

# **Use of Jacobians for Inverse Kinematics of Articulated Robots: A Study on Approximate Solutions**

---

A Thesis  
presented to  
the Faculty of the Graduate School  
University of Missouri-Columbia

---

In Partial Fulfillment  
Of the Requirements for the Degree  
Master in Science

---

by

**Uriel Jacket Trésor Demby's**

Dr. Guilherme N. DeSouza, Thesis Advisor

July 2020

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

USE OF JACOBIANS FOR INVERSE KINEMATICS OF  
ARTICULATED ROBOTS: A STUDY ON APPROXIMATE  
SOLUTIONS

Presented by Uriel Jacket Trésor Demby's,  
a candidate for the degree of Master of Science,  
and hereby certify that, in their opinion, it is worthy of acceptance.

---

Dr. Guilherme N. DeSouza, Associate Professor, Dept. of Electrical Engineering and  
Computer Science

---

Dr. Jeffrey Uhlmann, Associate Professor, Dept. of Electrical Engineering and Computer  
Science

---

Dr. Jung Hyup Kim, Assistant Professor, Dept. of Industrial and Manufacturing Systems  
Engineering

To my parents for their constant encouragements.

## ACKNOWLEDGMENTS

Graduate school has been the beginning of an interesting research journey that has led me to develop interests in robotics and artificial intelligence related topics. And I could not think of a better place than the Vision Guided and Intelligent Robotics (ViGIR) Laboratory in the Department of Electrical Engineering and Computer Science at the University of Missouri-Columbia to get started with it.

I would like to express my most sincere gratitude and utmost respect to Dr. Guilherme DeSouza, my academic advisor, for his valuable guidance and constant support in the work presented in this document. I am also grateful for his effort and challenging attitude to ensure my personal development and the improvement of my skills at every step along the way.

I would also like to thank Dr. Jeffrey Uhlmann and Dr. Jung Hyup Kim. The former, for his valuable explanations that have been helpful to guide some parts of the study presented here. And the latter, for agreeing to be part of my master thesis committee and review the work presented in this document.

Finally, I am grateful to the PhD students in ViGIR Laboratory; Ali Shafiekhani, Tushar Kanta Das Nakini and Yixiang Gao, that have supported, advised and challenged me throughout my master program.



# Contents

ACKNOWLEDGMENTS . . . . .	ii
LIST OF FIGURES . . . . .	vi
LIST OF TABLES . . . . .	xiv
ABSTRACT . . . . .	xviii
<b>1 Introduction</b>	<b>1</b>
<b>2 Terminology, Background and Related Work</b>	<b>4</b>
2.1 Serial Robots . . . . .	4
2.2 Analytical Methods . . . . .	8
2.3 Approximate Methods . . . . .	8
2.3.1 Data-driven Methods . . . . .	8
2.3.1.1 Survey on Data-driven Methods . . . . .	9
2.3.2 Numerical Methods . . . . .	12
2.3.2.1 Jacobian . . . . .	13
2.3.2.2 Matrix Inverse and Generalized Inverse . . . . .	14
2.3.2.3 Survey on Numerical Methods and Incommensurate Sys- tems . . . . .	16
<b>3 Proposed Approaches</b>	<b>18</b>
3.1 Data-driven methods . . . . .	18
3.1.1 Artificial and Fuzzy Neural Networks . . . . .	18
3.1.1.1 Multilayer-Perceptron . . . . .	20
3.1.1.2 Fuzzy Neural Networks . . . . .	22
3.1.2 Dataset generation . . . . .	24

3.1.3	Learning schemes . . . . .	25
3.2	Numerical method . . . . .	25
<b>4</b>	<b>Experimental Results and Discussions</b>	<b>29</b>
4.1	Data-Driven IK Methods . . . . .	29
4.1.1	Experimental Setup and Selected Robotic Arms . . . . .	29
4.1.2	Workspaces and Dataset Generation: . . . . .	31
4.1.3	Network Architecture Search . . . . .	35
4.1.3.1	All-Joints-ANN . . . . .	35
4.1.3.2	Individual-Joints-ANN . . . . .	42
4.1.3.3	Individual-Joints-ANFIS . . . . .	43
4.1.4	Experimental Results for the Best NN Architectural Choices . . . . .	46
4.1.5	Observations on the Experiments . . . . .	67
4.2	Numerical IK methods . . . . .	69
4.2.1	Experimental Setup and Selected Robotic Arms: . . . . .	69
4.2.2	Experimental Results: . . . . .	71
4.2.2.1	3 DoF Serial Robot . . . . .	73
4.2.2.2	4 DoF Serial Robot . . . . .	90
4.2.2.3	6 DoF Serial Robot . . . . .	111
4.2.3	Observations on the Experiments on Numeric Methods . . . . .	133
4.3	Comparison . . . . .	133
<b>5</b>	<b>Conclusion and Future Work</b>	<b>135</b>
	<b>Bibliography</b>	<b>137</b>
	<b>Appendices</b>	<b>144</b>
<b>A</b>	<b>Optimization Techniques</b>	<b>144</b>
A.1	Optimization methods for gradient descent algorithm . . . . .	144
A.1.1	Serial Gradient Descent . . . . .	144
A.1.2	Momentum . . . . .	145
A.1.3	Nesterov Momentum . . . . .	145

A.1.4	Adaptive Gradient (Adagrad)	146
A.1.5	RMSProp	146
A.1.6	Adaptive moments (Adam)	147
A.2	Optimization methods for the network initial predictions	148

# List of Figures

2.1	<i>Examples of different configurations of serial robotic manipulators [1]. . . . .</i>	5
2.2	<i>Mapping provided by the forward (<math>f_{FK}</math>) and inverse kinematics (<math>f_{IK}</math>) between Joint and Cartesian spaces. . . . .</i>	6
2.3	<i>An illustration of the D-H representation with the frames attached to each joint of a 6DoF Kinova robotic arm. . . . .</i>	7
2.4	<i>Taxonomy of the inverse kinematics methods. . . . .</i>	7
2.5	<i>An overview of how Jacobian based numerical methods rely on small perturbations applied to individual joints. . . . .</i>	13
3.1	<i>The Neural Networks Zoo compiled in [2]. . . . .</i>	19
3.2	<i>Network architectures of a Multilayer-Perceptron (MLP). . . . .</i>	20
3.3	<i>A fuzzy inference system organized in functional units as described in [3]. . . . .</i>	22
3.4	<i>Takagi-Sugeno Type-3 ANFIS with 2 inputs [3]. . . . .</i>	23
3.5	<i>Overview of various workspaces generated for a 6DoF Jaco Robotic Arm from Kinova Robotics. . . . .</i>	25
3.6	<i>Visual representation of the hybrid inverse kinematics solver proposed in [4]. . . . .</i>	27
4.1	<i>Robotic arms used in the Data-Driven Experiment in Section 4.1. . . . .</i>	30
4.2	<i>A depiction of the positions of the end-effector in the workspaces of the chosen robots for the structured dataset. The shapes of the data points in the workspaces depended mainly on the joint ranges presented in Table 4.3. . . . .</i>	33
4.3	<i>A depiction of the positions of the end-effector in the workspaces of the chosen robots for the random dataset. The shapes of the data points in the workspaces depended mainly on the joint ranges presented in Table 4.4. . . . .</i>	34

4.5	<i>Representation of the network architecture of the All-Joints-ANN. ANNs were implemented with Keras [5] using TensorFlow CPU backend running on a desktop with an Intel(R) Core(TM) i7-8700CPU and NVIDIA GTX 2060 GPU. . . . .</i>	35
4.4	<i>Overview of the learning process used in the implementation of the data-driven IK method. . . . .</i>	36
4.6	<i>Process allowing to search for the final network architectures used for the experiments. . . . .</i>	38
4.7	<i>Averaged training and validation losses from the search process of finding the final network architecture to be used for the experiments with the 4DoF robot. . . . .</i>	39
4.8	<i>Averaged training and validation losses from the search process of finding the final network architecture to be used for the experiments with the 5DoF robot. . . . .</i>	40
4.9	<i>Averaged training and validation losses from the search process of finding the final network architecture to be used for the experiments with the 6DoF robot. . . . .</i>	41
4.10	<i>Averaged training and validation losses from the search process of finding the final network architecture to be used for the experiments with the 7DoF robot. . . . .</i>	42
4.11	<i>Representation of the n network architectures for the Individual-Joints-ANN case study. ANNs were implemented with Keras [5] using TensorFlow CPU backend running on a desktop with an Intel(R) Core(TM) i7-8700CPU and NVIDIA GTX 2060 GPU. . . . .</i>	43
4.12	<i>Architecture of the Adaptive Neuro-Fuzzy Inference System (ANFIS) network used in this case study, with 7 inputs and 1 output. This structure was repeated n times: one for each DoF of the manipulator in question. . . . .</i>	44
4.13	<i>Visual depiction of the error values in Table 4.11 for the 4 DoF robot using the structured dataset. . . . .</i>	48
4.14	<i>Graphical summary of the maximum error values found in Tables 4.11, 4.13, 4.15 and 4.17 based on the structured datasets for each of the four robots and using all three neural network architectures in the case studies. . . . .</i>	52

4.15	<i>A visualization of the error in positioning of the end-effector in the workspace of the 4DoF robot related to the results summarized in Table 4.11 for the structured dataset. The points in blue color represent the desired positions and those in red color represent the predicted positions. While most predictions were close to their desired positions, not all of them fall within the accepted mm ranges required for several critical IK applications. . . .</i>	53
4.16	<i>A visualization of the error in positioning of the end-effector in the workspace of the 5DoF robot related to the results summarized in Table 4.13 for the structured dataset. The points in blue color represent the desired positions and those in red color represent the predicted positions. Once again, while most predictions were close to their desired positions, not all of them fall within the accepted mm ranges required for several critical IK applications. . . . .</i>	54
4.17	<i>A visualization of the error in positioning of the end-effector in the workspace of the 6DoF robot related to the results summarized in Table 4.15 for the structured dataset. The points in blue color represent the desired positions and those in red color represent the predicted positions. Once again, while most predictions were close to their desired positions, not all of them fall within the accepted mm ranges required for several critical IK applications. . . . .</i>	55
4.18	<i>A visualization of the error in positioning of the end-effector in the workspace of the 7DoF robot related to the results summarized in Table 4.17 for the structured dataset. The points in blue color represent the desired positions and those in red color represent the predicted positions. Once again, while most predictions were close to their desired positions, not all of them fall within the accepted mm ranges required for several critical IK applications. . . . .</i>	56
4.19	<i>Visual depiction of the error values in Table 4.19 for the 4 DoF robot using the random dataset. . . . .</i>	59

4.20	<i>Graphical summary of the maximum error values found in Tables 4.19, 4.21, 4.23 and 4.25 based on the random datasets for each of the four robots and using all three neural network architectures in the case studies.</i>	63
4.21	<i>A visualization of the error in positioning of the end-effector in the workspace of the 4DoF robot related to the results summarized in Table 4.19 for the random dataset. The points in blue color represent the desired positions and those in red color represent the predicted positions. Here, the results were better than for the structured dataset, but still not all of them were within the accepted mm ranges required for several critical IK applications.</i>	64
4.22	<i>A visualization of the error in positioning of the end-effector in the workspace of the 5DoF robot related to the results summarized in Table 4.21 for the random dataset. The points in blue color represent the desired positions and those in red color represent the predicted positions. Once again, the results presented here were better than for the structured dataset, but still not all of them were within the accepted mm ranges required for several critical IK applications.</i>	65
4.23	<i>A visualization of the error in positioning of the end-effector in the workspace of the 6DoF robot related to the results summarized in Table 4.23 for the random dataset. The points in blue color represent the desired positions and those in red color represent the predicted positions. Once again, the results presented here were better than for the structured dataset, but still not all of them were within the accepted mm ranges required for several critical IK applications.</i>	66
4.24	<i>A visualization of the error in positioning of the end-effector in the workspace of the 7DoF robot related to the results summarized in Table 4.25 for the random dataset. The points in blue color represent the desired positions and those in red color represent the predicted positions. Once again, the results presented here were better than for the structured dataset, but still not all of them were within the accepted mm ranges required for several critical IK applications.</i>	67

4.25	<i>A visualization of a prediction example error calculated between the predicted and desired end-effector positions of the 4DoF robot in the case of the data-driven IK method.</i>	68
4.26	<i>Incommensurate robotic arms used in the experiments related to numerical methods.</i>	70
4.27	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 3DoF robot when varying the units while using the MP Inverse with <math>\alpha = 1</math>.</i>	75
4.28	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 3DoF robot when varying the units while using the MP Inverse, for multiple values of <math>\alpha</math>.</i>	76
4.29	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 3DoF robot when varying the units while using the UC Inverse and <math>\alpha = 1</math>.</i>	77
4.30	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 3DoF robot when varying the units while using the UC Inverse, for multiple values of <math>\alpha</math>.</i>	78
4.31	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 3DoF robot when varying the units while using the MX Inverse and <math>\alpha = 1</math>.</i>	79
4.32	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 3DoF robot when varying the units while using the MX Inverse, for multiple values of <math>\alpha</math>.</i>	80
4.33	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 3DoF robot when varying the units while using the MP Inverse and <math>\alpha = 1</math>.</i>	81
4.34	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 3DoF robot when varying the units while using the MP Inverse, for multiple values of <math>\alpha</math>.</i>	82
4.35	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 3DoF robot when varying the units while using the MP Inverse and <math>\alpha = 1</math>.</i>	83
4.36	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 3DoF robot when varying the units while using the MP Inverse, for multiple values of <math>\alpha</math>.</i>	83
4.37	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 3DoF robot when varying the units while using the UC Inverse and <math>\alpha = 1</math>.</i>	84
4.38	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 3DoF robot when varying the units while using the UC Inverse, for multiple values of <math>\alpha</math>.</i>	85
4.39	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 3DoF robot when varying the units while using the UC Inverse and <math>\alpha = 1</math>.</i>	86



4.40	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 3DoF robot when varying the units while using the UC Inverse, for multiple values of <math>\alpha</math>.</i>	86
4.41	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 3DoF robot when varying the units while using the MX Inverse and <math>\alpha = 1</math>.</i>	87
4.42	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 3DoF robot when varying the units while using the MX Inverse, for multiple values of <math>\alpha</math>.</i>	88
4.43	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 3DoF robot when varying the units while using the MX Inverse and <math>\alpha = 1</math>.</i>	89
4.44	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 3DoF robot when varying the units while using the MX Inverse, for multiple values of <math>\alpha</math>.</i>	89
4.45	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 4DoF robot when varying the units while using the MP Inverse with <math>\alpha = 1</math>.</i>	93
4.46	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 4DoF robot when varying the units while using the MP Inverse, for multiple values of <math>\alpha</math>.</i>	94
4.47	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 4DoF robot when varying the units while using the UC Inverse and <math>\alpha = 1</math>.</i>	96
4.48	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 4DoF robot when varying the units while using the UC Inverse, for multiple values of <math>\alpha</math>.</i>	97
4.49	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 4DoF robot when varying the units while using the MX Inverse and <math>\alpha = 1</math>.</i>	99
4.50	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 4DoF robot when varying the units while using the MX Inverse, for multiple values of <math>\alpha</math>.</i>	99
4.51	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 4DoF robot when varying the units while using the MP Inverse and <math>\alpha = 1</math>.</i>	101
4.52	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 4DoF robot when varying the units while using the MP Inverse, for multiple values of <math>\alpha</math>.</i>	101
4.53	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 4DoF robot when varying the units while using the MP Inverse and <math>\alpha = 1</math>.</i>	103
4.54	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 4DoF robot when varying the units while using the MP Inverse, for multiple values of <math>\alpha</math>.</i>	103

4.55	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 4DoF robot when varying the units while using the UC Inverse and <math>\alpha = 1</math>.</i>	105
4.56	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 4DoF robot when varying the units while using the UC Inverse, for multiple values of <math>\alpha</math>.</i>	105
4.57	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 4DoF robot when varying the units while using the UC Inverse and <math>\alpha = 1</math>.</i>	107
4.58	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 4DoF robot when varying the units while using the UC Inverse, for multiple values of <math>\alpha</math>.</i>	107
4.59	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 4DoF robot when varying the units while using the MX Inverse and <math>\alpha = 1</math>.</i>	109
4.60	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 4DoF robot when varying the units while using the MX Inverse, for multiple values of <math>\alpha</math>.</i>	109
4.61	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 4DoF robot when varying the units while using the MX Inverse and <math>\alpha = 1</math>.</i>	110
4.62	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 4DoF robot when varying the units while using the MX Inverse, for multiple values of <math>\alpha</math>.</i>	111
4.63	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 6DoF robot when varying the units with the MP Inverse for <math>\alpha = 1</math>.</i>	116
4.64	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 6DoF robot when varying the units while using the MP Inverse and <math>\alpha = 1</math>.</i>	116
4.65	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 6DoF robot when varying the units while using the MP Inverse and <math>\alpha = 1</math>.</i>	117
4.66	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 6DoF robot when varying the units while using the MP Inverse, for multiple values of <math>\alpha</math>.</i>	118
4.67	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 6DoF robot when varying the units while using the MP Inverse, for multiple values of <math>\alpha</math>.</i>	118
4.68	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 6DoF robot when varying the units while using the MP Inverse, for multiple values of <math>\alpha</math>.</i>	119
4.69	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 6DoF robot when varying the units with the UC Inverse for <math>\alpha = 1</math>.</i>	123

4.70	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 6DoF robot when varying the units while using the UC Inverse and <math>\alpha = 1</math>.</i>	123
4.71	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 6DoF robot when varying the units while using the UC Inverse and <math>\alpha = 1</math>.</i>	124
4.72	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 6DoF robot when varying the units while using the UC Inverse, for multiple values of <math>\alpha</math>.</i>	125
4.73	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 6DoF robot when varying the units while using the UC Inverse, for multiple values of <math>\alpha</math>.</i>	125
4.74	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 6DoF robot when varying the units while using the UC Inverse, for multiple values of <math>\alpha</math>.</i>	126
4.75	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 6DoF robot when varying the units while using the MX Inverse and <math>\alpha = 1</math>.</i>	128
4.76	<i>Behavior of the trajectories of the end-effector for Motion 1 of the 6DoF robot when varying the units while using the MX Inverse, for multiple values of <math>\alpha</math>.</i>	128
4.77	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 6DoF robot when varying the units while using the MX Inverse and <math>\alpha = 1</math>.</i>	130
4.78	<i>Behavior of the trajectories of the end-effector for Motion 2 of the 6DoF robot when varying the units while using the MX Inverse, for multiple values of <math>\alpha</math>.</i>	130
4.79	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 6DoF robot when varying the units while using the MX Inverse and <math>\alpha = 1</math>.</i>	132
4.80	<i>Behavior of the trajectories of the end-effector for Motion 3 of the 6DoF robot when varying the units while using the MX Inverse, for multiple values of <math>\alpha</math>.</i>	132
A.1	<i>An example representation of the effects of common optimization techniques for back-propagation algorithm.</i>	144
A.2	<i>Overview of some of the learning schemes found in the literature for data-driven inverse kinematics solvers using all joints in the output layer.</i>	148

# List of Tables

4.1	<i>Overview of the twenty-four experiments conducted with MLP (All-Joints-ANN and Individual-Joints-ANN) and ANFIS (Individual-Joints-ANFIS).</i>	30
4.2	<i>D-H table with the parameters of the serial robots illustrated in Figure 4.1. The angles <math>\theta</math> and <math>\alpha</math> are expressed in degrees. The variables <math>d</math> and <math>a</math> are expressed in millimeters (mm).</i>	31
4.3	<i>Range of joint values used for generating the structured workspaces.</i>	32
4.4	<i>Range of joint values used for generating the random workspace.</i>	32
4.5	<i>An overview of the final number of hidden neurons retained for the structured and random datasets after network architecture search for all the manipulators.</i>	43
4.6	<i>An overview of the number of membership functions (MF) determined using Subtractive Clustering Algorithm [6] for structured and random ddtasets with the 4DoF manipulator.</i>	44
4.7	<i>An overview of the number of membership functions (MF) determined using Subtractive Clustering Algorithm [6] for structured and random ddtasets with the 5DoF manipulator.</i>	45
4.8	<i>An overview of the number of membership functions (MF) determined using Subtractive Clustering Algorithm [6] for structured and random ddtasets with the 6DoF manipulator.</i>	45
4.9	<i>An overview of the number of membership functions (MF) determined using Subtractive Clustering Algorithm [6] for structured and random ddtasets with the 7DoF manipulator.</i>	45
4.10	<i>Learning strategies for the experiment with the 4DoF robot reported in Table 4.11.</i>	46
4.11	<i>Results for 4 DoF robot using a structured dataset.</i>	47

4.12	<i>Learning strategies for the experiment with the 5DoF robot reported in Table 4.13.</i>	49
4.13	<i>Results for 5 DoF robot using the structured dataset.</i>	49
4.14	<i>Learning strategies for the experiment with the 6DoF robot reported in Table 4.15.</i>	50
4.15	<i>Results for 6 DoF robot using the structured dataset.</i>	50
4.16	<i>Learning strategies for the experiment with the 7DoF robot reported in Table 4.17.</i>	51
4.17	<i>Results for 7 DoF robot using the structured dataset.</i>	51
4.18	<i>Learning strategies for the experiment with the 4DoF robot reported in Table 4.19.</i>	57
4.19	<i>Results for 4 DoF robot using a random dataset.</i>	58
4.20	<i>Learning strategies for the experiment with the 5DoF robot reported in Table 4.21.</i>	60
4.21	<i>Results for 5 DoF robot using a random dataset.</i>	60
4.22	<i>Learning strategies for the experiment with the 6DoF robot reported in Table 4.23.</i>	61
4.23	<i>Results for 6 DoF robot using a random dataset.</i>	61
4.24	<i>Learning strategies for the experiment with the 7DoF robot reported in Table 4.17.</i>	62
4.25	<i>Results for 7 DoF robot using a random dataset.</i>	62
4.26	<i>Overview of the experiments conducted with the MP, UC and MX generalized inverses.</i>	70
4.27	<i>D-H Parameters of the serial robots illustrated in Figure 4.26 and used in the experiments with numerical methods. The angles <math>\theta</math> and <math>\alpha</math> are expressed in degrees. The variables <math>d</math> and <math>a</math> are shown in millimeters (mm) – but they were changed in the implementation to match the different choices of units.</i>	71
4.28	<i>Overview of the required input parameters and output results obtained from the implementations of the numerical method.</i>	72
4.29	<i>Experimental results obtained for Motion 1 of the 3DoF robot with <math>\alpha = 1</math> and MP Inverse.</i>	74

4.30	<i>Experimental results obtained for Motion 1 of the 3DoF robot with <math>\alpha = 1</math> and UC Inverse.</i>	77
4.31	<i>Experimental results obtained for Motion 1 of the 3DoF robot with <math>\alpha = 1</math> and MX Inverse.</i>	79
4.32	<i>Experimental results obtained for Motion 2 of the 3DoF robot with <math>\alpha = 1</math> and MP Inverse.</i>	81
4.33	<i>Experimental results obtained for Motion 3 of the 3DoF robot with <math>\alpha = 1</math> and MP Inverse.</i>	82
4.34	<i>Experimental results obtained for Motion 2 of the 3DoF robot with <math>\alpha = 1</math> and UC Inverse.</i>	84
4.35	<i>Experimental results obtained for Motion 3 of the 3DoF robot with <math>\alpha = 1</math> and UC Inverse.</i>	85
4.36	<i>Experimental results obtained for Motion 2 of the 3DoF robot with <math>\alpha = 1</math> and MX Inverse.</i>	87
4.37	<i>Experimental results obtained for Motion 3 of the 3DoF robot with <math>\alpha = 1</math> MX and Inverse.</i>	88
4.38	<i>Experimental results obtained for Motion 1 of the 4DoF robot with <math>\alpha = 1</math> and MP Inverse.</i>	92
4.39	<i>Experimental results obtained for Motion 1 of the 4DoF robot with <math>\alpha = 1</math> and UC Inverse.</i>	95
4.40	<i>Experimental results obtained for Motion 1 of the 4DoF robot with <math>\alpha = 1</math> and MX Inverse.</i>	98
4.41	<i>Experimental results obtained for Motion 2 of the 4DoF robot with <math>\alpha = 1</math> and MP Inverse.</i>	100
4.42	<i>Experimental results obtained for Motion 3 of the 4DoF robot with <math>\alpha = 1</math> and MP Inverse.</i>	102
4.43	<i>Experimental results obtained for Motion 2 of the 4DoF robot with <math>\alpha = 1</math> and UC Inverse.</i>	104
4.44	<i>Experimental results obtained for Motion 3 of the 4DoF robot with <math>\alpha = 1</math> and UC Inverse.</i>	106

4.45	<i>Experimental results obtained for Motion 2 of the 4DoF robot with <math>\alpha = 1</math> and MX Inverse.</i>	108
4.46	<i>Experimental results obtained with the 4DoF with <math>\alpha = 1</math> and MX Inverse.</i>	110
4.47	<i>Experimental results obtained for Motion 1 of the 6DoF robot with <math>\alpha = 1</math> and MP Inverse.</i>	113
4.48	<i>Experimental results obtained for Motion 2 of the 6DoF robot with <math>\alpha = 1</math> and MP Inverse.</i>	114
4.49	<i>Experimental results obtained for Motion 3 of the 6DoF robot with <math>\alpha = 1</math> and MP Inverse.</i>	115
4.50	<i>Experimental results obtained for Motion 1 of the 6DoF robot with <math>\alpha = 1</math> and UC Inverse.</i>	120
4.51	<i>Experimental results obtained for Motion 2 of the 6DoF robot with <math>\alpha = 1</math> and UC Inverse.</i>	121
4.52	<i>Experimental results obtained for Motion 3 of the 6DoF robot with <math>\alpha = 1</math> and UC Inverse.</i>	122
4.53	<i>Experimental results obtained for Motion 1 of the 6DoF robot with <math>\alpha = 1</math> and MX Inverse.</i>	127
4.54	<i>Experimental results obtained for Motion 2 of the 6DoF robot with <math>\alpha = 1</math> and MX Inverse.</i>	129
4.55	<i>Experimental results obtained for Motion 3 of the 6DoF robot with <math>\alpha = 1</math> and MX Inverse.</i>	131
4.56	<i>Qualitative comparison between data-driven and numerical approaches where (N/A) means not applicable.</i>	133

## ABSTRACT

In the context of articulated robotic manipulators, the Forward Kinematics (FK) is a highly non-linear function that maps joint configurations of the robot to poses of its end-effector. Furthermore, while in the most useful cases these functions are neither injective (one-to-one) nor surjective (onto), depending on the robot configuration – i.e. the sequence of prismatic versus revolute joints, and the number of Degrees of Freedom (DoF) – the associated Inverse Kinematics (IK) problem may be practically or even theoretically impossible to be solved analytically. Therefore, in the past decades, several approximate methods have been developed for many instances of IK problems. The approximate methods can be divided into two distinct categories: data-driven and numerical approaches. In the first case, data-driven approaches have been successfully used for small workspace domains (e.g., task-driven applications), but not fully explored for large ones, i.e. in task-independent applications where a more general IK is required. Similarly, and despite many successful implementations over the years, numerical solutions may fail if an improper matrix inverse is employed (e.g., Moore-Penrose generalized inverse).

In this research, we propose a systematic, robust and accurate numerical solution for the IK problem using the Unit-Consistent (UC) and the Mixed (MX) Inverse methods to invert the Jacobians derived from the Denavit-Hartenberg (D-H) representation of the FK for any robot. As we demonstrate, this approach is robust to whether the system is underdetermined (less than 6 DoF) or overdetermined (more than 6 DoF). We compare the proposed numerical solution to data driven solutions using different robots – with DoF varying from 3 to 7. We conclude that numerical solutions are easier to implement, faster, and more accurate than most data-driven approaches in the literature, specially for large workspaces as in task-independent applications. We particularly compared the proposed numerical approach against two data-driven approaches: Multi-Layer Perceptron (MLP)



and Adaptive Neuro-Fuzzy Inference System (ANFIS), while exploring various architectures of these Neural Networks (NN): i.e. number of inputs, number of outputs, depth, and number of nodes in the hidden layers.

# Chapter 1

## Introduction

For articulated robotic manipulators, the Inverse Kinematics (IK) is a difficult and highly non-linear function that creates a mapping between the end-effector poses (positions and orientations) and the joint configurations [1, 7–9]. Over the years, several strategies have been developed to solve the IK of robotic manipulators [1, 7–9]. These methods can be divided into two main broad groups: analytical and approximate methods. Analytical methods provide closed-form solutions, which are more reliable, accurate, and faster to obtain. However, depending on the robot joint configuration, i.e. the combinations of the robot joints, the number of Degrees of Freedom (DoF), and the redundancy nature of the manipulator, it can be difficult and sometimes even impossible to find closed-form solutions for the IK problem [8]. Thus, various approximate methods have been sought over the years, and they can be further sub-divided into three approaches: data-driven, numerical, and hybrid.

Most data-driven approaches use Artificial Neural Networks (ANN), which are trained on datasets generated beforehand using the Forward Kinematics (FK) [1,8] and the Denavit-Hartenberg (D-H) representation [10] of the target robot. Those datasets can represent two categories of workspace: task-driven and task-independent. For IK of task-driven workspaces, ANN methods have been successful in learning and predicting data points lying on specific trajectories (planar, circular, spring shapes, etc.) [11–15]. However, not much effort have been dedicated to investigating solutions for IK of task-independent workspaces. In this case, an ANN must learn a much more generic IK mapping and provide predictions over the entire workspace [16]. Since practical robot workspaces

have a very large number of poses on which the end-effector can be placed, the task of learning more generic IK mappings becomes challenging in terms of: generating good datasets [16] for training and testing; designing appropriate network architectures that can capture such vast workspace configuration [17]; and predicting reliable solutions that fall into the acceptable ranges for many robotic applications [18, 19].

At the same time, it is well accepted the fact that numerical approaches can accurately approximate the IK for task-independent workspaces, usually through an iterative process [7, 9]. However, despite all the successful implementations reported in the literature, numerical methods can still fail if an improper matrix inverse is employed. More specifically, numerical methods often resort to inverting Jacobian matrices, and whenever this Jacobian becomes singular – either underdetermined (less than 6 DoF) or overdetermined (more than 6 DoF) – its inverse is not uniquely defined. So, typically a *left* or *right* pseudo inverse, or more generically, a Moore-Penrose (MP) generalized inverse can be employed [20]. Unfortunately, MP inverses have been shown to fail in the case of incommensurate manipulators [21, 22]. As we know, robotic systems with incommensurate units are very common, as it refer to systems having elements expressed in different types of units [23, 24]: i.e. position vectors with units of distance such as meters, centimeters, etc., and orientations with units for angles such as radian or degrees, at the same time as joint vectors also combine angles (for revolute joints) and distances (for prismatic joints). Yet, for IK solutions in such cases to be considered successful, an arbitrary change in units should not affect the behavior of the system with respect to other units [23–26]. But again, MP inverses have been shown to not always provide reliable and stable control in the case of IK involving incommensurate units [21, 22] – it is hence said that MP does not satisfy unit consistency. Therefore, another generalized inverse, namely the Unit-Consistent inverse (UC), can be used instead [22] to address those cases. But once again, while the UC Inverse is consistent with respect to arbitrary change of units, it may also fail to provide consistent inverses in the presence of rotation of the reference frame or any frame prior in the robots D-H representation to the frame where the unit change occurred. When that is the case, instead of using the UC, one can use the Mixed Inverse ([22]) to selectively provide consistency with respect to arbitrary changes of both units and rotations of the coordinate system. The keyword here is “selectively”: a task that so far has been done by

manual, human inspection.

The main contributions of this thesis are twofold. First, it studies and compares two main paradigms under the category of approximate solutions for the IK problem: i.e for data-driven and numerical approximations. Second, it proposes a robust, systematic and reliable numerical solution for the IK using the Unit-Consistency (UC) and the Mixed (MX) Inverse methods for inverse Jacobians — whether underdetermined (less than 6 DoF) or overdetermined (more than 6 DoF) thru the inspection of the D-H representation of the robots. More specifically, two data-driven methods – MLP and ANFIS – are evaluated for task-independent workspaces. Then, a numerical solution is evaluated based on MP, UC, and MX Inverses using the D-H representation to select which inverse is correctly suited for the task. Finally, we compare the proposed numerical solution using five different robots with the number of DoF varying from 3 to 7.

In summary, this thesis focus on: 1) investigating the application of data-driven solutions for IK problem in the context of task-independent workspaces, while searching for appropriate network architectures for various generated datasets; and 2) providing a systematic and autonomous method to apply MX Inverses to numerical methods for the IK problem, while investigating the effects of varying the attenuation parameters  $\alpha$  on the end-effector behavior.

The rest of this thesis is organized as follows: Chapter 2 provides the terminology, background, and a quick survey on the related work. Chapter 3 presents the proposed data-driven and numerical approaches. Later, in Chapter 4, the experimental results of all the conducted experiments are presented. Also in this same chapter, we provide a comparative study between data-driven and numerical methods based on the conducted experiments on task-independent workspaces. Finally, in Chapter 5, we present the conclusion and future work.

# Chapter 2

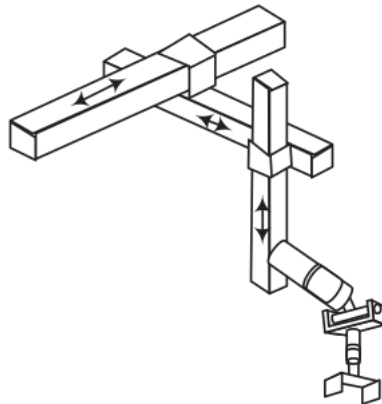
## Terminology, Background and Related Work

This chapter introduces the main concepts and techniques used in the work presented in this thesis. The different notations and terminologies used throughout the thesis are also established for a better understanding of the same thesis. A quick survey of the related IK methods is also provided.

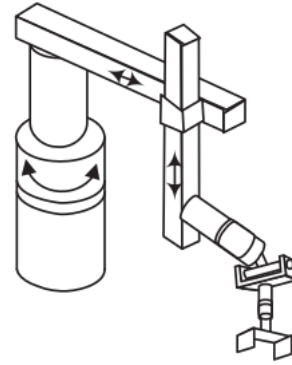
### 2.1 Serial Robots

Serial robotic manipulators are programmable and articulated mechanical devices consisting of a chain of links and joints, where the joints are either revolute (angular) or prismatic (linear). Each joint is carefully guided by an actuator (e.g., an electric or hydraulic motor equipped with positional encoders). The manipulators are designed and available in different configurations based on the combination of the joint types needed to reach the desired number of DoF and the workspace to be covered. For example, the popular 4 DoF SCARA manipulator (Selective Compliance Articulated Robot Arm) is made of three revolute (R) joints and one prismatic (P) joint – usually referred to as a RRPR (or 2RPR) joint configuration. In Figure 2.1, an overview of the different configurations are presented for several types of serial robots.

