterative or analytical methods to develop a solution to the problem of positioning a manipulator's end-effector. Often, the development of a closed form solution to this problem may be mathematically complex and computationally expensive, or may not even be possible. These limitations can be overcome by use of non-traditional approaches such as Artificial Neural Networks (ANNs). ANNs can identify and predict non-linear trends amongst data sets with and acceptable level of accuracy which makes them suitable for such an application.

In development of control algorithms, there often arise configurations where two or more joints of a manipulator no longer independently control the position and orientation of a manipulator's end-effector [1]. These configurations give rise to loci of subset(s) in a manipulator's work envelope known as kinematic singularities. Singularities are hard to visualize and plan around since they might exist in one or more configurations for any point in a manipulator's work envelope. For example, if a point (x,y,z) can be reached by a manipulator in ten different configurations, two of those ten joint configurations might be singular. Kinematic singularities arise because of the physical structure and attributes of a manipulator, and the relations between its joints. It is therefore important to design and build manipulators that can successfully avoid or minimize singularity configurations. This ensures robustness and accuracy of operations in manipulators [2].

Experimenting with variability in manipulator design is a challenging problem since most manipulators used in the industry today are flexible 5/6-axis articulated robotic arms (with rotational joints). These robotic arms are inspired from the human arms and their ability to rotate, position, and orient hands as shown in Figure 1 [3].
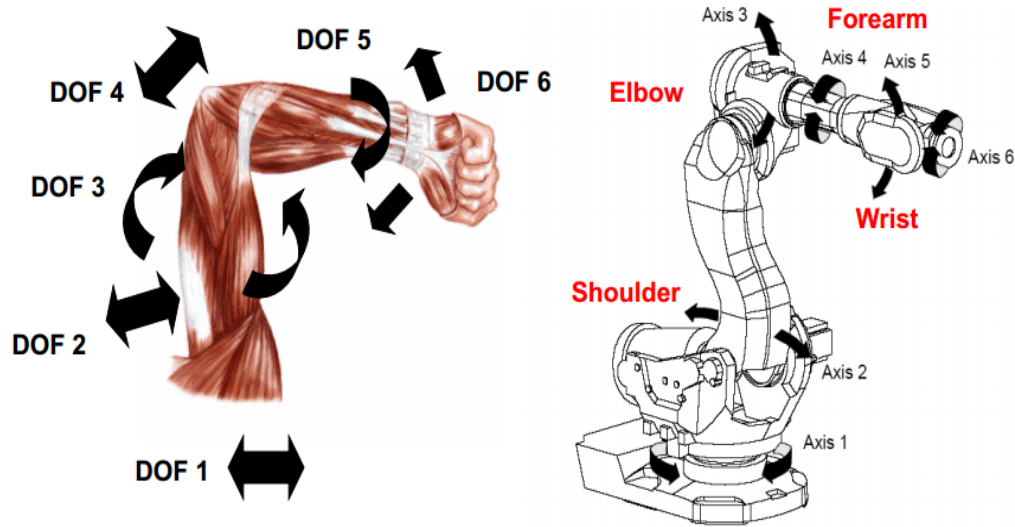
Figure 1: Articulated Robot Arm Inspired from a Human Arm [3]

Not many industrial manipulator designs exist that incorporate different joint types other than rotational joints. The use of articulated serial link robot arms in the industry today has evolved from gantry systems that could only be manipulated linearly along coordinate axes. The shift from traditional gantry (x-y-z) systems proved beneficial given the capability of flexible robot arms and their ability to handle complex tasks. However, through extensive research work and understanding into the functioning of flexible manipulators, the need for hybrid structures that incorporates a kinematic configuration of robot arms in conjunction with traditional Cartesian robots is realized.

Industrial manipulator manufacturers and developers provide specialized simulation software packages such as Workspace, RobotStudio, RobotSim, MotoSim etc., which can only analyze one or more specific classes of articulated manipulators. However, such software lack the capability of providing the user the freedom to reconfigure the functionality based on the structure of a manipulator. These software are primarily analytical tools rather than design tools, and simply simulate pre-programmed

work envelopes and trajectories. Existing software also only allow the users to change the range of joints for a robot configuration thereby adding to or limiting the reach of a manipulator's end-effector. The software also do not allow for any major change in topology and or volume of the work envelope. This inhibits the development of possible manipulator designs that may be specifically tailored and better suited to a customer need. The trend towards flexible manufacturing requires automation that can adapt to the same level of flexibility with decreasing cycle times and lead times, while increasing production capacity and quality [4]. It is therefore important to have manipulators that can adapt to a wide variety of tasks and processes with an acceptable level of functionality and control.

1.2 Research Purpose

Manipulators performance is critical to any industrial application. Manipulators however experience several challenges with respect to their performance that arise from their kinematic structure, reach limits within their workspace (work window), singularity conditions etc. A comprehensive analytical tool is therefore needed to optimize manipulator design and functionality without the need for extensive computation and planning. The purpose of this research is to develop a visual and analytical tool for the study of industrial manipulators. The methodology used for the development of this research tool is presented in Figure 2 below.
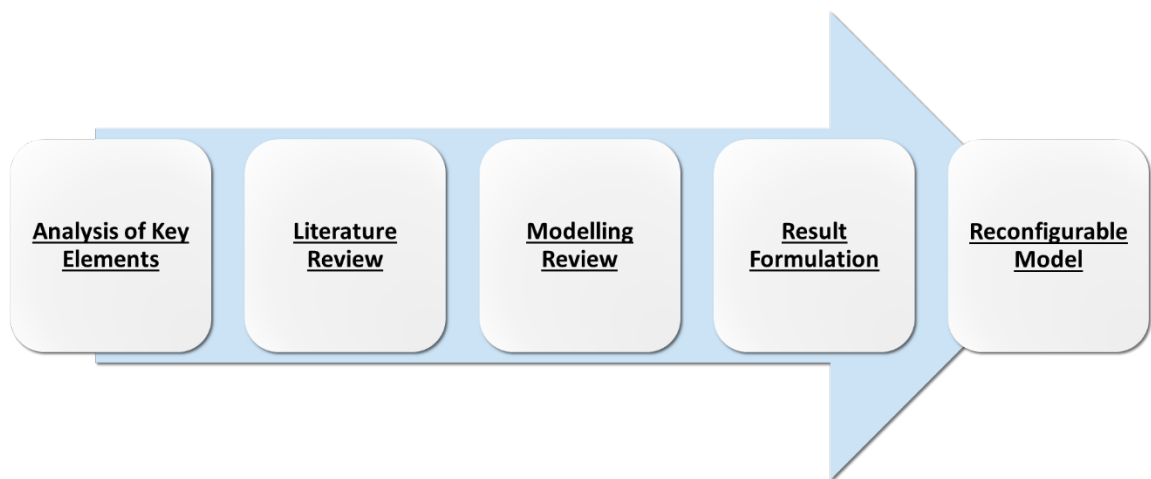
Figure 2: Research Methodology

This research provides an understanding of manipulator workspace where singularity conditions are identified and visually represented for an insight into the true work window. A robust inverse kinematic model is also developed using ANNs that provides a singularity free end-effector path through the workspace of the manipulator. The developed tool is capable of realizing the following tasks:

1. Reconfigurability: A virtual design tool capable of altering its computational capabilities for various 6-DOF (6 axis) manipulator configurations. The physical structure of a manipulator represented using its joints and link configurations can be postulated and or edited by the user. These configuration parameters can be based on any specific functionality requirements. The design tool will aid the analyses of any possible combinations of two manipulator joints types, namely, revolute (rotational) and prismatic (translational).

2. Workspace: A design tool that is capable of virtually generating, in 3-D space, and altering the topology and volume of a manipulator's work envelope based on the manipulator's kinematics structure. The reach parameters of a manipulator's tool (end-effector) could also be controlled using this tool by constraining the joint limits. This design tool will subsequently aid in development of tool path generation, path planning, travel path validation and optimization of reach conditions within robot work cell(s).

3. Inverse Kinematics: A design tool that is capable of computing a robust inverse kinematic solution for any input manipulator configuration provided by the user. The model will be able to present a solution that is computationally inexpensive unlike traditional geometric, iterative and analytical methods. The task will be achieved using non-conventional techniques (ANNs) that will predict an inverse kinematic solution within an acceptable confidence interval (90th-95th percentile).

4. Jacobian Matrix: A design tool that is capable of computing a Jacobian matrix for aid in analysis and control of manipulator motion. This Jacobian matrix will also aid in determination of kinematic singularities. The design tool can also be used as a basis for development of dynamic equations of motion, and transformation of forces and torques from the manipulator's end-effector to its joints [5].

5. <u>Kinematic Singularity</u>: A design tool that is capable of visually identifying loci of all singular points in a manipulator's workspace. The developed model can analyze and document every possible manipulator configuration for kinematic singularities. This will aid in development of a robust, continuous and singularity free control algorithm.

6. <u>Path Planning</u>: A design tool that is capable of providing a singularity free end-effector path through the workspace of a manipulator. The model can determine an error window in the joint space of the manipulator to provide a bounding space for inherent singularity.

1.3 Research Limitations

A mathematical model is developed in MATLAB platform for this research. All kinematic models i.e. the physical structure of manipulator joints and links have been visually represented in MATLAB through the use of robotic toolbox [6]. The model aids a user in development of the aforementioned research tasks. The following constraints define the limitations on the computational capability of the model:

1. Maximum Permissible Number of Joints (DOF): Six
2. Manipulator Type: Open Ended Kinematic Chains
3. Joint Types Permissible: Revolute (Rotational) and Prismatic (Translational)

CHAPTER 2

LITERATURE REVIEW

Significant research has been dedicated in the past towards the modelling of industrial manipulators. Majority of this research focuses on development and optimization of manipulator design and functionality in an industrial setting. A manipulator, because of its kinematic structure and joint configuration, poses inherent challenges such as kinematic singularities, complex inverse kinematic solution(s), trajectory planning, and travel path validation. Addressing such issues is therefore important for enhancing the robustness and accuracy of industrial manipulators. This chapter focuses on recent and notable developments in the field of industrial robotics which include manipulator modelling, traditional and non-conventional approaches to tackling the inverse kinematics problem, manipulator workspace generation, and singularity avoidance.

2.1 Manipulator Kinematics and Modelling Techniques

Yoshikawa [7] has proposed a measure of manipulability of robotic mechanisms for positioning and orienting end-effectors. Optimal postures and working positions have subsequently been defined for different manipulators from the viewpoint of its manipulability. The best postures and designs have been described as bearing resemblance to the human arms and fingers. The research paper provides an insight into the design and functionality of orthogonal, polar and cylindrical coordinate manipulators. It however does not focus on techniques to avoid non-optimal poses. Elkady, Mohammed, and Mohammed [8] have extended Yoshikawa's work to develop a new algorithm for measuring and optimizing the manipulability index of industrial manipulators. The technique is tested on PUMA 560 robot where a visual representation of the entire workspace is provided as a subset of varying manipulability. The research is significant for determining the most dexterous regions in a manipulator workspace. Pamanes and Zeghloul [9] have presented a technique for the optimal placement of robotic manipulators for a prescribed task using multiple kinematic criteria. An optimization problem is presented for this placement that takes into account several constraints such as upper and lower bounds, points in a path taken, number of joints etc.

The paper however does not address any collision avoidance techniques in the manipulator environment.

Work conducted by Djuric, Saidi, and ElMaraghy [10] demonstrates a multi-DOF kinematic structure consisting of both rotational and translational joints. A novelty methodology called n-GKM is presented by the author(s) which helps in developing an n-DOF global kinematic chain model. The research paper considers all possible kinematic structures in a 3 dimensional space, which is further divided into eight subspace and three planes. The paper provides the readers with a complete description of the D-H parameters and a visual representation of the multi DOF joints suitable for both robotic arm and multi axis CNC machines. The evaluation of this model is shown using all possible combinations of 2DOF kinematic structures i.e. RR, TT, RT, and TR. Computation of both forward and inverse kinematics for the n-GKM methodology has been demonstrated using the automatic separation method (ASM).

Laura and Khosla [11] have presented a Reconfigurable Modular Manipulator Systems (RMMS) method on automatically generating the kinematics of reconfigurable manipulators. The paper presents algorithm(s) for computation of forward and inverse kinematics of reconfiguring manipulators independent of the number, joint type, and shape of modules present. The model developed is applicable to redundant systems as well. The paper however does not focus on development of a reconfiguring structure using the proposed algorithm. Paredis and Khosla [12] have addressed the issue of determining the optimal manipulator configuration for any specific task using RMMS. The research addresses the kinematic design problem by developing an analytical solution for the inverse kinematics problem for a 2 DOF manipulator. Global optimization procedure is used to minimize the penalty of a manipulator design thereby resulting in an optimal kinematic configuration. The work presented however is only applicable to non-redundant manipulators.

Djuric and Urbanic [13] have also proposed a reconfigurable robot-based system for material deposition applications involving 2 ½ axis and 2 ½ axis + 2 axis tool paths. Various multi-action tool motions have been considered for development of four different robot based platforms. Reconfigurable parameters, K_1 and K_2, have been introduced in

modelling of the 2 DOF robot platform that help control the positive direction of each joint. The research paper provides an insight into the 2DOF manipulator response using the reconfigurable controller and factor(s) while suggesting investigation into higher DOF models. Several other research projects have also utilized the kinematic modelling methodology for multi-axis machine tools and its CNC applications. Xu et al. [14] have also presented a novel technique for modelling five axis machine tools using a methodology similar to the one used for modelling articulated robots using D-H parameters. This modelling technique is applied to CNC machines as the machine structure is treated as a single kinematic chain. A combination of two separate kinematic chains are used to model a single cutter chain which is considered as the end-effector for this structure. Since the machined surface depends on the path of the cutter (end-effector), trajectory planning is considered crucial for improving the process efficiency. Such modelling techniques allow for a unified structure that provides an in-depth exploration into the flexibility of five-axis machine tools. Work conducted by Du, Zhang, and Hong [15] provides a similar modelling technique for a three axis NC machine tool. The kinematic modelling is used to assess the geometric errors of CNC machine tools using a cross grid encoder. The error model encompasses the rotational and translational error component using an error transformation matrix of the machine tool. This method has been proven superior to traditional error component identification methods. The authors suggest using the novel technique for CMM's and other higher axis machine tools as well.

Lee and ElMaraghy [16] have emphasized the use of CAD based offline programming and analysis systems for robotic manipulators. ROBOSIM, a system developed for this research, determines the end-effector path, velocity calculations and singularity checks. Simulation of manipulator motion on computer workstations to tune any errors in the trajectory before real time implementation is proposed. Several advantages of offline programing have been put forth such as elimination of the need to have direct access to a robot, decreased production downtime, increase productivity, storage of data for posterity, and development of different task strategies. Disadvantages to offline programming include matching the simulation model to real time work environment, and the tedious task of creating a graphical CAD database.

9

2.2 Manipulator Workspace

Ceccarelli and Vinciguerra [17] have analyzed the workspace of a general open kinematic chain with four rotational joints by examining the effect of link parameters on its characteristics with the use of cross-sections. The authors emphasized the fact that three characteristics are important in evaluation of workspace, namely, the cross section, the volume and the existence of holes and voids. An algebraic formulation was developed for the robot's workspace from the envelop generation geometry. The workspace of the manipulator was theoretically calculated as the union of all toroidal surface workspaces by rotation of joint angles along each z-axis with respect to its base frame. The investigation found that the workspace of the manipulator was mostly affected by the ratio of the link lengths because of their ability to present voids and holes, and its twist angles. This technique is beneficial in analysis and synthesis of manipulators with rotational joints. Ottaviano, Husty and Ceccarelli [18] have presented a novel analysis on the workspace of industrial manipulators based on the level set reconstruction of their workspace. The method allows for determining the topologies of workspace of different manipulators based on their kinematic properties. Various numerical examples of orthogonal, ortho-parallel etc. manipulator types have been presented with singularities for surface S. The singularities of graph S are presented as singular configurations of the manipulator where it experiences more than normal singularity.

Liang and Ceccarelli [19] have also provided a parametric study and a classification procedure on all possible topologies of the feasible workspace of a general two revolute manipulator. The authors have selected four arbitrary boundary points on the torus workspace for generating design equations. However the method for selection of these arbitrary points for a feasible workspace is presented as an open ended problem. A classification approach was applied to compute all topologies of feasible workspace. Three different sub-regions for these topologies are then identified and analyzed to characterize workspace capabilities of 2R manipulators. Malek et al. [20] have presented an analytical technique for determining the boundary to a serial manipulator's workspace and any voids, if present, in that workspace. Voids in a workspace are identified by closed boundaries for which the acceleration form provides output normal to the outside of the enclosed surrounding space. A quadratic form has been devised for analyzing these

voids that are based on the acceleration analyses of the end-effector over singular surfaces. Such voids are identified as non-reachable spaces by a manipulator's end-effector. Voids and boundary conditions are identified in 3-D space for a 4R manipulator to demonstrate the robustness of the developed technique. The technique promises an effective method for analyzing workspace of serial manipulators.

A similar technique has been presented by Bohigas et al. [21] where a branch and prune technique isolates a set of singularities. These singularities are classified based on their correspondence to motion impediments in the manipulator workspace. The technique distinctly identifies all singularities and workspace topologies with any barriers present. The method is advantageous over other techniques because of its ability to converge higher dimensional boundary points without prior knowledge of the manipulator workspace. Goyal and Sethi [22] have determined the workspace of an RV-M1 Mitsubishi manipulator modelled using Denavit-Hartenberg parameters through use of MATLAB's robotics toolbox. The paper emphasizes that the workspace of a manipulator impacts its design, placement, and dexterity, and explores the method of finding singularity sets using the Jacobian rank deficiency conditions. These singularity sets when substituted in wrist accessible output set(s) of the robot, helped in determination of the workspace boundary. Examples of singularity sets at different configurations of the above mentioned manipulators are provided along with a visual representation in MATLAB.

Djuric et al. [23] have presented a technique to develop the functional and reachable workspace of serial 6 DOF manipulators for determining the effective travel path regions. The paper puts forth advantages of workspace visualization such as the ability to comprehensively assess manipulator configurations at design and redesign stages etc. A work window algorithm for the FANUC 6R family is provided along with singularity visualization at certain manipulator configuration(s). The research paper provides an evaluation of reduction in the work window of different manipulators at specific singularity joint configurations. Work done by Urbanic and Gudla [24] presents an estimation of the functional workspace of a manipulator using kinematic modelling and shape analyses. The outer boundary curves for an ABB IRB-140 manipulator are assessed

for functional workspace of a desired end-effector and tool orientation. Advantages of this technique include an understanding of the joint reach feasibility prior to on-site setups in a manufacturing environment. Djuric and Urbanic [25] have presented a similar technique for building reconfigurable alternatives and assessing the systems design through the use of functional workspace of manipulators. Since the work envelop does not allow for the operational feasibility of a manipulator, work window is introduced as a parameter that allows the kinematic structure to function under pre-defined conditions. The work window is graphically mapped at different tool orientations to compare the feasibility of operations for multiple kinematic chains in a manufacturing cell.

Alameldin et al. [26] have presented another technique for computation of 3D workspace of redundant manipulators. An algorithm is proposed as a hybrid between direct manipulator kinematics and screw theory. Screw theory is incorporated because of its ability to compute workspace points in pre-specified directions and no requirement for edge detection of boundary workspace unlike direct kinematics. The disadvantages of using screw theory presented are its exponential computation cost per point in the manipulator workspace, and the inability to identify holes and voids. Zein, Wenger, and Chablat [27] have presented an exhaustive study on the workspace topologies of orthogonal manipulators that have at least one D-H parameter as zero. Manipulators are classified in categories based on criteria such as size of feasible workspace subsets, existence and size of voids etc. 21 different categories are identified for 3R manipulators. The research is useful in analyzing the functional workspace of manipulators and identification of classes based on industrial needs. The research however is not practical for manipulators with higher DOF and for manipulators involving a combination of both translational and rotational joints.

Most workspace models presented in this section do not take into account the reconfigurability in design that may be introduced while analyzing the manipulator workspace. All workspace model(s) are based on pre-defined manipulator parameters and structural configurations. A need is therefore recognized for development of a tool that can generate and identify feasible workspace topologies for varying DOF open kinematic chains while accommodating combinations of different joint types.

2.3 Manipulator Singularity and Avoidance Techniques

Kim et al. [28] have presented a novel technique called the Task Reconstruction method that provides a solution to kinematic and algorithmic singularities. The method not only provides a singularity free trajectory but also guarantees task performance. The proposed method involves three tuning parameters in the reconstructed form of the desired task that allows for the formulation of a path through unknown singularities. Although, acceptable performance is achieved in cases involving only maximum of two subtasks. Another method of interest is presented by Liu and Zhang [29], where a damping reciprocal restrains or controls the joint velocities of a PUMA type of robot near singular points. The authors have demonstrated a technique for decomposing the inverse kinematics problem into subgroups with a trade-off in accuracy of velocity components in partial directions of the end-effector. According to this optimized method, the algorithm not only controls the sudden extreme changes in velocities near singular regions, it also helps to reduce the tracking of the end-effector. This method is highly beneficial in reducing the anomalies associated with manipulator singular positions. Zhunqing, Hairong, and Yuefa [30] have presented an algorithm for singularity control where line varieties and reciprocal screw theories are used to produce a full rank Jacobian matrix. The full rank allows singularity free motions when mapping from task space to joint space of a manipulator. Simulation results are provided for a PUMA robot demonstrating smooth velocity through singular regions. Similar analysis has been conducted by Fang and Lung-Wen [31] , and Hu et al. [32] where linearly dependent rows and columns of the manipulator Jacobian are isolated to allow feasible mapping between Cartesian and task space.

Pai and Leu [33] have presented a technique for symbolic computation and study of singularities for decoupled manipulators. An algebraic condition for genericity for three joint robots is presented using Jacobian determinants. The proposed method helps in mapping singularities as smooth manifolds in the joint space of the manipulator. A characterization of orientation singularities is provided in this paper for any arbitrary number of joints. It is observed that the robot is only generic if no adjacent joints of the manipulator are parallel. Djuric et al. [34] have provided a visual representation of the singularity zones through manipulation of fundamental kinematic equations. The

proposed technique helps in understanding singularity conditions for robot work cells and aids in travel path generation and manipulator layouts. Decoupling of Jacobian based on wrist and forearm joints is used to generate a loci of singular points for the FANUC family of manipulators. Also, the effect of link lengths on the topology of singular space is presented. This method is highly beneficial in analyzing the mechanical structure of a manipulator as means of singularity reduction. Huo and Baron [35] have developed a redundancy-resolution (RR) algorithm for optimizing the joint space trajectory of 6R arc welding manipulators. The authors have proposed a decomposition in the required instantaneous twist of a welding electrode in two orthogonal components. The symmetry axis of the electrode allows the two components to lie in either task space or redundant space. This technique efficiently optimizes the joint space trajectory and can be extended to tasks that require less than 6 6 DOF in their tool frame.

Stanisic and Duta [36] have provide a novel design of symmetrically actuated double pointing systems (SADPS) for eliminating singularities from manipulator wrists. The design includes two serially connected spherical pointing systems with a common center. The constraint functions of the developed system reduces the independent DOF to two thereby resulting in a symmetry of motion for the corresponding links in each pointing system of the double system structure. Superior dexterity of the SADOS system is also observed with a two or three DOF singularity and interference free manipulator wrist. Cheng et al. [37] have provided a technique (SICQP) to minimize the tracking errors in the singularity direction for a PUMA 560 robot. The method decomposes the workspace of the manipulator in singular and non-singular directions to provide extra redundancy to achievable directions. This method is effective and efficient in solving the inverse kinematic problem but requires decoupling of three-dimensional sub-problems.

Unlike traditional methods that depend on analysis of the Jacobian for computation of kinematic singularities, Ahmad and Luo [38] have considered inverse kinematic relationships to form triangular equations that reveal the structural properties of the manipulator and the Cartesian configurations of the end-effector where the manipulator is singular. This technique allows for computation of singularity states in terms of Cartesian parameters of the end-effector even when the joint offset angles are not zero or ninety

degrees. The method helps in trajectory verification of non-singular regions without the need for computing an inverse kinematics solution. It also helps in coordination of redundant robots. Analysis of less than twelve DOF redundant arms is also possible using this technique by splitting an arm into two sets of six DOF and/or less that six DOF manipulators. A higher accuracy of motion is observed with use of this method and the results are useful in trajectory verification and redundancy coordination in Cartesian space. Chiaverini and Egeland [39] have also presented a technique to handle the problem associated with singularities in six-joint manipulators. This techniques allows for successful removal of undesired commanded motions and presents an exact inverse kinematic solution for the remainder part which can be used for both off-line planning and real-time control. The authors have emphasized the problem in development of an algorithm apart from the traditional use of inverse of the Jacobian that supports both robustness and high accuracy of the manipulator. The method first determines degenerate directions corresponding to the singularities, after which a marginal window is defined around that singular region where the manipulator is treated as being singular. An inverse kinematic solution is then found for the remainder space that has minimum error and norm in end-effector coordinates and joint space respectively. Interpolation technique is finally used in the previously determined degenerate directions for a continual solution to the manipulator motion. This method demonstrated promising results for a 3R industrial manipulator with a trajectory through the wrist singularity and can be successfully used for similar manipulator configurations.

Work done by Yigit, Burghartm & Woern [40] demonstrates the development of alternate configurations to avoid singularities of a human like robotic arm. Yigit et al. solved the inverse kinematic problem by using a closed form solution and attempted to develop configurations that would avoid singularities. However, this approach resulted in loss of the reachable workspace of the robotic arm. The kinematic singularity was avoided by use of a combination of restriction and elongation of the arm segments to compensate for the loss in workspace.

Majority of the work provided in singularity analysis and avoidance techniques involves either manipulation of the Jacobian, restriction of joint motion or development

of new geometric method(s) to ensure smooth end-effector velocity through singular regions. A major drawback to these techniques is the complexity in modelling and much need priori knowledge of theoretical concepts. A need is therefore recognized for a simplified algorithm that can provide equally promising results but in fraction of the computation time. Also, the discussed techniques require some kind of manipulation with the physical geometry and/or joint configuration of the robots being studied. A solution to introducing such variation to a manipulator design is, however, not presented with any of the theoretical techniques.

2.4 *Inverse Kinematics using Artificial Neural Networks*

Prior research has proven ANNs as an important tool in robot path planning and control by successfully providing a solution to the inverse kinematics problem. The network accuracy using ANNs, however, has been a common problem encountered by various researchers in determining a solution. Kozakiewicz, Ogiso, and Miyake [41] have proposed a partitioned neural network architecture to improve the accuracy for an inverse kinematic problem. The partitioned layer, also referred to as the pre-processing layer, helped to divide the entire network into individual smaller networks where the weights of each partitioned network could be attenuated by concentrating on only one output. The network achieved high prediction accuracy for position joints but exhibited higher errors for orientation joints. Further work was suggested to obtain accurate learning and prediction results for the entire range of joints, especially the orientation joints. Lou and Brunn [42] have introduced an iterative approach for computing the inverse kinematic problem using ANNs with an offset error compensation method to improve the accuracy of the derived solution. The methodology was implemented since an offset error always existed when taking the iterative approach which had different values for each required end-effector position. The error compensation improved the accuracy of the network by reducing the average error from 4 to 0.001 percent for a 2 DOF manipulator. The work was extended in a two stage process to 6 DOF manipulators because of computing limitations. Ahmad and Guez [43] also used an iterative approach using ANNs to find the final predicted solution within a specified tolerance. The iterative process provided a two-fold increase in the computational efficiency of a 3 DOF planar robot and the PUMA 560 robot.

Yildirim and Eski [44] have presented a feed-forward neural network architecture with five different learning techniques namely, Online Back Propagation (OBP), Online Back Propagation Random (OBPR), Batch Back Propagation (BBP), Delta Bar Delta (DBD), and Quick Propagation (QP). These learning techniques were used to predict pre-defined target kinematic parameters of a PUMA 560 robot. It was determined from this study that QP was the best learning technique to update network weights. Here, the output(s) of the network exactly matched the target values with a root mean square (RMS) error of 0.21345. The drawback to this technique was the fact that robot(s) without wrist offsets lack rotational capabilities and did not have a closed form inverse kinematic solution. Therefore, this technique could only be implemented as a single-stage network.

Koker et al. [45] have also validated neural network as a tool for computing the inverse kinematics of a three joint robots. The developed network was able to predict the joint angles to its corresponding Cartesian (X,Y,Z) co-ordinates within an acceptable error range. Hasan et al. [46] have addressed the problem of kinematic control through singularity zone(s) by development of an ANN model that learns the characteristic of the robot system rather than specifying an explicit system model. The discussed model has Cartesian co-ordinates (X,Y,Z) of the end-effector, orientation angles (R,P,Y), and linear velocity of a 6 DOF robot as network inputs, and angular position and velocity as the network outputs. The maximum error percentages for the experimental data set introduced to this network were determined to be 6.72% for the Z-coordinate and 5.79% for the Y-orientation. This network model can be implemented for any serial manipulator with a reasonable accuracy. However, the paper did not explore different network topologies to further investigate the error reduction in the network.

Bingul, Ertunc, and Oysu [47] have explored three different end-effector orientation types, namely, homogeneous transformation matrix, Euler angles, and equivalent angle axis for training the ANN. The method is validated on a 6R manipulator with wrist offset. The results are satisfactory with errors as high as 10 degrees of data resolution. Feng, Yao-nan, and Yi-min [48] have presented a new algorithm called extreme learning machine (ELM) that randomly chooses input weights and analytically determines the

output weights in a single hidden layer feed-forward ANN. The proposed method provides good generalization performance, fast learning, and improved precision in development of an inverse kinematic solution.

ANNs provide a quicker response, and have proven to be useful for multiple satisfactory solution(s) to the inverse kinematics problem with real-time adaptive control [45] [46]. An inherent challenge with this technique has been the attempts in increasing the accuracy of the developed network. In the past, kinematic data from manipulators has demonstrated high variation and lower fitting rates when processed through ANNs. Moreover, every ANN architecture is tailored towards a specific configuration or class of robots. For example, a specific ANN model might only be able to provide an acceptable level of accuracy for non-wrist partitioned manipulators. An approach thus needs to be developed to tailor the kinematic data of a manipulator along with the ANN architecture for a universally acceptable model. Also, limited research exists that utilize ANNs as a technique for coping with kinematic singularities by either providing a robust inverse kinematic solution or by developing a path planning model for avoiding singularity zones.

CHAPTER 3

INDUSTRIAL ROBOTICS

Any electro-mechanical device operating under computer control with some degree of autonomy can generally be referred to as a robot. An industrial robot, however, as defined by International Organization for Standardization (ISO 8373) is "An automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications" [49]. Industrial robots used in the industry today have evolved from a union of teleoperators and Computer Numerical Control (CNC) machines [5]. They serve their purpose by substituting as labour for tasks that are impractical, undesirable, and repetitive for humans. The need for these industrial robots came into being from capital-intensive, large volume, and high precision manufacturing required in the automotive, and electrical goods industries [50]. According to 2012 statistics by the International Federation of Robotics (IRF), the worldwide market value for industrial robot systems is approximately $26 billion with a high number of robot density (industrial robots per 10,000 persons employed) in countries such as Korea (396), Japan (332), Canada (103) etc. [51].

Robots in the industry today have evolved since then to handle more complex tasks and adapt to different applications such as assembly, welding, machining, etc. that require high endurance, speed, and precision. The uses of industrial robots based on the type of industry and their applications are presented in Figure 3 and Figure 4. Handling of materials and process along with welding and soldering operations constitute the majority of applications of robots in the industry today. The physical structure and attributes of these industrial robots greatly vary on the nature of tasks they are required to perform. Industrial robot performance has significantly increased over the past few decades. Robots can now be controlled with an acceptable level of safety standards and performance which allows for human-robot collaboration in the same workplace [50]. This symbiosis has expanded the scope of industrial robots to other application areas and industries. Industrial robots are thus being required to have some level of flexibility and reconfigurability for such integration.

Use by Industry

Legend:
- Automotive — 33.2%
- Electrical and Electronics — 9.9%
- Chemical Rubber and Plastics — 9.4%
- Machinery — 4.3%
- Metal Products — 3.7%
- Communications — 2.5%
- Food — 1.5%
- Precision and Optics — 0.8%
- Other — 9.7%
- Unspecified — 25.0%
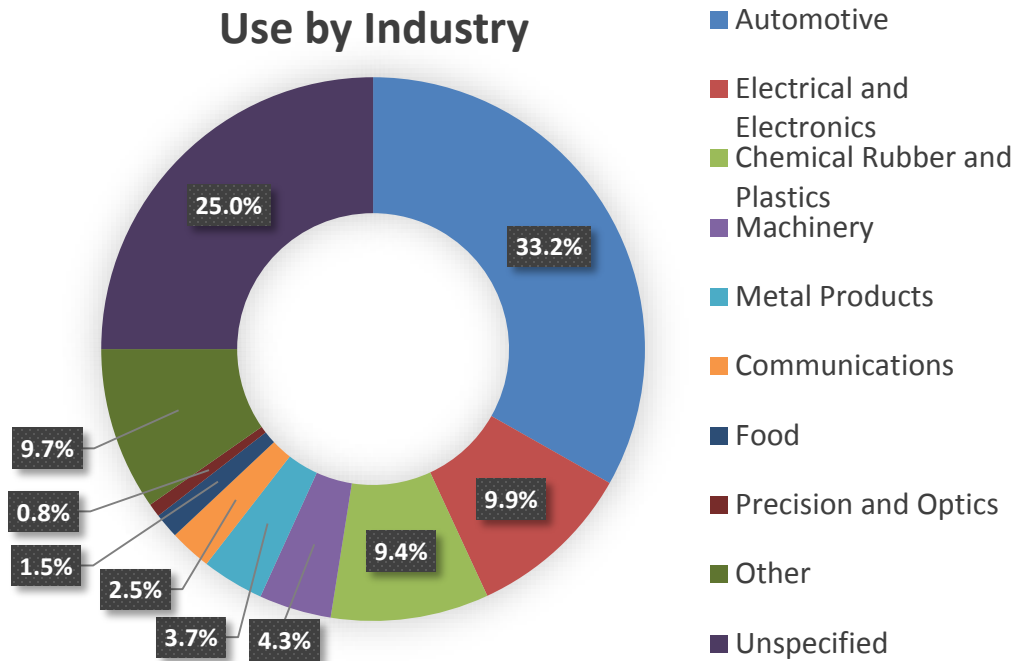
Figure 3: Use of Industrial Robots by Industry [52]

Use by Application

Legend:
- Handling of Materials and Processes — 35.4%
- Welding and Soldering — 28.9%
- Assembling and Disassembling — 13.0%
- Dispensing and Painting — 3.7%
- Cutting Milling and other Processing — 2.5%
- Other — 7.9%
- Unspecified — 8.6%
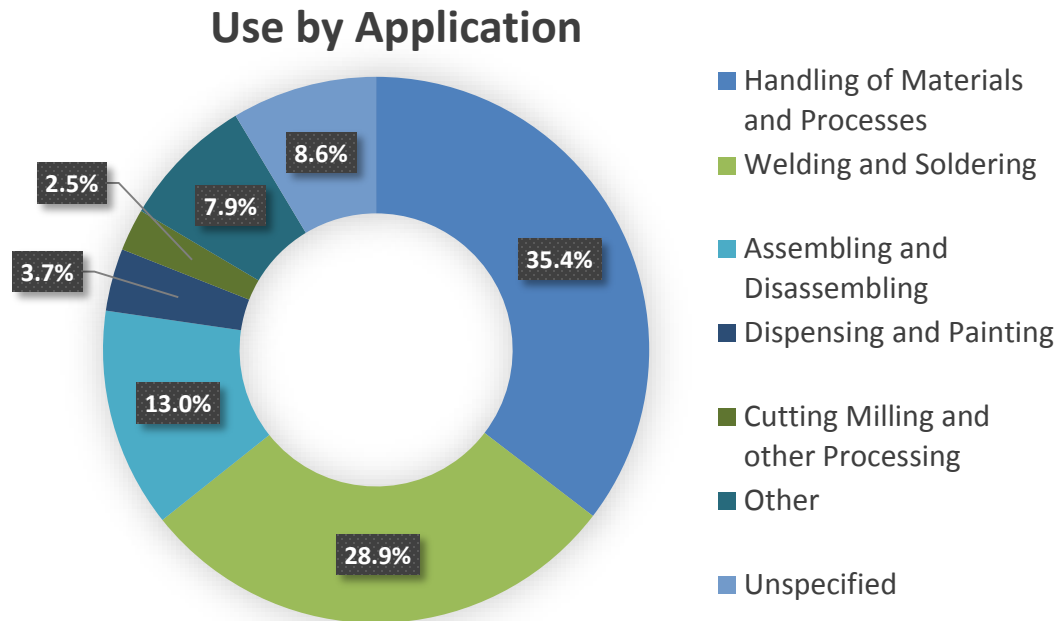
Figure 4: Use of Industrial Robots by Application [52]

3.1 Hardware and Software

Most industrial robots (manipulators) include some basic hardware and software components as seen in Figure 5. These components constitute the electro-mechanical framework, and the computer control or 'Artificial Intelligence' of the robot.

The Hardware components for a common industrial robotic system can be divided into the following five categories:

1. Robotic Arm: The robot arm constitutes the mechanical part of the robot and consists of joints, links, motors (actuators), sensor, shafts, gears, end-effector(s) etc.
2. Teach Pendant: The teach pendant is a remote device used to operate the robot manually. It serves as a user input device to feed commands to the robot.
3. Robot Controller: The robot controller constitutes all control circuits consisting of microprocessors, motors, sensor, electronics, interface connectors and power units for the robot arm to function.
4. Interface Computer: The interface computer is the program storage unit of the manipulator. It serves as a user interface between the operator and the controller.
5. System Software: The systems software constitutes the programmed data stored on the robot's memory chips. The different codes and functions here help convert sensor information into actuator commands thus providing the robot with 'artificial intelligence'. [53]
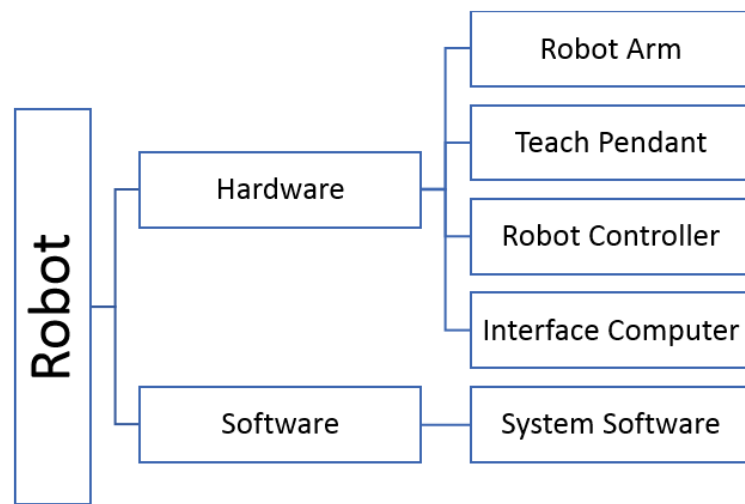
Figure 5: Industrial Robot Components

3.2 Symbolic Representation of Joints and Links

A robot manipulator's physical setup consists of sequence of links connected by different joints that form a kinematic chain. Combination of various joint types such as revolute, prismatic, twisting, ball and socket etc. are often used to interconnect links in industrial manipulators. This research addresses two commonly used joints, namely:

1. Revolute (Rotational): A revolute joint provides relative rotation about a single axis between two links. A revolute or rotational joint can be represented by the symbol 'R', with a joint variable 'θ'. The joint variable for a revolute joint determines the angular range or motion for that joint. Figure 6 demonstrates a kinematic chain with three rotational joints.

2. Prismatic (Translational): A prismatic joint provides relative translation along a single axis between two links. A prismatic or translational joint can be represented by the symbol 'T', with a joint variable 'd'. The joint variable for a prismatic joint determines the linear range of motion for that joint. Figure 7 demonstrates a kinematic chain with three translational joints.
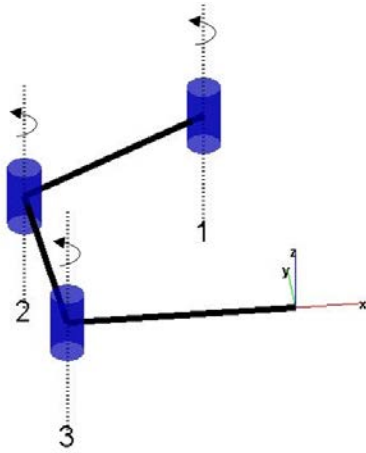
Figure 6: Rotational Joint(s)   Figure 7: Translational Joint(s)
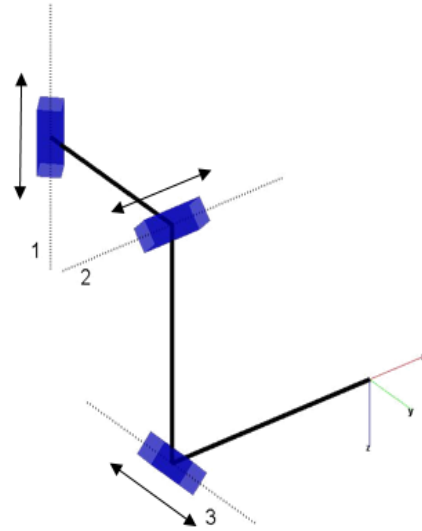
In building the reconfigurable model for this research, both rotational joint(s) (R) and translational joint(s) (T) are represented using a common joint variable, 'q'. A common joint variable (Equation 1) helps the model to adapt to the reconfiguring structure of a

manipulator without the need for changing subsequent parameters and equations. It also aids in the manipulation of the Jacobian matrix and development of manipulator workspace and singularity space. The use of this variable will be demonstrated subsequent chapters.

$$q_i = \left\{ \begin{array}{l} \theta_i \quad for\ all\ rotational\ joints \\ d_i \quad for\ all\ translational\ joints \end{array} \right. \tag{1}$$

3.3 Manipulator Classification

For an understanding of the manipulator workspace and kinematic singularities, it is important to first recognize the basic manipulator types used in the industry today. Nearly all industrial manipulators in use have six or less DOF ($\leq$ six independent joints). Of these joints, the first three joints form the arm of the robot and the latter the wrist. This is because a minimum of three joints are required to position (in X,Y,Z) the end-effector of a manipulator. Industrial manipulators are broadly classified in five different categories based on their forearm's mechanical structure, namely;

1. Linear (Cartesian and Gantry) (TTT): Linear manipulators are the most basic type of manipulators with three translational joints. Each joint allows a translation in one of the X, Y, or Z axis to position the end-effector. Linear manipulators are majorly used for pick and place, and handling applications.

2. Articulated (RRR): Articulated manipulators are the most common type of manipulators used in the industry today since they provide the greatest relative flexibility, and increased dexterity in a compact space. These robots have three rotational joints and are majorly used for operations such as welding, painting, assembly etc.

3. Spherical or Polar (RRT): Spherical or Polar manipulators derive their name from the fact that their axes form the spherical or polar coordinate system. These robots have two initial rotational joints and a third translational joint. Major applications of these robots are in the welding and casting industry.

4. SCARA (RRT): Selective Compliance Articulated Robot Arm (SCARA) manipulators are robots with two parallel rotational joints and a third translational

joint. This allows a robot to provide compliance in a plane. These robots are majorly used for pick and place work.

5. Cylindrical (RTT): Cylindrical manipulators derive their name from the fact that their axes form the cylindrical coordinate system. These robots have an initial rotational joint and two subsequent translational joints. Major applications of these robots are in the assembly, welding and casting industry.

A basic kinematic structure of the aforementioned manipulators is provided in Figure 8 below:
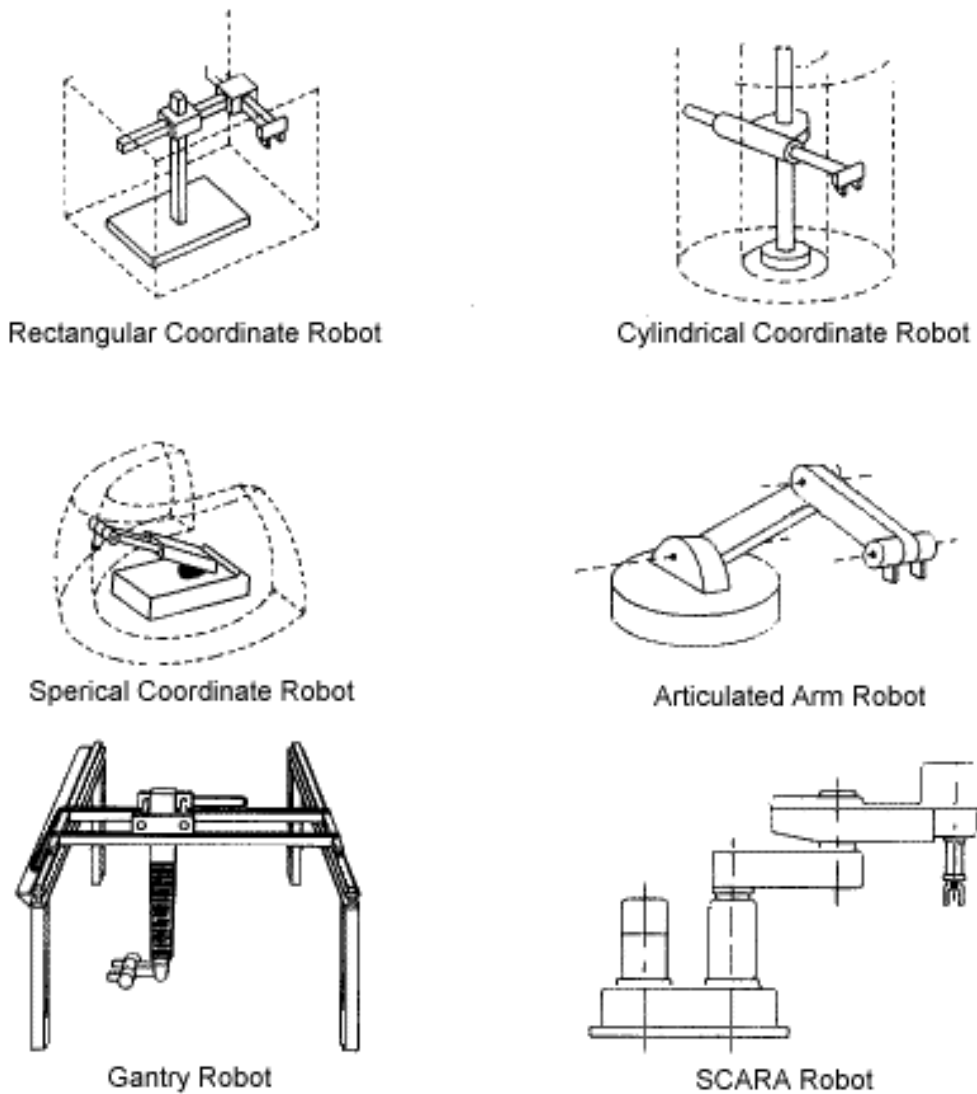
Figure 8: Kinematic Structures of Basic Manipulator Types [54]

3.4 Manipulator End-Effector Types and Application

As defined by the United States Occupational Safety and Health Administration (OSHA), a manipulator's end-effector is "An accessory device or tool specifically designed for attachment to the robot wrist or tool mounting plate to enable the robot to perform its intended task. (Examples may include gripper, spot-weld gun, arc-weld gun, spray- paint gun, or any other application tools.) [54]."

The forearm (first 3 joints) of the robot is responsible for positioning the end-effector while the wrist of the robot is responsible for orienting the end-effector. Not all industrial robots however, have an arm and wrist configuration. Many manipulator designs exist or can be generated with no wrist configuration as seen in Case study 10.2 in Chapter 10. The DOF for orienting an end-effector are determined by the DOF of the wrist [5]. A wrist configuration may have up to 3 DOF, namely:

1. Yaw: A counter-clockwise rotation about the z-axis.
2. Pitch: A counter-clockwise rotation about the y-axis.
3. Roll: A counter-clockwise rotation about the x-axis [55].

Figure 9 demonstrates a commonly used spherical wrist configuration. The spherical wrist effectively aids in decoupling the position and orientation of an end-effector [5].
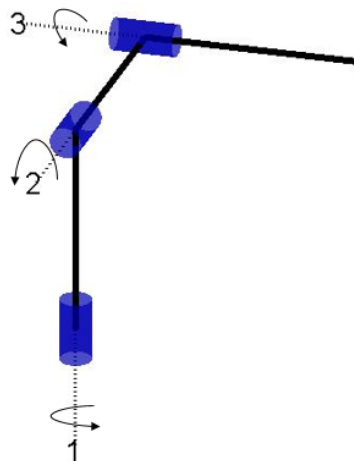
Figure 9: Spherical Wrist Configuration   Figure 10: Different Gripper End-Effectors [56]

The end-effector is the most critical part of the robot that performs the robot's intended function. A considerable amount of engineering work is therefore dedicated to the design and build of end-effectors. The mechanical structure of the end-effector depends on the type of application it is used for. End-effectors vary from simple open and close grippers used in material handling to complex tools for machining and performing tasks. Figure 10 above demonstrates three different types of gripper type end-effectors.

CHAPTER 4

MATHEMATICAL CONCEPTS

Understanding of some key mathematical concepts such as Degrees of Freedom (DOF), representation of position and orientation in Cartesian space, frame transformations, etc. is important before modelling of open-ended kinematic chains. The following sections in this chapter cover some of these important concepts.

4.1 Degrees of Freedom (DOF)

The number of Degrees of Freedom for any industrial manipulator is the number of axes of movement for that manipulator. This movement can be either a rotation about an axis if the joint is rotational (R), or it can be a translation along an axis if the joint is translational (T). It is however important to realize that the number of joints may not always equal to the Degrees of Freedom for a manipulator. For example, two rotational joints in a manipulator might rotate about a single axis. This cancels out one additional Degree of Freedom which would have been possible had both the joints not been rotating about the same axis.

The number of Degrees of Freedom required by a manipulator is determined by task required of the manipulator. As such, six Degrees of Freedom are required to locate any object in 3-D space. Three of these DOF represent the position of the object while the rest determine the orientation of the object in space. Therefore, depending on the positioning and orientation of a part, appropriate number of DOF are built into the manipulator for easier control. Manipulators with more than six Degrees of Freedom are referred to as redundant manipulators. These manipulators have additional DOF for increased mobility and flexibility [57]. An example of a redundant robot is the Canadarm. Figure 11 demonstrates an object defined using six degrees of freedom in 3-D space.
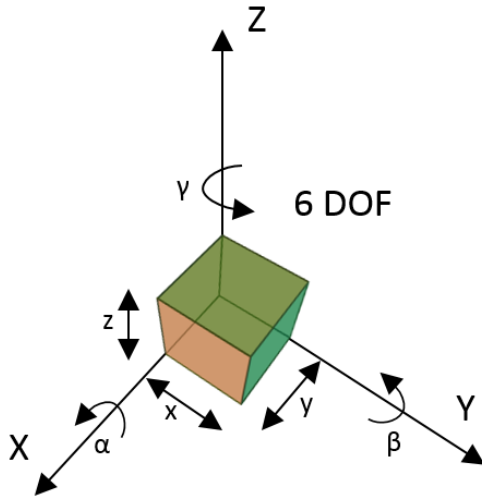
Figure 11: DOF of an Object in 3-D Space

4.2 Representation of Position and Orientation

Kinematic modelling of manipulators requires all links to be considered as rigid bodies. Coordinated frames are then rigidly (fixed location) attached as reference to these rigid bodies. These coordinate frames help in determining the position and orientation of any one frame with respect to another frame by means of frame transformations in 3-D space.
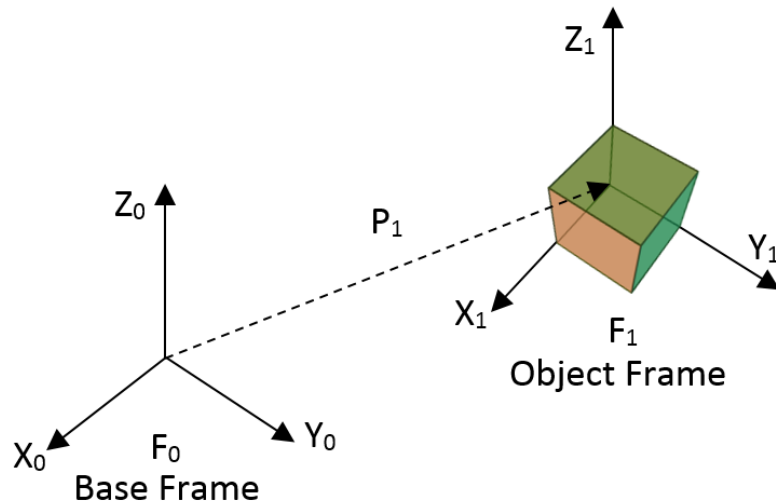
Figure 12: Object Frame with respect to Base Frame

For example, in Figure 12, the position (P matrix) of any object (Object Frame F_1) in space with respect to another object (Base Frame F_0) is defined using the X, Y, and Z

Cartesian coordinates as presented in Equation 2. Similarly, orientation (rotation matrix, R) of any Object Frame F_1 with respect to Base Frame F_0 in 3-D space is defined using three rotational angles (α, β, γ) around each reference axis (Figure 11). Here, α is the rotation about x-axis, β is the rotation about y-axis, and γ is the rotation about z-axis. These rotational angles collectively represent nine rotational elements as presented in Equation 3 [57].

$$P_1^0 = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{2}$$

$$R_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3}$$

The position and orientation, collectively called the 'pose', can thus be defined using 9 rotational elements and 3 position elements. These elements will subsequently be used as inputs for ANNs in determining an Inverse Kinematics solution.

4.3 Frame Transformation

In kinematic modelling, it is important to have an understanding of the position and orientation of the manipulator's end-effector with respect to the base of the manipulator. This kind of modelling requires the computation of position and orientation of a point in 3-D space from a previously known position and orientation of that point. For example, consider a point 'W' in Figure 13. The coordinate vector representing point W with respect to F_1 is given by Equation 4 as:

$$q^1 = \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} \tag{4}$$

It is then required to determine the coordinate vector that represents the point W with respect to F_0 given by Equation 5.

$$q^0 = \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix} \qquad (5)$$

From Figure 13, and Equation 5 the resultant vector v is determined in Equation 6.

$$v = p + u \qquad (6)$$

Substituting the vectors by their position are orientation, the position and orientation of v is obtained in Equation 7

$$v = q^0 = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = R_1^0 q^1 + P_1^0 \qquad (7)$$
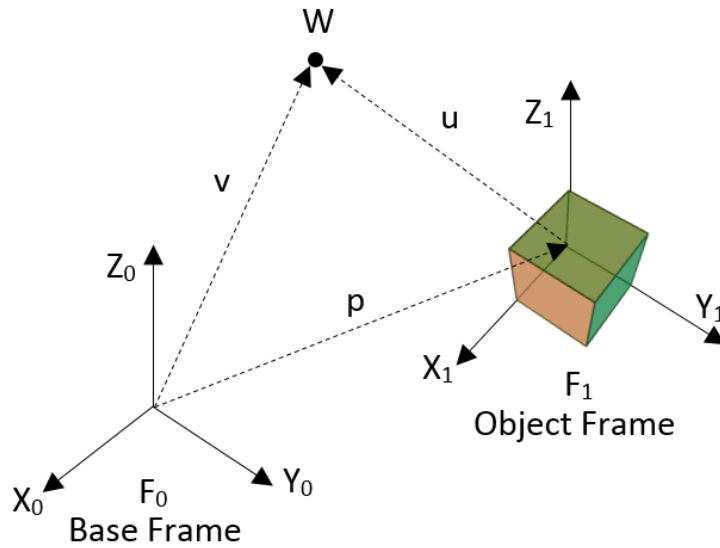
Figure 13: Frame Transformation

It can therefore be conclude that the position and orientation of a point W with respect to F_0, can be defined by a simple frame transformation as represented in Equation 8.

$$q^0 = T_1^0 \, q^1 \qquad (8)$$

where the transpose matrix T_1^0, transforms coordinate vectors from frame F_1 to F_0 [57].

4.4 Roll, Pitch and Yaw (RPY) Angles

Another way of representing the rotation matrix R, is through the Roll, Pitch and Yaw (RPY) angles represented by R(γ, β, α). These angles define the rotation of an object (Figure 11) through successive canonical rotations about the coordinate axes. Here,

1. Roll: Roll is counter-clockwise rotation of α about the x-axis.
2. Pitch: Pitch is counter-clockwise rotation of β about the y-axis.
3. Yaw: Yaw is a counter-clockwise rotation of γ about the z-axis.

It is important to note that these rotations are performed in the order of roll given by $R_x(\alpha)$, then pitch given by $R_y(\beta)$, and finally yaw given by $R_z(\gamma)$. The final rotation matrix however, is obtained by multiplying the angles in the order of yaw, pitch, and roll. This is because of the backward sequence of multiplication in frame transforms. The individual rotations and the final rotation matrix are provided in Equations 9, 10, 11, and 12.

$$R_z(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \tag{10}$$

$$R_x(\alpha) = \begin{bmatrix} 1 & 1 & 0 \\ 1 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{11}$$

$$R = R(\gamma, \beta, \alpha) = R_z(\gamma) . R_y(\beta) . R_x(\alpha) \tag{12}$$

The elements of this rotation matrix, R can then be manipulated to calculate the roll pitch and yaw angles. Table 1 below provides a solution to computing RPY angles from the rotation matrix, R:

Table 1: Computing RPY Angles from Rotation Matrix

	For R(3,1) ≠ ±1	For R(3,1) = -1	If R(3,1) = 1
α	$\text{atan2}\left(\dfrac{\frac{R(3,2)}{\cos(\beta)}}{\frac{R(3,3)}{\cos(\beta)}}\right)$	$\text{atan2}\left(\dfrac{R(1,2)}{R(2,2)}\right) + \gamma$	$\text{atan2}\left(\dfrac{-R(1,2)}{R(2,2)}\right) + \gamma$
β	$\text{atan2}\left(\dfrac{-R(3,1)}{\pm\sqrt{1-(R(3,1))^2}}\right)$	pi/2	-pi/2
γ	$\text{atan2}\left(\dfrac{\frac{R(2,1)}{\cos(\beta)}}{\frac{R(1,1)}{\cos(\beta)}}\right)$	Arbitrary	Arbitrary

For the purpose of this research, the orientation of the end-effector was defined using both RPY angles and through 9 individual rotational elements of the rotation matrix, R. However, through the development of the reconfigurable model, it was realized that superior results were achieved for ANNs when using elements of the rotation matrix, R (Equation 3), in computation of an inverse kinematics solution. The RPY angles provide a consolidated overview of an objects orientation with respect to a coordinate frame and are easier to document. For this reason, the orientation of an end-effector is usually represented using its RPY angles.

CHAPTER 5

KINEMATIC MODELLING OF MANIPULATORS

This research addresses the kinematic modelling of open ended kinematic chains that are widely used in the industry today. As previously mentioned, the kinematic modelling of manipulators requires frame transformation of coordinate frames attached to each link of the manipulator. These frame transformations help us to determine the forward kinematic solution for a manipulator. A forward kinematics solution helps determine the final position and orientation of the manipulator end-effector with its base for any possible combination of the manipulator's joint variable(s) (q). The forward kinematics solution can then be manipulated geometrically, analytically, or iteratively to derive an inverse kinematic solution. An inverse kinematic solution helps determine the values of all joint variable that would produce a required position and orientation of the manipulator's end-effector.

Any manipulator with n joints, has exactly $n+1$ links, since each joint connects two links of a manipulator. Therefore, any joint i, when actuated moves the link i, where the location of joint i is determined by link $i-1$ [5]. All joint variables, as previously mentioned are represented by 'q'. Thus any joint q_i can assume the value of θ_i if the joint is rotational, or d_i if the joint is translational.