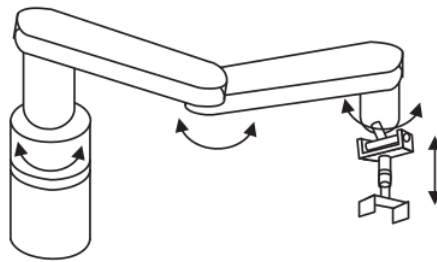
Generally, given the design configuration, the kinematics of a manipulator can be obtained by its link/joint chain structure, where each link is regarded fixed with respect to the



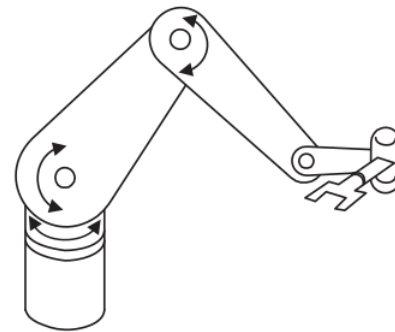
(a) Cartesian (PPP or 3P) Robotic Manipulator



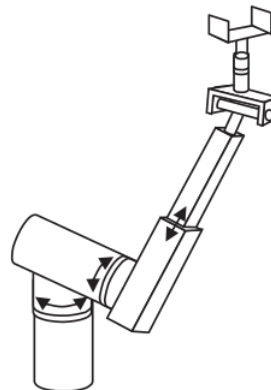
(b) Cylindrical (RPP or R2P) Robotic Manipulator - considering only first 3 DoF to account for its designation



(c) SCARA (RRPR or 2RPR) Robotic Manipulator - considering only first 4 DoF to account for its designation



(d) Articulated (RRRRRR or 6R) Robotic Manipulator - considering all 6 DoF to account for its designation



(e) Spherical (RRP or 2RP) Robotic Manipulator - considering only first 3 DoF to account for its designation

Figure 2.1: Examples of different configurations of serial robotic manipulators [1].

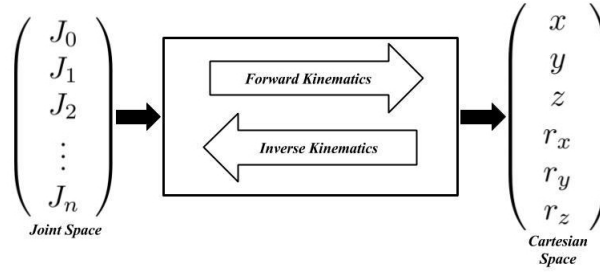


Figure 2.2: Mapping provided by the forward ( $f_{FK}$ ) and inverse kinematics ( $f_{IK}$ ) between Joint and Cartesian spaces.

next moving link in the chain. Every link in the chain moves according to the specific DoF of the associated joint. The assignment of coordinate frames, the order of the links/joints, and the variables representing the positions of the various coordinate frames are defined by standards, such as the Denavit-Hartenberg (D-H) representation [1, 10]. Two important functions arise from this setup and are required for programming robotic arms: the forward ( $f_{FK}$ ) and inverse ( $f_{IK}$ ) kinematics, where  $f_{FK}$  and  $f_{IK}$  are typically multivariate, non-linear and not necessarily either injective (one-to-one) or surjective (onto) functions.

More specifically,  $f_{FK}$  is the function that maps the pose (i.e. position and orientation in space) of the end-effector given the joint values, while  $f_{IK}$  is the function that returns the corresponding joint values needed to achieve a desired end-effector pose. For an arbitrary serial robot, we consider two vectors:  $\vec{D} = [x, y, z, r_x, r_y, r_z]^T$  and  $\vec{Q} = [J_0, J_1, J_2, \dots, J_N]^T$ , respectively describing the pose and joint vectors, such that:  $\vec{D} = f_{FK}(\vec{Q})$  and  $\vec{Q} = f_{IK}(\vec{D})$  and ideally, but quite rarely:

$$f_{IK}(\cdot) = f_{FK}(\cdot)^{-1} \quad (2.1)$$

Figure 2.2 depicts the mapping that the forward ( $f_{FK}$ ) and inverse kinematics ( $f_{IK}$ ) establish between Joint and Cartesian spaces. The solution for the forward kinematics ( $f_{FK}$ ) is relatively simple to compute using the chain structure and the Denavit-Hartenberg (D-H) representation. In fact, the D-H representation allows us to express the robot end-effector pose with respect to the robot base as a succession of translational and rotational transformations incurred by the frames attached to two consecutive links after activating the joint connecting those links. Figure 2.3 shows an illustration of the D-H representation of a robotic manipulator. On the other hand, a solution for the inverse kinematics ( $f_{IK}$ ) is not as easily provided by the D-H, and it can be difficult or even impossible to be derived

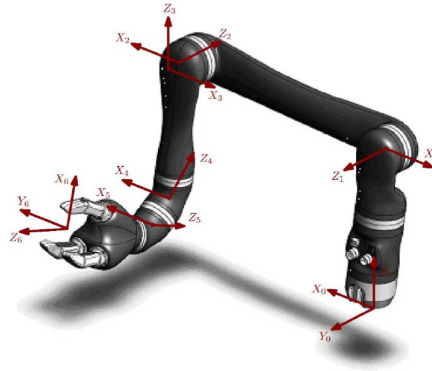


Figure 2.3: An illustration of the D-H representation with the frames attached to each joint of a 6DoF Kinova robotic arm.

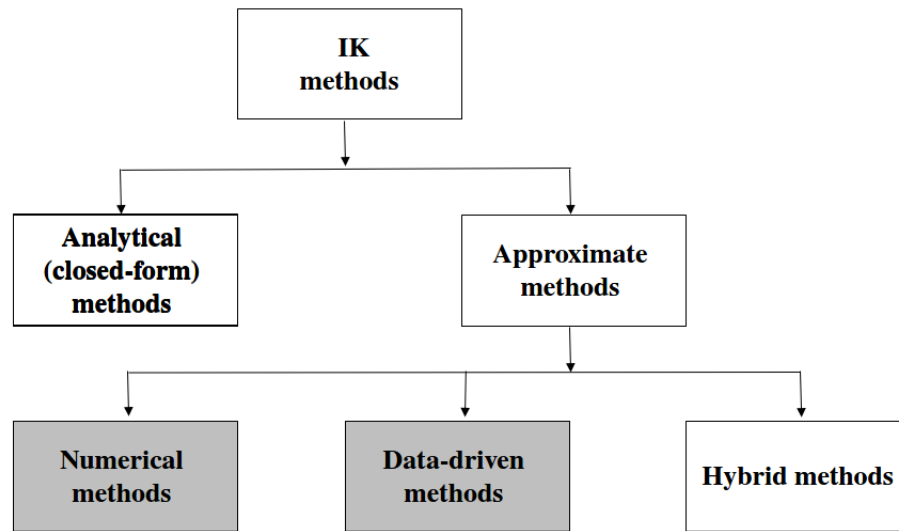


Figure 2.4: Taxonomy of the inverse kinematics methods.

by hand for some particular robots.

Several IK methods have been developed over the years and they can be divided into two main broad groups: analytical and approximate methods [1,7–9]. Approximate methods can be further sub-divided into data-driven, numerical, and hybrid approaches as depicted in Figure 2.4. This research focuses on the application of data-driven and numerical methods to various robots and their workspaces. In the next sections, an overview of analytical versus approximate methods is provided.



## 2.2 Analytical Methods

Analytical methods, also known as closed-form solutions, can provide more reliable, accurate, and faster to obtain solutions, while they also require a low computational cost [9]. They are basically implementations of an equation derived manually, from analytical inspection of the FK and its D-H parameters, and all joint variables associated with a given pose can be determined from a set of non-linear equations. Most systems with revolute and prismatic joints having a total of six DoF in a single chain can be solvable [8]. However, specially as the number of DoF increases, the design of the joint configuration needs to meet certain conditions for the IK to be solvable. For example, a six DoF robot, with a set of D-H parameters  $(\theta_i, d_i, a_i, \alpha_i)$ , needs to have some of its  $Z_i$  axis corresponding to the  $\theta_i$  angles intersecting each other; or have multiple rotation angles  $\alpha_i$  corresponding to the  $X_i$  axis equal to 0 or  $\pm 90$  degrees. In summary, depending on the robot workspace, the combinations of the robot joints, the designed number of DoF, and the redundancy nature of the manipulator, it can be difficult and sometimes even impossible to find closed-form solutions to the IK problem [8], and approximate solutions are hence required.

## 2.3 Approximate Methods

Approximate methods encompass all other non-analytical methods. They can be subdivided into data-driven, numerical, and hybrid, which will be explained next. However, since the mapping between Joint and Cartesian spaces can be very large, many systems approached the problem of approximating the IK by constraining the robot workspace. In these so-called *task-driven* workspaces, only a subset of the robot workspace is mapped, usually by a limited sequence of motions needed to complete a specific task. On the other hand, in *task-independent*, it is expected that the robot can occupy its entire workspace, as it can assume any of a very large number of joint configurations.

### 2.3.1 Data-driven Methods

Data-driven approaches use learning paradigms, such as Genetic Algorithms (GA), Artificial Neural Networks (ANN), etc. trained on particular datasets. These datasets consist

of various samples of end-effector poses and their required joint configurations. They describe the robot workspace over which the paradigm is required to learn – as mentioned earlier, under the assumption of a task-driven or task-independent workspace. So, data-driven approaches rely on the creation of such datasets using the Forward Kinematics (FK) and the Denavit-Hartenberg (DH) representation [16].

Perhaps, the most popular paradigm used in data-driven methods is in the form of ANN, but they are certainly not the only one. In the next section, we conduct a brief survey on data-driven IK methods mostly using ANN, GA, Support Vector Machines (SVM), etc.

### 2.3.1.1 Survey on Data-driven Methods

Many data-driven IK methods can be found in the literature and were developed over the years. As mentioned earlier, most of these methods adopted a learning-based strategy that relied on some type of ANN to estimate the joint values based on the poses of the robots. Among those ANN techniques, Multi-Layer Perceptron (MLP) has been extensively used for task-driven IK learning problems [11–15]. Besides limiting the problem to task-driven workspaces, early systems also considered only a small number of DoF. For example, in a study by NASA researchers [11], several single-hidden-layer networks ( $2/n_H/2$ ), with varying number of nodes in the hidden layer ( $n_H = 3, 6, 12, 50, 100$ ), were investigated to learn the inverse kinematics of a 2DoF robot. In this same study, Choi and Lawrence [11] also investigated a multi hidden-layer network ( $3/10/10/3$ ) to find the IK of a 3 DoF robot. In [27], the authors employed a ( $2/10/10/2$ ) MLP model to find the solution for the IK problem of a 2DoF robot. Even in more recent studies, such as by Srisuk et al. [14], a three-layer, single-output network ( $2/2/1$ ) was used to approximate the IK of a 3DoF robot. In this case: the network was not fully connected; both the *sine* and *cosine* functions were used as activation functions; and some of the initial values of the weights corresponded to the links (distance between two joints). But in other cases, such as by Jha and Biswal [12], a MLP was used to approximate the IK solution of a 4DoF SCARA robot based only on its position – no orientation. That is, the structure of their MLP only required the position of the end-effector as input to the network. The authors observed that the MLP employed provided minimum mean square error for a dataset of

1000 data points. In a more recent system, Csiszar et al. [13] used two MLP networks, a (3/50/50/4) and a (4/50/50/6), to learn the inverse kinematics of 3DoF and 4DoF serial robots respectively in a task-independent workspace. They generated a much larger dataset with 25000 data points for the 3DoF and 75000 data points for the 4DoF, but they constrained the workspaces by limiting the ranges of the joints of the robots. They also investigated the effects of joint misalignments and concluded that the misalignments investigated did not make any difference in the network predictions.

With the integration of fuzzy logic into the neural network paradigm, Adaptive-Neuro Fuzzy Inference System (ANFIS) [3] also became a sought out paradigm for the IK-learning problem [28, 29]. In the first case, Alavandar and Nigam [28] employed an ANFIS to learn the IK of two robots (2DoF and 3DoF). However, while also constraining the joint angles, the neuro-fuzzy network had to learn only the position in a  $X - Y$  planar workspace. In the second case, Narayan and Singla [29] addressed the IK of a more complex 4DoF SCARA robot, but the motion was restricted to a curved line in the robot workspace and the ANFIS was also trained based only on the robot position as input to predict its joint configuration.

It is important to emphasize that in all these works, either task-driven trajectories (circular, planar, etc) or limits on the number of DoF, in the range of the joints, or the position (i.e. no orientation) of the end-effector were imposed. Also, besides the lack of details regarding the generation of the datasets, these systems still required several training iterations, which resulted in long computations despite all the attempts in reducing the search spaces that the ANNs had to learn.

Two important factors motivated the investigation of better prediction models as the dimensionality of the ANN search spaces was set higher: i.e. the inclusion of the orientation of the robot in the input vector and use of robots with higher DoF [19, 30–33]. In a recent study, El-Sherbiny et al. [19] applied three soft-computing methods – MLP, ANFIS, and GA – to solve the inverse kinematics of a 5 DoF robotic arm. The input vector was represented by the end-effector pose (position and orientation). However, the study was limited to a specific spring-shape trajectory, hence not providing deeper insights on the performance of these methods in more generic robot workspace. Also, the author applied an error minimization algorithm to improve the preliminary joint outputs of the

MLP and ANFIS. In [30], while including the feedback of the current pose in the input pattern of a MLP, Almusawi et al. worked on the control of a 7 DoF Denso robotic arm, and they were able to adjust the network weights to achieve very low error. But as other works, the robot workspace was limited to a spring shape trajectory. On a different front, Chen and Lau [31] investigated the use of Support Vector Machines (SVM) to solve the inverse kinematics of a 7 DoF Mitsubishi PA-10 manipulator. But as in the other studies, the results were restricted to a particular robot in a limited workspace. Raptis and Tzafestas [32] exploited the generation and preprocessing of training data, data scaling, treatment of multiple solutions, and the reduction of the approximation error in order to speed up the training. In order to learn the pose vector, they employed multiple Radial Basis Function Networks (RBFN) and discussed two fuzzy logic solutions to improve the predicted joints of a 3 DoF. Bingul et al. [33] compared three different input representations for the orientation of the end-effector: homogeneous transformation, Euler angles and equivalent angle axis; to predict the target joint outputs based on the pose inputs with three different MLP (12/12/24/6, 6/12/24/6 and 7/12/24/6). For all the network architectures, the authors used the same dataset made of 8000 data points. They found that the prediction errors were smaller for the 12/12/24/6 configuration.

As alluded earlier, many system use ANN, but these are not the only tools available. Zhou et al. [18] developed an intelligent algorithm based on Extreme Learning Machine (ELM) and Sequential Mutation Genetic Algorithm (SMGA) to predict the IK of a 6 DoF. In their work, the preliminary inverse kinematics solution, computed by the ELM network was optimized using the SGMA; hence reaching a reduction in training and computational time with a better accuracy compared to traditional MLP. In another attempt to speed up the network convergence, Zhang et al. [34] employed six Radial Basis Function Networks (RBFN), each with a single output that predicted one of the 6 DoF of the manipulator. The authors concluded that RBFN performed better than MLP with similar architecture and used back-propagation as the training algorithm for the MLP. They used a dataset of 4096 data points and utilized Levenberg-Marquardt as training algorithm for the RBFN. Since some of the results achieved by these systems still need to be improved for sensitive robotics tasks, Köker et al. [35] designed a Neural Network Committee Machine (NNCM) and applied it to the IK of a 6 DoF redundant robotic manipulator. The authors varied the

number of MLP in the NNCM and an implementation made of ten MLP, each one trained based on separate datasets (each having 6000 data points), was reported to provide a better learning performance with smaller Mean Squared Error (MSE) values.

Finally, unsupervised learning has also been exploited, as a paper by Hoffman and Möller [36] investigated the use of Recurrent Neural Networks (RNN) and unsupervised learning to derive the kinematics (forward and inverse) of a 6DoF robotic arm. The RNN was built from a combination of Neural Gas and local Principal Component Analysis (PCA), and it was reported to perform better than a single-hidden-layer MLP having 200 neurons on the hidden layer. Barreto and Araújo [37] reviewed a series of studies that involved the use of Self-Organizing Maps (SOM) for kinematics-based robot control. After comparing the performance of SOM and supervised IK solvers like MLP and RBFN on several robotic tasks, they concluded that SOM were simpler to implement and faster in learning. Cavalcanti and Santana [38] also used a Self-Organizing Map (SOM) with different sizes (18x18, 36x36 and 72x72) to learn the inverse kinematics of an Aura robotic arm with only 2 DoF and confined to two well-defined trajectories: linear and planar. The unsupervised algorithm involved visual cues to learn the mapping between sensor information and control commands.

### 2.3.2 Numerical Methods

As the name suggests, numerical methods employ numerical approximations for the FK function  $\vec{D} = f_{FK}(\vec{Q})$ , or more directly, approximations for the IK function  $\vec{Q} = f_{IK}(\vec{D})$ . Usually, these methods are iterative and resort to the approximation of those function using Taylor series, i.e. Jacobian and Hessian matrices. The general idea, depicted in Figure 2.5, is that a succession of small perturbations are applied to each joint until the end-effector is brought from its initial pose to the desired, final pose. Within the family of IK numerical methods, we focus on methods reliant on the inverse Jacobian – i.e. methods that require only the computation of the Jacobian and/or its inverse to create a linear approximation for the robot path at each iteration of the process [9] (see Figure 2.5). This class of IK numerical solutions become especially interesting when the robotic system is incommensurate, and the Jacobian matrix is by definition or by its current configuration singular. Next, we revisit important concepts related to the Jacobian matrix, matrix in-

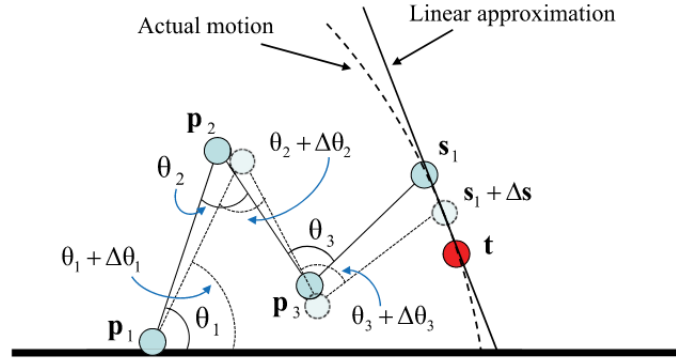


Figure 2.5: An overview of how Jacobian based numerical methods rely on small perturbations applied to individual joints.

verses and generalized inverses. We also conduct a brief survey of numerical methods and visit the notion of incommensurate robotic manipulators.

### 2.3.2.1 Jacobian

In multivariate calculus, the Jacobian  $J$  is defined as the matrix with the first-order partial derivatives of an arbitrary multivariate function  $f$ . Let us consider  $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ , with  $(x_1, \dots, x_n) \mapsto (f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n))$ , as the multivariate function of  $n$  inputs and  $m$  outputs. The Jacobian matrix  $J$  of  $f$  is then defined as:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \quad (2.2)$$

In the robotics context, this Jacobian  $J$  is often computed to approximate the forward kinematics function  $\vec{D} = f_{FK}(\vec{Q})$  using a simple first-order Taylor series, and the process of finding the inverse kinematics function  $\vec{Q} = f_{IK}(\vec{D})$  can be mathematically expressed by equation (2.1), which requires the computation of the inverse Jacobian matrix. That is:

$$\frac{\partial \vec{D}}{\partial t} = J \cdot \frac{\partial \vec{Q}}{\partial t} \implies \frac{\partial \vec{Q}}{\partial t} = J^{-1} \cdot \frac{\partial \vec{D}}{\partial t} \quad (2.3)$$

As it is the case for any matrix inverse, the Jacobian matrix needs to satisfy some properties for an inverse to exist and to be unique – e.g. non-singularity. However, many times

Jacobian matrices may be singular by definition or by circumstances related to the pose of the robot. Also, they may contain variables expressed in different units – i.e. combination of linear (prismatic) and angular (revolute) joints, which leads to incommensurate robotic systems. In the next section, the general notions of matrix inverses and generalized inverses are presented.

### 2.3.2.2 Matrix Inverse and Generalized Inverse

Generally, the inverse of a matrix can be uniquely defined for all non-singular and square matrices. Let us consider an arbitrary matrix  $A \in \mathbb{R}^{r \times c}$  having  $r$  rows and  $c$  columns. On one hand,  $A$  is said to be non-singular if it has a non-zero determinant ( $\det(A) \neq 0$ ) and it is square when  $r = c$ . In those cases, the inverse  $A^{-1}$  of  $A$  is uniquely defined and it satisfies all conditions in eq: (2.4). On the other hand,  $A$  is said to be singular if ( $\det(A) = 0$ ) and rectangular when either  $r < c$  or  $r > c$ . In addition, a rectangular matrix is always singular. For a singular matrix  $A$ , its inverse can be defined as a non-square  $r \times c$  generalized inverse  $A^{\sim -1}$ , which may satisfy some, but not necessarily all the conditions in eq: (2.4). Also, if  $A$  is non-singular and square, any generalized inverse should be  $A^{\sim -1} = A^{-1}$ .

The matrix inverses typically used in robotics and later to be used in this thesis are:

- **the 2-sided inverse** is a unique matrix  $A^{-1}$  computed from a square, non-singular matrix  $A$  that satisfies the following conditions (C#) [39].

$$\begin{cases} A^{-1}.A = I & (C1) \\ A.A^{-1} = I & (C2) \\ A.A^{-1} = A^{-1}.A & (C3) \end{cases} \quad (2.4)$$

- **the left pseudo-inverse**  $A^{-L}$  is defined for a full column-rank matrix (i.e. columns are independent of each other). Usually,  $A^{-L}$  is derived for an over-determined system of equations, i.e. where  $r > c$ . In that case:

$$\begin{aligned} A^{-L} &= (A^T.A)^{-1}.A^T \\ (A^T.A)^{-1}.A^T.A &= A^{-L}.A = I \end{aligned} \quad (2.5)$$

Clearly,  $A^{-L}$  satisfies only condition C1 above.

- **the right pseudo-inverse**  $A^{-R}$  is defined for a full row rank matrix (rows are independent to each other). Usually,  $A^{-R}$  is derived for an under-determined system of equation where  $r < c$ .

$$\begin{aligned} A^{-R} &= A^T.(A.A^T)^{-1} \\ A.A^T.(A.A^T)^{-1} &= A.A^{-R} = I \end{aligned} \quad (2.6)$$

This time,  $A^{-R}$  satisfies only condition C2 above.

- **the Moore-Penrose generalized inverse**  $A^{-P}$  can be defined based on the Singular Value Decomposition (SVD) [40–42] of  $A$  known as  $A = U.S.V^*$  by:

$$A^{-P} = (U.S.V^*)^{-P} = V.S^{-P}.U^* \quad (2.7)$$

where  $A^{-P}$  is the Moore-Penrose generalized inverse of  $A$ ;  $U^*$  is the conjugate transpose of  $U$ , the matrix with the eigenvectors of  $A$ ;  $V^*$  is the conjugate transpose of  $V$ , the matrix with the singular values (eigenvalues) of  $A$ ;  $I$  is the identity matrix;  $r$  is the number of rows and  $c$  is the number of columns of  $A$ . In the case of the MP Inverse, it will satisfy C1 only if  $A$  is over-determined and C2 only if  $A$  is under-determined.

- **the Unit-Consistent inverse**  $A^{-U}$  is an inverse defined with the property of invariance with respect to the choice of units used on different state variables, e.g, miles, kilometers, meters, centimeters, etc... The definition of the UC Inverse derives from the Unit-Invariant Singular Value Decomposition (UI-SVD) [22]. Hence, taking the inverse of UI-SVD gives the formula of the UC Inverse:

$$\begin{aligned} A &= D.(U.S.V^*).E \\ (A)^{-U} &= (D.(U.S.V^*).E)^{-U} \\ (A)^{-U} &= E^{-U}.V.S^{-U}.U^*.D^{-U} \\ A^{-U} &= E^{-1}.V.S^{-U}.U^*.D^{-1} \end{aligned} \quad (2.8)$$

where  $A^{-U}$  is the UC inverse of  $A$ ;  $U^*$  is the same conjugate transpose of  $U$ , the matrix of eigenvectors of  $A$ ;  $V^*$  is also the conjugate transpose of  $V$ , the matrix of eigenvalues of  $A$ ; while  $E$  and  $D$  are diagonal matrices that take into account the scale due to the change of units.

- **the Mixed inverse**  $A^{-M}$  is an inverse that selectively provides invariance with respect to arbitrary changes of units as well as with respect to rotations [22]. The



Mixed inverse is derived using the concept of block matrix inverse, where variables requiring unit consistency are block-partitioned in the top left of the  $A$ , and the variables requiring rotation consistency are block-partitioned in the bottom right of  $A$ . Next, block matrix inverse is applied to  $A$  to produce  $A^{-M}$  such that:

$$A = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix} \quad (2.9)$$

$$A^{-M} = \begin{bmatrix} (W - XZ^{-P}Y)^{-U} & -W^{-U}X(Z - YW^{-U}X)^{-P} \\ -Z^{-P}Y(W - XZ^{-P}Y)^{-U} & (Z - YW^{-U}X)^{-P} \end{bmatrix}$$

Robotics applications that require matrix inversion of the Jacobian will employ the two-sided inverse whenever the same Jacobian matrix is square and non-singular. However, when that is not the case, left/right pseudo-inverses and the MP inverse are typically a substitute for the two-sided inverse. This can represent a problem for systems requiring properties such as unit and/or rotation invariance [22, 24]. In this thesis, the above matrix inverses are exploited to create a systematic solution for the IK computation of any arbitrary serial robot using the numerical method developed in [4].

### 2.3.2.3 Survey on Numerical Methods and Incommensurate Systems

Robotic systems with incommensurate units refer to systems having different types of units [23, 24]. In such a system, a pose vector may combine positions with units of distance, and orientations with units of angle, while a joint vector may also combine revolute joints with units of angle and prismatic joints with units of distance. For such systems to be considered stable, an arbitrary change in a specific unit should not affect the behavior of the whole system [23–26]. Numerical approaches for IK have frequently defined the MP Inverse as the inverse for singular Jacobian matrices. However, the MP Inverse does not always guarantee the stability of the robotic systems of interest as it may produce inconsistent or erroneous results and lead researchers astray in the presence of incommensurate systems [21, 22, 43–45]. In fact, Schwartz et al. [23, 24, 44, 46] conducted several investigations on the effects of the units involved in the control of incommensurate robotic systems for which control algorithms use eigenvectors, eigenvalues or singular values of the Jacobian matrix. Those investigations concluded that the use of Singular Value De-

composition (SVD) in Jacobian based kinematics methods yields erroneous or arbitrary solutions. One of the possibilities found in the literature, to circumvent the inconsistencies of incommensurate systems is the use of user-defined units-adjusting weights [47,48]. However, the choice of units-adjusting weights just adds another level of arbitrariness between systems and applications, as mentioned in [23]. In the same sense, Uhlmann [22] developed two generalized inverses UC and Mixed Inverses applicable to incommensurate robotic systems, while Zhang and Uhlmann [21] used an incommensurate 3DoF manipulator to show that the MP inverse was affecting the end-effector trajectories and making the system unstable when the units were being varied. They used the UC inverse and succeeded in achieving a more reliable control of the manipulator end-effector.

# Chapter 3

## Proposed Approaches

This thesis focuses on data-driven and numerical methods. This section presents the methodology of these approaches. For IK data-driven methods, MLP and ANFIS are described and used in the IK approximation of robotic arms in unconstrained workspaces based on generated data-points. For numerical approaches, a robust, fast, and accurate numerical approach is presented.

### 3.1 Data-driven methods

#### 3.1.1 Artificial and Fuzzy Neural Networks

In this section, the concepts behind two of the data-driven techniques, used to approximate the inverse kinematics of serial robots in this study, are presented. Later, the results that they provided will be analyzed and discussed in chapter 4.

Inspired from the human brain, Artificial Neural Networks (ANN) are massively parallel distributed processors made up of simple processing units (neurons) that store knowledge and make it available for use [49]. The knowledge is acquired through the learning process and stored in the synaptic weights ( $w_{ji}$ ) and biases ( $b_i$ ). The learning process is a procedure during which, based on several network parameters (learning rate  $\alpha$ , momentum  $\beta$ , etc.), a learning algorithm modifies the weights and biases until the network achieves a good generalization. In that sense, the generalization is achieved when the network is able to produce reasonable outputs for inputs not encountered during the learning.

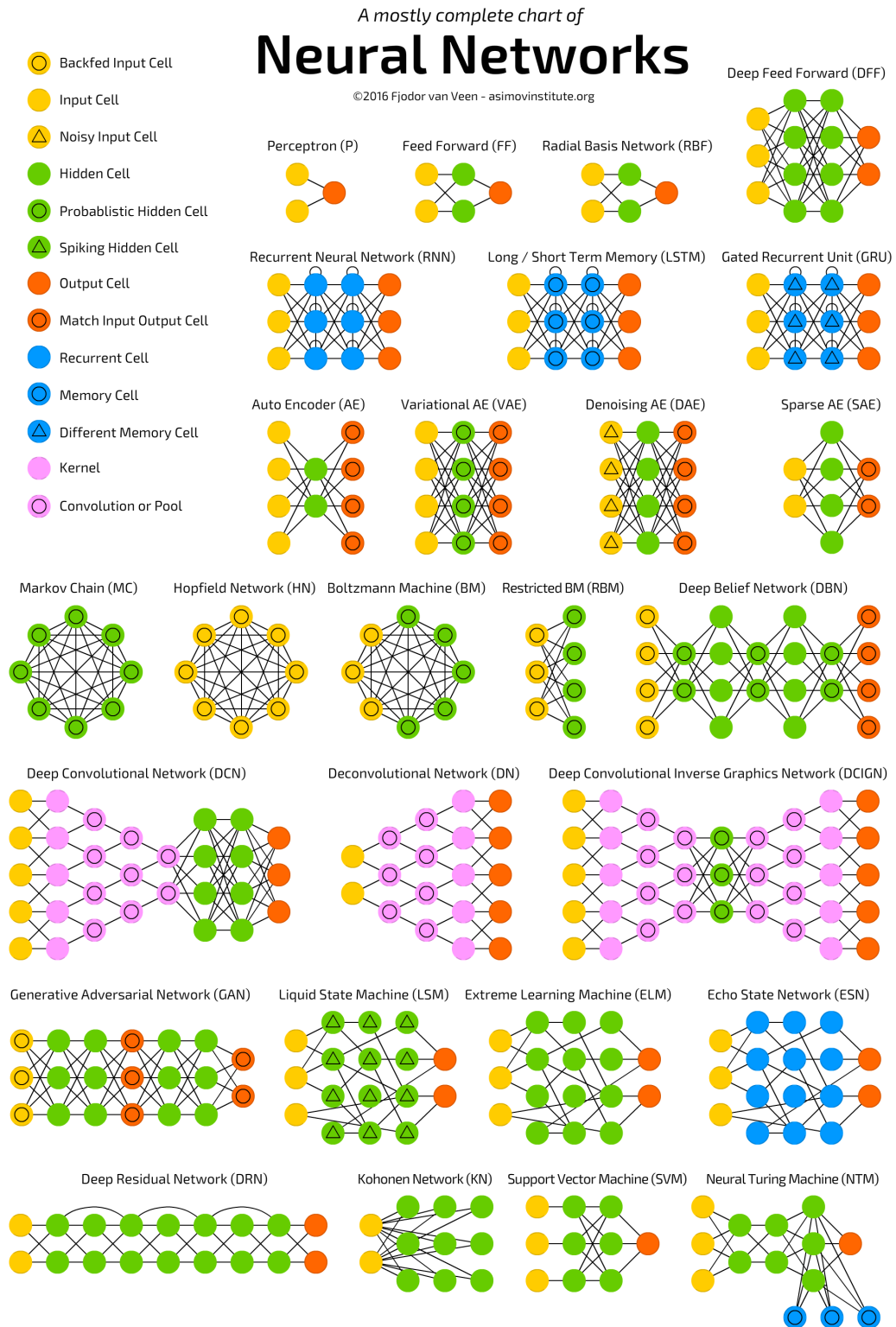
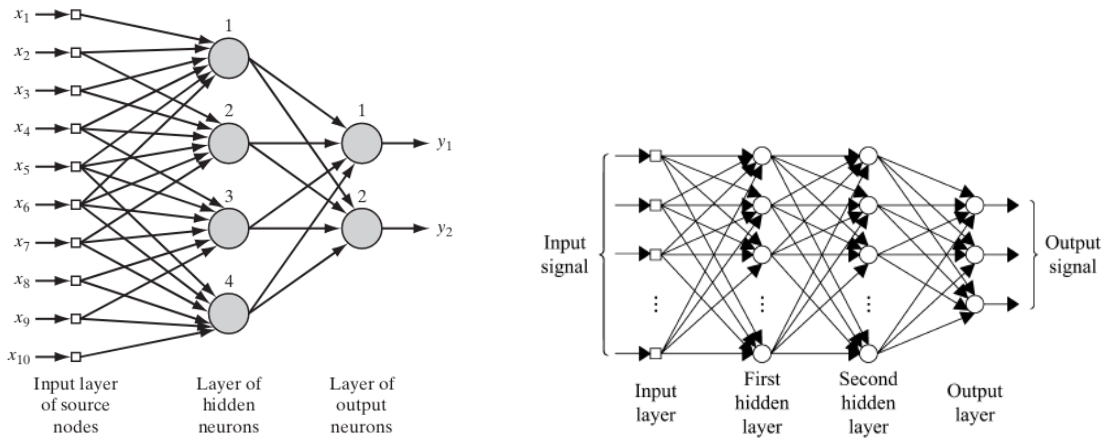


Figure 3.1: *The Neural Networks Zoo compiled in [2].*



(a) A MLP with one input layer (ten neurons), one hidden layer (four neurons) and one output layer (two neurons) presented in [49]. (b) A MLP with one input layer, two hidden layers and one output layer (three neurons) presented in [50].

Figure 3.2: Network architectures of a Multilayer-Perceptron (MLP).

Over the years, several ANN architectures have been developed and used either in a supervised or unsupervised manner by the machine learning community. In Figure 3.1, some of those architectures have been presented. In general, network structures are often expressed in the form of  $n_I/n_H/n_O$  where  $n_I$  denotes the number of input neurons,  $n_H$  the number of hidden neurons and  $n_O$  the number of output neurons [51]. For example: a (2/3/7/4) network would denote a network having 2 input neurons at the input layer, 3 hidden neurons at the first hidden layer, 7 hidden neurons at the second hidden layer and 4 output neurons at the output layer. We will adopt the same expression to denote network structure throughout this thesis.

### 3.1.1.1 Multilayer-Perceptron

In this study, a Multilayer-Perceptron (MLP) or fully connected Feed-Forward Neural Network (FF) has been investigated for the IK approximation. Figure 3.2 shows two MLP architectures. This type of network, used in a supervised manner, performs input-output mappings to capture a representation of a dataset knowledge through the modification of the weights and biases during the learning. The learning also known as the training phase is divided into two passes: the forward-pass and backward-pass.

For an arbitrary ANN neuron  $j$ , the forward-pass requires finding the induced local field  $v_j$  and the local output  $y_j$  at that neuron:

$$v_j^{(h)}(k) = \sum_{i=0}^{n_{h-1}} w_{ji}(k) y_i^{(h-1)}(k) \quad (3.1)$$

$$y_i^{(h)}(k) = \varphi_j(v_j^{(h)}(k)) \quad (3.2)$$

where:  $h$  is the current hidden layer,  $h - 1$  is the previous hidden layer,  $n_{h-1}$  is the number of hidden neurons,  $k$  is the current training sample,  $w_{ji}$  is the weight connecting the previous neuron  $i$  to the following neuron  $j$  and  $\varphi_j$  is the activation function.

The core of the learning happens at the output neurons where the instantaneous error  $e_j$  is continually found between desired and predicted outputs. Then, the instantaneous error  $E$  of the network is found and used for the backward pass.

$$e_j(k) = \hat{y}_i(k) - y_i(k) \quad (3.3)$$

$$E(k) = \frac{1}{2} \sum_j^{n_o} e_j^2(k) \quad (3.4)$$

where:  $\hat{y}_i$  is the desired output,  $y_i$  is the predicted output and  $n_o$  is the number of output neurons on the output layer.

The learning algorithm usually used in the backward pass is back-propagation. For an arbitrary ANN neuron  $j$ , the backward pass requires calculating the local gradient  $\delta$ , updating the weights  $w_{ji}$  between two consecutive neurons  $i$  and  $j$  and updating the biases  $b_j$ .

$$\delta_j^{(h)}(k) = \begin{cases} e_j^{(H)}(k) \varphi_j'(v_j^{(H)}) \\ \varphi_j'(v_j^{(h)}) \sum_{i=0}^{n_{h+1}} \delta_j^{(h+1)}(k) * w_{ji}^{(h+1)}(k) \end{cases} \quad (3.5)$$

$$w_{ji}^{(h)}(k+1) = w_{ji}^{(h)}(k) - \alpha \left( \delta_j^{(h)}(k) y_i^{(h-1)}(k) \right) \quad (3.6)$$

$$b_j^{(h)}(k+1) = b_j^{(h)}(k) + \alpha \left( \delta_j^{(h)}(k) \right) \quad (3.7)$$

where:  $H$  denotes calculations belonging to the output neuron,  $h$  denotes calculation per-

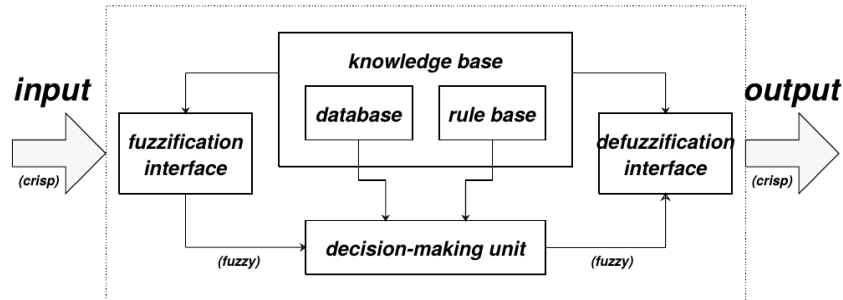


Figure 3.3: A fuzzy inference system organized in functional units as described in [3].

formed at the hidden neuron,  $\beta$  is the momentum and  $\alpha$  is the learning rate.

The learning modifies the network parameters (weights and biases) to adapt the prediction of the desired output. The integration of fuzzy logic into the ANN forward-pass gave birth to a fuzzy neural network widely used in the approximation of the inverse kinematics.

### 3.1.1.2 Fuzzy Neural Networks

As highlighted above, a Fuzzy Neural Network (FNN) came from the integration of fuzzy logic (in the form of fuzzy-rule-based systems) into the ANN forward-pass. An FNN that has been investigated in the IK approximation is the Adaptive Neuro-Fuzzy Inference System (ANFIS) developed in 1993 by J. -S Jang [3]. It is based on fuzzy “*If-Then*” rules and fuzzy inference systems. A fuzzy “*If-Then*” rule is a fuzzy conditional statement in the form of “*If A, Then B*” where A and B are characterized by membership functions. When computing the inverse kinematics, fuzzy “*If-Then*” rules used are in the form described by equation (3.12). Additionally, a fuzzy inference system is made of five functional units: a fuzzification interface unit - which transforms the crisp inputs into degrees of match with linguistic values; a database unit - which defines the membership functions of the fuzzy sets used in the fuzzy rules; a rule base unit - which contains fuzzy “*If-Then*” rules; a decision-making unit - which performs the inference operations on the rules; and a defuzzification interface unit - which transforms the fuzzy results into crisp outputs. Figure 3.3 shows the organization of a fuzzy inference system in functional units.

ANFIS architecture, made of five layers, is presented in Figure 3.4 for a Takagi-Sugeno fuzzy inference system. The first layer is made of adaptive nodes which do not require incoming weights but instead directly map the node input to the node output using a node

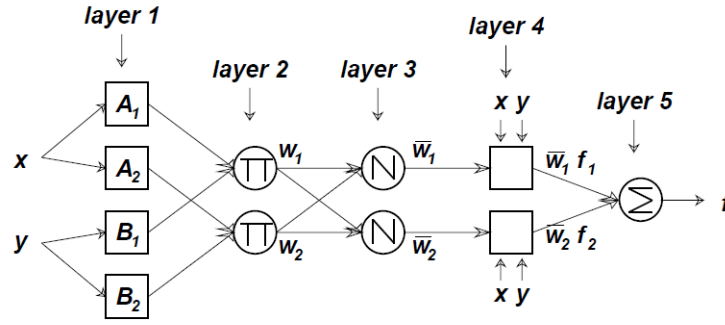


Figure 3.4: Takagi-Sugeno Type-3 ANFIS with 2 inputs [3].

function. Considering  $O_i^j$  the output representation of the  $j$ -th layer at the  $i$ -th input element from the pose vector  $D = [X, Y, Z, q_w, q_x, q_y, q_z]^T$ .

The outputs of the first layer ( $O_i^1$ ) are called the degrees of membership of each input element:

$$\begin{aligned}
 O_{A_i}^1 &= r_{A_i}(X) \\
 O_{B_i}^1 &= r_{B_i}(Y) \\
 O_{C_i}^1 &= r_{C_i}(Y) \\
 O_{D_i}^1 &= r_{D_i}(q_w) \\
 O_{E_i}^1 &= r_{E_i}(q_x) \\
 O_{F_i}^1 &= r_{F_i}(q_y) \\
 O_{G_i}^1 &= r_{G_i}(q_z)
 \end{aligned} \tag{3.8}$$

where:  $r_{A_i}$ ,  $r_{B_i}$ ,  $r_{C_i}$ ,  $r_{D_i}$ ,  $r_{E_i}$ ,  $r_{F_i}$  and  $r_{G_i}$  are the Gaussian fuzzy membership functions applied to the crisp elements of the input pose vector  $D$ .

The outputs of the second layer ( $O_i^2$ ) are called the firing strengths obtained by taking the product of the degrees of membership for each  $i$ -th input element:

$$O_i^2 = W_i = O_{A_i}^1 * O_{B_i}^1 * O_{C_i}^1 * O_{D_i}^1 * O_{E_i}^1 * O_{F_i}^1 * O_{G_i}^1 \tag{3.9}$$

The outputs of the third layer ( $O_i^3$ ) are called the normalized firing strengths. They are the ratio between the  $i$ -th firing strength and the summation of all firing strengths ( $n$ ):

$$O_i^3 = \bar{W}_i = \frac{W_i}{\sum_{i=1}^n W_i} \tag{3.10}$$

The outputs of the fourth layer ( $O_i^4$ ) are the defuzzified predicted outputs:



$$\begin{aligned}
O_i^4 &= \bar{W}_i * F_i \\
O_i^4 &= \bar{W}_i * (u_{Ai} * X + u_{Bi} * Y + u_{Ci} * Z + u_{Di} * q_x + u_{Ei} * q_y + u_{Fi} * q_z + u_{Gi} * q_w + u_{Hi})
\end{aligned} \tag{3.11}$$

where:  $u_{Ai}$ ,  $u_{Bi}$ ,  $u_{Ci}$ ,  $u_{Ei}$ ,  $u_{Fi}$ ,  $u_{Gi}$  and  $u_{Hi}$  are referred to as consequent parameters derived from fuzzy “If-Then” rules during the training. The rules are defined as: If  $X$  is  $r_A$ ,  $Y$  is  $r_B$ ,  $Z$  is  $r_C$ ,  $q_x$  is  $r_D$ ,  $q_y$  is  $r_E$ ,  $q_z$  is  $r_F$ , and  $q_w$  is  $r_G$ , then:

$$F = u_A * X + u_B * Y + u_C * Z + u_D * q_x + u_E * q_y + u_F * q_z + u_G * q_w + u_H \tag{3.12}$$

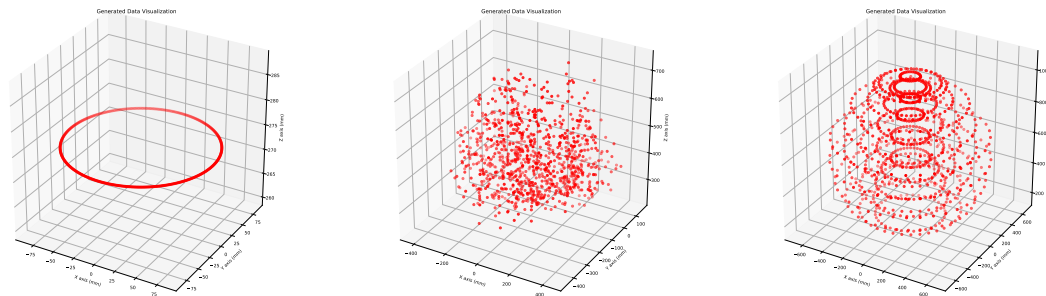
The outputs of the fifth layer ( $O^5$ ) are the aggregated defuzzified outputs:

$$O^5 = \sum_i O_i^4 \tag{3.13}$$

In this thesis, the hybrid algorithm (a combination of least squares and back-propagation algorithms) was adopted as learning algorithm based on Matlab [52] fuzzy toolbox implementation of ANFIS.

### 3.1.2 Dataset generation

The IK being specific to each robotic arm, there is no publicly available dataset to be used for IK learning problems. Therefore, the generation of a synthetic dataset is an essential step in the use of data-driven IK approaches. Based on the forward kinematics and joints variations, the synthetic dataset is made of end-effector poses (position and orientation) and joint configurations. The joints can be varied in either a structured or random fashion in the robot workspace. Figure 3.5 shows some example datasets that can be generated for serial robots. In the training process, the end-effector poses are used as ANN inputs and the corresponded joint configurations are used as desired ANN outputs. The robot workspace can be constrained or unconstrained when generating the dataset by limiting the variation ranges of the joints. In the literature, data-driven IK methods have provided good predictions in constrained workspaces. In this thesis, a preliminary investigation on the performance of MLP and ANFIS approaches have been made for unconstrained (e.g., task-independent) workspaces.



(a) A structured workspace made of a single trajectory of 360 data points. (b) A random workspace of 1000 data points. (c) A structured workspace of 1080 data points.

Figure 3.5: Overview of various workspaces generated for a 6DoF Jaco Robotic Arm from Kinova Robotics.

### 3.1.3 Learning schemes

In the literature, data-driven IK methods have been exploited with various forms of learning schemes in the ANN training to achieve acceptable prediction results. In Figure A.2, an overview of these learning schemes is presented where the ANN learns all the joints at the same time. These same schemes may be used for single output ANN where a network learns one joint at a time to predict the inverse kinematics of a robotic arm. The input parameters representing the orientation of the end-effector can be expressed based on the elements of the rotation matrix, the *Roll – Pitch – Yaw* angles or the quaternion representation. However, no extensive work has been done to determine which of the learning schemes provides better prediction results. Also, in the literature, the authors do not provide information justifying the choice of the learning schemes used in their work.

## 3.2 Numerical method

A robust and fast numerical IK approach, that relies on the numerical estimations of the inverse Jacobian matrix, was developed by Siavash and DeSouza [4]. Provided the D-H representation of a robotic arm and the final desired position; the algorithm is able to find a solution to the IK achieving sub-millimeter and sub-degrees accuracies when such a solution exists. Also, the serial and parallel implementations [4, 53] allow the algorithm to be faster and easily adaptable to various applications. Instead of building the Jacobian

matrix in one step and taking its inverse, the algorithm builds the Jacobian matrix step by step by applying a small perturbation to each joint at a time as depicted by Figure 3.6.

That is, let us consider the two vectors:  $\vec{D}(t)$  and  $\vec{Q}(t)$ . On the one hand,  $\vec{D}(t) = [X, Y, Z, Ro, Pi, Ya]^T$  is the end-effector pose vector where  $(X, Y, Z)$  components describe its position and  $(Ro, Pi, Ya)$  components describe its orientation with *Roll*, *Pitch* and *Yaw* angles. On the other hand,  $\vec{Q}(t) = [Q_1, Q_2, \dots, Q_n]^T$  is the current joint configuration vector where  $Q_i = d_i$  represents a prismatic (or linear) joint,  $Q_i = \theta_i$  represents a revolute (or rotative) joint and  $n$  the number of DoF. Based on the background equation (2.3), here the Jacobian matrix is derived by the following:

$$\vec{D}(t) = f_{FK}(\vec{Q}(t)) \quad (3.14)$$

$$\begin{bmatrix} X \\ Y \\ Z \\ Ro \\ Pi \\ Ya \end{bmatrix} = \begin{bmatrix} f_1(\vec{Q}(t)) \\ f_2(\vec{Q}(t)) \\ f_3(\vec{Q}(t)) \\ f_4(\vec{Q}(t)) \\ f_5(\vec{Q}(t)) \\ f_6(\vec{Q}(t)) \end{bmatrix} \quad (3.15)$$

$$\partial \vec{D} = J * \partial \vec{Q} \quad (3.16)$$

$$\text{Where: } J = \begin{bmatrix} \frac{\partial f_1(\vec{Q}(t))}{\partial Q_1} & \frac{\partial f_1(\vec{Q}(t))}{\partial Q_2} & \dots & \frac{\partial f_1(\vec{Q}(t))}{\partial Q_n} \\ \frac{\partial f_2(\vec{Q}(t))}{\partial Q_1} & \frac{\partial f_2(\vec{Q}(t))}{\partial Q_2} & \dots & \frac{\partial f_2(\vec{Q}(t))}{\partial Q_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_6(\vec{Q}(t))}{\partial Q_1} & \frac{\partial f_6(\vec{Q}(t))}{\partial Q_2} & \dots & \frac{\partial f_6(\vec{Q}(t))}{\partial Q_n} \end{bmatrix}$$

As shown in Figure 3.6, the Jacobian matrix is numerically estimated by applying a small disturbance to each joint variable allowing to build the Jacobian one column at a

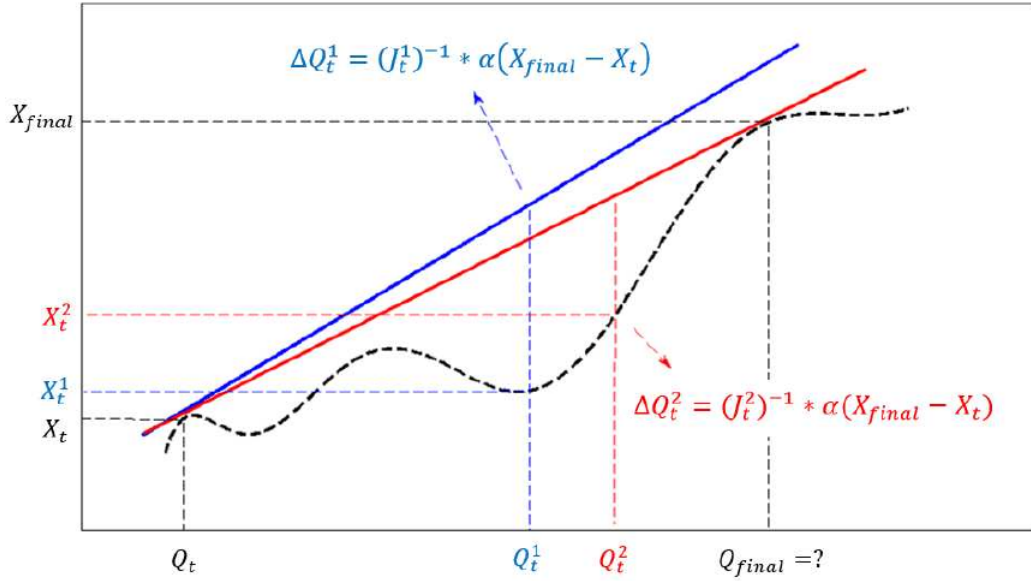


Figure 3.6: Visual representation of the hybrid inverse kinematics solver proposed in [4].

time. All the steps followed in this approach is found in the pseudo-code 3.1 of the serial implementation of this numerical IK approach [4].

---

**Algorithm 3.1** Numerical IK Solver based on Inverse Jacobian [4]

---

1: <b>procedure</b> IK( $\vec{Q}_{final}$ )	▷ $\vec{Q}$ is the final joint configuration vector
2: $n_Q \leftarrow$ number of joints	▷ $n_Q$ is related to the number of DoF
3: $\vec{Q}_{t_0} \leftarrow$ joints configuration	▷ $\vec{Q}_{t_0}$ is the initial joint configuration vector
4: $\vec{X}_{t_0} = f(\vec{Q}_{t_0})$	▷ $\vec{X}_{t_0}$ is the initial pose from the forward kinematics
5: <b>while</b> $\ \vec{X}_t - X_{final}\  > \epsilon_r$ <b>do</b>	▷ Repeat until the desired error $\epsilon_r$ is reached
6: <b>for each joint</b> $c$ <b>do</b>	▷ Build the jacobian matrix columnwise
7: $J_c = \frac{\partial \vec{X}}{\partial \vec{q}_c} = \vec{X}_t - f(\vec{Q}_t + \partial \vec{q}_c)$	▷ Add small perturbation to each joint
8: <b>end for</b>	
9: $J_t = \vec{X}_t - f(\vec{Q}_t + \partial \vec{q}_t)$	▷ Build the entire jacobian columnwise
10: $\Delta \vec{Q}_t = J_t^{-P} * \alpha_t(X_{final} - \vec{X}_t)$	▷ $\Delta \vec{Q}_t$ is obtained with the inverse jacobian
11: $\vec{Q}_{t+1} = \vec{Q}_t + \Delta \vec{Q}_t$	▷ Get the next corresponding joint vector
12: $\vec{X}_{t+1} = f(\vec{Q}_t)$	▷ Get the next corresponding pose vector
13: <b>end while</b>	
14: <b>return</b> $\vec{Q}_{final} = \vec{Q}_{t+1}$	▷ The final estimated joint configuration is returned
15: <b>end procedure</b>	

---

In this thesis, we focus on two important aspects depicted by the equation (3.17) expressed by the line 10 of the algorithm 3.1: the attenuation parameter  $\alpha_t$  and the inverse Jacobian  $J_t^{-P}$ .

$$\Delta \vec{Q} = J_t^{-P} * \alpha_t (\vec{X}_{final} - \vec{X}_t) \quad (3.17)$$

The inverse Jacobian  $J_t^{-P}$  is obtained using the MP Inverse. Because the MP Inverse does not always guarantee reliable IK solutions in the presence of incommensurate systems as previously explained in Chapter 2. We propose to investigate the use of the UC and MX Inverses in the algorithm and its role for more stable IK solutions.

The attenuation parameter  $\alpha$  is a parameter between  $[0, 1]$  of the algorithm that characterizes how much the algorithm needs to take between the current estimated pose and the final desired pose of the end-effector. It can affect the path of the end-effector, and also the convergence time of the algorithm [53]. We conduct experiments with the serial implementation of the algorithm to investigate the role of  $\alpha$  in the behavior of the end-effector trajectory.

# Chapter 4

## Experimental Results and Discussions

This chapter presents the results of all conducted experiments. Consequently, the evaluations of the performances, advantages, and disadvantages of the investigated approaches are presented for different robotic manipulators.

### 4.1 Data-Driven IK Methods

#### 4.1.1 Experimental Setup and Selected Robotic Arms

In this section, we investigated the level of accuracy provided by the use of *MLP* and *ANFIS* networks to predict the IK of robotic manipulators in task-free workspaces. All twenty-four experiments performed are reported here. Table 4.1 presents an overview of all these experiments. The primary goal of the experiments was to evaluate the potential of the neural networks under three case studies:

*Case study 1 - All-Joints-ANN*: For each specific robot, one MLP learns all the output joints at the same time (see Figure 4.5).

*Case study 2 - Individual-Joints-ANN*: For the same robot, each MLP learns one output joint at a time (see Figure 4.11).

*Case study 3 - Individual-Joints-ANFIS*: For each robot, each ANFIS learns one output joint at a time (see Figure 4.12)..

For each case study, MSE was used for quality measurement of the predictions. In all experiments, serial robots with 4, 5, 6, and 7 DoF were investigated. The 4 and 6 DoF

Experiments	Robotic Arm	Dataset Type	Network Type
1, 2, 3	4DoF	Structured	All-Joints-ANN Individual-Joints-ANN Individual-Joints-ANFIS
4, 5, 6	5DoF		
7, 8, 9	6DoF		
10, 11, 12	7DoF		
13, 14, 15	4DoF	Random	
16, 17, 18	5DoF		
19, 20, 21	6DoF		
22, 23, 24	7DoF		

Table 4.1: Overview of the twenty-four experiments conducted with MLP (All-Joints-ANN and Individual-Joints-ANN) and ANFIS (Individual-Joints-ANFIS).

manipulators had a combination of prismatic and revolute joints, while the 5 and 7 DoF manipulators had only revolute joints. These robotic manipulators are illustrated in Figure 4.1. Additionally, Table 4.2 presents the D-H parameters of the chosen robotic arms.

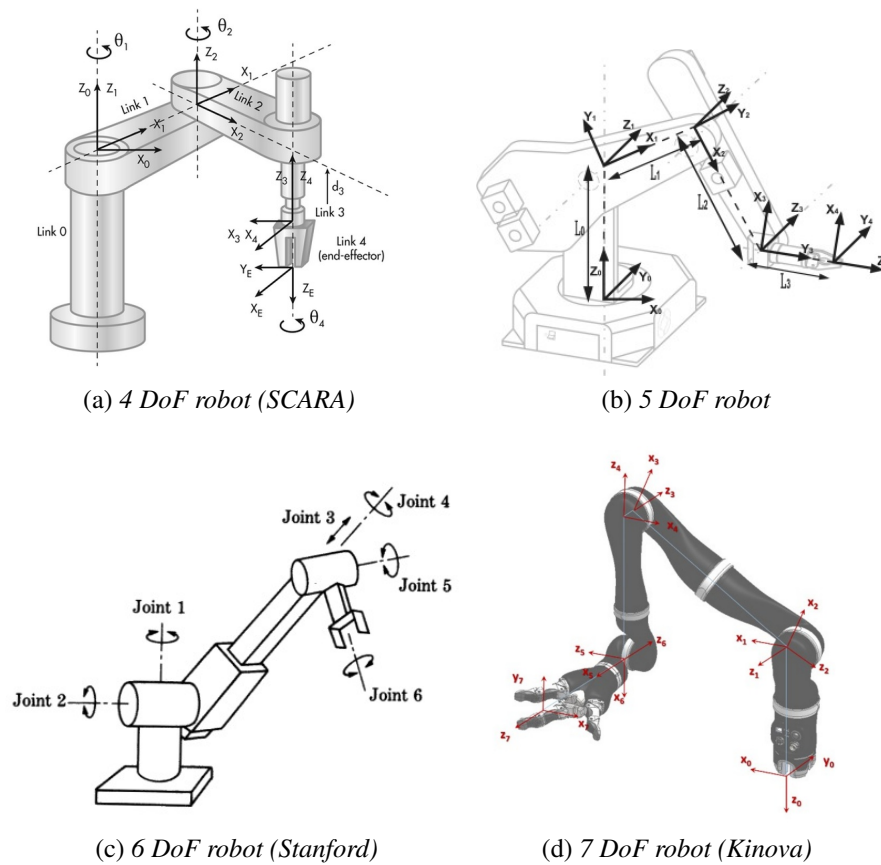


Figure 4.1: Robotic arms used in the Data-Driven Experiment in Section 4.1.

$i$	$\theta$	$d$	$a$	$\alpha$
1	$\theta_1$	400	250	0
2	$\theta_2$	0	150	180
3	0	$d_3$	0	0
4	$\theta_4$	150	0	0

(a) 4 DoF (SCARA)

$i$	$\theta$	$d$	$a$	$\alpha$
1	$\theta_1$	380	0	0
2	$\theta_2$	0	280	-90
3	$\theta_3$	0	280	0
4	$\theta_4$	0	80	-90
5	$\theta_5$	0	0	90

(b) 5 DoF

$i$	$\theta$	$d$	$a$	$\alpha$
1	$\theta_1$	0	0	-90
2	$\theta_2$	140	0	90
3	0	$d_3$	0	0
4	$\theta_4$	0	0	-90
5	$\theta_5$	0	0	90
6	$\theta_6$	8.5	0	0

(c) 6 DoF (Stanford)

$i$	$\theta$	$d$	$a$	$\alpha$
1	$\theta_1$	-275.5	0	90
2	$\theta_2$	0	0	90
3	$\theta_3$	-410	0	90
4	$\theta_4$	-9.8	0	90
5	$\theta_5$	-311.1	0	90
6	$\theta_6$	0	0	90
7	$\theta_7$	-263.8	0	180

(d) 7 DoF (Kinova)

Table 4.2: D-H table with the parameters of the serial robots illustrated in Figure 4.1. The angles  $\theta$  and  $\alpha$  are expressed in degrees. The variables  $d$  and  $a$  are expressed in millimeters (mm).

### 4.1.2 Workspaces and Dataset Generation:

Even though we always considered task-free workspaces in this research, for each of the robotic arm used, two different types of datasets were generated for training, validation and testing: one *structured* and one *random* dataset. While the structured dataset consisted of end-effector poses following a certain pattern in the robot workspace, the random dataset consisted of randomly generated end-effector poses. Both datasets were evaluated using all three NN approaches described in the case studies above. During the generation of the datasets, the robotic joints were constrained within certain ranges due to the size and shape of the robots, and to limit the number of data points in the workspace. Those ranges are presented in Tables 4.3 and 4.4 respectively for the structured and random datasets. The D-H tables for the four manipulators are presented in Table 4.2, and the actual positions of the end-effectors are illustrated by Figures 4.2 and 4.3 for the structured and random datasets, respectively.



$i$	joint	range
1	$\theta_1$	$[0^\circ, 360^\circ]$
2	$\theta_2$	$[50^\circ, 100^\circ]$
3	$d_3$	$[100, 250mm]$
4	$\theta_4$	$[30^\circ, 60^\circ]$

(a) 4 DoF robotic arm (SCARA)

$i$	joint	range
1	$\theta_1$	$[0^\circ, 360^\circ]$
2	$\theta_2$	$[30^\circ, 60^\circ]$
3	$\theta_3$	$[20^\circ, 70^\circ]$
4	$\theta_4$	$[0^\circ, 50^\circ]$
5	$\theta_5$	$[200^\circ, 240^\circ]$

(b) 5 DoF robotic arm

$i$	joint	range
1	$\theta_1$	$[0^\circ, 360^\circ]$
2	$\theta_2$	$[50^\circ, 150^\circ]$
3	$d_3$	$[100, 220mm]$
4	$\theta_4$	$[50^\circ, 90^\circ]$
5	$\theta_5$	$[200^\circ, 240^\circ]$
6	$\theta_6$	$[0^\circ, 20^\circ]$

(c) 6 DoF robotic arm (Stanford)

$i$	joint	range
1	$\theta_1$	$[0^\circ, 360^\circ]$
2	$\theta_2$	$[30^\circ, 60^\circ]$
3	$\theta_3$	$[90^\circ, 140^\circ]$
4	$\theta_4$	$[180^\circ, 200^\circ]$
5	$\theta_5$	$[180^\circ, 200^\circ]$
6	$\theta_6$	$[0^\circ, 20^\circ]$
7	$\theta_7$	$[0^\circ, 20^\circ]$

(d) 7 DoF robotic arm (Kinova)

Table 4.3: Range of joint values used for generating the structured workspaces.

$i$	joint	range
1	$\theta_1$	$[0^\circ, 180^\circ]$
2	$\theta_2$	$[0^\circ, 120^\circ]$
3	$d_3$	$[0, 80mm]$
4	$\theta_4$	$[0^\circ, 90^\circ]$

(a) 4 DoF robotic arm (SCARA)

$i$	joint	range
1	$\theta_1$	$[0^\circ, 180]$
2	$\theta_2$	$[50^\circ, 150^\circ]$
3	$\theta_3$	$[50^\circ, 120^\circ]$
4	$\theta_4$	$[0^\circ, 50^\circ]$
5	$\theta_5$	$[0^\circ, 20^\circ]$

(b) 5 DoF robotic arm

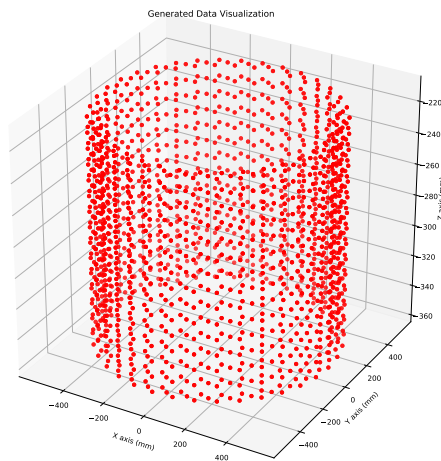
$i$	joint	range
1	$\theta_1$	$[0^\circ, 180^\circ]$
2	$\theta_2$	$[60^\circ, 150^\circ]$
3	$d_3$	$[0, 200mm]$
4	$\theta_4$	$[50^\circ, 90^\circ]$
5	$\theta_5$	$[60^\circ, 90^\circ]$
6	$\theta_6$	$[30^\circ, 60^\circ]$

(c) 6 DoF robotic arm (Stanford)

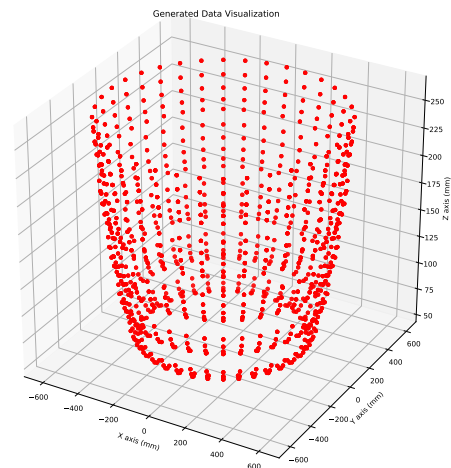
$i$	joint	range
1	$\theta_1$	$[0^\circ, 360^\circ]$
2	$\theta_2$	$[0^\circ, 60^\circ]$
3	$\theta_3$	$[0^\circ, 30^\circ]$
4	$\theta_4$	$[0^\circ, 30^\circ]$
5	$\theta_5$	$[0^\circ, 30^\circ]$
6	$\theta_6$	$[0^\circ, 20^\circ]$
7	$\theta_7$	$[0^\circ, 20^\circ]$

(d) 7 DoF robotic arm (Kinova)

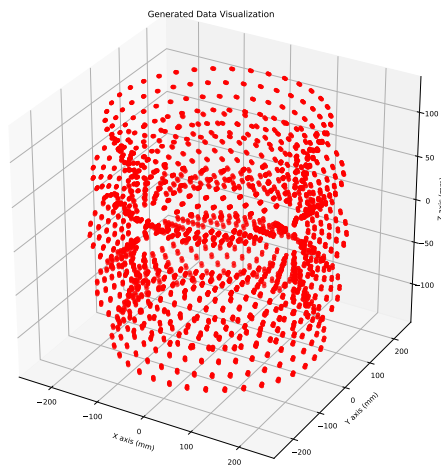
Table 4.4: Range of joint values used for generating the random workspace.



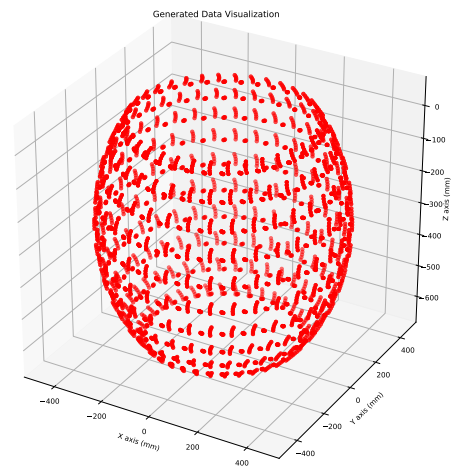
(a) The actual 4800 data points in the structured dataset used for the 4 DoF robot (SCARA)



(b) The actual 5400 data points in the structured dataset used for the 5 DoF robot

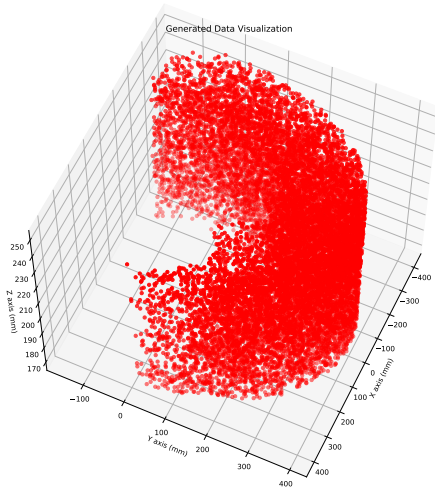


(c) The actual 8640 data points in the structured dataset used for the 6 DoF robot (Stanford)

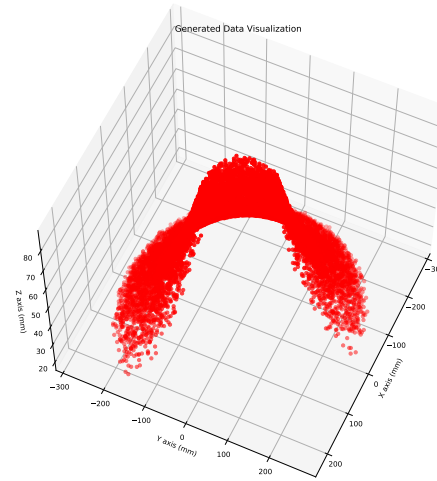


(d) The actual 7200 data points in the structured dataset used for the 7 DoF robot (Kinova)

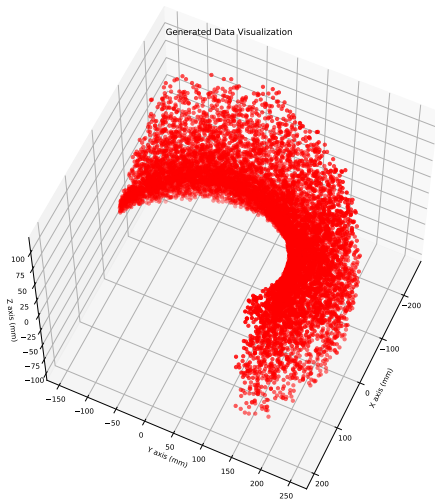
Figure 4.2: A depiction of the positions of the end-effector in the workspaces of the chosen robots for the structured dataset. The shapes of the data points in the workspaces depended mainly on the joint ranges presented in Table 4.3.



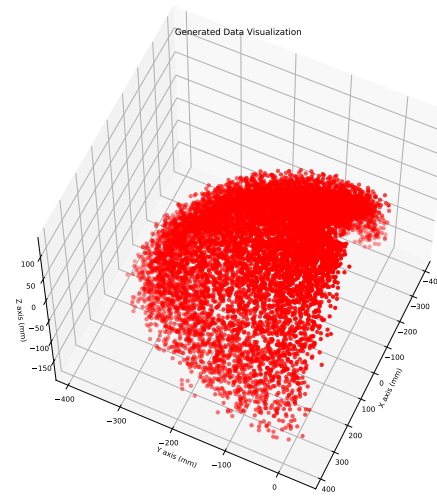
(a) The actual 15,000 data points in the random dataset used for the 4 DoF robot (SCARA)



(b) The actual 15,000 data points in the random dataset used for the 5 DoF robot



(c) The actual 10,000 data points in the random dataset used for the 6 DoF robot (Stanford)



(d) The actual 10,000 data points in the random dataset used for the 7 DoF robot (Kinova)

Figure 4.3: A depiction of the positions of the end-effector in the workspaces of the chosen robots for the random dataset. The shapes of the data points in the workspaces depended mainly on the joint ranges presented in Table 4.4.