As standard convention, a Cartesian coordinate frame F_0 is rigidly attached to the base (i.e. link $i-1$) of the manipulator. All subsequent frame transformations for the manipulator are performed by referencing this frame F_0 to other coordinate frames. Cartesian coordinate frames are attached to each link of a robot, starting with the base frame all the way to the end-effector. The position and orientation of each frame can be expressed through the homogeneous transformation matrices. It is important to note that all frames are rigidly attached to each link. This assumption is made so that the position and orientation of a manipulator's end-effector can be determined with respect to any particular frame of interest, and is always constant irrespective of the configuration of the manipulator. [5] For example, a SCARA (RRT) robot (Figure 14) is kinematically modelled in Figure 15.

Figure 14: SCARA Robot [58]
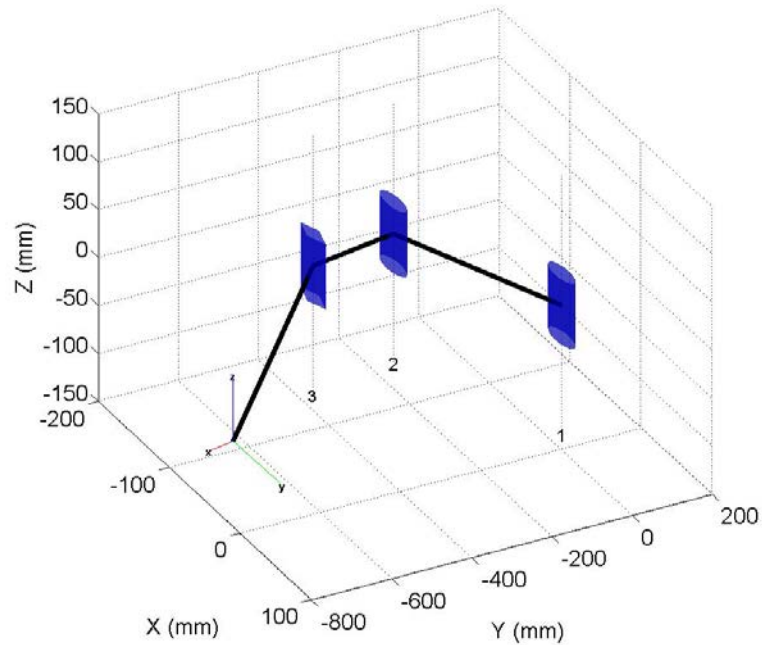
Figure 15: Kinematic Modelling of SCARA Robot

5.1 Denavit-Hartenberg (D-H) Parameters

Denavit-Hartenberg (D-H) parameters are set of standardized rules that are used in defining Cartesian coordinate frames attached to the manipulator links. These parameters help define position and orientation of one frame with respect to its preceding frame.
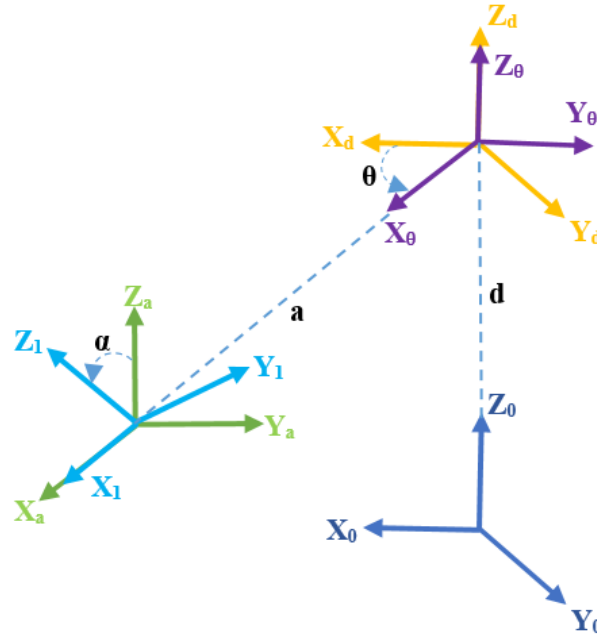
Figure 16: D-H Parameters. Adapted from [53]

The D-H parameters for defining the pose of any coordinate frame i (F_1) with respect to its preceding frame i-1 (F_0) are comprised of the following four parameters (see Figure 16):

1. Link Offset (d_i): It is the distance measured along Z_0 axis to the point of intersection of X_1 axis and Z_0 axis.
2. Link Angle (θ_i): It is the angle between X_0 axis and X_1 axis measured in a plane normal to Z_0.
3. Link Length (a_i): It is the distance between Z_0 axis and Z_1 axis measured along X_1 axis.
4. Link Twist (α_i): It is the angle between Z_0 axis and Z_1 axis measured in a plane normal to X_1 axis [5].

The direction of Link Angle and Link Twist is determined using the right hand rule. It is important to note that the D-H parameters are implemented in the order of sequence of d_i, θ_i, a_i, and α_i respectively. The homogeneous transformation matrix between two successive links is defined using their D-H parameters. For example, the kinematic model

of the SCARA robot in Figure 15 is developed using D-H parameters presented in Table 2 below:

Table 2: D-H Parameters of SCARA Robot

Robot: SCARA (RRT)						
Joint	D-H parameters				Lower Joint Limit	Upper Joint Limit
	Link Offset (mm)	Joint Angle (rad)	Link Length (mm)	Twist Angle (rad)		
1	$d_1 =$ 1	$\theta_1 =$ θ_1	$a_1 =$ 225	$\alpha_1 =$ 0	-2.22	2.22
2	$d_2 =$ 1	$\theta_2 =$ θ_2	$a_2 =$ 225	$\alpha_2 =$ 0	-2.53	2.53
3	$d_3 =$ d_3	$\theta_3 =$ 0	$a_3 =$ 225	$\alpha_3 =$ 0	-297	-97

5.2 *Homogeneous Frame Transformations*

The homogenous transformation matrices help define rigid motions of Cartesian coordinate frames in a matrix formulation. A general structure of a homogenous transform matrix, A_i^{i-1} is represented in Equation 13 below.

$$A_i^{i-1} = \begin{bmatrix} R_{3x3} & P_{3x1} \\ 000 & 1 \end{bmatrix}$$
(13)

In kinematic modelling, the top left corner of the homogeneous transform matrix represents the rotation matrix (R_{3x3}), the top right corner represents the position matrix (vector P_{3x1}), the zeroes represent perspective and 1 represents the scaling factor. The matrix *A* represents the pose elements of frame *i* with respect to frame *i-1*. A basic homogeneous transformation matrix is computed from the D-H parameters using Equation 14.

$$A_i^{i-1} = Trans(Z, d_i)\, Rot(Z, \theta_i)\, Trans(X, a_i)\, Rot(X, \alpha_i)$$
(14)

Here, the sequence of multiplication is followed in the order of D-H parameters. The sequence being translation of d_i in Z_{i-1} axis, rotation of angle θ_i about the Z_{i-1} axis,

translation of a_i in direction of X_θ axis, and lastly the rotation of angle α_i about the X_i axis. These individual rotations and translations are represented in Equations 15-18.

$$Trans(Z, d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

$$Rot(Z, \theta_i) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}$$

$$Trans(X, a_i) = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

$$Rot(X, \alpha_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$

Substituting Equations $15 - 18$ in Equation 14, A_i^{i-1} can be represented as:

$$A_i^{i-1} = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i)\sin(\theta_i) & \sin(\alpha_i)\sin(\theta_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & -\sin(\alpha_i)\cos(\theta_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

The homogenous transformation matrix from Equation 19 is representative of all four D-H parameters and determines the pose for frame F_i with respect to F_{i-1}. A_i^{i-1} is of considerable significance because of its use in computation of the forward kinematics equation(s) and determination of complete workspace for a manipulator.

5.3 Joint Space

The joint space or configuration space of a manipulator is the set of all possible combinations of joint variables for a manipulator. Each joint variable of a manipulator has a defined range of motion that is represented as a vector. The combinations of these vectors in order of their joints defines the joint space of the manipulator. The number of vector(s) in the joint space is equal to the number of joints in a manipulator. For a manipulator with n joints and a range of i values for each joint configuration, the joint vectors can be defined with Equation 20. The joint space is then defined by Equation 21 as i^n sets of these vectors.

$$q_n = [q_n^1 \quad q_n^2 \quad ... \quad q_n^i]^T \tag{20}$$

$$q = [q_1 \quad q_2 \quad ... \quad q_n]^T \tag{21}$$

For example, for a SCARA (RRT, n=3) robot, if all joint variables assume 10 values each, then the joint space for that manipulator will have 1000 (10^3) sets of Equation 22.

$$q = [\theta_1 \quad \theta_2 \quad d_3]^T \tag{22}$$

5.4 Cartesian Space

The Cartesian space, v of a manipulator is the set of all possible combinations of position and orientation of the manipulator's end-effector. The Cartesian space has 6 DOF since it can always be represented by 3 position vectors and 3 orientation vectors (RPY angles) as represented by Equation 23.

$$v = [x \quad y \quad z \quad \alpha \quad \beta \quad \gamma]^T \tag{23}$$

Since the position and orientation of the end-effector is determined by the joint configuration of a manipulator, all sets in Cartesian space can be mapped back to at least one set in the manipulator's joint space. Since homogenous transformation matrices represent the pose of a manipulator's end-effector, they are used to define Cartesian space

of a manipulator. The developed reconfigurable model for this research uses elements from the pose matrices for improved ANN performance as described in Chapter 6.

5.5 Forward Kinematics

Forward kinematics for rigid manipulators is concerned with the computation of a manipulator's end-effector position and orientation for every known possible combination of its joint variables. Forward kinematic computations are straightforward and there always exist a forward kinematic solution for a manipulator in its joint space. For any n-link manipulator, the forward kinematic computation can be mapped from a configuration set in the joint space to a point in the Cartesian space of the manipulator using Equation 24.

$$Joint\ Space\ (q_1, q_2, q_3 \dots q_n\ ) \xrightarrow{f} Cartesian\ Space\ (\alpha_n, \beta_n, \gamma_n, p_x, p_y, p_z) \quad (24)$$

The forward kinematic equation(s) are computed using the homogeneous transformation matrices. These matrices are multiplied in succession to obtain the homogenous transformation for joint i with respect to frame, F_0, as seen in Equation 25.

$$A_i^0 = A_1^0 . A_2^1 . A_3^2 . \dots A_i^{i-1} = \begin{bmatrix} n_x & b_x & t_x & p_x \\ n_y & b_y & t_y & p_y \\ n_z & b_z & t_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} for\ i = 1,2.., k \quad (25)$$

where n, b, and t represent orientation vectors for defining the orientation of link k. For example, for a SCARA robot (RRT) (Figure 15), we obtain the forward kinematic equations by multiplying all three individual homogenous matrices in Equation 19.

A_3^0

$$= \begin{bmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) & 0 & 450 * \cos(\theta_1 + \theta_2) + 225 * \cos(\theta_1) \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) & 0 & 450 * \sin(\theta_1 + \theta_2) + 225 * \sin(\theta_1) \\ 0 & 0 & 1 & d_3 + 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (26)$$

The forward kinematics equation (Equation 26) here can now be substituted with the joint variable ranges from Table 2 to obtain the position and orientation of the SCARA robot's end-effector thereby defining its work envelop or complete workspace. It is important to note that the position and orientation of the end-effector is found with respect to the base frame, F_0, of a robot.

5.6 Workspace and Taskspace

The workspace of an industrial manipulator is a manifold of all points reachable by the manipulator's end-effector. Each point in a manipulator's workspace can be realized in at least one position and orientation configuration. The topology and volume of the workspace is determined by the mechanical structure of a manipulator and its joint configurations. The workspace is divided into two categories:

1. Dexterous Workspace: The dexterous workspace is a collection of all points in a manipulator's workspace that the end-effector can reach in all possible orientations. For example, if the joint configuration allows the manipulator to be oriented in all of its possible 10 orientations at a point 'P' in 3-D space. The point P is then said to be a part of the dexterous workspace of a manipulator.
2. Reachable Workspace: The reachable workspace is the collection of all points in a manipulator's workspace that the end-effector can reach in at least one orientation. For example, if the joint configuration of the manipulator allows the manipulator to be oriented in only 2 of its possible 10 orientations at a point 'Q'. The point Q is the said to be a part of the reachable workspace of the manipulator. The dexterous workspace of the manipulator is therefore a subset of the reachable workspace of a manipulator. [2]

The workspace of the manipulator is formulated using the forward kinematics equation(s) of the manipulator in Equation 25. Each point in the workspace is representative of the position matrix of the manipulator. The reconfigurable model presented in this research helps visually map the workspace of any manipulator configuration. This analysis helps to understand and appropriately modify a manipulator's geometric properties and its associated mechanisms for a desired workspace topology and volume. A sound understanding of the workspace also helps in

path planning for the end-effector through the manipulator's taskspace. Appendix A provides the third angle orthographic projections and an isometric view of the SCARA (RRT) robot's workspace discussed previously in this text.

The taskspace of a manipulator on the other hand is determined by the task required of the manipulator's end-effector. The taskspace has a varying dimensionality which is determined by the Degrees of Freedom needed to accomplish a task. The maximum dimension of the task space is 6 since the position and orientation of any object can be defined using 6 DOF. For example, if a manipulator is only concerned with positioning its end-effector regardless of the orientation, the task space for that manipulator has a dimension of 3. It is important to note that the joint space of the manipulator should be equal to its task space for a realizable inverse kinematic solution.

5.7 Inverse Kinematics

The inverse kinematics problem is related to the joint space of the industrial manipulators and depends strictly on the structure and configuration of a given manipulator. The end-effector of a manipulator works in Cartesian space but the actuators required to control the individual links work in its joint space. Thus, the computation of these joint variables from the end-effector position and orientation in Cartesian space is known as the inverse kinematics problem and is an essential tool for control of manipulators. For any n-link manipulator, the inverse kinematic computation can be mapped from the Cartesian space to the joint space of the manipulator using Equation 27.

$$Cartesian\ Space\ (\alpha_n, \beta_n, \gamma_n, p_x, p_y, p_z) \xrightarrow{f^{-1}} Joint\ Space\ (q_1, q_2, q_3 \ldots q_n\ ) \qquad (27)$$

The equations for computing an inverse kinematic solution are generated by comparing and analyzing Equation 25 with a forward kinematics solution for any manipulator. For example, the inverse kinematic equation(s) for a SCARA robot with a known forward kinematic solution can be analyzed from Equation 28 – Equation 32.

$$n_x = \cos(\theta_1 + \theta_2) \qquad (28)$$

$$n_y = \sin(\theta_1 + \theta_2) \tag{29}$$

$$p_x = 450 * \cos(\theta_1 + \theta_2) + 225 * \cos(\theta_1) \tag{30}$$

$$p_y = 450 * \sin(\theta_1 + \theta_2) + 225 * \sin(\theta_1) \tag{31}$$

$$p_z = d_3 + 2 \tag{32}$$

An inverse kinematics problem is therefore a reverse computation of the forward kinematics problem. The inverse kinematics solution for planar, and 3 or less DOF can be easily determined through some geometric, algebraic, and or analytical manipulations. However, with increasing DOF, the inverse kinematics problem proves to be mathematically complex and computationally expensive. With increasing DOF, kinematic decoupling of joint variables is often challenging and a closed form solution may not always be possible. For algebraic manipulations, the expressions for the joint variables are primarily determined from the x, y, and z coordinates of the position vector. Since, it is possible to have more than one solution to a coordinate point, it can be challenging to obtain inverse kinematic solutions for higher DOF manipulators. For example, four possible inverse kinematic solution(s) of a PUMA 560 robot are presented in Figure 17 below.
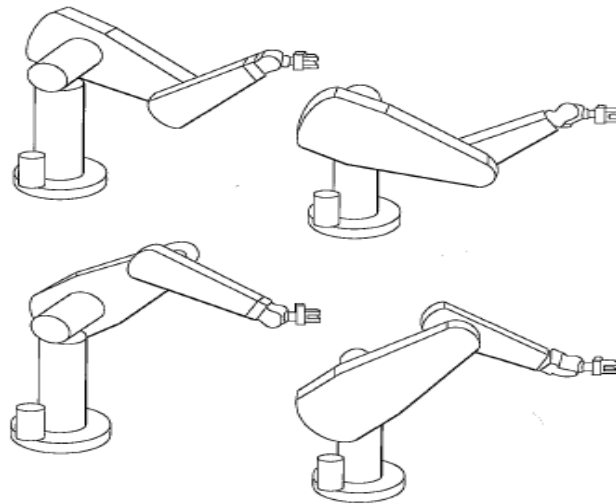
Figure 17: Four Different Inverse Kinematic Solution for PUMA 560 Robot [59]

The formulation of an inverse kinematic solution has a wide range of applications in the field of robotics. Most of the problems involving a robotic manipulator deal with orienting and positioning the end-effector in the Cartesian space. An efficient way to control the end-effector is through effective control of the actuated joints of the robot, which lie in the manipulator's joint space. It is therefore essential to map the Cartesian space constraints into the robot's joint space using inverse kinematics computations [60]. In cases where a closed form solution is not possible, a numerical method might be utilized to determine a possible set of solutions for the joint variables. There has been extensive research in the field of robotics for developing inverse kinematic solution(s) for specific robot models, configurations, and types. However, no universal model for computation of the inverse kinematics problem exists which can provide a solution with an acceptable level of accuracy. This research addresses the issue of developing a non-conventional technique of addressing this problem through the use of ANNs using discrete data sets.

CHAPTER 6

ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks (ANNs) are biologically inspired mathematical models that learn from their environment, similar to the neurons in the human nervous system. These mathematical models consist of multiple interconnected neurons that act as adaptive, and generally non-linear learning machines [61] [62]. The neurons in an ANN are its processing elements that help approximate any finite non-linear model to determine the relationship between its dependent and independent variables. The interconnectivity of these neurons defines the topology of an ANN.

ANNs are used for a variety of tasks including classification, clustering, prediction etc. This is because these networks can acquire and store knowledge through a defined learning process. [62] A feed-forward back-propagation multilayer perceptron (MLP) neural network model with supervised learning technique is used for this research to address the inverse kinematic problem in industrial manipulators. The results from this model are further discussed in the cases studies presented in Chapter 10. An understanding into the network architecture and its function are presented in this chapter to help realize the configuration of an optimum network used for this research.

6.1 Trade-off between Generalization and Accuracy

Generalization is the capability of an ANN to negate the effect of noise or any peculiarities that might be present within a dataset. Generally, a robust network with a good generalization capability provides a well fitted curve through the training data set. As a general rule, the simpler the network architecture, the better is its generalization capability. An accurate network on the other hand has a superior fit to training data than a network with good generalization capability. However, the trade-off here is the complexity and brittleness of a network. A brittle network is only tailored to the specific dataset it was trained on. Such a network lacks the capability of generalizing similar dataset(s). It is therefore important to optimize the degree of complexity of the neural network for a model that is both accurate and generalizes well [63].

6.2 Network Architecture

A basic ANN architecture consists of data inputs that are connected to neurons. The neurons process this input information and provide data outputs. All information in an ANN flows through the connections between these inputs, neurons, and outputs. These connections are scaled by adjustable parameters called weights, w_{ij} [61]. The weights of a neural network impart flexibility to the network thereby helping it to adapt and learn the pattern(s) in a data set. The bias (generally assumed a value of 1) in a network represents the factors that are not accounted for by the input variables

A Multi-Layer Perceptron (MLP) network architecture is used in this research because of its capability to perform complex prediction tasks. Figure 18 represents a general MLP architecture. Here, n represents the number of inputs, m represents the number of neurons in the hidden layer, x_n represents the input variables, z represents network output, a_{nm} represents the weight from the nth input variable to the mth neuron in the hidden layer, b_m represents the weight from the mth neuron in the hidden layer to the output layer, a_{0m} represent the bias to the mth neuron in the hidden layer, b_0 represents the bias to the output layer, and σ, and f(.) represent the activation functions used in the neurons.
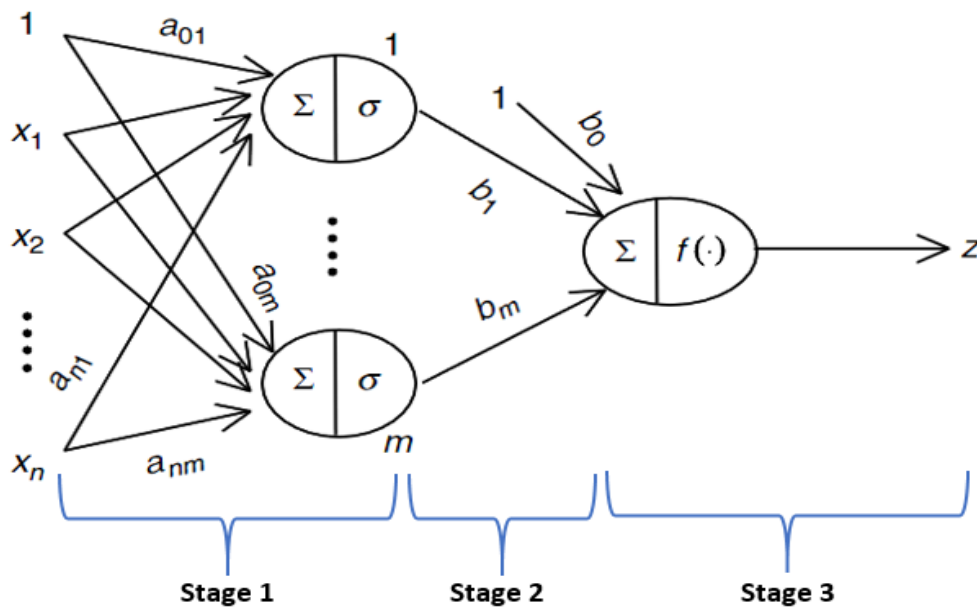
Figure 18: Multi-Layer Perceptron Architecture [62]

45

The network learning can be described in the following stages:

1. Stage 1: The hidden neurons sum the weighted inputs and pass them through the activation function.
2. Stage 2: The outputs from the hidden layer are fed to the output layer with a second set of weights and a bias.
3. Stage 3: The output layer passes the weighted sum of its inputs through a linear or non-linear activation function to the network's output.

The output(s) from the output layer make up the network outputs(s). The network output(s) are subsequently analyzed for network performance and errors.

6.3 Network Learning

A feed-forward back-propagation batch learning with a supervised learning technique is used to train the network. A feed-forward network structure only allows a unidirectional flow of data through the network. The flow of data is usually from the input layer through the hidden layer, and finally to the output layer. Feedback loops or cycles are not permitted in a feed-forward network.

Learning for an ANN is the adjustment of its weights and biases to minimize error in the network. A back-propagation learning type is used in the network developed for this research. Back-propagation of error allows the network to calculate the error at each output and adjust the value of weights that caused the error accordingly, thereby reducing the overall error in the network. The effect of each weight on the error is determined by the value of the weight and the error on the unit above [63]. The error is thus back-propagated through the network for optimization of the weights such that if the same dataset is provided to the network again, the error is lower than the previous result. The error indicator considered for the network performance is the mean squared error (MSE) value represented in Equation 33. The MSE value determines the accuracy of prediction over all the training patterns for a given network

$$E = \frac{1}{2N}\sum_{i}^{N}(t_i - z_i)^2 \tag{33}$$

where, E is the MSE value, t_i is the target for the ith training pattern, z_i is the predicted output for the ith training pattern, and N is the total number of training patterns. A batch learning technique involves the network learning after the entire data set has been presented to it, or more simply when one whole epoch is run. "An epoch refers to a single pass of all input patterns in a perceptron during the training phase [62]." The network computes a resultant error gradient with respect to weights from the average of error gradients from each point in the dataset. The error is minimized in the direction of the descent indicated by this resultant gradient. [62]

A supervised learning technique trains the network by providing a target to the network along with its corresponding input during training phase. This allows the network to be exposed to a known response. The network subsequently learns the system behaviour under specific conditions characterized by the data presented to it [64]. The Levenberg-Marquardt (LM) learning algorithm is used here to adjust and update the weights of the network. LM is a hybrid learning technique based of the Gradient Descent and Newton's method. The algorithm as presented in Equation 34, helps to train a network to attain a global minimum error by minimizing the first derivatives or gradients to zero [62]. This training algorithm is known to demonstrate superior performance and efficiency by adjusting the learning rate of the network repeatedly. [65]

$$\Delta w_m = - \frac{d_m}{d_m^s + e^\lambda} \tag{34}$$

where, d_m is the first derivative of error, d_m^s is the second derivative of error, and λ is the damping factor.

6.4 Activation Function

Activation functions are the processors of data in a neuron and help the weights in the network identify and learn trends in a dataset. These functions can help introduce non-linearity into the network which allows the network to process complex, and non-linear datasets. Activation functions in the hidden layer(s) are non-linear continuous functions. The continuity of these functions allows them to be differentiable. This

property aids in the adjustment of the network weights during backpropagtion of errors [62]. Generally, non-linear sigmoid functions are used as processors in MLPs [61]. The sigmoid function class can be classified in three common non-linearities, namely, logistic, hyperbolic tangent, and threshold functions. The logistic function (Equation 35) constrains the input data within a range of [0, 1] and is represented by Figure 19. The activation function, used for this research, for the network's hidden layer is the hyperbolic tangent function given in Equation 36.

$$logsig(u) = \frac{1}{1 + e^{-\beta u}} \tag{35}$$

$$tanh(u) = \frac{1 - e^{-2u}}{1 + e^{-2u}} \tag{36}$$

where, β is the slope parameter, and u is any value from a dataset. Hyperbolic tangent functions constrain the data from [-1, 1] as seen from Figure 20. Unlike the logistic function, this function is beneficial when the data set to be trained has both positive and negative values in its input dataset and target dataset. The data can then be normalized before being fed to the network for an improved performance. It is also important to note that an asymmetric hyperbolic tangent function leads to a faster learning by requiring fewer number of patterns presented to it than non-symmetric logistic function [63].
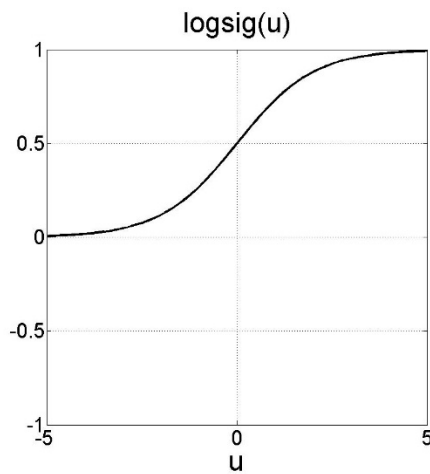
Figure 19: Logistic Function   Figure 20 : Hyperbolic Tangent Function [62]
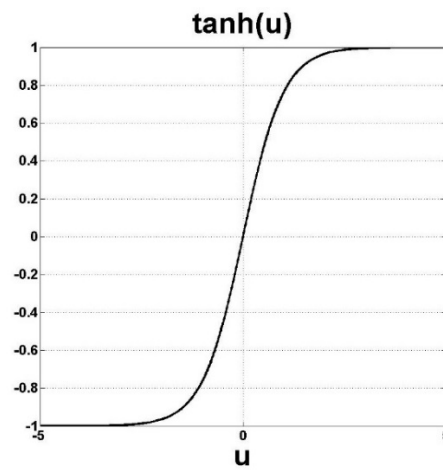
48

A threshold activation function maps the data based on a predefined threshold, t. If the input value is above the threshold, then the output is t_1. If the input value is below the threshold, the output value is t_0. The threshold function acts as a binary classifier and is best suited for clustering, and pattern recognition applications. Figure 21 represents a threshold function with $t_1=1$, and $t_0=0$ given by Equation 37.

$$threshold(u) = \{ \begin{matrix} t_0 & if\ input < 0 \\ t_1 & if\ input > 0 \end{matrix} \tag{37}$$
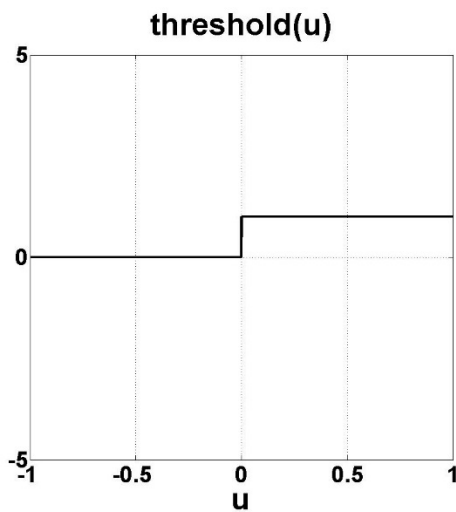
Figure 21: Threshold Function   Figure 22: Linear Function
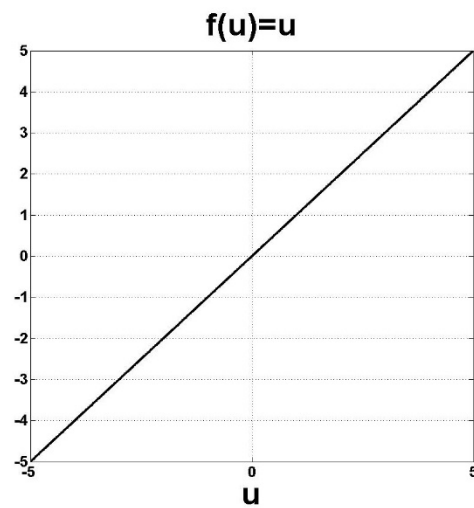
The output from a neuron using any activation function, f, is given by Equation 38.

$$Neuron\ Output = f(\sum_{j=1}^{n} w_j x_j + b) \tag{38}$$

The output layer on the other hand uses a linear activation function given by Equation 39, as its processing unit. Unlike the hyperbolic tangent function, the linear activation function (Figure 22) does not constrain the data but rather scales it linearly. This helps attain a true output value with respect to the network input.

$$f(u) = u \tag{39}$$

6.5 Data Pre-Processing and Post Processing

Data pre-processing is an important step in the data mining process. The quality of data and its results can significantly be improved with the correct pre-processing techniques. One such technique, normalization, has been used here for the development of an inverse kinematic solution using ANNs. The physical attributes of a manipulator dictate its D-H parameters. These parameters are often a different scale than the joint variable ranges of the manipulator. The difference in scale may mask the effect of one variable on another. Normalization of data is therefore essential to scale all input and target datasets in a pre-defined range. The pre-defined range used for training the network is [-1, 1]. This guarantees a stable convergence of weights and biases. Normalization also helps to identify the true effect of any one variable on another variable. Two normalization techniques, namely, min-max normalization (Equation 40), and z-score normalization (Equation 41) have been applied to the dataset(s) used for training the network.

$$X' = \frac{a + (X - X_{min})(b - a)}{X_{max} - X_{min}} \quad \text{or,} \tag{40}$$

$$X' = \frac{X - \bar{X}}{\sigma} \tag{41}$$

where, X denotes any value in the data set, X' denotes the normalized value of X, a = -1, b = 1, $\bar{X}$ = mean of the given variable, σ is the standard deviation of the dataset, X_{max}, and X_{min} are the maximum and minimum values in the dataset respectively.

The network outputs from a normalized input set are also normalized. All values in the output data set therefore need to be reverted to scale. The scale for de-normalizing an output dataset is determined from the range of target dataset supplied to the network.

6.6 Division of Data

The ANNs for this research were developed with the aid of the Graphical User Interface (GUI) Neural Network (NN) Toolbox in the MATLAB environment. For

training a network, all data was divided at random into three mutually exclusive and collectively exhaustive categories, namely:

1. <u>Training Set</u>: The training set is a percentage of the original data provided to a network to adjust the weights of the network during training. The training set used for this research accounts for 80% of the original data selected at random by the NN Toolbox.

2. <u>Validation Set</u>: The validation set is a percentage of the original data provided to a network to minimize over fitting. The validation set verifies if an increase in accuracy over the training set yields an accuracy in the validation set as well. The network starts over fitting if the accuracy over the training set increases while the accuracy over the validation set decreases or remains constant. The training of a network should be stopped at this point. For this research, the validation set accounts for 10% of the original data selected at random by the NN Toolbox.

3. <u>Testing Set</u>: The testing set is a percentage of the original data provided to a network to independently measure the networks performance and prediction capability after training has commenced. For this research, the testing set accounts for 10% of the original data selected at random by the NN Toolbox [66].

It is important to note that a data division percentage of 80-10-10 was chosen for the input data set over the MATLAB default percentage of 70-15-15. This configuration was selected since the ANN yielded a superior performance when compared to the default configuration. Better performance was achieved since the network was able to train over a larger dataset range while the validation and testing dataset performance remained constant.

6.7 Network Prediction Capability

After learning commences, input(s) from a known input-target dataset are introduced to the trained ANN. The network is simulated over the inputs to obtain network outputs. These outputs are the predicted values from the trained network. The outputs are compared with the known targets for errors in prediction. The relative percentage error in prediction is calculated, for a target dataset with no zero values, using Equation 42.

$$E_P = \frac{t_i - z_i}{t_i} X \ 100 \qquad (42)$$

where, E_P is the percentage error in prediction, t_i is the ith target value of the dataset, and z_i is the ith output from the ANN. If the target dataset contains values that are zero, a percentage error cannot be computed since the numerator in Equation 42 would require division with 0. In such cases, absolute error is computed using Equation 43.

$$E_A = |t_i - z_i| \qquad (43)$$

where, E_A is the absolute error in prediction. It is important to note that the absolute error requires reverting values back to scale if the input dataset had previously been normalized.

6.8 Inverse Kinematics using Artificial Neural Networks

Inverse kinematics problem are classified as ill-posed problems in modelling of ANNs. An ill-posed problem is characterized by a consistent mapping of a single input on one or more output(s). In such a case, the network learning averages all possible solutions thereby yielding a poor performance [63]. In the case of industrial manipulators, when mapping the end-effector position and orientation to the joint variable configuration of a manipulator, an ill-posed problem is experienced. The problem arises because of several joint configurations producing the same end-effector pose. The network thus generalizes the dataset to produce an outcome with low accuracy.

This research presents ANNs as a non-conventional approach in solving the inverse kinematic problem in industrial manipulators. ANNs can be used in development of a robust and singularity free inverse kinematic solution. Figure 23 presents the network architecture used for this research. The network uses a dataset of 12 inputs which represent the position of the end-effector (p_x, p_y, p_z), and the orientation of the end-effector $(n_x, n_y, n_z, b_x, b_y, b_z, t_x, t_y, t_z)$ from the forward kinematics equations. The targets and network outputs are the configurations of the joint variables $(q_1, q_2, q_3, q_4, q_5, q_6)$ that produce the input position and orientation.
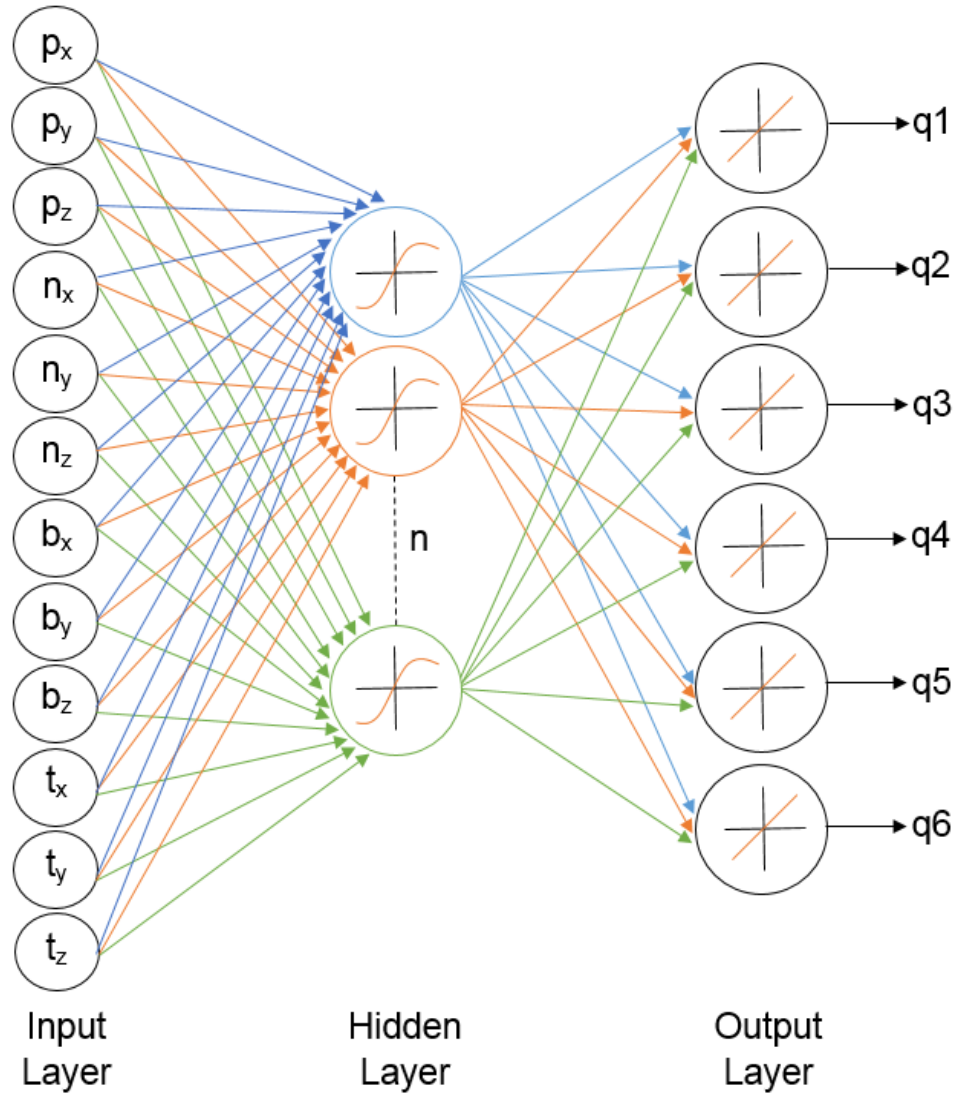
Figure 23: ANN Architecture for Inverse Kinematics Problem

6.8.1 Challenges in developing an ANN Architecture

The network architecture was initially designed with only 6 inputs $(p_x, p_y, p_z, \alpha, \beta, \gamma)$ as the rotation matrix was consolidated into its corresponding RPY angles. This architecture demonstrated a far lower performance compared to using all 9 elements of the rotational matrix. This is because the network learning increasing with an increase in input parameters for a given number of outputs. The increase in 6 additional parameters help better define the joint configurations and the error can be generalized over a wider range of dataset. Figure 43 (Appendix A) shows the outputs of the network

(blue) completely superimposed on the network targets (green) thereby indicating a perfectly trained network for a SCARA Robot.

An inherent challenge while developing the ANN model for this research was availability of target data (joint variable configurations) for any assumed position and orientation of the manipulator's end-effector. Previously known target data is required for supervised learning as well as in the computation of errors in prediction (E_P). Due to the unavailability of inverse kinematic solution(s) for most industrial manipulators, a forward kinematics solution was first developed for each manipulator type. An input dataset was developed with the joint configurations used for the forward kinematic computation with the outputs from the forward kinematic computation as network inputs. These network outputs were subsequently compared with the targets for evaluating network performance.

Accuracy of the network was another challenge faced while developing an optimized network. It was observed that large amounts of data decreased the performance of the network because of the increase in complexity of the data set. For example, for a 6 DOF robot with 10 joint values for each joint variable configuration, 1 million joint configurations and their corresponding end-effector pose configurations were generated. The network learning process therefore involved processing of 18 million joint configurations (12 inputs + 6 outputs). Due to computational limitations (Intel® Core™ i7-3770 CPU @ 3.40 GHz processor, 16.0 GB RAM), such large data could not be processed through the Neural Network Toolbox in MATLAB. This data set was broken down into subsets by taking a smaller range of values within a single joint variable for ease of processing. Three different approach were experimented with to obtain a higher network performance, namely:

1. Restructuring the ANN: Altering the ANN architecture was the first approach taken to solve the aforementioned problem. This involved addition of neurons in the hidden layer as well as addition of several other hidden layers each with varying number of neurons. The network performance, however, did not substantially increase after 55 neurons in the first hidden layer. Restructuring the network henceforth only increased the complexity of the network. A single hidden

layer (SHL) network architecture with the least amount of complexity and comparable performance was therefore considered optimal.

2. <u>Different Learning Techniques</u>: Different learning techniques apart from feed-forward back propagation were tested for an increase in network performance. These techniques involved Elman back propagation, generalized regression, cascade forward back propagation etc. A feed forward back propagation network, however, provided the least amount of error in the system, and with a superior performance amongst all compared techniques.

3. <u>Reducing the Dataset Complexity</u>: Instead of splitting a large data set into subsets, smaller datasets were created with fewer joint configurations. This helped reduce the complexity of the dataset by significantly decreasing the learning required by the network. An optimal number of three joint configurations for each joint variable (729 joint and pose configurations) demonstrated superior results over any other approach taken to improve the network accuracy. The computation time of the network also decreased substantially with this approach. The trade-off for this approach was that only a range of 3 joint variable values could be trained with the developed network at any given time. Different classes of joint configuration therefore need to be developed when using this method. Chapter 10 presents case studies on two different manipulator configurations with an inverse kinematic solution for each manipulator type.

6.8.2 *Generalization and Accuracy of the ANN Model*

ANNs provide promising results in development of inverse kinematic solution(s). Moreover, the complexity of a solution is decreased since complicated coupled equations from iterative methods are not explored. ANNs also greatly reduce the computation time required for development of a solution as compared with other traditional geometric, iterative, and analytical methods. A challenge with ANN models, however, is the accuracy of a developed network. An acceptable level of accuracy is needed to make confident predictions. A model with an optimized complexity is required for an accurate model with good generalization capability.

Before modelling a dataset in Neural Networks, we assume that there is some acceptable level of noise present. Noise may arise from presence of singularities, error due to approximation etc. Since, reliable predictions for such a model cannot be made beforehand, the model needs to possess an optimal generalization ability (Figure 24), in order to prevent over fitting (Figure 25) or under fitting (Figure 26) due to high or low model flexibility.
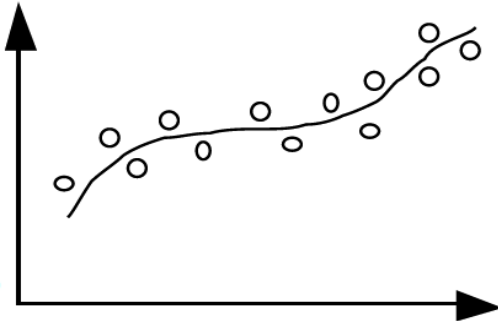
Figure 24: ANN with good generalization [62]   Figure 25: Over-fitted ANN with high flexibility [62]
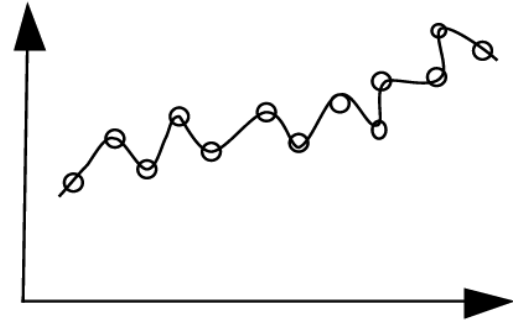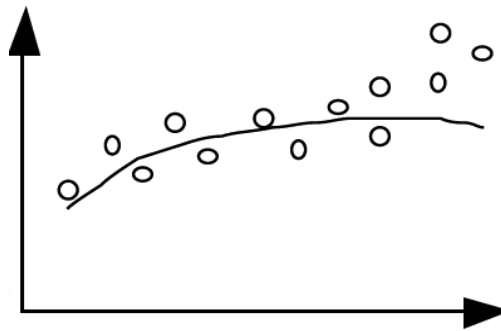
Figure 26: Under-fitted ANN with low flexibility [62]

For improving generalization, the network's DOF need to be lowered, which is achieved by reducing the number of free parameters, or the weights to each hidden neuron. These hidden neuron weights are directly proportional to the flexibility of the network. Reducing the number of neurons thus reduces over fitting. One has to be careful since excessive reduction in hidden neurons causes the model to under fit. In an early stopping approach, if the weights are allowed to grow enough during training and then training is stopped, it is possible to restrain the network from over fitting. A performance plot provides the epoch at which the lowest validation performance is achieved. After this point, over fitting sets into the model and the validation performance increases with

training. If weights are taken for the network at an optimal point where the validation performance is best, the network would fit sufficiently but not too close, which is an indication of a well-trained model [62].

Figure 27 represents the ANN model used in development of an inverse kinematic solution for the previously mentioned SCARA (RRT) robot. Using trial and error, it is observed that an ANN with 55 neurons in a hidden layer provides the optimal network generalization and accuracy. For training the network, a sample input and target data set was created from the forward kinematics model of SCARA Robot. Each joint variable was split in 25 equal sections over its range as given by Table 15 in Appendix A. The joint space of the manipulator therefore consisted of 15625 joint combinations (25^3) which were used as network targets. Each of these combinations produced an end-effector pose which were used as network inputs. Table 3 provides a summary of the performance indicators for the trained ANN.
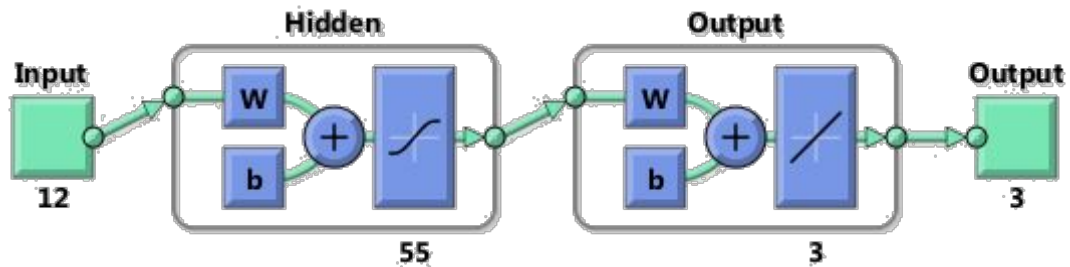
Figure 27: ANN Architecture for SCARA Robot

Table 3: ANN Performance Indicators for SCARA Robot

S. No.	ANN Network Indicator	Result
1	Total Epochs	501
2	Epoch for Best Validation Performance	501
3	Overall Regression (R) Value	1
4	Mean Square Error (MSE)	0.0000009
5	Training Performance	0.0000009
6	Testing Performance	0.0000010
7	Validation Performance	0.0000010
8	Error Histogram Center (Bell Curve)	-0.000072

The best validation performance, as seen from Figure 40 (Appendix A), was obtained at epoch 501. The network training was manually aborted at epoch 501 since excellent network performance was achieved. The regression plot in Figure 41 (Appendix A) demonstrates a fitness between the network outputs and target values. A perfect fit is indicated by a regression (R) value of 1. A perfect fit is achieved since all points in the network input data are unique. Moreover, an ill-posed problem is not encountered since every point in the input dataset is mapped to exactly one corresponding target data. Figure 43 (Appendix A) provides a comparison between the network outputs and targets for q_1, q_2, and q_3 as a solution to the inverse kinematics problem. It is observed that the outputs and targets for q_1, q_2, and q_3 completely superimposed on each other. This indicates a robust inverse kinematics solution for the SCARA manipulator. The network performance indicator, MSE, has an extremely low value of 0.0000009 (assume zero). A low MSE value indicates a good accuracy of prediction. The individual performance values for the training, testing, and validation dataset are extremely low and are around the MSE value as well. The error histogram in Figure 42 (Appendix A) determines the frequency of errors concentrated over a range. A well fit network has the maximum frequency of errors around zero. In the network trained for SCARA robot, the maximum errors in all training, validation, and testing dataset are concentrated at -0.000072. The error histogram here displays a perfect normal distribution (bell shaped curve) centered nearly at zero, thereby depicting a 95% and above confidence interval in prediction of joint variables.

CHAPTER 7

JACOBIAN: VELOCITY KINEMATICS

The Jacobian matrix is an essential tool in the analysis and control of manipulator motion. It is used in several aspects of robot manipulation including trajectory and path planning, singularity analysis, derivation of dynamic equations of motion etc. A Jacobian is the first derivative of the pose matrix of a manipulator. Mathematically, it defines the Cartesian linear and angular end-effector velocity relationship to a manipulator's joint variable velocities in its joint space. The Jacobian matrix thus computes the end-effector motion and Cartesian velocity caused by the actuation and rate of change of joints of a manipulator [57]. The derivation of a manipulator's Jacobian is highly dependent on the kinematic structure of the manipulator and its joint configurations. It is therefore essential to model a Jacobian that can adapt to changing kinematic structure(s) of any manipulator type. Two common techniques to model the Jacobian are the Newton-Euler Recursive method and the Vector Cross Multiplication (VCM) method. This research utilizes the Newton-Euler Recursive (NER) method because of its capability to be extended for calculation of dynamics equations of motion for a manipulator. During the course of this research, it was also realized that the NER method provides seamless integration into the development of a reconfigurable model without the need for assessment of several additional parameters when compared to the VCM method.

7.1 Newton Euler Recursive Method

The computation of the Newton-Euler Recursive equation(s) begin with defining the rotation matrix and the position matrix from the forward kinematics equations of a manipulator (Equation 25). The rotation matrix, its transpose, and the position matrix that define the orientation of a frame F_i with respect to F_{i-1} are represented in Equations 44-46 respectively.

$$R_i^{i-1} = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i)\sin(\theta_i) & \sin(\alpha_i)\sin(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & -\sin(\alpha_i)\cos(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) \end{bmatrix} \ for \ i = 1,2, \dots, n \qquad (44)$$

$$R_{i-1}^i = \left[R_i^{i-1}\right]^T \qquad (45)$$

$$P_i^{i-1} = \begin{bmatrix} a_i\cos(\theta_i) \\ a_i\sin(\theta_i) \\ d_i \end{bmatrix} \quad for \; i = 1,2,\dots,n \tag{46}$$

The angular and linear velocity vectors are subsequently determined for all joint variables. These joint rate vectors are the first derivatives of the joint variables and are defined in Equation 47 and Equation 48.

$$\dot{\theta}_i^{i-1} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_i \end{bmatrix} \quad for \; i = 1,2,\dots,n \quad for \; rotational \; joints \tag{47}$$

$$\dot{P}_i^{i-1} = \begin{bmatrix} 0 \\ 0 \\ \ddot{p}_i \end{bmatrix} \quad for \; i = 1,2,\dots,n \quad for \; translational \; joints \tag{48}$$

The next step involves determining the angular velocities ($^i\omega_i^0$), and linear velocities ($^iv_i^0$), of each link, i, based on the joint variable type. Equations 49-50 represent the angular velocities for rotational and translational joint types, and Equations 51-52 represent the linear velocities for rotational and translational joint types.

$$^i\omega_i^0 = R_{i-1}^i \left[^{i-1}\omega_{i-1}^0 + \dot{\theta}_i^{i-1} \right] \quad for \; rotational \; joints \tag{49}$$

$$^i\omega_i^0 = R_{i-1}^i \left[^{i-1}\omega_{i-1}^0 \right] \qquad for \; translational \; joints \tag{50}$$

$$^iv_i^0 = R_{i-1}^i \, ^{i-1}v_{i-1}^0 + \, ^i\omega_i^0 \, X \left(R_{i-1}^i P_i^{i-1} \right) \qquad for \; rotational \; joints \tag{51}$$

$$^iv_i^0 = R_{i-1}^i \, ^{i-1}v_{i-1}^0 + \, ^{i-1}\omega_{i-1}^0 \, X \left(R_{i-1}^i P_i^{i-1} \right) + R_{i-1}^i \dot{P}_i^{i-1} \tag{52}$$
$$for \; translational \; joints$$

After computation of angular and linear velocities for the last link of the manipulator, the generalized velocity vector (V) of the end-effector is computed using Equation 53.

$$V = \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{53}$$

For example, for a 6 DOF manipulator, the generalized velocity vector is a 6x6 matrix represented by Equation 54.

$$V = \begin{bmatrix} {}^{6}v_{6}^{0} \\ {}^{6}\omega_{6}^{0} \end{bmatrix} \tag{54}$$

The Jacobian matrix, $J(q)$, of a manipulator with respect to its end-effector is calculated from the generalized velocity vector(V) by extracting the joint velocity vector(s), $\dot{q}$. The joint velocity vectors vary depending on the type of joint for each link as represented in Equation 55.

$$\dot{q}_{i}^{i-1} = \begin{cases} \dot{\theta}_{i}^{i-1} \ for \ rotational \ joint \ i \\ \dot{P}_{i}^{i-1} \ for \ translational \ joint \ i \end{cases} \tag{55}$$

The Jacobian matrix, $J(q)$, is represented in Equation 56.

$$V = J(q)\,\dot{q} \tag{56}$$

This Jacobian matrix can further be divided into two submatrices representing the Jacobian for linear velocities, J_v and the Jacobian for angular velocities, J_ω, as represented in Equation 57.

$$J(q) = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \tag{57}$$

The dimension of the Jacobian matrix is dependent on the number of joints of a manipulator, n, and the dimension of the task space, t. For an n-DOF manipulator, the Jacobian matrix has a dimension of *txn*. Since most industrial manipulators are required to position as well as orient its end-effector, the dimension of the task space is generally 6. The dimension of the manipulator Jacobian is therefore usually *6xn*. In such as case, the dimensions of both J_v and J_ω will be *3xn*.

The Jacobian matrix with respect to the base frame, F_0, is calculated using the Equation 58.

$$J(q)_B = R_n^0 \, J(q) \qquad (58)$$

where, n is the number of joints in a manipulator, and R_n^0 represents the rotation matrix defining the orientation of the end-effector with respect to the base frame of the manipulator. Once computed, $J(q)_B$ is further analyzed for any kinematic singularities present in the manipulator. For example, for a SCARA (RRT) manipulator previously discussed in this text, the generalized velocity vector, V and the Jacobian matrix in the base frame, F_0, are represented by Equations 59-60 respectively.

$$V = J(q)\,\dot{q} = \begin{bmatrix} 225 * \sin(\theta_2) & 0 & 0 \\ 225 * \cos(\theta_2) + 450 & 450 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{p}_3 \end{bmatrix} \qquad (59)$$

$$J(q)_B = \begin{bmatrix} -450 * \sin(\theta_1 + \theta_2) - 225 * \sin(\theta_1) & -450 * \sin(\theta_1 + \theta_2) & 0 \\ 450 * \cos(\theta_1 + \theta_2) + 225 * \cos(\theta_1) & 450 * \cos(\theta_1 + \theta_2) & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \qquad (60)$$

7.2 Wrist Partitioned Manipulators

Consider a general 6-DOF articulated industrial manipulator with a forearm configuration in its first 3 joints, and a wrist configuration in its last 3 joints. If the velocity reference point is considered as the center of the manipulator's wrist, the Jacobian matrix, $J(q)_B$, can be further simplified into another matrix, J_W, with 4 sub-matrices as represented in Equation 61 [67].

$$J(q)_B = J_W = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \qquad (61)$$

Here, J_{11}, and J_{22} are 3X3 matrices can that can be individually analyzed for decoupling singularities. Often in some manipulator geometries where the last three joint variables only affect the orientation of the end-effector, J_{12} will be a zero matrix of a dimension 3x3. This zero block matrix simplifies the decoupling process which is discussed further in detail in Chapter 8. The simplification of the manipulator Jacobian ($J(q)$ in any frame) into sub-matrices can help identify the relations between the forearm and wrist configurations, and the linear and angular velocity vectors [57] using Equations 62-67.

$$J_{11}\dot{q}_a = \begin{bmatrix} J(q)_{11} & J(q)_{12} & J(q)_{13} \\ J(q)_{21} & J(q)_{22} & J(q)_{23} \\ J(q)_{31} & J(q)_{32} & J(q)_{33} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} \tag{62}$$

$$J_{12}\dot{q}_b = \begin{bmatrix} J(q)_{14} & J(q)_{15} & J(q)_{16} \\ J(q)_{24} & J(q)_{25} & J(q)_{26} \\ J(q)_{34} & J(q)_{35} & J(q)_{36} \end{bmatrix} \begin{bmatrix} \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix} \tag{63}$$

$$J_{21}\dot{q}_a = \begin{bmatrix} J(q)_{41} & J(q)_{42} & J(q)_{43} \\ J(q)_{51} & J(q)_{52} & J(q)_{53} \\ J(q)_{61} & J(q)_{62} & J(q)_{63} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} \tag{64}$$

$$J_{22}\dot{q}_b = \begin{bmatrix} J(q)_{44} & J(q)_{45} & J(q)_{46} \\ J(q)_{54} & J(q)_{55} & J(q)_{56} \\ J(q)_{64} & J(q)_{65} & J(q)_{66} \end{bmatrix} \begin{bmatrix} \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix} \tag{65}$$

$$v = J_{11}\dot{q}_a + J_{12}\dot{q}_b \tag{66}$$

$$\omega = J_{21}\dot{q}_a + J_{22}\dot{q}_b \tag{67}$$

For example, for any wrist partitioned Cartesian manipulator, $J_{12} = J_{21} = 0$, thereby verifying that the linear velocity of the end-effector is independent of the rotational joints in the manipulator's wrist. Also, the angular velocity is independent of the first three translational joints [57].

CHAPTER 8

KINEMATIC SINGULARITIES

The American National Standard for Industrial Robots and Robot Systems – Safety Requirements (ANSI/RIA R15.06-1999) defines kinematic singularity as "a condition caused by the collinear alignment of two or more robot axes resulting in unpredictable robot motion and velocities" [68]. A manipulator's performance is therefore greatly depreciated at or near singular regions. It is thus crucial to understand the functionality and reachable workspace, void of any singularities, for a manipulator's enhanced performance in an industrial setting [2].

Kinematic singularities in manipulators arise due to a loss of DOF in its end-effector [2]. At such an instance, two or more joints of a manipulator do not independently control the position and orientation of the end-effector [1]. For example for a SCARA (RRT) manipulator, the singularity region is marked red in color in Figure 28 below and Figure 38 (Appendix A). At these singular regions, the position and orientation of the SCARA manipulator is only controlled by one rotational joint and one translational joint.
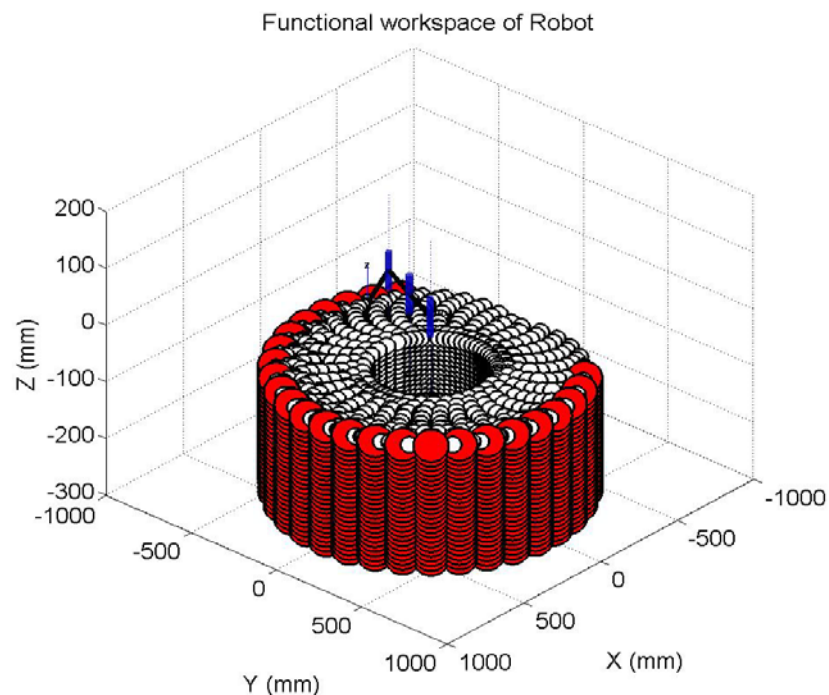
Figure 28: Singularity Space of SCARA Robot

Kinematic Singularities are of particular interest for the following reasons:

1. Knowledge about singularities provides an insight into the reachable and functional workspace for the end-effector of a manipulator.
2. Singular configurations (boundary singularities) may sometimes help define the boundary of the manipulator's workspace.
3. Singularity can be used as design tool for defining the joint limits and the mechanical structure of a manipulator.
4. Singularities help determine configurations for unattainable directions of motion.
5. At singular configurations, small motion of the manipulator's end-effector may cause a large movement in the joint variables.
6. At or near singular configurations, the control algorithm of a manipulator fails, resulting in large joint velocities and accelerations for the smooth operation of the manipulator.
7. Singular configuration may correspond to non-unique, zero or infinite inverse kinematic solutions to a manipulator [69] [5] [2].

During manipulator control, singularity conditions may arise during the inverse mapping from the manipulator's Cartesian space to its joint space [70]. By modifying Equation 56, it can be seen that the joint velocity vector of the manipulator in its joint space, can be mapped to the generalized velocity vector of the manipulator in Cartesian space using Equation 68.

$$\dot{q} = [J(q)]^{-1} V \qquad (68)$$

Singularities can therefore be mathematically determined by analyzing the inverse of the Jacobian matrix for the manipulator being studied. From a mathematical standpoint, singularities arise as a local or instantaneous phenomena from the rank deficiency of the Jacobian matrix [69]. To realize a solution to Equation 68, the Jacobian matrix of a manipulator should be non-singular, and be of a rank equal to the dimension of the joint velocity vector and generalized velocity vector. One method of analyzing a kinematic singularity is through the computation of the determinant of an *nxn* subset, J_n of the manipulator Jacobian, where *n* represents the number of joints. A square subset is

analyzed for a non-square Jacobian (with 6 or less DOF) since an inverse of a non-square matrix does not exist. Mathematically, the inverse of a Jacobian matrix is represented in Equation 69.

$$[J(q)]^{-1} = \frac{C_{ji}}{|J(q)|} \tag{69}$$

where, C_{ji} represents a matrix of cofactors (adjugate matrix) of the Jacobian being analyzed, and $|J(q)|$ represents its determinant. For a non-invertible singular Jacobian, the determinant of the matrix is zero as represented in Equation 70.

$$Singular\ Jacobian: |J(q)| = 0 \tag{70}$$

For the SCARA (RRT) manipulator, the 3x3 subset of its Jacobian matrix being analyzed is the Jacobian matrix of linear velocities in base frame, $J_v(q)_B$ represented by Equation 71. The determinant of this Jacobian is represented by Equation 72:

$$J_v(q)_B = \begin{bmatrix} -450*\sin(\theta_1+\theta_2)-225*\sin(\theta_1) & -450*\sin(\theta_1+\theta_2) & 0 \\ 450*\cos(\theta_1+\theta_2)+225*\cos(\theta_1) & 450*\cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{71}$$

$$|J_v(q)_B| = 101250*\sin(\theta_2) \tag{72}$$

The determinant of the Jacobian, $|J_v(q)_B|$ assumes the value 0 when $\theta_2 = 0$ or *pi radians*. Since *pi radians* does not lie in the joint space of the manipulator (see Table 2), the singular condition for this robot arises when its second joint variable reaches *0 radians*. In this case, the singular space is on the boundary of the manipulator because at $\theta_2 = 0$ *radians*, the arm of the manipulator is fully extended and cannot move any farther from its base.

8.1 Types of Singularities

With respect to general wrist partitioned industrial manipulators, kinematic singularities can be classified based on the joint configuration(s) of the manipulator. The two most common types of kinematic singularities are:

1. Forearm Singularity: In wrist partitioned 6 DOF manipulators, forearm singularities arise because of the motion of the forearm caused by first three joints of the manipulator. These singularities are often experienced at the workspace boundary of the manipulator when the manipulator arm is fully extended or retracted. Arm singularities are therefore sometimes referred to as boundary singularities or internal singularities based of the arm configuration. Forearm singularities can be identified by analyzing the J_{11} subset of the Jacobian matrix for a manipulator. A forearm singularity can be mathematically represented using Equation 73.

$$|J_{11}| = 0 \qquad\qquad (73)$$

For a wrist partitioned SCARA robot, a forearm singularity is observed at $\theta_2 = 0$ or *pi radians* (Equation 72), as seen in Figure 38 (Appendix A). At this configuration, the arm of the manipulator is at its maximum radial distance from the base of the manipulator as seen from Figure 29 below.
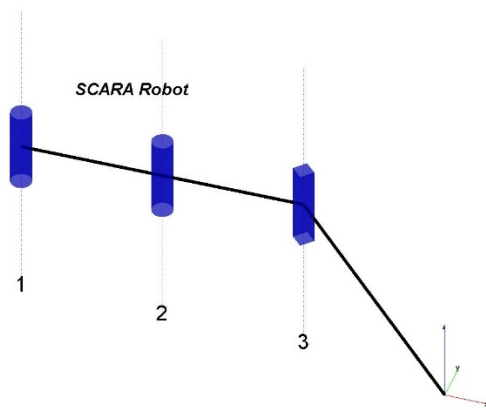
Figure 29: SCARA Robot   Figure 30: PUMA 560 Robot
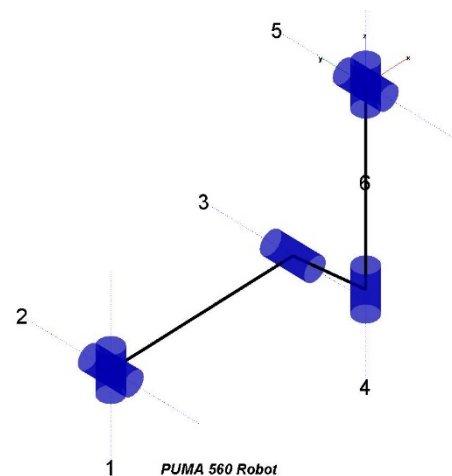
2. <u>Wrist Singularity</u>: In wrist partitioned 6 DOF manipulators, wrist singularities arise because of the motion of wrist cause by the last three joints of the manipulator. When two of the three rotational joints of the wrist become collinear, their equal and opposite rotation about their individual axis cancels out any possible change in orientation of the end-effector [5]. These types of singularities can only be excluded from the joint space by imposing restrictions on the joint variables. Wrist singularities can be identified by analyzing the J_{22} subset of the Jacobian matrix for a manipulator. A wrist singularity can be mathematically represented using Equation 74.

$$|J_{22}| = 0 \tag{74}$$

For example, for a PUMA 560 robot in Figure 30, a wrist singularity is observed at $\theta_5 = 0$ or *pi radians*, where the axis of the fourth and the sixth joint become collinear. A wrist singularity is challenging to visually analyze, since an orientation at a specific point in the Cartesian workspace of a manipulator may be realizable in multiple wrist configurations. It is possible that only a few of those wrist configuration(s) are singular.
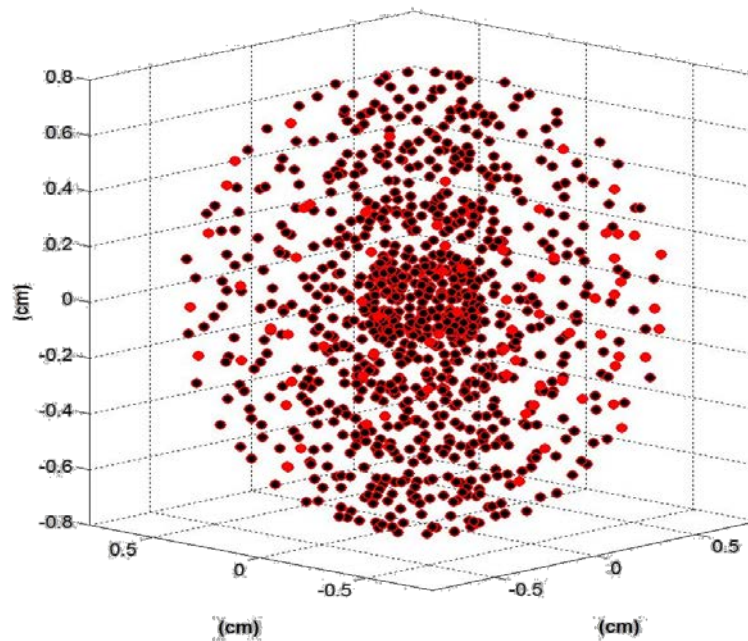
Figure 31: PUMA Wrist Singularity [66]

The corresponding position of the end-effector, however would still be represented as being singular. Figures 31 and 32, show all points in the workspace (black in color) of the PUMA 560 robot as singular points (red in color) since θ_5 = 0 is realizable at every point in the robot's workspace.
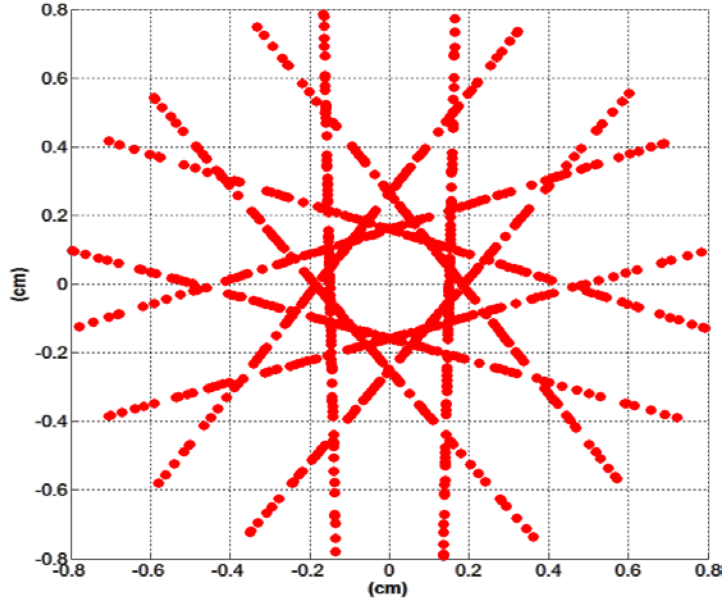
Figure 32: PUMA Wrist Singularity (Top View) [66]

The decoupling of singularities for wrist partitioned manipulators can therefore reduce the computational time and effort in calculating singular configurations. For a manipulator with $J_{12} = 0$, the Jacobian matrix and its singularity condition can thus be represented using Equations 75 and 76, respectively.

$$J(q)_B = J_W = \begin{bmatrix} J_{11} & 0_{3X3} \\ J_{21} & J_{22} \end{bmatrix} \tag{75}$$

$$|J(q)_B| = |J_{11}||J_{22}| \tag{76}$$

8.2 Singularity Free Geometric Path Planning

Geometric path planning is the task of defining a set of Cartesian co-ordinates that define the end-effector's path between two known coordinates in a manipulator's workspace. Path planning is an important part in intelligent control of manipulators, and involves generating an optimized and collision free path through the manipulators'

workspace [71]. Path planning for industrial manipulators can be categorized in three different categories, namely:

1. Point-to-Point (P2P) Path: Point-to-Point path planning involves generating a path between two discrete points within the manipulator's workspace. The path generation using this method may vary for any spatial location of the initial and ending point.

2. Controlled Path: Controlled path planning involves a manipulator's end-effector following a predictable or controlled path through its workspace. The coordinates of the path are pre-determined based on the manipulator's task.

3. Continuous Path: Continuous path planning involves storing a close succession of spatial points in the controller's memory from any teaching sequence. The path defined in the teaching sequence is then replayed from the memory for a defined task. [54]

Singularities are inherent to any manipulator's geometry and design. Development of a singularity free geometric path for an end-effector is important for robust manipulator control. P2P path planning is often challenging since a path generated might involve maneuvering a manipulator's end-effector through singularity zone(s). Singularities can truly be eliminated from a manipulator's workspace by imposing restriction on the range of motion of its joint variables (in joint space). One solution to the problem of path planning thus involves defining a path around the singularity zone(s). A path around any singularity zone may involve:

1. Avoiding a singular point in the manipulator's workspace completely. For example, a non-singular point, P_1, is chosen over a singular point P_0 in a path being defined.

2. Maneuvering the end-effector through a singular point in a non-singular joint configuration. For example, if a point, P_2, is singular in joint configuration q_a, but not in joint configuration q_b, then configuration q_b is selected while maneuvering the end-effector through point P_2.

ANNs are presented here as a non-conventional technique to aid in a singularity free end-effector path generation. An ANN is previously trained for development of an inverse kinematic solution for a specific manipulator configuration (Section 6.8). A data set in Cartesian space consisting of known singularity points is then simulated over the trained network for outputs. The output(s) from the ANN model are compared to the known joint variable configuration(s) for singular points in the manipulator's workspace. The comparison helps visually identify a singularity error window which can be developed in joint space of a manipulator for avoiding singularities.

For example, for the SCARA (RRT) robot, a set of known 625 singularity points (each point with 3 position variables and 9 orientation variables) is normalized between [-1,1]. This normalized dataset is simulated over the inverse kinematic model for the SCARA robot. The ANN output, predicted joint variables, are reverted to scale and compared for error with the theoretical known joint variables of each of the 625 singularity points. Table 4 below shows the absolute error between the predicted joint variable values and the theoretical joint variable values for 5 sample points.

Table 4: Theoretical vs. Predicted Joint Variable Error

Sample No.	Joint Variable (Known)			Joint Variable (Predicted)			Absolute Error		
	q1 (rad)	q2 (rad)	q3 (mm)	q1 (rad)	q2 (rad)	q3 (mm)	E(q1) (rad)	E(q2) (rad)	E(q3) (mm)
1	-2.22	0.00	-280.33	-2.22	0.00	-280.34	0.00	0.00	0.00
2	-1.66	0.00	-205.33	-1.66	0.00	-205.33	0.00	0.00	0.00
3	1.48	0.00	-272.00	1.48	-0.01	-272.00	0.00	0.01	0.00
4	1.11	0.00	-163.67	1.11	0.00	-163.67	0.00	0.00	0.00
5	0.74	0.00	-230.33	0.74	0.00	-230.34	0.00	0.00	0.00

Table 5: Max. and Min. Error in Joint Variable Prediction

Absolute Error	Predicted vs. Theoretical Joint Variables		
	q1 (rad)	q2 (rad)	q3 (mm)
Maximum	0.00	0.01	0.00
Minimum	0.00	0.00	0.00

An absolute error value is chosen since the joint variables assume the value 0 at some points. A relative percentage error for such a point would not be possible (error would be infinite). The maximum and minimum error in all joint variables for all 625 points are presented in Table 5 above.

The theoretical (red) and predicted (blue) joint variables are mapped to their respective Cartesian space in Figure 33 below. It can be observed that that the predicted singularity is well superimposed over the theoretical singularity with barely any error. Moreover, there is very minimalistic deviation from the outer boundary of the workspace where the manipulator singularity exists. The predicted singularity is exactly able to map the radial distance of the theoretical singularity, thereby confirming a robust and well trained ANN.

A comparison of each of the predicted vs. theoretical joint variable values for singularity is presented in Figure 44 (Appendix A). Minimal deviation is observed between the predicted joint values and the theoretical values at the extremes of the joint angle range as seen from the Joint Variable 1 graph. This error arises because of the lowered generalization capability of a high DOF ANN model. A majority of the deviation from theoretical values in Joint Variable 2 is observed between [-0.01, 0.02] *radians*. This confirms the ability of the ANN to accurately predict the singularity condition for a manipulator. The predicted value for Joint Variable 3 is very accurately mapped since it is a translational joint and does not control the orientation of an end-effector. This reduces the variables in Cartesian space needed to be mapped to the joint space, thereby increasing ANN model accuracy.
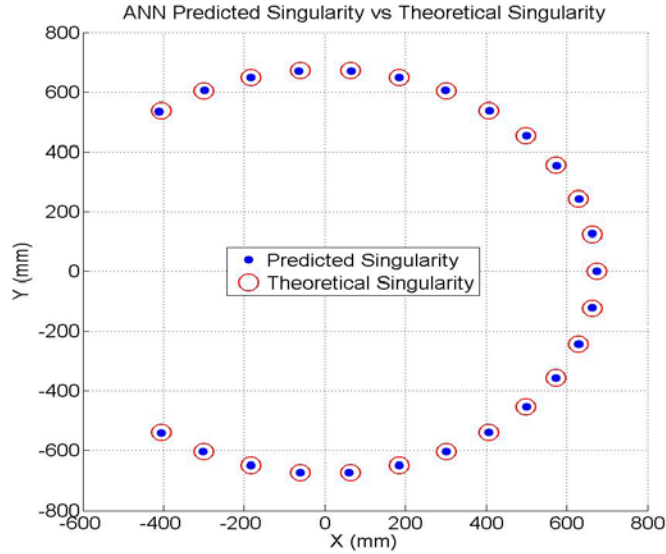
Figure 33: Predicted vs. Theoretical Singularity (Top View)

An error window for each joint variable can therefore be formulated in joint space when planning the path of an end-effector in Cartesian space using the developed ANN model. The error window is determined by adding and subtracting the absolute maximum error from the joint variable values of its respective class. Using this technique, a boundary to the joint variable values contributing to the kinematic singularity in the manipulator workspace can be determined. For example, for the SCARA manipulator, the error window for Sample 1(Table 4) is defined in Table 6 below:

Table 6: Error Window for Path Planning

Sample 1	q_1 (rad)	Error Window	q_2 (rad)	Error Window	q_3 (mm)	Error Window
Upper Limit	-2.22 ± 0.00	⬆ -2.22	0 ± 0.01	⬆ 0.01	-280.33 ± 0.00	⬆ -280.33
Lower Limit		⬇ -2.22		⬇ -0.01		⬇ -280.33

The end-effector path can therefore be planned around these error windows. For example, when the q_2 approaches a value close to a range [-0.01, 0.01] *rad*, an alternate path is taken by the second joint to avoid the oncoming singularity configuration. This technique is especially beneficial when prior singularity conditions for a manipulator are unknown.

CHAPTER 9

RECONFIGURABLE MODEL

The purpose of this research is the development of a reconfigurable tool for modelling of industrial manipulators that can adapt to changes from user based inputs. The mechanical structure along with the joint configurations decides the functionality of any industrial manipulator. Functionality of a manipulator incorporates:

1. Dexterity: Dexterity of a manipulator is its ability to perform a range of tasks in different ways.
2. Flexibility: Flexibility of a manipulator is its generalized ability to adapt to planned or anticipated tasks.
3. Reconfigurability: Reconfigurability is the ability of a manipulator to alter its modules and configuration for a specific task.

The mathematical model (Appendix D) developed for this research can be reconfigured and tailored to accommodate various kinematic structures. The ability of the model to compute various parameters based upon change in structure and configuration allows the user to evaluate different functional aspects of any manipulator type. Various manipulator designs, including the ones that are unexplored, can therefore be studied, evaluated, and optimized with the use of this model. The MATLAB platform is used to code the mathematical model. MATLAB was chosen because of its user-friendly interface, ease of data analysis, and availability of a Neural Network Toolbox for ANN computations. The mathematical model is currently built for up to six joint (6 DOF) industrial manipulator types. It can however, be expanded with ease to model higher DOF. The mathematical model, presented in Figure 34 below, requires the following in inputs:

1. Joint Type: The model requires the user to specify each joint as rotational or translational in their order of sequence. The type and sequence of inputs can be altered by the user depending on the configuration of the manipulator needed.

2. <u>D-H Parameters:</u> The mathematical model requires the user to input all D-H parameters required to model the manipulator configuration of interest, as well as the range of motion for all joint variables.
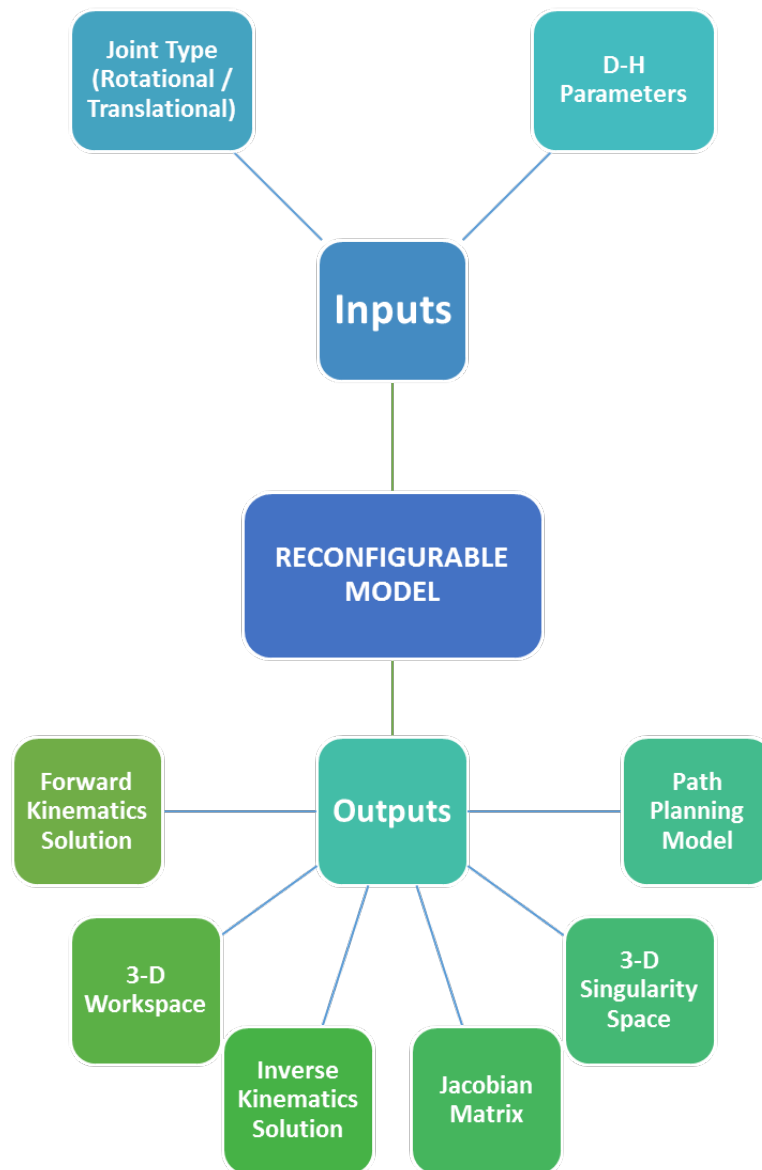
Figure 34: Reconfigurable Model

Based on the user inputs, the model successfully computes and evaluates the following:

1. <u>Forward Kinematics Solution</u>: The model first computes all individual homogenous transformation matrices, 'A(i-1)(i)', for a manipulator. The transformation matrices are subsequently used to develop a forward kinematics homogeneous matrix 'A0(n)'. This matrix is stored in symbolic form which allows manipulation of its position and orientation matrix equation(s) at a later stage. The user can input any joint configuration set value at this stage to obtain a forward kinematics solution.

2. <u>3-D Workspace</u>: The model starts by splitting the range of each joint variable into a set of values defined by an interval called 'steps'. For example, if the value for steps is 3, each joint variable will have 3 joint values. The values in a range are randomized to prevent formation of classes in a continuous dataset. Dividing data into classes will have a much lower accuracy since the ANN may generalize output data to average the classes. Additionally, it also prevents ANN training at the same orientation of the end-effector. The model forms the manipulator's joint space by making all possible combinations of each joint variable. For example, for a 6 DOF manipulator, if each joint assumes 3 values in its range, the total combinations of 6 joints will be 729 (3^6). Each of the joint angle set in the joint space of the manipulator is mapped to its Cartesian space, and the position and rotation matrices are determined. The values from these matrices defines the 3-D position and orientation of the end-effector. The position of each point is plotted to obtain the complete workspace of the manipulator. For example, 729 joint configurations would provide 729 Cartesian coordinates that are represented as the complete workspace. During 3-D plotting of the workspace, the model eliminates all similar points based on their (X, Y, Z) values. This is done to prevent model memory from overloading if the number of points that define the workspace are large.

3. <u>Inverse Kinematics Solution</u>: The position and orientation of each point in the manipulator workspace defines the inputs for the inverse kinematics ANN model. The corresponding joint space of the network inputs defines the network targets. All inputs and targets are pre-processed by being normalized using either min-

max or z-score normalization before being fed to the ANN. The ANN model architecture used for this research has 55 fixed neurons in its hidden layer with hyperbolic tangent activation function since this configuration provides an optimal model generalization and accuracy. The network is trained on predefined parameters after which the network's performance indicators and plots are generated. The outputs from the ANN are stored and reverted to scale. Absolute error is defined at this stage between the network outputs and the targets. The error plots for each joint variable are generated to give the user an understanding of variation in prediction of each joint variable.

4. Jacobian Matrix: The reconfigurable model defines all angular and linear velocity vectors for each joint variable of the manipulator. Newton Euler Recursive Method calculations are subsequently carried out to determine the Jacobian of linear and angular velocity elements for the end-effector with respect to the base frame of the manipulator. If the manipulator is wrist-partitioned, sub-matrices J_{11} and J_{22} are determined from the Jacobian matrix for decoupling of forearm and wrist joints respectively.

5. 3-D Singularity Space: The model computes all kinematic singularity conditions present in the manipulator configuration by analyzing its Jacobian. The joint variable combinations that produce singularities are subsequently identified. The joint space of the manipulator is modified with the newly determined joint variable combinations that produce these singularities. The new joint space is mapped to its corresponding Cartesian space using the manipulator's forward kinematics equations. The new position and orientation matrices developed help visually identify the loci of kinematic singularities present in the manipulator workspace. All singular points are identified in the color red.

6. Path Planning Model: Once singular points are visually identified, the position and orientation of each singular point is normalized and simulated over the previously developed inverse kinematics model. The simulation results are compared to the joint variable combinations (targets) previously determined while developing the singularity space. The absolute error between network output and target for each joint variable helps determine an error window around each

singular configuration in joint space. During path planning, this model can be effectively used to avoid singular conditions. A boundary to the loci of singular points is determined using the error window which can help refrain the end-effector from accessing certain part(s) of the manipulator's workspace in specific joint configurations. The error plots for each joint variable are generated to give the user an understanding of variation in prediction of each joint variable that causes singularity.

The complete workspace and singularity space models of the manipulator are developed using a step size of 10. The reconfigurable model thus determines 1 million (10^6) joint configurations and their respective position and orientation matrices. Such large amounts of data (18 million variables) cannot be processed through Neural Network Toolbox for MATLAB because of computational constraints. A smaller step size is therefore chosen for all neural network models. In determining the amount of data to be processed for the inverse kinematics and path planning model, various step sizes such as 3, 4, 5 etc. were experimented with. A step size of 3 provided the most accurate ANN model results over any other step size chosen. The accuracy with a smaller step sized increased because of the reduced level of complexity in the dataset. A step size of 3 was therefore selected as the default step size for developing ANN models. A drawback to a smaller step size is the need for defining classes of inverse kinematics solution(s) for any manipulator configuration. For example, a step size of 3 will only provide 729 points in the manipulator workspace that can be used as inputs to an ANN model. Thus an inverse kinematics solution that caters only to a specific subset (729 points) of the total workspace can be determined at any given point. New joint variable values are thus needed to define another inverse kinematics solution for a different subset of the workspace and so forth. The complete inverse kinematics model for a manipulator is determined by unifying individual classes of solutions developed. It is important to note that the task space for a manipulator may only involve certain paths(s) of actual mechanical work. For example, a welding robot may only be required to weld along a curvilinear path defined by a string of points. It is therefore justified to develop inverse kinematic model(s) that can encompass certain required points of work.

CHAPTER 10

CASE STUDIES AND RESULTS

For the purpose of this research, the robustness of the developed model is tested on two different kinematic structures namely:

1. 6 DOF Industrial Robot: A FANUC M16iB/20 robot is chosen for this study since the kinematic structure (RRRRRR) of this robot has a wrist configuration in its last 3 joints. Wrist partitioned robots are the most common types of manipulators used in the industry today. FANUC M16iB/20 (Figure 35) is a popular industrial manipulator used for several material handling applications.

2. 6 Axis CNC Machine: A multi-axis CNC was chosen for this study for two purposes. Firstly, to test the robustness of the developed algorithm when analyzing a kinematic structure with a combination of both rotational and translational joint types. Secondly, to show the wide range of applications of the developed model. The reconfigurable model is able to analyze any 6 axis machine structure that can be kinematically modelled such as the 6 Axis CNC (RRRTTT) (Figure 36).

Figure 35: FANUC M16iB/20 Robot [72]   Figure 36: 6 Axis CNC Machine
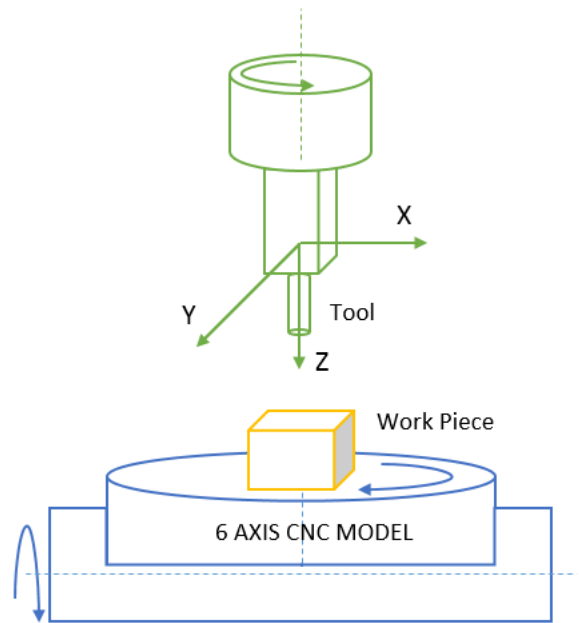
10.1 6 DOF Industrial Robot: FANUC M16iB/20

Note: All results for the FANUC M16iB robot are presented in Appendix B.

A kinematic model of the FANUC M16iB/20 robot, provided in Figure 46 (Appendix B), is first generated to analyze the configuration of the manipulator. Table 16 represents the D-H parameters used to model the manipulator along with the range of motion for each rotational joint. To generate the manipulator's total workspace and to compute its corresponding singularity space, a step size of 10 was chosen that yielded 10^6 joint configurations. Each of these configurations when processed through the forward kinematics equation, A06 (Appendix B MATLAB Output), yielded the same number of configurations in Cartesian space. The joint angle range for each joint with a step size of 10 is represented in Table 17. Out of the 1 million points generated, it was observed that the Cartesian space had only 100,000 unique points based on their (X, Y, Z) coordinates. This implies that the model generated 10 orientation configurations per coordinate point in the manipulator's workspace.

The complete 3-D workspace of the FANUC manipulator has a spherical topology and is represented in Figure 47. From the top view of the total workspace (Figure 48), a cylindrical void is observed exactly in the middle of the spherical workspace. This void area is inaccessible by the end-effector of the FANUC manipulator. From the top, front and, right view, it is observed that the total workspace fans out from a center point with the number of spokes equal to the steps used to build the workspace. This implies that all possible combinations of each set of joint variables produce a subset spoke of the manipulator workspace. If the step size were increased, the workspace would not demonstrate any voids between its spokes but would still have a void in the center.

Since the robot is wrist partitioned, both subsets, J_{11} and J_{22}, are analyzed for forearm and wrist singularities, respectively, as seen from Appendix B (MATLAB Outputs). Kinematic singularity condition for the FANUC M16iB robot is only observed at the manipulator wrist and is represented in Equation 77.

$$Singularity\ Condition: \theta_5 = 0 \tag{77}$$

Figure 49 represents the total workspace and singularity space for the FANUC manipulator. At first look, minimal singularity space (red coloured points) is observed since only singularities at workspace boundaries are visible. From Figure 50, which represents the total singularity space, it is observed that the majority of the kinematic singularity (internal) is present within the manipulator workspace. For computation of an inverse kinematics solution, a random subset of joint configurations with a step size of 3 is chosen from the joint space of the manipulator. Table 18 represents the joint angles values used for training the ANN. The model reruns on the new joint configurations and first develops a subset of the total workspace and singularity space as represented in Figure 51. The Cartesian space configuration of this subset workspace is normalized and provided to the network as inputs. The joint angle configurations are normalized and provided as targets. It can be seen from Figure 52 that an inverse kinematics solution for the robot being studied is computed in merely 6 seconds and 23 epoch runs. The error histogram from Figure 53 shows the concentration of errors from the trained network at a fairly low value of 0.0205. The error histogram demonstrates a good normalization curve with majority errors between the ranges of ±0.4. The regression plot from Figure 54, shows an overall R value of 98.68% thereby indicating a well-trained network. Best validation performance for this network was reached at epoch 17 as seen from Figure 55. The validation fail check was reached at epoch 23 as seen from Figure 56. Here, the network gradient and learning rate (mu) curve for the network can also be observed for each epoch run. The inputs and bias to the hidden and output layer are provided in Appendix B. A summary of the ANN Inverse Kinematic results are provided in Table 7.

Table 7: ANN Results for FANUC M16iB/20

S.No.	ANN Network Indicator	Result
1	Total Epochs	23
2	Epoch for Best Validation Performance	17
3	Overall Regression (R) Value	0.9868
4	Mean Square Error (MSE)	0.0198
5	Training Performance	0.0105
6	Testing Performance	0.0638
7	Validation Performance	0.0505
8	Error Histogram Center (Bell Curve)	0.0205

A comparison between the network outputs and targets (joint configurations) is presented in Figure 57. The network is very accurate in predicting the first 3 joints of the manipulator. It is observed that the predicted outputs of the network almost superimpose on the target values. However, some variation is observed in Joints 4, 5, and 6 which form the wrist of the manipulator. This variation arises due to the generalization properties of the developed ANN. The purpose of an ANN is to determine a generalized trend between the input and output parameters of a given manipulator, rather than mapping exact points which leads to an over-fitted model. It is important to realize that multiple wrist configurations may exist for every given set of position coordinates (X,Y,Z) in the input data set. These wrist configurations primarily contribute to the orientation of the manipulator's end-effector. For each set of unique position coordinates, the wrist can therefore assume a specific set of joint configurations. As a result, during the training phase, the ANN network attempts to predict a generalized model for Joints 4, 5, and 6 for these multiple wrist configurations. Hence, when a new input set of parameters is introduced to the network, the network attempts to predict an overall generalized result for the last three joints based on their average thereby reducing network accuracy. One method to increase the network accuracy is to generate an input dataset that has only one orientation associated with a unique coordinate point. This will map one single point in Cartesian space to only one combination of joint value set thereby increasing the network accuracy. Figure 58 represents a plot of absolute residual errors between aforementioned networks outputs and targets due to the network generalization.

To develop a path planning model, the singularity points from the subset Cartesian space are simulated over the trained network to provide predicted joint angle configurations for singularity. Figure 59 represents a comparison between the ANN predicted and theoretical joint configurations. It is observed that there is minimalistic error between the predicted and theoretical values for the first 3 joints. Although, there is noticeable error in joint prediction for the last three joints. This error can be ignored because the path of a manipulator can be determined irrespective of the orientation (controlled by wrist joints) of its end-effector. High network accuracy is achieved for the first three joints that are responsible for controlling the position of the end-effector in

wrist partitioned robots. This can be seen from Figure 60 which represents the absolute residual errors between the predicted and theoretical joint configurations.   The predicted joint variables are mapped to their corresponding Cartesian space and compared with the singularity values as represented in Figure 61. It is observed that the predicted singularity of the model is fairly accurate when compared to the theoretical singularity. This validates the robustness of the path planning model as well as the robustness of the developed inverse kinematic model using ANNs. The absolute errors in joint space between ANN predicted and theoretical joint angle configurations are presented in Table 8. Table 10 represents the maximum and minimum errors in joint prediction which help define an error window (Table 9) to aid in path planning.

Table 8: Sample Theoretical vs. Predicted Joint Variable Error

Sample No.	Joint Variable (Known)			Joint Variable (Predicted)			Absolute Error		
	q_1 (rad)	q_2 (rad)	q_3 (rad)	q_1 (rad)	q_2 (rad)	q_3 (rad)	$E(q_1)$ (rad)	$E(q_2)$ (rad)	$E(q_3)$ (rad)
1	1.65	1.80	3.37	1.64	1.74	3.35	0.00	0.06	0.02
2	1.44	-0.09	0.52	1.45	-0.12	0.57	0.02	0.03	0.05
3	0.87	-0.58	0.52	0.86	-0.50	0.46	0.01	0.07	0.06

Sample No.	Joint Variable (Known)			Joint Variable (Predicted)			Absolute Error		
	q_4 (rad)	q_5 (rad)	q_6 (rad)	q_4 (rad)	q_5 (rad)	q_6 (rad)	$E(q_4)$ (rad)	$E(q_5)$ (rad)	$E(q_6)$ (rad)
1	-2.32	0.00	3.09	-1.82	-0.24	-0.12	0.50	0.24	3.21
2	-3.09	0.00	-2.28	-2.49	0.20	-0.08	0.60	0.20	2.20
3	-2.32	0.00	1.65	-2.38	-0.03	1.61	0.06	0.03	0.04

Table 9: Sample Error Window for Path Planning

Sample 1	q₁ (rad)	Error Window	q₂ (rad)	Error Window	q₃ (rad)	Error Window
Upper Limit	$1.65 \pm$ 0.12	⬆ 1.77	1.80 ± 0.14	⬆ 1.94	3.37 ± 0.17	⬆ 3.54
Lower Limit		⬇ 1.53		⬇ 1.66		⬇ 3.20

Sample 1	q₄ (rad)	Error Window	q₅ (rad)	Error Window	q₆ (rad)	Error Window
Upper Limit	$-2.32 \pm$ 2.2	⬆ -0.12	0 ± 1.50	⬆ 1.50	3.09 ± 4.40	⬆ 7.49
Lower Limit		⬇ -4.52		⬇ -1.50		⬇ -1.31

Table 10: Max. and Min. Error in Joint Variable Prediction

Absolute Error	Predicted vs. Theoretical Joint Variables					
	q₁ (rad)	q₂ (rad)	q₃ (rad)	q₄ (rad)	q₅ (rad)	q₆ (rad)
Maximum	0.12	0.14	0.17	2.20	1.50	4.40
Minimum	0.00	0.00	0.00	0.02	0.00	0.08
Average	0.022	0.058	0.056	0.56	0.46	1.5

10.2 6 Axis CNC Machine

Note: All results for the 6 Axis CNC Machine are presented in Appendix C.

A 6 Axis CNC machine with a rotary table and an X,Y,Z axis tool with a rotating axis is chosen for this study. A common example of such a CNC machinery is the high speed precision milling CNC machines used in the industry today. A kinematic model of the CNC machine, provided in Figure 62 (Appendix C), is first generated to analyze and accurately model the configuration of the machine. The tool of the machine is considered as the end-effector of a manipulator, the tool axes of motion are represented by 3 translational joints and a rotational joint. The rotary table of the CNC machine is represented by 2 rotational joints. The individual components are clubbed and modelled as an open kinematic chain with 6 DOF (RRRTTT). The developed kinematic chain (CNC manipulator) emulates the behaviour of the CNC machine with respect to its function.

Table 19 represents the D-H parameters used to model the CNC manipulator along with the range of motion for each joint variable. Similar to the previous case study, a step size of 10 was chosen to generate the manipulator's total workspace and to compute its corresponding singularity space which yielded 10^6 joint configurations. Each of these configurations when processed through the forward kinematics equation, A06 (Appendix C MATLAB Outputs), yielded the same number of configurations in Cartesian space. The joint variable range for each joint with a step size of 10 is represented in Table 20. Out of the 1 million points generated, it was observed that there were no repeated points based on the X, Y, Z coordinate values, and therefore the Cartesian space consisted of a set of unique 1 million configurations.

The complete 3-D workspace of the CNC manipulator, represented in Figure 63, has a topology of a spirally coiled gastropod shell flattened at one end. From the top and front view of the total workspace (Figure 64), a void towards the center as well as the flattened end of the workspace can be seen. This void area is inaccessible by the end-effector of the CNC manipulator.

Since the CNC manipulator does not have wrist configuration, the Jacobian in the base frame is analyzed for any kinematic singularities that may be present as seen from Appendix C (MATLAB Outputs). A kinematic singularity condition for the CNC manipulator is only observed when the second joint assumes a specific value thereby cancelling the effect of the first and third joints on one another. The singularity condition is represented in Equation 78.

$$Singularity\ Condition: \theta_2 = 0 \tag{78}$$

Figure 65 represents the total workspace and singularity space for the CNC manipulator. From this figure, the singularity space for the manipulator is only observed as a single coiled path (red colour) at the boundary of the workspace. From Figure 66, which represents the total singularity space, it is observed that internal singularities are also present in the manipulator workspace. The total singularity space for the CNC manipulator is therefore a planar subsection of the total workspace that extends along the z-axis. Analysis of such visual representations of the singularity zone(s) is useful in

evaluating and enhancing manipulator functionality and performance. The task space manipulators can therefore be planned for by taking into account the singularity space and not just the total workspace of the manipulator.

For computation of an inverse kinematics solution, a random subset of joint configurations with a step size of 3 is chosen from the joint space of the manipulator. Table 21 represents the joint angle values used for training the ANN. The model reevaluates on the newly provided joint configurations and develops a subset of the total workspace and singularity space as represented in Figure 67. The Cartesian space configuration of this subset workspace is normalized and provided to the network as inputs. The joint angle configurations are normalized and provided as targets. It can be seen from Figure 68 that it takes only about a minute and a half and 316 epochs to develop an inverse kinematics solution for the CNC manipulator. The error histogram from Figure 69 shows the concentration of errors from the trained network nearly at zero thereby representing a well-trained network. The error histogram demonstrates an excellent normally distributed curve with the majority of errors in the range of ±0.006. The regression plot from Figure 70, shows an overall R value of 99.99% thereby indicating that the network outputs perfectly fit to the supplied targets. The network training was prematurely stopped at epoch 316 where the best validation performance for this network was reached, as seen in Figure 71. Early stoppage was executed since the performance of the network had reached nearly zero. Therefore, no validation fail checks were performed as seen from Figure 72. From the same figure we observe that the network gradient and learning rate (mu) had reached a fairly low value indicating a satisfactory training process. The inputs and bias values to the hidden and output layer are provided in Appendix C (MATLAB Outputs). A summary of the ANN Inverse Kinematic results are provided in Table 11 below. The network has a high accuracy since each input point was mapped to a unique combination of joint set configurations. A comparison between the network outputs and targets (joint configurations) is presented in Figure 73. Because of the high accuracy of the trained network, the outputs are completely superimposed onto the supplied targets for joint variables. The network performance is validated from the residual error plot presented in Figure 74 which represents minimalistic absolute residual errors between the networks outputs and targets.

Table 11: ANN Results for 6 Axis CNC Machine

S.No.	ANN Network Indicator	Result
1	Total Epochs	316
2	Epoch for Best Validation Performance	316
3	Overall Regression (R) Value	0.99999
4	Mean Square Error (MSE)	0.000011
5	Training Performance	0.000007
6	Testing Performance	0.000029
7	Validation Performance	0.000029
8	Error Histogram Center (Bell Curve)	0.000218

To develop a path planning model, the singularity points from the subset Cartesian space are simulated over the trained network to provide predicted joint variable configurations for singularity. Figure 75 represents a comparison between the ANN predicted and theoretical joint configurations. Since very minimal variation is observed, it can be concluded that the network is able to predict the singularity configurations very accurately. Figure 76 represents an absolute residual errors plot between the predicted and theoretical joint configurations. The predicted joint variables are mapped to their corresponding Cartesian space and compared with the singularity points as presented in Figure 77. It is observed that the predicted singularity of the model completely superimposes the theoretical singularity. This validates the robustness of the path planning model as well as the robustness of the developed inverse kinematic mode using ANNs. The absolute errors in joint space between the ANN predicted and theoretical joint variable configurations are presented in Table 12. Table 14 represents the maximum and minimum errors in joint prediction which help define an error window (Table 13) to aid in path planning.

Table 12: Sample Theoretical vs. Predicted Joint Variable Error

Sample No.	Joint Variable (Known)			Joint Variable (Predicted)			Absolute Error		
	q_1 (rad)	q_2 (rad)	q_3 (rad)	q_1 (rad)	q_2 (rad)	q_3 (rad)	$E(q_1)$ (rad)	$E(q_2)$ (rad)	$E(q_3)$ (rad)
1	0.70	0.00	1.56	0.70	-0.02	1.56	0.00	0.02	0.00
2	0.48	0.00	-1.03	0.48	-0.01	-1.02	0.00	0.01	0.01
3	-0.92	0.00	0.14	-0.94	-0.01	0.15	0.02	0.01	0.01

Sample No.	Joint Variable (Known)			Joint Variable (Predicted)			Absolute Error		
	q_4 (m)	q_5 (m)	q_6 (m)	q_4 (m)	q_5 (m)	q_6 (m)	$E(q_4)$ (m)	$E(q_5)$ (m)	$E(q_6)$ (m)
1	0.00	0.00	0.39	-0.02	0.00	0.37	0.02	0.00	0.03
2	-0.26	-0.18	-0.21	-0.25	-0.19	-0.24	0.01	0.01	0.03
3	-0.13	-0.18	-0.21	-0.14	-0.17	-0.22	0.01	0.01	0.01

Table 13: Error Window for Path Planning

Sample 1	q_1 (rad)	Error Window	q_2 (rad)	Error Window	q_3 (rad)	Error Window
Upper Limit	0.7 ± 0.03	⬆ 0.73	0 ± 0.01	⬆ 0.01	1.56 ± 0.04	⬆ 1.60
Lower Limit		⬇ 0.67		⬇ -0.01		⬇ 1.52

Sample 1	q_4 (m)	Error Window	q_5 (m)	Error Window	q_6 (m)	Error Window
Upper Limit	0 ± 0.03	⬆ 0.03	0 ± 0.01	⬆ 0.01	0.39 ± 0.02	⬆ 0.41
Lower Limit		⬇ -0.03		⬇ -0.01		⬇ 0.37

Table 14: Max. and Min. Error in Joint Variable Prediction

Absolute Error	Predicted vs. Theoretical Joint Variables					
	q_1 (rad)	q_2 (rad)	q_3 (rad)	q_4 (m)	q_5 (m)	q_6 (m)
Maximum	0.03	0.01	0.04	0.03	0.01	0.02
Minimum	0.00	0.01	0.00	0.00	0.00	0.00
Average	0.007	0.014	0.006	0.018	0.006	0.015

10.3 Reconfigurable Model Applications

From the two case studies presented, the robustness of the developed model can be validated. The reconfigurable model can be used to analyze and validate the performance criterion for a wide range of industrial manipulators as well as non-conventional machinery structures that can be parameterized in a similar fashion to kinematic manipulators. Unlike other software that can only cater to standard manipulator configurations, the developed model can reconfigure to any manipulator configuration based on user inputs and generate results accordingly.

The model can be used as a design tool for development of kinematic structures based on pre-defined functional requirements and for downstream optimization problems. It also serves as an excellent tool for workspace and singularity analysis by not only theoretically computing the functional workspace of the model but by also providing a 3-D visual understanding of the manipulator reach and functionality. The model is also successfully able to provide a non-conventional and computationally inexpensive solution to the problem of inverse kinematics by using ANNs. This technique is highly beneficial in developing a path planning and collision detection model. The proposed method can also successfully aid in development of robotic work cells where it is crucial to understand the reach conditions of robot(s) with respect to their environment [66].

CONCLUSIONS AND FUTURE WORK

A reconfigurable model is developed to gain an insight into the functionality of industrial manipulators and optimization of their performance. The developed reconfigurable model is successfully able to provide a forward kinematics solution, an inverse kinematic solution, a 3D visual representation of workspace and kinematic singularity, an analysis of the manipulator Jacobian, and a model to aid in path planning of robots. The model provides promising results for both wrist and non-wrist partitioned manipulators as well other machinery structures such as CNC machines that can be modelled kinematically. This model can be successfully used for optimizing the placement of industrial manipulators in an industrial setting and understanding their reach conditions based on an analysis of their functional workspace.

This research lays the foundation for the development of a reconfigurable model that can adapt to various manipulator configurations and provide the aforementioned analytical tools. Future work for expanding the scope of analyses incudes:

1. Modelling of higher DOF redundant robots and machine structures
2. Expanding on the type of manipulator joints to be modelled
3. Developing dynamic equations of motion for a manipulator by expanding on the Newton-Euler Recursive method
4. Incorporating simultaneous analysis of several kinematic chains and optimizing their placement with respect to one another within the same work cell
5. Developing a trajectory planning and collision detection model
6. Incorporating a wider range of joint variable ranges for training the ANN model
7. Expanding on the ANN model architecture for an improved accuracy in prediction of wrist configurations

The developed tool will aid to further research in the field of industrial robotics. It will also help robot designers, manufacturers, as well as end-users to understand the true functionality and capabilities of any manipulator. The research can ultimately be extended to incorporate complex robot structures such as parallel link manipulators etc.

REFERENCES

[1]   Adept Technology Inc., "Six-Axis Robot Configuration Singularities, Use of the V+ MV.SL_MOVE Routine and the SPEED.LIMIT Parameter," Adept Tehnology Inc., United States of America, 2007.

[2]   L. Aggarwal, R. J. Urbanic and K. Aggarwal, "A Reconfigurable Algorithm for Identifying and Validating Functional Workspace of Industrial Manipulators," in *SAE World Congress 2014,* Detroit, 2014.

[3]   M. Tyson, "Robotics 101," ABB, May 2013. [Online]. Available: http://www02.abb.com/global/zaabb/zaabb011.nsf/bf177942f19f4a98c1257148003 b7a0a/0f55ae5d8012b4a6c1257b7200358d2e/$file/dmro+1+-+introduction+to+robotics.pdf. [Accessed 11 May 2014].

[4]   A. G. Gudla, *A methodology to determine the functional workspace of a 6R robot using forward kinematics and geometrical methods,* Windsor: University of Windsor, 2012.

[5]   M. W. Spong, S. Hutchinson and M. Vidyasagar, Robot Modeling and Control, Hoboken, NJ: John Wiley & Sons Inc., 2006.

[6]   P. I. Corke, "Robotics, Vision & Control: Fundamental Algorithms in MATLAB," Springer, 2011.

[7]   T. Yoshikawa, "Manipulability of Robotic Mechanisms," *The International Journal of Robotics Research,* vol. 4, no. 3, pp. 3-9, 1985.

[8]   A. Y. Elkady, M. Mohammed and T. Sobh, "A New Algorithm for Measuring and Optimizing the Manipulability Index," *Journal of Intelligent and Robotic Systems,* vol. 59, no. 1, pp. 75-86, 2010.

[9]   G. Pamanes and S. Zeghloul, "Optimal placement of robotic manipulators using multiple kinematic criteria," in *IEEE International Conference on Robotics and Automation,* Sacramento, 1991.

[10]   A. M. Djuric, R. A. Saidi and W. ElMaraghy, "Global Kinematic Model Generation for n-DOF Reconfigurable Machinery Structure," in *6th annual IEEE Conference on Automation Science and Engineering,* Toronto, 2010.

[11] L. K. Khosla and P. K., "Automatic generation of forward and inverse kinematics for a reconfigurable modular manipulator system," *Journal of Robotic Systems,* vol. 7, no. 4, pp. 599-619, 1990.

[12] C. J. J. Paredis and P. K. Khosla, "Kinematic Design of Serial Link Manipulators from Task Specifications," *The International Journal of Robotics Research,* vol. 12, no. 3, pp. 274-287, 1993.

[13] A. Djuric and R. J. Urbanic, "Design of a reconfigurable robot-based system for material deposition applications," in *IEEE International Conference on Electro/Information Technology*, Windsor, 2009.

[14] L. H. T. H.-y. K. S. L. X. Hai-Yin Xu, "A novel kinematic model for five-axis machine tools," *The International Journal of Advanced Manufacturing Technology,* vol. 67, no. 5-8, pp. 1297-1307, 2013.

[15] Z. Du, S. Zhang and M. Hong, "Development of a multi-step measuring method for motion accuracy of NC machine toolsbased on cross grid encoder," *International Journal of Machine Tools & Manufacture,* vol. 50, no. 3, pp. 270-280, 2010.

[16] D. M. A. Lee and W. H. ElMaraghy, "ROBOSIM: a CAD-based off-line programming and analysis system for robotic manipulators," *Computer-Aided Engineering Journal,* vol. 7, no. 5, pp. 141-148, 1990.

[17] M. Ceccarelli and A. Vinciguerra, "On the workspace of general 4R manipulators," *International Journal of Robotics Research,* vol. 14, no. 2, pp. 152-160, 1995.

[18] E. Ottaviano, M. Husty and M. Ceccarelli, "A Study on Workspace Topologies of 3R Industrial-Type Manipulators," in *International Conference on Automation, Quality and Testing, Robotics*, 2006.

[19] C. Liang and M. Ceccarelli, "Feasible workspace regions for general two-revolute manipulator," *Frontiers of Mechanical Engineering,* vol. 6, no. 4, pp. 397-408, 2011.

[20] K. Abdel-Malek, H.-J. Yeh and S. Othman, "Interior and exterior boundaries to the workspace of mechanical manipulators," *Robotics and Computer-Integrated Manufacturing,* vol. 16, no. 5, pp. 365-376, 2000.

[21] O. Bohigas, M. Manubens and L. Ros, "A Complete Method for Workspace Boundary Determination on General Structure Manipulators," *IEEE Transactions*

on Robotics, vol. 28, no. 5, pp. 993-1006, 2012.

[22] K. Goyal and D. Sethi, "AN ANALYTICAL METHOD TO FIND WORKSPACE OF A ROBOTIC MANIPULATOR," *Journal of Mechanical Engineering,* vol. 41, no. 1, pp. 25-30, 2010.

[23] A. Djuric, J. Urbanic, M. Filipovic and L. Kevac, "Effective Work Region Visualization for Serial 6 DOF Robots," in *5th International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV)*, Munich, 2013.

[24] R. J. Urbanic and A. Gudla, "Functional Workspace Estimation of a Robot using Forward Kinematics, D-H Parameters, and Shape Analyses," in *ASME 11th Biennial Conference on Engineering Systems Design and Analysis*, Nates, 2012.

[25] A. Djuric and R. J. Urbanic, "Utilizing the Functional Work Space Evaluation Tool for Assessing a System Design and Reconfiguration Alternatives," in *Robotic Systems - Applications, Control and Programming*, A. Dutta, Ed., 2012.

[26] T. K. Alameldin, N. Badler, T. Sobh and R. Mihali, "A Computational Approach for Constructing of the Reachable Workspaces for Redundant Manipulators," *Journal of Scientific Computing,* vol. 2, no. 1, pp. 48-52, 2003.

[27] M. Zein, P. Wenger and D. Chablat, "An exhaustive study of the workspace topologies of all 3R orthogonal manipulators with geometric simplifications," *International Workshop on Computational Kinematics Special Issue,* vol. 41, no. 8, pp. 971-986, 2006.

[28] J. Kim, G. Marani, W. K. Chung and J. Yuh, "Task reconstruction method for real-time singularity avoidance for robotic manipulators," *Advanced Robotics,* vol. 20, no. 4, pp. 453-481, 2006.

[29] H. Liu and T. Zhang, "Browse Conference Publications > Mechatronics and Automation (... Help Working with Abstracts," in *International Conference on Mechatronics and Automation (ICMA)*, Xi'an, 2010.

[30] H. Zhunqing, F. Hairong and F. Yuefa, "New solution algorithm for singularity control of serial manipulators," in *IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, 2002.

[31] Y. Fang and L.-W. Tsai, "Feasible Motion Solutions for Serial Manipulators at Singular Configurations," *Journal of Mechanical Design,* vol. 125, no. 1, pp. 61-

69, 2003.

[32] Z.-Q. Hu, Z.-G. Fu and H.-R. Fang, "Study of singularity robust inverse of Jacobian matrix for manipulator," in *International Conference on Machine Learning and Cybernetics*, 2002.

[33] D. Pai and M. Leu, "Generic singularities of robot manipulators," in *IEEE International Conference on Robotics and Automation*, Scottsdale, 1989.

[34] A. Djuric, M. Filipovic, L. Kevac and J. Urbanic, "Singularity Analysis for a 6 DOF Family of Robots," in *roceedings of the 5th International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV 2013)*, Munich, 2013.

[35] L. Huo and L. Baron, "The joint-limits and singularity avoidance in robotic welding," *Industrial Robot: An International Journal,* vol. 35, no. 5, pp. 456-464, 2008.

[36] M. Stanisic and O. Duta, "Symmetrically actuated double pointing systems: the basis of singularity-free robot wrists," *IEEE Transactions on Robotics and Automation,* vol. 6, no. 5, pp. 562-569, 1990.

[37] F.-T. Cheng, T.-L. Hour, Y.-Y. Sun and T.-H. Chen, "Study and resolution of singularities for a 6-DOF PUMA manipulator," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 1997.

[38] S. Ahmad and S. Luo, "Analysis of kinematic singularities for robot manipulators in Cartesian coordinate parameters," in *IEEE International Conference on Robotics and Automation*, Philadelphia, 1988.

[39] S. Chiaverini and O. Egeland, "A solution to the singularity problem for six-joint manipulators," in *IEEE International Conference on Robotics and Automation*, Cincinnati, 1990.

[40] S. Yigit, C. Burghart and H. Woern, "Avoiding Singularities of Inverse Kinematics for a Redundant Robot Arm for Safe Human Robot Co-operation," in *CCCT*, 2003.

[41] C. Kozakiewicz, T. Ogiso and N. Miyake, "Partitioned neural network architecture for inverse kinematic calculation of a 6 DOF robot manipulator," in *IEEE International Joint Conference on Neural Networks*, 1991.

[42] Y. F. Lou and P. Brunn, "A hybrid artificial neural network inverse kinematic

solution for accurate robot path control," *Journal of System and Control Engineering,* vol. 213, no. 1, pp. 23-32, 1999.

[43] Z. Ahmad and A. Guez, "On the solution to the inverse kinematic problem," in *IEEE International Conference on Robotics and Automation*, Cincinnati, 1990.

[44] Ş. Yıldırım and İ. Eski, "A QP Artificial Neural Network inverse kinematic solution for accurate robot path control," *Journal of Mechanical Science and Technology,* vol. 20, no. 7, pp. 917-928, 2006.

[45] R. Köker, C. Öz, T. Çakar and H. Ekiz, "A study of neural network based inverse kinematics solution for a three-joint robot," *Robotics and Autonomous Systems: Patterns and Autonomous Control,* vol. 49, no. 3-4, pp. 227-234, 2004.

[46] A. T. Hasan, N. Ismaila, A. Hamoudab, A. Ishak, M. M.H. and H. Al-Assadia, "Artificial neural network-based kinematics Jacobian solution for serial manipulator passing through singular configurations," *Advances in Engineering Software,* vol. 41, no. 2, pp. 359-367, 2010.

[47] Z. Bingul, H. Ertunc and C. Oysu, "Comparison of inverse kinematics solutions using neural network for 6R robot manipulator with offset," in *ICSC Congress on Computational Intelligence Methods and Applications*, Istanbul, 2005.

[48] Y. Feng, W. Yao-nan and Y. Yi-min, "Inverse Kinematics Solution for Robot Manipulator based on Neural Network under Joint Subspace," *International Journal of Computers, Communications & Control,* vol. 7, no. 3, pp. 459-472, 2012.

[49] International Organization for Standardization, "Industrial Robotics," International Federation of Robotics (IFR), [Online]. Available: http://www.ifr.org/industrial-robots/. [Accessed May 2014].

[50] M. Hägele, . K. Nilsson and J. N. Pires, "Industrial Robotics," in *Springer Handbook of Robotics*, Springer Berlin Heidelberg, 2008, pp. 963-986.

[51] International Federation of Robotics, "Industrial Robot Statistics," International Federation of Robotics, 2013.

[52] E. Guizzo, "The Rise of the Machines," IEEE Spectrum, December 2008. [Online]. Available: http://spectrum.ieee.org/robotics/industrial-robots/the-rise-of-the-machines.

[53] A. Djuric, "Introduction to Robotics," University of Windsor, Windsor, 2013.

[54] United States Department of Labour, "OSHA Technical Manual (OTM): Section IV: Chapter 4," January 1999. [Online]. Available: https://www.osha.gov/dts/osta/otm/otm_iv/otm_iv_4.html#2.

[55] S. M. LaValle, "Planning Algorithms," Canbridge University Press, 2006. [Online]. Available: http://planning.cs.uiuc.edu/node102.html.