Finally, Figure 4.4 depicts the entire learning process adopted for the experiments in this section. This process describes the main steps followed: from the choice of the robotic arm to the learning process when using a data-driven IK method. As the figure implies, first a robot is selected together with its dataset. Next, the dataset is randomly split into training, validation and testing subsets. The training subset is used to train the paradigm (ANN or ANFIS) by successively applying each pose from the training set and

comparing with the desired joint configuration (output of the NN). Finally, the error in prediction is used to correct the internal parameters of the paradigm (learn) according to their learning strategies. The same Figure 4.4 can be used to explain the validation and testing phases of the learning by simply considering the corresponding subsets instead of the training subset as in the figure.

The learning strategies and the partition of the dataset into training, validation and testing will be further detailed in the next sections.

### 4.1.3 Network Architecture Search

#### 4.1.3.1 All-Joints-ANN

The MLP architecture presented in Figure 4.5 was used for the All-Joints-ANN case study. This network comprised 4 hidden layers, 7 inputs neurons ( $[X, Y, Z, q_w, q_x, q_y, q_z]$ ), and outputs  $Q_i = d_i$  for the prismatic or  $Q_i = \theta_i$  for the revolute joints, where  $i = 1, \dots, n$  and  $n = 4, 5, 6$  and 7 represents the DoF of the manipulators. For this experiment, all networks had  $n$  output neurons (e.g,  $n = 5$  outputs neurons for the 5 DoF robot).

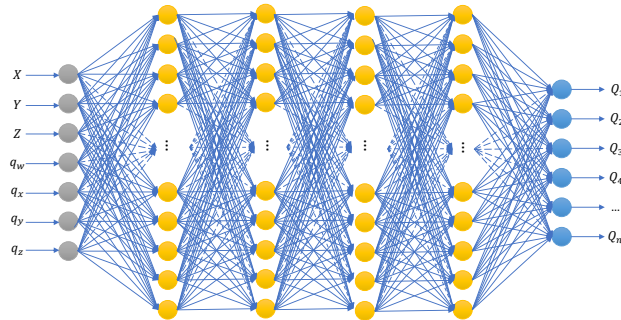


Figure 4.5: Representation of the network architecture of the All-Joints-ANN. ANNs were implemented with Keras [5] using TensorFlow CPU backend running on a desktop with an Intel(R) Core(TM) i7-8700CPU and NVIDIA GTX 2060 GPU.

When designing network architectures, authors often relies on their design experience or on Neural Architecture Search (NAS) methods [54, 55]. In this study, while the input and output neurons respectively on the input and output layers were fixed based on the datasets, the best dataset-related number of neurons on the hidden layers were determined

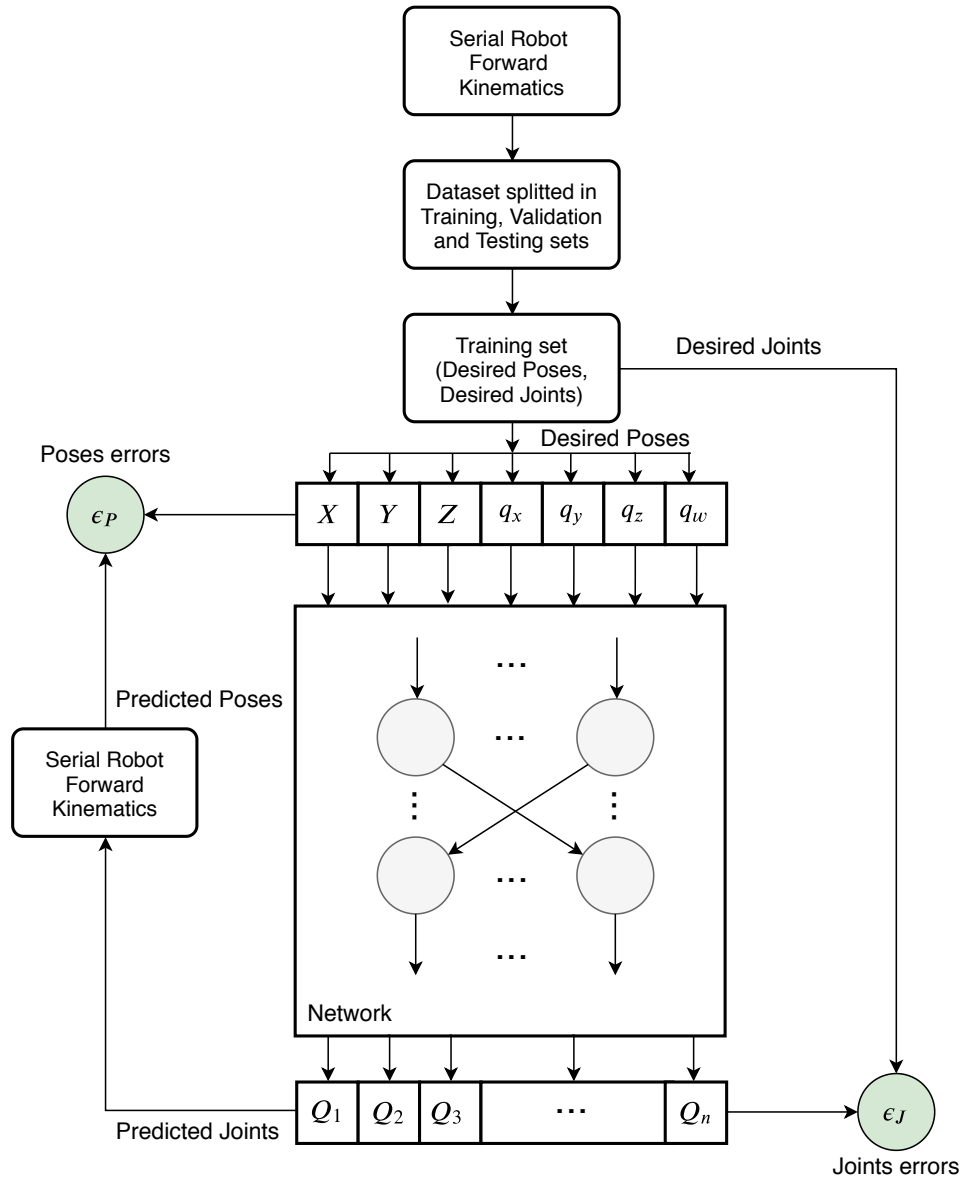


Figure 4.4: Overview of the learning process used in the implementation of the data-driven IK method.

through a simple but time-consuming neural architectural search. The best number of hidden nodes in each of the four hidden layers for the final network architecture was determined upon the search process detailed in Figure 4.6. In fact, with each of the MLP network type considered in this study, ten training experiments were run while varying the number of hidden nodes ( $n_H$ ) in the hidden layers ( $H$ ). Initially the network architecture would have  $n_H = 2$  for all  $H = 4$  hidden layers. Then, the network would be trained and the loss would be evaluated and recorded. After training this architecture, a new network architecture would be considered and trained with  $n_H = n_H + 2$  for all  $H = 4$  hidden layers, and this process would be repeated until  $n_H = 30$ . The search for the final network architecture was performed with each of the generated datasets for the four robotic arms.

For the All-Joints-ANN, the averaged training and validations losses monitored during the searches for the MLP architecture are presented in Figures 4.7, 4.8, 4.9 and 4.10

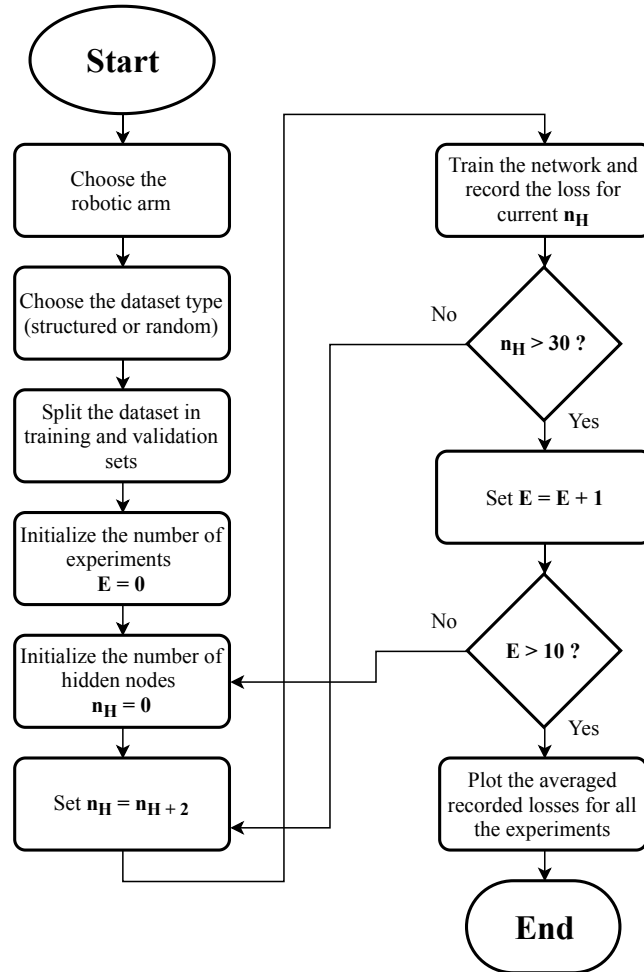
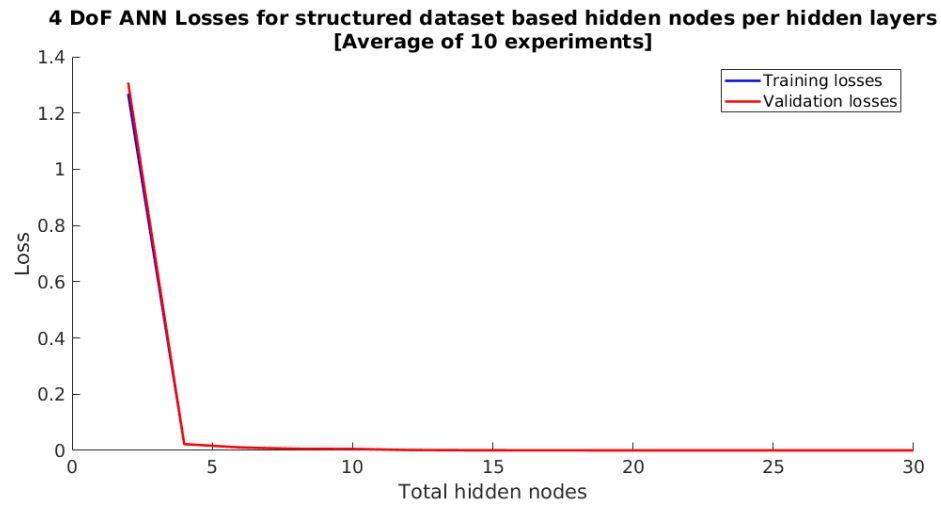
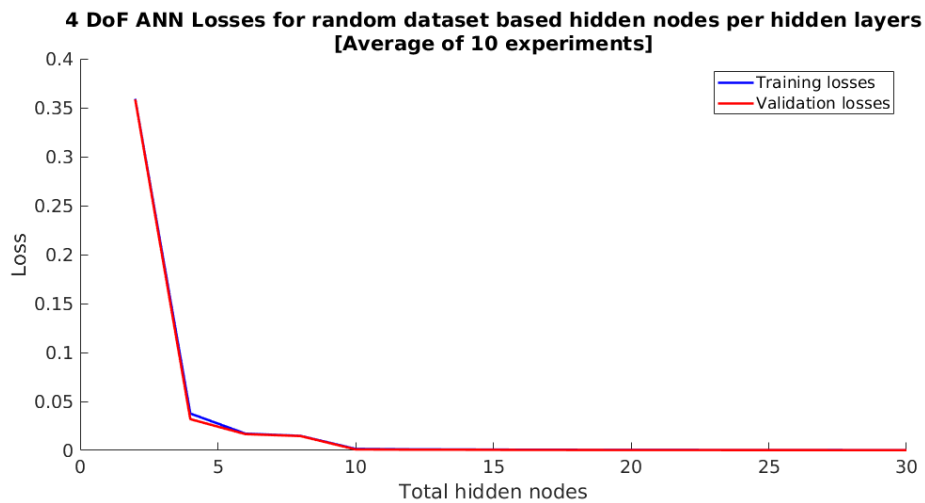


Figure 4.6: Process allowing to search for the final network architectures used for the experiments.



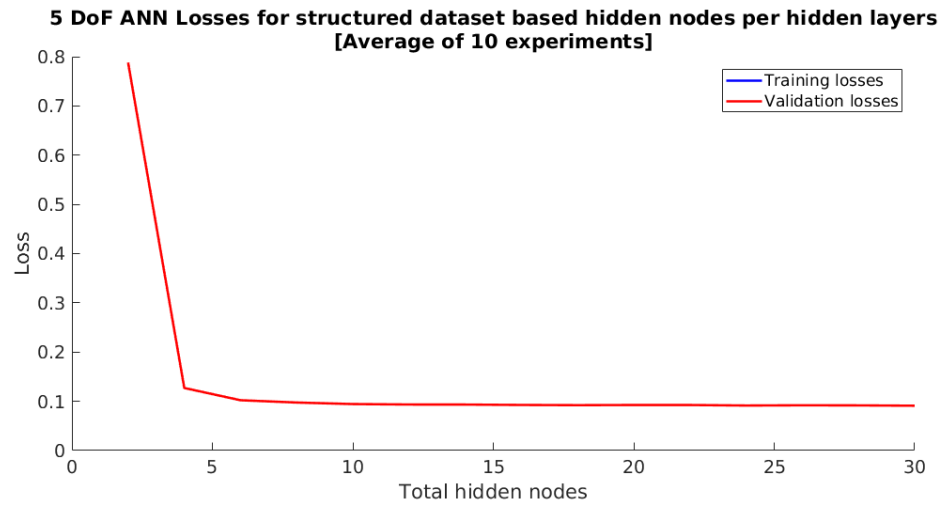
(a)



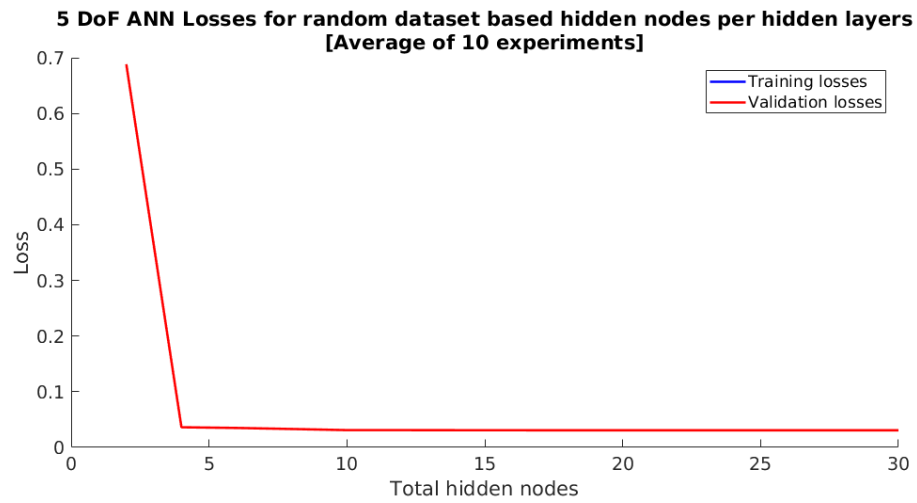
(b)

Figure 4.7: Averaged training and validation losses from the search process of finding the final network architecture to be used for the experiments with the 4DoF robot.



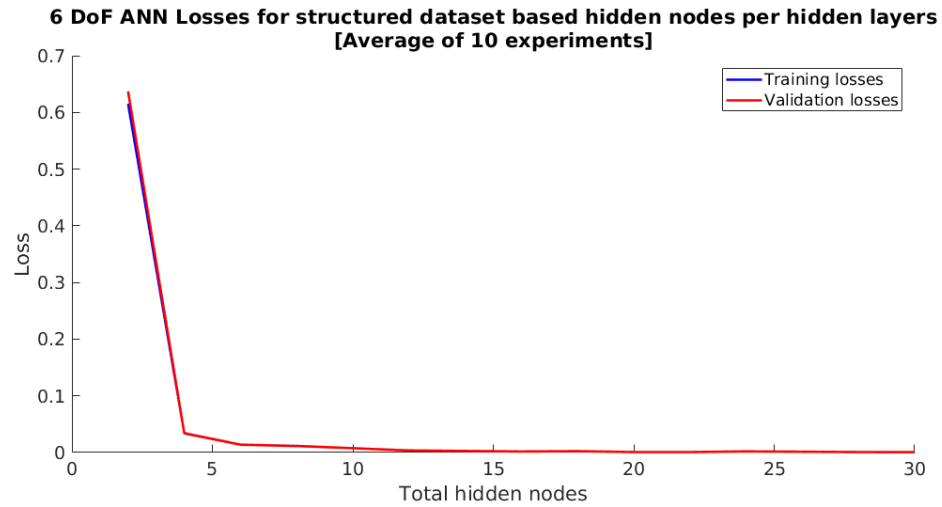


(a)

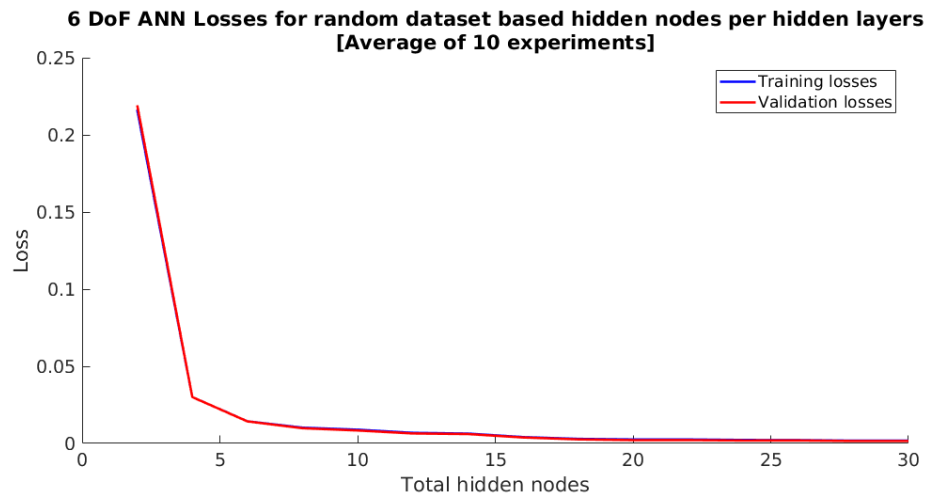


(b)

Figure 4.8: Averaged training and validation losses from the search process of finding the final network architecture to be used for the experiments with the 5DoF robot.

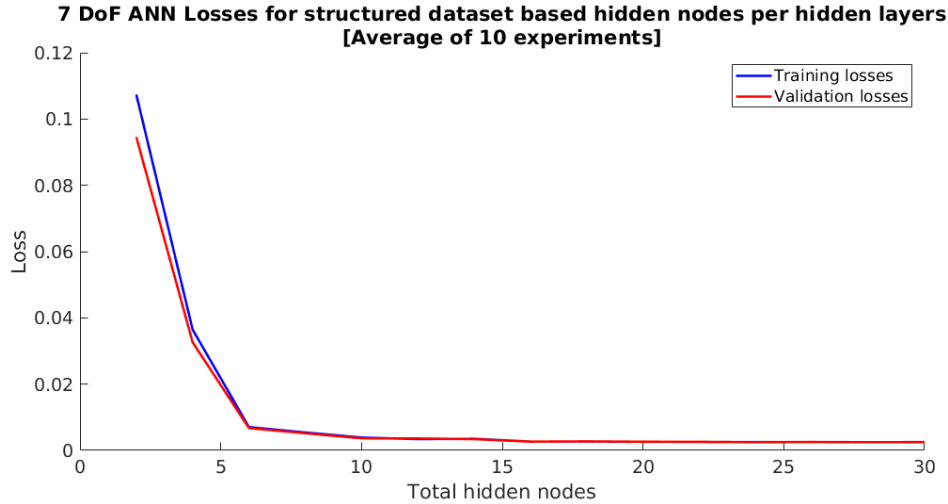


(a)

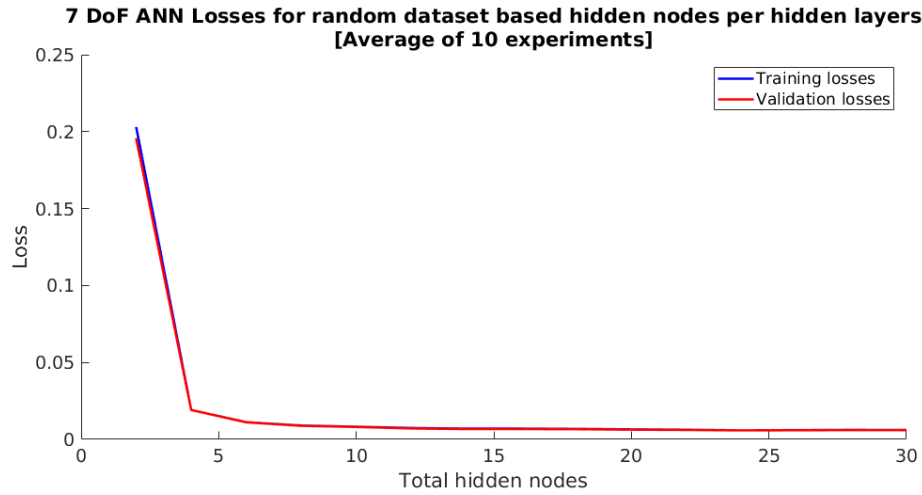


(b)

Figure 4.9: Averaged training and validation losses from the search process of finding the final network architecture to be used for the experiments with the 6DoF robot.



(a)



(b)

Figure 4.10: Averaged training and validation losses from the search process of finding the final network architecture to be used for the experiments with the 7DoF robot.

Tables 4.5 presents the final number of hidden neurons retained at all hidden layers after the network architecture search using the structured and random datasets for all the manipulators.

#### 4.1.3.2 Individual-Joints-ANN

Similarly, the MLP architecture presented in Figure 4.11 was used for the Individual-Joints-ANN case study. This network also comprised 4 hidden layers, 7 inputs neurons ( $[X, Y, Z, q_w, q_x, q_y, q_z]$ ), and outputs  $Q_i = d_i$  for the prismatic or  $Q_i = \theta_i$  for the revolute joints, where  $i = 1, \dots, n$  and  $n = 4, 5, 6$  and 7 represents the DoF of the manipulators. For

Structured datasets	4DoF	5DoF	6DoF	7DoF
Nodes/Hidden Layer(1,2,3,4)	8	10	14	18

(a) *Structured datasets*

Random datasets	4DoF	5DoF	6DoF	7DoF
Nodes/Hidden Layer(1,2,3,4)	10	12	20	24

(b) *Random datasets*

Table 4.5: An overview of the final number of hidden neurons retained for the structured and random datasets after network architecture search for all the manipulators.

this experiment, each network had only one output neuron, which required the training of  $n$  networks: one for each DoF of the robotic arm (i.e.,  $n$  networks for the  $n$  DoF robot).

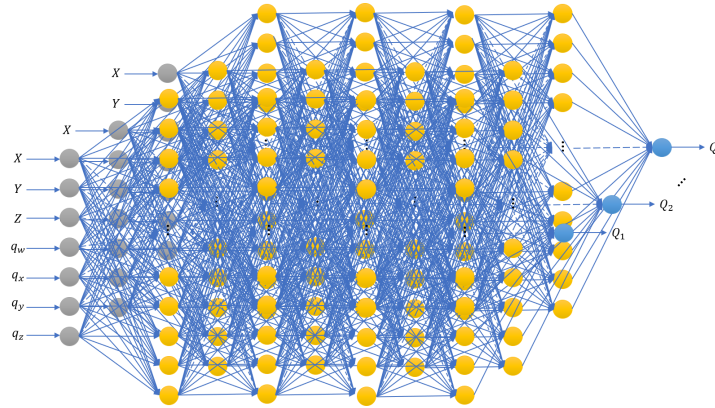


Figure 4.11: Representation of the  $n$  network architectures for the Individual-Joints-ANN case study. ANNs were implemented with Keras [5] using TensorFlow CPU backend running on a desktop with an Intel(R) Core(TM) i7-8700CPU and NVIDIA GTX 2060 GPU.

For this Individual-Joints-ANN case, the search for the best number of nodes in the hidden layers followed a process similar to the All-Joints-ANN case. For sake of space, the plots with the average losses during training and validation are being omitted.

#### 4.1.3.3 Individual-Joints-ANFIS

For the *Individual-Joints-ANFIS* case study, the network architecture was based on the original ANFIS paper by Jang ([3]), which had a fixed set of hidden layers and hidden neurons – this was also presented in Chapter 3. This network also comprised 4 hidden layers, 7 inputs neurons ( $[X, Y, Z, q_w, q_x, q_y, q_z]$ ), and outputs  $Q_i = d_i$  for the prismatic or  $Q_i = \theta_i$  for the revolute joints, where  $i = 1, n$  and  $n = 4, 5, 6$  and 7 represents the DoF

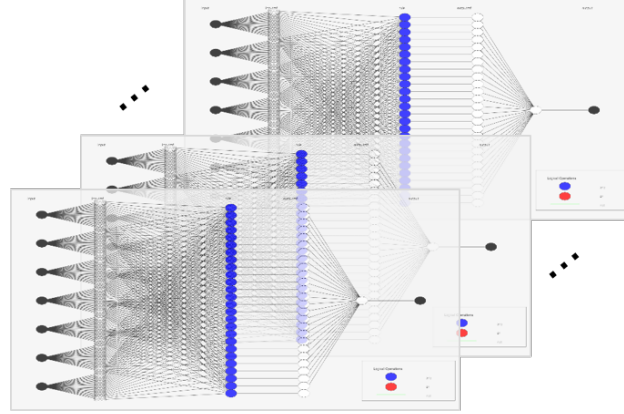


Figure 4.12: Architecture of the Adaptive Neuro-Fuzzy Inference System (ANFIS) network used in this case study, with 7 inputs and 1 output. This structure was repeated  $n$  times: one for each DoF of the manipulator in question.

Joints	Structured dataset: MF/input	Random dataset: MF/input
$\theta_1$	14	36
$\theta_2$	25	51
$d_3$	15	37
$\theta_4$	27	42

Table 4.6: An overview of the number of membership functions (MF) determined using Subtractive Clustering Algorithm [6] for structured and random datasets with the 4DoF manipulator.

of the manipulators. As in the previous case, each network had only one output neuron, which also required the training of  $n$  networks: one for each DoF of the robotic arm (i.e.,  $n$  networks for the  $n$  DoF robot).

In Figure 4.12, we illustrate the architecture used in the *Individual-Joints-ANFIS* case study. The ANFIS implementation used the Matlab Fuzzy Toolbox.

When training the *Individual-Joints-ANFIS* networks, the number of membership functions was found using the subtractive clustering algorithm [6]. This algorithm builds a fuzzy inference system where the number membership functions are extracted based on the input and output variables through clustering. Tables 4.6, 4.7, 4.8 and 4.9 present the number of membership functions used per network input respectively for the 4, 5, 6 and 7 DoF manipulators.

Joints	Structured dataset: MF/input	Random dataset: MF/input
$\theta_1$	12	22
$\theta_2$	23	21
$\theta_3$	12	32
$\theta_4$	19	51
$\theta_5$	21	57

Table 4.7: An overview of the number of membership functions (MF) determined using Subtractive Clustering Algorithm [6] for structured and random datasets with the 5DoF manipulator.

Joints	Structured dataset: MF/input	Random dataset: MF/input
$\theta_1$	12	11
$\theta_2$	11	13
$d_3$	16	17
$\theta_4$	19	19
$\theta_5$	19	20
$\theta_6$	19	20

Table 4.8: An overview of the number of membership functions (MF) determined using Subtractive Clustering Algorithm [6] for structured and random datasets with the 6DoF manipulator.

Joints	Structured dataset: MF/input	Random dataset: MF/input
$\theta_1$	13	12
$\theta_2$	11	13
$d_3$	21	23
$\theta_4$	11	20
$\theta_5$	11	22
$\theta_6$	21	21
$\theta_7$	21	20

Table 4.9: An overview of the number of membership functions (MF) determined using Subtractive Clustering Algorithm [6] for structured and random datasets with the 7DoF manipulator.

#### 4.1.4 Experimental Results for the Best NN Architectural Choices

In this section, we present the experimental results for the three data-driven methods used. The main goal of these experiments was twofold: 1) to determine which number of outputs leads to the best performance for the ANNs – i.e. the *All-Joints* vs. *Individual-Joints* approaches for the ANN; and 2) to evaluate the performance of ANN and ANFIS under the same per-joint basis – i.e. the *Individual-Joints* approaches for ANN and ANFIS. All three approaches were applied to all four robots using the datasets as described earlier. Both the predicted joint configurations (Joint-MSE) and the reconstructed end-effector Cartesian poses (Pose-MSE) were computed. For all the experiments, the datasets were randomly divided into 80% for training, 10% for validation and 10% for testing sets.

##### Structured Datasets

Tables 4.11, 4.13, 4.15 and 4.17 summarize the results for goals #1 (columns 1 vs. 2) and #2 (columns 2 vs. 3), using the structured datasets for robots with 4, 5, 6, and 7 DoF, respectively. In general, i.e. for all cases studied using the structured datasets, the errors (MSE) in joint and reconstructed pose were not as expected. That is, even though this is a challenging problem – to learn the large search space of task-independent IK scenarios – our expectations were for errors in the order of millimeter for distances and sub-degree for angles. Yet, we observed errors in joint angles as large as 7 degrees in average, and a few cm’s for reconstructed positions of the end effector.

	All-Joints-ANN	Individual-Joints-ANN	Individual-Joints-ANFIS
Aggregation (in the forward pass)	Induced local field	Induced local field	Product (layer 2) and Weighted average (layer 4)
Learning algorithm (in the backward pass)	Back-propagation	Back-propagation	Hybrid (Back-propagation + Least Square)
Activation or Membership function	Relu	Relu	Gaussian
Epochs	1000	1000	1000

Table 4.10: Learning strategies for the experiment with the 4DoF robot reported in Table 4.11.

	All-Joints-ANN (min/max)	Individual-Joints-ANN (min/max)	Individual-Joints-ANFIS (min/max)
Joint-MSE in $rad^2/m^2$			
$\theta_1$	4.175e-4 (0.000e+0/2.280e-2)	<b>1.054e-4</b> (0.000e+0/2.158e-3)	1.469e-2 (1.949e-8/1.612e-1)
$\theta_2$	3.632e-3 (3.542e-11/4.058e-2)	5.910e-4 (1.450e-9/9.340e-3)	<b>2.707e-5</b> (6.064e-10/2.301e-4)
$d_3$	1.744e-6 (1.000e-14/2.270e-5)	2.136e-9 (6.400e-15/5.335e-8)	<b>3.043e-34</b> (0.000e+0/3.081e-33)
$\theta_4$	2.342e-3 (2.111e-9/2.222e-2)	1.071e-4 (3.629e-11/1.779e-3)	<b>5.250e-5</b> (1.499e-12/8.284e-4)
Reconstructed Pose-MSE in $rad^2/m^2$			
$X$	6.318e-5 (4.464e-14/1.767e-3)	<b>7.683e-6</b> (3.807e-11/7.767e-5)	5.924e-4 (1.016e-9/7.493e-3)
$Y$	3.796e-5 (1.342e-9/7.781e-4)	<b>1.250e-5</b> (1.265e-11/1.764e-4)	1.200e-3 (1.681e-10/2.090e-2)
$Z$	1.744e-6 (1.000e-14/2.270e-5)	2.136e-9 (6.400e-15/5.335e-8)	<b>3.043e-34</b> (0.000e+0/3.081e-33)
$Ro$	1.152e-3 (1.166e-9/2.577e-2)	<b>6.152e-4</b> (8.962e-12/5.879e-3)	1.477e-2 (1.322e-10/1.573e-1)
$Pi$	2.879e-35 (1.125e-41/4.070e-34)	<b>4.543e-36</b> (1.859e-40/5.194e-35)	1.282e-34 (2.888e-41/1.417e-33)
$Ya$	1.991e-35 (3.749e-42/2.122e-34)	<b>3.842e-36</b> (9.033e-41/6.187e-35)	9.119e-35 (2.033e-41/9.062e-34)

Table 4.11: Results for 4 DoF robot using a structured dataset.



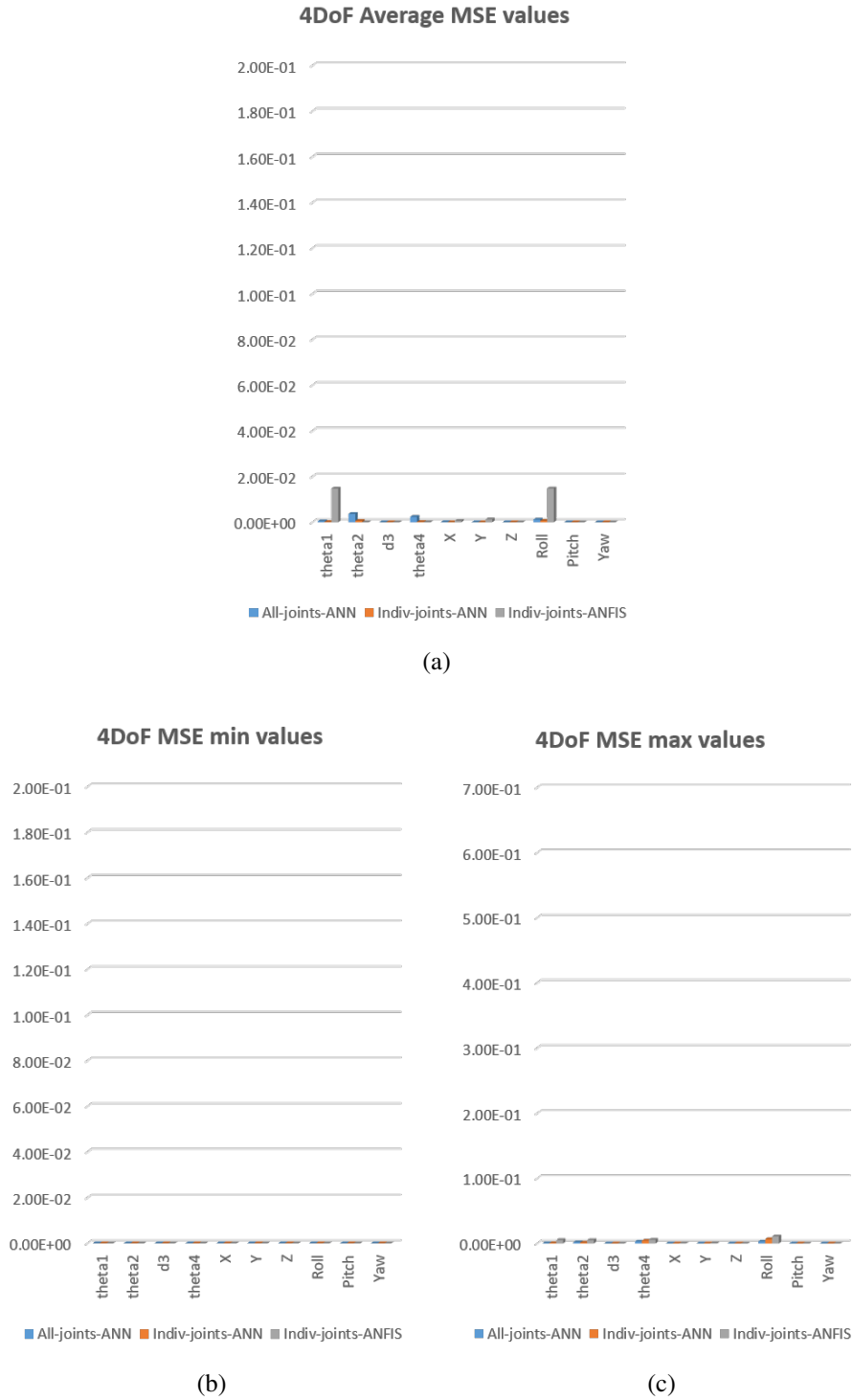


Figure 4.13: Visual depiction of the error values in Table 4.11 for the 4 DoF robot using the structured dataset.

	All-Joints-ANN	Individual-Joints-ANN	Individual-Joints-ANFIS
Aggregation (in the forward pass)	Induced local field	Induced local field	Product (layer 2) and Weighted average (layer 4)
Learning algorithm (in the backward pass)	Back-propagation	Back-propagation	Hybrid (Back-propagation + Least Square)
Activation or Membership function	Relu	Relu	Gaussian
Epochs	1000	1000	1000

Table 4.12: *Learning strategies for the experiment with the 5DoF robot reported in Table 4.13.*

	All-Joints-ANN (min/max)	Individual-Joints-ANN (min/max)	Individual-Joints-ANFIS (min/max)
Joint-MSE in $rad^2/m^2$			
$\theta_1$	<b>1.306e-4</b> (0.000e+0/2.099e-3)	4.690e-4 (0.000e+0/9.104e-3)	3.332e-3 (1.251e-9/4.765e-1)
$\theta_2$	1.033e-6 (2.639e-12/5.120e-5)	<b>2.030e-9</b> (2.079e-15/1.738e-8)	1.224e-6 (1.499e-12/5.756e-4)
$\theta_3$	1.054e-4 (3.451e-10/2.594e-3)	3.009e-5 (9.605e-12/6.294e-4)	<b>5.879e-7</b> (2.804e-5/2.893e-12)
$\theta_4$	1.381e-4 (0.000e+0/3.057e-3)	1.050e-4 (0.000e+0/1.340e-3)	<b>5.447e-7</b> (1.499e-12/9.371e-6)
$\theta_5$	1.294e-4 (4.168e-13/6.564e-3)	2.738e-4 (1.041e-8/9.043e-3)	<b>1.276e-6</b> (7.346e-11/2.169e-5)
Reconstructed Pose-MSE in $rad^2/m^2$			
$X$	<b>2.156e-5</b> (1.059e-10/4.663e-4)	7.303e-5 (8.659e-14/2.321e-3)	1.667e-4 (7.641e-11/9.743e-3)
$Y$	<b>2.011e-5</b> (3.306e-11/4.062e-4)	6.595e-5 (7.190e-11/1.250e-3)	5.704e-4 (2.289e-11/8.013e-2)
$Z$	5.470e-6 (5.073e-14/1.158e-4)	2.129e-6 (1.132e-13/6.206e-5)	<b>3.873e-8</b> (5.269e-14/1.423e-6)
$R_o$	<b>1.314e-4</b> (1.966e-10/3.845e-3)	1.707e-4 (3.944e-11/4.596e-3)	9.198e-4 (2.601e-10/1.799e-1)
$P_i$	<b>1.362e-4</b> (2.796e-11/5.378e-3)	2.787e-4 (2.524e-9/8.654e-3)	5.238e-4 (1.617e-12/3.880e-2)
$Y_a$	<b>1.032e-4</b> (1.208e-11/2.304e-3)	2.614e-4 (3.069e-10/5.809e-3)	1.847e-3 (5.777e-12/2.629e-1)

Table 4.13: *Results for 5 DoF robot using the structured dataset.*

	All-Joints-ANN	Individual-Joints-ANN	Individual-Joints-ANFIS
Aggregation (in the forward pass)	Induced local field	Induced local field	Product (layer 2) and Weighted average (layer 4)
Learning algorithm (in the backward pass)	Back-propagation	Back-propagation	Hybrid (Back-propagation + Least Square)
Activation or Membership function	Relu	Relu	Gaussian
Epochs	1000	1000	1000

Table 4.14: *Learning strategies for the experiment with the 6DoF robot reported in Table 4.15.*

	All-Joints-ANN (min/max)	Individual-Joints-ANN (min/max)	Individual-Joints-ANFIS (min/max)
Joint-MSE in $rad^2/m^2$			
$\theta_1$	6.474e-4 (0.000e+0/2.740e-1)	<b>3.908e-4</b> (0.000e+0/6.586e-4)	2.407e-3 (7.855e-10/1.142e-1)
$\theta_2$	2.295e-4 (5.370e-10/2.758e-3)	6.499e-4 (2.795e-9/1.627e-2)	<b>4.103e-5</b> (1.349e-11/5.591e-4)
$d_3$	3.160e-6 (6.350e-12/5.738e-5)	1.472e-6 (1.369e-13/1.097e-4)	<b>2.750e-7</b> (0.000e+0/5.108e-6)
$\theta_4$	3.290e-4 (1.602e-9/9.016e-3)	<b>1.424e-5</b> (4.543e-13/9.027e-3)	5.328e-4 (9.288e-10/5.512e-3)
$\theta_5$	2.698e-4 (3.244e-11/3.276e-3)	3.494e-4 (5.178e-11/3.827e-2)	<b>2.649e-5</b> (5.931e-10/3.832e-4)
$\theta_6$	<b>5.070e-4</b> (0.000e+0/1.624e-2)	5.466e-3 (0.000e+0/3.046e-2)	5.275e-4 (1.261e-9/1.163e-2)
Reconstructed Pose-MSE in $rad^2/m^2$			
X	9.991e-6 (1.466e-11/2.546e-3)	<b>6.480e-6</b> (1.071e-10/2.110e-4)	4.419e-5 (5.038e-11/3.662e-3)
Y	1.143e-5 (3.390e-12/4.488e-3)	<b>8.054e-6</b> (2.949e-14/3.338e-4)	4.031e-5 (6.604e-14/1.472e-3)
Z	3.947e-6 (2.855e-13/6.294e-5)	1.188e-5 (5.744e-13/2.791e-4)	<b>4.796e-7</b> (1.168e-14/6.569e-6)
$Ro$	<b>8.679e-4</b> (2.687e-11/3.239e-1)	4.904e-3 (1.939e-10/4.061e-2)	1.058e-3 (2.372e-10/2.435e-2)
$Pi$	<b>4.179e-4</b> (4.210e-9/8.106e-2)	4.875e-4 (2.139e-12/7.424e-3)	6.337e-4 (6.883e-10/3.257e-2)
$Ya$	<b>2.207e-4</b> (5.681e-9/4.914e-3)	2.666e-4 (1.003e-10/4.213e-3)	1.202e-3 (9.736e-11/6.775e-2)

Table 4.15: *Results for 6 DoF robot using the structured dataset.*

	All-Joints-ANN	Individual-Joints-ANN	Individual-Joints-ANFIS
Aggregation (in the forward pass)	Induced local field	Induced local field	Product (layer 2) and Weighted average (layer 4)
Learning algorithm (in the backward pass)	Back-propagation	Back-propagation	Hybrid (Back-propagation + Least Square)
Activation or Membership function	Relu	Relu	Gaussian
Epochs	1000	1000	1000

Table 4.16: Learning strategies for the experiment with the 7DoF robot reported in Table 4.17.

	All-Joints-ANN (min/max)	Individual-Joints-ANN (min/max)	Individual-Joints-ANFIS (min/max)
Joint-MSE in $rad^2/m^2$			
$\theta_1$	<b>1.928e-4</b> (0.000e+0/5.390e-3)	2.053e-4 (0.000e+0/7.417e-3)	9.160e-3 (3.479e-8/7.113e-1)
$\theta_2$	1.188e-4 (2.944e-11/1.584e-3)	2.637e-5 (5.907e-11/3.967e-4)	<b>5.256e-7</b> (5.997e-12/7.537e-6)
$\theta_3$	6.782e-3 (1.600e-7/4.759e-2)	5.430e-3 (2.091e-9/2.077e-2)	<b>3.544e-3</b> (3.866e-9/2.190e-2)
$\theta_4$	8.136e-6 (2.840e-10/1.668e-4)	4.480e-8 (2.359e-14/5.524e-7)	<b>3.597e-9</b> (5.397e-11/7.346e-6)
$\theta_5$	5.676e-6 (9.324e-12/7.803e-5)	1.061e-7 (2.359e-14/3.015e-6)	<b>2.698e-8</b> (3.244e-12/3.276e-5)
$\theta_6$	7.440e-3 (2.384e-3/1.306e-2)	7.586e-3 (6.557e-3/8.690e-3)	<b>1.855e-6</b> (4.248e-13/1.515e-5)
$\theta_7$	1.553e-2 (0.000e+0/3.046e-2)	7.346e-3 (3.508e-3/1.282e-2)	<b>3.545e-3</b> (1.034e-9/1.842e-2)
Reconstructed Pose-MSE in $rad^2/m^2$			
$X$	2.258e-4 (6.342e-10/1.726e-3)	<b>2.154e-4</b> (1.821e-9/1.221e-3)	2.787e-4 (1.567e-10/2.750e-2)
$Y$	<b>2.198e-4</b> (2.653e-10/1.301e-3)	2.384e-4 (9.914e-11/1.460e-3)	1.156e-3 (3.590e-11/9.004e-2)
$Z$	6.123e-5 (5.872e-11/5.402e-4)	5.672e-5 (6.077e-11/2.900e-4)	<b>2.771e-7</b> (1.110e-12/5.846e-6)
$R_o$	1.031e-2 (4.560e-8/5.077e-2)	<b>1.841e-3</b> (3.018e-9/1.428e-2)	2.986e-3 (2.035e-9/4.633e-1)
$P_i$	7.950e-3 (7.702e-5/2.489e-2)	8.015e-3 (3.222e-3/2.132e-2)	<b>5.925e-3</b> (1.077e-9/5.236e-1)
$Y_a$	2.683e-4 (4.003e-10/2.025e-3)	<b>2.005e-4</b> (5.618e-9/1.356e-3)	8.970e-4 (7.625e-12/1.631e-1)

Table 4.17: Results for 7 DoF robot using the structured dataset.

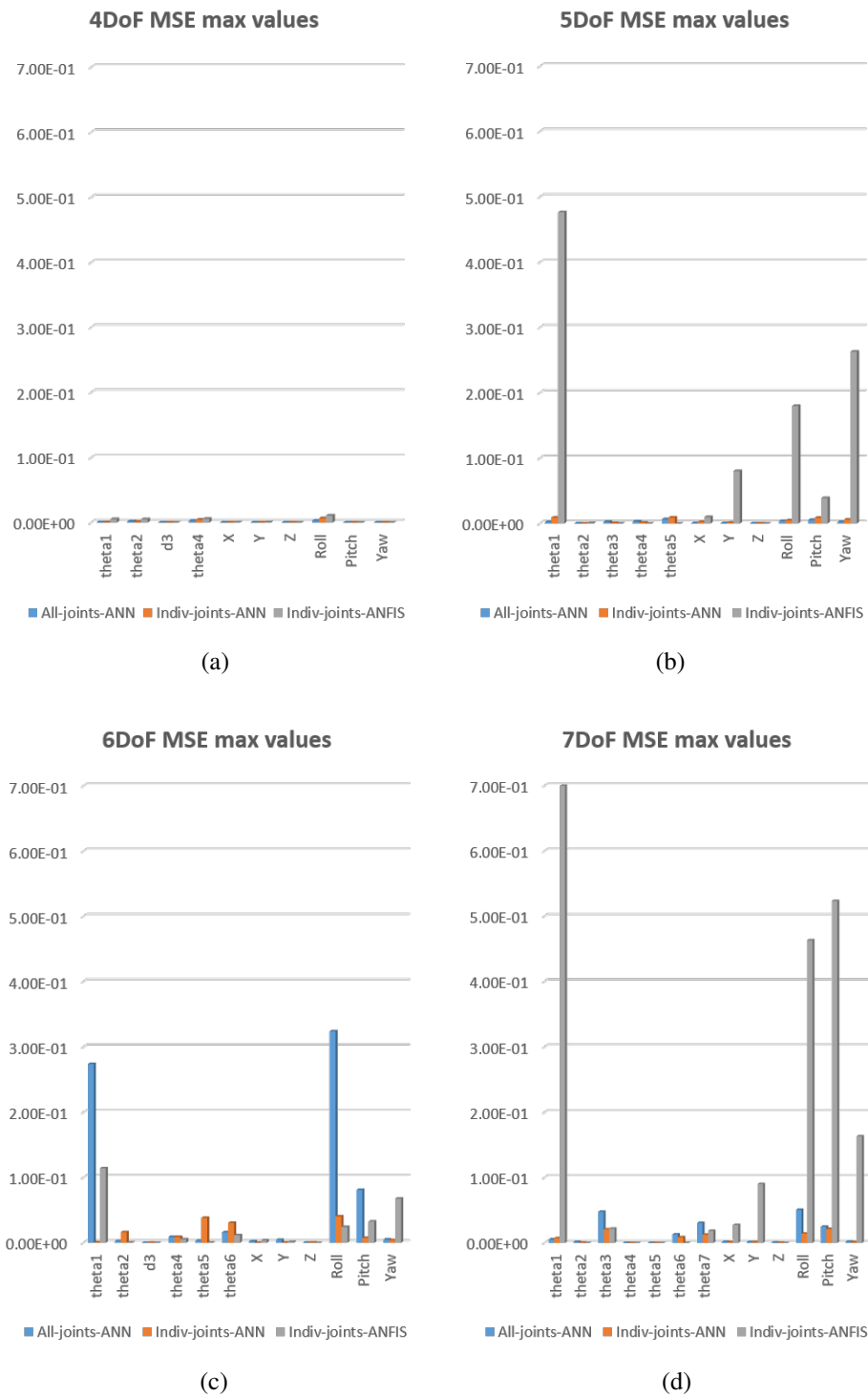


Figure 4.14: Graphical summary of the maximum error values found in Tables 4.11, 4.13, 4.15 and 4.17 based on the structured datasets for each of the four robots and using all three neural network architectures in the case studies.

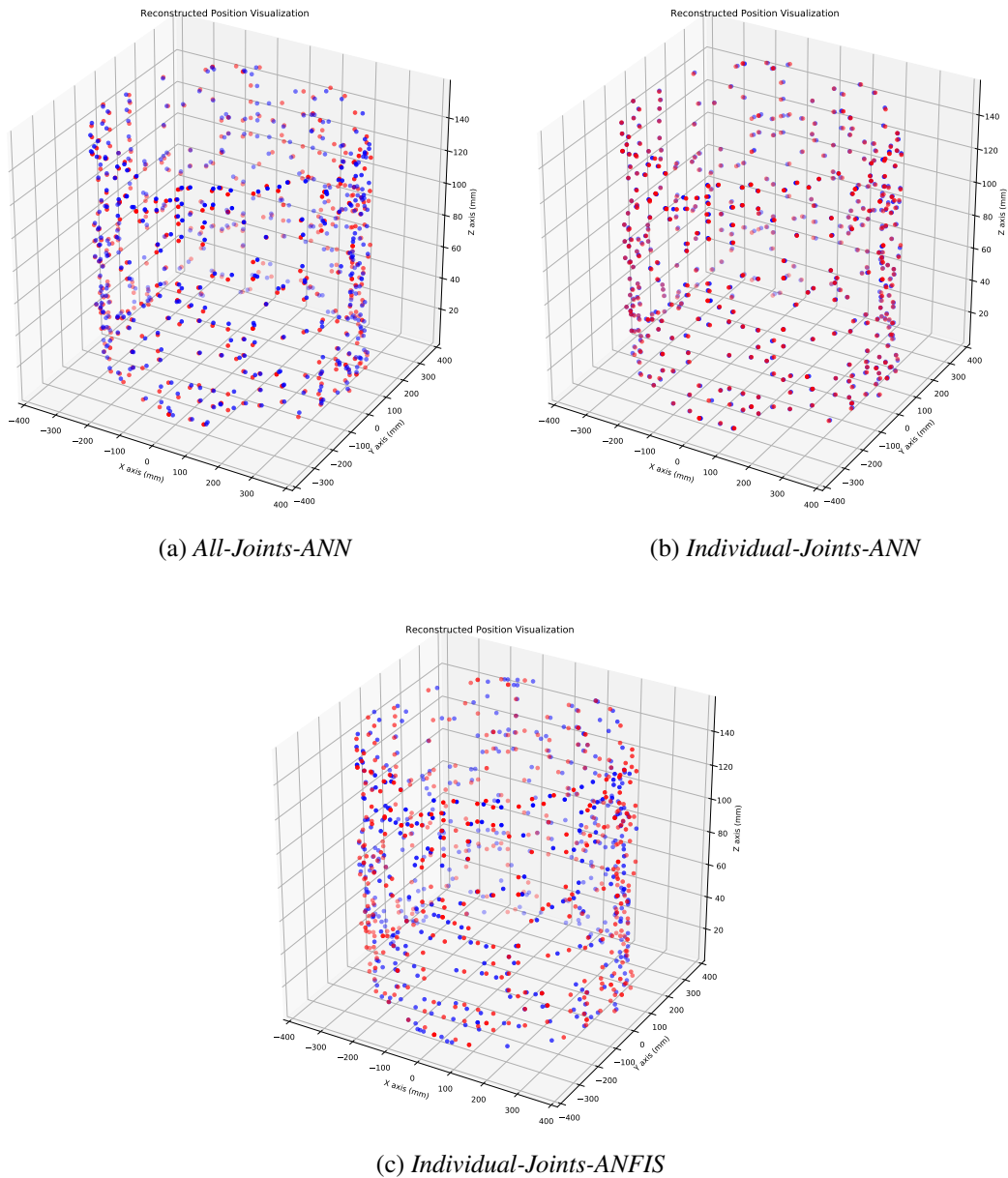


Figure 4.15: A visualization of the error in positioning of the end-effector in the workspace of the 4DoF robot related to the results summarized in Table 4.11 for the structured dataset. The points in blue color represent the desired positions and those in red color represent the predicted positions. While most predictions were close to their desired positions, not all of them fall within the accepted mm ranges required for several critical IK applications.

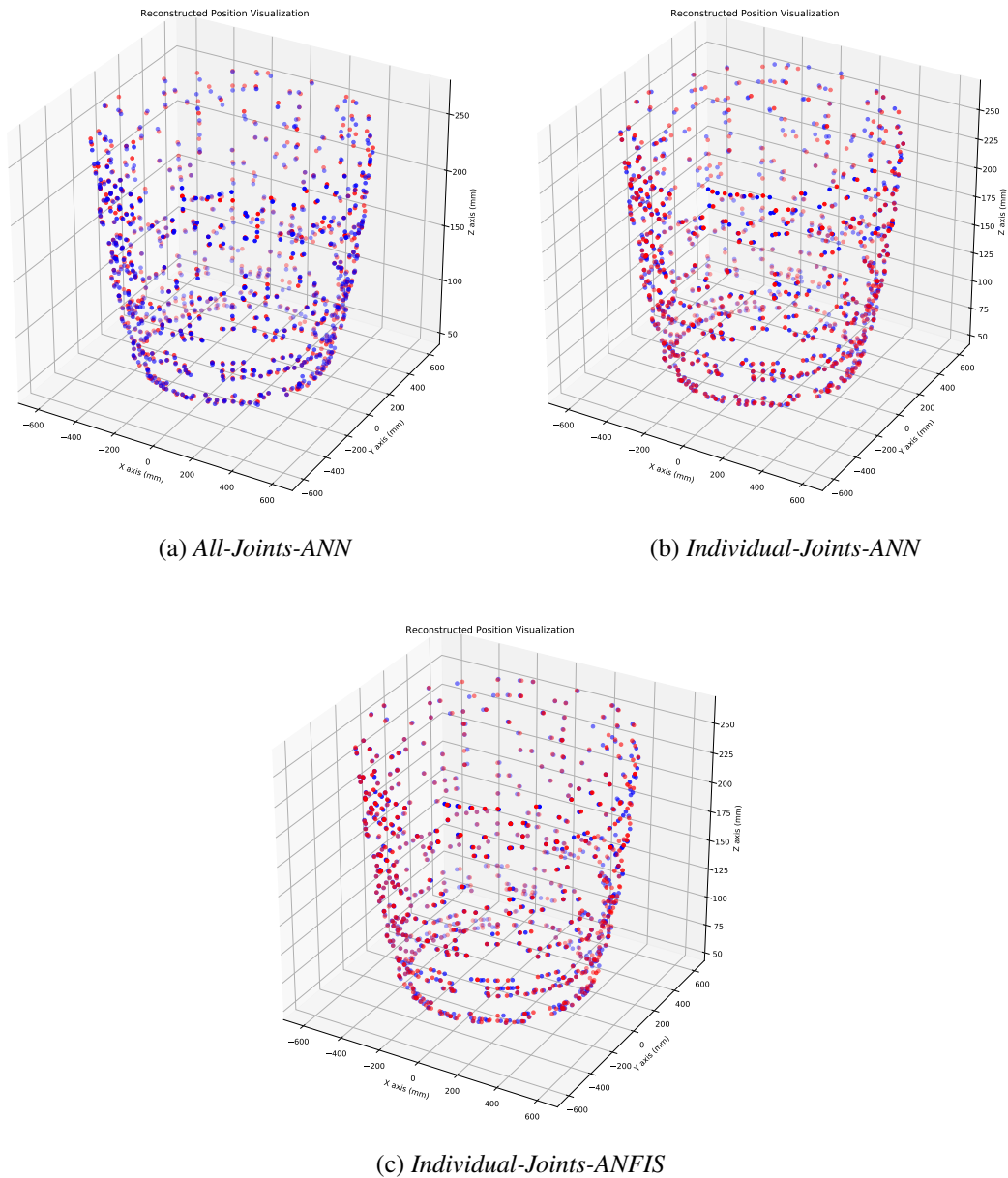


Figure 4.16: A visualization of the error in positioning of the end-effector in the workspace of the 5DoF robot related to the results summarized in Table 4.13 for the structured dataset. The points in blue color represent the desired positions and those in red color represent the predicted positions. Once again, while most predictions were close to their desired positions, not all of them fall within the accepted mm ranges required for several critical IK applications.

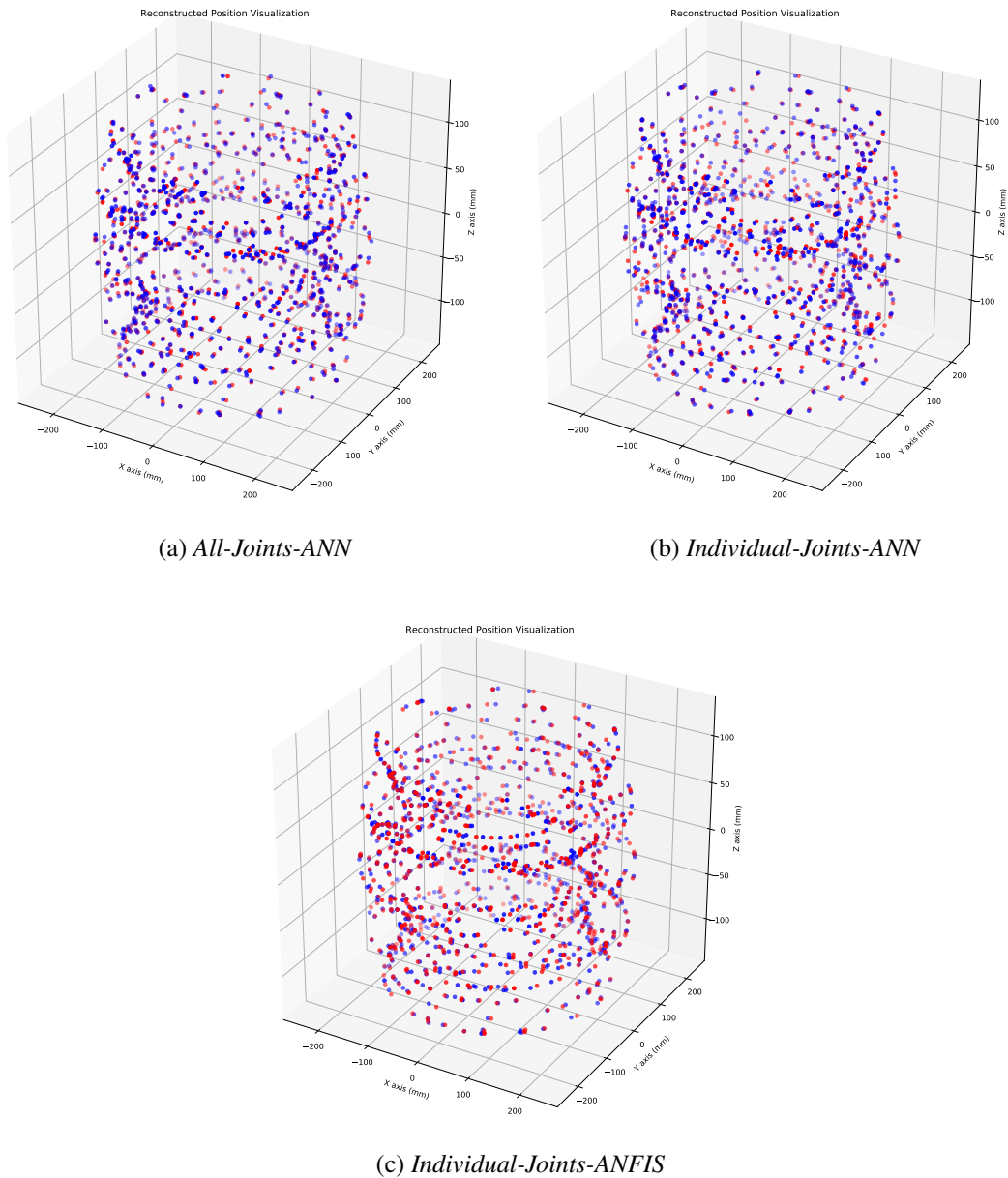


Figure 4.17: A visualization of the error in positioning of the end-effector in the workspace of the 6DoF robot related to the results summarized in Table 4.15 for the structured dataset. The points in blue color represent the desired positions and those in red color represent the predicted positions. Once again, while most predictions were close to their desired positions, not all of them fall within the accepted mm ranges required for several critical IK applications.



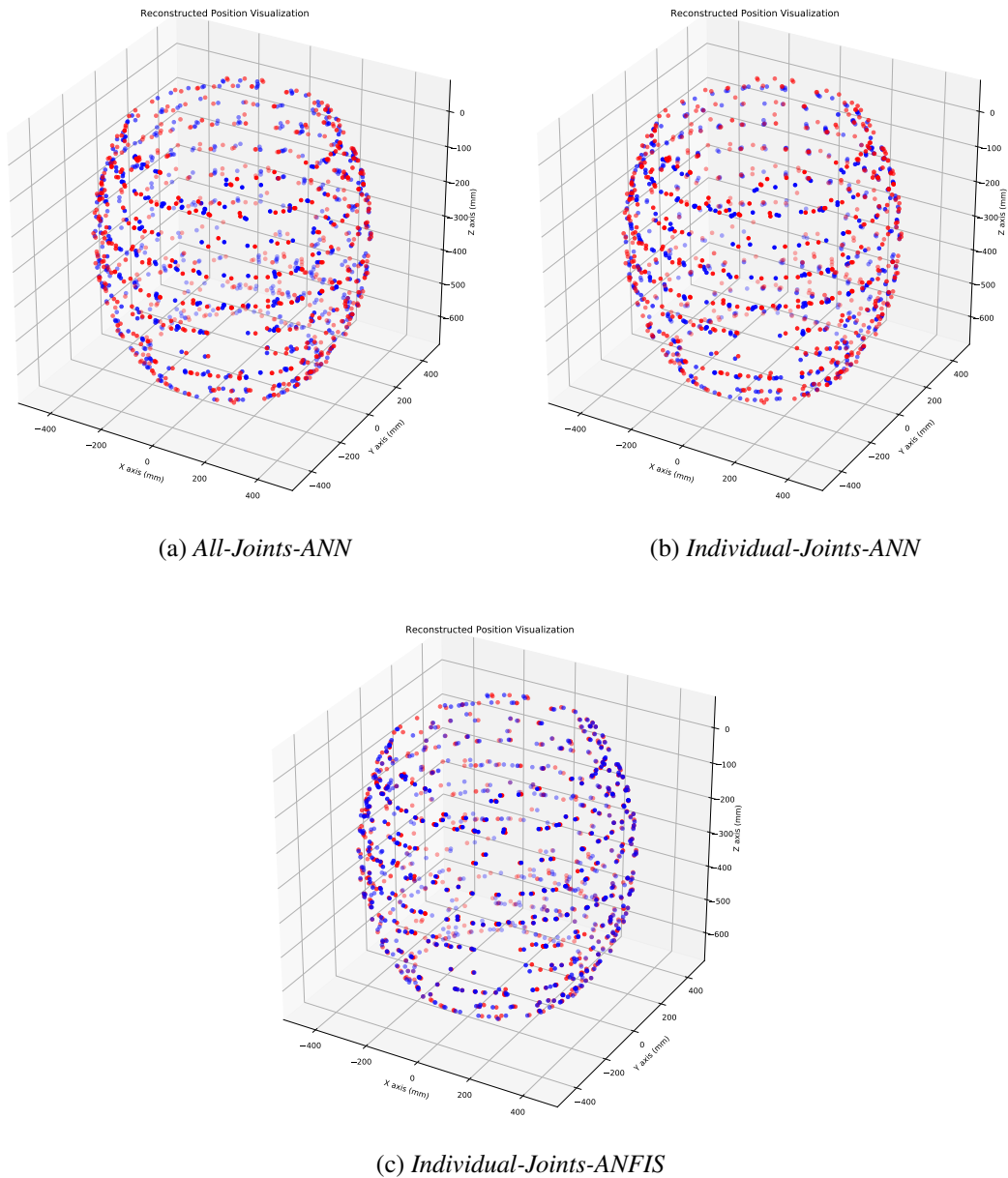


Figure 4.18: A visualization of the error in positioning of the end-effector in the workspace of the 7DoF robot related to the results summarized in Table 4.17 for the structured dataset. The points in blue color represent the desired positions and those in red color represent the predicted positions. Once again, while most predictions were close to their desired positions, not all of them fall within the accepted mm ranges required for several critical IK applications.

### Random Datasets

The results obtained with the random datasets, led us to restrict the vast space on which the NNs had to learn the IK. That is, while still keeping the poses random, we limited the

workspaces for the next datasets to approximately two quadrants (i.e. joint 1 =  $[0 - 180]$ ) for all robots, at the same time that the other joints were roughly kept the same as in Table 4.4 – with the exception of some slightly larger ranges for the 4-DoF robot. As before, Tables 4.19, 4.21, 4.23 and 4.25 summarize the results for goals #1 (column 1 vs. 2) and #2 (column 2 vs. 3), using the random datasets for robots with 4, 5, 6, and 7 DoF, respectively. Here, the results improved, but only after we increased the training set from a few thousands for the structured datasets to 15,000 in the random datasets. Also, since the search space in which the NNs needed to learn was made smaller, it was expected that the error would also decrease.

	All-Joints-ANN	Individual-Joints-ANN	Individual-Joints-ANFIS
Aggregation (in the forward pass)	Induced local field	Induced local field	Product (layer 2) and Weighted average (layer 4)
Learning algorithm (in the backward pass)	Back-propagation	Back-propagation	Hybrid (Back-propagation + Least Square)
Activation or Membership function	Relu	Relu	Gaussian
Epochs	1000	1000	1000

Table 4.18: *Learning strategies for the experiment with the 4DoF robot reported in Table 4.19.*

	All-Joints-ANN (min/max)	Individual-Joints-ANN (min/max)	Individual-Joints-ANFIS (min/max)
Joint-MSE in $rad^2/m^2$			
$\theta_1$	<b>6.742e-6</b> (7.719e-12/3.014e-4)	1.812e-5 (1.401e-11/4.044e-4)	1.560e-4 (7.164e-10/5.206e-3)
$\theta_2$	3.651e-5 (8.648e-13/1.784e-3)	<b>2.854e-5</b> (5.401e-14/1.289e-3)	3.020e-4 (6.264e-10/5.108e-3)
$d_3$	1.414e-8 (5.166e-14/1.131e-6)	8.706e-11 (2.140e-17/2.154e-8)	<b>7.507e-15</b> (3.825e-22/2.545e-14)
$\theta_4$	<b>2.175e-5</b> (6.889e-13/2.881e-3)	3.476e-5 (3.900e-13/4.256e-3)	3.241e-4 (1.251e-10/5.582e-3)
Reconstructed Pose-MSE in $rad^2/m^2$			
$X$	<b>1.323e-7</b> (1.822e-13/2.125e-6)	8.042e-7 (3.863e-13/2.666e-5)	7.216e-6 (8.741e-12/2.522e-4)
$Y$	<b>1.470e-7</b> (6.376e-14/6.275e-6)	1.117e-6 (6.875e-13/3.747e-5)	9.965e-6 (8.133e-16/4.766e-4)
$Z$	1.414e-8 (5.166e-14/1.131e-6)	8.706e-11 (2.140e-17/2.154e-8)	<b>7.507e-15</b> (3.825e-22/2.545e-14)
$Ro$	<b>1.103e-5</b> (4.184e-11/3.111e-3)	5.339e-5 (3.190e-13/6.525e-3)	6.695e-4 (1.709e-12/1.076e-2)
$Pi$	<b>1.284e-37</b> (3.620e-49/8.565e-36)	1.731e-37 (9.664e-49/4.950e-36)	2.049e-36 (4.829e-45/3.809e-35)
$Ya$	<b>1.273e-37</b> (7.178e-45/9.278e-36)	1.844e-37 (4.175e-46/5.701e-36)	2.147e-36 (9.253e-44/6.470e-35)

Table 4.19: Results for 4 DoF robot using a random dataset.