[56] C. Bernier, "Different End Effectors on the Market," Robotiq, July 2013. [Online]. Available: http://blog.robotiq.com/bid/65660/Different-End-Effectors-on-the-Market.

[57] R. Manseur, Robot Modelling and Kinematics, Boston: Da Vinci Engineering Press, 2006.

[58] Mitsubishi Electric, "Mitsubishi Electric Industrial Robot," Jun 2006. [Online]. Available: http://www.abcontrols.com/robotic_arms/mitsubisi_robotic_arms/mitsubishi_scara _robot_abcontrols.pdf.

[59] J. J. Craig, Introduction to Robotics: Mechanics and Control, Upper Saddle River: Pearson Eductaion International, 2005.

[60] P. Allen, "Inverse Kinematics," Columbia University, October 2010. [Online]. Available: http://www.cs.columbia.edu/~allen/F13/NOTES/invkin.pdf.

[61] J. C. Principe, N. R. Euliano and W. C. Lefebvre, Neural and Adaptive Systems: Fundamentals through Simulations, New York: John Wiley & Sons Inc., 2000.

[62] S. Samarasinghe, Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition, Boca Raton: Taylor and Francis Group, 2007.

[63] K. Swingler, Applying Neural Networks: A Practical Guide, San Francisco: Academia Press, 2001.

[64] A. J. Levesque, *Driver Modelling for Risk Assessment,* University of Windsor, 2012.

[65] I. Towfic and J. Johrendt, "A Neural Network Approach for Predicting Collision Severity," in *SAE World Congress 2014*, 2014.

[66] L. Aggarwal, K. Aggarwal and R. J. Urbanic, "Use of artificial neural networks for the development of an inverse kinematic solution and visual identification of singularity zone(s)," in *47th CIRP Conference on Manufacturing Systems*, Windsor, 2014.

[67] V. D. Tourassis and M. H. A. Jr, "Task decoupling in robot manipulators," *Journal of Intelligent and Robotic Systems,* vol. 14, no. 3, pp. 283-302, 1995.

[68] A. R.-1. Robot Safety Standard, "Risk Assessment and Methodology and Specific Guidelines for Safeguarding Robotic Systems," [Online]. Available: http://www.welding-robots.com/articles/viewing/robot-safety-standard-ansi-ria-r15-06-1999-on-risk-assessment-and-methodology-and-specific-guideline.

[69] P. Donelan, "Singularities of Robot Manipulators," in *Singularity Theory*, London, World Scientific Publishing Co. , 2005, pp. 189-218.

[70] P. Allen, "Kinematic Singularities and Jacobians," Columbia University, September 2013. [Online]. Available: http://www.cs.columbia.edu/~allen/F13/NOTES/jacobians.pdf.

[71] O.Hachour, "Path planning of Autonomous Mobile robot," *International Journal of Systems Applications, Engineering and Development,* vol. 2, no. 4, pp. 178-190, 2008.

[72] RobotWorkz, "FANUC M-16iB/20 RJ3iB".

[73] H.-Y. Xu, L. Hu, T. Hon-yuen, K. Shi and L. Xu, "A novel kinematic model for five-axis machine tools," *The International Journal of Advanced Manufacturing Technology,* vol. 67, no. 5-8, pp. 1297-1307, 2013.

APPENDICES

Appendix A: Results for SCARA Robot

Table 15: SCARA Joint Variable Range

S. No.	q_1 (rad)	q_2 (rad)	q_3 (mm)
1	-2.22	-2.53	-297.00
2	-2.03	-2.32	-288.67
3	-1.85	-2.11	-280.33
4	-1.66	-1.90	-272.00
5	-1.48	-1.69	-263.67
6	-1.29	-1.48	-255.33
7	-1.11	-1.26	-247.00
8	-0.92	-1.05	-238.67
9	-0.74	-0.84	-230.33
10	-0.55	-0.63	-222.00
11	-0.37	-0.42	-213.67
12	-0.18	-0.21	-205.33
13	0.00	0.00	-197.00
14	0.18	0.21	-188.67
15	0.37	0.42	-180.33
16	0.55	0.63	-172.00
17	0.74	0.84	-163.67
18	0.92	1.05	-155.33
19	1.11	1.27	-147.00
20	1.29	1.48	-138.67
21	1.48	1.69	-130.33
22	1.66	1.90	-122.00
23	1.85	2.11	-113.67
24	2.03	2.32	-105.33
25	2.22	2.53	-97.00
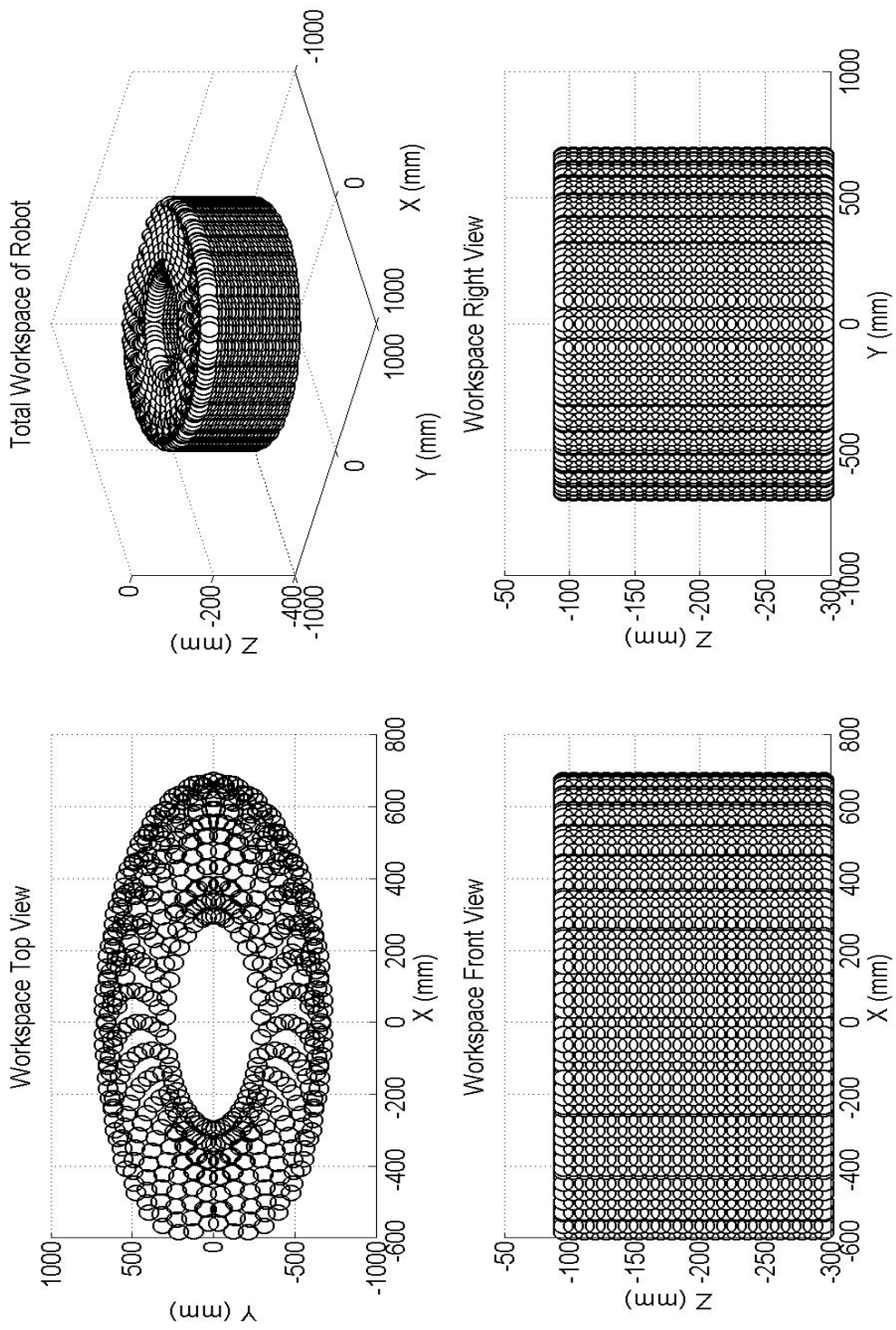
Figure 37: Complete Workspace of SCARA Robot

99
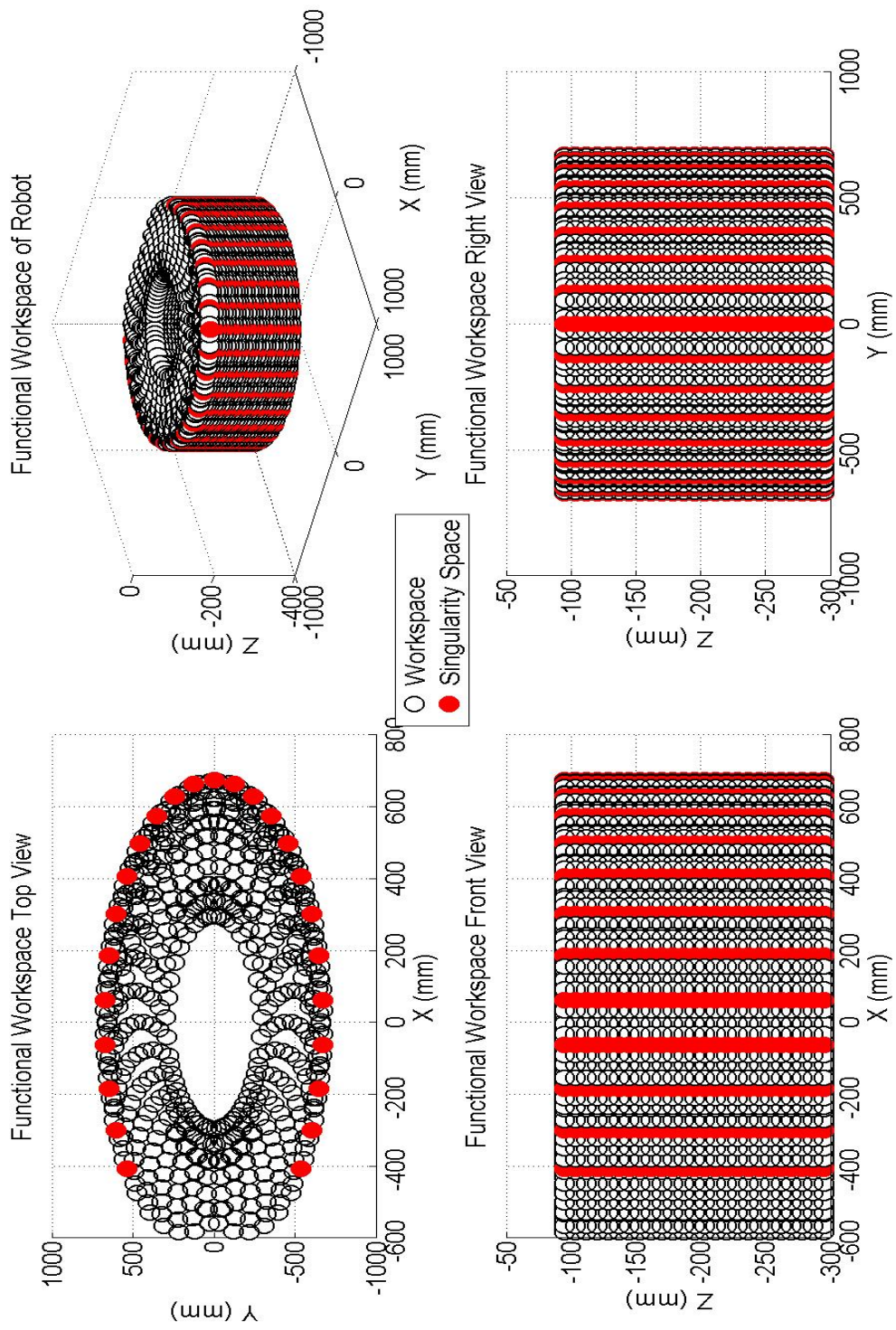
Figure 38: SCARA Workspace and Singularity Space
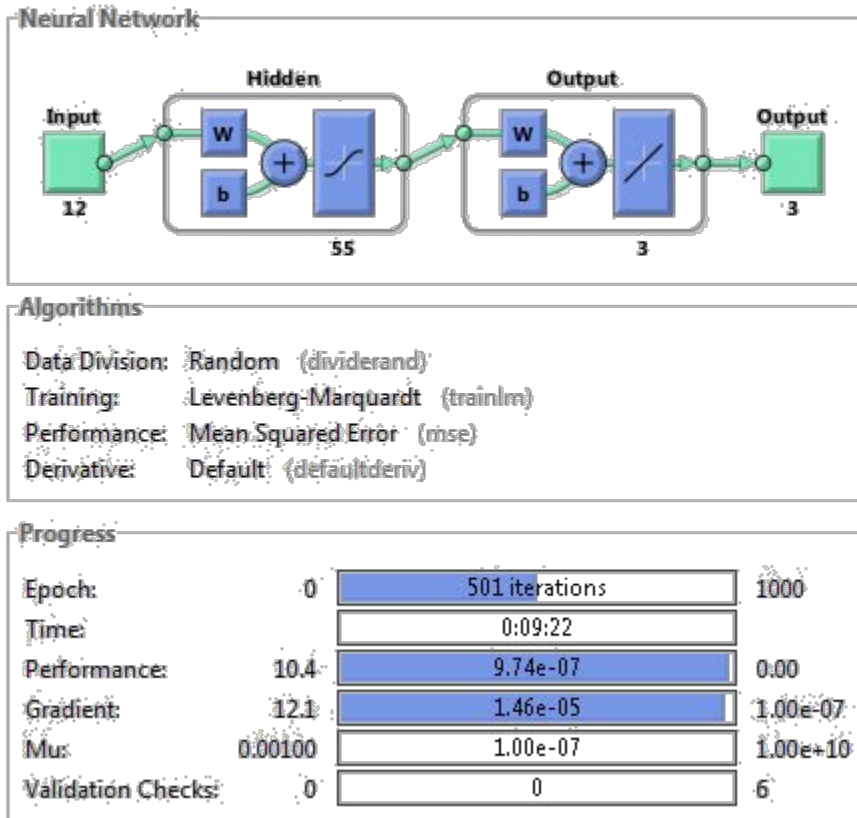
Figure 39: ANN Architecture for SCARA Robot
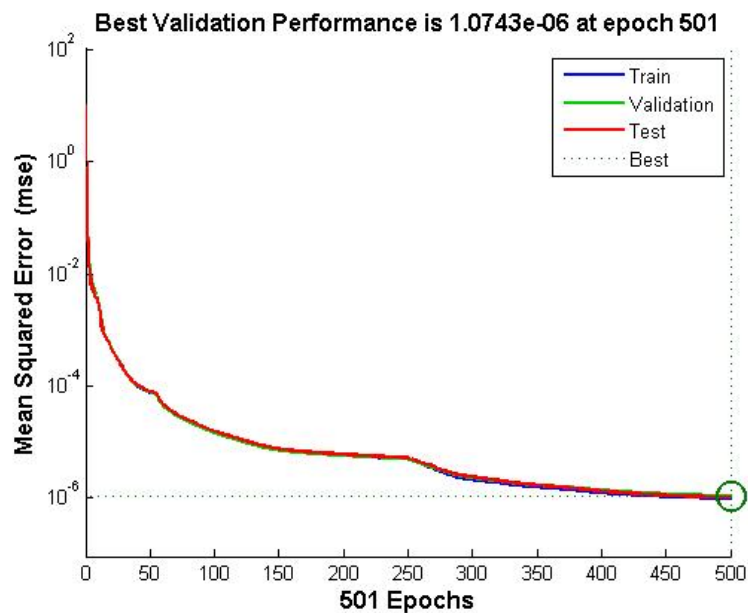
Figure 40: Performance Plot for SCARA Robot
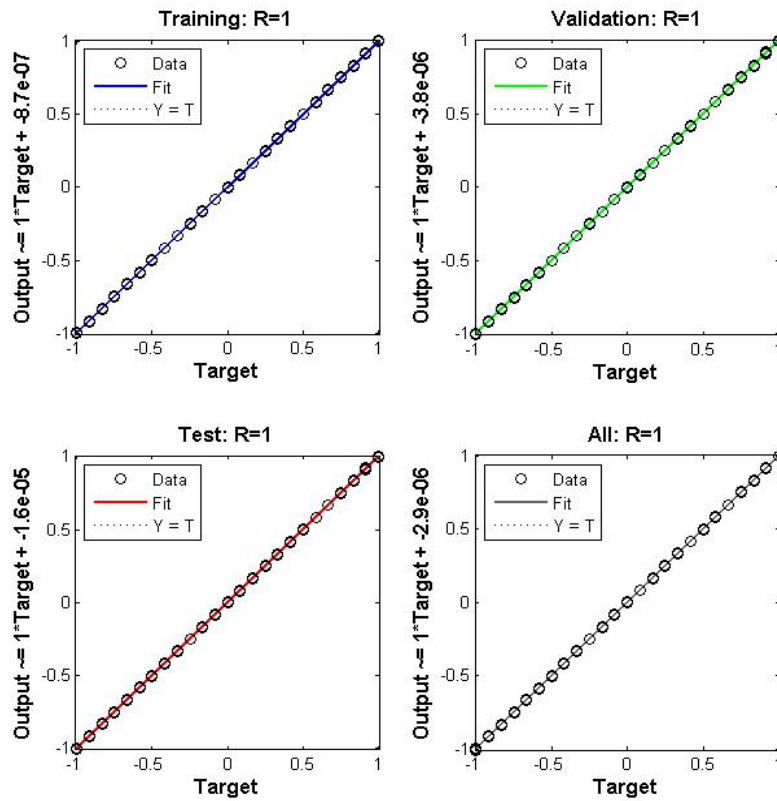
Figure 41: Regression Plot for SCARA Robot
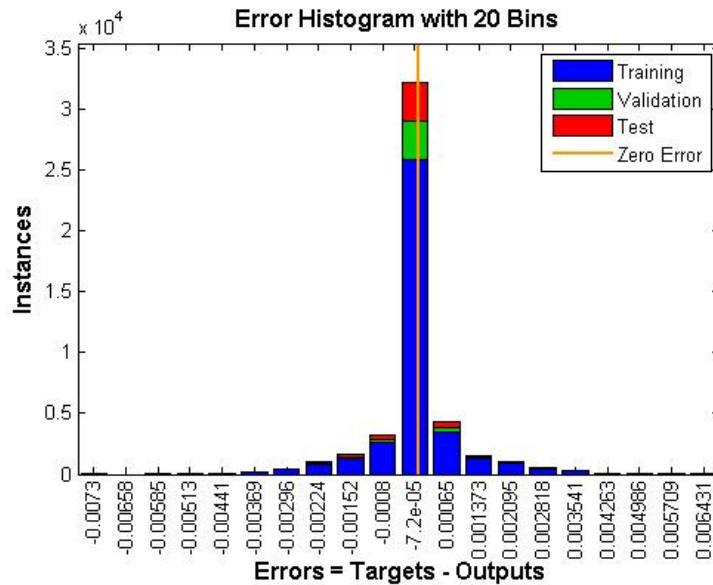
Figure 42: Error Histogram for SCARA Robot
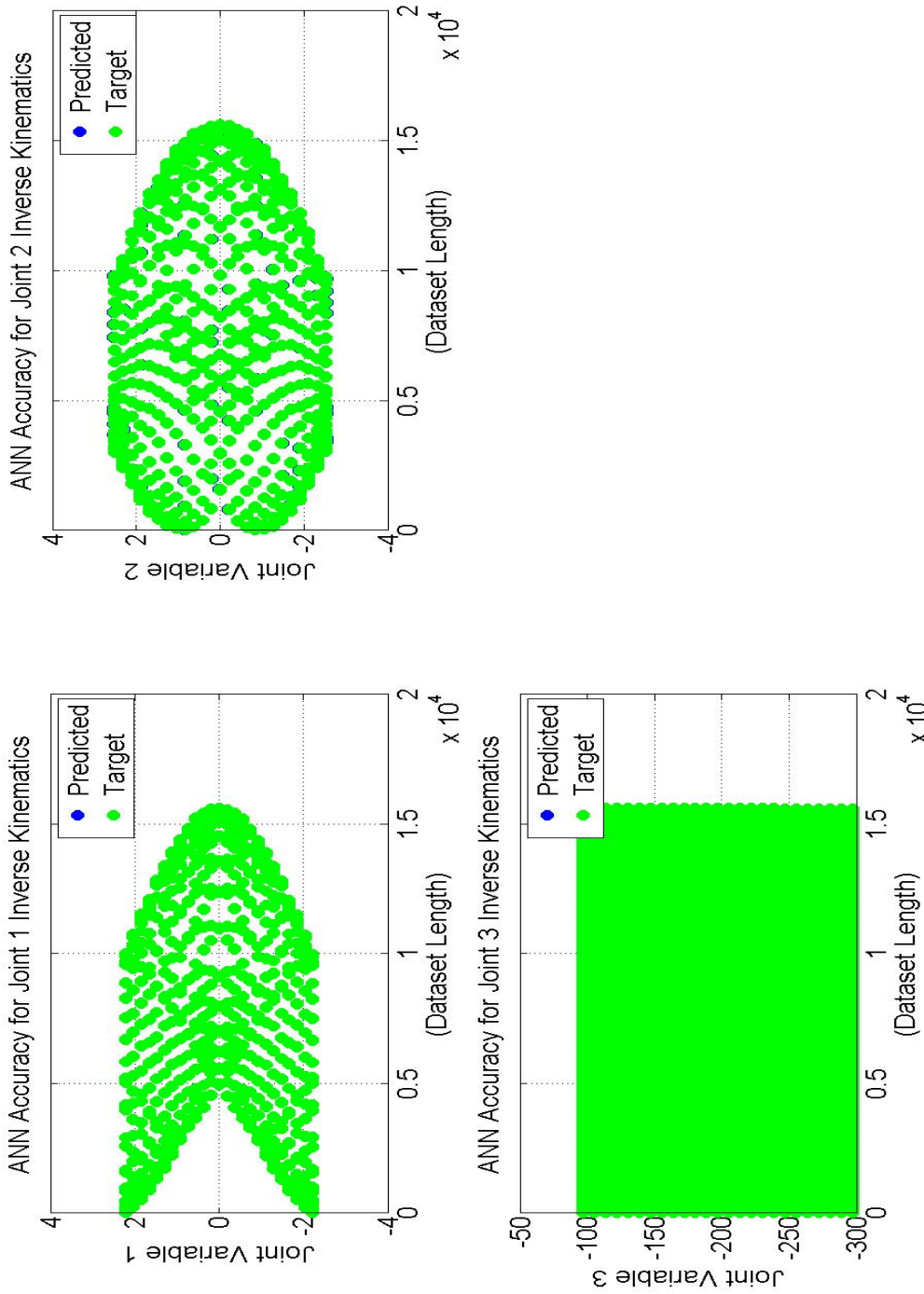
Figure 43: Inverse Kinematics Prediction for SCARA Robot
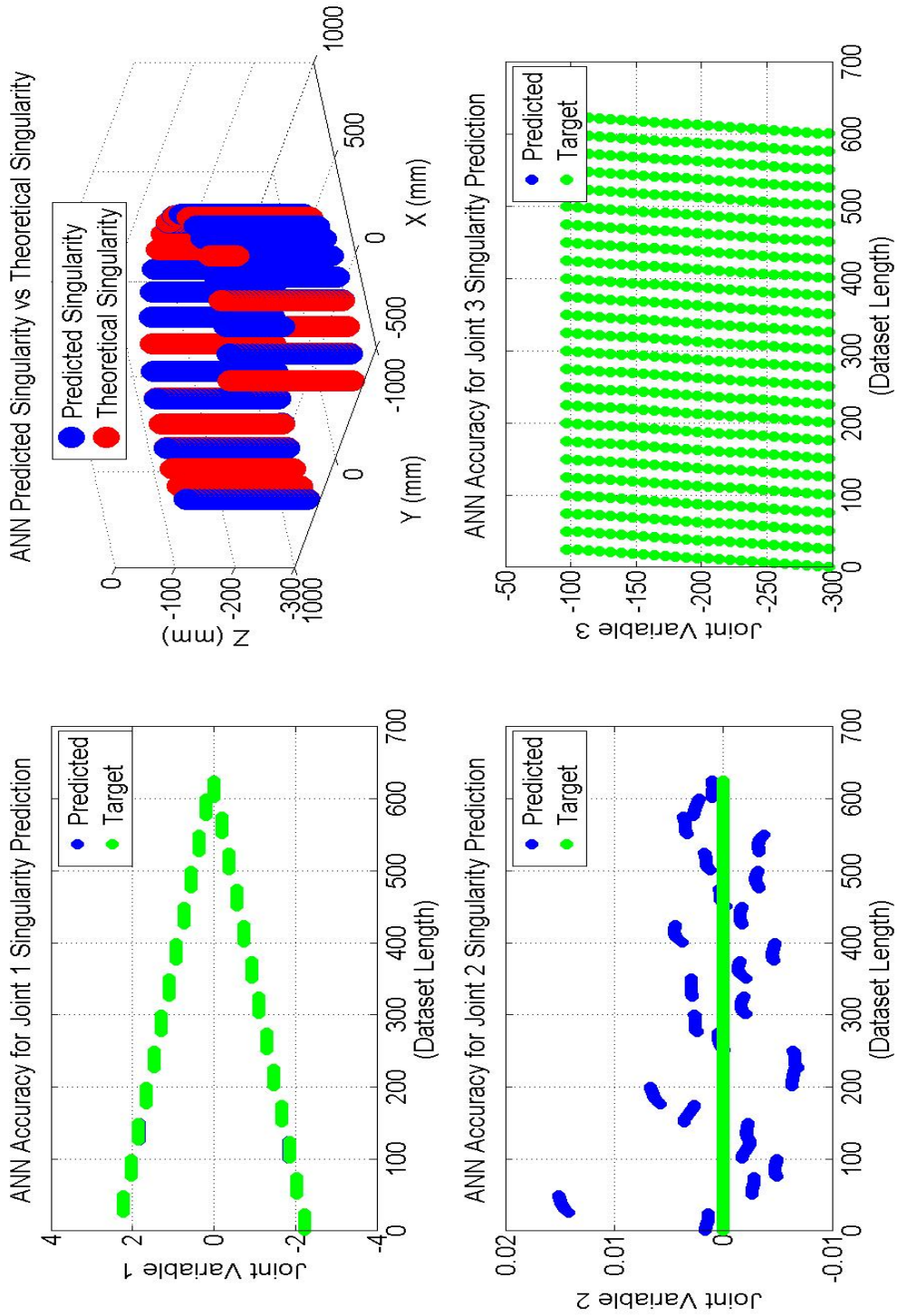
Figure 44: SCARA Singularity Joint Prediction
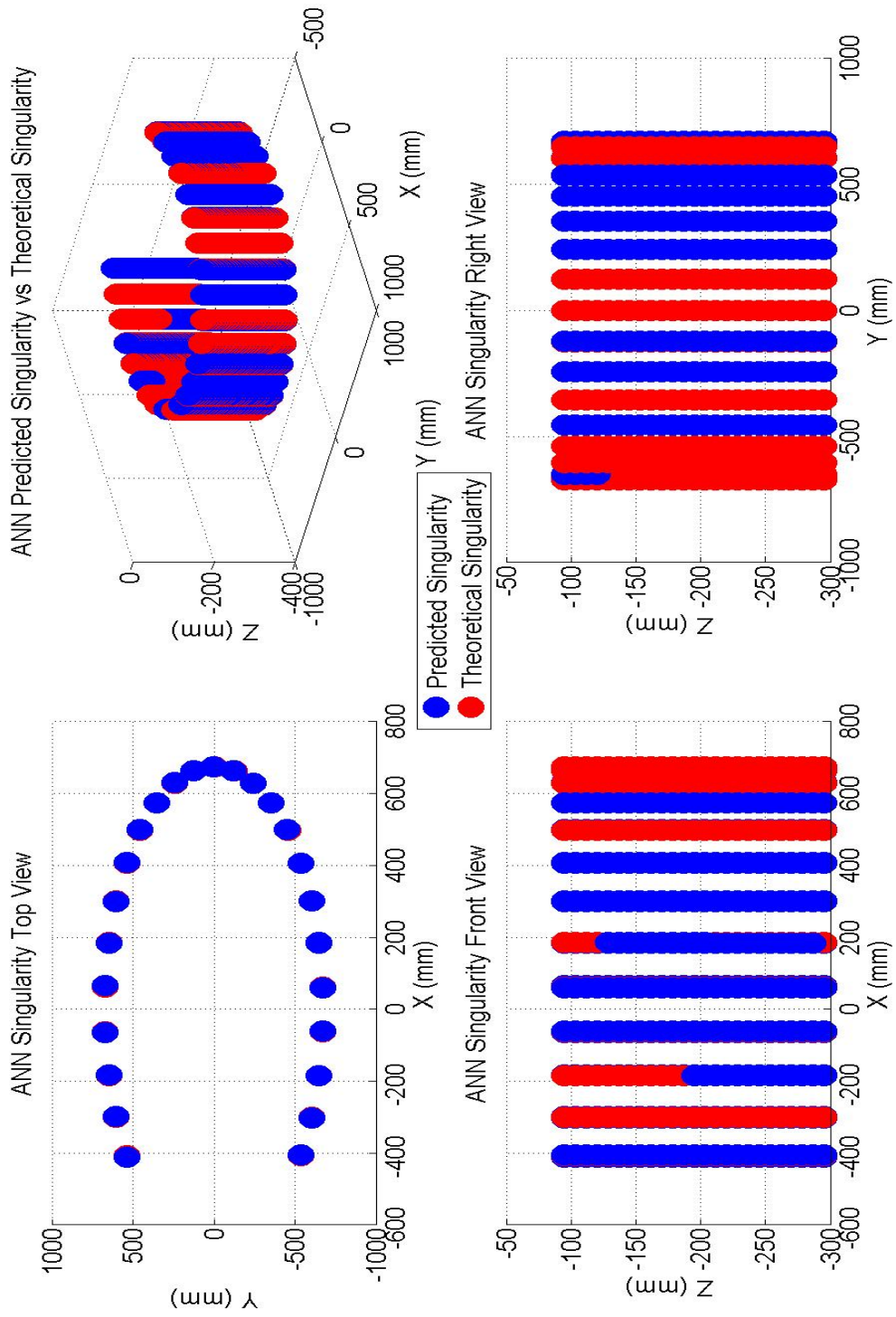
Figure 45: Theoretical *vs.* ANN Predicted Singularity

Appendix B: Results for FANUC M16iB/20 Robot
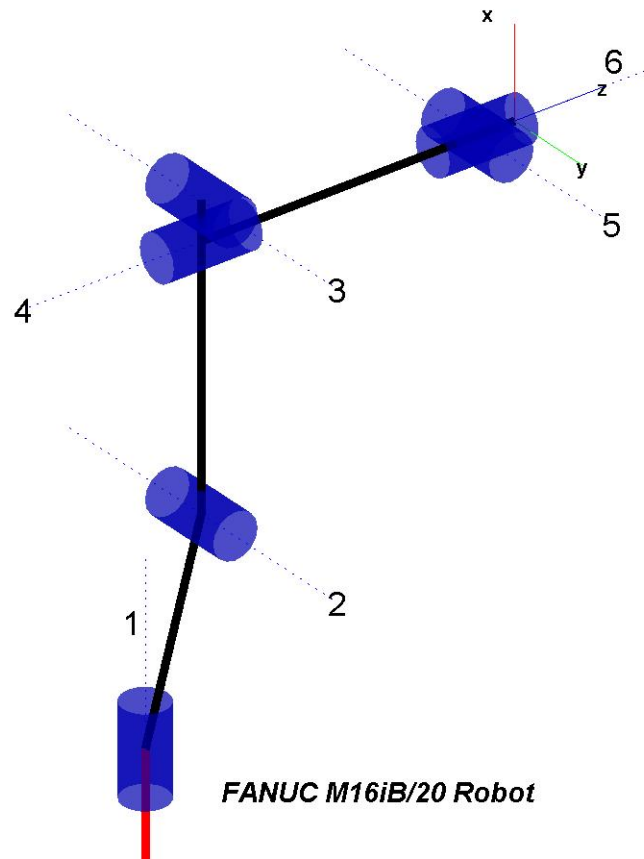
Figure 46: FANUC M16iB/20 Robot

Table 16: D-H Parameters for FANUC M16iB/20 Robot

Joint	Link Offset (m)		Joint Angle (rad)		Link Length (m)		Twist Angle (rad)		Lower Joint Limit	Upper Joint Limit
						Robot: Fanuc M16iB/20				
				D-H parameters						
1	$d_1 =$	0.525	$\theta_1 =$	θ_1	$a_1 =$	0.150	$\alpha_1 =$	-pi/2	-2.97	2.97
2	$d_2 =$	0	$\theta_2 =$	θ_2	$a_2 =$	0.770	$\alpha_2 =$	0	-1.57	2.79
3	$d_3 =$	0	$\theta_3 =$	θ_3	$a_3 =$	0.100	$\alpha_3 =$	pi/2	-2.97	5.06
4	$d_4 =$	0.740	$\theta_4 =$	θ_4	$a_4 =$	0	$\alpha_4 =$	-pi/2	-3.49	3.49
5	$d_5 =$	0	$\theta_5 =$	θ_5	$a_5 =$	0	$\alpha_5 =$	pi/2	-2.44	2.44
6	$d_6 =$	0.100	$\theta_6 =$	θ_6	$a_6 =$	0	$\alpha_6 =$	0	-7.85	7.85

Table 17: FANUC Joint Angle Range for Workspace Generation

Angle Configuration Range for Workspace Generation						
S. No.	q1 (rad)	q2 (rad)	q3 (rad)	q4 (rad)	q5 (rad)	q6 (rad)
1	-2.97	-1.57	-2.97	-3.49	-2.44	-7.85
2	-2.31	-1.09	-2.08	-2.71	-1.90	-6.11
3	-1.65	-0.60	-1.18	-1.94	-1.36	-4.36
4	-0.99	-0.12	-0.29	-1.16	-0.81	-2.62
5	-0.33	0.37	0.60	-0.39	-0.27	-0.87
6	0.33	0.85	1.49	0.39	0.27	0.87
7	0.99	1.34	2.39	1.16	0.81	2.62
8	1.65	1.82	3.28	1.94	1.36	4.36
9	2.31	2.31	4.17	2.71	1.90	6.11
10	2.97	2.79	5.06	3.49	2.44	7.85

Table 18: FANUC Joint Angle Range for Training ANN

Angle Configuration Range for Training ANN						
S. No.	q1 (rad)	q2 (rad)	q3 (rad)	q4 (rad)	q5 (rad)	q6 (rad)
1	1.44	1.80	3.90	-3.09	1.82	-2.28
2	1.65	-0.58	3.37	-2.32	-1.32	3.09
3	0.87	-0.09	0.52	-0.19	0.99	1.65
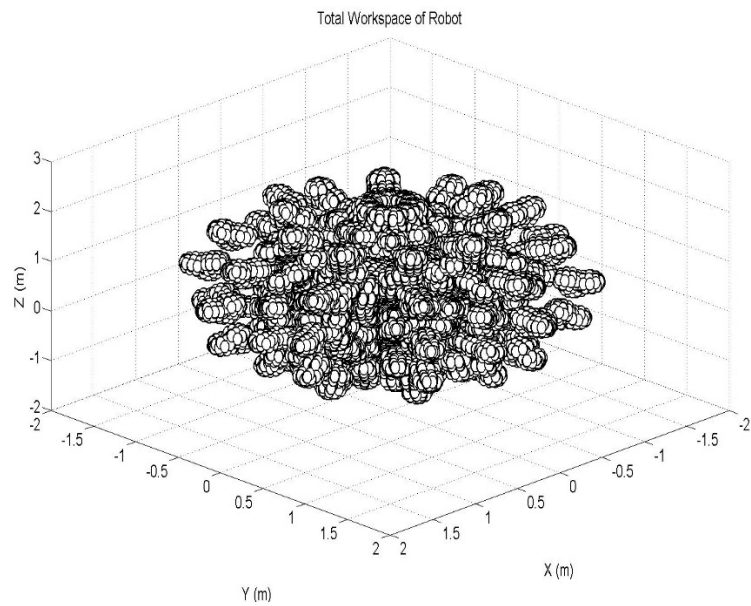
Figure 47: Workspace of FANUC M16iB/20 Robot
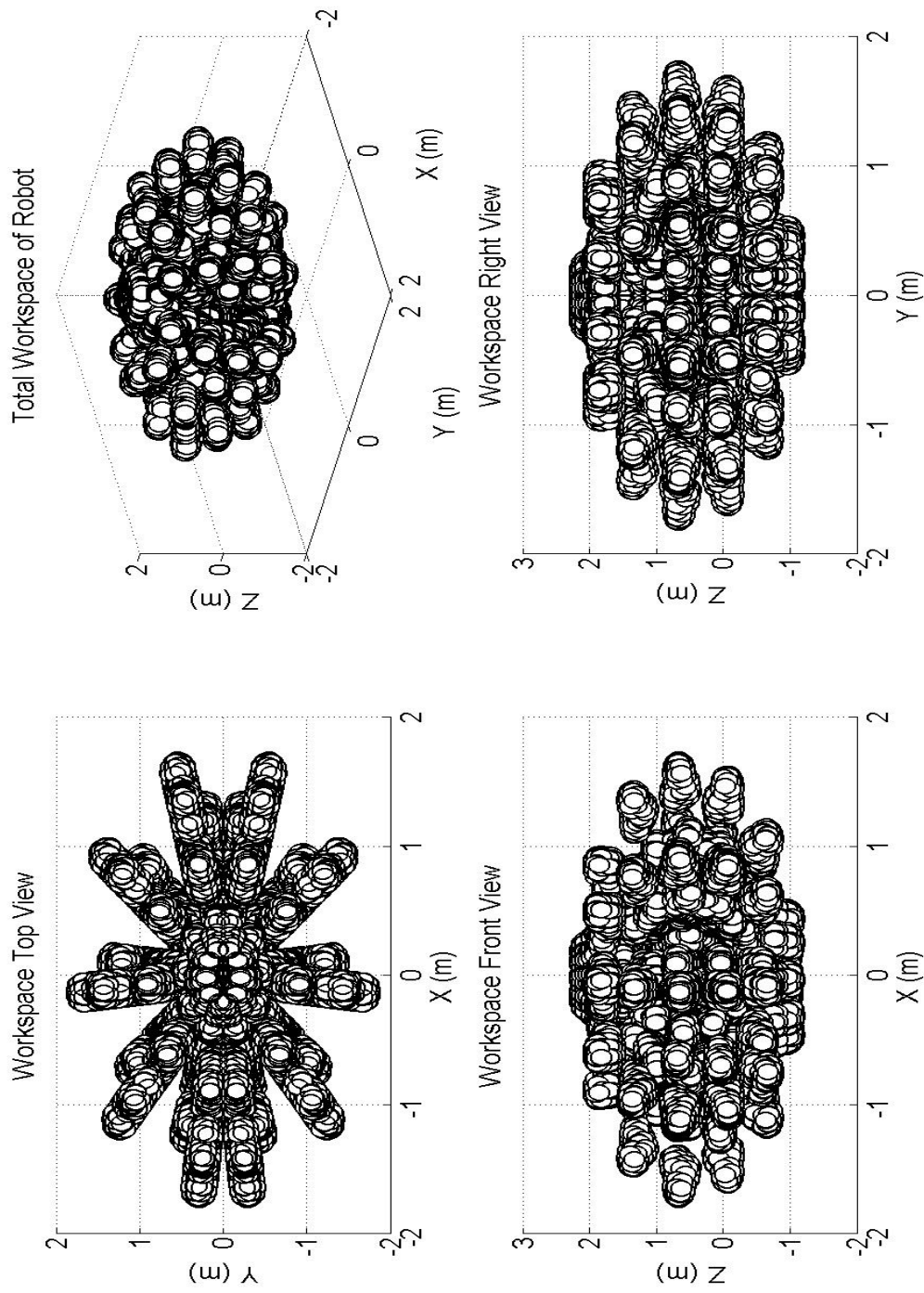
Figure 48: Total Workspace for FANUC M16iB/20 Robot

Functional Workspace Top View

Functional Workspace of Robot

Functional Workspace Front View

Functional Workspace Right View

O Workspace
● Singularity Space
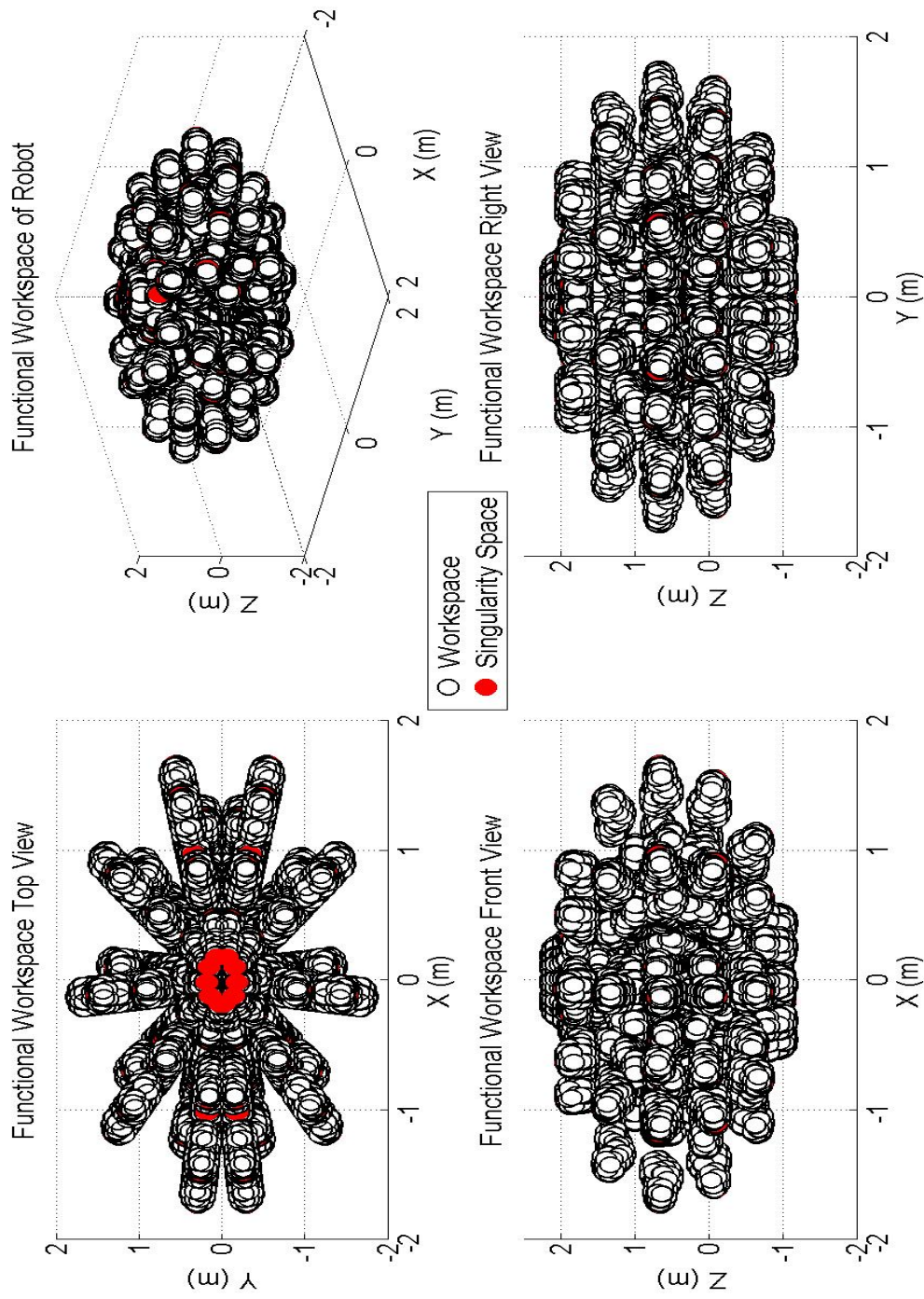
Figure 49: Workspace and Singularity Space for FANUC M16iB/20 Robot

109
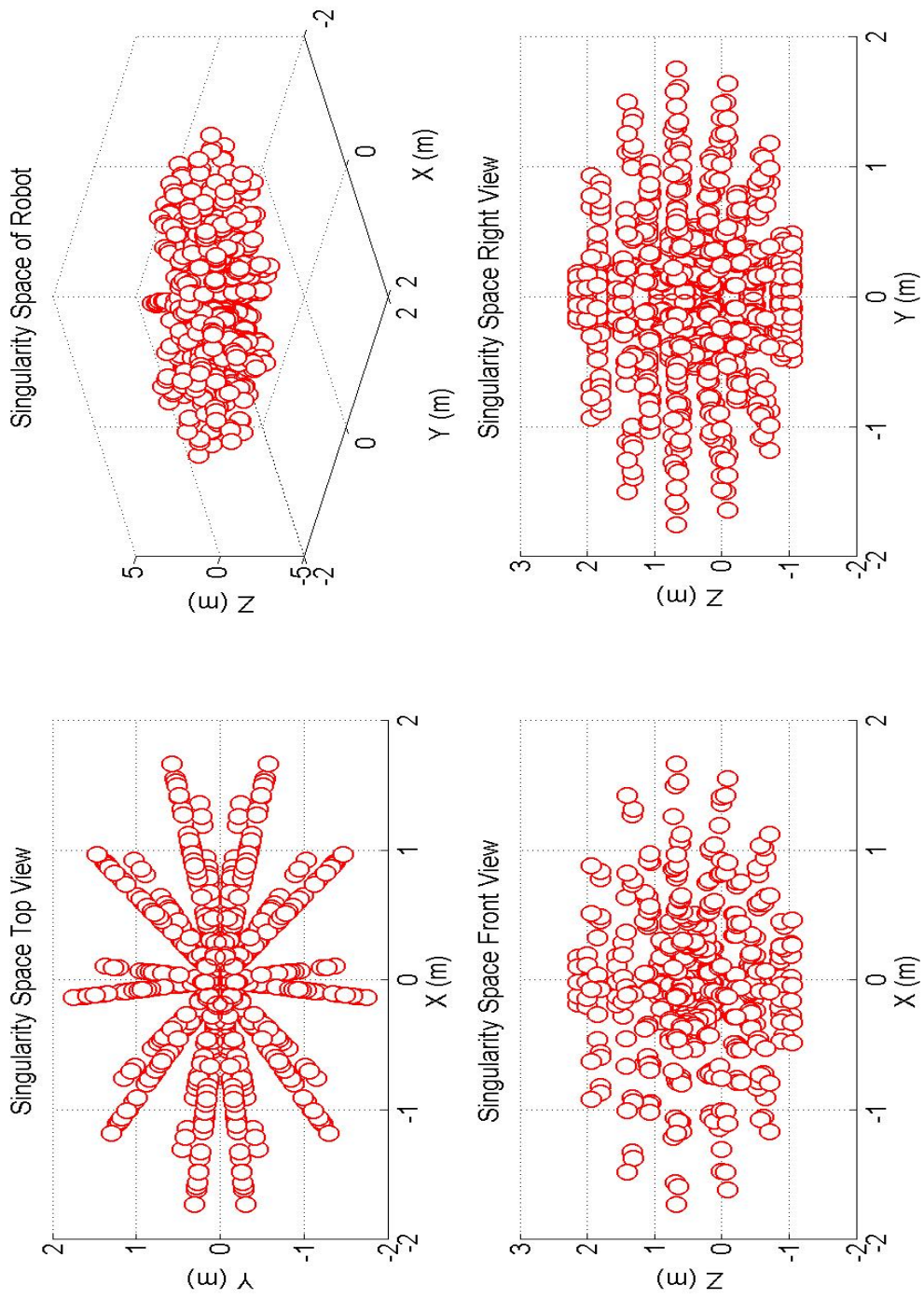
Figure 50: Singularity Space of FANUC M16iB/20 Robot
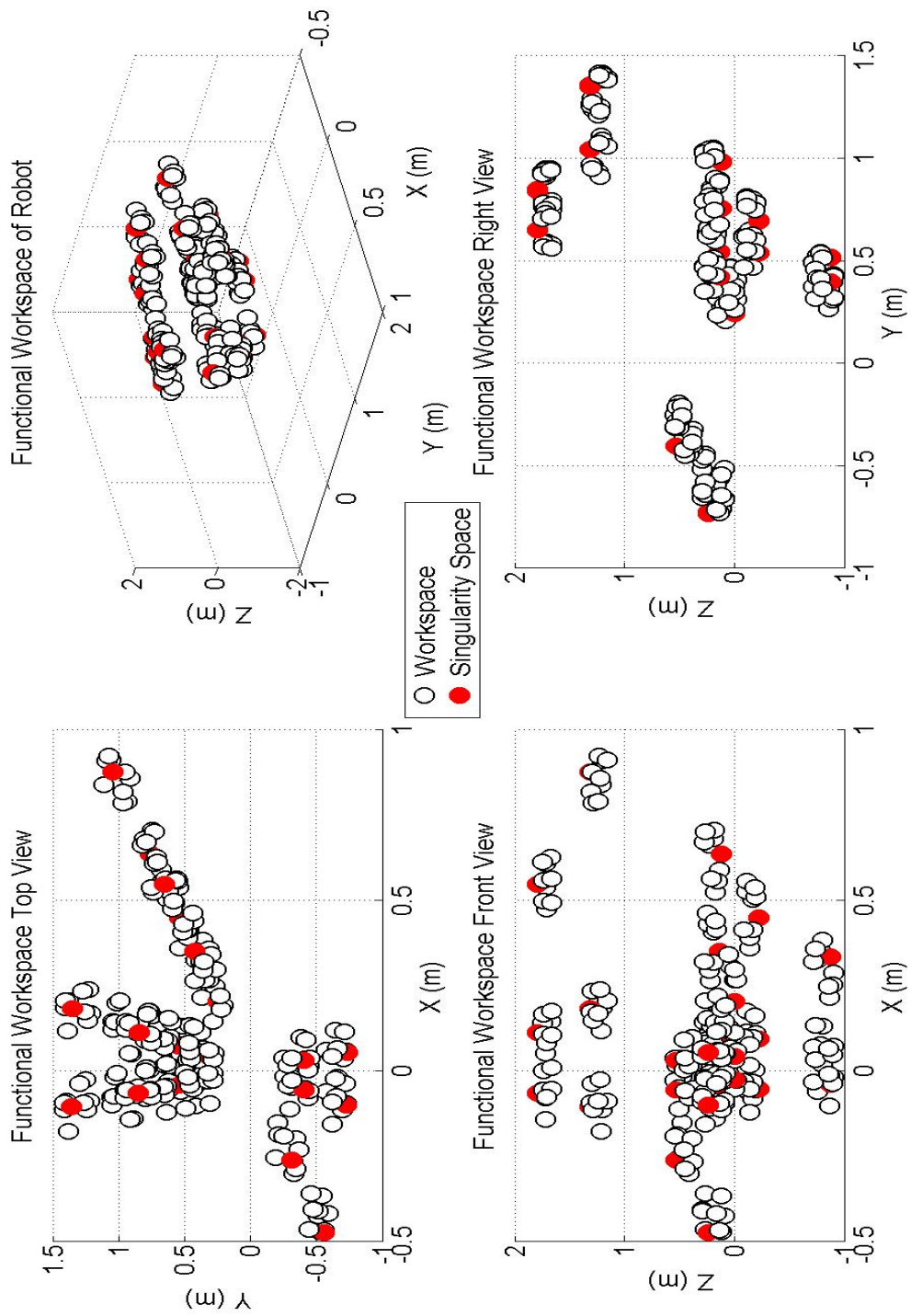
Figure 51 : Functional Workspace of FANUC M16iB/20 Robot
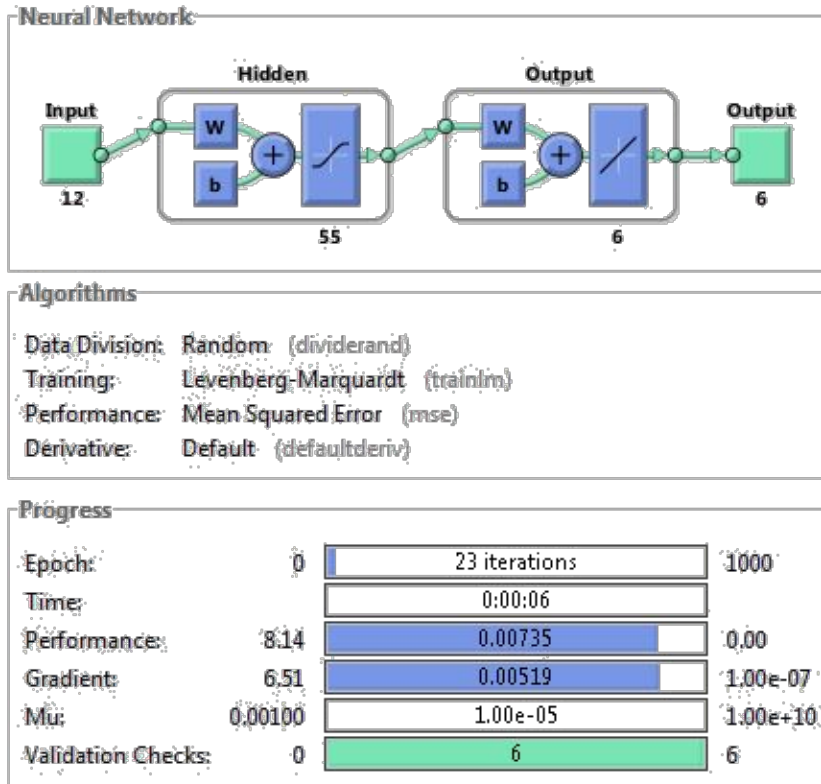
Figure 52: ANN Architecture for FANUC M16iB/20 Robot
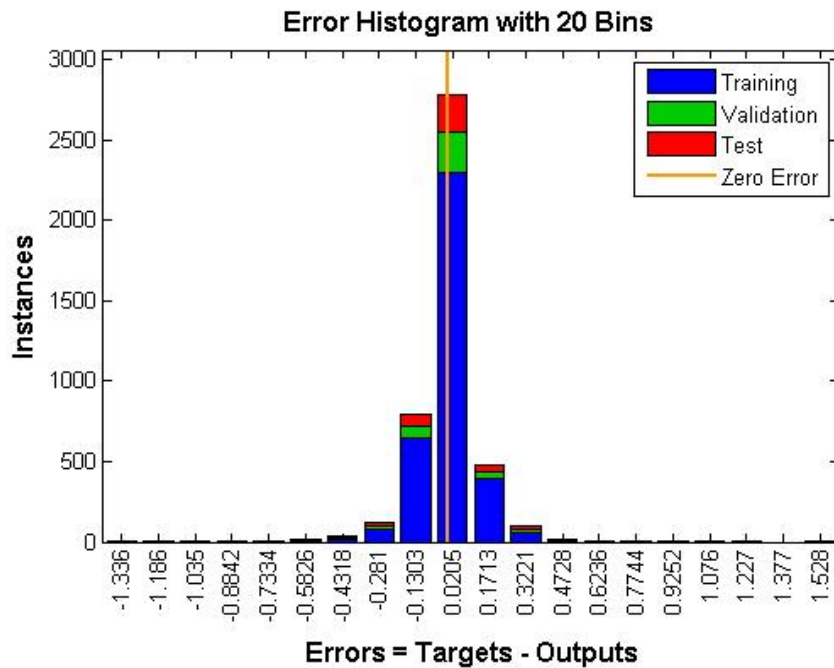
Figure 53: Error Histogram for FANUC M16iB/20 Robot
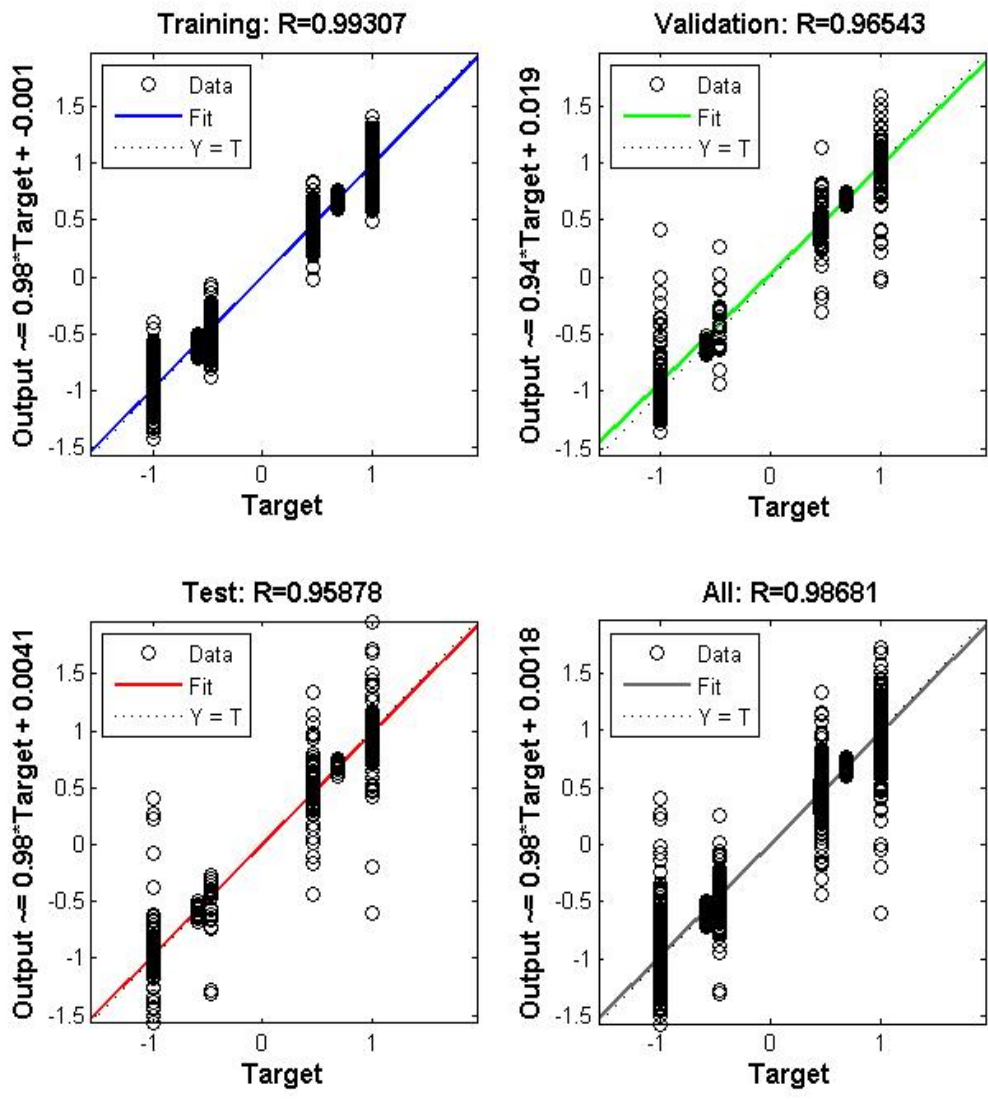
Figure 54: Regression Plot for FANUC M16iB/20
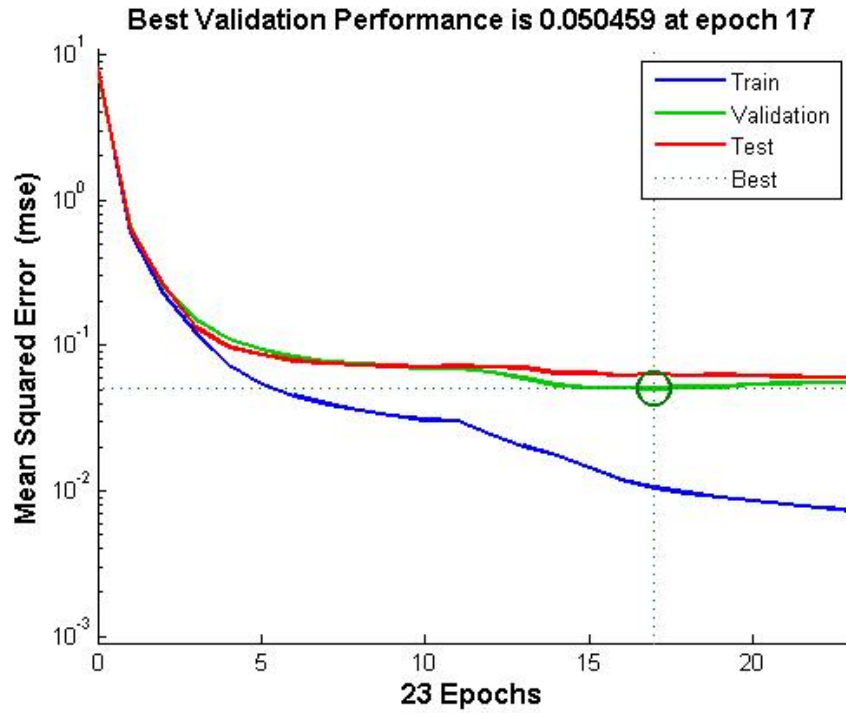
Figure 55: Performance Plot for FANUC M16iB/20 Robot
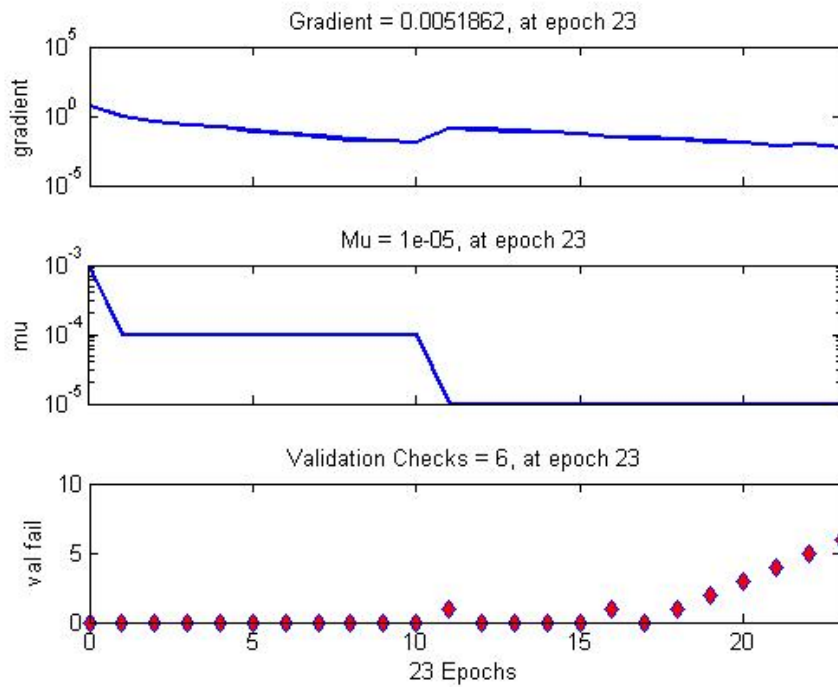
Figure 56: Training State Plot for FANUC M16iB/20 Robot

114
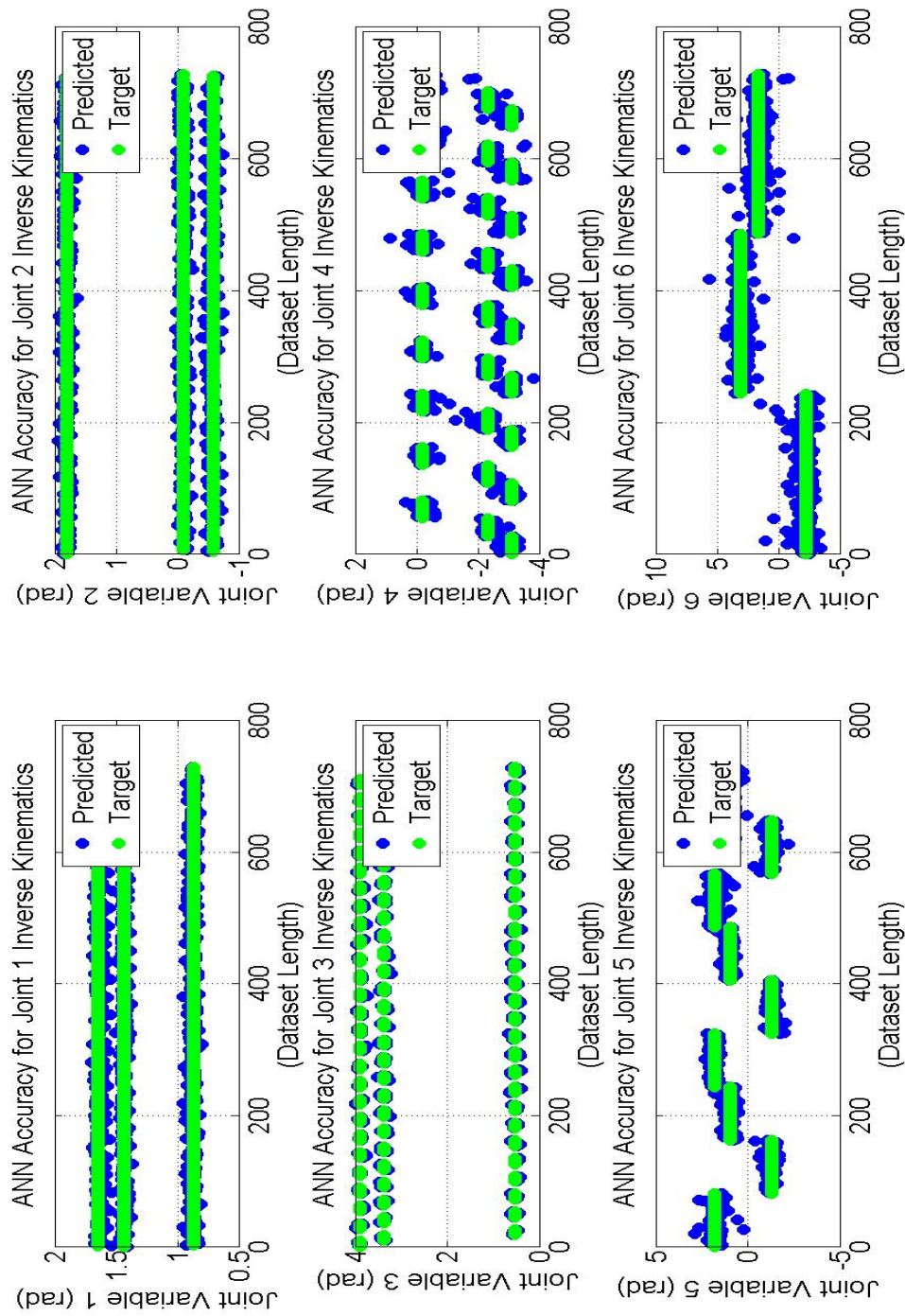
Figure 57: Inverse Kinematics Prediction for FANUC M16iB/20 Robot
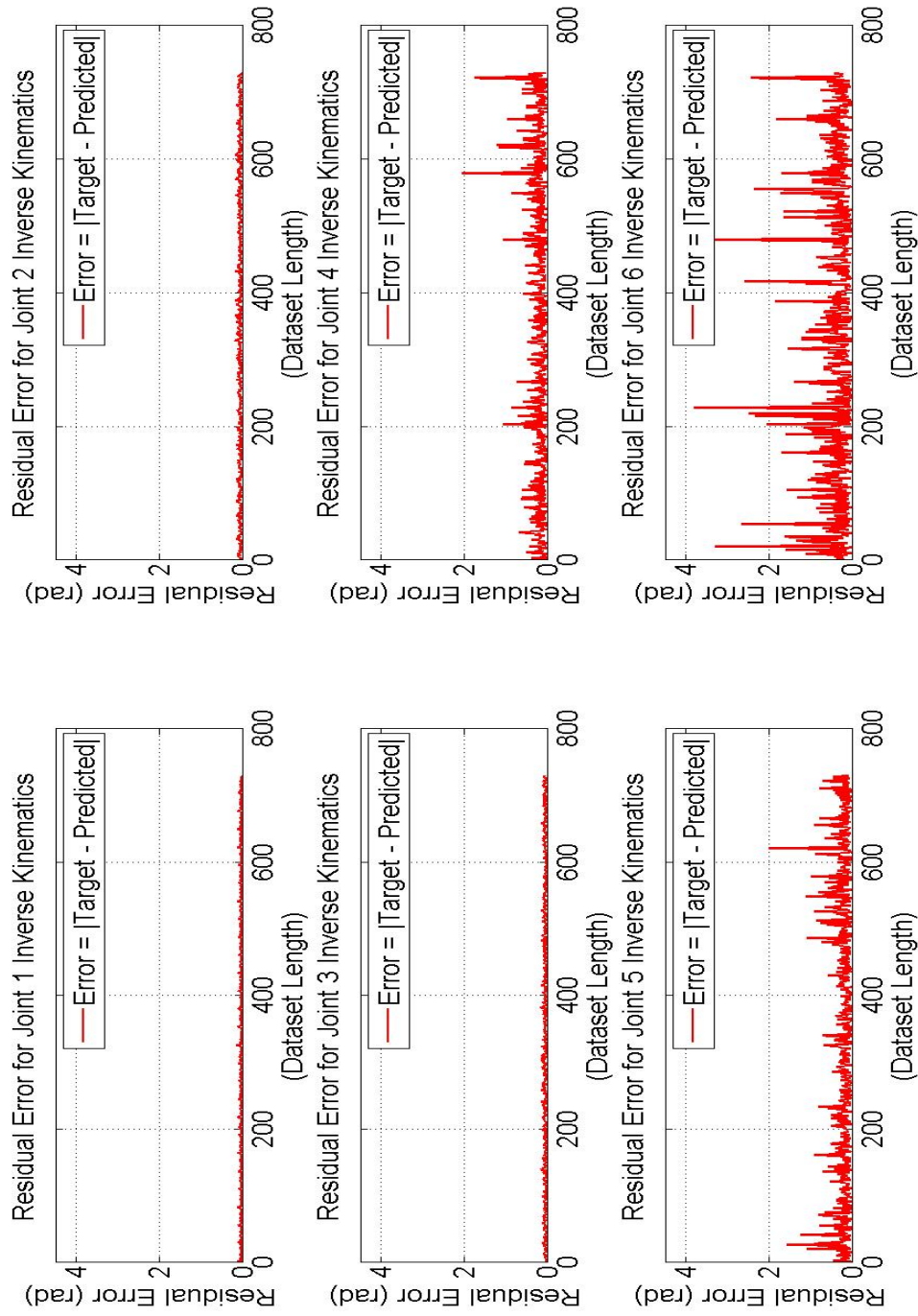
Figure 58: Absolute Residual Error in Inverse Kinematics Prediction for FANUC M16iB/20 Robot
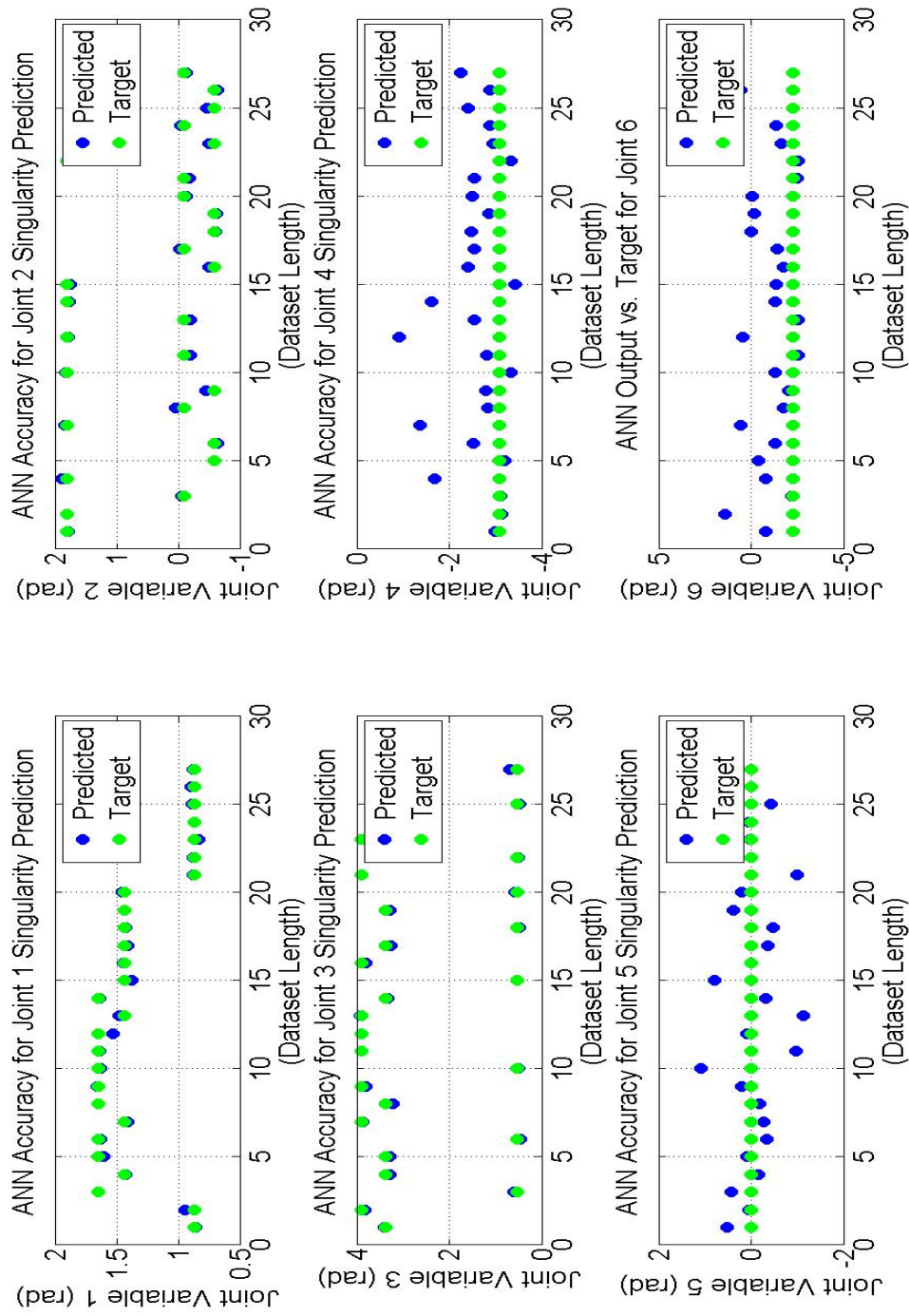
Figure 59: FANUC M16iB/20 Singularity Joint Prediction
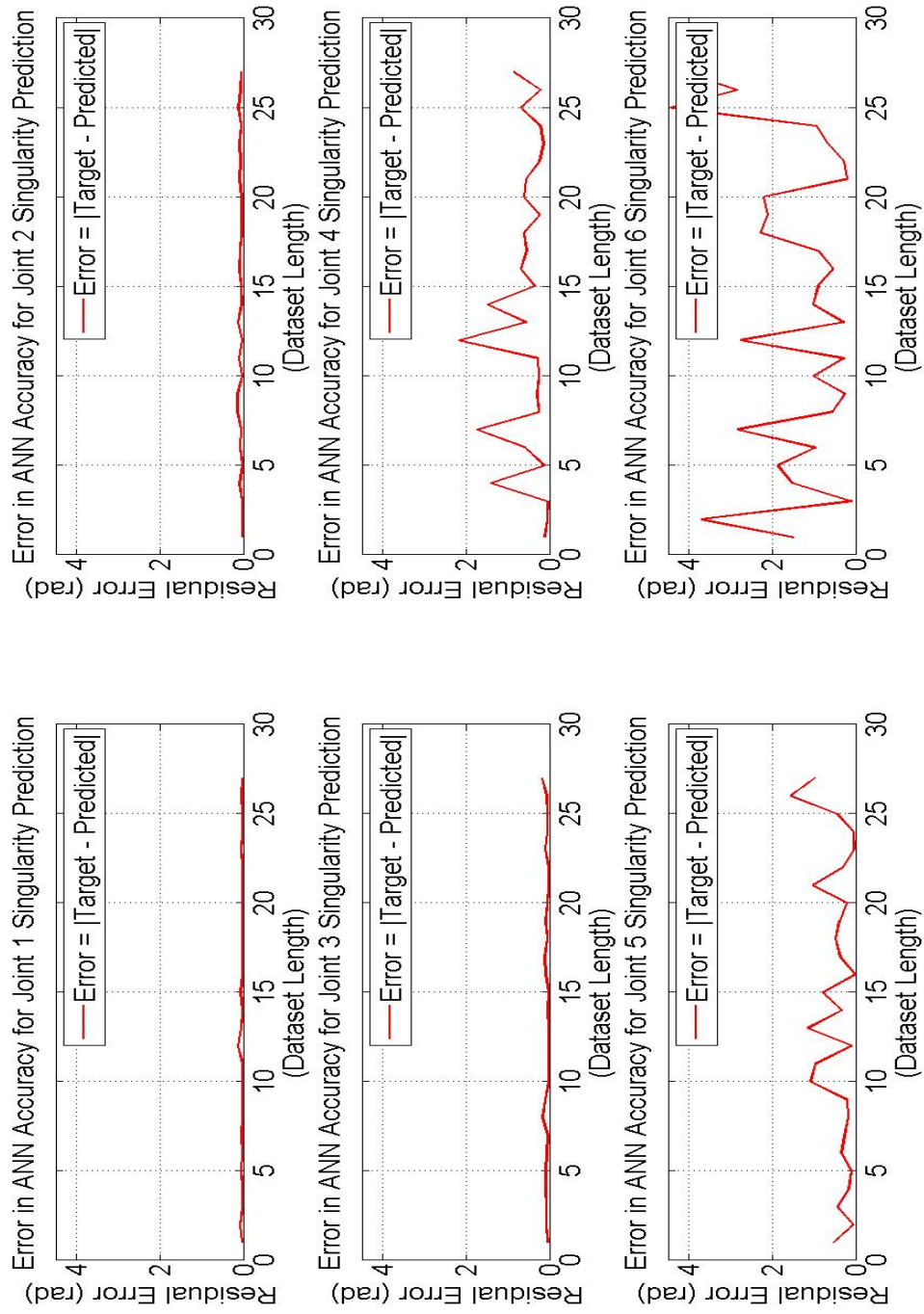
Figure 60: Absolute Residual Error in FANUC M16iB/20 Singularity Joint Prediction
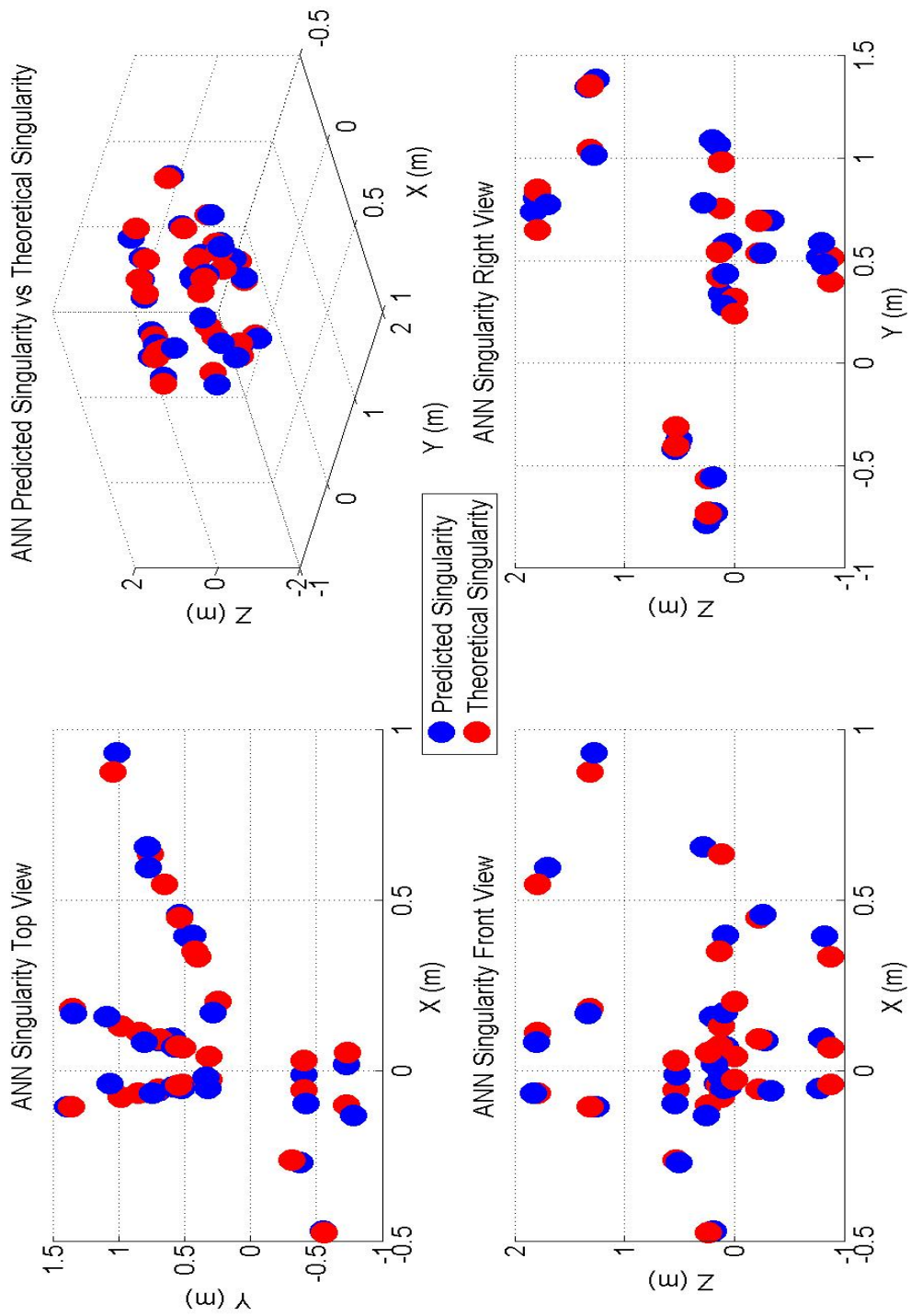
Figure 61: Theoretical vs. ANN Predicted Singularity

MATLAB Output for FANUC M16iB/20 Robot:

```
A06 =

[ - sin(theta6)*(cos(theta4)*sin(theta1) - sin(theta4)*(cos(theta1)*sin(theta2)*sin
(theta3) - cos(theta1)*cos(theta2)*cos(theta3))) - cos(theta6)*(cos(theta5)*(sin(theta1)
*sin(theta4) + cos(theta4)*(cos(theta1)*sin(theta2)*sin(theta3) - cos(theta1)*cos(theta2)
*cos(theta3))) + sin(theta5)*(cos(theta1)*cos(theta2)*sin(theta3) + cos(theta1)*cos
(theta3)*sin(theta2))), sin(theta6)*(cos(theta5)*(sin(theta1)*sin(theta4) + cos(theta4)*
(cos(theta1)*sin(theta2)*sin(theta3) - cos(theta1)*cos(theta2)*cos(theta3))) + sin
(theta5)*(cos(theta1)*cos(theta2)*sin(theta3) + cos(theta1)*cos(theta3)*sin(theta2))) -
cos(theta6)*(cos(theta4)*sin(theta1) - sin(theta4)*(cos(theta1)*sin(theta2)*sin(theta3) -
cos(theta1)*cos(theta2)*cos(theta3))), cos(theta5)*(cos(theta1)*cos(theta2)*sin(theta3) +
cos(theta1)*cos(theta3)*sin(theta2)) - sin(theta5)*(sin(theta1)*sin(theta4) + cos(theta4)
*(cos(theta1)*sin(theta2)*sin(theta3) - cos(theta1)*cos(theta2)*cos(theta3))), (3*cos
(theta1))/20 + (77*cos(theta1)*cos(theta2))/100 - (cos(theta1)*sin(theta2)*sin(theta3))
/10 - (sin(theta1)*sin(theta4)*sin(theta5))/10 + (sin(theta2 + theta3)*cos(theta1)*cos
(theta5))/10 + (cos(theta1)*cos(theta2)*cos(theta3))/10 + (37*cos(theta1)*cos(theta2)*sin
(theta3))/50 + (37*cos(theta1)*cos(theta3)*sin(theta2))/50 + (cos(theta1)*cos(theta2)*cos
(theta3)*cos(theta4)*sin(theta5))/10 - (cos(theta1)*cos(theta4)*sin(theta2)*sin(theta3)
*sin(theta5))/10]
[   sin(theta6)*(cos(theta1)*cos(theta4) + sin(theta4)*(sin(theta1)*sin(theta2)*sin
(theta3) - cos(theta2)*cos(theta3)*sin(theta1))) + cos(theta6)*(cos(theta5)*(cos(theta1)
*sin(theta4) - cos(theta4)*(sin(theta1)*sin(theta2)*sin(theta3) - cos(theta2)*cos(theta3)
*sin(theta1))) - sin(theta5)*(cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin
(theta1)*sin(theta2))), cos(theta6)*(cos(theta1)*cos(theta4) + sin(theta4)*(sin(theta1)
*sin(theta2)*sin(theta3) - cos(theta2)*cos(theta3)*sin(theta1))) - sin(theta6)*(cos
(theta5)*(cos(theta1)*sin(theta4) - cos(theta4)*(sin(theta1)*sin(theta2)*sin(theta3) -
cos(theta2)*cos(theta3)*sin(theta1))) - sin(theta5)*(cos(theta2)*sin(theta1)*sin(theta3)
+ cos(theta3)*sin(theta1)*sin(theta2))), sin(theta5)*(cos(theta1)*sin(theta4) - cos
(theta4)*(sin(theta1)*sin(theta2)*sin(theta3) - cos(theta2)*cos(theta3)*sin(theta1))) +
cos(theta5)*(cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2)),
(3*sin(theta1))/20 + (77*cos(theta2)*sin(theta1))/100 + (37*cos(theta2)*sin(theta1)*sin
(theta3))/50 + (37*cos(theta3)*sin(theta1)*sin(theta2))/50 + (cos(theta1)*sin(theta4)*sin
(theta5))/10 - (sin(theta1)*sin(theta2)*sin(theta3))/10 + (sin(theta2 + theta3)*cos
(theta5)*sin(theta1))/10 + (cos(theta2)*cos(theta3)*sin(theta1))/10 + (cos(theta2)*cos
(theta3)*cos(theta4)*sin(theta1)*sin(theta5))/10 - (cos(theta4)*sin(theta1)*sin(theta2)
*sin(theta3)*sin(theta5))/10]
[
sin(theta2 + theta3)*sin(theta4)*sin(theta6) - cos(theta6)*(cos(theta2 + theta3)*sin
(theta5) + sin(theta2 + theta3)*cos(theta4)*cos(theta5)),
sin(theta6)*(cos(theta2 + theta3)*sin(theta5) + sin(theta2 + theta3)*cos(theta4)*cos
(theta5)) + sin(theta2 + theta3)*cos(theta6)*sin(theta4),
cos(theta2 + theta3)*cos(theta5) - sin(theta2 + theta3)*cos(theta4)*sin(theta5),
(37*cos(theta2 + theta3))/50 - sin(theta2 + theta3)/10 - (77*sin(theta2))/100 + (sin
(theta4 - theta5)*sin(theta2 + theta3))/20 - (sin(theta2 + theta3)*sin(theta4 + theta5))
/20 + (cos(theta2 + theta3)*cos(theta5))/10 + 21/40]
[
0,
0,
0,
1]
```

Input weights =

ans =

  Columns 1 through 9

```
         -2.19        -1.36         0.91        -1.44         1.73        -0.31✓
-1.77         0.38        -0.92
          0.01        -0.03        -0.01        -0.08         0.02         0.04✓
-0.01         0.02         0.15
         -1.93        -0.12        -1.05         0.76        -0.16        -0.76✓
-0.66        -2.46        -0.08
         -0.39        -0.37         0.06        -0.60         0.26        -0.42✓
0.13        -0.29        -0.36
         -0.01         0.65         0.01        -0.76        -1.01         0.17✓
1.45         0.46        -0.39
         -0.00        -0.01        -0.65         0.01         0.01         0.03✓
0.01         0.01         0.04
         -0.11        -1.19         0.68         0.12         1.21        -0.49✓
-0.27         1.79         0.41
          0.65        -1.12        -0.04         0.00         0.39         0.15✓
-0.72        -1.82         0.11
         -0.26        -0.44         0.03         0.29        -0.93         0.79✓
1.30        -0.95        -1.99
          0.22        -0.81        -0.05        -0.21         0.20         0.50✓
0.41         0.30        -0.12
         -1.30         0.05         0.23        -0.84         0.57         0.10✓
-1.03         1.05        -0.98
         -0.92         0.76         0.16         0.88        -0.41         0.18✓
1.56         1.09        -0.22
          1.15        -0.81        -0.47         0.57         0.37        -0.07✓
-0.80         0.83        -0.02
          0.02        -0.00        -0.09        -0.06         0.02        -0.03✓
0.01        -0.01        -0.15
          0.23         0.23         0.51        -0.75        -1.00        -0.15✓
-0.16        -0.79         0.61
         -0.62        -1.60        -0.14         0.06         0.93         0.10✓
0.91        -1.45         0.43
         -0.03         0.24        -0.20        -0.06         0.52         0.51✓
0.47         0.09         0.03
          0.08         0.12        -0.33        -0.14        -0.05         0.42✓
0.07        -0.04         0.06
          0.63        -0.58        -0.03         1.06         0.24        -0.70✓
-0.78        -0.08         0.38
         -0.58         1.70        -0.58         0.81        -0.54        -1.13✓
0.76         0.51         1.02
         -0.46         0.94        -0.99        -0.28        -0.38         0.26✓
0.76         2.17        -0.66
         -0.01         0.00        -0.03         0.06         0.05        -0.05✓
-0.10        -0.00         0.48
```

```
         0.81      -1.98       -0.40        0.44        0.44        1.26↙
-0.97       -1.23        0.71
         0.33       1.69       -0.37        0.77        0.07        0.84↙
 1.94        1.55        0.41
        -0.01      -0.19        0.15        0.08        0.00       -0.18↙
 0.33        0.14       -0.07
        -0.18       0.08       -1.12        0.02        0.47        0.35↙
 0.46        1.64        0.45
        -0.05      -1.30       -0.69       -0.11        0.37       -0.73↙
 0.50       -0.26        0.18
        -0.25       1.48       -0.57       -0.74        0.40        0.34↙
 0.73       -0.50       -1.49
         0.02       0.00        0.65       -0.04       -0.01        0.08↙
-0.00       -0.01       -0.05
        -0.34      -1.13       -0.06        0.01       -0.36       -0.16↙
 0.68       -0.83       -0.25
        -0.82       0.65       -0.03        0.47       -0.24       -0.36↙
 0.22        1.11        0.06
         0.15       0.53        0.28       -0.39        0.25        0.07↙
-0.24        0.13       -0.52
        -0.44      -0.97        0.44       -0.90       -1.14       -0.37↙
 1.40        0.99       -0.33
         0.61      -0.61        1.41       -0.83       -0.25        0.01↙
 0.33       -0.38       -0.89
        -0.66       0.44        0.50       -0.63       -1.15        0.57↙
-0.92       -0.98        0.22
         0.85      -0.88        0.61        0.03        0.22       -0.02↙
 1.88        1.35        0.25
        -0.57       0.56        0.37        0.18        0.17       -0.31↙
 0.42       -0.17       -0.49
        -0.86       0.76       -0.95        0.23        0.83       -0.00↙
-0.16       -0.00        0.52
         0.62      -0.88        0.05        0.79       -0.23       -0.47↙
 0.27        0.08        0.11
        -1.02       0.97       -0.88        0.74       -0.93        0.68↙
-0.74       -0.08        0.28
        -0.37       0.18       -0.33        0.22       -0.05       -0.91↙
-0.53       -0.17        0.06
        -1.87       0.14       -0.79       -0.54        0.10       -0.02↙
-0.17       -2.14       -0.51
         0.09       0.14       -0.27       -0.07        0.29       -0.41↙
-0.64        0.15        0.71
         0.51      -1.09        0.21       -0.38        0.54        0.38↙
 1.04       -1.71       -0.09
        -0.06       0.08        0.06       -0.00        0.09       -0.17↙
 0.03       -0.00        0.10
        -0.08       0.81       -0.90       -0.67        0.10       -0.85↙
 0.49       -0.39        0.62
        -0.31      -0.49       -0.06       -0.38        0.53       -0.54↙
-0.27        0.28       -0.55
        -0.65       0.92        1.06       -0.18       -0.63       -0.74↙
```

```
-1.96        0.38        0.20
          -1.62       1.28        0.70       -0.11        0.23       -0.11✓
 0.56       -0.37        0.05
          -0.50       -0.75       -0.02        0.57        1.39       -0.19✓
-1.36        0.23        0.69
           2.20        0.56        1.06       -0.19        0.95       -0.34✓
-0.95        1.85        0.81
           0.52        0.17        0.79       -0.05       -0.06       -0.21✓
-0.29        0.86        0.34
          -0.04        0.40       -0.82       -0.22       -0.43       -1.24✓
-1.11       -0.31       -0.47
           0.07       -1.03       -0.73       -0.97        0.21       -0.22✓
 1.65        1.31       -1.06
          -1.40        0.21        1.69        0.59        0.08        1.33✓
 0.24       -1.06        0.17
```

Columns 10 through 12

```
-0.71       -0.20        0.61
-0.19       -0.75       -2.37
-0.64       -0.94        0.54
 0.11        0.34        0.52
 0.43       -0.01        0.71
 4.66       -0.40       -0.53
-1.15       -1.39       -0.79
-0.40        0.88       -0.66
 0.55        0.70       -0.22
-0.68       -1.73        0.31
-1.07       -0.15        0.72
-0.64       -0.43        0.23
 0.92       -0.51       -0.82
 0.11        0.35        1.75
 0.82       -0.28       -0.90
 0.44       -0.53       -0.53
 1.17       -1.67        0.29
 2.03       -2.85        0.57
 0.23       -0.68       -0.60
-0.72       -1.65       -2.00
-0.28       -0.34        0.82
 0.41        1.68       -5.84
-0.04        1.39       -1.26
-0.29       -0.62       -0.38
-0.42       -1.48       -0.53
 0.16       -1.12        0.73
-0.75       -0.35        1.43
-1.48        0.53       -0.09
-4.60       -0.88        0.60
 0.04        0.89        0.81
 0.84       -0.65       -0.31
 0.45        0.21        0.22
-1.91       -1.12        1.12
```

```
      1.17          0.37          0.42
      0.78          0.38         -1.22
     -0.18         -0.23          0.28
     -0.71         -0.33         -0.23
      1.05         -2.14          0.18
     -0.92         -2.03          1.29
      1.25         -2.00          0.30
     -0.22          0.80          0.01
      2.03          0.64         -0.19
     -0.81         -1.25          0.82
     -0.27          0.16         -0.27
     -0.77          2.32         -0.75
     -1.02          0.93          0.05
      0.28          1.81          1.65
      0.31          0.53          0.44
     -1.88         -0.89          0.29
     -1.02          0.25         -1.26
      0.22         -0.93         -0.05
      0.63         -0.02         -0.14
      0.23          0.85         -0.42
     -0.32          0.07          0.74
     -0.57         -1.25          0.02
```

Layer weights =

ans =

  Columns 1 through 9

```
          0.02         -0.01         -0.04         -0.15          0.04          1.75
 -0.01          0.09          0.00
          0.03          0.98         -0.04         -0.23         -0.09          0.04
  0.02         -0.06         -0.01
         -0.02         -1.56          0.02          0.15          0.05          0.06
 -0.03          0.04          0.03
         -0.69         -0.15          1.44          1.35         -1.68          0.25
  0.13         -0.48         -0.27
          0.37          0.95          1.36         -0.62          1.39          0.01
  1.12          0.52          0.81
          1.03         -0.05          0.07          0.96         -1.91          0.17
 -0.64          0.55          0.16
```

  Columns 10 through 18

```
         -0.15          0.04          0.07         -0.07          0.45          0.01
  0.00         -0.11         -0.44
          0.20         -0.05         -0.02         -0.03         -1.00          0.03
  0.02         -0.08          0.20
         -0.02          0.03         -0.01         -0.03         -1.27         -0.02
 -0.02         -0.05         -0.06
          0.26         -0.25         -0.06         -0.43          0.94          0.11
```

```
 0.42        -0.86         0.77
              1.39        -0.88         0.55         0.78        -1.15        -0.37↙
-0.15        -1.50         0.83
             -0.08         0.02        -0.80         0.55        -0.62        -0.60↙
-0.92         0.70        -0.36


    Columns 19 through 27

              0.03         0.01         0.04         0.22        -0.02        -0.01↙
-0.21        -0.02         0.04
              0.01        -0.02        -0.01        -1.02         0.04         0.02↙
 0.36         0.01        -0.02
              0.00        -0.01         0.00         0.13        -0.03        -0.03↙
 0.06         0.03         0.03
             -0.25        -0.10        -0.63         0.24        -0.53         0.57↙
 0.32         0.49         0.11
             -0.96        -0.72         0.16        -0.78        -0.71         0.19↙
-1.73         0.46        -0.75
             -1.14         0.05         0.08         0.06         0.15        -0.09↙
-0.28        -0.36         0.29


    Columns 28 through 36

             -0.02         2.05        -0.03        -0.06        -0.02         0.02↙
-0.02         0.00         0.01
             -0.01         0.03        -0.03        -0.01         0.05        -0.02↙
 0.00         0.01        -0.04
              0.03         0.03        -0.07        -0.05        -0.03        -0.01↙
-0.03         0.02        -0.01
              0.04         0.21        -0.09        -0.42         0.55        -0.39↙
-0.51         0.22         0.10
             -0.47        -0.04         0.92         0.23         1.73        -0.15↙
 0.24         0.64         0.37
              0.02         0.07         0.23         0.64        -1.38        -0.43↙
-0.56         0.55         1.10


    Columns 37 through 45

              0.06        -0.00         0.04        -0.05        -0.10        -0.04↙
 0.08         0.01         0.60
             -0.07         0.06        -0.02        -0.00         0.03         0.00↙
-0.11        -0.02         0.34
              0.07        -0.02         0.02        -0.02        -0.14        -0.02↙
 0.14        -0.02        -0.29
              0.58        -0.51         0.16         0.71        -0.72        -0.05↙
-0.14        -0.27         1.21
              0.68         0.69         0.75        -0.45         1.26         0.33↙
 1.52         0.27         0.76
              1.39        -0.63        -0.77         0.66         1.24        -0.48↙
 0.19         1.25        -1.05
```

125

Columns 46 through 54

```
        -0.06        -0.04         0.09         0.07        -0.01         0.00
-0.12        -0.02         0.06
         0.03         0.04        -0.06         0.04         0.00        -0.01
 0.00         0.04        -0.04
         0.00        -0.01        -0.01         0.01        -0.01        -0.01
-0.06        -0.02         0.02
        -0.40        -0.18        -0.50         0.46        -0.67        -0.10
 1.87         1.30         0.19
         0.26        -0.93        -1.01         0.06         1.02         0.31
 1.23        -0.53        -0.97
        -0.48         0.87        -0.13         1.00        -1.31         1.37
 0.60        -0.01        -0.25
```

  Column 55

```
         -0.08
          0.05
         -0.04
         -0.15
          0.47
         -1.23
```

Input bias =

ans =

```
          3.61
         -1.69
          3.34
          1.35
          2.37
          1.74
          2.60
         -1.08
         -1.52
         -1.10
          1.50
          1.32
         -1.22
         -0.80
         -0.67
          1.33
         -0.13
          0.92
         -0.89
          1.19
         -0.01
          0.75
         -0.38
```

126

```
        -0.58
        -0.05
         0.41
         0.09
        -0.11
        -1.06
        -0.11
        -0.11
        -0.30
        -1.23
         0.48
        -0.32
        -0.34
         0.56
        -1.00
         1.33
        -0.94
        -0.59
        -0.89
         0.80
         1.38
        -0.80
        -1.13
        -1.88
        -1.99
        -2.39
        -2.36
         2.53
        -1.63
        -1.17
         1.54
        -2.85
```

Layer bias =

ans =

```
        -0.40
         0.92
        -2.05
         0.23
        -0.46
        -1.33
```

Network_Performance =

    0.0198

Training_Performance =

0.0105

Validation_Performance =

        0.0505

Testing_Performance =

        0.0638

Jacobian in Base Frame
[0.1*sin(theta1)*sin(theta2)*sin(theta3) - 0.77*cos(theta2)*sin(theta1) - 0.74*cos↙
(theta2)*sin(theta1)*sin(theta3) - 0.74*cos(theta3)*sin(theta1)*sin(theta2) - 0.1*cos↙
(theta1)*sin(theta4)*sin(theta5) - 0.15*sin(theta1) - 0.1*cos(theta2)*cos(theta3)*sin↙
(theta1) - 0.1*cos(theta2)*cos(theta5)*sin(theta1)*sin(theta3) - 0.1*cos(theta3)*cos↙
(theta5)*sin(theta1)*sin(theta2) - 0.1*cos(theta2)*cos(theta3)*cos(theta4)*sin(theta1)↙
*sin(theta5) + 0.1*cos(theta4)*sin(theta1)*sin(theta2)*sin(theta3)*sin(theta5), -0.01*cos↙
(theta1)*(77.0*sin(theta2) - 74.0*cos(theta2)*cos(theta3) + 10.0*cos(theta2)*sin(theta3)↙
+ 10.0*cos(theta3)*sin(theta2) + 74.0*sin(theta2)*sin(theta3) + 10.0*cos(theta5)*sin↙
(theta2)*sin(theta3) - 10.0*cos(theta2)*cos(theta3)*cos(theta5) + 10.0*cos(theta2)*cos↙
(theta4)*sin(theta3)*sin(theta5) + 10.0*cos(theta3)*cos(theta4)*sin(theta2)*sin(theta5)), ↙
-0.02*cos(theta1)*(5.0*cos(theta2)*sin(theta3) - 37.0*cos(theta2)*cos(theta3) + 5.0*cos↙
(theta3)*sin(theta2) + 37.0*sin(theta2)*sin(theta3) + 5.0*cos(theta5)*sin(theta2)*sin↙
(theta3) - 5.0*cos(theta2)*cos(theta3)*cos(theta5) + 5.0*cos(theta2)*cos(theta4)*sin↙
(theta3)*sin(theta5) + 5.0*cos(theta3)*cos(theta4)*sin(theta2)*sin(theta5)), -0.1*sin↙
(theta5)*(cos(theta4)*sin(theta1) + cos(theta1)*cos(theta2)*cos(theta3)*sin(theta4) - 1.0↙
*cos(theta1)*sin(theta2)*sin(theta3)*sin(theta4)), 0.1*cos(theta1)*cos(theta2)*cos↙
(theta3)*cos(theta4)*cos(theta5) - 0.1*cos(theta1)*cos(theta2)*sin(theta3)*sin(theta5) -↙
0.1*cos(theta1)*cos(theta3)*sin(theta2)*sin(theta5) - 0.1*cos(theta5)*sin(theta1)*sin↙
(theta4) - 0.1*cos(theta1)*cos(theta4)*cos(theta5)*sin(theta2)*sin(theta3), ↙
0]
[0.15*cos(theta1) + 0.77*cos(theta1)*cos(theta2) - 0.1*cos(theta1)*sin(theta2)*sin↙
(theta3) - 0.1*sin(theta1)*sin(theta4)*sin(theta5) + 0.1*cos(theta1)*cos(theta2)*cos↙
(theta3) + 0.74*cos(theta1)*cos(theta2)*sin(theta3) + 0.74*cos(theta1)*cos(theta3)*sin↙
(theta2) + 0.1*cos(theta1)*cos(theta2)*cos(theta5)*sin(theta3) + 0.1*cos(theta1)*cos↙
(theta3)*cos(theta5)*sin(theta2) + 0.1*cos(theta1)*cos(theta2)*cos(theta3)*cos(theta4)↙
*sin(theta5) - 0.1*cos(theta1)*cos(theta4)*sin(theta2)*sin(theta3)*sin(theta5), -0.01*sin↙
(theta1)*(77.0*sin(theta2) - 74.0*cos(theta2)*cos(theta3) + 10.0*cos(theta2)*sin(theta3)↙
+ 10.0*cos(theta3)*sin(theta2) + 74.0*sin(theta2)*sin(theta3) + 10.0*cos(theta5)*sin↙
(theta2)*sin(theta3) - 10.0*cos(theta2)*cos(theta3)*cos(theta5) + 10.0*cos(theta2)*cos↙
(theta4)*sin(theta3)*sin(theta5) + 10.0*cos(theta3)*cos(theta4)*sin(theta2)*sin(theta5)), ↙
-0.02*sin(theta1)*(5.0*cos(theta2)*sin(theta3) - 37.0*cos(theta2)*cos(theta3) + 5.0*cos↙
(theta3)*sin(theta2) + 37.0*sin(theta2)*sin(theta3) + 5.0*cos(theta5)*sin(theta2)*sin↙
(theta3) - 5.0*cos(theta2)*cos(theta3)*cos(theta5) + 5.0*cos(theta2)*cos(theta4)*sin↙
(theta3)*sin(theta5) + 5.0*cos(theta3)*cos(theta4)*sin(theta2)*sin(theta5)), 0.1*sin↙
(theta5)*(cos(theta1)*cos(theta4) - 1.0*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta4) +↙
sin(theta1)*sin(theta2)*sin(theta3)*sin(theta4)), 0.1*cos(theta1)*cos(theta5)*sin(theta4)↙
- 0.1*cos(theta2)*sin(theta1)*sin(theta3)*sin(theta5) - 0.1*cos(theta3)*sin(theta1)*sin↙

                                    128

(theta2)*sin(theta5) + 0.1*cos(theta2)*cos(theta3)*cos(theta4)*cos(theta5)*sin(theta1) -
0.1*cos(theta4)*cos(theta5)*sin(theta1)*sin(theta2)*sin(theta3),
0]
[
0,   0.1*sin(theta2)*sin(theta3) - 0.1*cos(theta2)*cos(theta3) -
0.74*cos(theta2)*sin(theta3) - 0.74*cos(theta3)*sin(theta2) - 0.77*cos(theta2) - 0.1*cos
(theta2)*cos(theta5)*sin(theta3) - 0.1*cos(theta3)*cos(theta5)*sin(theta2) - 0.1*cos
(theta2)*cos(theta3)*cos(theta4)*sin(theta5) + 0.1*cos(theta4)*sin(theta2)*sin(theta3)
sin(theta5), - 1.0(sin(theta6)*(cos(theta2 + theta3)*sin(theta5) + sin(theta2 + theta3)
*cos(theta4)*cos(theta5)) + sin(theta2 + theta3)*cos(theta6)*sin(theta4))*(0.1*cos
(theta4)*sin(theta6) + 0.74*cos(theta6)*sin(theta4) + sin(theta6)*(0.1*sin(theta5) + 0.74
*cos(theta4)*cos(theta5)) + 0.1*cos(theta5)*cos(theta6)*sin(theta4)) - 1.0*(cos(theta6)*
(cos(theta2 + theta3)*sin(theta5) + sin(theta2 + theta3)*cos(theta4)*cos(theta5)) - 1.0
*sin(theta2 + theta3)*sin(theta4)*sin(theta6))*(0.1*cos(theta4)*cos(theta6) - 0.74*sin
(theta4)*sin(theta6) + cos(theta6)*(0.1*sin(theta5) + 0.74*cos(theta4)*cos(theta5)) - 0.1
*cos(theta5)*sin(theta4)*sin(theta6)) - 1.0*(0.1*cos(theta5) - 0.74*cos(theta4)*sin
(theta5))*(cos(theta2 + theta3)*cos(theta5) - 1.0*sin(theta2 + theta3)*cos(theta4)*sin
(theta5)),
0.1*sin(theta2 + theta3)*sin(theta4)*sin(theta5),
- 0.1*cos(theta2 + theta3)*sin(theta5) - 0.1*sin(theta2 + theta3)*cos(theta4)*cos
(theta5),
0]
[
0,
-1.0*sin(theta1),
-1.0*sin(theta1),
sin(theta2 + theta3)*cos(theta1),
cos(theta1)*sin(theta2)*sin(theta3)*sin(theta4) - 1.0*cos(theta1)*cos(theta2)*cos(theta3)
*sin(theta4) - 1.0*cos(theta4)*sin(theta1), cos(theta5)*(cos(theta1)*cos(theta2)*sin
(theta3) + cos(theta1)*cos(theta3)*sin(theta2)) - 1.0*sin(theta5)*(sin(theta1)*sin
(theta4) + cos(theta4)*(cos(theta1)*sin(theta2)*sin(theta3) - 1.0*cos(theta1)*cos(theta2)
*cos(theta3)))]
[
0,
cos(theta1),
cos(theta1),
sin(theta2 + theta3)*sin(theta1),
cos(theta1)*cos(theta4) - 1.0*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta4) + sin
(theta1)*sin(theta2)*sin(theta3)*sin(theta4), cos(theta5)*(cos(theta2)*sin(theta1)*sin
(theta3) + cos(theta3)*sin(theta1)*sin(theta2)) + sin(theta5)*(cos(theta1)*sin(theta4) -
1.0*cos(theta4)*(sin(theta1)*sin(theta2)*sin(theta3) - 1.0*cos(theta2)*cos(theta3)*sin
(theta1)))]
[
(cos(theta2 + theta3)*cos(theta5) - 1.0*sin(theta2 + theta3)*cos(theta4)*sin(theta5))^2 +
(cos(theta6)*(cos(theta2 + theta3)*sin(theta5) + sin(theta2 + theta3)*cos(theta4)*cos
(theta5)) - 1.0*sin(theta2 + theta3)*sin(theta4)*sin(theta6))^2 + (sin(theta6)*(cos
(theta2 + theta3)*sin(theta5) + sin(theta2 + theta3)*cos(theta4)*cos(theta5)) + sin
(theta2 + theta3)*cos(theta6)*sin(theta4))^2,
0,
0,
cos(theta2 + theta3),

sin(theta2 + theta3)*sin(theta4), ✔
cos(theta2 + theta3)*cos(theta5) - 1.0*sin(theta2 + theta3)*cos(theta4)*sin(theta5)]

Jacobian Subset J11
[0.1*sin(theta1)*sin(theta2)*sin(theta3) - 0.77*cos(theta2)*sin(theta1) - 0.74*cos✔
(theta2)*sin(theta1)*sin(theta3) - 0.74*cos(theta3)*sin(theta1)*sin(theta2) - 0.1*cos✔
(theta1)*sin(theta4)*sin(theta5) - 0.15*sin(theta1) - 0.1*cos(theta2)*cos(theta3)*sin✔
(theta1) - 0.1*cos(theta2)*cos(theta5)*sin(theta1)*sin(theta3) - 0.1*cos(theta3)*cos✔
(theta5)*sin(theta1)*sin(theta2) - 0.1*cos(theta2)*cos(theta3)*cos(theta4)*sin(theta1)✔
*sin(theta5) + 0.1*cos(theta4)*sin(theta1)*sin(theta2)*sin(theta3)*sin(theta5), -0.01*cos✔
(theta1)*(77.0*sin(theta2) - 74.0*cos(theta2)*cos(theta3) + 10.0*cos(theta2)*sin(theta3)✔
+ 10.0*cos(theta3)*sin(theta2) + 74.0*sin(theta2)*sin(theta3) + 10.0*cos(theta5)*sin✔
(theta2)*sin(theta3) - 10.0*cos(theta2)*cos(theta3)*cos(theta5) + 10.0*cos(theta2)*cos✔
(theta4)*sin(theta3)*sin(theta5) + 10.0*cos(theta3)*cos(theta4)*sin(theta2)*sin(theta5)), ✔
-0.02*cos(theta1)*(5.0*cos(theta2)*sin(theta3) - 37.0*cos(theta2)*cos(theta3) + 5.0*cos✔
(theta3)*sin(theta2) + 37.0*sin(theta2)*sin(theta3) + 5.0*cos(theta5)*sin(theta2)*sin✔
(theta3) - 5.0*cos(theta2)*cos(theta3)*cos(theta5) + 5.0*cos(theta2)*cos(theta4)*sin✔
(theta3)*sin(theta5) + 5.0*cos(theta3)*cos(theta4)*sin(theta2)*sin(theta5))]
[0.15*cos(theta1) + 0.77*cos(theta1)*cos(theta2) - 0.1*cos(theta1)*sin(theta2)*sin✔
(theta3) - 0.1*sin(theta1)*sin(theta4)*sin(theta5) + 0.1*cos(theta1)*cos(theta2)*cos✔
(theta3) + 0.74*cos(theta1)*cos(theta2)*sin(theta3) + 0.74*cos(theta1)*cos(theta3)*sin✔
(theta2) + 0.1*cos(theta1)*cos(theta2)*cos(theta5)*sin(theta3) + 0.1*cos(theta1)*cos✔
(theta3)*cos(theta5)*sin(theta2) + 0.1*cos(theta1)*cos(theta2)*cos(theta3)*cos(theta4)✔
*sin(theta5) - 0.1*cos(theta1)*cos(theta4)*sin(theta2)*sin(theta3)*sin(theta5), -0.01*sin✔
(theta1)*(77.0*sin(theta2) - 74.0*cos(theta2)*cos(theta3) + 10.0*cos(theta2)*sin(theta3)✔
+ 10.0*cos(theta3)*sin(theta2) + 74.0*sin(theta2)*sin(theta3) + 10.0*cos(theta5)*sin✔
(theta2)*sin(theta3) - 10.0*cos(theta2)*cos(theta3)*cos(theta5) + 10.0*cos(theta2)*cos✔
(theta4)*sin(theta3)*sin(theta5) + 10.0*cos(theta3)*cos(theta4)*sin(theta2)*sin(theta5)), ✔
-0.02*sin(theta1)*(5.0*cos(theta2)*sin(theta3) - 37.0*cos(theta2)*cos(theta3) + 5.0*cos✔
(theta3)*sin(theta2) + 37.0*sin(theta2)*sin(theta3) + 5.0*cos(theta5)*sin(theta2)*sin✔
(theta3) - 5.0*cos(theta2)*cos(theta3)*cos(theta5) + 5.0*cos(theta2)*cos(theta4)*sin✔
(theta3)*sin(theta5) + 5.0*cos(theta3)*cos(theta4)*sin(theta2)*sin(theta5))]
[✔
0,   0.1*sin(theta2)*sin(theta3) - 0.1*cos(theta2)*cos(theta3) -✔
0.74*cos(theta2)*sin(theta3) - 0.74*cos(theta3)*sin(theta2) - 0.77*cos(theta2) - 0.1*cos✔
(theta2)*cos(theta5)*sin(theta3) - 0.1*cos(theta3)*cos(theta5)*sin(theta2) - 0.1*cos✔
(theta2)*cos(theta3)*cos(theta4)*sin(theta5) + 0.1*cos(theta4)*sin(theta2)*sin(theta3)✔
sin(theta5), - 1.0(sin(theta6)*(cos(theta2 + theta3)*sin(theta5) + sin(theta2 + theta3)✔
*cos(theta4)*cos(theta5)) + sin(theta2 + theta3)*cos(theta6)*sin(theta4))*(0.1*cos✔
(theta4)*sin(theta6) + 0.74*cos(theta6)*sin(theta4) + sin(theta6)*(0.1*sin(theta5) + 0.74✔
*cos(theta4)*cos(theta5)) + 0.1*cos(theta5)*cos(theta6)*sin(theta4)) - 1.0*(cos(theta6)*✔
(cos(theta2 + theta3)*sin(theta5) + sin(theta2 + theta3)*cos(theta4)*cos(theta5)) - 1.0✔
*sin(theta2 + theta3)*sin(theta4)*sin(theta6))*(0.1*cos(theta4)*cos(theta6) - 0.74*sin✔
(theta4)*sin(theta6) + cos(theta6)*(0.1*sin(theta5) + 0.74*cos(theta4)*cos(theta5)) - 0.1✔
*cos(theta5)*sin(theta4)*sin(theta6)) - 1.0*(0.1*cos(theta5) - 0.74*cos(theta4)*sin✔
(theta5))*(cos(theta2 + theta3)*cos(theta5) - 1.0*sin(theta2 + theta3)*cos(theta4)*sin✔
(theta5))]

Jacobian Subset J22
[sin(theta2 + theta3)*cos(theta1), cos(theta1)*sin(theta2)*sin(theta3)*sin(theta4) - 1.0✔
*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta4) - 1.0*cos(theta4)*sin(theta1), cos✔

130

(theta5)*(cos(theta1)*cos(theta2)*sin(theta3) + cos(theta1)*cos(theta3)*sin(theta2)) -↙
1.0*sin(theta5)*(sin(theta1)*sin(theta4) + cos(theta4)*(cos(theta1)*sin(theta2)*sin↙
(theta3) - 1.0*cos(theta1)*cos(theta2)*cos(theta3)))]
[sin(theta2 + theta3)*sin(theta1),   cos(theta1)*cos(theta4) - 1.0*cos(theta2)*cos↙
(theta3)*sin(theta1)*sin(theta4) + sin(theta1)*sin(theta2)*sin(theta3)*sin(theta4), cos↙
(theta5)*(cos(theta2)*sin(theta1)*sin(theta3) + cos(theta3)*sin(theta1)*sin(theta2)) +↙
sin(theta5)*(cos(theta1)*sin(theta4) - 1.0*cos(theta4)*(sin(theta1)*sin(theta2)*sin↙
(theta3) - 1.0*cos(theta2)*cos(theta3)*sin(theta1)))]
[cos(theta2 + theta3),↙
sin(theta2 + theta3)*sin(theta4),↙
cos(theta2 + theta3)*cos(theta5) - 1.0*sin(theta2 + theta3)*cos(theta4)*sin(theta5)]

Singularity Equation
The Robot has a Wrist Configuration
Forearm Singularity Equation
(2849*cos(theta2))/50000 - (8547*cos(theta3))/100000 - (77*sin(theta2))/10000 + (231*sin↙
(theta3))/20000 - (219373*cos(theta2)*cos(theta3))/500000 + (77*cos(theta2)*cos(theta5))↙
/10000 - (231*cos(theta3)*cos(theta5))/20000 + (5929*cos(theta2)*sin(theta3))/100000 -↙
(2849*cos(theta2)*cos(theta3)^2)/25000 - (6468*cos(theta3)^2*sin(theta2))/15625 - (77*cos↙
(theta4)^2*sin(theta2))/10000 + (2849*cos(theta3)*sin(theta2)*sin(theta3))/25000 -↙
(77*cos(theta4)*sin(theta2)*sin(theta5))/5000 + (231*cos(theta4)*sin(theta3)*sin(theta5))↙
/20000 - (77*cos(theta2)*cos(theta3)^2*cos(theta5))/5000 - (2849*cos(theta3)^2*cos↙
(theta5)*sin(theta2))/25000 + (77*cos(theta3)^2*cos(theta4)^2*sin(theta2))/10000 -↙
(77*cos(theta3)^2*cos(theta5)^2*sin(theta2))/10000 + (77*cos(theta4)^2*cos(theta5)^2*sin↙
(theta2))/10000 - (5929*cos(theta2)*cos(theta3)*cos(theta5))/100000 - (6468*cos(theta2)↙
*cos(theta3)*sin(theta3))/15625 + (2849*cos(theta2)*cos(theta4)*sin(theta5))/50000 +↙
(77*cos(theta2)*cos(theta3)*cos(theta4)^2*sin(theta3))/10000 - (77*cos(theta2)*cos↙
(theta3)*cos(theta5)^2*sin(theta3))/10000 - (2849*cos(theta2)*cos(theta3)^2*cos(theta4)↙
*sin(theta5))/25000 + (77*cos(theta3)^2*cos(theta4)*sin(theta2)*sin(theta5))/5000 -↙
(2849*cos(theta2)*cos(theta3)*cos(theta5)*sin(theta3))/25000 + (77*cos(theta2)*cos↙
(theta4)*cos(theta5)*sin(theta5))/10000 + (77*cos(theta3)*cos(theta5)*sin(theta2)*sin↙
(theta3))/5000 + (5929*cos(theta2)*cos(theta4)*sin(theta3)*sin(theta5))/100000 - (77*cos↙
(theta3)^2*cos(theta4)^2*cos(theta5)^2*sin(theta2))/10000 - (77*cos(theta2)*cos(theta3)↙
^2*cos(theta4)*cos(theta5)*sin(theta5))/5000 - (77*cos(theta2)*cos(theta3)*cos(theta4)↙
^2*cos(theta5)^2*sin(theta3))/10000 + (77*cos(theta2)*cos(theta3)*cos(theta4)*sin(theta3)↙
*sin(theta5))/5000 + (2849*cos(theta3)*cos(theta4)*sin(theta2)*sin(theta3)*sin(theta5))↙
/25000 + (77*cos(theta3)*cos(theta4)*cos(theta5)*sin(theta2)*sin(theta3)*sin(theta5))↙
/5000