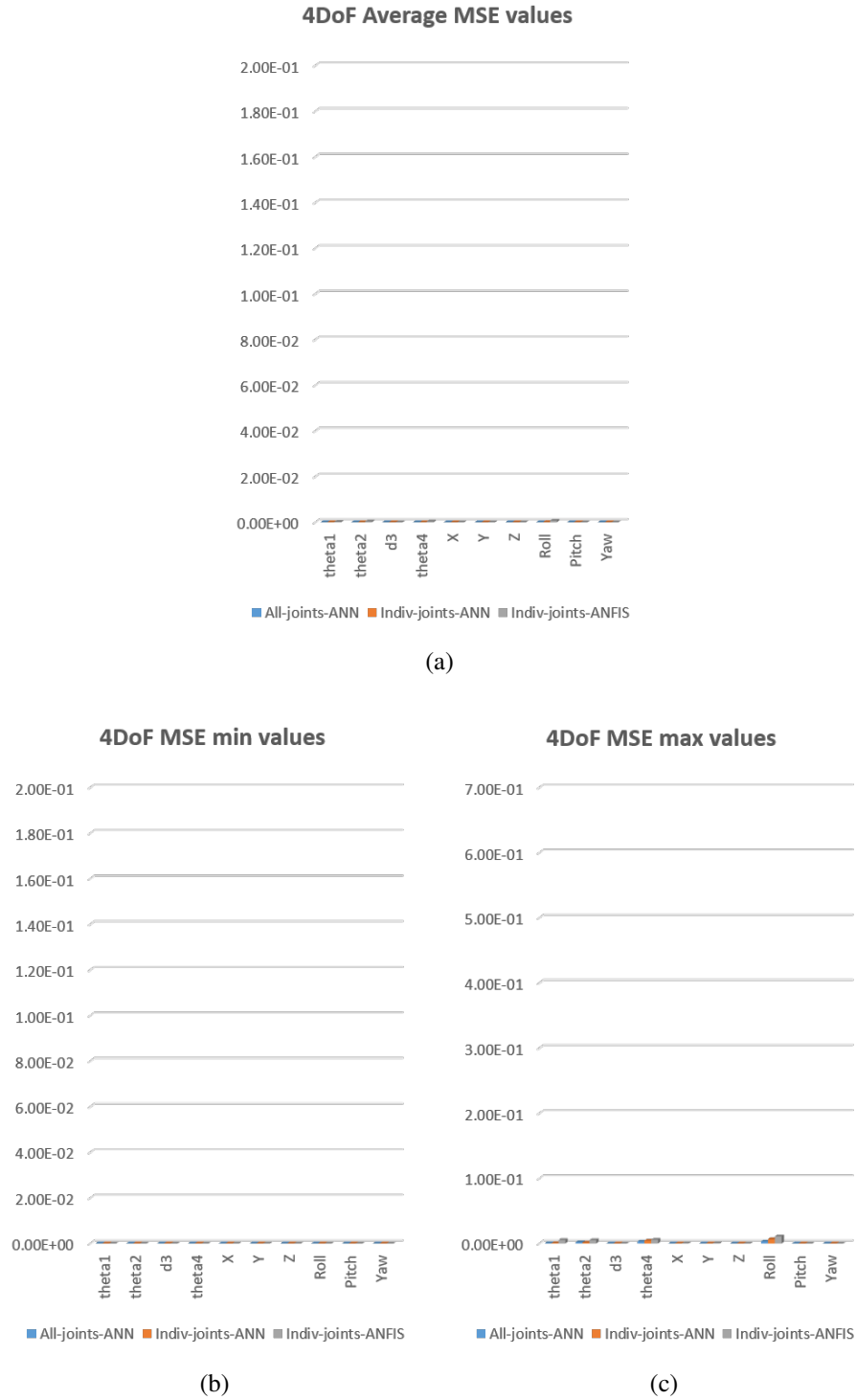


Figure 4.19: Visual depiction of the error values in Table 4.19 for the 4 DoF robot using the random dataset.

	All-Joints-ANN	Individual-Joints-ANN	Individual-Joints-ANFIS
Aggregation (in the forward pass)	Induced local field	Induced local field	Product (layer 2) and Weighted average (layer 4)
Learning algorithm (in the backward pass)	Back-propagation	Back-propagation	Hybrid (Back-propagation + Least Square)
Activation or Membership function	Relu	Relu	Gaussian
Epochs	1000	1000	1000

Table 4.20: Learning strategies for the experiment with the 5DoF robot reported in Table 4.21.

	All-Joints-ANN (min/max)	Individual-Joints-ANN (min/max)	Individual-Joints-ANFIS (min/max)
Joint-MSE in $rad^2/m^2$			
$\theta_1$	<b>8.060e-3</b> (1.759e-10/3.208e-2)	8.070e-3 (3.059e-9/3.018e-2)	8.105e-3 (4.610e-9/3.201e-2)
$\theta_2$	<b>8.079e-3</b> (4.792e-9/3.003e-2)	8.093e-3 (1.264e-9/2.780e-2)	8.090e-3 (1.577e-8/3.128e-2)
$\theta_3$	5.812e-6 (1.860e-11/1.118e-3)	9.677e-7 (1.593e-14/2.510e-4)	<b>7.310e-9</b> (8.278e-15/8.100e-7)
$\theta_4$	6.683e-6 (3.325e-11/7.886e-4)	2.017e-6 (4.370e-14/3.026e-4)	<b>3.005e-8</b> (2.456e-15/1.616e-6)
$\theta_5$	1.965e-3 (3.309e-11/1.295e-2)	<b>1.965e-3</b> (5.219e-13/1.202e-2)	1.966e-3 (2.023e-10/1.192e-2)
Reconstructed Pose-MSE in $rad^2/m^2$			
$X$	4.818e-7 (1.645e-13/1.058e-4)	7.883e-7 (1.780e-15/2.043e-4)	<b>4.413e-7</b> (1.531e-13/2.137e-5)
$Y$	<b>7.239e-7</b> (2.797e-13/9.721e-5)	7.791e-7 (1.461e-13/3.675e-5)	9.402e-7 (3.454e-14/5.525e-5)
$Z$	4.674e-8 (2.014e-15/3.643e-6)	6.732e-9 (9.137e-18/1.749e-7)	<b>1.039e-10</b> (1.207e-17/8.057e-9)
$Ro$	6.816e-6 (6.972e-16/6.088e-4)	3.049e-6 (2.071e-12/8.866e-5)	<b>1.148e-6</b> (3.126e-14/2.746e-5)
$Pi$	7.397e-6 (5.144e-12/7.715e-4)	1.015e-5 (2.903e-15/5.765e-4)	<b>6.676e-6</b> (3.848e-13/1.916e-4)
$Ya$	<b>1.463e-5</b> (2.349e-12/3.842e-4)	3.328e-5 (6.891e-13/3.349e-3)	2.903e-5 (1.530e-11/1.123e-3)

Table 4.21: Results for 5 DoF robot using a random dataset.

	All-Joints-ANN	Individual-Joints-ANN	Individual-Joints-ANFIS
Aggregation (in the forward pass)	Induced local field	Induced local field	Product (layer 2) and Weighted average (layer 4)
Learning algorithm (in the backward pass)	Back-propagation	Back-propagation	Hybrid (Back-propagation + Least Square)
Activation or Membership function	Relu	Relu	Gaussian
Epochs	1000	1000	1000

Table 4.22: Learning strategies for the experiment with the 6DoF robot reported in Table 4.23.

	All-Joints-ANN (min/max)	Individual-Joints-ANN (min/max)	Individual-Joints-ANFIS (min/max)
Joint-MSE in $rad^2/m^2$			
$\theta_1$	<b>3.161e-5</b> (1.803e-11/8.168e-4)	5.147e-5 (2.763e-14/1.498e-3)	4.271e-5 (2.907e-11/1.567e-3)
$\theta_2$	<b>1.479e-4</b> (5.265e-13/9.901e-3)	2.368e-4 (8.412e-12/9.688e-3)	2.653e-4 (6.411e-10/1.017e-2)
$d_3$	5.105e-7 (1.657e-14/1.793e-5)	<b>3.506e-7</b> (1.228e-13/1.296e-5)	5.194e-7 (1.474e-13/1.704e-5)
$\theta_4$	<b>5.053e-5</b> (1.685e-12/1.359e-3)	7.562e-5 (2.799e-11/3.370e-3)	1.639e-4 (9.757e-12/2.632e-3)
$\theta_5$	<b>3.309e-2</b> (1.126e-7/1.059e-1)	3.312e-2 (6.184e-7/1.029e-1)	3.316e-2 (1.153e-7/1.024e-1)
$\theta_6$	3.291e-2 (1.065e-8/1.017e-1)	3.278e-2 (1.514e-4/9.708e-2)	<b>3.272e-2</b> (1.470e-6/1.014e-1)
Reconstructed Pose-MSE in $rad^2/m^2$			
X	<b>2.120e-7</b> (2.468e-13/4.434e-6)	4.873e-7 (4.377e-14/1.153e-5)	4.533e-7 (2.947e-12/1.167e-5)
Y	<b>2.194e-7</b> (2.723e-13/5.328e-6)	3.753e-7 (2.493e-12/8.731e-6)	4.059e-7 (2.852e-12/1.563e-5)
Z	<b>1.324e-7</b> (3.833e-14/3.620e-6)	6.091e-7 (8.738e-13/8.866e-6)	1.012e-6 (2.587e-12/2.219e-5)
$Ro$	<b>1.470e-5</b> (4.037e-12/3.684e-4)	8.152e-5 (3.904e-11/1.174e-3)	1.408e-4 (2.980e-10/1.732e-3)
$Pi$	<b>1.590e-5</b> (7.366e-12/6.308e-4)	5.060e-5 (1.906e-11/1.205e-3)	1.010e-4 (1.468e-11/1.484e-3)
$Ya$	<b>1.579e-5</b> (7.990e-12/4.203e-4)	4.943e-5 (1.393e-12/1.555e-3)	8.242e-5 (1.239e-11/1.951e-3)

Table 4.23: Results for 6 DoF robot using a random dataset.

	All-Joints-ANN	Individual-Joints-ANN	Individual-Joints-ANFIS
Aggregation (in the forward pass)	Induced local field	Induced local field	Product (layer 2) and Weighted average (layer 4)
Learning algorithm (in the backward pass)	Back-propagation	Back-propagation	Hybrid (Back-propagation + Least Square)
Activation or Membership function	Relu	Relu	Gaussian
Epochs	1000	1000	1000

Table 4.24: Learning strategies for the experiment with the 7DoF robot reported in Table 4.17.

	All-Joints-ANN (min/max)	Individual-Joints-ANN (min/max)	Individual-Joints-ANFIS (min/max)
Joint-MSE in $rad^2/m^2$			
$\theta_1$	<b>3.239e-4</b> (4.544e-10/2.303e-2)	3.721e-4 (1.381e-9/2.438e-2)	3.643e-4 (5.613e-12/2.432e-2)
$\theta_2$	<b>4.594e-5</b> (1.664e-10/1.076e-3)	9.508e-5 (1.109e-9/1.771e-3)	5.808e-5 (1.600e-11/1.387e-3)
$\theta_3$	<b>6.027e-4</b> (1.157e-9/1.907e-2)	6.919e-4 (7.019e-13/2.211e-2)	6.909e-4 (5.479e-9/1.336e-2)
$\theta_4$	2.041e-5 (2.000e-12/2.538e-3)	1.662e-5 (7.667e-11/2.380e-4)	<b>8.727e-6</b> (4.304e-11/1.839e-4)
$\theta_5$	4.514e-3 (2.400e-12/2.605e-2)	<b>4.444e-3</b> (2.722e-8/2.536e-2)	4.491e-3 (6.254e-11/2.690e-2)
$\theta_6$	3.870e-3 (2.069e-9/2.702e-2)	<b>3.807e-3</b> (1.178e-7/2.428e-2)	3.938e-3 (1.528e-8/2.731e-2)
$\theta_7$	3.870e-3 (2.069e-9/2.702e-2)	<b>3.807e-3</b> (1.178e-7/2.428e-2)	3.938e-3 (1.528e-8/2.731e-2)
Reconstructed Pose-MSE in $rad^2/m^2$			
X	<b>1.440e-6</b> (5.553e-13/9.361e-5)	4.927e-6 (2.344e-11/6.523e-5)	2.235e-6 (2.408e-12/5.250e-5)
Y	<b>1.182e-6</b> (4.018e-12/6.045e-5)	3.615e-6 (1.194e-11/3.923e-5)	2.458e-6 (2.437e-12/6.889e-5)
Z	<b>5.610e-7</b> (4.001e-14/1.067e-5)	5.517e-6 (2.236e-12/9.786e-5)	1.156e-6 (2.482e-15/4.351e-5)
$R_o$	<b>6.522e-5</b> (6.228e-13/8.757e-3)	3.410e-4 (1.476e-9/9.604e-3)	1.170e-4 (4.332e-10/5.419e-3)
$P_i$	<b>2.370e-5</b> (2.724e-11/9.293e-4)	1.313e-4 (3.078e-12/1.574e-3)	3.333e-5 (1.424e-11/9.293e-4)
$Y_a$	<b>1.292e-5</b> (1.197e-11/7.342e-4)	3.075e-5 (7.408e-11/4.651e-4)	2.290e-5 (5.497e-12/4.218e-4)

Table 4.25: Results for 7 DoF robot using a random dataset.

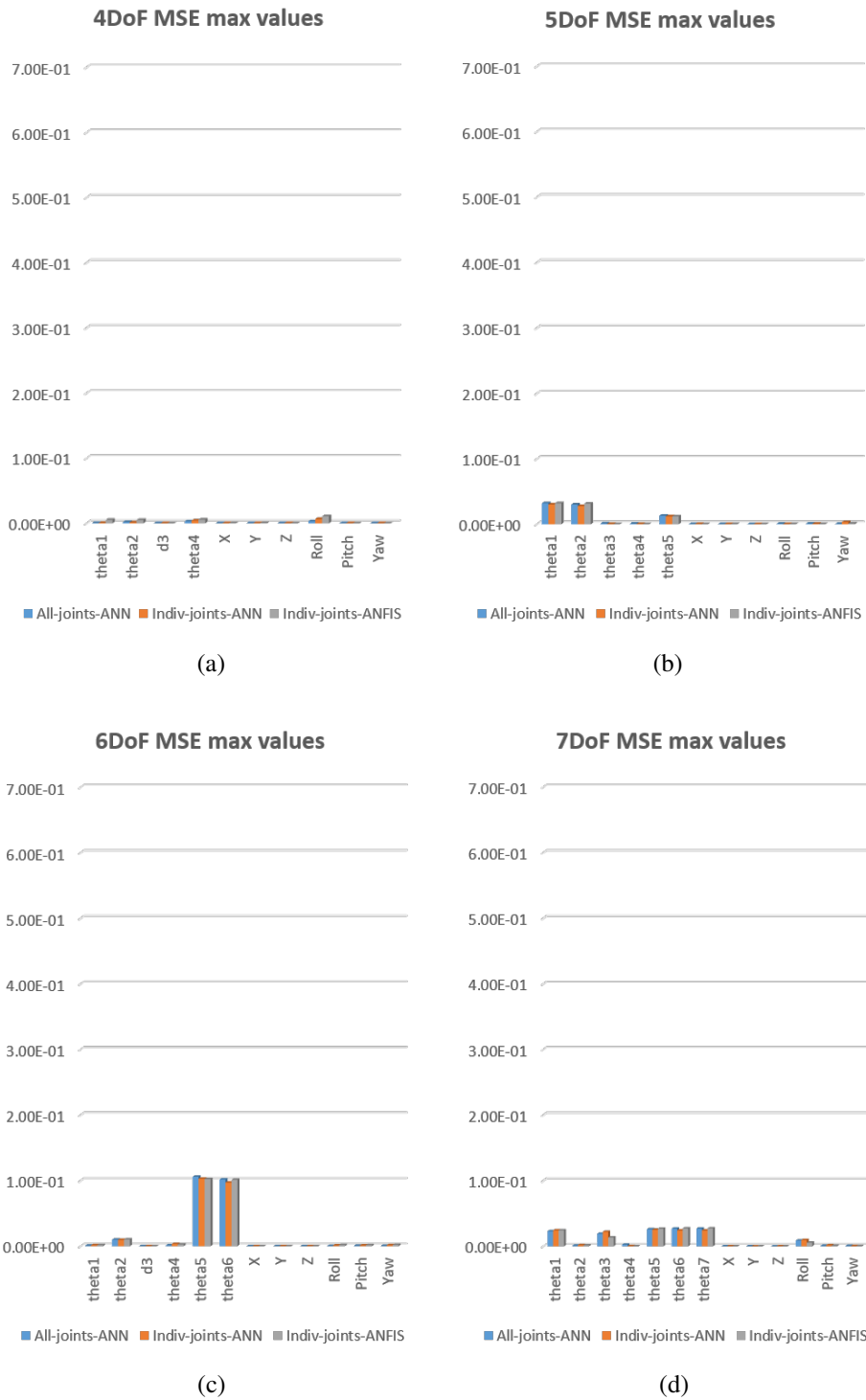


Figure 4.20: Graphical summary of the maximum error values found in Tables 4.19, 4.21, 4.23 and 4.25 based on the random datasets for each of the four robots and using all three neural network architectures in the case studies.



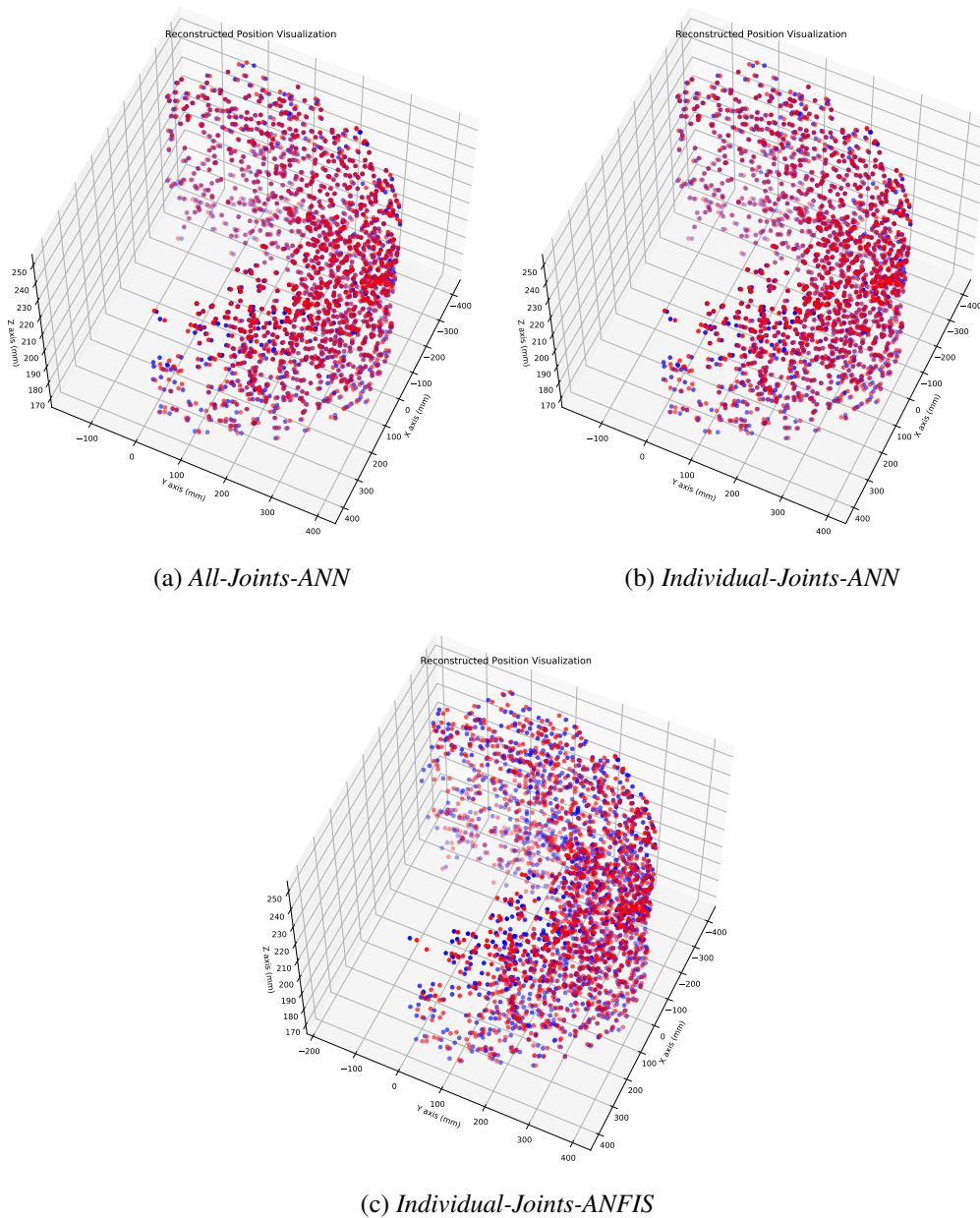


Figure 4.21: A visualization of the error in positioning of the end-effector in the workspace of the 4DoF robot related to the results summarized in Table 4.19 for the random dataset. The points in blue color represent the desired positions and those in red color represent the predicted positions. Here, the results were better than for the structured dataset, but still not all of them were within the accepted mm ranges required for several critical IK applications.

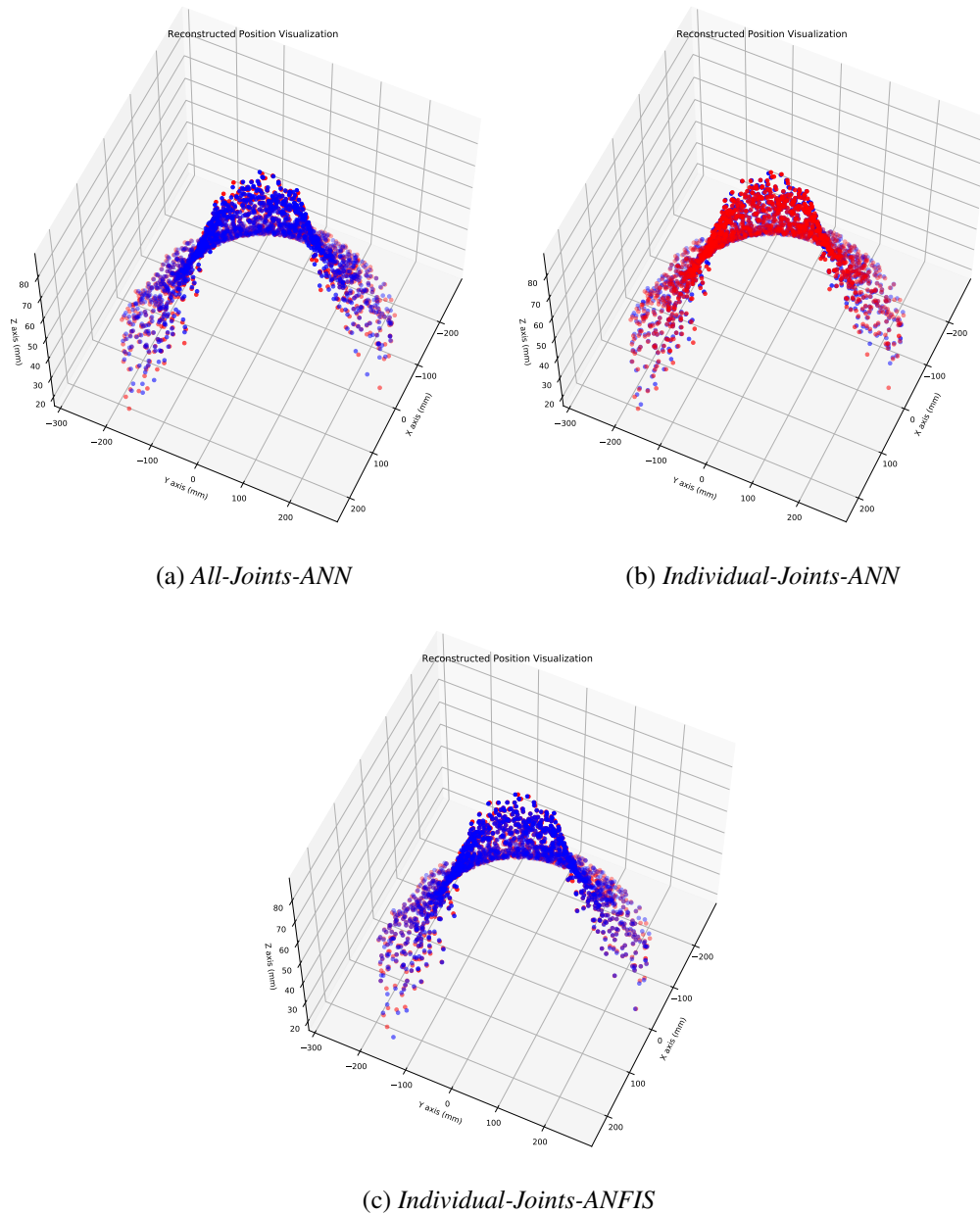


Figure 4.22: A visualization of the error in positioning of the end-effector in the workspace of the 5DoF robot related to the results summarized in Table 4.21 for the random dataset. The points in blue color represent the desired positions and those in red color represent the predicted positions. Once again, the results presented here were better than for the structured dataset, but still not all of them were within the accepted mm ranges required for several critical IK applications.

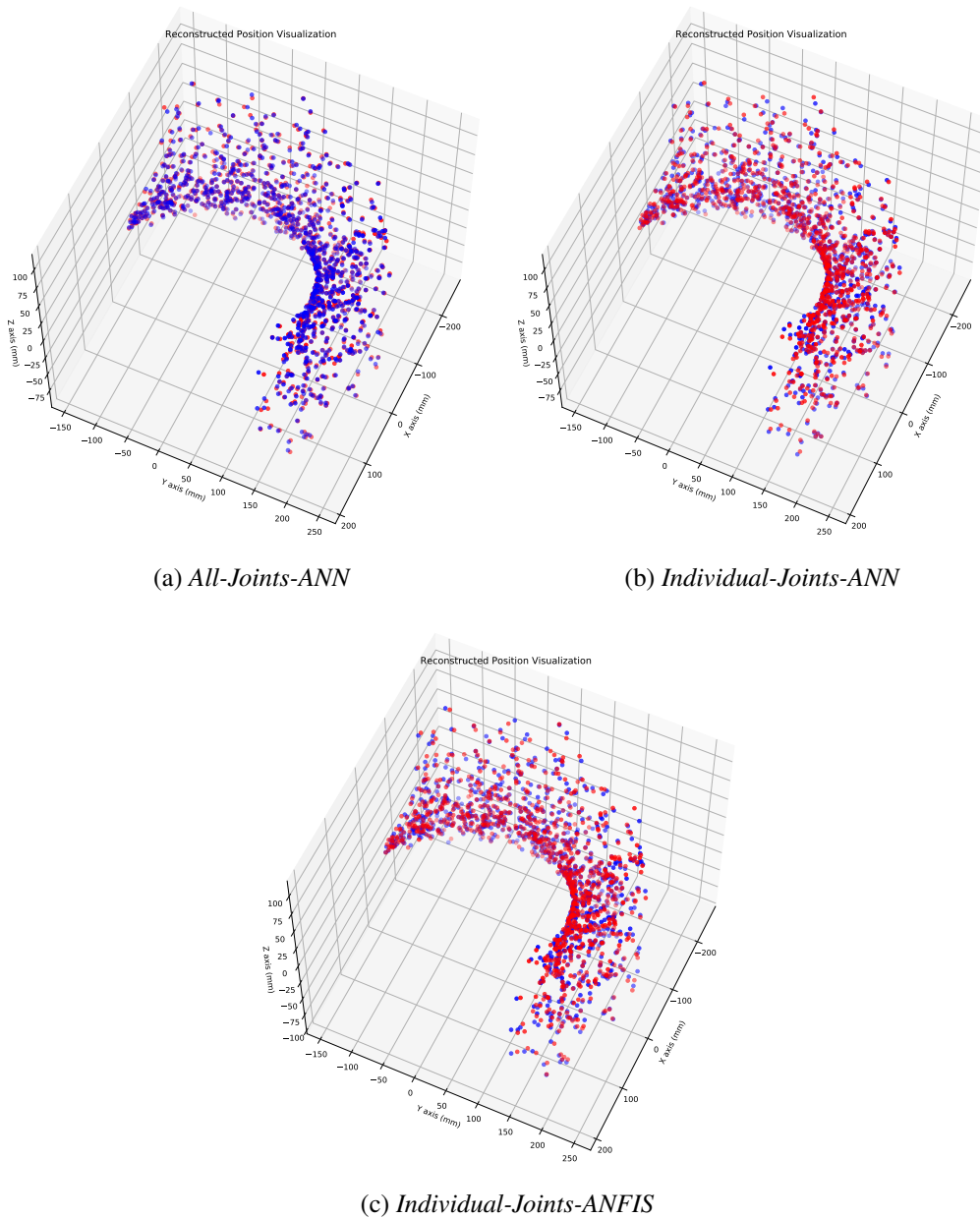


Figure 4.23: A visualization of the error in positioning of the end-effector in the workspace of the 6DoF robot related to the results summarized in Table 4.23 for the random dataset. The points in blue color represent the desired positions and those in red color represent the predicted positions. Once again, the results presented here were better than for the structured dataset, but still not all of them were within the accepted mm ranges required for several critical IK applications.

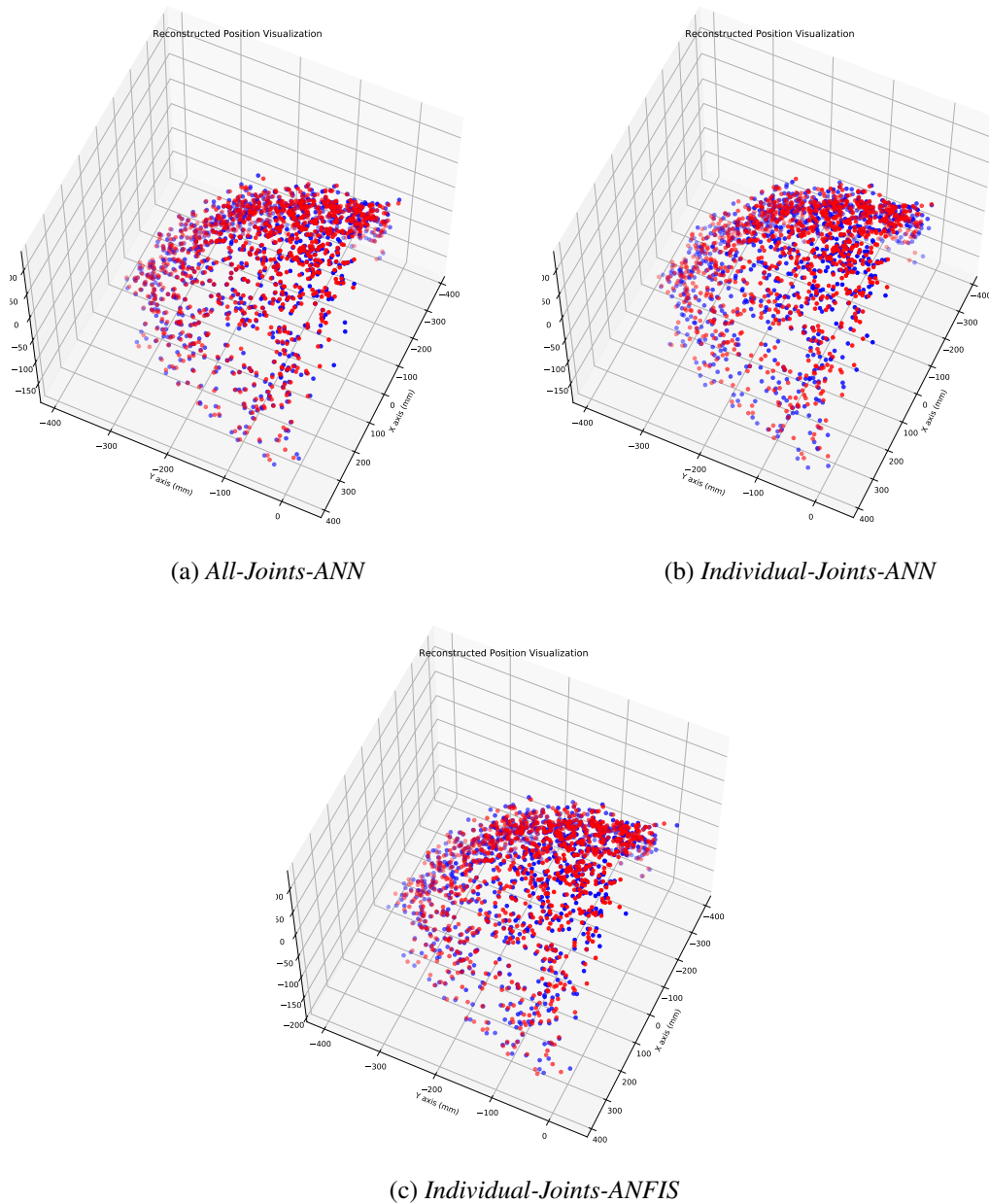


Figure 4.24: A visualization of the error in positioning of the end-effector in the workspace of the 7DoF robot related to the results summarized in Table 4.25 for the random dataset. The points in blue color represent the desired positions and those in red color represent the predicted positions. Once again, the results presented here were better than for the structured dataset, but still not all of them were within the accepted mm ranges required for several critical IK applications.

### 4.1.5 Observations on the Experiments

From the visualizations of the workspaces (see Figures 4.15, 4.16, 4.17, 4.18 for the structured datasets and Figures 4.21, 4.22, 4.23, 4.24 for the random datasets) where the

desired and predicted end-effector positions are plotted together, we can develop a sense on the prediction errors. This analysis is helpful in realizing how close the predicted positions fall with respect to their desired positions in the robot workspace. Nevertheless, as we will notice in Section 4.2, none of the investigated data-driven approaches performed in a comparable way to the numerical IK method. Most of the data-driven predictions in the task-free scenarios are not able to achieve sub-millimeters accuracies for positions and sub-degrees accuracies for orientations. In Figure 4.25, we highlight an example of the error between the predicted and desired positions of the 4DoF robot in the case of the data-driven IK method. While it may be considered small in many other NN applications, the error in such highlighted example is  $3.83\text{mm}$ , which may not be acceptable for many critical robotic applications.

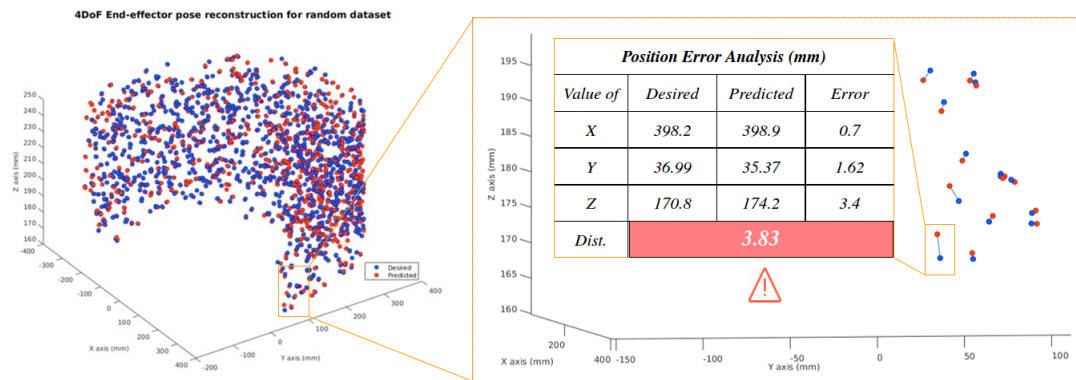


Figure 4.25: A visualization of a prediction example error calculated between the predicted and desired end-effector positions of the 4DoF robot in the case of the data-driven IK method.

In that sense, numerical methods provide better accuracy in the estimation of IK solutions. However, these methods may provide erroneous results depending on the properties of generalized inverse that is applied to compute the inverse Jacobian matrix. In the next section, we compare the use of the Moore-Penrose (MP), Unit-Consistent (UC) and Mixed generalized inverses in the numerical method presented in Chapter 3 and originally developed in [4].

## 4.2 Numerical IK methods

### 4.2.1 Experimental Setup and Selected Robotic Arms:

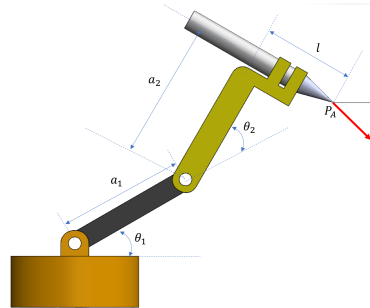
In this section, we investigated the stability of the numerical approach for the IK problem presented in [4] based on the type of matrix inverse used, and its effects vis-a-vis the attenuation factor  $\alpha$ . The experiments here aimed mainly to look at the stability of systems for incommensurate variables when their units are changed and rotations are applied. An example of this type of system would be a serial robotic manipulator with a combination of prismatic and revolute joints. In general, any generalized inverse would provide a mathematically correct solution, however our goal was to investigate matrix inverse solutions that provide consistently smooth trajectories despite the unit used.

For these experiments, serial robots with 3, 4, and 6 DoF were investigated. All three robots had a combination of prismatic and revolute joints in their designs, hence being incommensurate systems. These robotic manipulators are illustrated in Figure 4.26. Table 4.27, presents the D-H parameters of these chosen incommensurate robotic arms.

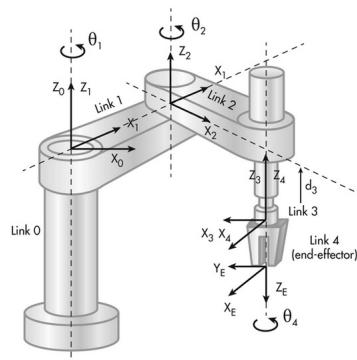
The behavior of the systems should be the same when the system is used with some variables expressed in related units. For some of these systems [21], the inverse Jacobian is computed using the Moore-Penrose (*MP*) pseudo-inverse in the process of finding the IK. However, the *MP* pseudo-inverse has been shown to change the behavior of the whole system when the units are changed [21,43,47]. Hence, for the computation of the Inverse Jacobian Matrix, the *MP* inverse was replaced by the Unit-Consistent (*UC*) inverse and Mixed (*MX*) inverse whenever applicable to do so in the numerical method. Table 4.26 presents a general overview of all the experiments we conducted. For some of the experiments, the attenuation factor  $\alpha$  was varied from 0.01 to 1 and the end-effector trajectories were observed. The goal of varying the attenuation factor was to check if smoother trajectories could be achieved whenever the chosen generalized inverse failed to compensate for unit changes. As the table shows,

Experiments	Robotic arm	Units varied for linear joints	Successful Inverses
3 different/arbitrary motions Various choices for the attenuation factor	3DoF	$m, dm, cm, mm$	$UC$ and $MX$
	4DoF	$m, dm, cm, mm$	$MP$ and $MX$
	6DoF	$m, dm, cm, mm$	$MP, UC$ and $MX$

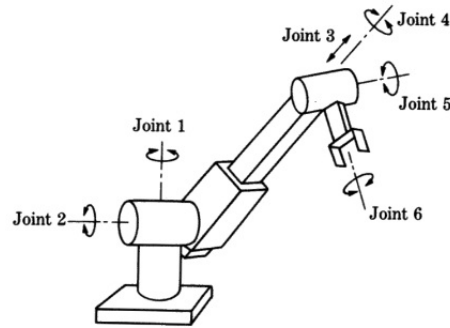
Table 4.26: Overview of the experiments conducted with the  $MP$ ,  $UC$  and  $MX$  generalized inverses.



(a) 3 DoF planar robot (from [21])



(b) 4 DoF robotic arm (SCARA)



(c) 6 DoF robotic arm (Stanford)

Figure 4.26: Incommensurate robotic arms used in the experiments related to numerical methods.

$i$	$\theta$	$d$	$a$	$\alpha$
1	$\theta_1$	0	1000	0
2	$\theta_2$	0	1100	90
3	0	$d_3$	0	0

(a) 3 DoF robotic arm (from [21])

$i$	$\theta$	$d$	$a$	$\alpha$
1	$\theta_1$	400	250	0
2	$\theta_2$	0	150	180
3	0	$d_3$	0	0
4	$\theta_4$	150	0	0

(b) 4 DoF robotic arm (SCARA)

$i$	$\theta$	$d$	$a$	$\alpha$
1	$\theta_1$	0	0	-90
2	$\theta_2$	140	0	90
3	0	$d_3$	0	0
4	$\theta_4$	0	0	-90
5	$\theta_5$	0	0	90
6	$\theta_6$	8.5	0	0

(c) 6 DoF robotic arm (Stanford)

Table 4.27: D-H Parameters of the serial robots illustrated in Figure 4.26 and used in the experiments with numerical methods. The angles  $\theta$  and  $\alpha$  are expressed in degrees. The variables  $d$  and  $a$  are shown in millimeters (mm) – but they were changed in the implementation to match the different choices of units.

### 4.2.2 Experimental Results:

In this section, we present the experimental results for the proposed numerical method. The goal of these experiments is to demonstrate the behavior of the end-effector trajectories when varying the units of the linear joints ( $mm$ ,  $cm$ ,  $dm$ , and  $m$ ) while applying different generalized inverses ( $MP$ ,  $UC$ , and  $MX$ ) to different robot architectures (D-H representation). As Table 4.26 indicates and it will be made clearer later in this section, none of the generalized inverses can be applied 'blindly' (in our case,  $MP$ ,  $UC$ , and  $MX$ , but the same can be said for *Left and Right Pseudo Inverses*). In fact, even the  $MX$  inverse has to be applied in a correct and systematic way in order to guarantee that the appropriate  $MP$  or  $UC$  inverse will be applied to the correct variables in the block partitions of the  $MX$  inverse. In the numerical method proposed in [4, 53], that was not being done and one of the main contributions of this research is to develop a systematic criteria for doing so. This new numerical method has been implemented by modifying both the serial and parallel modes in the original implementation from [4, 53]. Table 4.28 provides



Type of values	User defined values
Input	D-H parameters
	Initial joint configuration
	Number of iterations
	Attenuation factor ( $\alpha$ )
	Joint perturbations
	Desired final pose
	Desired pose errors
	Number of threads
	Noise
Output	Estimated final pose
	Number of iterations
	Computation time

Table 4.28: Overview of the required input parameters and output results obtained from the implementations of the numerical method.

the input and output information related to the algorithm in both modes. However, here we present only the experiments with the serial implementation of the algorithm. In this implementation, the algorithm does not use multi-threading to parallelize the search of a IK solution, as it did in its parallel implementation. But since the results are practically the same – except for execution time – we are, once again, omitting the results from the parallel implementation.

In the next three sections, we present three cases to illustrate the different behaviors of generalized inverses (refer to Table 4.26): 1) a 3DoF robot for which the *UC* inverse works, the *MP* inverse fails, and the *MX* inverse reduces to an *UC*; 2) a 4DoF robot where the *MP* inverse works, the *UC* inverse may or may not work, depending on the path of the robot, and the *MX* reduces to an *MP*; and 3) a 6DoF for which both the *UC* and the *MP* work and the block assignments in the *MX* is irrelevant since the Jacobian is non-singular and all three inverses reduce to a full-rank matrix inverse. Despite the above, the proposed numeric method *will always* apply the *MX* inverse, but by first inspecting the D-H table and the current robot configurations, and checking for the alignment (or not) between the axes of the revolute and prismatic joints. If the first joint affects the second one, the appropriate joint variables will be distributed among the different blocks of the *MX* so they can be handled separately by the *UC* or *MP* inverses.

### 4.2.2.1 3 DoF Serial Robot

In this section, we present a robot for which the  $UC$  inverse is required, and there is no place for the  $MP$  inverse. Also, as Table 4.26 indicates, a  $MX$  inverse can and should still be applied, however, when that is done correctly, the  $MX$  reduces to the  $UC$ .

#### Matrix Block-partition for the Application of the Mixed Inverse ( $MX$ ) to the 3DoF Robot (RRP + End-Effector Position)

As explained in Chapter 3, while the  $MP$  and  $UC$  inverses take as input the entire Jacobian matrix to compute its inverses, the  $MX$  relies on the concept of block matrix inverse to combine the use of the  $MP$  and  $UC$  generalized inverses. The only question remaining is to which block the variables should be assigned. As a rule of thumb, linear quantities that are affected by revolute joints earlier in the joint-link chain must be handled by the  $UC$  inverse, as this will provide unit consistency. The remaining variables must be handled by the  $MP$  inverse. In the case of the 3DoF serial manipulator with a RRP configuration, an inspection of the two early rotations will show that all variables are affected by them. So, the block partitioning for the Jacobian for this 3DoF robot becomes:

Given the Jacobian,

$$J = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix} = \begin{bmatrix} \frac{\partial X}{\partial \theta_1} & \frac{\partial X}{\partial \theta_2} & \frac{\partial X}{\partial l} \\ \frac{\partial Y}{\partial \theta_1} & \frac{\partial Y}{\partial \theta_2} & \frac{\partial Y}{\partial l} \end{bmatrix}$$

and the fact that all variables should be in the top-left block – so it is handled by the

$UC$  inverse – we have:

$$W = \begin{bmatrix} \frac{\partial X}{\partial \theta_1} & \frac{\partial X}{\partial \theta_2} & \frac{\partial X}{\partial l} \\ \frac{\partial Y}{\partial \theta_1} & \frac{\partial Y}{\partial \theta_2} & \frac{\partial Y}{\partial l} \end{bmatrix} \text{ as a } 2 \times 3 \text{ matrix}$$

$$X = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \text{ is a } 2 \times 1 \text{ matrix of zeros}$$

$$Y = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \text{ is a } 1 \times 3 \text{ matrix of zeros}$$

$$Z = [0] \text{ is a } 1 \times 1 \text{ matrix of zeros}$$

Then, the  $Mx$  inverse can be computed as follows, and hence reducing to a simple  $UC$  inverse.

$$J^{-M} = \begin{bmatrix} (W - XZ^{-P}Y)^{-U} & -W^{-U}X(Z - YW^{-U}X)^{-P} \\ -Z^{-P}Y(W - XZ^{-P}Y)^{-U} & (Z - YW^{-U}X)^{-P} \end{bmatrix}$$

$$J^{-M} = \begin{bmatrix} W^{-U} & 0 \\ 0 & 0 \end{bmatrix}$$

Since  $J^{-M}$  is a 4x3 matrix with one row and one column of zeros, we can assume the inverse Jacobian to match its original dimensions as a 3x2 matrix.

For each experiment we considered three arbitrary motions given by three pairs of initial and final poses. The experiments for these three motions will be detailed next.

### 3-DoF using Motion 1

- **3-DoF Robot using the MP Generalized Inverse**

First, we applied the Moore-Penrose generalized inverse with respect to Motion 1 of the 3DoF robot. Table 4.29 summarizes the overall results for Motion 1 by presenting the input parameters provided to the algorithm versus the number of iterations and final error in the position of the end-effector with respect to the desired position. It is important to mention that the attenuation parameter was kept constant and equal to 1 – unless specified otherwise.

Inverse used	MP Inverse							
	m case		dm case		cm case		mm case	
Initial Joints	$\vec{Q} =$	30 30 -0.7	$\vec{Q} =$	30 30 -7	$\vec{Q} =$	30 30 -70	$\vec{Q} =$	30 30 -700
Initial Position	$\vec{D} =$	0.8098 1.8026	$\vec{D} =$	8.098 18.026	$\vec{D} =$	80.98 180.26	$\vec{D} =$	809.8 1802.6
Desired Final Position	$\vec{D} =$	1.7873 2.8587	$\vec{D} =$	17.873 28.587	$\vec{D} =$	178.73 285.87	$\vec{D} =$	1787.3 2858.7
Calculated Position	$\vec{D} =$	1.7873 2.8586	$\vec{D} =$	17.8765 28.5857	$\vec{D} =$	178.7301 285.8699	$\vec{D} =$	1787.2 2858.6
Position Error	0.000069m		0.003760dm		0.000086cm		0.110255mm	
Iterations	8		7		6		109	
Computation time	0.062s		0.065s		0.059s		0.068s	

Table 4.29: *Experimental results obtained for Motion 1 of the 3DoF robot with  $\alpha = 1$  and MP Inverse.*

Next, Figure 4.27 shows the paths of the end-effector when the units of the linear joints in the 3DoF robot are varied from *m* to *dm*, *cm* and finally to *mm*, still for Motion 1. As expected, the behavior of the robot is quite different and unpredictable when the units

are varied. That is, since MP inverses do not guarantee unit consistency, a simple change of units causes the robot to follow quite different paths.

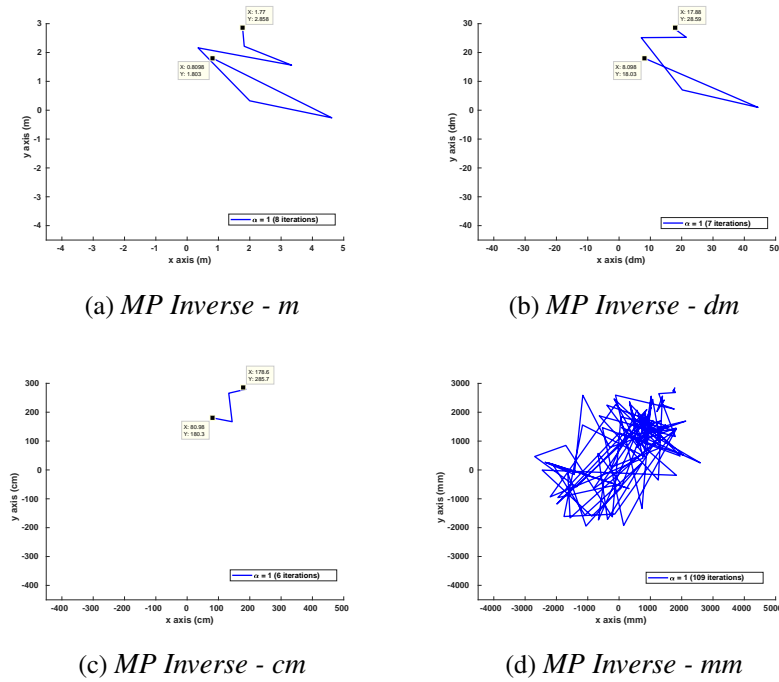


Figure 4.27: Behavior of the trajectories of the end-effector for Motion 1 of the 3DoF robot when varying the units while using the MP Inverse with  $\alpha = 1$ .

Next, we studied the effects of the attenuation parameter on the path followed by the end-effector under different units and still with the MP inverse. Figure 4.28 shows these results, and as it can be observed, the attenuation parameter cannot remedy the effects of the unit changes. In other words, the simple use of a generalized inverse that does not guarantee unit consistency can cause from mild differences to severe oscillations in the trajectory of the robot end-effector.

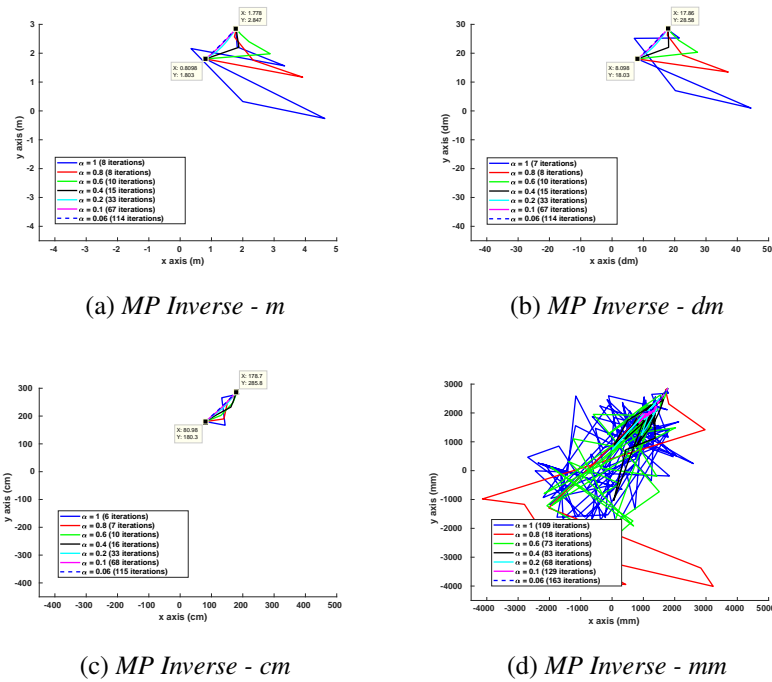


Figure 4.28: Behavior of the trajectories of the end-effector for Motion 1 of the 3DoF robot when varying the units while using the MP Inverse, for multiple values of  $\alpha$ .

• **3-DoF Robot using the UC Generalized Inverse**

Next, we applied the Unit Consistency generalized inverse to the same Motion 1 of the 3DoF robot. As before, Table 4.30 summarizes the overall results for Motion 1 by presenting the input parameters provided to the algorithm versus the number of iterations and final error in the position of the end-effector with respect to the desired position. Once again, the attenuation parameter was kept constant and equal to 1 – unless specified otherwise.

Inverse used	UC Inverse							
Results for:	m case		dm case		cm case		mm case	
Initial Joints	$\vec{Q} =$	30 30 -0.7	$\vec{Q} =$	30 30 -7	$\vec{Q} =$	30 30 -70	$\vec{Q} =$	30 30 -700
Initial Position	$\vec{D} =$	0.8098 1.8026	$\vec{D} =$	8.098 18.026	$\vec{D} =$	80.98 180.26	$\vec{D} =$	809.8 1802.6
Desired Final Position	$\vec{D} =$	1.7873 2.8587	$\vec{D} =$	17.873 28.587	$\vec{D} =$	178.73 285.87	$\vec{D} =$	1787.3 2858.7
Calculated Position	$\vec{D} =$	1.7872 2.8585	$\vec{D} =$	17.8722 28.5870	$\vec{D} =$	178.7223 285.8487	$\vec{D} =$	1787.2 2858.5
Position Error	0.000227m		0.002265dm		0.022650cm		0.226501mm	
Iterations	5		5		5		5	
Computation time	0.041s		0.047s		0.043s		0.068s	

Table 4.30: Experimental results obtained for Motion 1 of the 3DoF robot with  $\alpha = 1$  and UC Inverse.

Also as before, Figure 4.29 shows the paths of the end-effector when the units of the linear joints in the 3DoF robot are varied from *m* to *dm*, *cm* and finally to *mm*, still for Motion 1. As expected, the behavior of the robot is now exactly the same, no matter what units are used. That is, since UC inverses do guarantee unit consistency, no matter what units one chooses, the robot follows the exact same path.

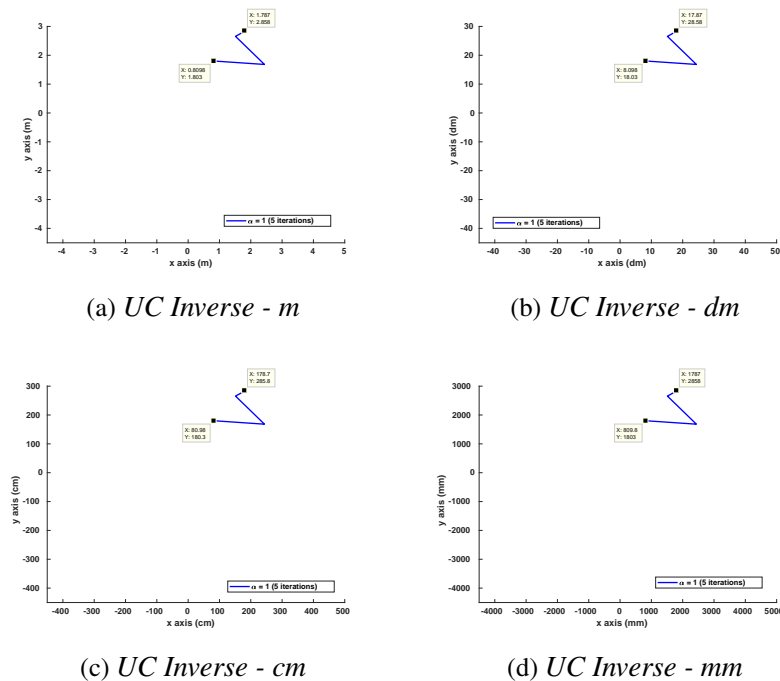


Figure 4.29: Behavior of the trajectories of the end-effector for Motion 1 of the 3DoF robot when varying the units while using the UC Inverse and  $\alpha = 1$ .

Finally, we studied the effects of the attenuation parameter  $\alpha$  on the path followed by the end-effector under different units and still using the  $UC$  inverse. Figure 4.30 shows these results. As also expected, while the attenuation parameter can help smoothing the path, it has no negative effect on the path followed by the robot. In this case, since the  $UC$  already provided stability regarding changes of units, the attenuation parameter can perform its role effectively.

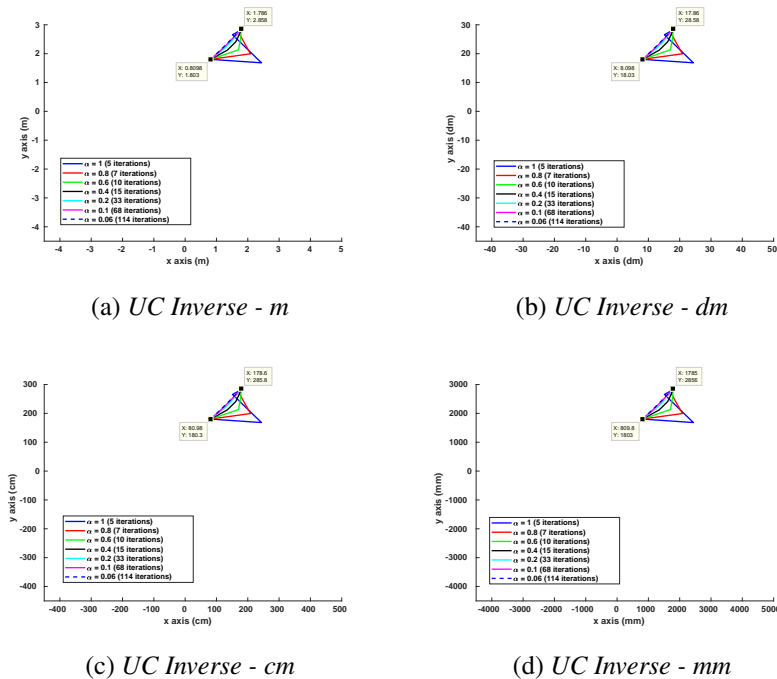


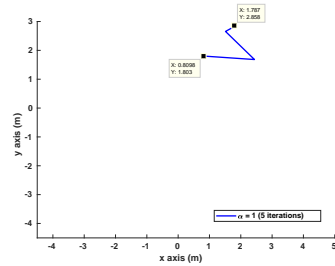
Figure 4.30: Behavior of the trajectories of the end-effector for Motion 1 of the 3DoF robot when varying the units while using the  $UC$  Inverse, for multiple values of  $\alpha$ .

### • 3-DoF Robot using the MX Generalized Inverse

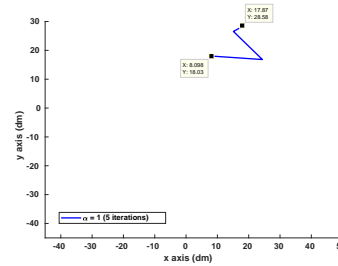
The last generalized inverse used was the Mixed inverse, which was applied to the same Motion 1 of the 3DoF robot. As we explained at the beginning of this section, the  $MX$  inverse reduces to the  $UC$  inverse. So, we expected and we did observe the exact same results as for the  $UC$  inverse. Table 4.31, and Figures 4.31 and 4.32 below summarize these results for the  $MX$  inverse.

Inverse used	MX Inverse							
	m case		dm case		cm case		mm case	
Initial Joints	$\vec{Q} =$	$\begin{bmatrix} 30 \\ 30 \\ -0.7 \end{bmatrix}$	$\vec{Q} =$	$\begin{bmatrix} 30 \\ 30 \\ -7 \end{bmatrix}$	$\vec{Q} =$	$\begin{bmatrix} 30 \\ 30 \\ -70 \end{bmatrix}$	$\vec{Q} =$	$\begin{bmatrix} 30 \\ 30 \\ -700 \end{bmatrix}$
Initial Position	$\vec{D} =$	$\begin{bmatrix} 0.8098 \\ 1.8026 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} 8.098 \\ 18.026 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} 80.98 \\ 180.26 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} 809.8 \\ 1802.6 \end{bmatrix}$
Desired Final Position	$\vec{D} =$	$\begin{bmatrix} 1.7873 \\ 2.8587 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} 17.873 \\ 28.587 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} 178.73 \\ 285.87 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} 1787.3 \\ 2858.7 \end{bmatrix}$
Calculated Position	$\vec{D} =$	$\begin{bmatrix} 1.7872 \\ 2.8585 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} 17.8722 \\ 28.5870 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} 178.7223 \\ 285.8487 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} 1787.2 \\ 2858.5 \end{bmatrix}$
Position Error	0.000227m		0.002265dm		0.022650cm		0.226501mm	
Iterations	5		5		5		5	
Computation time	0.042s		0.046s		0.043s		0.048s	

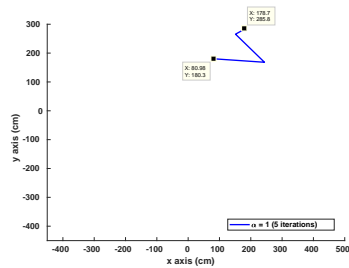
Table 4.31: Experimental results obtained for Motion 1 of the 3DoF robot with  $\alpha = 1$  and MX Inverse.



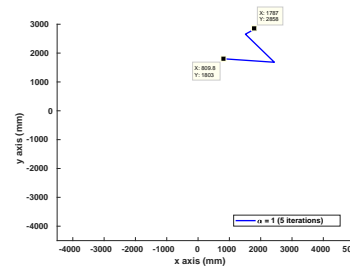
(a) MX Inverse - m



(b) MX Inverse - dm



(c) MX Inverse - cm



(d) MX Inverse - mm

Figure 4.31: Behavior of the trajectories of the end-effector for Motion 1 of the 3DoF robot when varying the units while using the MX Inverse and  $\alpha = 1$ .



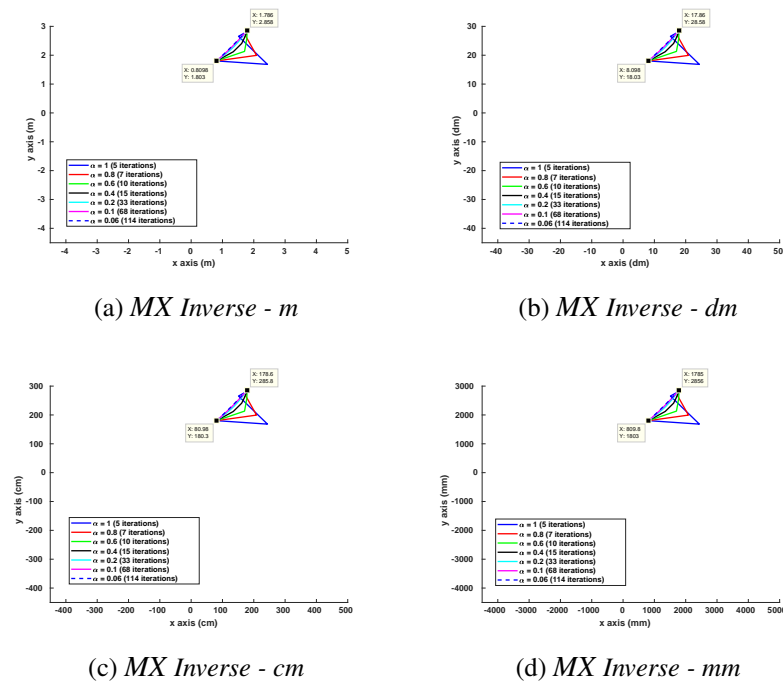


Figure 4.32: Behavior of the trajectories of the end-effector for Motion 1 of the 3DoF robot when varying the units while using the *MX Inverse*, for multiple values of  $\alpha$ .

### 3-DoF using Motions 2 and 3

- **3-DoF Robot using the MP Generalized Inverse**

Next, we executed the same tests – i.e. to investigate: 1) the effects of changing units (from *m* to *dm*, *cm* and finally to *mm*) on the trajectory of the end-effector; and 2) the effects of different attenuation parameters on the same trajectories – for two additional and arbitrary motions of the 3-DoF robot: Motion 2 and Motion 3. The goal here was to demonstrate that the problems encountered earlier were not a function of the initial and final position of the end-effector (ie. the potential path followed by the robot), but indeed the properties of the different generalized inverses.

Table 4.32 and Figures 4.33 and 4.34 summarize the results and demonstrate the same conclusions reached above for Motion 1 and the *MP* inverse, now applied to Motion 2. At the same time, Table 4.33 and Figures 4.35 and 4.36 summarize the results and demonstrate the same conclusions for Motion 3 also using the *MP* inverse.

Inverse used	MP Inverse			
	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ -0.7 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ -7 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ -70 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ -700 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.8098 \\ 1.8026 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 8.098 \\ 18.026 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 80.98 \\ 180.26 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 809.8 \\ 1802.6 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} -1.5 \\ 0.5 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -15 \\ 5 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -150 \\ 50 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -1500 \\ 500 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} -1.5000 \\ 0.5000 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -15.0000 \\ 4.9999 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -149.999 \\ 49.998 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -1500.00 \\ 500.00 \end{bmatrix}$
Position Error	0.000014m	0.000140dm	0.001399cm	0.246989mm
Iterations	6	5	10	9
Computation time	0.054s	0.053s	0.120s	0.097s

Table 4.32: Experimental results obtained for Motion 2 of the 3DoF robot with  $\alpha = 1$  and MP Inverse.

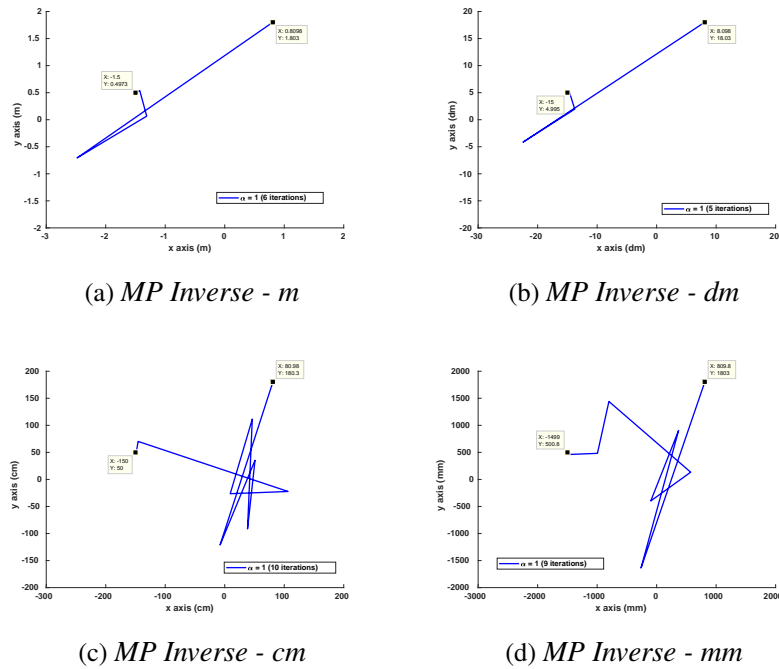


Figure 4.33: Behavior of the trajectories of the end-effector for Motion 2 of the 3DoF robot when varying the units while using the MP Inverse and  $\alpha = 1$ .

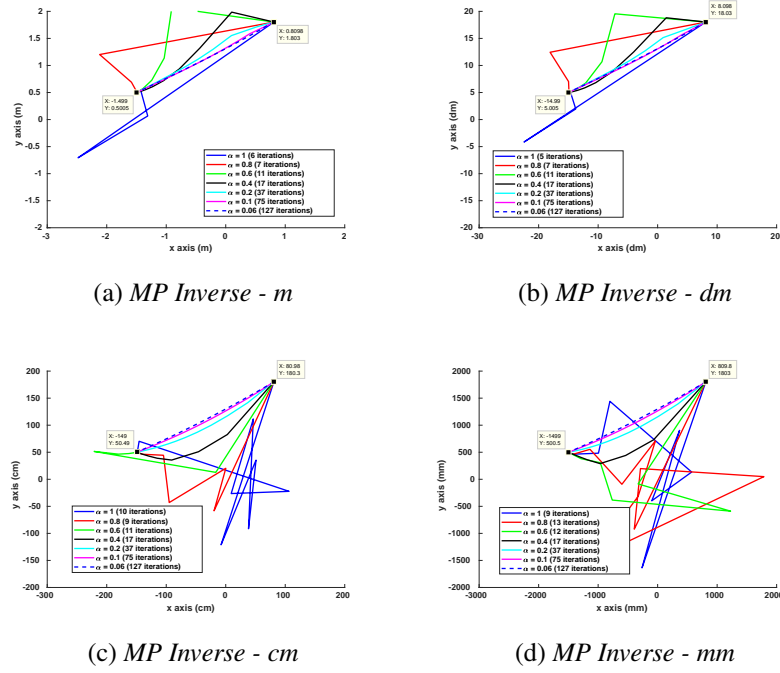


Figure 4.34: Behavior of the trajectories of the end-effector for Motion 2 of the 3DoF robot when varying the units while using the MP Inverse, for multiple values of  $\alpha$ .

Inverse used	MP Inverse			
	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ -0.7 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ -7 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ -70 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ -700 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.8098 \\ 1.8026 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 8.098 \\ 18.026 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 80.98 \\ 180.26 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 809.8 \\ 1802.6 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} 1.5 \\ -0.86 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 15 \\ -8.6 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 150 \\ 86 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -1500 \\ 1500 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} 1.5000 \\ -0.8599 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 15.0000 \\ -8.5999 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 150.0000 \\ -86.0000 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 1500.00 \\ -860.00 \end{bmatrix}$
Position Error	0.000069m	0.000063dm	0.000007cm	0.000077mm
Iterations	5	5	14	16
Computation time	0.045s	0.035s	0.093s	0.122s

Table 4.33: Experimental results obtained for Motion 3 of the 3DoF robot with  $\alpha = 1$  and MP Inverse.

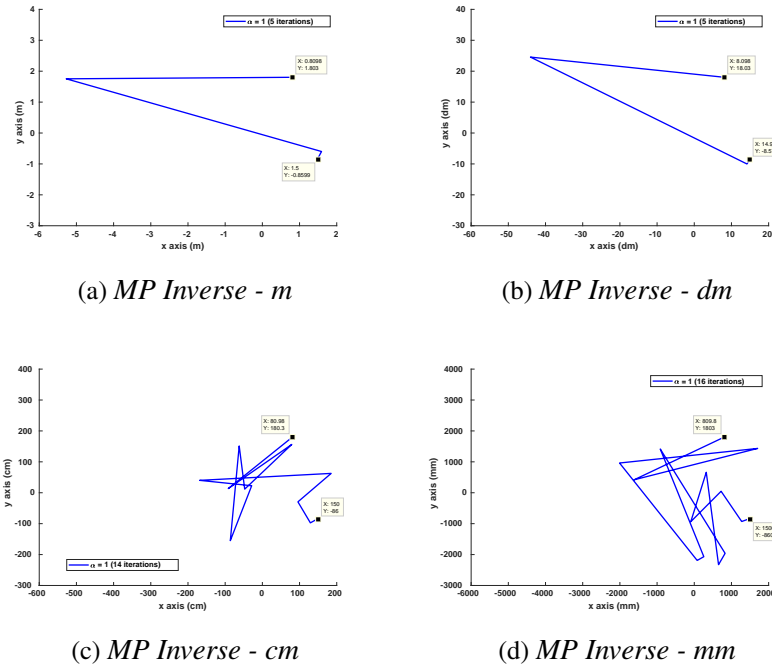


Figure 4.35: Behavior of the trajectories of the end-effector for Motion 3 of the 3DoF robot when varying the units while using the MP Inverse and  $\alpha = 1$ .