Warning: 1 equations in 6 variables. New variables might be introduced.
> In C:\Program Files\MATLAB\R2013a\toolbox\symbolic\symbolic\symengine.p>symengine at 56
  In mupadengine.mupadengine>mupadengine.evalin at 97
  In mupadengine.mupadengine>mupadengine.feval at 150
  In solve at 172
Forearm Singularity Solution(s)

SE1 =

[z, 2*atan((50*cos(z2)*sin(z3) - 370*cos(z1)*sin(z1) + 1344*cos(z1)^2 + 25*cos(z2)^2 +↙
370*cos(z1)^2*cos(z3) - 25*cos(z1)^2*cos(z2)^2 + 25*cos(z1)^2*cos(z3)^2 - 25*cos(z2)↙
^2*cos(z3)^2 + (9250*cos(z3) - 527065*cos(z1) + 71225*sin(z1) - 142450*cos(z1)*cos(z3) -↙

131

1043400*cos(z1)*sin(z1) + 9625*cos(z3)*sin(z1) + 70950*cos(z2)*sin(z3) + 1882494*cos(z1)^2 + 1054130*cos(z1)^3 + 1250*cos(z2)^2 + 1943236*cos(z1)^4 + 625*cos(z3)^2 + 625*cos(z2)^4 + 35650*sin(z1)^2 - 9625*cos(z1)*cos(z3)^2 + 509120*cos(z1)^2*cos(z3) + 284900*cos(z1)^3*cos(z3) + 1031560*cos(z1)^4*cos(z3) - 517440*cos(z1)*sin(z1)^2 + 3686606*cos(z1)^2*sin(z1) + 2500*cos(z2)^3*sin(z3) + 65950*cos(z1)^2*cos(z2)^2 + 34400*cos(z1)^2*cos(z3)^2 - 1250*cos(z1)^2*cos(z2)^4 + 19250*cos(z1)^3*cos(z3)^2 - 1250*cos(z2)^2*cos(z3)^2 - 67200*cos(z1)^4*cos(z2)^2 + 206600*cos(z1)^4*cos(z3)^2 + 625*cos(z1)^4*cos(z2)^4 + 18500*cos(z1)^4*cos(z3)^3 - 1250*cos(z2)^4*cos(z3)^2 + 625*cos(z1)^4*cos(z3)^4 + 625*cos(z2)^4*cos(z3)^4 + 1943236*cos(z1)^2*sin(z1)^2 + 36725*cos(z2)^2*sin(z3)^2 - 64700*cos(z1)^2*cos(z2)^2*cos(z3)^2 - 18500*cos(z1)^2*cos(z2)^2*cos(z3)^3 - 1250*cos(z1)^2*cos(z2)^2*cos(z3)^4 + 2500*cos(z1)^2*cos(z2)^4*cos(z3)^2 + 65950*cos(z1)^4*cos(z2)^2*cos(z3)^2 + 18500*cos(z1)^4*cos(z2)^2*cos(z3)^3 - 1250*cos(z1)^2*cos(z2)^4*cos(z3)^4 + 1250*cos(z1)^4*cos(z2)^2*cos(z3)^4 - 1250*cos(z1)^4*cos(z2)^4*cos(z3)^2 + 625*cos(z1)^4*cos(z2)^4*cos(z3)^4 + 142450*cos(z2)*sin(z1)*sin(z3) - 67200*cos(z1)^2*cos(z2)^2*sin(z1)^2 + 206600*cos(z1)^2*cos(z3)^2*sin(z1)^2 - 141900*cos(z1)^2*cos(z2)^2*sin(z3)^2 + 625*cos(z1)^2*cos(z2)^4*sin(z1)^2 + 18500*cos(z1)^2*cos(z3)^3*sin(z1)^2 + 625*cos(z1)^2*cos(z3)^4*sin(z1)^2 + 625*cos(z2)^2*cos(z3)^2*sin(z3)^2 + 139400*cos(z1)^4*cos(z2)^2*sin(z3)^2 + 35650*cos(z2)^2*sin(z1)^2*sin(z3)^2 - 9250*cos(z1)*cos(z2)^2*sin(z1) - 142450*cos(z1)*cos(z3)*sin(z1)^2 - 27750*cos(z1)*cos(z3)^2*sin(z1) + 1552320*cos(z1)^2*cos(z3)*sin(z1) - 1250*cos(z1)*cos(z3)^3*sin(z1) - 141900*cos(z1)^2*cos(z2)*sin(z3) + 1054130*cos(z1)^3*cos(z2)*sin(z3) + 1250*cos(z2)*cos(z3)^2*sin(z3) + 139400*cos(z1)^4*cos(z2)*sin(z3) + 71300*cos(z2)*sin(z1)^2*sin(z3) + 18500*cos(z1)^2*cos(z2)^2*cos(z3) - 18500*cos(z1)^4*cos(z2)^2*cos(z3) + 9625*cos(z1)*cos(z2)^2*sin(z1)^2 - 71225*cos(z1)^2*cos(z2)^2*sin(z1) - 9625*cos(z1)*cos(z3)^2*sin(z1)^2 + 1031560*cos(z1)^2*cos(z3)*sin(z1)^2 + 213675*cos(z1)^2*cos(z3)^2*sin(z1) + 9625*cos(z1)^2*cos(z3)^3*sin(z1) - 5000*cos(z1)^2*cos(z2)^3*sin(z3) + 9250*cos(z2)^2*cos(z3)*sin(z3)^2 + 2500*cos(z1)^4*cos(z2)^3*sin(z3) - 2500*cos(z2)^3*cos(z3)^2*sin(z3) + 71225*cos(z2)^2*sin(z1)*sin(z3)^2 - 277900*cos(z1)*cos(z3)*sin(z1) - 527065*cos(z1)*cos(z2)*sin(z3) + 18500*cos(z2)*cos(z3)*sin(z3) - 1250*cos(z1)*cos(z2)^2*cos(z3)*sin(z1) - 9625*cos(z1)*cos(z2)*cos(z3)^2*sin(z3) - 37000*cos(z1)^2*cos(z2)*cos(z3)*sin(z3) + 284900*cos(z1)^3*cos(z2)*cos(z3)*sin(z3) + 37000*cos(z1)^4*cos(z2)*cos(z3)*sin(z3) - 498190*cos(z1)*cos(z2)*sin(z1)^2*sin(z3) - 427350*cos(z1)^2*cos(z2)*sin(z1)*sin(z3) - 9250*cos(z1)*cos(z2)^3*sin(z1)*sin(z3) + 65950*cos(z1)^2*cos(z2)^2*cos(z3)^2*sin(z1)^2 + 18500*cos(z1)^2*cos(z2)^2*cos(z3)^3*sin(z1)^2 - 2500*cos(z1)^2*cos(z2)^2*cos(z3)^2*sin(z3)^2 + 1250*cos(z1)^2*cos(z2)^2*cos(z3)^4*sin(z1)^2 - 1250*cos(z1)^2*cos(z2)^4*cos(z3)^2*sin(z1)^2 + 625*cos(z1)^2*cos(z2)^4*cos(z3)^4*sin(z1)^2 + 2500*cos(z1)^4*cos(z2)^2*cos(z3)^2*sin(z3)^2 + 139400*cos(z1)^2*cos(z2)^2*sin(z1)^2*sin(z3)^2 + 9250*cos(z1)*cos(z2)^2*cos(z3)^2*sin(z1) - 9625*cos(z1)^2*cos(z2)^2*cos(z3)*sin(z1) + 1250*cos(z1)*cos(z2)^2*cos(z3)^3*sin(z1) - 2500*cos(z1)^2*cos(z2)*cos(z3)^2*sin(z3) + 19250*cos(z1)^3*cos(z2)*cos(z3)^2*sin(z3) + 2500*cos(z1)^4*cos(z2)*cos(z3)^2*sin(z3) - 18500*cos(z1)*cos(z2)^2*sin(z1)*sin(z3)^2 + 139400*cos(z1)^2*cos(z2)*sin(z1)^2*sin(z3) + 9625*cos(z1)*cos(z2)^3*sin(z1)^2*sin(z3) + 9625*cos(z2)^2*cos(z3)*sin(z1)*sin(z3)^2 - 142450*cos(z1)*cos(z2)*cos(z3)*sin(z3) - 1061900*cos(z1)*cos(z2)*sin(z1)*sin(z3) + 19250*cos(z2)*cos(z3)*sin(z1)*sin(z3) - 9625*cos(z1)*cos(z2)^2*cos(z3)^2*sin(z1)^2 - 18500*cos(z1)^2*cos(z2)^2*cos(z3)*sin(z1)^2 + 71225*cos(z1)^2*cos(z2)^2*cos(z3)^2*sin(z1) + 9625*cos(z1)^2*cos(z2)^2*cos(z3)^3*sin(z1) - 37000*cos(z1)^2*cos(z2)^2*cos(z3)*sin(z3)^2 + 5000*cos(z1)^2*cos(z2)^3*cos(z3)^2*sin(z3) + 37000*cos(z1)^4*cos(z2)^2*cos(z3)*sin(z3)^2 - 2500*cos(z1)^4*cos(z2)^3*cos(z3)^2*sin(z3) + 19250*cos(z1)*cos(z2)^2*sin(z1)^2*sin(z3)^2 - 142450*cos(z1)^2*cos(z2)^2*sin(z1)*sin(z3)^2 + 2500*cos(z1)^2*cos(z2)^3*sin(z1)^2*sin(z3) - 142450*cos(z1)*cos(z2)*cos(z3)*sin(z1)^2*sin(z3) - 27750*cos(z1)*cos(z2)*cos(z3)^2*sin(z1)*sin(z3) - 57750*cos(z1)^2*cos(z2)*cos(z3)*sin(z1)*sin(z3) - 1250*cos(z1)*cos(z2)*cos(z3)^3*sin(z1)*sin(z3) - 1250*cos(z1)*cos(z2)^3*cos(z3)*sin(z1)

```
*sin(z3) + 37000*cos(z1)^2*cos(z2)^2*cos(z3)*sin(z1)^2*sin(z3)^2 - 2500*cos(z1)^2*cos(z2)
^3*cos(z3)^2*sin(z1)^2*sin(z3) - 9625*cos(z1)*cos(z2)*cos(z3)^2*sin(z1)^2*sin(z3) -
2500*cos(z1)*cos(z2)^2*cos(z3)*sin(z1)*sin(z3)^2 + 37000*cos(z1)^2*cos(z2)*cos(z3)*sin
(z1)^2*sin(z3) + 9250*cos(z1)*cos(z2)^3*cos(z3)^2*sin(z1)*sin(z3) + 1250*cos(z1)*cos(z2)
^3*cos(z3)^3*sin(z1)*sin(z3) + 2500*cos(z1)^2*cos(z2)^2*cos(z3)^2*sin(z1)^2*sin(z3)^2 -
280400*cos(z1)*cos(z2)*cos(z3)*sin(z1)*sin(z3) + 2500*cos(z1)^2*cos(z2)*cos(z3)^2*sin(z1)
^2*sin(z3) - 19250*cos(z1)^2*cos(z2)^2*cos(z3)*sin(z1)*sin(z3)^2 - 9625*cos(z1)*cos(z2)
^3*cos(z3)^2*sin(z1)^2*sin(z3) + 34850)^(1/2) + 25*cos(z1)^2*cos(z2)^2*cos(z3)^2 - 50*cos
(z1)^2*cos(z2)*sin(z3) - 50*cos(z1)*cos(z3)*sin(z1) - 370*cos(z1)*cos(z2)*sin(z1)*sin(z3)
- 50*cos(z1)*cos(z2)*cos(z3)*sin(z1)*sin(z3) + 25)/(1147*cos(z1) - 25*cos(z3) - 155*sin
(z1) + 155*cos(z1)*cos(z3) + 1344*cos(z1)*sin(z1) - 185*cos(z2)*sin(z3) + 370*cos(z1)^2 +
50*cos(z1)^2*cos(z3) - 155*cos(z2)*sin(z1)*sin(z3) - 25*cos(z1)*cos(z2)^2*sin(z1) +
25*cos(z1)*cos(z3)^2*sin(z1) + 370*cos(z1)^2*cos(z2)*sin(z3) + 370*cos(z1)*cos(z3)*sin
(z1) - 25*cos(z2)*cos(z3)*sin(z3) + 50*cos(z1)^2*cos(z2)*cos(z3)*sin(z3) + 25*cos(z1)*cos
(z2)^2*cos(z3)^2*sin(z1) - 50*cos(z1)*cos(z2)*sin(z1)*sin(z3) - 185)), z1, z2, z3, z4]
[ z, -2*atan((370*cos(z1)*sin(z1) - 50*cos(z2)*sin(z3) - 1344*cos(z1)^2 - 25*cos(z2)^2 -
370*cos(z1)^2*cos(z3) + 25*cos(z1)^2*cos(z2)^2 - 25*cos(z1)^2*cos(z3)^2 + 25*cos(z2)
^2*cos(z3)^2 + (9250*cos(z3) - 527065*cos(z1) + 71225*sin(z1) - 142450*cos(z1)*cos(z3) -
1043400*cos(z1)*sin(z1) + 9625*cos(z3)*sin(z1) + 70950*cos(z2)*sin(z3) + 1882494*cos(z1)
^2 + 1054130*cos(z1)^3 + 1250*cos(z2)^2 + 1943236*cos(z1)^4 + 625*cos(z3)^2 + 625*cos(z2)
^4 + 35650*sin(z1)^2 - 9625*cos(z1)*cos(z3)^2 + 509120*cos(z1)^2*cos(z3) + 284900*cos(z1)
^3*cos(z3) + 1031560*cos(z1)^4*cos(z3) - 517440*cos(z1)*sin(z1)^2 + 3686606*cos(z1)^2*sin
(z1) + 2500*cos(z2)^3*sin(z3) + 65950*cos(z1)^2*cos(z2)^2 + 34400*cos(z1)^2*cos(z3)^2 -
1250*cos(z1)^2*cos(z2)^4 + 19250*cos(z1)^3*cos(z3)^2 - 1250*cos(z2)^2*cos(z3)^2 -
67200*cos(z1)^4*cos(z2)^2 + 206600*cos(z1)^4*cos(z3)^2 + 625*cos(z1)^4*cos(z2)^4 +
18500*cos(z1)^4*cos(z3)^3 - 1250*cos(z2)^4*cos(z3)^2 + 625*cos(z1)^4*cos(z3)^4 + 625*cos
(z2)^4*cos(z3)^4 + 1943236*cos(z1)^2*sin(z1)^2 + 36725*cos(z2)^2*sin(z3)^2 - 64700*cos
(z1)^2*cos(z2)^2*cos(z3)^2 - 18500*cos(z1)^2*cos(z2)^2*cos(z3)^3 - 1250*cos(z1)^2*cos(z2)
^2*cos(z3)^4 + 2500*cos(z1)^2*cos(z2)^4*cos(z3)^2 + 65950*cos(z1)^4*cos(z2)^2*cos(z3)^2 +
18500*cos(z1)^4*cos(z2)^2*cos(z3)^3 - 1250*cos(z1)^2*cos(z2)^4*cos(z3)^4 + 1250*cos(z1)
^4*cos(z2)^2*cos(z3)^4 - 1250*cos(z1)^4*cos(z2)^4*cos(z3)^2 + 625*cos(z1)^4*cos(z2)^4*cos
(z3)^4 + 142450*cos(z2)*sin(z1)*sin(z3) - 67200*cos(z1)^2*cos(z2)^2*sin(z1)^2 +
206600*cos(z1)^2*cos(z3)^2*sin(z1)^2 - 141900*cos(z1)^2*cos(z2)^2*sin(z3)^2 + 625*cos(z1)
^2*cos(z2)^4*sin(z1)^2 + 18500*cos(z1)^2*cos(z3)^3*sin(z1)^2 + 625*cos(z1)^2*cos(z3)
^4*sin(z1)^2 + 625*cos(z2)^2*cos(z3)^2*sin(z3)^2 + 139400*cos(z1)^4*cos(z2)^2*sin(z3)^2 +
35650*cos(z2)^2*sin(z1)^2*sin(z3)^2 - 9250*cos(z1)*cos(z2)^2*sin(z1) - 142450*cos(z1)*cos
(z3)*sin(z1)^2 - 27750*cos(z1)*cos(z3)^2*sin(z1) + 1552320*cos(z1)^2*cos(z3)*sin(z1) -
1250*cos(z1)*cos(z3)^3*sin(z1) - 141900*cos(z1)^2*cos(z2)*sin(z3) + 1054130*cos(z1)^3*cos
(z2)*sin(z3) + 1250*cos(z2)*cos(z3)^2*sin(z3) + 139400*cos(z1)^4*cos(z2)*sin(z3) +
71300*cos(z2)*sin(z1)^2*sin(z3) + 18500*cos(z1)^2*cos(z2)^2*cos(z3) - 18500*cos(z1)^4*cos
(z2)^2*cos(z3) + 9625*cos(z1)*cos(z2)^2*sin(z1)^2 - 71225*cos(z1)^2*cos(z2)^2*sin(z1) -
9625*cos(z1)*cos(z3)^2*sin(z1)^2 + 1031560*cos(z1)^2*cos(z3)*sin(z1)^2 + 213675*cos(z1)
^2*cos(z3)^2*sin(z1) + 9625*cos(z1)^2*cos(z3)^3*sin(z1) - 5000*cos(z1)^2*cos(z2)^3*sin
(z3) + 9250*cos(z2)^2*cos(z3)*sin(z3)^2 + 2500*cos(z1)^4*cos(z2)^3*sin(z3) - 2500*cos(z2)
^3*cos(z3)^2*sin(z3) + 71225*cos(z2)^2*sin(z1)*sin(z3)^2 - 277900*cos(z1)*cos(z3)*sin(z1)
- 527065*cos(z1)*cos(z2)*sin(z3) + 18500*cos(z2)*cos(z3)*sin(z3) - 1250*cos(z1)*cos(z2)
^2*cos(z3)*sin(z1) - 9625*cos(z1)*cos(z2)*cos(z3)^2*sin(z3) - 37000*cos(z1)^2*cos(z2)*cos
(z3)*sin(z3) + 284900*cos(z1)^3*cos(z2)*cos(z3)*sin(z3) + 37000*cos(z1)^4*cos(z2)*cos(z3)
*sin(z3) - 498190*cos(z1)*cos(z2)*sin(z1)^2*sin(z3) - 427350*cos(z1)^2*cos(z2)*sin(z1)
*sin(z3) - 9250*cos(z1)*cos(z2)^3*sin(z1)*sin(z3) + 65950*cos(z1)^2*cos(z2)^2*cos(z3)
^2*sin(z1)^2 + 18500*cos(z1)^2*cos(z2)^2*cos(z3)^3*sin(z1)^2 - 2500*cos(z1)^2*cos(z2)
```

^2*cos(z3)^2*sin(z3)^2 + 1250*cos(z1)^2*cos(z2)^2*cos(z3)^4*sin(z1)^2 - 1250*cos(z1)
^2*cos(z2)^4*cos(z3)^2*sin(z1)^2 + 625*cos(z1)^2*cos(z2)^4*cos(z3)^4*sin(z1)^2 + 2500*cos
(z1)^4*cos(z2)^2*cos(z3)^2*sin(z3)^2 + 139400*cos(z1)^2*cos(z2)^2*sin(z1)^2*sin(z3)^2 +
9250*cos(z1)*cos(z2)^2*cos(z3)^2*sin(z1) - 9625*cos(z1)^2*cos(z2)^2*cos(z3)*sin(z1) +
1250*cos(z1)*cos(z2)^2*cos(z3)^3*sin(z1) - 2500*cos(z1)^2*cos(z2)*cos(z3)^2*sin(z3) +
19250*cos(z1)^3*cos(z2)*cos(z3)^2*sin(z3) + 2500*cos(z1)^4*cos(z2)*cos(z3)^2*sin(z3) -
18500*cos(z1)*cos(z2)^2*sin(z1)*sin(z3)^2 + 139400*cos(z1)^2*cos(z2)*sin(z1)^2*sin(z3) +
9625*cos(z1)*cos(z2)^3*sin(z1)^2*sin(z3) + 9625*cos(z2)^2*cos(z3)*sin(z1)*sin(z3)^2 -
142450*cos(z1)*cos(z2)*cos(z3)*sin(z3) - 1061900*cos(z1)*cos(z2)*sin(z1)*sin(z3) +
19250*cos(z2)*cos(z3)*sin(z1)*sin(z3) - 9625*cos(z1)*cos(z2)^2*cos(z3)^2*sin(z1)^2 -
18500*cos(z1)^2*cos(z2)^2*cos(z3)*sin(z1)^2 + 71225*cos(z1)^2*cos(z2)^2*cos(z3)^2*sin(z1)
+ 9625*cos(z1)^2*cos(z2)^2*cos(z3)^3*sin(z1) - 37000*cos(z1)^2*cos(z2)^2*cos(z3)*sin(z3)
^2 + 5000*cos(z1)^2*cos(z2)^3*cos(z3)^2*sin(z3) + 37000*cos(z1)^4*cos(z2)^2*cos(z3)*sin
(z3)^2 - 2500*cos(z1)^4*cos(z2)^3*cos(z3)^2*sin(z3) + 19250*cos(z1)*cos(z2)^2*sin(z1)
^2*sin(z3)^2 - 142450*cos(z1)^2*cos(z2)^2*sin(z1)*sin(z3)^2 + 2500*cos(z1)^2*cos(z2)
^3*sin(z1)^2*sin(z3) - 142450*cos(z1)*cos(z2)*cos(z3)*sin(z1)^2*sin(z3) - 27750*cos(z1)
*cos(z2)*cos(z3)^2*sin(z1)*sin(z3) - 57750*cos(z1)^2*cos(z2)*cos(z3)*sin(z1)*sin(z3) -
1250*cos(z1)*cos(z2)*cos(z3)^3*sin(z1)*sin(z3) - 1250*cos(z1)*cos(z2)^3*cos(z3)*sin(z1)
*sin(z3) + 37000*cos(z1)^2*cos(z2)^2*cos(z3)*sin(z1)^2*sin(z3)^2 - 2500*cos(z1)^2*cos(z2)
^3*cos(z3)^2*sin(z1)^2*sin(z3) - 9625*cos(z1)*cos(z2)*cos(z3)^2*sin(z1)^2*sin(z3) -
2500*cos(z1)*cos(z2)^2*cos(z3)*sin(z1)*sin(z3)^2 + 37000*cos(z1)^2*cos(z2)*cos(z3)*sin
(z1)^2*sin(z3) + 9250*cos(z1)*cos(z2)^3*cos(z3)^2*sin(z1)*sin(z3) + 1250*cos(z1)*cos(z2)
^3*cos(z3)^3*sin(z1)*sin(z3) + 2500*cos(z1)^2*cos(z2)^2*cos(z3)^2*sin(z1)^2*sin(z3)^2 -
280400*cos(z1)*cos(z2)*cos(z3)*sin(z1)*sin(z3) + 2500*cos(z1)^2*cos(z2)*cos(z3)^2*sin(z1)
^2*sin(z3) - 19250*cos(z1)^2*cos(z2)^2*cos(z3)*sin(z1)*sin(z3)^2 - 9625*cos(z1)*cos(z2)
^3*cos(z3)^2*sin(z1)^2*sin(z3) + 34850)^(1/2) - 25*cos(z1)^2*cos(z2)^2*cos(z3)^2 + 50*cos
(z1)^2*cos(z2)*sin(z3) + 50*cos(z1)*cos(z3)*sin(z1) + 370*cos(z1)*cos(z2)*sin(z1)*sin(z3)
+ 50*cos(z1)*cos(z2)*cos(z3)*sin(z1)*sin(z3) - 25)/(1147*cos(z1) - 25*cos(z3) - 155*sin
(z1) + 155*cos(z1)*cos(z3) + 1344*cos(z1)*sin(z1) - 185*cos(z2)*sin(z3) + 370*cos(z1)^2 +
50*cos(z1)^2*cos(z3) - 155*cos(z2)*sin(z1)*sin(z3) - 25*cos(z1)*cos(z2)^2*sin(z1) +
25*cos(z1)*cos(z3)^2*sin(z1) + 370*cos(z1)^2*cos(z2)*sin(z3) + 370*cos(z1)*cos(z3)*sin
(z1) - 25*cos(z2)*cos(z3)*sin(z3) + 50*cos(z1)^2*cos(z2)*cos(z3)*sin(z3) + 25*cos(z1)*cos
(z2)^2*cos(z3)^2*sin(z1) - 50*cos(z1)*cos(z2)*sin(z1)*sin(z3) - 185)), z1, z2, z3, z4]

Wrist Singularity Equation
-sin(theta5)

Wrist Singularity Solution(s)

134

```
SE2 =

[ v8, u8, y8, x664, 0, z]

Absolute Errors in Joint Space
Max Error in Joint 1 = 0.12    Min Error in Joint 1 = 0.00039
Max Error in Joint 2 = 0.14    Min Error in Joint 2 = 0.0024
Max Error in Joint 3 = 0.17    Min Error in Joint 3 = 0.004
Max Error in Joint 4 = 2.2     Min Error in Joint 4 = 0.028
Max Error in Joint 5 = 1.5     Min Error in Joint 5 = 0.004
Max Error in Joint 6 = 4.4     Min Error in Joint 6 = 0.082
Absolute Errors in Cartesian Space
Max Error in x-x     = 1.5     Min Error in x-x     = 0.029
Max Error in x-y     = 1.6     Min Error in x-y     = 0.028
Max Error in x-z     = 0.63    Min Error in x-z     = 0.00058
Max Error in y-x     = 1.6     Min Error in y-x     = 0.037
Max Error in y-y     = 1.5     Min Error in y-y     = 0.084
Max Error in y-z     = 1.3     Min Error in y-z     = 0.0022
Max Error in z-x     = 1.4     Min Error in z-x     = 0.029
Max Error in z-y     = 1.1     Min Error in z-y     = 0.0011
Max Error in z-z     = 1.9     Min Error in z-z     = 0.005
Max Error in x       = 0.06    Min Error in x       = 0.00022
Max Error in y       = 2.1     Min Error in y       = 0.00035
Max Error in z       = 2.6     Min Error in z       = 0.022
```

Appendix C: Results for CNC Manipulator
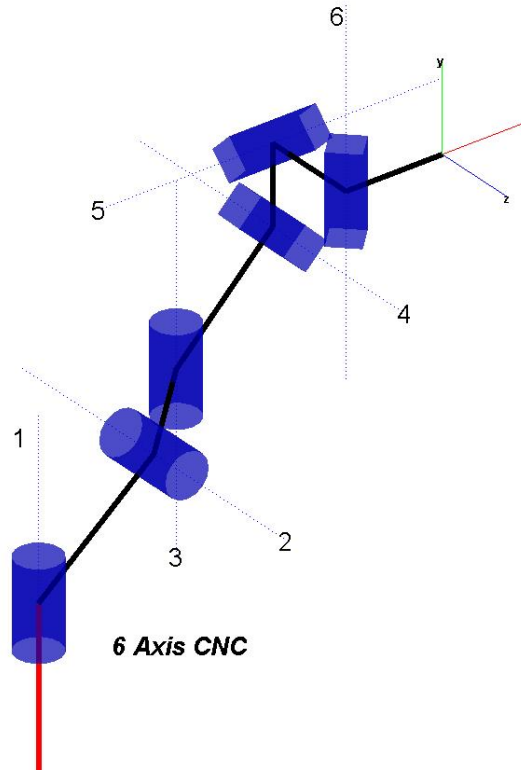
Figure 62: Kinematic Model of 6 Axis CNC Machine

Table 19: D-H Parameters of CNC Manipulator

Joint	Link Offset (m)		Joint Angle (rad)		Link Length (m)		Twist Angle (rad)		Lower Joint Limit	Upper Joint Limit
Robot: 6 Axis CNC										
1	$d_1 =$	0.5	$\theta_1 =$	θ_1	$a_1 =$	0.6	$\alpha_1 =$	-pi/2	-1.74	1.74
2	$d_2 =$	0.5	$\theta_2 =$	θ_2	$a_2 =$	0.5	$\alpha_2 =$	pi/2	-1.74	1.74
3	$d_3 =$	0.5	$\theta_3 =$	θ_3	$a_3 =$	0.5	$\alpha_3 =$	-pi/2	-1.74	1.74
4	$d_4 =$	d_4	$\theta_4 =$	-pi/2	$a_4 =$	0.4	$\alpha_4 =$	-pi/2	-0.4	0.4
5	$d_5 =$	d_5	$\theta_5 =$	pi/2	$a_5 =$	0.5	$\alpha_5 =$	-pi/2	-0.3	0.3
6	$d_6 =$	d_6	$\theta_6 =$	-pi/2	$a_6 =$	0.5	$\alpha_6 =$	-pi/2	-0.4	0.4

Table 20: CNC Manipulator Joint Angle Range for Workspace Generation

Angle Configuration Range for Workspace Generation						
S. No.	q_1 *(rad)*	q_2 *(rad)*	q_3 *(rad)*	q_4 *(m)*	q_5 *(m)*	q_6 *(m)*
1	-1.75	-1.75	-1.75	-0.4	-0.30	-0.4
2	-1.36	-1.36	-1.36	-0.31	-0.23	-0.31
3	-0.97	-0.97	-0.97	-0.22	-0.17	-0.22
4	-0.58	-0.58	-0.58	-0.13	-0.10	-0.13
5	-0.19	-0.19	-0.19	-0.04	-0.03	-0.04
6	0.19	0.19	0.19	0.04	0.03	0.04
7	0.58	0.58	0.58	0.13	0.10	0.13
8	0.97	0.97	0.97	0.22	0.17	0.22
9	1.36	1.36	1.36	0.31	0.23	0.31
10	1.75	1.75	1.75	0.4	0.30	0.4

Table 21: CNC Manipulator Joint Angle Range for Training ANN

Angle Configuration Range for Training ANN						
S. No.	q_1 *(rad)*	q_2 *(rad)*	q_3 *(rad)*	q_4 *(m)*	q_5 *(m)*	q_6 *(m)*
1	0.7	0.54	-1.03	-0.13	0.00	-0.21
2	-0.92	1.24	1.56	-0.26	-0.18	0.39
3	0.48	-0.01	0.14	0	-0.11	0.34
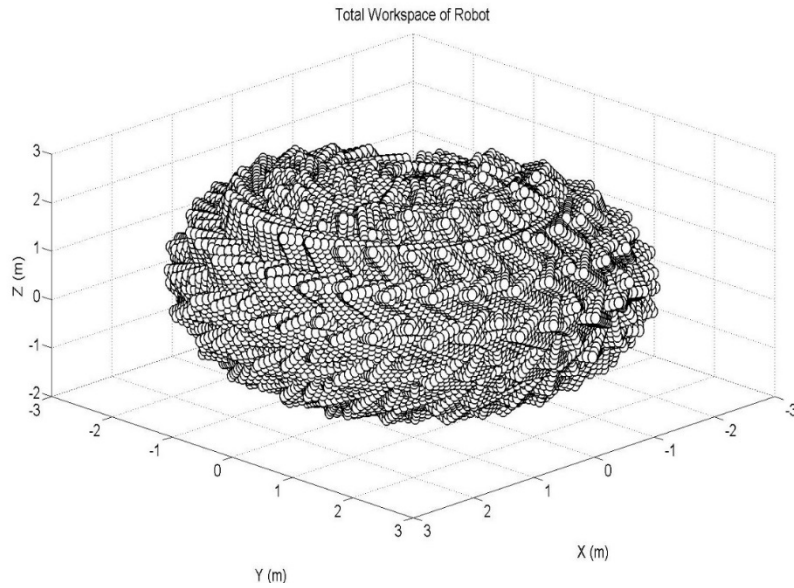
Figure 63: Workspace of CNC Manipulator

Total Workspace of Robot

Workspace Top View

Workspace Right View

Workspace Front View
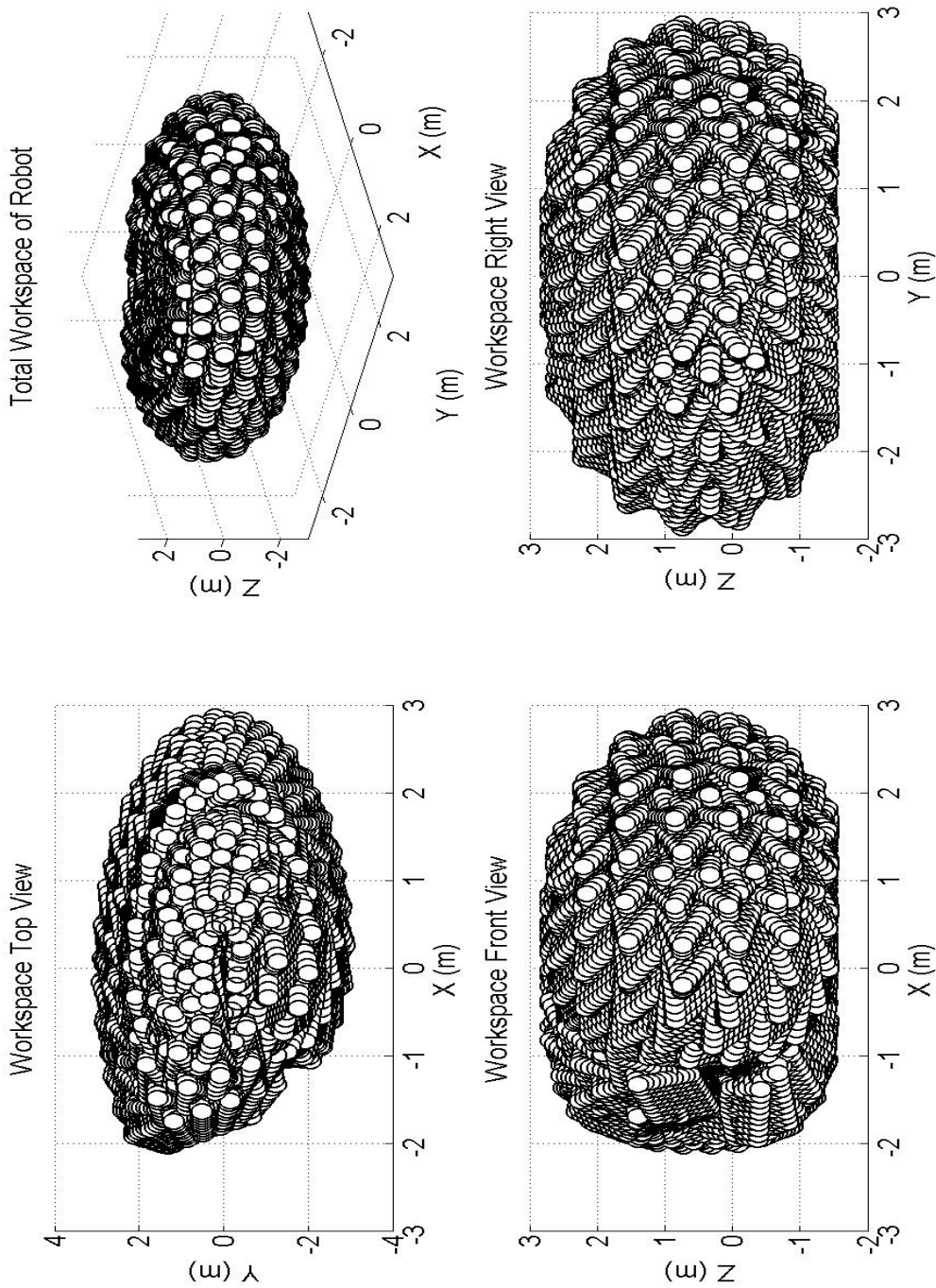
Figure 64: Total Workspace for CNC Manipulator

138
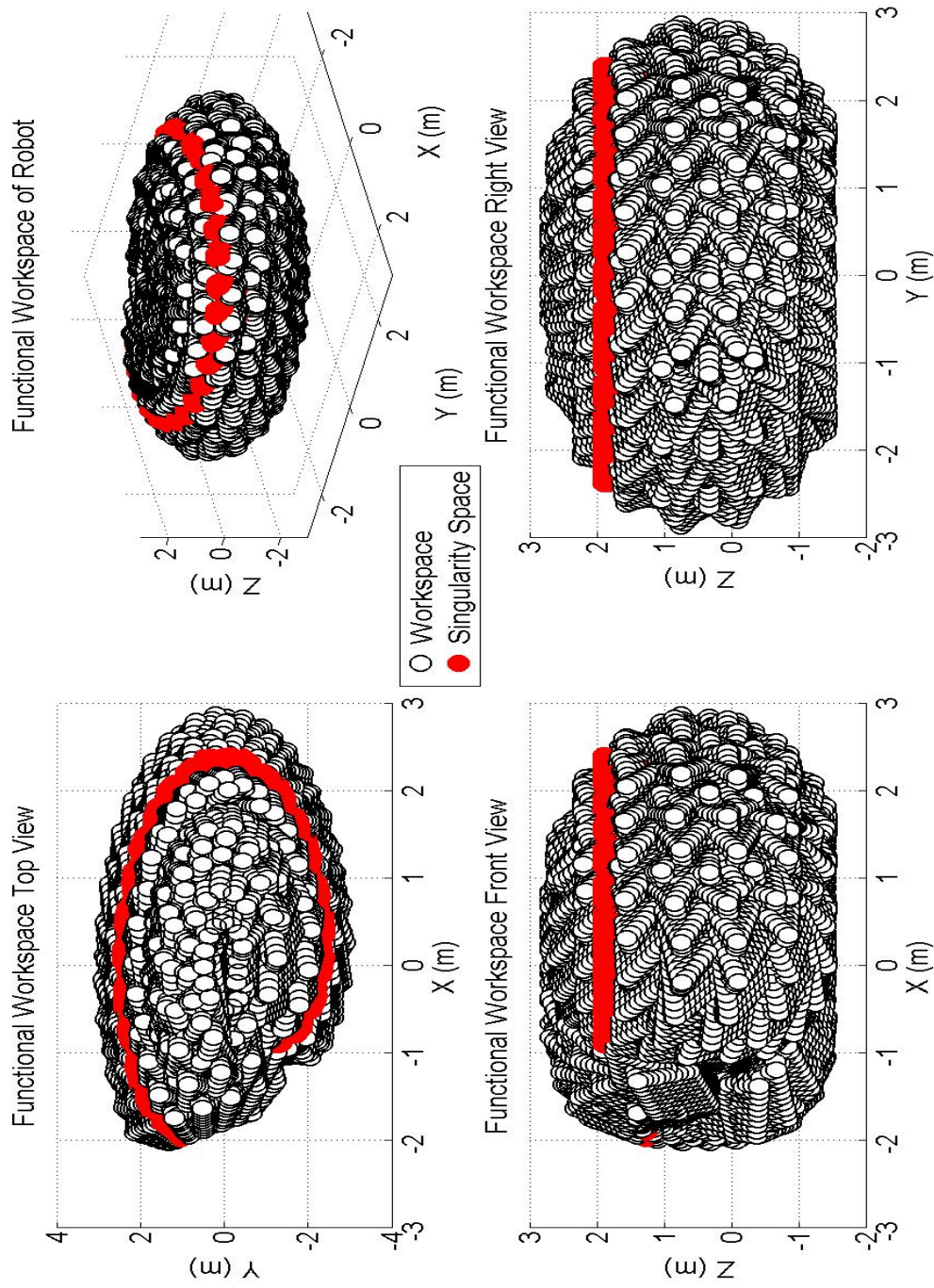
Figure 65: Workspace and Singularity Space for CNC Manipulator
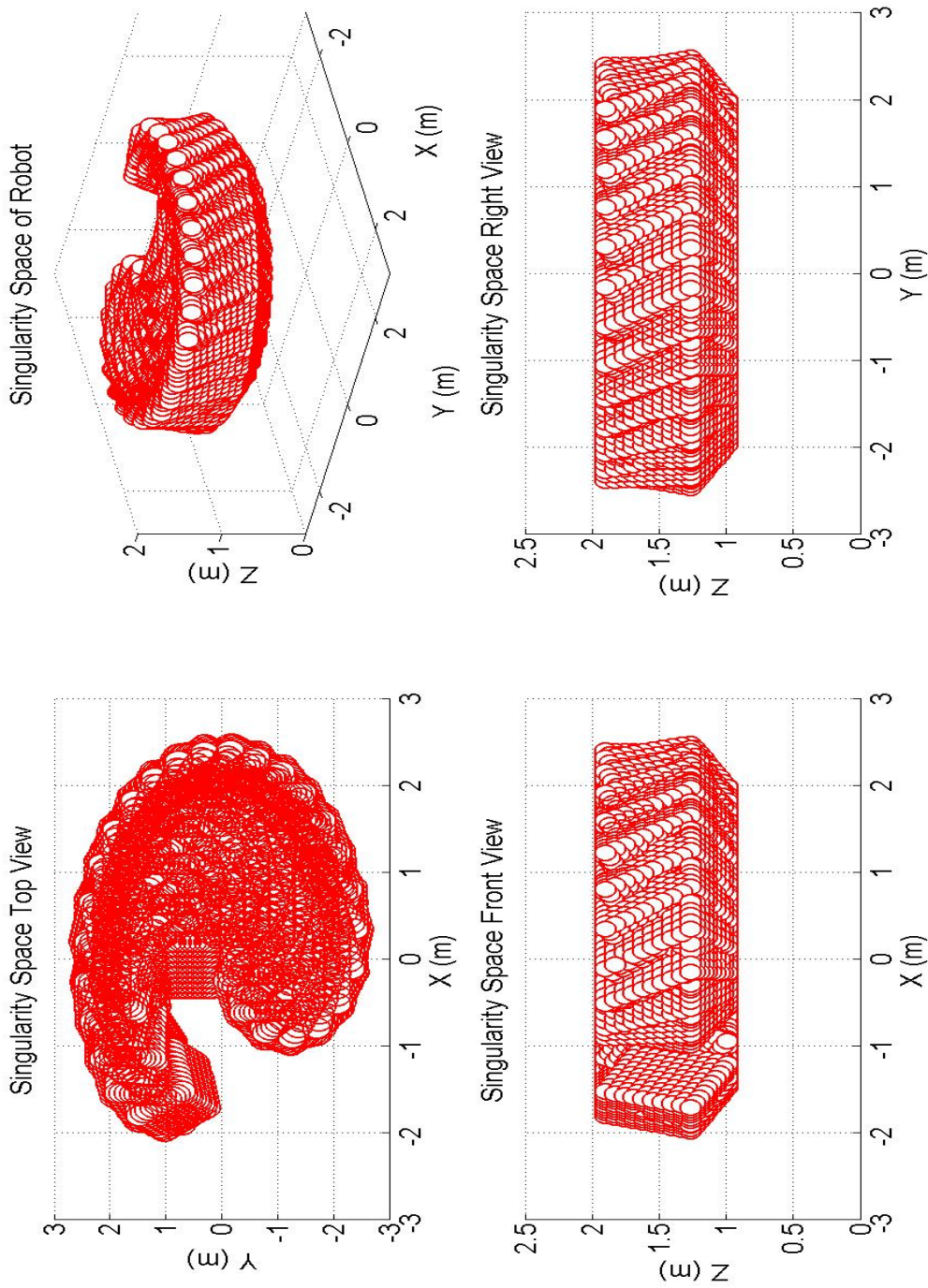
Figure 66: Singularity Space for CNC Manipulator
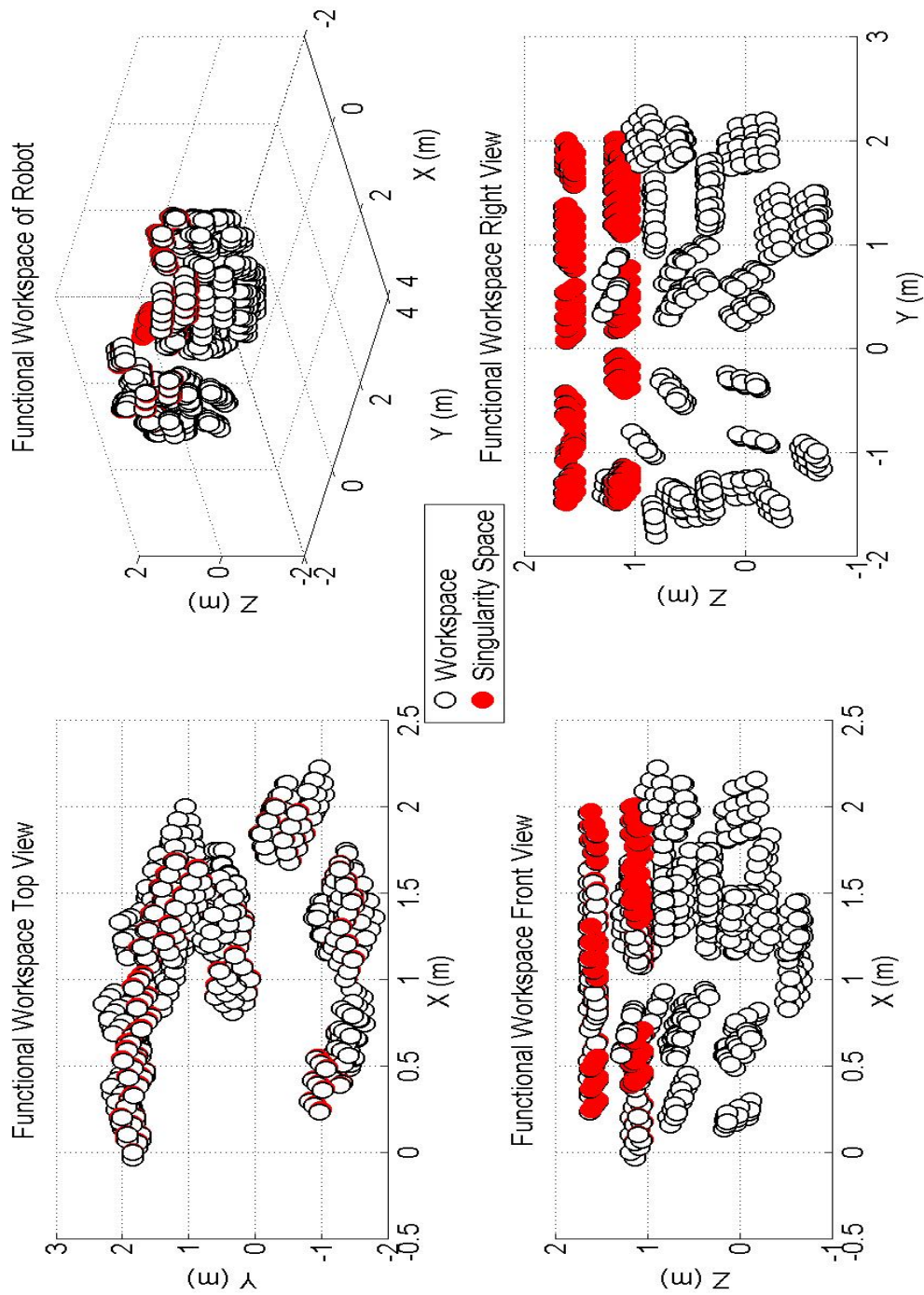
Figure 67: Functional Workspace of CNC Manipulator
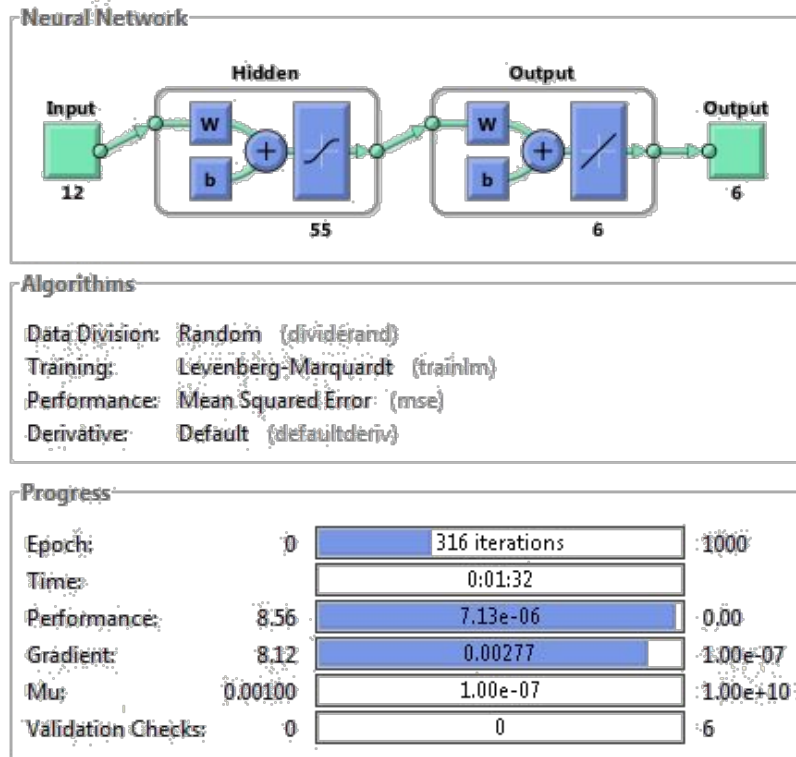
Figure 68: ANN Architecture for CNC Manipulator
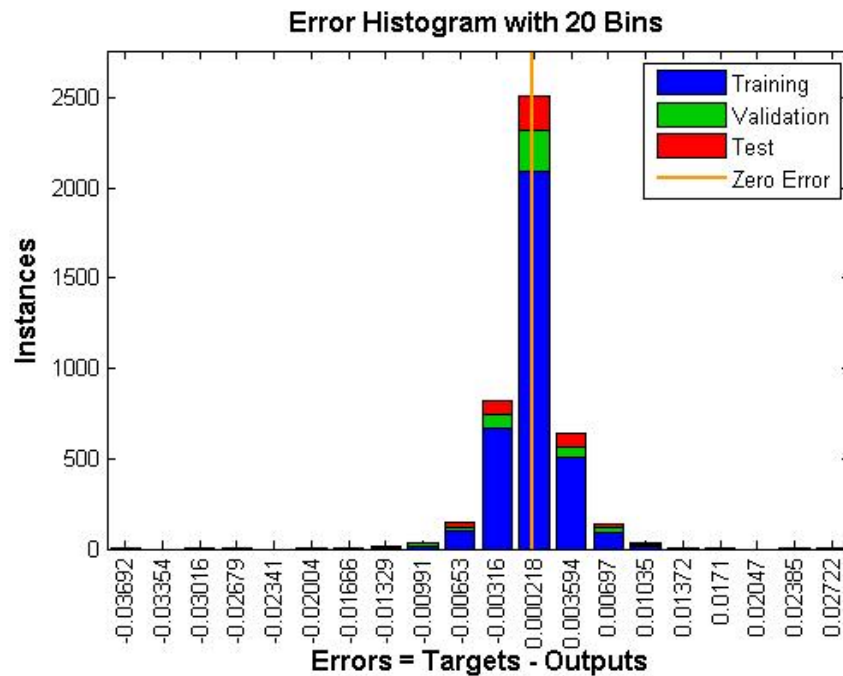
Figure 69: Error Histogram for CNC Manipulator
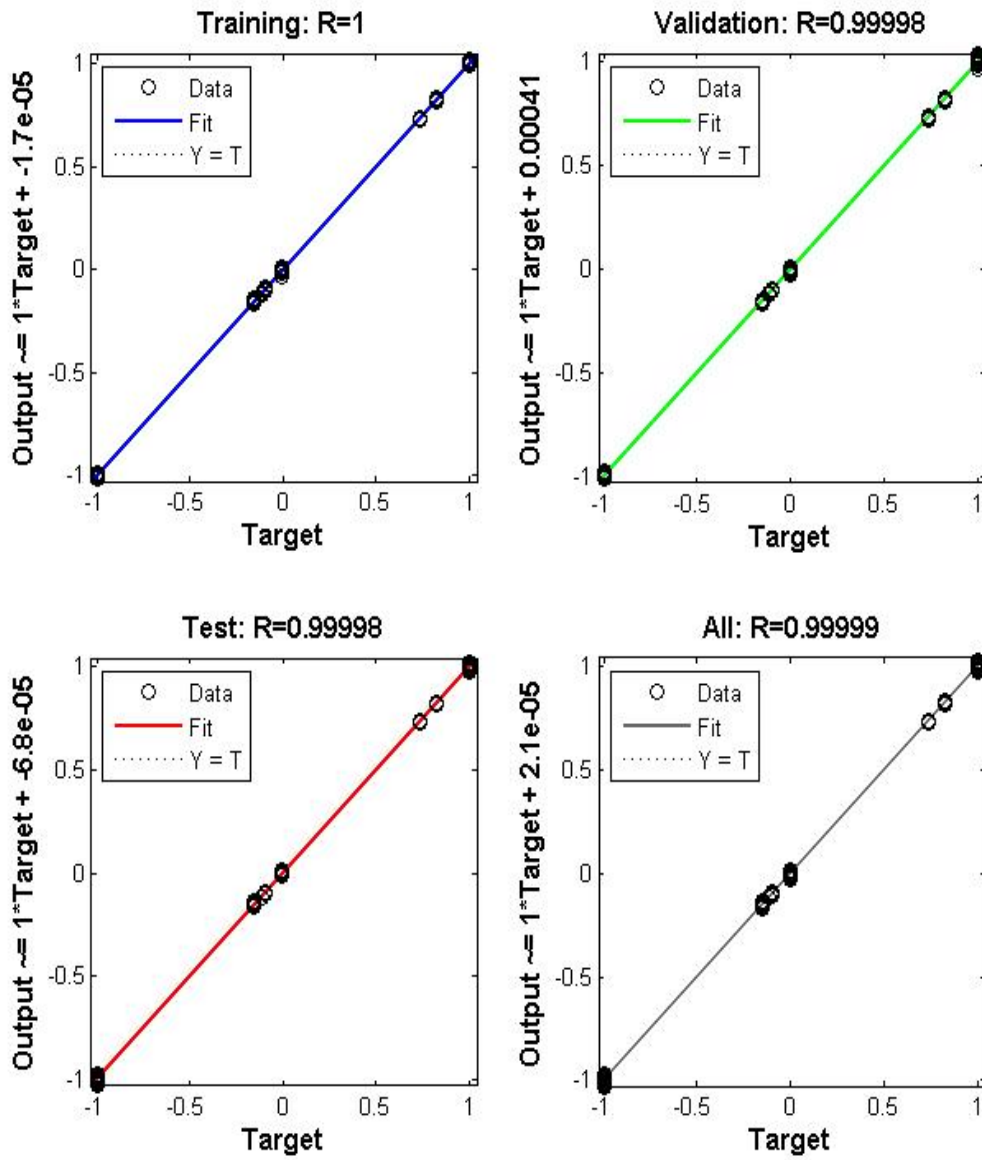
Figure 70: Regression Plot for CNC Manipulator
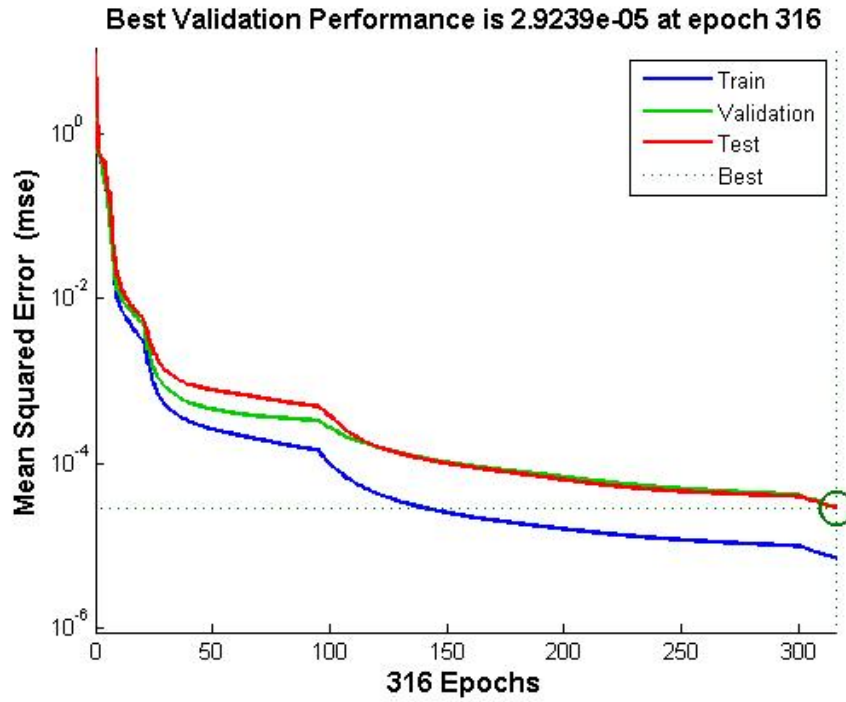
Figure 71: Performance Plot for CNC Manipulator
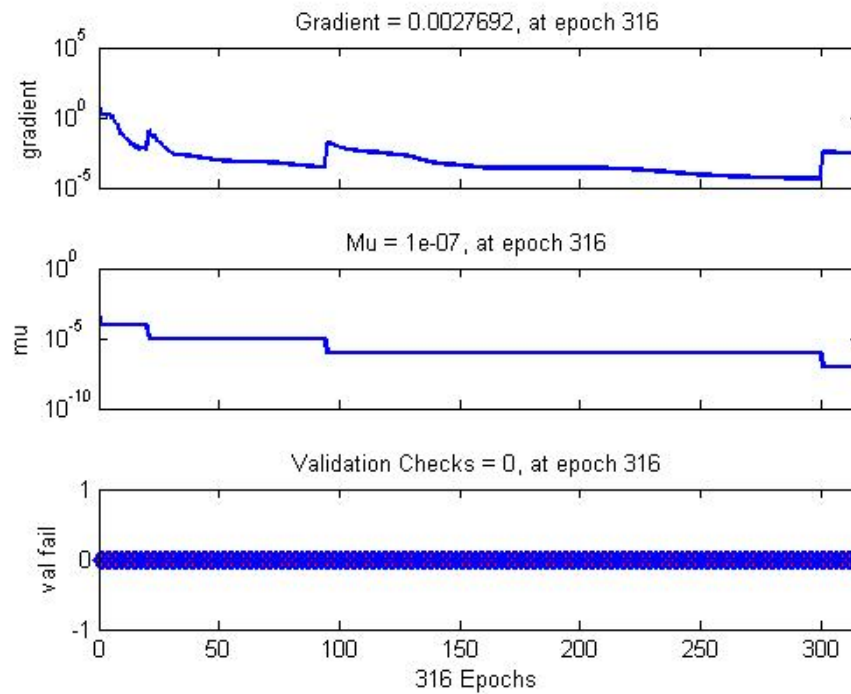
Figure 72: Training State Plot for CNC Manipulator

144
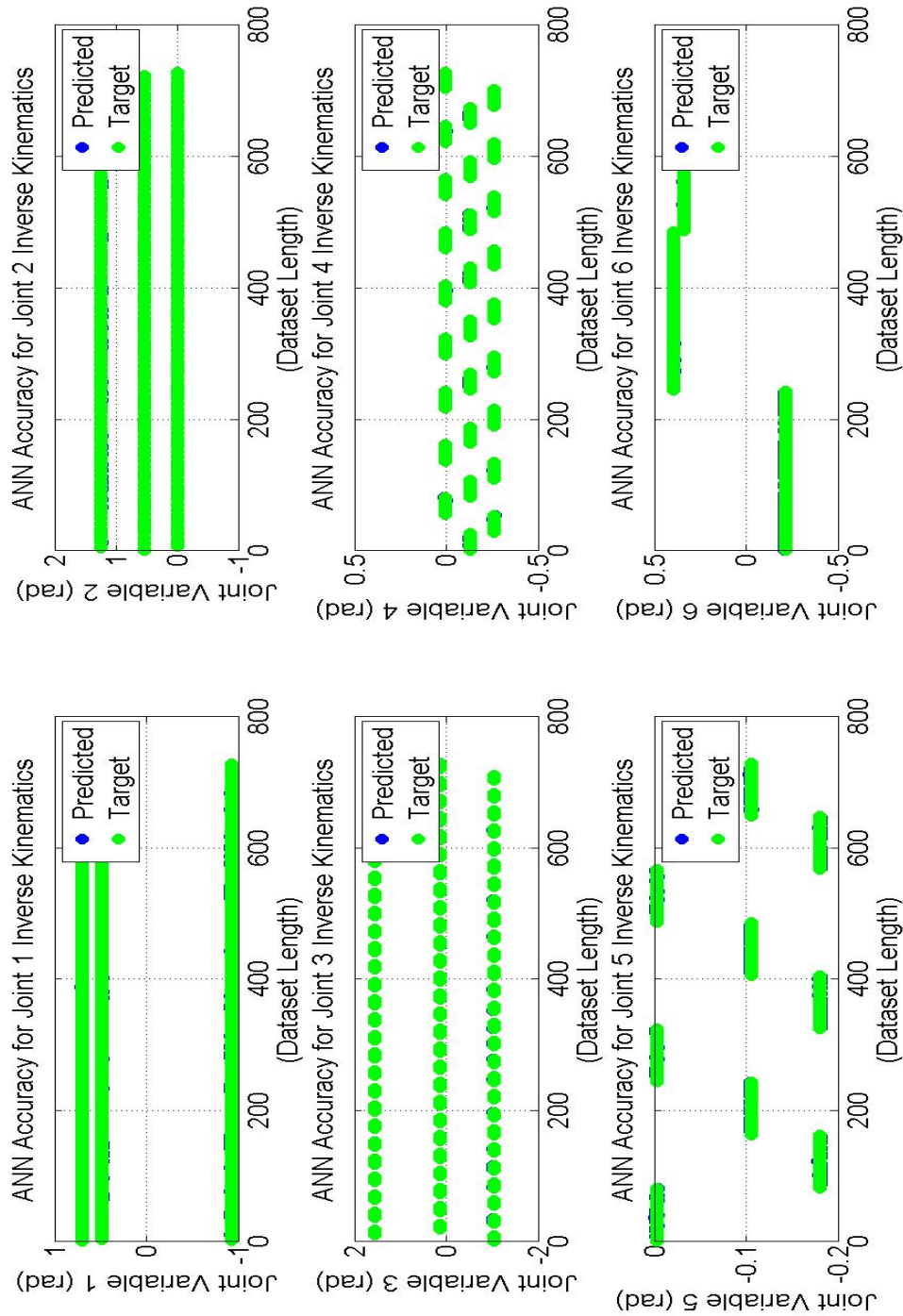
Figure 73: Inverse Kinematics Prediction for CNC Manipulator
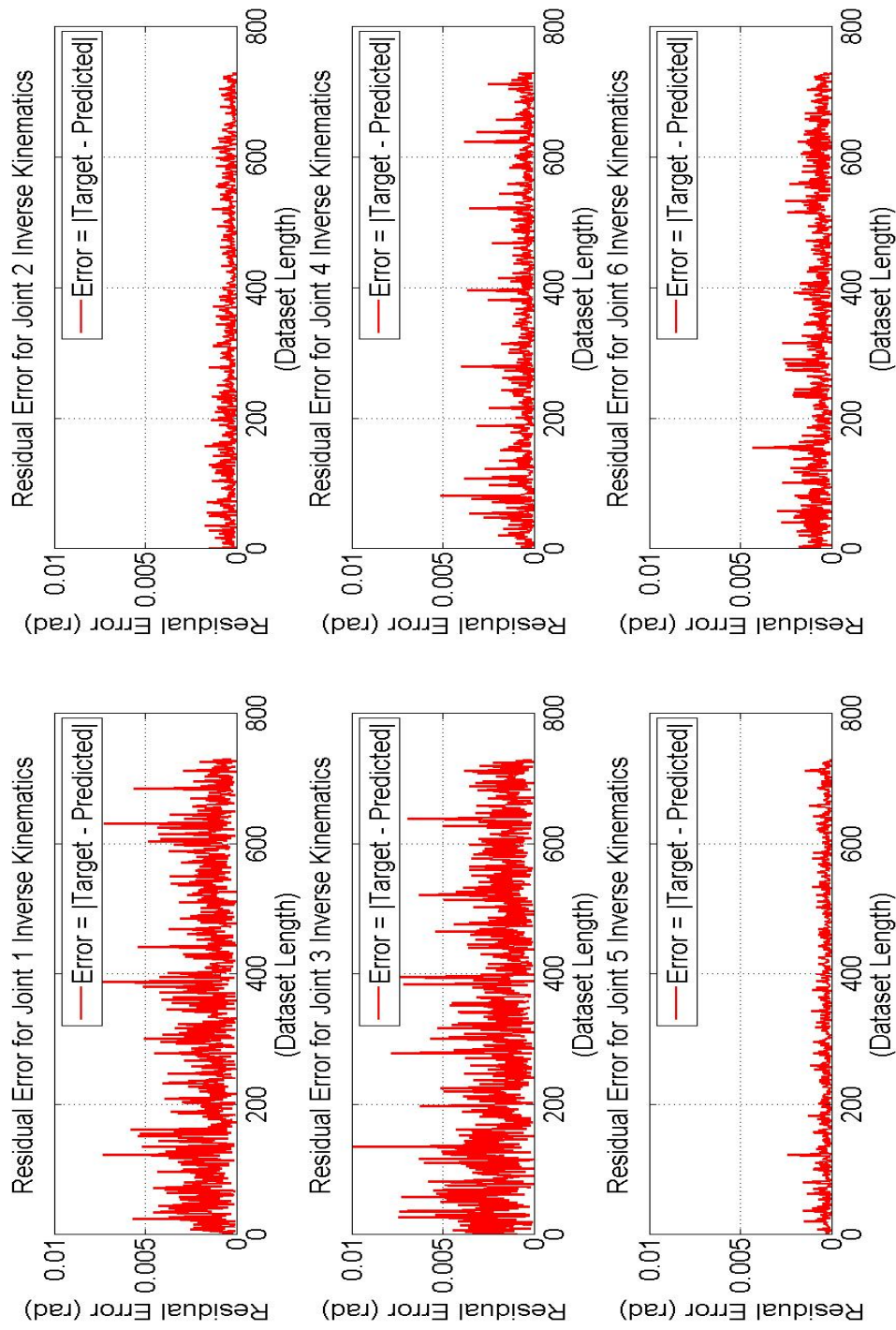
Figure 74: Absolute Residual Error in Inverse Kinematics Prediction for CNC Manipulator
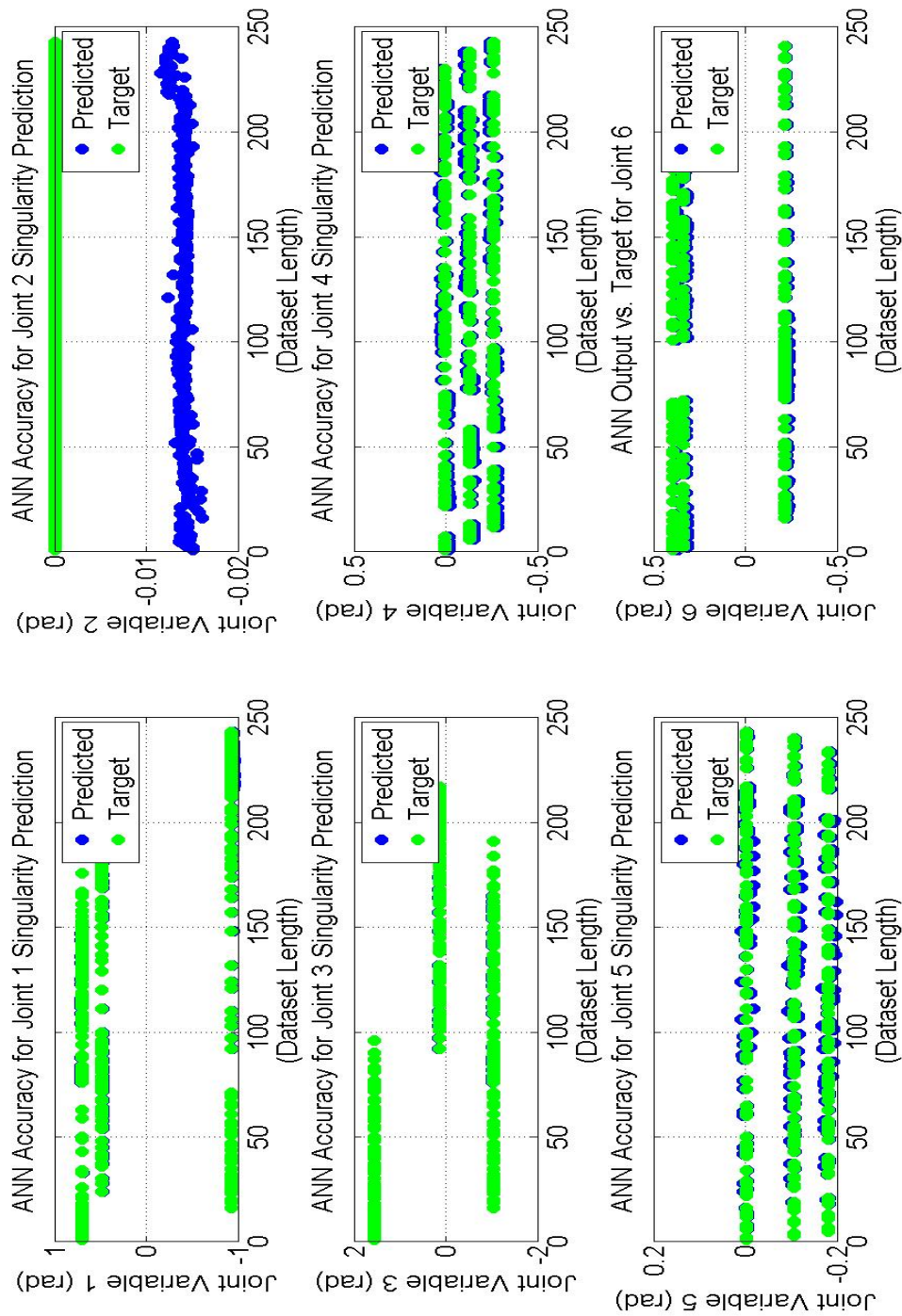
Figure 75: CNC Manipulator Singularity Joint Prediction
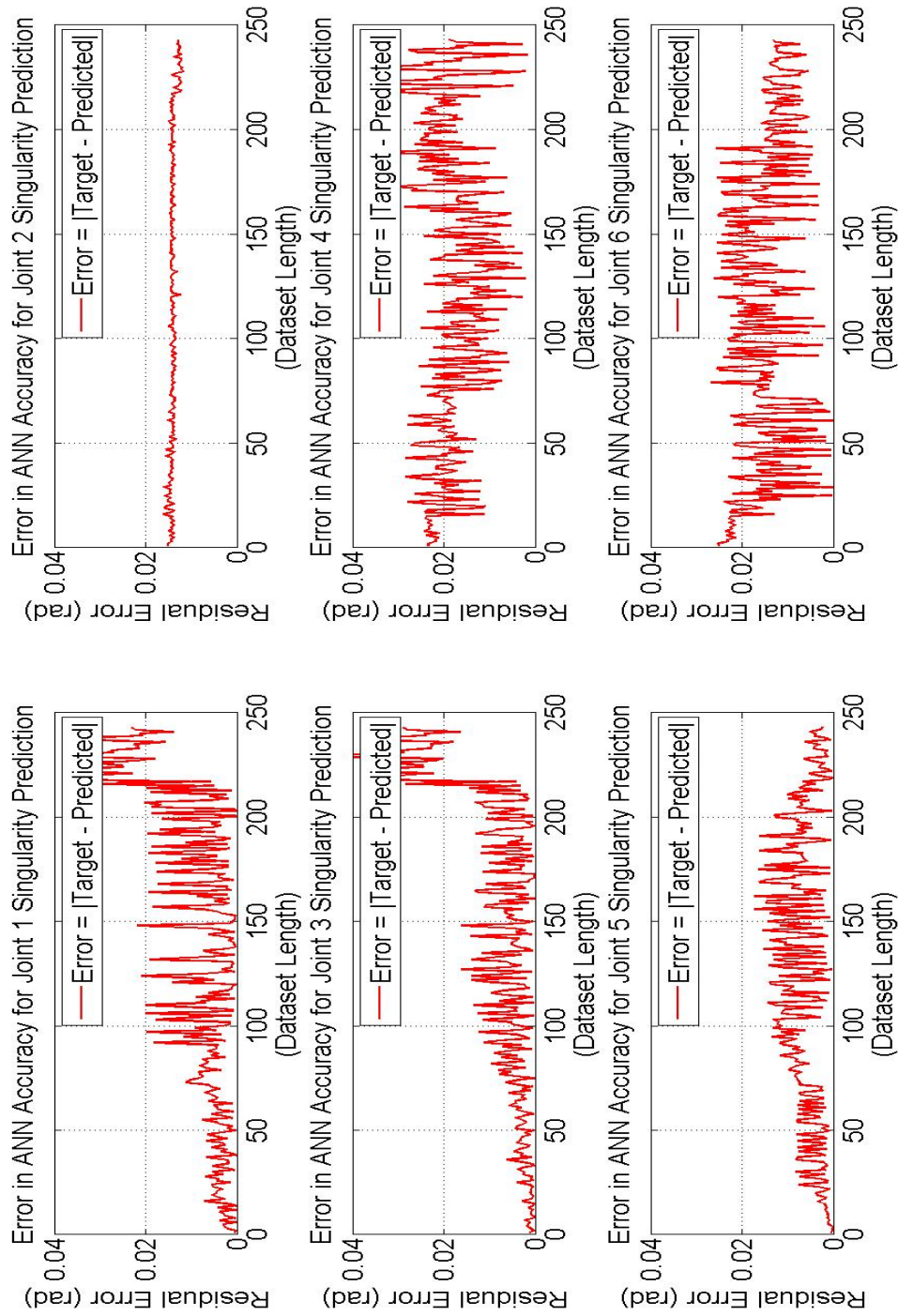
Figure 76: Absolute Residual Error in CNC Manipulator Singularity Joint Prediction
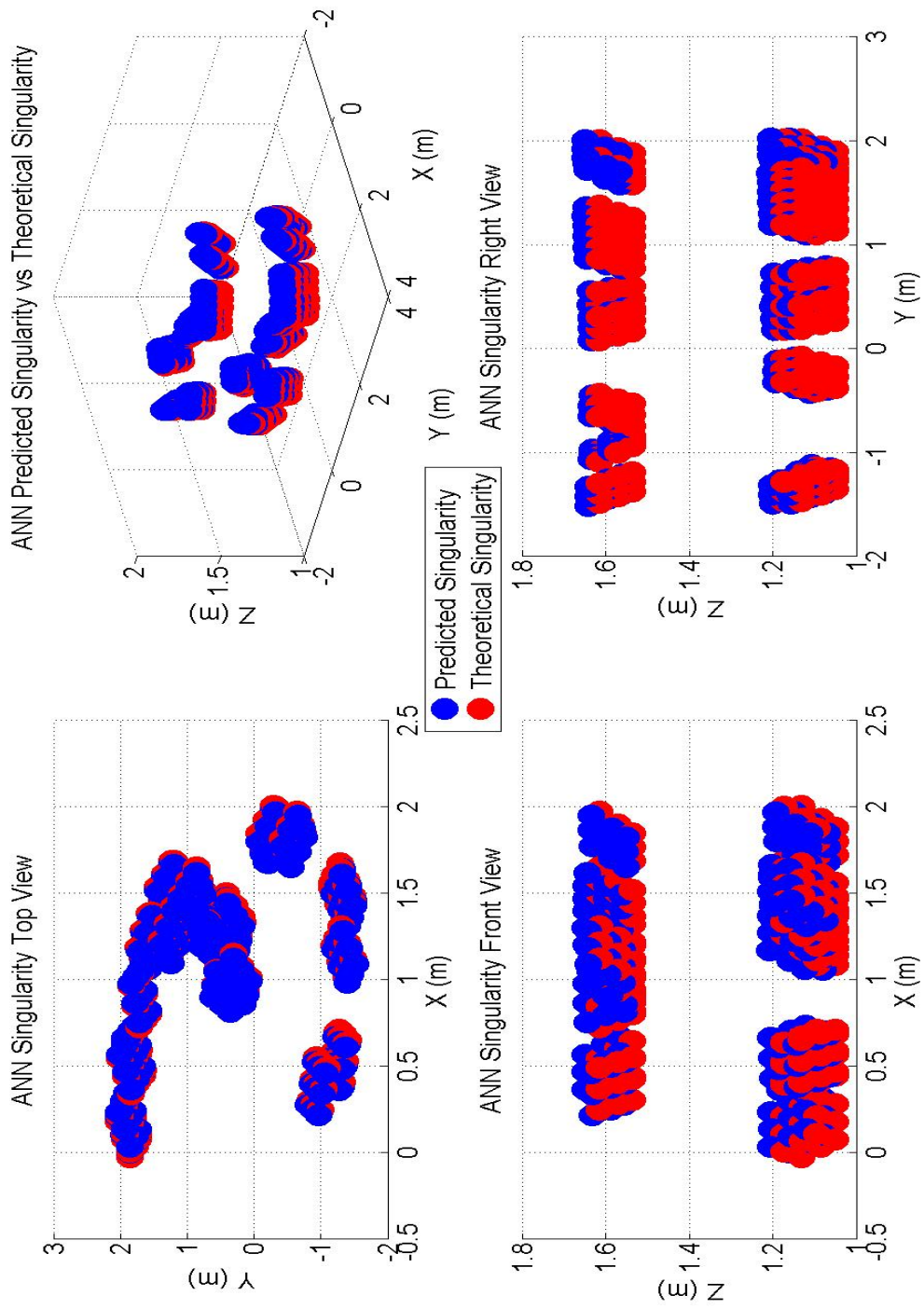
Figure 77: Theoretical vs. ANN Predicted Singularity

MATLAB Output for FANUC M16iB/20 Robot:

```
A06 =

[ (2119411203975849584257560679025816138647158610135*cos(theta1)*sin(theta2))
/3653754093327257295509212081790707549139833135744 - (1624930591299869207383706972457*cos
(theta3)*sin(theta1))/4056481920730334084789450257032 -
(51829985975349589581146940717047233806199740898981*sin(theta1)*sin(theta3))
/730750818665451459101842416358141509827966271488 +
(51829985975349589581146940717047233806199740898981*cos(theta1)*cos(theta2)*cos(theta3))
/730750818665451459101842416358141509827966271488 - (1624930591299869207383706972457*cos
(theta1)*cos(theta2)*sin(theta3))/4056481920730334084789450257032,
(162103109278564056613316834439721*cos(theta1)*sin(theta2))
/2028240960365167042394725128601 + (4035888341267763*cos(theta3)*sin(theta1))
/9007199254740992 + (1624930591299869207383706972457*sin(theta1)*sin(theta3))
/4056481920730334084789450257032 - (1624930591299869207383706972457*cos(theta1)*cos
(theta2)*cos(theta3))/4056481920730334084789450257032 + (4035888341267763*cos(theta1)
*cos(theta2)*sin(theta3))/9007199254740992, (162103109278564056613316834439721*cos(theta3)
*sin(theta1))/2028240960365167042394725128601 -
(2874480072420620022044353827333987691336125521*cos(theta1)*sin(theta2))
/18268770466636286477546060408953537745699151567872 -
(2119411203975849584257560679025816138647158610135*sin(theta1)*sin(theta3))
/3653754093327257295509212081790707549139833135744 +
(2119411203975849584257560679025816138647158610135*cos(theta1)*cos(theta2)*cos(theta3))
/3653754093327257295509212081790707549139833135744 + (162103109278564056613316834439721*cos
(theta1)*cos(theta2)*sin(theta3))/2028240960365167042394725128601, (3*cos(theta1))/5 -
sin(theta1)/2 + (cos(theta1)*cos(theta2))/2 +
(692271311144806441947056773204977948708861426731*cos(theta1)*sin(theta2))
/730750818665451459101842416358141509827966271488 + (200155071178903204985590073311031*cos
(theta3)*sin(theta1))/81129638414606681695789005144064 -
(11338193656033425938791416884058787141115622994645*sin(theta1)*sin(theta3))
/146150163733090291820368483271628301965593254297 - d4*cos(theta3)*sin(theta1) +
(4035888341267763*d5*cos(theta1)*sin(theta2))/9007199254740992 -
(162103109278564056613316834439721*d6*cos(theta1)*sin(theta2))
/2028240960365167042394725128601 - (4035888341267763*d6*cos(theta3)*sin(theta1))
/9007199254740992 - (4026203041061939*d5*sin(theta1)*sin(theta3))/4503599627370496 -
(1624930591299869207383706972457*d6*sin(theta1)*sin(theta3))
/4056481920730334084789450257032 +
(11338193656033425938791416884058787141115622994645*cos(theta1)*cos(theta2)*cos(theta3))
/146150163733090291820368483271628301965593254297 +
(200155071178903204985590073311031*cos(theta1)*cos(theta2)*sin(theta3))
/81129638414606681695789005144064 + (4026203041061939*d5*cos(theta1)*cos(theta2)*cos
(theta3))/4503599627370496 + (1624930591299869207383706972457*d6*cos(theta1)*cos(theta2)
*cos(theta3))/4056481920730334084789450257032 - d4*cos(theta1)*cos(theta2)*sin(theta3) -
(4035888341267763*d6*cos(theta1)*cos(theta2)*sin(theta3))/9007199254740992]
[ (1624930591299869207383706972457*cos(theta1)*cos(theta3))
/4056481920730334084789450257032 + (51829985975349589581146940717047233806199740898981*cos
(theta1)*sin(theta3))/730750818665451459101842416358141509827966271488 +
(2119411203975849584257560679025816138647158610135*sin(theta1)*sin(theta2))
/3653754093327257295509212081790707549139833135744 - (1624930591299869207383706972457*cos
(theta2)*sin(theta1)*sin(theta3))/4056481920730334084789450257032 +
(51829985975349589581146940717047233806199740898981*cos(theta2)*cos(theta3)*sin(theta1))
```

/7307508186654514591018424163581415098279662711488, (16210310927856405661316834439721*sin
(theta1)*sin(theta2))/20282409603651670423947251286016 -
(16249305912998692073837066972457*cos(thetal)*sin(theta3))
/40564819207303340847894502572032 - (4035888341267763*cos(theta1)*cos(theta3))
/9007199254740992 + (4035888341267763*cos(theta2)*sin(thetal)*sin(theta3))
/9007199254740992 - (16249305912998692073837066972457*cos(theta2)*cos(theta3)*sin
(theta1))/40564819207303340847894502572032,
(211941120397584958425756067902581613864715861035*cos(thetal)*sin(theta3))
/365375409332725729550921208179070754913983135744 - (16210310927856405661316834439721*cos
(thetal)*cos(theta3))/20282409603651670423947251286016 -
(28744800724206200220443538273339876913361255211*sin(theta1)*sin(theta2))
/182687704666362864775460604089535377456991567872 + (16210310927856405661316834439721*cos
(theta2)*sin(thetal)*sin(theta3))/20282409603651670423947251286016 +
(211941120397584958425756067902581613864715861035*cos(theta2)*cos(theta3)*sin(thetal))
/365375409332725729550921208179070754913983135744, cos(theta1)/2 + (3*sin(theta1))/5 -
(20015507117890320498559007331031*cos(thetal)*cos(theta3))
/81129638414606681695789005144064 + (cos(theta2)*sin(thetal))/2 +
(1133819365603342593879141688405878714115622994645*cos(thetal)*sin(theta3))
/1461501637330902918203684832716283019655932542976 +
(692271311144806441947056773204977948708861426731*sin(thetal)*sin(theta2))
/730750818665451459101842416358141509827966271488 + (20015507117890320498559007331031*cos
(theta2)*sin(thetal)*sin(theta3))/81129638414606681695789005144064 + d4*cos(thetal)*cos
(theta3) + (4035888341267763*d6*cos(thetal)*cos(theta3))/9007199254740992 +
(4026203041061939*d5*cos(thetal)*sin(theta3))/4503599627370496 +
(16249305912998692073837066972457*d6*cos(thetal)*sin(theta3))
/40564819207303340847894502572032 + (4035888341267763*d5*sin(thetal)*sin(theta2))
/9007199254740992 - (16210310927856405661316834439721*d6*sin(thetal)*sin(theta2))
/20282409603651670423947251286016 +
(1133819365603342593879141688405878714115622994645*cos(theta2)*cos(theta3)*sin(thetal))
/1461501637330902918203684832716283019655932542976 + (4026203041061939*d5*cos(theta2)*cos
(theta3)*sin(thetal))/4503599627370496 + (16249305912998692073837066972457*d6*cos(theta2)
*cos(theta3)*sin(thetal))/40564819207303340847894502572032 - d4*cos(theta2)*sin(thetal)
*sin(theta3) - (4035888341267763*d6*cos(theta2)*sin(thetal)*sin(theta3))
/9007199254740992]
[
(211941120397584958425756067902581613864715861035*cos(theta2))
/365375409332725729550921208179070754913983135744 -
(518299859753495895811469407170472338061997408981*cos(theta3)*sin(theta2))
/730750818665451459101842416358141509827966271488 + (16249305912998692073837066972457*sin
(theta2)*sin(theta3))/40564819207303340847894502572032,
(16210310927856405661316834439721*cos(theta2))/20282409603651670423947251286016 +
(16249305912998692073837066972457*cos(theta3)*sin(theta2))
/40564819207303340847894502572032 - (4035888341267763*sin(theta2)*sin(theta3))
/9007199254740992,
- (28744800724206200220443538273339876913361255211*cos(theta2))
/182687704666362864775460604089535377456991567872 -
(211941120397584958425756067902581613864715861035*cos(theta3)*sin(theta2))
/365375409332725729550921208179070754913983135744 - (16210310927856405661316834439721*sin
(theta2)*sin(theta3))/20282409603651670423947251286016,
(692271311144806441947056773204977948708861426731*cos(theta2))
/730750818665451459101842416358141509827966271488 - sin(theta2)/2 -

(1133819365603342593879141688405878714115622994645*cos(theta3)*sin(theta2)) ↙
/1461501637330902918203684832716283019655932542976 -↙
(200155071178903204985590073310310*sin(theta2)*sin(theta3)) ↙
/8112963841460668169578900514064 + (4035888341267763*d5*cos(theta2))/9007199254740992 -↙
(16210310927856405661316834439721*d6*cos(theta2))/20282409603651670423947251286016 -↙
(4026203041061939*d5*cos(theta3)*sin(theta2))/4503599627370496 -↙
(16249305912998692073837066972457*d6*cos(theta3)*sin(theta2)) ↙
/40564819207303340847894502572032 + d4*sin(theta2)*sin(theta3) + (4035888341267763*d6*sin↙
(theta2)*sin(theta3))/9007199254740992 + 1/2]
[↙
0,↙
0,↙
0,↙
1]

Input weights =

ans =

  Columns 1 through 9

	0.84	0.35	-0.23	1.64	0.09	-0.15↙
0.37	-0.61	-1.55				
	-1.01	-1.97	3.44	3.00	1.65	-2.86↙
-2.47	-0.28	0.00				
	-0.33	-0.69	-0.25	0.69	0.22	0.48↙
-2.60	0.23	0.76				
	-0.51	-1.32	1.40	-0.52	0.01	1.62↙
0.44	1.32	0.04				
	0.99	-0.38	1.54	0.39	-0.43	-0.38↙
-0.86	-0.78	-0.27				
	-1.89	-0.29	1.95	1.08	-5.62	1.95↙
2.29	-0.77	1.24				
	-0.26	-0.68	1.06	0.45	0.94	0.06↙
0.10	0.28	-0.34				
	2.97	0.41	0.43	-0.79	-0.29	-0.97↙
-0.50	-0.53	-0.92				
	-0.32	-1.42	-0.48	0.34	-1.07	0.24↙
3.03	0.14	-0.95				
	0.52	0.86	0.20	0.60	-0.64	0.62↙
1.68	0.42	-0.25				
	2.34	-0.08	-0.57	1.14	0.10	0.62↙
-0.47	-0.30	-0.35				
	0.83	1.02	-0.04	-0.13	0.13	0.20↙
-1.37	-0.72	-0.16				
	1.82	2.12	1.21	-1.14	-2.00	0.76↙
0.53	-1.33	0.91				
	-0.76	1.56	-0.99	0.18	-1.32	-0.49↙
1.82	-0.63	-0.21				
	0.50	0.45	-0.69	-0.27	-0.51	-0.57↙
-1.47	1.00	-0.71				

```
                -0.08           0.31           1.03          0.49          0.10          0.88✓
    -0.30           1.06            0.53
                 2.76           2.37           0.27          1.93          0.96          1.01✓
     1.35           1.88           -0.61
                 1.40          -0.58           1.88          0.14         -0.13         -1.43✓
    -0.49           1.97            0.84
                -0.72           0.31           0.17          0.10         -2.08          0.41✓
    -1.12          -0.32           -0.45
                 0.23           0.32          -0.59         -1.30         -0.00          0.48✓
    -1.69           1.13            0.42
                -0.09           0.78          -1.96          2.15          1.21          1.64✓
    -0.52           0.90            1.26
                 1.51          -0.90           0.98          0.73         -0.35         -0.21✓
    -0.25          -0.27            0.14
                 0.56          -0.19          -2.75         -0.53         -0.61         -0.29✓
     0.30          -0.27            0.17
                 0.31          -0.01           0.00         -0.02         -1.23          1.26✓
    -1.97          -0.25            1.13
                 0.47          -0.36           0.76         -1.17          0.83         -0.23✓
     0.70          -1.39            0.56
                 1.28          -0.89           0.92         -1.12         -0.00          2.11✓
     0.96           1.49            0.54
                 1.24          -2.11           0.76          1.89          1.37         -0.40✓
     1.26           1.09            3.69
                 1.97           1.31          -1.08         -0.59         -0.10          0.78✓
     1.37          -1.42           -1.26
                 0.82          -0.73          -0.75          2.26         -1.54         -1.09✓
     0.78           1.25           -2.37
                 0.49           1.71          -1.32         -1.04         -0.02          0.67✓
     3.63           0.77            0.06
                -0.59           0.05          -1.83         -0.67          0.33          0.68✓
    -0.14           1.18           -1.77
                -0.87           0.91           1.82         -0.48          0.25          1.11✓
    -0.41          -0.35           -0.05
                -1.73           0.42          -0.27          0.60          3.43         -0.26✓
     0.56           0.17            0.51
                 0.52          -0.04           1.32          0.18          0.26         -0.17✓
     0.99          -1.86            0.76
                 1.31          -1.23           0.26          0.41         -1.20         -1.31✓
     1.85          -1.49           -2.05
                 0.73           0.88          -1.76         -0.95         -1.49         -1.93✓
    -0.37          -2.34           -0.08
                 1.00          -1.10           0.55          0.13         -0.79         -1.62✓
     0.33           0.23           -1.13
                -1.77           1.63          -0.66          1.09          1.25         -0.30✓
     0.82           1.19            0.11
                -1.06          -1.55          -1.37          1.23         -1.18         -0.64✓
    -0.26          -1.02            0.84
                -0.40          -0.34          -0.44         -0.08         -0.24         -0.40✓
    -0.43           0.79           -0.43
                 0.74          -1.15          -0.06         -0.78         -0.57          0.37✓
```

```
-0.14        0.51       -0.81
         1.09       0.12      -0.30     -0.92     -0.10      0.52✔
 0.99        0.15       -0.52
        -1.85      -1.32      -2.32      0.27      1.38      1.91✔
-0.87       -0.40       -1.08
        -2.36       0.53      -0.71     -0.65      1.55      0.22✔
-0.71       -0.74       -0.03
        -0.40       1.91      -1.76     -1.55     -1.52      0.47✔
 1.03        0.51        0.70
        -1.72       0.27      -0.73     -0.69      0.10     -0.72✔
 0.24        0.44       -0.62
         0.99       0.10       0.32     -0.66     -0.90      0.27✔
 0.65        0.37       -1.43
        -0.28      -0.21       1.06     -0.63      1.43     -0.51✔
 0.05        0.52       -0.70
        -0.50       0.25      -0.11      0.87      0.70     -0.64✔
-0.81        0.23       -1.40
        -1.11      -1.89       0.56     -0.66     -0.86      0.04✔
 0.79       -0.04       -0.06
        -0.58       0.52       1.96      0.62     -0.65      0.58✔
 0.99        0.33       -0.27
         0.61       0.68       1.41      0.56     -0.01     -0.45✔
 0.79        0.96        0.25
        -0.12       0.02      -0.22      0.13     -0.12     -0.20✔
-1.44       -1.86        0.34
         0.54      -0.30       0.54      1.60     -0.04     -1.97✔
-0.15        0.79        0.84
         0.66       0.60       0.25      1.05     -0.19      0.63✔
-0.45        0.27       -1.58
```

Columns 10 through 12

```
        -0.65        0.05        0.71
         0.79       -6.81       -1.63
        -0.02       -2.01       -0.30
        -0.19       -0.05       -0.38
        -0.37       -0.49        0.08
        -0.05       -0.55       -0.14
         0.58       -0.24       -0.43
        -0.38        0.84        0.48
        -0.21        0.67        0.53
         0.48       -3.80       -0.93
         0.53        0.50       -0.09
        -0.34       -0.18        0.26
        -0.09        0.83       -0.36
         0.06        0.89        0.05
        -0.56       -0.21        0.32
        -0.44       -0.47       -0.50
         0.47       -3.53       -1.01
         0.31       -0.03        0.64
        -0.15        1.40       -0.10
```

```
        0.51        -0.03         0.26
        0.29        -3.17        -0.44
        0.54        -0.99         0.22
       -0.24         0.07        -0.32
       -0.26        -0.97        -0.26
        0.17        -1.00         0.08
       -0.36         1.09        -0.11
        0.69        -5.87        -1.52
       -0.33         2.30         0.44
       -0.60        -0.00         0.44
       -0.44         3.61         1.48
       -0.33         0.01        -0.51
        0.14        -0.41        -0.01
        0.44        -3.74        -0.46
       -0.24        -0.40        -0.46
        0.25        -0.16         0.32
        0.30        -1.07         0.03
       -0.40        -0.05        -0.04
       -0.06        -0.89         0.11
       -0.63         5.79         1.54
       -0.38        -0.35         0.29
        0.43        -0.27        -0.11
       -0.31         0.28        -0.49
       -0.50         0.72         0.05
       -0.52        -0.56        -0.15
       -0.17         1.38         0.15
        0.47         0.56        -0.06
       -0.19         1.27         0.99
       -0.08        -0.39        -0.33
       -0.44        -0.27        -0.55
       -0.18        -1.08         0.11
        0.00        -0.82        -0.20
       -0.03         0.10         0.05
        0.00        -0.00         0.03
       -0.42         1.08        -0.16
       -0.56        -0.59         0.41
```

Layer weights =

ans =

  Columns 1 through 9

```
        0.02         0.23        -0.05         0.72        -0.36         0.39↙
-0.24         0.18        -0.11
       -0.01        -0.00         0.01        -0.07         0.03        -0.00↙
 0.02        -0.01         0.01
       -0.01        -0.16         0.13        -0.81         0.41        -0.07↙
 0.26        -0.19         0.11
        1.79        -1.22         2.83        -0.06        -2.13        -2.69↙
-3.29        -0.63         4.20
```

```
        -1.68          -0.56          -0.89           3.25          -5.92           4.94✓
3.94          -5.33           3.29
         1.03          -0.12          -1.22           0.52          -2.84           0.56✓
1.88          -0.05          -0.40


   Columns 10 through 18


        -0.66          -0.31          -0.35          -0.26          -0.50           0.19✓
0.26          -0.45           0.21
        -0.01           0.02           0.08           0.02          -0.07          -0.03✓
-0.01           0.01          -0.01
         0.38           0.22           0.03           0.13           0.11          -0.17✓
-0.09           0.30          -0.21
         3.89           1.04          -0.87           0.54           4.56           4.80✓
-5.21           2.00           2.11
         2.51           4.82          -1.38           2.99          -6.32           0.23✓
-0.94           1.68          -2.55
         0.36          -0.73           4.68           0.42           1.17          -1.45✓
-0.73           0.34          -1.41


   Columns 19 through 27


         0.15           0.07          -0.13           0.10          -0.81          -0.02✓
0.16          -0.57          -0.37
         0.00          -0.00          -0.00           0.01          -0.01           0.01✓
-0.01           0.01           0.01
        -0.02          -0.08           0.14          -0.08           0.54          -0.06✓
-0.27           0.42           0.23
        -1.66           3.29           1.30          -1.99          -2.20           3.09✓
-4.63          -0.12           1.98
         3.81           1.94          -2.56           0.24          -1.66          -0.14✓
3.55          -2.78           1.62
         2.91          -1.01          -0.37          -0.13           0.18           2.36✓
-2.35           2.02           0.24


   Columns 28 through 36


        -0.21          -0.03          -0.23           0.03          -0.40          -0.21✓
0.08           0.08           0.31
         0.00          -0.02           0.00           0.00           0.10           0.01✓
0.03          -0.03          -0.02
         0.13          -0.01           0.13           0.02           0.62           0.17✓
-0.00          -0.06          -0.29
         3.31           1.53           1.69          -0.48          -1.94           1.27✓
0.21           0.91          -4.15
         0.74           3.70           1.14           4.03          -5.05          -1.49✓
-6.54          -3.73           0.82
        -0.14          -1.29           0.18           1.12          -2.39           0.02✓
1.16          -0.50           1.81


   Columns 37 through 45
```

```
        -0.47         0.26        -0.32        -0.31         0.16        -0.12
0.05         0.19         0.54
         0.05        -0.03         0.00        -0.08        -0.01         0.03
0.02         0.01         0.00
         0.45        -0.20         0.18         0.01        -0.42        -0.05
0.04         0.05        -0.19
         0.25        -4.69         1.62        -2.93        -4.88        -5.30
1.51        -1.49         0.84
        -4.72        -0.49         1.62         2.91        -1.18        -1.12
-3.05         4.05        -5.56
        -0.34        -0.43         0.22        -1.92         0.19         0.64
-1.75         0.57        -0.91

   Columns 46 through 54

         0.11        -0.10         0.30         0.07        -0.23         1.94
-0.56         0.13        -0.37
        -0.01         0.01        -0.01         0.02        -0.01        -0.03
-0.28         1.45        -0.00
         0.13         0.07        -0.44        -0.09         0.14        -0.84
1.99         0.04         0.29
        -2.67         2.03         0.24         1.13        -1.53        -2.18
-0.83         1.09        -3.91
        -5.64         2.45        -0.38        -6.13        -4.29         0.38
1.86         4.50         1.15
        -2.04        -1.02         5.27        -0.55         4.74         1.01
-0.34         0.28         1.30

   Column 55

         -0.20
          0.00
          0.15
         -5.52
          2.21
          0.25

Input bias =

ans =

         -2.25
         -4.34
          2.54
          1.04
         -1.34
          2.68
         -1.51
         -2.86
         -2.15
```

```
                -1.52
                -1.83
                -1.05
                 0.22
                 1.41
                 1.23
                 0.86
                -2.58
                -1.13
                -1.14
                -0.78
                 0.96
                -0.98
                -0.69
                 1.58
                -0.13
                 0.11
                -3.82
                 0.47
                 0.11
                -0.51
                -0.38
                 0.03
                -0.59
                -0.21
                -1.22
                 1.42
                -2.18
                 0.09
                -2.47
                -1.31
                 1.55
                -1.26
                 2.92
                -1.85
                 1.53
                -1.39
                -2.46
                -0.61
                 0.85
                 2.96
                 2.16
                 1.61
                 2.14
                 1.17
                 1.37
```

Layer bias =

ans =

```
                         -2.43
                         -0.14
                          0.27
                          1.34
                          2.98
                         -1.20


Network_Performance =

    1.1554e-05


Training_Performance =

    7.1317e-06


Validation_Performance =

    2.9239e-05


Testing_Performance =

    2.9182e-05


Jacobian in Base Frame
[ (0.79923*cos(theta3)*sin(theta1) - 0.15734*cos(theta1)*sin(theta2) - 0.58006*sin↙
(theta1)*sin(theta3) + 0.58006*cos(theta1)*cos(theta2)*cos(theta3) + 0.79923*cos(theta1)↙
*cos(theta2)*sin(theta3))*(0.078672*sin(theta2) - 0.47954*cos(theta3) - 0.096734*cos↙
(theta2) + 0.34804*sin(theta3) - 0.68965*cos(theta2)*cos(theta3) - 0.10958*cos(theta2)↙
*sin(theta3) - 0.64633*cos(theta3)*sin(theta2) + 0.6884*sin(theta2)*sin(theta3) + 0.40058↙
*d5*cos(theta2) + 0.40058*d6*cos(theta2) + 0.44807*d4*cos(theta3)*sin(theta2) - 0.79923↙
*d5*cos(theta3)*sin(theta2) + 0.20077*d6*cos(theta3)*sin(theta2) - 0.40058*d4*sin(theta2)↙
*sin(theta3) + 0.894*d6*sin(theta2)*sin(theta3)) - 1.0*(0.79923*cos(theta1)*sin(theta2) +↙
0.44807*cos(theta3)*sin(theta1) + 0.40058*sin(theta1)*sin(theta3) - 0.40058*cos(theta1)↙
*cos(theta2)*cos(theta3) + 0.44807*cos(theta1)*cos(theta2)*sin(theta3))*(0.62365*cos↙
(theta2) + 0.26884*cos(theta3) + 0.39962*sin(theta2) + 0.24035*sin(theta3) + 0.023749*cos↙
(theta2)*cos(theta3) + 0.42432*cos(theta2)*sin(theta3) + 0.42432*cos(theta3)*sin(theta2)↙
+ 0.69664*sin(theta2)*sin(theta3) - 0.894*d5*cos(theta2) - 0.44807*d5*cos(theta3)*sin↙
(theta2) - 0.894*d4*sin(theta2)*sin(theta3)) - 1.0*(0.40058*cos(theta3)*sin(theta1) -↙
0.58006*cos(theta1)*sin(theta2) + 0.70927*sin(theta1)*sin(theta3) - 0.70927*cos(theta1)↙
*cos(theta2)*cos(theta3) + 0.40058*cos(theta1)*cos(theta2)*sin(theta3))*(0.3576*cos↙
(theta2) + 0.24035*cos(theta3) - 0.29003*sin(theta2) + 0.42556*sin(theta3) - 0.15435*cos↙
(theta2)*cos(theta3) + 0.55492*cos(theta2)*sin(theta3) + 0.5999*cos(theta3)*sin(theta2) -↙
0.095627*sin(theta2)*sin(theta3) - 0.20077*d5*cos(theta2) - 0.79923*d6*cos(theta2) +↙
0.894*d4*cos(theta3)*sin(theta2) + 0.40058*d5*cos(theta3)*sin(theta2) - 0.40058*d6*cos↙
(theta3)*sin(theta2) + 0.20077*d4*sin(theta2)*sin(theta3) + 0.44807*d6*sin(theta2)*sin↙
(theta3)), (0.79923*cos(theta3)*sin(theta1) - 0.15734*cos(theta1)*sin(theta2) - 0.58006↙
*sin(theta1)*sin(theta3) + 0.58006*cos(theta1)*cos(theta2)*cos(theta3) + 0.79923*cos↙
```

(theta1)*cos(theta2)*sin(theta3))*(0.6884*cos(theta3) + 0.64633*sin(theta3) - 0.40058
*d4*cos(theta3) + 0.894*d6*cos(theta3) - 0.44807*d4*sin(theta3) + 0.79923*d5*sin(theta3)
- 0.20077*d6*sin(theta3) + 0.078672) + (0.40058*cos(theta3)*sin(theta1) - 0.58006*cos
(theta1)*sin(theta2) + 0.70927*sin(theta1)*sin(theta3) - 0.70927*cos(theta1)*cos(theta2)
*cos(theta3) + 0.40058*cos(theta1)*cos(theta2)*sin(theta3))*(0.095627*cos(theta3) +
0.5999*sin(theta3) - 0.20077*d4*cos(theta3) - 0.44807*d6*cos(theta3) + 0.894*d4*sin
(theta3) + 0.40058*d5*sin(theta3) - 0.40058*d6*sin(theta3) + 0.29003) - 1.0*(0.69664*cos
(theta3) - 0.42432*sin(theta3) - 0.894*d4*cos(theta3) + 0.44807*d5*sin(theta3) + 0.39962)
*(0.79923*cos(theta1)*sin(theta2) + 0.44807*cos(theta3)*sin(theta1) + 0.40058*sin(theta1)
*sin(theta3) - 0.40058*cos(theta1)*cos(theta2)*cos(theta3) + 0.44807*cos(theta1)*cos
(theta2)*sin(theta3)),   (0.894*d5 - 0.62365)*(0.79923*cos(theta1)*sin(theta2) +
0.44807*cos(theta3)*sin(theta1) + 0.40058*sin(theta1)*sin(theta3) - 0.40058*cos(theta1)
*cos(theta2)*cos(theta3) + 0.44807*cos(theta1)*cos(theta2)*sin(theta3)) + (0.40058*d5 +
0.40058*d6 - 0.096734)*(0.79923*cos(theta3)*sin(theta1) - 0.15734*cos(theta1)*sin(theta2)
- 0.58006*sin(theta1)*sin(theta3) + 0.58006*cos(theta1)*cos(theta2)*cos(theta3) + 0.79923
*cos(theta1)*cos(theta2)*sin(theta3)) + (0.20077*d5 + 0.79923*d6 - 0.3576)*(0.40058*cos
(theta3)*sin(theta1) - 0.58006*cos(theta1)*sin(theta2) + 0.70927*sin(theta1)*sin(theta3)
- 0.70927*cos(theta1)*cos(theta2)*cos(theta3) + 0.40058*cos(theta1)*cos(theta2)*sin
(theta3)), 8.0303e-18*cos(theta1)*cos(theta2)*cos(theta3) - 1.0*cos(theta3)*sin(theta1) -
8.0303e-18*sin(theta1)*sin(theta3) - 1.6022e-17*cos(theta1)*sin(theta2) - 1.0*cos(theta1)
*cos(theta2)*sin(theta3), 0.44807*cos(theta1)*sin(theta2) - 0.894*sin(theta1)*sin(theta3)
+ 0.894*cos(theta1)*cos(theta2)*cos(theta3), 0.40058*cos(theta1)*cos(theta2)*cos(theta3)
- 0.44807*cos(theta3)*sin(theta1) - 0.40058*sin(theta1)*sin(theta3) - 0.79923*cos(theta1)
*sin(theta2) - 0.44807*cos(theta1)*cos(theta2)*sin(theta3)]
[(0.40058*cos(theta1)*cos(theta3) + 0.70927*cos(theta1)*sin(theta3) + 0.58006
*sin(theta1)*sin(theta2) - 0.40058*cos(theta2)*sin(theta1)*sin(theta3) + 0.70927*cos
(theta2)*cos(theta3)*sin(theta1))*(0.3576*cos(theta2) + 0.24035*cos(theta3) - 0.29003*sin
(theta2) + 0.42556*sin(theta3) - 0.15435*cos(theta2)*cos(theta3) + 0.55492*cos(theta2)
*sin(theta3) + 0.5999*cos(theta3)*sin(theta2) - 0.095627*sin(theta2)*sin(theta3) -
0.20077*d5*cos(theta2) - 0.79923*d6*cos(theta2) + 0.894*d4*cos(theta3)*sin(theta2) +
0.40058*d5*cos(theta3)*sin(theta2) - 0.40058*d6*cos(theta3)*sin(theta2) + 0.20077*d4*sin
(theta2)*sin(theta3) + 0.44807*d6*sin(theta2)*sin(theta3)) + (0.58006*cos(theta1)*sin
(theta3) - 0.79923*cos(theta1)*cos(theta3) - 0.15734*sin(theta1)*sin(theta2) + 0.79923
*cos(theta2)*sin(theta1)*sin(theta3) + 0.58006*cos(theta2)*cos(theta3)*sin(theta1))*
(0.078672*sin(theta2) - 0.47954*cos(theta3) - 0.096734*cos(theta2) + 0.34804*sin(theta3)
- 0.68965*cos(theta2)*cos(theta3) - 0.10958*cos(theta2)*sin(theta3) - 0.64633*cos(theta3)
*sin(theta2) + 0.6884*sin(theta2)*sin(theta3) + 0.40058*d5*cos(theta2) + 0.40058*d6*cos
(theta2) + 0.44807*d4*cos(theta3)*sin(theta2) - 0.79923*d5*cos(theta3)*sin(theta2) +
0.20077*d6*cos(theta3)*sin(theta2) - 0.40058*d4*sin(theta2)*sin(theta3) + 0.894*d6*sin
(theta2)*sin(theta3)) + (0.44807*cos(theta1)*cos(theta3) + 0.40058*cos(theta1)*sin
(theta3) - 0.79923*sin(theta1)*sin(theta2) - 0.44807*cos(theta2)*sin(theta1)*sin(theta3)
+ 0.40058*cos(theta2)*cos(theta3)*sin(theta1))*(0.62365*cos(theta2) + 0.26884*cos(theta3)
+ 0.39962*sin(theta2) + 0.24035*sin(theta3) + 0.023749*cos(theta2)*cos(theta3) + 0.42432
*cos(theta2)*sin(theta3) + 0.42432*cos(theta3)*sin(theta2) + 0.69664*sin(theta2)*sin
(theta3) - 0.894*d5*cos(theta2) - 0.44807*d5*cos(theta2)*sin(theta2) - 0.894*d4*sin
(theta2)*sin(theta3)), (0.58006*cos(theta1)*sin(theta3) - 0.79923*cos(theta1)*cos(theta3)
- 0.15734*sin(theta1)*sin(theta2) + 0.79923*cos(theta2)*sin(theta1)*sin(theta3) + 0.58006
*cos(theta2)*cos(theta3)*sin(theta1))*(0.6884*cos(theta3) + 0.64633*sin(theta3) - 0.40058
*d4*cos(theta3) + 0.894*d6*cos(theta3) - 0.44807*d4*sin(theta3) + 0.79923*d5*sin(theta3)
- 0.20077*d6*sin(theta3) + 0.078672) - 1.0*(0.40058*cos(theta1)*cos(theta3) + 0.70927*cos
(theta1)*sin(theta3) + 0.58006*sin(theta1)*sin(theta2) - 0.40058*cos(theta2)*sin(theta1)

160

*sin(theta3) + 0.70927*cos(theta2)*cos(theta3)*sin(theta1))*(0.095627*cos(theta3) +
0.5999*sin(theta3) - 0.20077*d4*cos(theta3) - 0.44807*d6*cos(theta3) + 0.894*d4*sin
(theta3) + 0.40058*d5*sin(theta3) - 0.40058*d6*sin(theta3) + 0.29003) + (0.44807*cos
(theta1)*cos(theta3) + 0.40058*cos(theta1)*sin(theta3) - 0.79923*sin(theta1)*sin(theta2)
- 0.44807*cos(theta2)*sin(theta1)*sin(theta3) + 0.40058*cos(theta2)*cos(theta3)*sin
(theta1))*(0.69664*cos(theta3) - 0.42432*sin(theta3) - 0.894*d4*cos(theta3) + 0.44807
*d5*sin(theta3) + 0.39962), (0.40058*d5 + 0.40058*d6 - 0.096734)*(0.58006*cos(theta1)*sin
(theta3) - 0.79923*cos(theta1)*cos(theta3) - 0.15734*sin(theta1)*sin(theta2) + 0.79923
*cos(theta2)*sin(theta1)*sin(theta3) + 0.58006*cos(theta2)*cos(theta3)*sin(theta1)) - 1.0
*(0.20077*d5 + 0.79923*d6 - 0.3576)*(0.40058*cos(theta1)*cos(theta3) + 0.70927*cos
(theta1)*sin(theta3) + 0.58006*sin(theta1)*sin(theta2) - 0.40058*cos(theta2)*sin(theta1)
*sin(theta3) + 0.70927*cos(theta2)*cos(theta3)*sin(theta1)) - 1.0*(0.894*d5 - 0.62365)*
(0.44807*cos(theta1)*cos(theta3) + 0.40058*cos(theta1)*sin(theta3) - 0.79923*sin(theta1)
*sin(theta2) - 0.44807*cos(theta2)*sin(theta1)*sin(theta3) + 0.40058*cos(theta2)*cos
(theta3)*sin(theta1)), 1.0*cos(theta1)*cos(theta3) + 8.0303e-18*cos(theta1)*sin(theta3) -
1.6022e-17*sin(theta1)*sin(theta2) - 1.0*cos(theta2)*sin(theta1)*sin(theta3) + 8.0303e-
19*cos(theta2)*cos(theta3)*sin(theta1), 0.894*cos(theta1)*sin(theta3) + 0.44807*sin
(theta1)*sin(theta2) + 0.894*cos(theta2)*cos(theta3)*sin(theta1), 0.44807*cos(theta1)*cos
(theta3) + 0.40058*cos(theta1)*sin(theta3) - 0.79923*sin(theta1)*sin(theta2) - 0.44807
*cos(theta2)*sin(theta1)*sin(theta3) + 0.40058*cos(theta2)*cos(theta3)*sin(theta1)]
[
(0.58006*cos(theta2) - 0.70927*cos(theta3)*sin(theta2) + 0.40058*sin(theta2)*sin(theta3))
*(0.3576*cos(theta2) + 0.24035*cos(theta3) - 0.29003*sin(theta2) + 0.42556*sin(theta3) -
0.15435*cos(theta2)*cos(theta3) + 0.55492*cos(theta2)*sin(theta3) + 0.5999*cos(theta3)
*sin(theta2) - 0.095627*sin(theta2)*sin(theta3) - 0.20077*d5*cos(theta2) - 0.79923*d6*cos
(theta2) + 0.894*d4*cos(theta3)*sin(theta2) + 0.40058*d5*cos(theta3)*sin(theta2) -
0.40058*d6*cos(theta3)*sin(theta2) + 0.20077*d4*sin(theta2)*sin(theta3) + 0.44807*d6*sin
(theta2)*sin(theta3)) - 1.0*(0.15734*cos(theta2) + 0.58006*cos(theta3)*sin(theta2) +
0.79923*sin(theta2)*sin(theta3))*(0.078672*sin(theta2) - 0.47954*cos(theta3) - 0.096734
*cos(theta2) + 0.34804*sin(theta3) - 0.68965*cos(theta2)*cos(theta3) - 0.10958*cos
(theta2)*sin(theta3) - 0.64633*cos(theta3)*sin(theta2) + 0.6884*sin(theta2)*sin(theta3) +
0.40058*d5*cos(theta2) + 0.40058*d6*cos(theta2) + 0.44807*d4*cos(theta3)*sin(theta2) -
0.79923*d5*cos(theta3)*sin(theta2) + 0.20077*d6*cos(theta3)*sin(theta2) - 0.40058*d4*sin
(theta2)*sin(theta3) + 0.894*d6*sin(theta2)*sin(theta3)) - 1.0*(0.79923*cos(theta2) +
0.40058*cos(theta3)*sin(theta2) - 0.44807*sin(theta2)*sin(theta3))*(0.62365*cos(theta2) +
0.26884*cos(theta3) + 0.39962*sin(theta2) + 0.24035*sin(theta3) + 0.023749*cos(theta2)
*cos(theta3) + 0.42432*cos(theta2)*sin(theta3) + 0.42432*cos(theta3)*sin(theta2) +
0.69664*sin(theta2)*sin(theta3) - 0.894*d5*cos(theta2) - 0.44807*d5*cos(theta3)*sin
(theta2) - 0.894*d4*sin(theta2)*sin(theta3)),
- 1.0*(0.79923*cos(theta2) + 0.40058*cos(theta3)*sin(theta2) - 0.44807*sin(theta2)*sin
(theta3))*(0.69664*cos(theta3) - 0.42432*sin(theta3) - 0.894*d4*cos(theta3) + 0.44807
*d5*sin(theta3) + 0.39962) - 1.0*(0.58006*cos(theta2) - 0.70927*cos(theta3)*sin(theta2) +
0.40058*sin(theta2)*sin(theta3))*(0.095627*cos(theta3) + 0.5999*sin(theta3) - 0.20077
*d4*cos(theta3) - 0.44807*d6*cos(theta3) + 0.894*d4*sin(theta3) + 0.40058*d5*sin(theta3)
- 0.40058*d6*sin(theta3) + 0.29003) - 1.0*(0.15734*cos(theta2) + 0.58006*cos(theta3)*sin
(theta2) + 0.79923*sin(theta2)*sin(theta3))*(0.6884*cos(theta3) + 0.64633*sin(theta3) -
0.40058*d4*cos(theta3) + 0.894*d6*cos(theta3) - 0.44807*d4*sin(theta3) + 0.79923*d5*sin
(theta3) - 0.20077*d6*sin(theta3) + 0.078672),
(0.894*d5 - 0.62365)*(0.79923*cos(theta2) + 0.40058*cos(theta3)*sin(theta2) - 0.44807*sin
(theta2)*sin(theta3)) - 1.0*(0.40058*d5 + 0.40058*d6 - 0.096734)*(0.15734*cos(theta2) +
0.58006*cos(theta3)*sin(theta2) + 0.79923*sin(theta2)*sin(theta3)) - 1.0*(0.58006*cos

(theta2) - 0.70927*cos(theta3)*sin(theta2) + 0.40058*sin(theta2)*sin(theta3))*(0.20077*d5
+ 0.79923*d6 - 0.3576),
1.0*sin(theta2)*sin(theta3) - 8.0303e-18*cos(theta3)*sin(theta2) - 1.6022e-17*cos
(theta2),   0.44807*cos(theta2) -
0.894*cos(theta3)*sin(theta2),
0.44807*sin(theta2)*sin(theta3) - 0.40058*cos(theta3)*sin(theta2) - 0.79923*cos(theta2)]
[
7.1963e-18*cos(theta1)*cos(theta2)^2*cos(theta3) - 1.6022e-17*cos(theta1)*sin(theta3) -
8.0303e-18*sin(theta1)*sin(theta2) - 3.5982e-18*cos(theta2)*sin(theta1)*sin(theta3) -
3.5982e-18*cos(theta1)*cos(theta3) + 3.2044e-17*cos(theta1)*cos(theta2)^2*sin(theta3) +
1.6061e-17*cos(theta3)^2*sin(theta1)*sin(theta2) - 1.6162e-17*cos(theta1)*cos(theta2)*sin
(theta2) + 1.6022e-17*cos(theta2)*cos(theta3)*sin(theta1) + 1.0786e-17*cos(theta1)*cos
(theta2)*cos(theta3)^2*sin(theta2) - 1.0786e-17*cos(theta3)*sin(theta1)*sin(theta2)*sin
(theta3) + 1.6061e-17*cos(theta1)*cos(theta2)*cos(theta3)*sin(theta2)*sin(theta3),
3.5982e-18*cos(theta1)*sin(theta2)*sin(theta3) - 8.0303e-18*cos(theta1)*cos(theta2) -
1.0786e-17*cos(theta3)^2*sin(theta1) - 1.0*sin(theta1) - 1.6061e-17*cos(theta3)*sin
(theta1)*sin(theta3) + 1.6061e-17*cos(theta1)*cos(theta2)*cos(theta3)^2 - 1.6022e-17*cos
(theta1)*cos(theta3)*sin(theta2) - 1.0786e-17*cos(theta1)*cos(theta2)*cos(theta3)*sin
(theta3),
1.0*cos(theta1)*sin(theta2) + 1.6022e-17*cos(theta3)*sin(theta1) - 3.5982e-18*sin(theta1)
*sin(theta3) + 3.5982e-18*cos(theta1)*cos(theta2)*cos(theta3) + 1.6022e-17*cos(theta1)
*cos(theta2)*sin(theta3),
0,
0,
0]
[
8.0303e-18*cos(theta1)*sin(theta2) - 3.5982e-18*cos(theta3)*sin(theta1) - 1.6022e-17*sin
(theta1)*sin(theta3) - 1.6162e-17*cos(theta2)*sin(theta1)*sin(theta2) - 1.6061e-17*cos
(theta1)*cos(theta3)^2*sin(theta2) + 7.1963e-18*cos(theta2)^2*cos(theta3)*sin(theta1) +
3.2044e-17*cos(theta2)^2*sin(theta1)*sin(theta3) - 1.6022e-17*cos(theta1)*cos(theta2)*cos
(theta3) + 3.5982e-18*cos(theta1)*cos(theta2)*sin(theta3) + 1.0786e-17*cos(theta2)*cos
(theta3)^2*sin(theta1)*sin(theta2) + 1.0786e-17*cos(theta1)*cos(theta3)*sin(theta2)*sin
(theta3) + 1.6061e-17*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta2)*sin(theta3),
1.0*cos(theta1) - 8.0303e-18*cos(theta2)*sin(theta1) + 1.0786e-17*cos(theta1)*cos(theta3)
^2 - 1.6022e-17*cos(theta3)*sin(theta1)*sin(theta2) + 3.5982e-18*sin(theta1)*sin(theta2)
*sin(theta3) + 1.6061e-17*cos(theta2)*cos(theta3)^2*sin(theta1) + 1.6061e-17*cos(theta1)
*cos(theta3)*sin(theta3) - 1.0786e-17*cos(theta2)*cos(theta3)*sin(theta1)*sin(theta3),
3.5982e-18*cos(theta1)*sin(theta3) - 1.6022e-17*cos(theta1)*cos(theta3) + 1.0*sin(theta1)
*sin(theta2) + 1.6022e-17*cos(theta2)*sin(theta1)*sin(theta3) + 3.5982e-18*cos(theta2)
*cos(theta3)*sin(theta1),
0,
0,
0]
[
(0.15734*cos(theta2) + 0.58006*cos(theta3)*sin(theta2) + 0.79923*sin(theta2)*sin(theta3))
^2 + (0.58006*cos(theta2) - 0.70927*cos(theta3)*sin(theta2) + 0.40058*sin(theta2)*sin
(theta3))^2 + (0.79923*cos(theta2) + 0.40058*cos(theta3)*sin(theta2) - 0.44807*sin
(theta2)*sin(theta3))^2,
8.0303e-18*sin(theta2) - 1.6022e-17*cos(theta2)*cos(theta3) + 3.5982e-18*cos(theta2)*sin
(theta3) - 1.6061e-17*cos(theta3)^2*sin(theta2) + 1.0786e-17*cos(theta3)*sin(theta2)*sin
(theta3),

```
1.0*cos(theta2) - 3.5982e-18*cos(theta3)*sin(theta2) - 1.6022e-17*sin(theta2)*sin↙
(theta3),↙
0,↙
0,↙
0]


(49738315904765780500327411641698900517028355859719137304430275175595935057947251100819858↙
7840563435604130020208410864003541731652420270300928441356102712361072210148836268108469↙9
1969332858657817882705410907874553250172117940008896586034946744523381778942439368470484↙0
9812983031456427179985142759467722448635064589248698292971888119520l1*sin(theta2))↙
/5563590774986284438640031450559556725159910797032425675699365672051224279213746123593288↙
5972818168186054976565058958882713758619825090273161853126012111375244844087293568574798↙9
8261153449853227322096500951541013099642239292264430204254325158790381723965751186358353↙5
04294864488066709964015266969779070426508273725009424610404010l380096


Singularity Equation
Warning: 1 equations in 6 variables. New variables might be introduced.
> In C:\Program Files\MATLAB\R2013a\toolbox\symbolic\symbolic\symengine.p>symengine at 56
  In mupadengine.mupadengine>mupadengine.evalin at 97
  In mupadengine.mupadengine>mupadengine.feval at 150
  In solve at 172
Warning: The solutions are parametrized by the symbols:
u13 = R_
v13 = R_
x669 = R_
y13 = R_
z    = C_

> In solve at 190
 Singularity Solution(s)

SE3 =

[ x669, 0, z, v13, u13, y13]

Absolute Errors in Joint Space
Max Error in Joint 1 = 0.036    Min Error in Joint 1 = 3.6e-05
Max Error in Joint 2 = 0.016     Min Error in Joint 2 = 0.012
Max Error in Joint 3 = 0.041     Min Error in Joint 3 = 2.3e-05
Max Error in Joint 4 = 0.033      Min Error in Joint 4 = 0.0017
Max Error in Joint 5 = 0.017       Min Error in Joint 5 = 2e-06
Max Error in Joint 6 = 0.027       Min Error in Joint 6 = 0.0002
Absolute Errors in Cartesian Space
Max Error in x-x    = 1.5       Min Error in x-x     = 0.0011
Max Error in x-y    = 1.2      Min Error in x-y     = 0.0025
Max Error in x-z    = 1.4       Min Error in x-z     = 3.6e-05
Max Error in y-x    = 1.3       Min Error in y-x     = 0.00011
Max Error in y-y    = 0.7       Min Error in y-y     = 0.00039
Max Error in y-z    = 1.9       Min Error in y-z     = 4e-05
Max Error in z-x    = 0.011      Min Error in z-x     = 0.0048
Max Error in z-y    = 0.0096       Min Error in z-y     = 0.0042
```

```
Max Error in z-z    = 0.012    Min Error in z-z    = 0.0051
Max Error in x      = 0.047    Min Error in x      = 5.7e-05
Max Error in y      = 3.3     Min Error in y      = 0.00014
Max Error in z      = 0.6     Min Error in z      = 0.0029
>>
```

Appendix D: M-Code for Reconfigurable Model

```matlab
clear
clc
disp('© Luv Aggarwal')
format bank;
syms pi theta1dot theta2dot theta3dot theta4dot theta5dot theta6dot
d1dot d2dot d3dot d4dot d5dot d6dot;

Link_1=input('Enter Link 1 Type Rotational(0) or Translational(1)\n');
Link_2=input('Enter Link 2 Type Rotational(0) or Translational(1)\n');
Link_3=input('Enter Link 3 Type Rotational(0) or Translational(1)\n');
Link_4=input('Enter Link 4 Type Rotational(0) or Translational(1)\n');
Link_5=input('Enter Link 5 Type Rotational(0) or Translational(1)\n');
Link_6=input('Enter Link 6 Type Rotational(0) or Translational(1)\n');

alpha1=input('Input value for alpha1 (degrees)\n');
alpha2=input('Input value for alpha2 (degrees)\n');
alpha3=input('Input value for alpha3 (degrees)\n');
alpha4=input('Input value for alpha4 (degrees)\n');
alpha5=input('Input value for alpha5 (degrees)\n');
alpha6=input('Input value for alpha6 (degrees)\n');

a1=input('Input value for a1(units)\n');
a2=input('Input value for a2(units)\n');
a3=input('Input value for a3(units)\n');
a4=input('Input value for a4(units)\n');
a5=input('Input value for a5(units)\n');
a6=input('Input value for a6(units)\n');


if (Link_1)==0;
    syms theta1
    d1=input('Input value for d1 (units)\n');
    q1_min=double(input('Input value for theta1 minimum
(deg)\n')*pi/180);
    q1_max=double(input('Input value for theta1 maximum
(deg)\n')*pi/180);
    q1dot=theta1dot;
    t1=theta1; % For solving Singularity Equation

else
    syms d1
    theta1=input('Input value for theta1 (degrees)\n');
    q1_min=input('Input value for d1 minimum (units)\n');
    q1_max=input('Input value for d1 maximum (units)\n');
    q1dot=d1dot;
    t1=d1;

end

if Link_2==0;
    syms theta2
```

```matlab
    d2=input('Input value for d2(units)\n');
    q2_min=double(input('Input value for theta2 minimum
(deg)\n')*pi/180);
    q2_max=double(input('Input value for theta2 maximum
(deg)\n')*pi/180);
    q2dot=theta2dot;
    t2=theta2;
else
    syms d2
    theta2=input('Input value for theta2 (degrees)\n');
    q2_min=input('Input value for d2 minimum (units)\n');
    q2_max=input('Input value for d2 maximum (units)\n');
    q2dot=d2dot;
    t2=d2;
end

if Link_3==0;
    syms theta3
    d3=input('Input value for d3(units)\n');
    q3_min=double(input('Input value for theta3 minimum
(deg)\n')*pi/180);
    q3_max=double(input('Input value for theta3 maximum
(deg)\n')*pi/180);
    q3dot=theta3dot;
    t3=theta3;
else
    syms d3
    theta3=input('Input value for theta3 (degrees)\n');
    q3_min=input('Input value for d3 minimum (units)\n');
    q3_max=input('Input value for d3 maximum (units)\n');
    q3dot=d3dot;
    t3=d3;
end

if Link_4==0;
    syms theta4
    d4=input('Input value for d4(units)\n');
    q4_min=double(input('Input value for theta4 minimum
(deg)\n')*pi/180);
    q4_max=double(input('Input value for theta4 maximum
(deg)\n')*pi/180);
    q4dot=theta4dot;
    t4=theta4;
else
    syms d4
    theta4=input('Input value for theta4 (degrees)\n');
    q4_min=input('Input value for d4 minimum (units)\n');
    q4_max=input('Input value for d4 maximum (units)\n');
    q4dot=d4dot;
    t4=d4;
end

if Link_5==0;
    syms theta5
    d5=input('Input value for d5(units)\n');
```

```matlab
    q5_min=double(input('Input value for theta5 minimum
(deg)\n')*pi/180);
    q5_max=double(input('Input value for theta5 maximum
(deg)\n')*pi/180);
    q5dot=theta5dot;
    t5=theta5;
else
    syms d5
    theta5=input('Input value for theta5 (degrees)\n');
    q5_min=input('Input value for d5 minimum (units)\n');
    q5_max=input('Input value for d5 maximum (units)\n');
    q5dot=d5dot;
    t5=d5;
end

if Link_6==0;
    syms theta6
    d6=input('Input value for d6(units)\n');
    q6_min=double(input('Input value for theta6 minimum
(deg)\n')*pi/180);
    q6_max=double(input('Input value for theta6 maximum
(deg)\n')*pi/180);
    q6dot=theta6dot;
    t6=theta6;
else
    syms d6
    theta6=input('Input value for theta6 (degrees)\n');
    q6_min=input('Input value for d6 minimum (units)\n');
    q6_max=input('Input value for d6 maximum (units)\n');
    q6dot=d6dot;
    t6=d6;
end


%Link 1
% disp('Transformation Matrix for Rotational Joint 1')
A01=simplify([cos(theta1) -cos(alpha1*pi/180)*sin(theta1)
sin(alpha1*pi/180)*sin(theta1) a1*cos(theta1);sin(theta1)
cos(alpha1*pi/180)*cos(theta1) -sin(alpha1*pi/180)*cos(theta1)
a1*sin(theta1);0 sin(alpha1*pi/180) cos(alpha1*pi/180) d1;0 0 0 1]);
R01=simplify([A01(1,1) A01(1,2) A01(1,3);A01(2,1) A01(2,2)
A01(2,3);A01(3,1) A01(3,2) A01(3,3)]);
R10=transpose(R01);
P01=[A01(1,4);A01(2,4);A01(3,4)];

% disp('Transformation Matrix for Rotational Joint 2')
A12=simplify([cos(theta2) -cos(alpha2*pi/180)*sin(theta2)
sin(alpha2*pi/180)*sin(theta2) a2*cos(theta2);sin(theta2)
cos(alpha2*pi/180)*cos(theta2) -sin(alpha2*pi/180)*cos(theta2)
a2*sin(theta2);0 sin(alpha2*pi/180) cos(alpha2*pi/180) d2;0 0 0 1]);
R12=simplify([A12(1,1) A12(1,2) A12(1,3);A12(2,1) A12(2,2)
A12(2,3);A12(3,1) A12(3,2) A12(3,3)]);
R21=transpose(R12);
P12=[A12(1,4);A12(2,4);A12(3,4)];

% disp('Transformation Matrix for Rotational Joint 3')
```

```matlab
A23=simplify([cos(theta3) -cos(alpha3*pi/180)*sin(theta3)
sin(alpha3*pi/180)*sin(theta3) a3*cos(theta3);sin(theta3)
cos(alpha3*pi/180)*cos(theta3) -sin(alpha3*pi/180)*cos(theta3)
a3*sin(theta3);0 sin(alpha3*pi/180) cos(alpha3*pi/180) d3;0 0 0 1]);
R23=simplify([A23(1,1) A23(1,2) A23(1,3);A23(2,1) A23(2,2)
A23(2,3);A23(3,1) A23(3,2) A23(3,3)]);
R32=transpose(R23);
P23=[A23(1,4);A23(2,4);A23(3,4)];

% disp('Transformation Matrix for Rotational Joint 4')
A34=simplify([cos(theta4) -cos(alpha4*pi/180)*sin(theta4)
sin(alpha4*pi/180)*sin(theta4) a4*cos(theta4);sin(theta4)
cos(alpha4*pi/180)*cos(theta4) -sin(alpha4*pi/180)*cos(theta4)
a4*sin(theta4);0 sin(alpha4*pi/180) cos(alpha4*pi/180) d4;0 0 0 1]);
R34=simplify([A34(1,1) A34(1,2) A34(1,3);A34(2,1) A34(2,2)
A34(2,3);A34(3,1) A34(3,2) A34(3,3)]);
R43=transpose(R34);
P34=[A34(1,4);A34(2,4);A34(3,4)];

% disp('Transformation Matrix for Rotational Joint 5')
A45=simplify([cos(theta5) -cos(alpha5*pi/180)*sin(theta5)
sin(alpha5*pi/180)*sin(theta5) a5*cos(theta5);sin(theta5)
cos(alpha5*pi/180)*cos(theta5) -sin(alpha5*pi/180)*cos(theta5)
a5*sin(theta5);0 sin(alpha5*pi/180) cos(alpha5*pi/180) d5;0 0 0 1]);
R45=simplify([A45(1,1) A45(1,2) A45(1,3);A45(2,1) A45(2,2)
A45(2,3);A45(3,1) A45(3,2) A45(3,3)]);
R54=transpose(R45);
P45=[A45(1,4);A45(2,4);A45(3,4)];

% disp('Transformation Matrix for Rotational Joint 6')
A56=simplify([cos(theta6) -cos(alpha6*pi/180)*sin(theta6)
sin(alpha6*pi/180)*sin(theta6) a6*cos(theta6);sin(theta6)
cos(alpha6*pi/180)*cos(theta6) -sin(alpha6*pi/180)*cos(theta6)
a6*sin(theta6);0 sin(alpha6*pi/180) cos(alpha6*pi/180) d6;0 0 0 1]);
R56=simplify([A56(1,1) A56(1,2) A56(1,3);A56(2,1) A56(2,2)
A56(2,3);A56(3,1) A56(3,2) A56(3,3)]);
R65=transpose(R56);
P56=[A56(1,4);A56(2,4);A56(3,4)];

%Forward Kinematics
% disp('Forward Kinematics')
A06=simplify(A01*A12*A23*A34*A45*A56);
R06=simplify([A06(1,1) A06(1,2) A06(1,3);A06(2,1) A06(2,2)
A06(2,3);A06(3,1) A06(3,2) A06(3,3)]);
R60=transpose(R06);
P06=[A06(1,4);A06(2,4);A06(3,4)];

% Total Workspace

R006 = R06; % R06 is stored in R006 fot the purpose of calucatinj
Jacobian
P=[A06(1,1);A06(1,2);A06(1,3);A06(2,1);A06(2,2);A06(2,3);A06(3,1);A06(3,
2);A06(3,3);A06(1,4);A06(2,4);A06(3,4)];

% P = A06(:,4);
```

```matlab
syms q1 q2 q3 q4 q5 q6
if (Link_1)==0;
P=subs(P, theta1, q1);
R06=subs(R06,theta1,q1);
else
P=subs(P, d1, q1);
R06=subs(R06,d1,q1);
end

if (Link_2)==0;
P=subs(P, theta2, q2);
R06=subs(R06,theta2,q2);
else
P=subs(P, d2, q2);
R06=subs(R06,d2,q2);
end

if (Link_3)==0;
P=subs(P, theta3, q3);
R06=subs(R06,theta3,q3);
else
P=subs(P, d3, q3);
R06=subs(R06,d3,q3);
end

if (Link_4)==0;
P=subs(P, theta4, q4);
R06=subs(R06,theta4,q4);
else
P=subs(P, d4, q4);
R06=subs(R06,d4,q4);
end

if (Link_5)==0;
P=subs(P, theta5, q5);
R06=subs(R06,theta5,q5);
else
P=subs(P, d5, q5);
R06=subs(R06,d5,q5);
end

if (Link_6)==0;
P=subs(P, theta6, q6);
R06=subs(R06,theta6,q6);
else
P=subs(P, d6, q6);
R06=subs(R06,d6,q6);
end

%Plotting position and orientation
steps = 10;
q1_range = linspace(q1_min, q1_max,steps)';
q2_range = linspace(q2_min, q2_max,steps)';
q3_range = linspace(q3_min, q3_max,steps)';
q4_range = linspace(q4_min, q4_max,steps)';
q5_range = linspace(q5_min, q5_max,steps)';
```

```matlab
q6_range = linspace(q6_min, q6_max,steps)';

angle_config =combvec (q1_range', q2_range', q3_range', q4_range',
q5_range', q6_range')';
fwdkin=zeros((steps)^6,12); % Change if the number of joints change
Q_sym=[q1 q2 q3 q4 q5 q6];

for i=1:length(angle_config)
        Q_set=angle_config(i,:);
             fwdkin(i,:)= double(subs(P,Q_sym,Q_set));
end

K1= [angle_config fwdkin];
% All Angle configurations
Q1 =K1(:,1)';
Q2 =K1(:,2)';
Q3 =K1(:,3)';
Q4 =K1(:,4)';
Q5 =K1(:,5)';
Q6 =K1(:,6)';

% All Orientations about x,y,z
x_x =K1(:,7)';
x_y =K1(:,8)';
x_z =K1(:,9)';
y_x =K1(:,10)';
y_y =K1(:,11)';
y_z =K1(:,12)';
z_x =K1(:,13)';
z_y =K1(:,14)';
z_z =K1(:,15)';

% Cartesian Coordinates x,y,z
x =K1(:,16)';
y =K1(:,17)';
z =K1(:,18)';

figure(1)
subplot(2,2,1);
plot3(x',y',z','o','MarkerSize',15,'MarkerEdgeColor','k','MarkerFaceColo
r','w','LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('X (m)','FontSize',20);
ylabel('Y (m)','FontSize',20);
title('Workspace Top View','FontSize',20);
view([0 90]) % X-Y

subplot(2,2,2);
plot3(x',y',z','o','MarkerSize',15,'MarkerEdgeColor','k','MarkerFaceColo
r','w','LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('X (m)','FontSize',20);
ylabel('Y (m)','FontSize',20);
```

```matlab
zlabel('Z (m)','FontSize',20);
title('Total Workspace of Robot','FontSize',20);
view([45 45 45]) % X-Y-Z

subplot(2,2,3);
plot3(x',y',z','o','MarkerSize',15,'MarkerEdgeColor','k','MarkerFaceColo
r','w','LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('X (m)','FontSize',20);
zlabel('Z (m)','FontSize',20);
title('Workspace Front View','FontSize',20);
view([0 0]) % X-Z

subplot(2,2,4);
plot3(x',y',z','o','MarkerSize',15,'MarkerEdgeColor','k','MarkerFaceColo
r','w','LineWidth',2);
grid on;
set(gca,'fontsize',20)
ylabel('Y (m)','FontSize',20);
zlabel('Z (m)','FontSize',20);
title('Workspace Right View','FontSize',20);
view([90 0]); % Y-Z

% Neural Network Inputs and Targets for training the network
% Step 1: Normalizing all inputs and targets between [-1,1] for IK Soln
[q1_n,PS1] = mapminmax(Q1);
[q2_n,PS2] = mapminmax(Q2);
[q3_n,PS3] = mapminmax(Q3);
[q4_n,PS4] = mapminmax(Q4);
[q5_n,PS5] = mapminmax(Q5);
[q6_n,PS6] = mapminmax(Q6);

[x_x_n,PS7] = mapminmax(x_x);
[x_y_n,PS8] = mapminmax(x_y);
[x_z_n,PS9] = mapminmax(x_z);

[y_x_n,PS10] = mapminmax(y_x);
[y_y_n,PS11] = mapminmax(y_y);
[y_z_n,PS12] = mapminmax(y_z);

[z_x_n,PS13] = mapminmax(z_x);
[z_y_n,PS14] = mapminmax(z_y);
[z_z_n,PS15] = mapminmax(z_z);

[x_n,PS16] = mapminmax(x);
[y_n,PS17] = mapminmax(y);
[z_n,PS18] = mapminmax(z);

input =[x_x_n; x_y_n; x_z_n; y_x_n; y_y_n; y_z_n; z_x_n; z_y_n; z_z_n;
x_n; y_n; z_n];
target =[q1_n; q2_n; q3_n; q4_n; q5_n; q6_n];

% Solve an Input-Output Fitting problem with a Neural Network
% This script assumes these variables are defined:
```

171

```matlab
%
%   input - input data.
%   target - target data.

inputs = input;
targets = target;

% Create a Fitting Network
hiddenLayerSize = [55];
net = fitnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';   % Divide up every sample
net.divideParam.trainRatio = 80/100;
net.divideParam.valRatio = 10/100;
net.divideParam.testRatio = 10/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainlm';  % Levenberg-Marquardt

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);
%Display network weights and bias values
disp 'Input weights ='
net.iw{1,1}
disp 'Layer weights ='
net.lw{2,1}
disp 'Input bias ='
net.b{1}
disp 'Layer bias ='
net.b{2}

%Display network Training parameters
% disp 'Training parameters ='
net.trainParam;
```

```matlab
% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
format short;
Network_Performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};

Training_Performance = perform(net,trainTargets,outputs)
Validation_Performance = perform(net,valTargets,outputs)
Testing_Performance = perform(net,testTargets,outputs)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
% figure, plotperform(tr)
% figure, plottrainstate(tr)
% figure, plotfit(net,inputs,targets)
% figure, plotregression(targets,outputs)
% figure, ploterrhist(errors)

format bank;
% Compare target with network output for IK
q1_np = mapminmax('reverse',outputs(1,:),PS1);
q2_np = mapminmax('reverse',outputs(2,:),PS2);
q3_np = mapminmax('reverse',outputs(3,:),PS3);
q4_np = mapminmax('reverse',outputs(4,:),PS4);
q5_np = mapminmax('reverse',outputs(5,:),PS5);
q6_np = mapminmax('reverse',outputs(6,:),PS6);

figure(2)
subplot(3,2,1);
plot(q1_np,'o','MarkerSize',10,'MarkerEdgeColor','b','MarkerFaceColor','b','LineWidth',2);
hold all
plot(Q1,'o','MarkerSize',10,'MarkerEdgeColor','g','MarkerFaceColor','g','LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('(Dataset Length)','FontSize',20)
ylabel('Joint Variable 1 (rad)','FontSize',20)
title('ANN Accuracy for Joint 1 Inverse Kinematics','FontSize',20)
legend('Predicted','Target')

subplot(3,2,2);
plot(q2_np,'o','MarkerSize',10,'MarkerEdgeColor','b','MarkerFaceColor','b','LineWidth',2);
hold all
```

```matlab
plot(Q2,'o','MarkerSize',10,'MarkerEdgeColor','g','MarkerFaceColor','g',
'LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('(Dataset Length)','FontSize',20)
ylabel('Joint Variable 2 (rad)','FontSize',20)
title('ANN Accuracy for Joint 2 Inverse Kinematics','FontSize',20)
legend('Predicted','Target')

subplot(3,2,3);
plot(q3_np,'o','MarkerSize',10,'MarkerEdgeColor','b','MarkerFaceColor','
b','LineWidth',2);
hold all
plot(Q3,'o','MarkerSize',10,'MarkerEdgeColor','g','MarkerFaceColor','g',
'LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('(Dataset Length)','FontSize',20)
ylabel('Joint Variable 3 (rad)','FontSize',20)
title('ANN Accuracy for Joint 3 Inverse Kinematics','FontSize',20)
legend('Predicted','Target')

subplot(3,2,4);
plot(q4_np,'o','MarkerSize',10,'MarkerEdgeColor','b','MarkerFaceColor','
b','LineWidth',2);
hold all
plot(Q4,'o','MarkerSize',10,'MarkerEdgeColor','g','MarkerFaceColor','g',
'LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('(Dataset Length)','FontSize',20)
ylabel('Joint Variable 4 (rad)','FontSize',20)
title('ANN Accuracy for Joint 4 Inverse Kinematics','FontSize',20)
legend('Predicted','Target')

subplot(3,2,5);
plot(q5_np,'o','MarkerSize',10,'MarkerEdgeColor','b','MarkerFaceColor','
b','LineWidth',2);
hold all
plot(Q5,'o','MarkerSize',10,'MarkerEdgeColor','g','MarkerFaceColor','g',
'LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('(Dataset Length)','FontSize',20)
ylabel('Joint Variable 5 (rad)','FontSize',20)
title('ANN Accuracy for Joint 5 Inverse Kinematics','FontSize',20)
legend('Predicted','Target')

subplot(3,2,6);
plot(q6_np,'o','MarkerSize',10,'MarkerEdgeColor','b','MarkerFaceColor','
b','LineWidth',2);
hold all
plot(Q6,'o','MarkerSize',10,'MarkerEdgeColor','g','MarkerFaceColor','g',
'LineWidth',2);
grid on;
set(gca,'fontsize',20)
```

```matlab
xlabel('(Dataset Length)','FontSize',20)
ylabel('Joint Variable 6 (rad)','FontSize',20)
title('ANN Accuracy for Joint 6 Inverse Kinematics','FontSize',20)
legend('Predicted','Target')

% Residual Error Plot
figure(3)
subplot(3,2,1);
plot(abs(Q1-q1_np),'-r','LineWidth',2);
hold all
grid on;
set(gca,'fontsize',20)
% axis([0 800 0 4.5])
axis([0 800 0 0.01])
xlabel('(Dataset Length)','FontSize',20)
ylabel('Residual Error (rad)','FontSize',20)
title('Residual Error for Joint 1 Inverse Kinematics','FontSize',20)
legend('Error = |Target - Predicted|')

subplot(3,2,2);
plot(abs(Q2-q2_np),'-r','LineWidth',2);
hold all
grid on;
set(gca,'fontsize',20)
% axis([0 800 0 4.5])
axis([0 800 0 0.01])
xlabel('(Dataset Length)','FontSize',20)
ylabel('Residual Error (rad)','FontSize',20)
title('Residual Error for Joint 2 Inverse Kinematics','FontSize',20)
legend('Error = |Target - Predicted|')

subplot(3,2,3);
plot(abs(Q3-q3_np),'-r','LineWidth',2);
hold all
grid on;
set(gca,'fontsize',20)
% axis([0 800 0 4.5])
axis([0 800 0 0.01])
xlabel('(Dataset Length)','FontSize',20)
ylabel('Residual Error (rad)','FontSize',20)
title('Residual Error for Joint 3 Inverse Kinematics','FontSize',20)
legend('Error = |Target - Predicted|')

subplot(3,2,4);
plot(abs(Q4-q4_np),'-r','LineWidth',2);
hold all
grid on;
set(gca,'fontsize',20)
% axis([0 800 0 4.5])
axis([0 800 0 0.01])
xlabel('(Dataset Length)','FontSize',20)
ylabel('Residual Error (rad)','FontSize',20)
title('Residual Error for Joint 4 Inverse Kinematics','FontSize',20)
legend('Error = |Target - Predicted|')

subplot(3,2,5);
```

```matlab
plot(abs(Q5-q5_np),'-r','LineWidth',2);
hold all
grid on;
set(gca,'fontsize',20)
% axis([0 800 0 4.5])
axis([0 800 0 0.01])
xlabel('(Dataset Length)','FontSize',20)
ylabel('Residual Error (rad)','FontSize',20)
title('Residual Error for Joint 5 Inverse Kinematics','FontSize',20)
legend('Error = |Target - Predicted|')

subplot(3,2,6);
plot(abs(Q6-q6_np),'-r','LineWidth',2);
hold all
grid on;
set(gca,'fontsize',20)
% axis([0 800 0 4.5])
axis([0 800 0 0.01])
xlabel('(Dataset Length)','FontSize',20)
ylabel('Residual Error (rad)','FontSize',20)
title('Residual Error for Joint 6 Inverse Kinematics','FontSize',20)
legend('Error = |Target - Predicted|')

% Angular and Linear Velocities

% Joint Angular Velocities
syms pi;

format bank;
qdot01=[0;0;q1dot];
qdot12=[0;0;q2dot];
qdot23=[0;0;q3dot];
qdot34=[0;0;q4dot];
qdot45=[0;0;q5dot];
qdot56=[0;0;q6dot];

omega000=[0;0;0];
v000=[0;0;0];

if Link_1==0
omega101=simplify(R10*(omega000+qdot01));
v101=simplify((R10*v000)+cross(omega101,(R10*P01)));
else
omega101=simplify(R10*(omega000));
v101=simplify((R10*v000)+cross(omega000,(R10*P01))+(R10*qdot01));
end

if Link_2==0
omega202=simplify(R21*(omega101+qdot12));
v202=simplify((R21*v101)+cross(omega202,(R21*P12)));
else
omega202=simplify(R21*omega101);
v202=simplify((R21*v101)+cross(omega101,(R21*P12))+(R21*qdot12));
end
```

176

```matlab
if Link_3==0
omega303=simplify(R32*(omega202+qdot23));
v303=simplify((R32*v202)+cross(omega303,(R32*P23)));
else
omega303=simplify(R32*(omega202));
v303=simplify((R32*v202)+cross(omega202,(R32*P23))+(R32*qdot23));
end


if Link_4==0
omega404=simplify(R43*(omega303+qdot34));
v404=simplify((R43*v303)+cross(omega404,(R43*P34)));
else
omega404=simplify(R43*(omega303));
v404=simplify((R43*v303)+cross(omega303,(R43*P34))+(R43*qdot34));
end


if Link_5==0
omega505=simplify(R54*(omega404+qdot45));
v505=simplify((R54*v404)+cross(omega505,(R54*P45)));
else
omega505=simplify(R54*(omega404));
v505=simplify((R54*v404)+cross(omega404,(R54*P45))+(R54*qdot45));
end


if Link_6==0
omega606=simplify(R65*(omega505+qdot56));
v606=simplify((R65*v505)+cross(omega606,(R65*P56)));
else
omega606=simplify(R65*(omega505));
v606=simplify((R65*v505)+cross(omega505,(R65*P56))+(R65*qdot56));
end


disp('Jacobian in Base Frame')
VE=[v606;omega606];
J_Variable=[q1dot;q2dot;q3dot;q4dot;q5dot;q6dot];
JE=jacobian(VE,J_Variable);
JBv=simplify(R006*[JE(1,1) JE(1,2) JE(1,3) JE(1,4) JE(1,5)
JE(1,6);JE(2,1) JE(2,2) JE(2,3) JE(2,4) JE(2,5) JE(2,6);JE(3,1) JE(3,2)
JE(3,3) JE(3,4) JE(3,5) JE(3,6)]);
JBw=simplify(R006*[JE(4,1) JE(4,2) JE(4,3) JE(4,4) JE(4,5)
JE(4,6);JE(5,1) JE(5,2) JE(5,3) JE(5,4) JE(5,5) JE(5,6);JE(6,1) JE(6,2)
JE(6,3) JE(6,4) JE(6,5) JE(6,6)]);
JB=[JBv;JBw];
J11=[JB(1,1) JB(1,2) JB(1,3); JB(2,1) JB(2,2) JB(2,3); JB(3,1) JB(3,2)
JB(3,3)];
J22=[JB(4,4) JB(4,5) JB(4,6); JB(5,4) JB(5,5) JB(5,6); JB(6,4) JB(6,5)
JB(6,6)];
disp(vpa(JB,5))


%Z_Integers , Q_ = Rational Numbers , R_ = Real Numbers, C_ = Complex
Numbers

if a4+a4+a6==0
   disp('Jacobian Subset J11')
```

```matlab
    disp(vpa(J11,5))
    disp('Jacobian Subset J22')
    disp(vpa(J22,5))
    S1=simplify(det(J11));
    S2=simplify(det(J22));
    fprintf(2,'Singularity Equation\n')
    fprintf(2,'The Robot has a Wrist Configuration\n')
    if S1==0
      fprintf(2,'Robot always has a Forearm Singularity\n')
    else
        fprintf(2,'Forearm Singularity Equation\n')
        disp(S1)

        SE1=
solve(S1==0,t1,t2,t3,t4,t5,t6,'Real',true,'IgnoreProperties',true,'Ignor
eAnalyticConstraints', true);
            if Link_1==0
                t101=SE1.theta1;
            else
                t101=SE1.d1;
            end

            if Link_2==0
                t102=SE1.theta2;
            else
                t102=SE1.d2;
            end

            if Link_3==0
                t103=SE1.theta3;
            else
                t103=SE1.d3;
            end

            if Link_4==0
                t104=SE1.theta4;
            else
                t104=SE1.d4;
            end

            if Link_5==0
                t105=SE1.theta5;
            else
                t105=SE1.d5;
            end

            if Link_6==0
                t106=SE1.theta6;
            else
                t106=SE1.d6;
            end
            fprintf(2,'Forearm Singularity Solution(s)\n')
            SE1 = [t101 t102 t103 t104 t105 t106]
    end
```

```matlab
    if S2==0
      fprintf(2,'Robot always has a Wrist Singularity\n')
    else
        fprintf(2,'Wrist Singularity Equation\n')
        disp(S2)
          SE2=
solve(S2==0,t1,t2,t3,t4,t5,t6,'Real',true,'IgnoreProperties',true,'Ignor
eAnalyticConstraints', true);
            if Link_1==0
                t201=SE2.theta1;
            else
                t201=SE2.d1;
            end

            if Link_2==0
                t202=SE2.theta2;
            else
                t202=SE2.d2;
            end

            if Link_3==0
                t203=SE2.theta3;
            else
                t203=SE2.d3;
            end

            if Link_4==0
                t204=SE2.theta4;
            else
                t204=SE2.d4;
            end

            if Link_5==0
                t205=SE2.theta5;
            else
                t205=SE2.d5;
            end

            if Link_6==0
                t206=SE2.theta6;
            else
                t206=SE2.d6;
            end
            fprintf(2,'Wrist Singularity Solution(s)\n')
            SE2 = [t201 t202 t203 t204 t205 t206]
    end


else
    S3=simplify(det(JB));
    if S3==0
%       fprintf(2,'Robot is always Singular\n')
    else
      disp(S3)
    end
```

```matlab
    fprintf(2,'Singularity Equation\n')
    SE3=
solve(S3==0,t1,t2,t3,t4,t5,t6,'Real',true,'IgnoreProperties',true,'Ignor
eAnalyticConstraints', true);
        if Link_1==0
            t301=SE3.theta1;
        else
            t301=SE3.d1;
        end

        if Link_2==0
            t302=SE3.theta2;
        else
            t302=SE3.d2;
        end

        if Link_3==0
            t303=SE3.theta3;
        else
            t303=SE3.d3;
        end

        if Link_4==0
            t304=SE3.theta4;
        else
            t304=SE3.d4;
        end

        if Link_5==0
            t305=SE3.theta5;
        else
            t305=SE3.d5;
        end

        if Link_6==0
            t306=SE3.theta6;
        else
            t306=SE3.d6;
        end

        fprintf(2,' Singularity Solution(s)\n')

        SE3 = [t301 t302 t303 t304 t305 t306]
end

 %Plotting Singularity
 q1_range_new = q1_range;
 q2_range_new = 0;
 q3_range_new = q3_range;
 q4_range_new = q4_range;
 q5_range_new = q5_range;                         %change fwd_kin dimension
 q6_range_new = q6_range;

angle_config_s =combvec (q1_range_new', q2_range_new', q3_range_new',
q4_range_new', q5_range_new', q6_range_new')';
```

```matlab
fwdkin_s=zeros((steps)^5,12); % change every time
Q_sym=[q1 q2 q3 q4 q5 q6];

for i=1:length(angle_config_s)
        Q_set_s=angle_config_s(i,:);
            fwdkin_s(i,:)= double(subs(P,Q_sym,Q_set_s));
end

K_s= [angle_config_s fwdkin_s];
[~, loc_s] = unique(K_s(:,16:18),'rows');
K1_s=K_s(loc_s,:);

% All Angle configurations
Q1_s =K1_s(:,1)';
Q2_s =K1_s(:,2)';
Q3_s =K1_s(:,3)';
Q4_s =K1_s(:,4)';
Q5_s =K1_s(:,5)';
Q6_s =K1_s(:,6)';

% All Orientations about x,y,z
x_x_s =K1_s(:,7)';
x_y_s =K1_s(:,8)';
x_z_s =K1_s(:,9)';
y_x_s =K1_s(:,10)';
y_y_s =K1_s(:,11)';
y_z_s =K1_s(:,12)';
z_x_s =K1_s(:,13)';
z_y_s =K1_s(:,14)';
z_z_s =K1_s(:,15)';

% Cartesian Coordinates x,y,z
x_s =K1_s(:,16)';
y_s =K1_s(:,17)';
z_s =K1_s(:,18)';

figure(5)
subplot(2,2,1);
plot3(x',y',z','o','MarkerSize',15,'MarkerEdgeColor','k','MarkerFaceColo
r','w','LineWidth',2);
hold all
plot3(x_s',y_s',z_s','o','MarkerSize',15,'MarkerEdgeColor','r','MarkerFa
ceColor','r','LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('X (m)','FontSize',20);
ylabel('Y (m)','FontSize',20);
title('Functional Workspace Top View','FontSize',20);
legend('Workspace','Singularity Space')
view([0 90]) % X-Y

subplot(2,2,2);
plot3(x',y',z','o','MarkerSize',15,'MarkerEdgeColor','k','MarkerFaceColo
r','w','LineWidth',2);
```

```matlab
hold all
plot3(x_s',y_s',z_s','o','MarkerSize',15,'MarkerEdgeColor','r','MarkerFa
ceColor','r','LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('X (m)','FontSize',20);
ylabel('Y (m)','FontSize',20);
zlabel('Z (m)','FontSize',20);
title('Functional Workspace of Robot','FontSize',20);
legend('Workspace','Singularity Space')
view([45 45 45]) % X-Y-Z

subplot(2,2,3);
plot3(x',y',z','o','MarkerSize',15,'MarkerEdgeColor','k','MarkerFaceColo
r','w','LineWidth',2);
hold all
plot3(x_s',y_s',z_s','o','MarkerSize',15,'MarkerEdgeColor','r','MarkerFa
ceColor','r','LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('X (m)','FontSize',20);
zlabel('Z (m)','FontSize',20);
title('Functional Workspace Front View','FontSize',20);
legend('Workspace','Singularity Space')
view([0 0]) % X-Z

subplot(2,2,4);
plot3(x',y',z','o','MarkerSize',15,'MarkerEdgeColor','k','MarkerFaceColo
r','w','LineWidth',2);
hold all
plot3(x_s',y_s',z_s','o','MarkerSize',15,'MarkerEdgeColor','r','MarkerFa
ceColor','r','LineWidth',2);
grid on;
set(gca,'fontsize',20)
ylabel('Y (m)','FontSize',20);
zlabel('Z (m)','FontSize',20);
title('Functional Workspace Right View','FontSize',20);
legend('Workspace','Singularity Space')
view([90 0]); % Y-Z


% Neural Network Inputs and Targets for training the network
% Step 1: Normalizing all inputs and targets between [-1,1] for IK Soln

q1_n_s = mapminmax('apply',Q1_s,PS1);
q2_n_s = mapminmax('apply',Q2_s,PS2);
q3_n_s = mapminmax('apply',Q3_s,PS3);
q4_n_s = mapminmax('apply',Q4_s,PS4);
q5_n_s = mapminmax('apply',Q5_s,PS5);
q6_n_s = mapminmax('apply',Q6_s,PS6);

x_x_n_s = mapminmax('apply',x_x_s,PS7);
x_y_n_s = mapminmax('apply',x_y_s,PS8);
x_z_n_s = mapminmax('apply',x_z_s,PS9);
```

```matlab
y_x_n_s = mapminmax('apply',y_x_s,PS10);
y_y_n_s = mapminmax('apply',y_y_s,PS11);
y_z_n_s = mapminmax('apply',y_z_s,PS12);

z_x_n_s = mapminmax('apply',z_x_s,PS13);
z_y_n_s = mapminmax('apply',z_y_s,PS14);
z_z_n_s = mapminmax('apply',z_z_s,PS15);

x_n_s = mapminmax('apply',x_s,PS16);
y_n_s = mapminmax('apply',y_s,PS17);
z_n_s = mapminmax('apply',z_s,PS18);

input_s =[x_x_n_s; x_y_n_s; x_z_n_s; y_x_n_s; y_y_n_s; y_z_n_s; z_x_n_s;
z_y_n_s; z_z_n_s; x_n_s; y_n_s; z_n_s];
target_s =[q1_n_s; q2_n_s; q3_n_s; q4_n_s; q5_n_s; q6_n_s];

%Simulate network with test data

outputs_p = sim(net,input_s);

q1_p = mapminmax('reverse',outputs_p(1,:),PS1);
q2_p = mapminmax('reverse',outputs_p(2,:),PS2);
q3_p = mapminmax('reverse',outputs_p(3,:),PS3);
q4_p = mapminmax('reverse',outputs_p(4,:),PS4);
q5_p = mapminmax('reverse',outputs_p(5,:),PS5);
q6_p = mapminmax('reverse',outputs_p(6,:),PS6);
angle_config_p = [q1_p', q2_p', q3_p' q4_p', q5_p', q6_p'];

fwdkin_p=zeros(length(input_s),12);          % change every time
% fwdkin_p=zeros((steps)^5,12);  % change every time
Q_sym=[q1 q2 q3 q4 q5 q6];

for i=1:length(angle_config_p)
        Q_set_p=angle_config_p(i,:);
            fwdkin_p(i,:)= double(subs(P,Q_sym,Q_set_p));
end

K_p= [angle_config_p fwdkin_p];
[~, loc] = unique(K_p(:,16:18),'rows');
K1_p=K_p(loc,:);

% All Angle configurations
Q1_p =K1_p(:,1)';
Q2_p =K1_p(:,2)';
Q3_p =K1_p(:,3)';
Q4_p =K1_p(:,4)';
Q5_p =K1_p(:,5)';
Q6_p =K1_p(:,6)';

% All Orientations about x,y,z
x_x_p =K1_p(:,7)';
x_y_p =K1_p(:,8)';
x_z_p =K1_p(:,9)';
y_x_p =K1_p(:,10)';
y_y_p =K1_p(:,11)';
```

```
y_z_p =K1_p(:,12)';
z_x_p =K1_p(:,13)';
z_y_p =K1_p(:,14)';
z_z_p =K1_p(:,15)';

% Cartesian Coordinates x,y,z

x_p =K1_p(:,16)';
y_p =K1_p(:,17)';
z_p =K1_p(:,18)';

figure(6)
subplot(2,2,1)
plot3(x_p',y_p',z_p','o','MarkerSize',20,'MarkerEdgeColor','b','MarkerFa
ceColor','b','LineWidth',2);
hold all
plot3(x_s',y_s',z_s','o','MarkerSize',20,'MarkerEdgeColor','r','MarkerFa
ceColor','r','LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('X (m)','FontSize',20);
ylabel('Y (m)','FontSize',20);
title('ANN Singularity Top View','FontSize',20)
legend('Predicted Singularity','Theoretical Singularity')
view([0 90]) % X-Y

subplot(2,2,2);
plot3(x_p',y_p',z_p','o','MarkerSize',20,'MarkerEdgeColor','b','MarkerFa
ceColor','b','LineWidth',2);
hold all
plot3(x_s',y_s',z_s','o','MarkerSize',20,'MarkerEdgeColor','r','MarkerFa
ceColor','r','LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('X (m)','FontSize',20);
ylabel('Y (m)','FontSize',20);
zlabel('Z (m)','FontSize',20);
title('ANN Predicted Singularity vs Theoretical
Singularity','FontSize',20)
legend('Predicted Singularity','Theoretical Singularity')
view([45 45 45]) % X-Y-Z

subplot(2,2,3);
plot3(x_p',y_p',z_p','o','MarkerSize',20,'MarkerEdgeColor','b','MarkerFa
ceColor','b','LineWidth',2);
hold all
plot3(x_s',y_s',z_s','o','MarkerSize',20,'MarkerEdgeColor','r','MarkerFa
ceColor','r','LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('X (m)','FontSize',20);
zlabel('Z (m)','FontSize',20);
title('ANN Singularity Front View','FontSize',20)
legend('Predicted Singularity','Theoretical Singularity')
view([0 0]) % X-Z
```

```matlab
subplot(2,2,4);
plot3(x_p',y_p',z_p','o','MarkerSize',20,'MarkerEdgeColor','b','MarkerFa
ceColor','b','LineWidth',2);
hold all
plot3(x_s',y_s',z_s','o','MarkerSize',20,'MarkerEdgeColor','r','MarkerFa
ceColor','r','LineWidth',2);
grid on;
set(gca,'fontsize',20)
ylabel('Y (m)','FontSize',20);
zlabel('Z (m)','FontSize',20);
title('ANN Singularity Right View','FontSize',20)
legend('Predicted Singularity','Theoretical Singularity')
view([90 0]); % Y-Z

% Absolute Error
E_q1_p_s = abs(Q1_s'-q1_p');
E_q2_p_s = abs(Q2_s'-q2_p');
E_q3_p_s = abs(Q3_s'-q3_p');
E_q4_p_s = abs(Q4_s'-q4_p');
E_q5_p_s = abs(Q5_s'-q5_p');
E_q6_p_s = abs(Q6_s'-q6_p');


E_x_x_p_s = abs(x_x_s' - x_x_p')';
E_x_y_p_s = abs(x_y_s' - x_y_p')';
E_x_z_p_s = abs(x_z_s' - x_z_p')';
E_y_x_p_s = abs(y_x_s' - y_x_p')';
E_y_y_p_s = abs(y_y_s' - y_y_p')';
E_y_z_p_s = abs(y_z_s' - y_z_p')';
E_z_x_p_s = abs(z_x_s' - z_x_p')';
E_z_y_p_s = abs(z_y_s' - z_y_p')';
E_z_z_p_s = abs(z_z_s' - z_z_p')';


E_x_p_s = abs(x_s' - x_p')';
E_y_p_s = abs(y_s' - y_p')';
E_z_p_s = abs(z_s' - z_p')';
disp('Absolute Errors in Joint Space')
disp(['Max Error in Joint 1 = ' num2str(max(E_q1_p_s),2)  '   Min Error
in Joint 1 = ' num2str(min(E_q1_p_s),2)])
disp(['Max Error in Joint 2 = ' num2str(max(E_q2_p_s),2)  '    Min Error
in Joint 2 = ' num2str(min(E_q2_p_s),2)])
disp(['Max Error in Joint 3 = ' num2str(max(E_q3_p_s),2)  '    Min Error
in Joint 3 = ' num2str(min(E_q3_p_s),2)])
disp(['Max Error in Joint 4 = ' num2str(max(E_q4_p_s),2)  '    Min
Error in Joint 4 = ' num2str(min(E_q4_p_s),2)])
disp(['Max Error in Joint 5 = ' num2str(max(E_q5_p_s),2)  '    Min
Error in Joint 5 = ' num2str(min(E_q5_p_s),2)])
disp(['Max Error in Joint 6 = ' num2str(max(E_q6_p_s),2)  '    Min
Error in Joint 6 = ' num2str(min(E_q6_p_s),2)])


disp('Absolute Errors in Cartesian Space')
disp(['Max Error in x-x     = ' num2str(max(E_x_x_p_s),2) '    Min
Error in x-x     = ' num2str(min(E_x_x_p_s),2)])
disp(['Max Error in x-y     = ' num2str(max(E_x_y_p_s),2) '    Min Error
in x-y     = ' num2str(min(E_x_y_p_s),2)])
disp(['Max Error in x-z     = ' num2str(max(E_x_z_p_s),2) '    Min
Error in x-z     = ' num2str(min(E_x_z_p_s),2)])
```

185

```matlab
disp(['Max Error in y-x     = ' num2str(max(E_y_x_p_s),2) '    Min
Error in y-x     = ' num2str(min(E_y_x_p_s),2)])
disp(['Max Error in y-y     = ' num2str(max(E_y_y_p_s),2) '    Min
Error in y-y     = ' num2str(min(E_y_y_p_s),2)])
disp(['Max Error in y-z     = ' num2str(max(E_y_z_p_s),2) '    Min
Error in y-z     = ' num2str(min(E_y_z_p_s),2)])
disp(['Max Error in z-x     = ' num2str(max(E_z_x_p_s),2) '   Min Error
in z-x     = ' num2str(min(E_z_x_p_s),2)])
disp(['Max Error in z-y     = ' num2str(max(E_z_y_p_s),2) '    Min
Error in z-y     = ' num2str(min(E_z_y_p_s),2)])
disp(['Max Error in z-z     = ' num2str(max(E_z_z_p_s),2) '    Min
Error in z-z     = ' num2str(min(E_z_z_p_s),2)])
disp(['Max Error in x       = ' num2str(max(E_x_p_s),2)   '    Min
Error in x       = ' num2str(min(E_x_p_s),2)])
disp(['Max Error in y       = ' num2str(max(E_y_p_s),2)   '    Min
Error in y       = ' num2str(min(E_y_p_s),2)])
disp(['Max Error in z       = ' num2str(max(E_z_p_s),2)   '    Min
Error in z       = ' num2str(min(E_z_p_s),2)])

disp('Absolute Errors in Joint Space')
disp(['Average Error in Joint 1 = ' num2str(mean(E_q1_p_s),2)])
disp(['Average Error in Joint 2 = ' num2str(mean(E_q2_p_s),2)])
disp(['Average Error in Joint 3 = ' num2str(mean(E_q3_p_s),2)])
disp(['Average Error in Joint 4 = ' num2str(mean(E_q4_p_s),2)])
disp(['Average Error in Joint 5 = ' num2str(mean(E_q5_p_s),2)])
disp(['Average Error in Joint 6 = ' num2str(mean(E_q6_p_s),2)])

figure(7)
subplot(3,2,1);
plot(q1_p,'o','MarkerSize',10,'MarkerEdgeColor','b','MarkerFaceColor','b
','LineWidth',2);
hold all
plot(Q1_s,'o','MarkerSize',10,'MarkerEdgeColor','g','MarkerFaceColor','g
','LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('(Dataset Length)','FontSize',20)
ylabel('Joint Variable 1 (rad)','FontSize',20)
title('ANN Accuracy for Joint 1 Singularity Prediction','FontSize',20)
legend('Predicted','Target')

subplot(3,2,2);
plot(q2_p,'o','MarkerSize',10,'MarkerEdgeColor','b','MarkerFaceColor','b
','LineWidth',2);
hold all
plot(Q2_s,'o','MarkerSize',10,'MarkerEdgeColor','g','MarkerFaceColor','g
','LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('(Dataset Length)','FontSize',20)
ylabel('Joint Variable 2 (rad)','FontSize',20)
title('ANN Accuracy for Joint 2 Singularity Prediction','FontSize',20)
legend('Predicted','Target')

subplot(3,2,3);
```

```matlab
plot(q3_p,'o','MarkerSize',10,'MarkerEdgeColor','b','MarkerFaceColor','b
','LineWidth',2);
hold all
plot(Q3_s,'o','MarkerSize',10,'MarkerEdgeColor','g','MarkerFaceColor','g
','LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('(Dataset Length)','FontSize',20)
ylabel('Joint Variable 3 (rad)','FontSize',20)
title('ANN Accuracy for Joint 3 Singularity Prediction','FontSize',20)
legend('Predicted','Target')

subplot(3,2,4);
plot(q4_p,'o','MarkerSize',10,'MarkerEdgeColor','b','MarkerFaceColor','b
','LineWidth',2);
hold all
plot(Q4_s,'o','MarkerSize',10,'MarkerEdgeColor','g','MarkerFaceColor','g
','LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('(Dataset Length)','FontSize',20)
ylabel('Joint Variable 4 (rad)','FontSize',20)
title('ANN Accuracy for Joint 4 Singularity Prediction','FontSize',20)
legend('Predicted','Target')

subplot(3,2,5);
plot(q5_p,'o','MarkerSize',10,'MarkerEdgeColor','b','MarkerFaceColor','b
','LineWidth',2);
hold all
plot(Q5_s,'o','MarkerSize',10,'MarkerEdgeColor','g','MarkerFaceColor','g
','LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('(Dataset Length)','FontSize',20)
ylabel('Joint Variable 5 (rad)','FontSize',20)
title('ANN Accuracy for Joint 5 Singularity Prediction','FontSize',20)
legend('Predicted','Target')

subplot(3,2,6);
plot(q6_p,'o','MarkerSize',10,'MarkerEdgeColor','b','MarkerFaceColor','b
','LineWidth',2);
hold all
plot(Q6_s,'o','MarkerSize',10,'MarkerEdgeColor','g','MarkerFaceColor','g
','LineWidth',2);
grid on;
set(gca,'fontsize',20)
xlabel('(Dataset Length)','FontSize',20)
ylabel('Joint Variable 6 (rad)','FontSize',20)
title('ANN Output vs. Target for Joint 6','FontSize',20)
legend('Predicted','Target')

%Residual Error Plot

figure(8)
subplot(3,2,1);
plot(abs(Q1_s-q1_p),'-r','LineWidth',2);
```

```matlab
hold all
grid on;
set(gca,'fontsize',20)
% axis([0 30 0 4.5])
axis([0 250 0 0.04])
xlabel('(Dataset Length)','FontSize',20)
ylabel('Residual Error (rad)','FontSize',20)
title('Error in ANN Accuracy for Joint 1 Singularity
Prediction','FontSize',20)
legend('Error = |Target - Predicted|')

subplot(3,2,2);
plot(abs(Q2_s-q2_p),'-r','LineWidth',2);
hold all
grid on;
set(gca,'fontsize',20)
% axis([0 30 0 4.5])
axis([0 250 0 0.04])
xlabel('(Dataset Length)','FontSize',20)
ylabel('Residual Error (rad)','FontSize',20)
title('Error in ANN Accuracy for Joint 2 Singularity
Prediction','FontSize',20)
legend('Error = |Target - Predicted|')

subplot(3,2,3);
plot(abs(Q3_s-q3_p),'-r','LineWidth',2);
hold all
grid on;
set(gca,'fontsize',20)
% axis([0 30 0 4.5])
axis([0 250 0 0.04])
xlabel('(Dataset Length)','FontSize',20)
ylabel('Residual Error (rad)','FontSize',20)
title('Error in ANN Accuracy for Joint 3 Singularity
Prediction','FontSize',20)
legend('Error = |Target - Predicted|')

subplot(3,2,4);
plot(abs(Q4_s-q4_p),'-r','LineWidth',2);
hold all
grid on;
set(gca,'fontsize',20)
% axis([0 30 0 4.5])
axis([0 250 0 0.04])
xlabel('(Dataset Length)','FontSize',20)
ylabel('Residual Error (rad)','FontSize',20)
title('Error in ANN Accuracy for Joint 4 Singularity
Prediction','FontSize',20)
legend('Error = |Target - Predicted|')

subplot(3,2,5);
plot(abs(Q5_s-q5_p),'-r','LineWidth',2);
hold all
grid on;
set(gca,'fontsize',20)
% axis([0 30 0 4.5])
```

```matlab
axis([0 250 0 0.04])
xlabel('(Dataset Length)','FontSize',20)
ylabel('Residual Error (rad)','FontSize',20)
title('Error in ANN Accuracy for Joint 5 Singularity
Prediction','FontSize',20)
legend('Error = |Target - Predicted|')

subplot(3,2,6);
plot(abs(Q6_s-q6_p),'-r','LineWidth',2);
hold all
grid on;
set(gca,'fontsize',20)
% axis([0 30 0 4.5])
axis([0 250 0 0.04])
xlabel('(Dataset Length)','FontSize',20)
ylabel('Residual Error (rad)','FontSize',20)
title('Error in ANN Accuracy for Joint 6 Singularity
Prediction','FontSize',20)
legend('Error = |Target - Predicted|')
```

Appendix E: Permission from SAE to Reprint Paper 2014-01-0734

Dear Luv,

Thank you for your correspondence requesting permission to reprint SAE paper 2014-01-0734 – which you co-authored – in your MASc thesis titled 'Reconfigurable Validation Model for Identifying Kinematic Singularities and Reach Conditions within a Work Cell' for University of Windsor. ON, Canada.

Permission is hereby granted, and subject to the following conditions:

- Permission is for this one-time single use only. New requests are required for further use or distribution of the SAE material.
- The following credit statement must appear on the paper: "Reprinted with permission Copyright © 2014 SAE International. This paper may not be printed, copied, distributed or forwarded without prior permission from SAE."
- We also request that you credit the original source (author, paper number and SAE) in the reference section of your thesis.
- This permission does not cover any third party copyrighted work which may appear in the material requested.

Please feel free to contact me if you need further assistance. Good luck with your thesis!

Best regards,

_____

Terri Kelly
Intellectual Property Rights Administrator

SAE INTERNATIONAL
400 Commonwealth Drive
Warrendale, PA 15096

o +1.724.772.4095
f +1.724-776-9765
e terri@sae.org
www.sae.org

Appendix F: Permission from Procedia CIRP to Reprint Paper 17 (2014) 812 – 817

Dear Mr. Aggarwal,

Elsevier is very pleased to announce that starting from February 1st, 2014 Procedia journal will be published under Creative commons license, in particular under *Creative Commons Attribution Non-Commercial No Derivatives license* (CC-BY-NC-ND). CC-BY-NC-ND license gives the authors similar rights to the ones under Procedia Exclusive License agreement.

Under CC-BY-NC-ND **the authors** would retain:
• Copyright of the article
• Patent, trademark and other intellectual property rights in the article
• The right for proper attribution and credit for the published work
• The right to reuse their own work in the same way readers can as defined by CC-BY-NC-ND license.
Under CC-BY-NC-ND **the users** are allowed to copy and distribute the article, provided this is not done for commercial purposes and the article is not changed or edited in any way. The author must be attributed and must not be represented as endorsing the use made of the work. This also does not allow users to text or data mine the article. You are invited to visit Elsevier open access page to learn
more: http://www.elsevier.com/about/open-access/open-access-policies/oa-license-policy?a=133551.
Please share this information with Procedia authors. The authors will automatically receive a copy of creative commons license that they will need to sign and send back by email to Elsevier.
Kind Regards,
Jeniv

Jeniv Praveen Kumar
Journal Manager - Global Journals Production
Elsevier India
(A division of Reed Elsevier India Pvt. Ltd.)

International Tech Park | Crest – 12th Floor | Taramani Road | Taramani | Chennai 600 113 | India
Tel: +91 44 42994854 |
E-mail: j.praveenkumar@elsevier.com; url: www.elsevier.com
Line Manager: L.Rajaram@elsevier.com

VITA AUCTORIS

NAME: Luv Aggarwal

PLACE OF BIRTH: Rohtak, Haryana, India

YEAR OF BIRTH: 1991

EDUCATION: University of Windsor, B.A.Sc. Industrial
Engineering with Minor in Business
Administration (Hons.) Co-op, Windsor, ON,
2012

University of Windsor, M.A.Sc. Mechanical
Engineering, Windsor, ON, 2014

Lean Six Sigma Black Belt, Certification,
Lawrence Technological University, Southfield,
Michigan, 2014