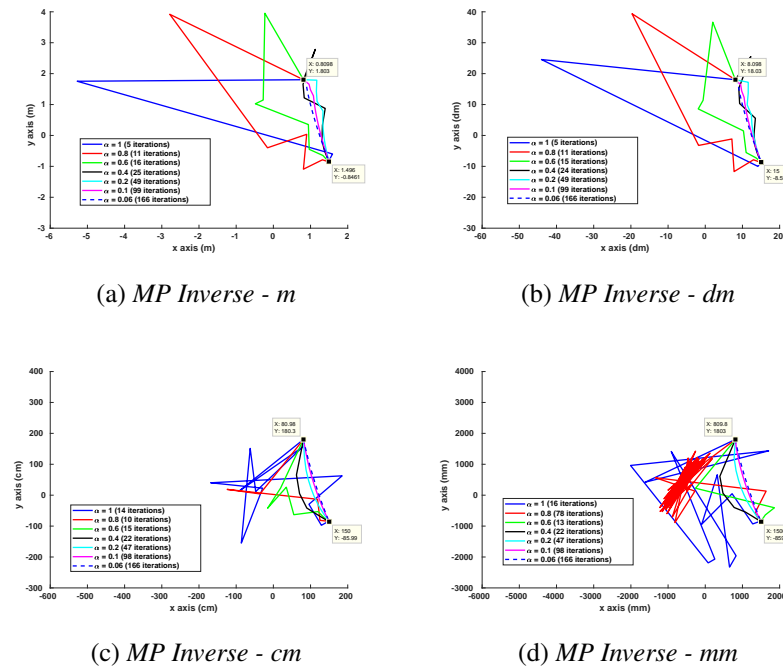


Figure 4.36: Behavior of the trajectories of the end-effector for Motion 3 of the 3DoF robot when varying the units while using the MP Inverse, for multiple values of  $\alpha$ .

• 3-DoF Robot using the UC Generalized Inverse

Once again, the Unit Consistency generalized inverse was applied to two additional and arbitrary motions of the robot and the results are presented in Table 4.34 and Figures 4.37 and 4.38 for Motion 2, and Table 4.35 and Figures 4.39 and 4.40 for Motion 3. As it was the case with Motion 1, with Motions 2 and 3 the UC inverse works consistently despite changes of units or the attenuation factor.

Inverse used	UC Inverse			
	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ -0.7 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ -7 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ -70 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ -700 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.8098 \\ 1.8026 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 8.098 \\ 18.026 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 80.98 \\ 180.26 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 809.8 \\ 1802.6 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} -1.5 \\ 0.5 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -15 \\ 5 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -150 \\ 50 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -1500 \\ 500 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} -1.5000 \\ 0.5000 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -15.0000 \\ 4.9999 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -149.999 \\ 49.998 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -1500.00 \\ 500.00 \end{bmatrix}$
Position Error	0.000014m	0.000140dm	0.001399cm	0.246989mm
Iterations	6	6	6	6
Computation time	0.054s	0.053s	0.053s	0.057s

Table 4.34: Experimental results obtained for Motion 2 of the 3DoF robot with  $\alpha = 1$  and UC Inverse.

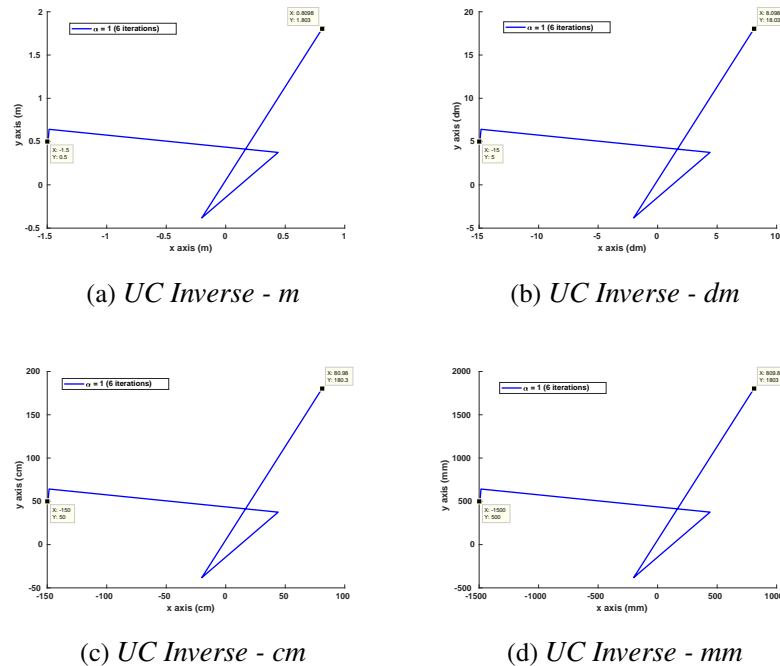


Figure 4.37: Behavior of the trajectories of the end-effector for Motion 2 of the 3DoF robot when varying the units while using the UC Inverse and  $\alpha = 1$ .

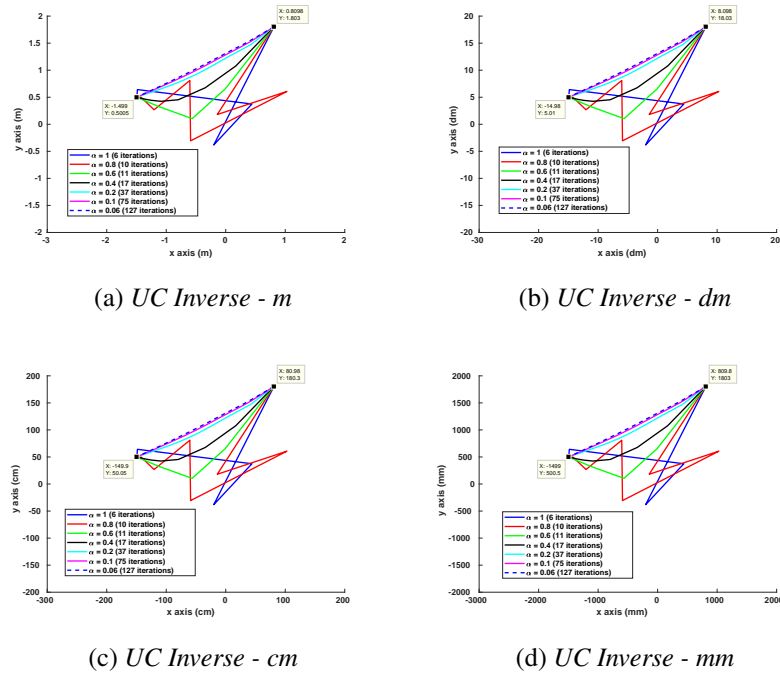


Figure 4.38: Behavior of the trajectories of the end-effector for Motion 2 of the 3DoF robot when varying the units while using the UC Inverse, for multiple values of  $\alpha$ .

Inverse used	UC Inverse			
	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{matrix} 30 \\ 30 \\ -0.7 \end{matrix}$	$\vec{Q} = \begin{matrix} 30 \\ 30 \\ -7 \end{matrix}$	$\vec{Q} = \begin{matrix} 30 \\ 30 \\ -70 \end{matrix}$	$\vec{Q} = \begin{matrix} 30 \\ 30 \\ -700 \end{matrix}$
Initial Position	$\vec{D} = \begin{matrix} 0.8098 \\ 1.8026 \end{matrix}$	$\vec{D} = \begin{matrix} 8.098 \\ 18.026 \end{matrix}$	$\vec{D} = \begin{matrix} 80.98 \\ 180.26 \end{matrix}$	$\vec{D} = \begin{matrix} 809.8 \\ 1802.6 \end{matrix}$
Desired Final Position	$\vec{D} = \begin{matrix} 1.5 \\ -0.86 \end{matrix}$	$\vec{D} = \begin{matrix} 15 \\ -8.6 \end{matrix}$	$\vec{D} = \begin{matrix} 150 \\ -86 \end{matrix}$	$\vec{D} = \begin{matrix} 1500 \\ -860 \end{matrix}$
Calculated Position	$\vec{D} = \begin{matrix} 1.5000 \\ 0.8600 \end{matrix}$	$\vec{D} = \begin{matrix} 15.0000 \\ -8.6000 \end{matrix}$	$\vec{D} = \begin{matrix} 150.0000 \\ -86.0000 \end{matrix}$	$\vec{D} = \begin{matrix} 1500.00 \\ 860.00 \end{matrix}$
Position Error	0.000000m	0.000001dm	0.000005cm	0.000052mm
Iterations	11	11	11	11
Computation time	0.090s	0.085s	0.099s	0.081s

Table 4.35: Experimental results obtained for Motion 3 of the 3DoF robot with  $\alpha = 1$  and UC Inverse.

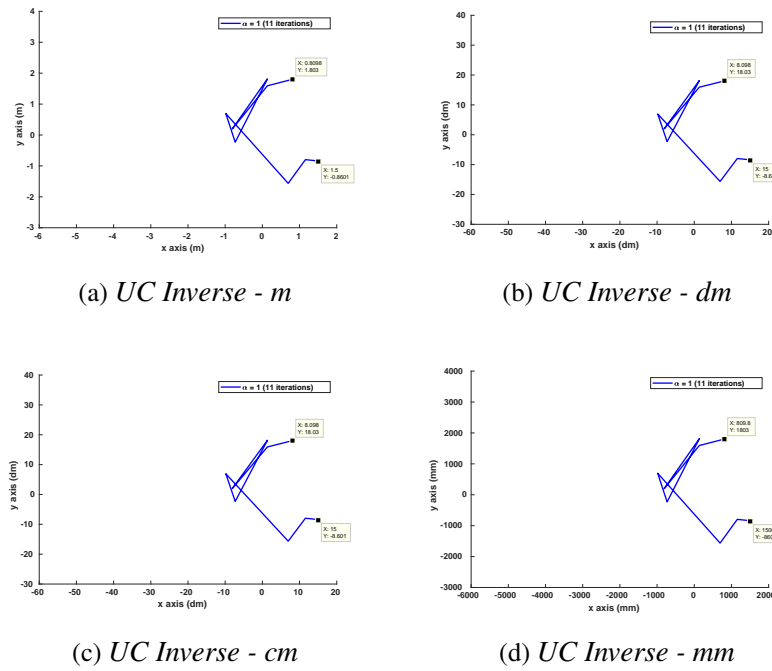


Figure 4.39: Behavior of the trajectories of the end-effector for Motion 3 of the 3DoF robot when varying the units while using the UC Inverse and  $\alpha = 1$ .

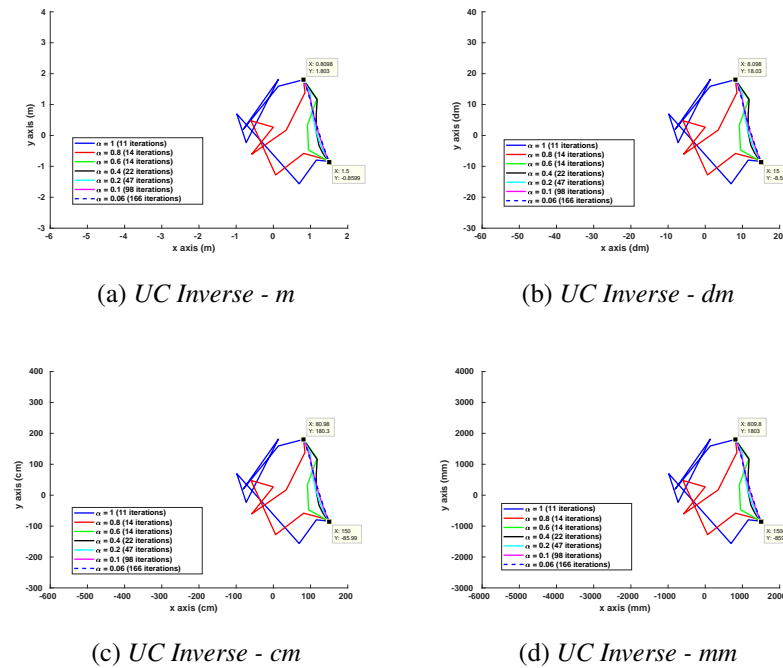


Figure 4.40: Behavior of the trajectories of the end-effector for Motion 3 of the 3DoF robot when varying the units while using the UC Inverse, for multiple values of  $\alpha$ .

• 3-DoF Robot using the MX Generalized Inverse

Finally, the Mixed inverse was applied to the same two additional and arbitrary motions of the robot and the results are presented in Table 4.36 and Figures 4.41 and 4.42 for Motion 2, and Table 4.37 and Figures 4.43 and 4.44 for Motion 3. As it was the case with Motion 1, with Motions 2 and 3 the *MX* inverse also works exactly like the *UC* inverse, and still consistently despite changes of units or the attenuation factor.

Inverse used	MX Inverse							
	m case		dm case		cm case		mm case	
Initial Joints	$\vec{Q} =$	$\begin{bmatrix} 30 \\ 30 \\ -0.7 \end{bmatrix}$	$\vec{Q} =$	$\begin{bmatrix} 30 \\ 30 \\ -7 \end{bmatrix}$	$\vec{Q} =$	$\begin{bmatrix} 30 \\ 30 \\ -70 \end{bmatrix}$	$\vec{Q} =$	$\begin{bmatrix} 30 \\ 30 \\ -700 \end{bmatrix}$
Initial Position	$\vec{D} =$	$\begin{bmatrix} 0.8098 \\ 1.8026 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} 8.098 \\ 18.026 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} 80.98 \\ 180.26 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} 809.8 \\ 1802.6 \end{bmatrix}$
Desired Final Position	$\vec{D} =$	$\begin{bmatrix} -1.5 \\ 0.5 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} -15 \\ 5 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} -150 \\ 50 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} -1500 \\ 500 \end{bmatrix}$
Calculated Position	$\vec{D} =$	$\begin{bmatrix} -1.5000 \\ 0.5000 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} -15.0000 \\ 4.9999 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} -149.999 \\ 49.998 \end{bmatrix}$	$\vec{D} =$	$\begin{bmatrix} -1500.00 \\ 500.00 \end{bmatrix}$
Position Error	0.000014m		0.000140dm		0.001399cm		0.013986mm	
Iterations	6		6		6		6	
Computation time	0.060s		0.059s		0.061s		0.058s	

Table 4.36: Experimental results obtained for Motion 2 of the 3DoF robot with  $\alpha = 1$  and *MX* Inverse.

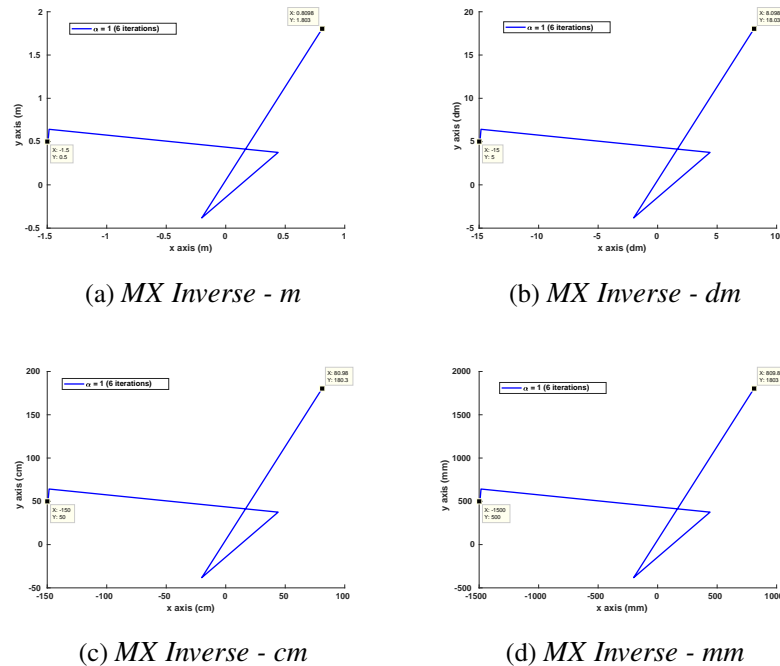


Figure 4.41: Behavior of the trajectories of the end-effector for Motion 2 of the 3DoF robot when varying the units while using the *MX* Inverse and  $\alpha = 1$ .



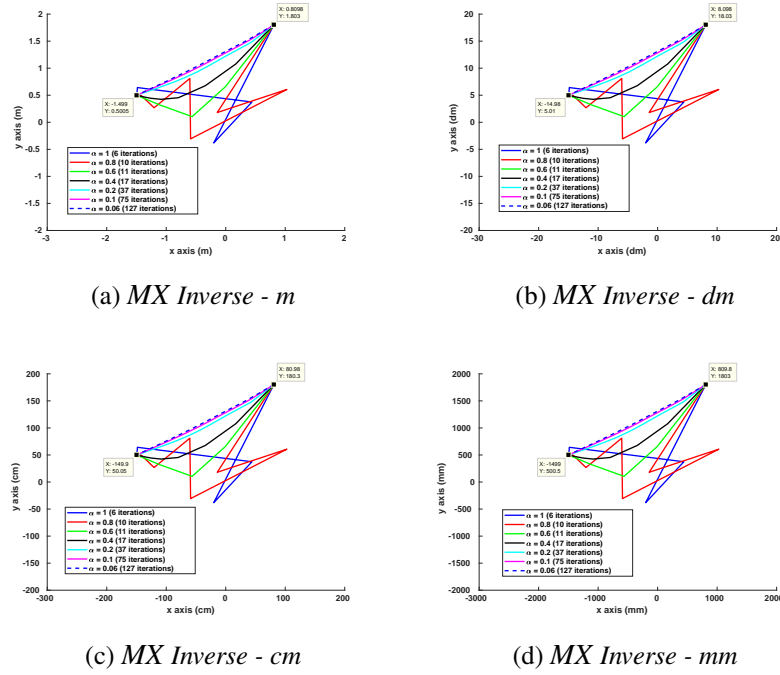


Figure 4.42: Behavior of the trajectories of the end-effector for Motion 2 of the 3DoF robot when varying the units while using the MX Inverse, for multiple values of  $\alpha$ .

Inverse used	MX Inverse			
	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{matrix} 30 \\ 30 \\ -0.7 \end{matrix}$	$\vec{Q} = \begin{matrix} 30 \\ 30 \\ -7 \end{matrix}$	$\vec{Q} = \begin{matrix} 30 \\ 30 \\ -70 \end{matrix}$	$\vec{Q} = \begin{matrix} 30 \\ 30 \\ -700 \end{matrix}$
Initial Position	$\vec{D} = \begin{matrix} 0.8098 \\ 1.8026 \end{matrix}$	$\vec{D} = \begin{matrix} 8.098 \\ 18.026 \end{matrix}$	$\vec{D} = \begin{matrix} 80.98 \\ 180.26 \end{matrix}$	$\vec{D} = \begin{matrix} 809.8 \\ 1802.6 \end{matrix}$
Desired Final Position	$\vec{D} = \begin{matrix} 1.5 \\ -0.86 \end{matrix}$	$\vec{D} = \begin{matrix} 15 \\ -8.6 \end{matrix}$	$\vec{D} = \begin{matrix} 150 \\ -86 \end{matrix}$	$\vec{D} = \begin{matrix} 1500 \\ -860 \end{matrix}$
Calculated Position	$\vec{D} = \begin{matrix} 1.5000 \\ 0.8600 \end{matrix}$	$\vec{D} = \begin{matrix} 15.0000 \\ -8.6000 \end{matrix}$	$\vec{D} = \begin{matrix} 150.0000 \\ -86.0000 \end{matrix}$	$\vec{D} = \begin{matrix} 1500.00 \\ 860.00 \end{matrix}$
Position Error	0.000000m	0.000001dm	0.000005cm	0.000052mm
Iterations	11	11	11	11
Computation time	0.092s	0.087s	0.097s	0.087s

Table 4.37: Experimental results obtained for Motion 3 of the 3DoF robot with  $\alpha = 1$  MX and Inverse.

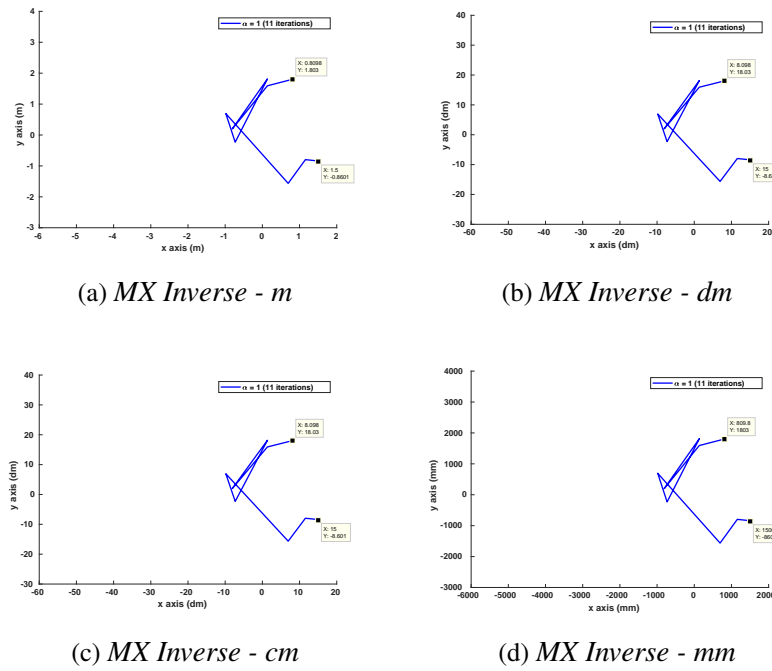


Figure 4.43: Behavior of the trajectories of the end-effector for Motion 3 of the 3DoF robot when varying the units while using the MX Inverse and  $\alpha = 1$ .

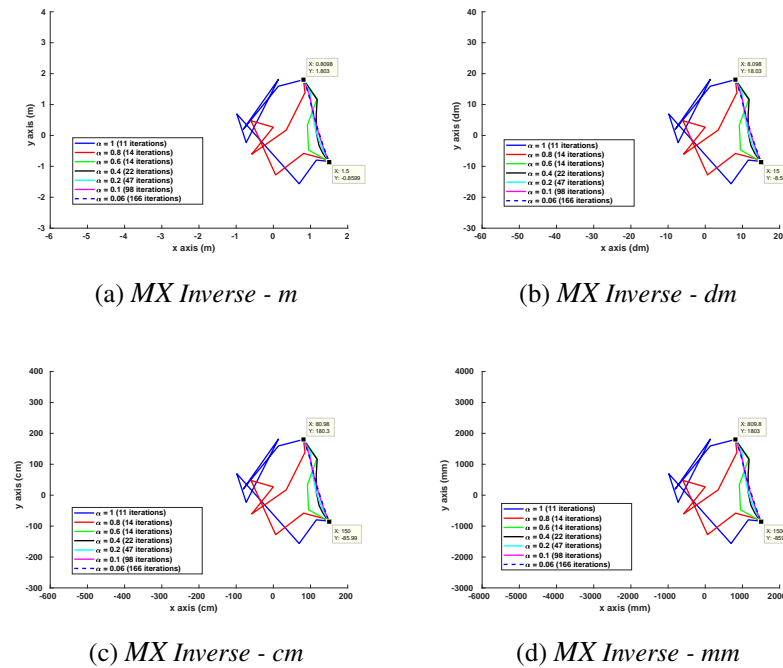


Figure 4.44: Behavior of the trajectories of the end-effector for Motion 3 of the 3DoF robot when varying the units while using the MX Inverse, for multiple values of  $\alpha$ .

### 4.2.2.2 4 DoF Serial Robot

In the previous case, the 3DoF robot had a prismatic joint that was affected by previous rotations in the chain. In this section, we present a robot for which all rotation and prismatic joints are aligned and hence the *MP* inverse suffices. Once again, a *MX* inverse can and should still be applied, however, when that is done correctly, the *MX* now reduces to the *MP* inverse (see Table 4.26).

#### Matrix Block-partition for the Application of the Mixed Inverse (*MX*) to the 4DoF Robot (RRPR + End-Effector Pose)

In order to apply the concept of block matrix inverse to the *MX* inverse we need to identify to which block the variables should be assigned. Based on the rule of thumb already established, this 4DoF serial manipulator with a RRPR configuration has a linear joint that appears after two of its revolute joints in its chain of the D-H representation. So, it might be possible that this linear joint variable will need to be included in the block *W* of the *MX* inverse. However, their order in the chain is a necessary but not sufficient condition, and since the axis of translation of the linear joint is aligned (parallel) to the axes of rotation of the revolute joints, the linear joint is not affected by these rotations, and hence it does not need to be included in the block *W* and any change of units will be adequately handled by the *MP* Inverse. So, the block partitioning for the Jacobian for this 4DoF robot becomes:

Given the Jacobian,

$$J = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix} = \begin{bmatrix} \frac{\partial X}{\partial \theta_1} & \frac{\partial X}{\partial \theta_2} & \frac{\partial X}{\partial d_3} & \frac{\partial X}{\partial \theta_4} \\ \frac{\partial Y}{\partial \theta_1} & \frac{\partial Y}{\partial \theta_2} & \frac{\partial Y}{\partial d_3} & \frac{\partial Y}{\partial \theta_4} \\ \frac{\partial Z}{\partial \theta_1} & \frac{\partial Z}{\partial \theta_2} & \frac{\partial Z}{\partial d_3} & \frac{\partial Z}{\partial \theta_4} \\ \frac{\partial R_o}{\partial \theta_1} & \frac{\partial R_o}{\partial \theta_2} & \frac{\partial R_o}{\partial d_3} & \frac{\partial R_o}{\partial \theta_4} \\ \frac{\partial P_i}{\partial \theta_1} & \frac{\partial P_i}{\partial \theta_2} & \frac{\partial P_i}{\partial d_3} & \frac{\partial P_i}{\partial \theta_4} \\ \frac{\partial Y_a}{\partial \theta_1} & \frac{\partial Y_a}{\partial \theta_2} & \frac{\partial Y_a}{\partial d_3} & \frac{\partial Y_a}{\partial \theta_4} \end{bmatrix}$$

and the fact that all variables should be in the bottom-right block – so it is handled by the UC inverse – we have:

$$W = [0] \text{ as a } 1 \times 1 \text{ matrix of zeros}$$

$$X = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \text{ is a } 1 \times 4 \text{ matrix of zeros}$$

$$Y = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ is a } 6 \times 1 \text{ matrix of zeros}$$

$$Z = J \text{ is a } 6 \times 4 \text{ matrix}$$

Then, the  $Mx$  inverse can be computed as follows, and hence reducing to a simple MP inverse:

$$J^{-M} = \begin{bmatrix} (W - XZ^{-P}Y)^{-U} & -W^{-U}X(Z - YW^{-U}X)^{-P} \\ -Z^{-P}Y(W - XZ^{-P}Y)^{-U} & (Z - YW^{-U}X)^{-P} \end{bmatrix}$$

$$J^{-M} = \begin{bmatrix} 0 & 0 \\ 0 & Z^{-P} \end{bmatrix}$$

Since  $J^{-M}$  is a  $5 \times 7$  matrix with one row and one column of zeros, we can assume the inverse Jacobian to match its original dimensions as a  $4 \times 6$  matrix.

For each experiment, we also considered three arbitrary motions given by three pairs of initial and final poses. The experiments for these three motions will be detailed next.

#### 4-DoF using Motion 1

- **4-DoF Robot using the MP Generalized Inverse**

First, we applied the Moore-Penrose generalized inverse with respect to Motion 1 of the 4DoF robot. Table 4.38 summarizes the overall results for Motion 1 by presenting the input parameters provided to the algorithm versus the number of iterations and final error in the position of the end-effector with respect to the desired position. It is important to mention that the attenuation parameter was kept constant and equal to 1 – unless specified otherwise. As the table shows, the initial configuration of the joint variables for Motion 1 was  $\vec{Q} = \left[ \theta_1 = 0, \theta_2 = 0, d_3 = 0, \theta_4 = 90 \right]^T$ .

Inverse used	MP Inverse			
Results for:	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 90 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.6000 \\ 0 \\ -0.1145 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 6.000 \\ 0 \\ -1.145 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 60.00 \\ 0 \\ -11.45 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 600.0 \\ 0 \\ 114.5 \\ -90 \\ 0 \\ 0 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} 0.5 \\ 0.3 \\ -0.15 \\ -85 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 5 \\ 3 \\ -1.5 \\ -85 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 50 \\ 30 \\ -15 \\ -85 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 500 \\ 300 \\ -150 \\ -85 \\ 0 \\ 0 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} 0.5000 \\ 0.3000 \\ -0.1500 \\ -85.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 4.9997 \\ 2.9998 \\ -1.5000 \\ -85.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 49.9970 \\ 29.9984 \\ -15.0000 \\ -85.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 499.9699 \\ 299.9841 \\ -150.0000 \\ -85.0000 \\ 0 \\ 0 \end{bmatrix}$
Position Error	0.000034m	0.000341dm	0.003407cm	0.034073mm
Orientation Error	0.000000°	0.000000°	0.000000°	0.000000°
Iterations	5	5	5	5
Computation time	0.074s	0.065s	0.059s	0.073s

Table 4.38: Experimental results obtained for Motion 1 of the 4DoF robot with  $\alpha = 1$  and MP Inverse.

Next, Figure 4.45 shows the paths of the end-effector when the units of the linear joints in the 4DoF robot are varied from *m* to *dm*, *cm* and finally to *mm*, still for Motion 1. Because, the axes of rotation of the revolute joints are parallel to the axis of translation of the linear joint, the behavior of the end-effector is the same when the units are varied. That is, changes of units do not affect the paths of the end-effector and the system is handled consistently by simply using a *MP* inverse.

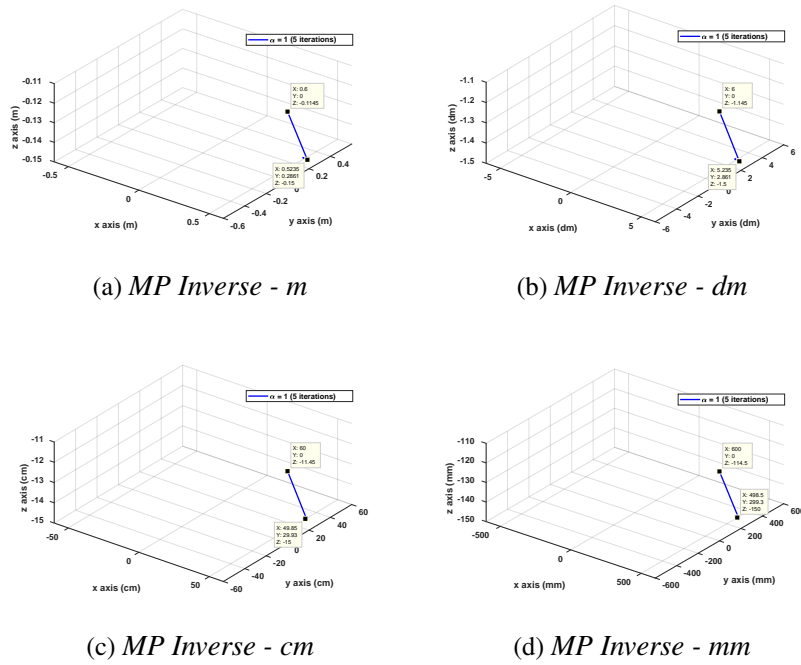


Figure 4.45: Behavior of the trajectories of the end-effector for Motion 1 of the 4DoF robot when varying the units while using the *MP Inverse* with  $\alpha = 1$ .

Next, we studied the effects of the attenuation parameter  $\alpha$  on the path followed by the end-effector under different units and still with the *MP inverse*. Figure 4.46 shows these results, and as it can be observed, while  $\alpha$  can be effectively used to make the path smoother, the paths followed by each choice of  $\alpha$  are exactly the same despite the change of units.

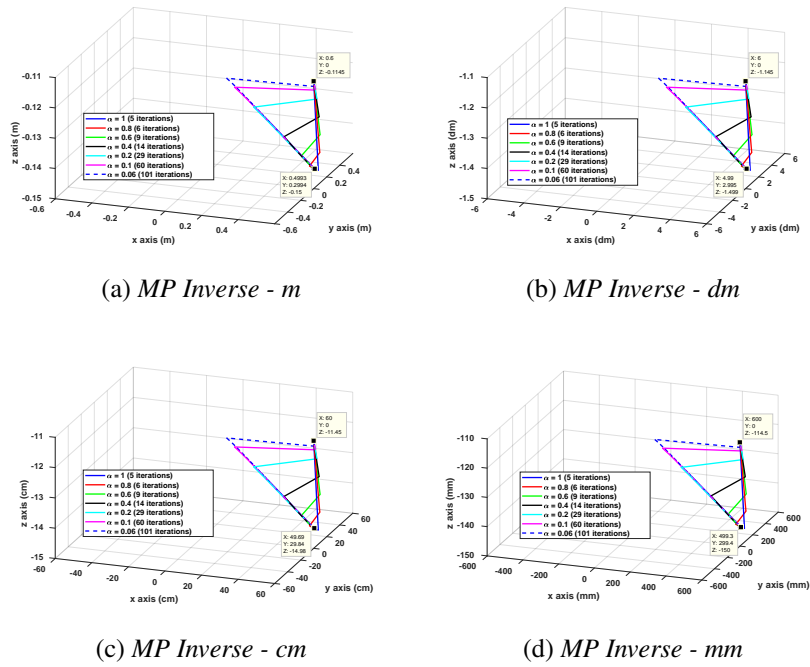


Figure 4.46: Behavior of the trajectories of the end-effector for Motion 1 of the 4DoF robot when varying the units while using the MP Inverse, for multiple values of  $\alpha$ .

#### • 4-DoF Robot using the UC Generalized Inverse

Next, we applied the Unit Consistency generalized inverse to the same Motion 1 of the 4DoF robot, i.e. with the initial value of the joint vector still set to:

$$\vec{Q} = \left[ \theta_1 = 0, \quad \theta_2 = 0, \quad d_3 = 0, \quad \theta_4 = 90 \right]^T.$$

As before, Table 4.39 summarizes the overall results for Motion 1 by presenting the input parameters provided to the algorithm versus the number of iterations and final error in the position of the end-effector with respect to the desired position. Once again, the attenuation parameter was kept constant and equal to 1.

Inverse used	UC Inverse			
Results for:	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 90 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.6000 \\ 0 \\ -0.1145 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 6.000 \\ 0 \\ -1.145 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 60.00 \\ 0 \\ -11.45 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 600.0 \\ 0 \\ 114.5 \\ -90 \\ 0 \\ 0 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} 0.5 \\ 0.3 \\ -0.15 \\ -85 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 5 \\ 3 \\ -1.5 \\ -85 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 50 \\ 30 \\ -15 \\ -85 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 500 \\ 300 \\ -150 \\ -85 \\ 0 \\ 0 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} 0.5000 \\ 0.2999 \\ -0.1500 \\ -85.0436 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 5.0020 \\ 2.9946 \\ -1.5000 \\ -85.0902 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 49.9976 \\ 29.9988 \\ -15.0000 \\ -85.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 500.0028 \\ 300.1528 \\ -150.0000 \\ -85.0471 \\ 0 \\ 0 \end{bmatrix}$
Position Error	0.043556m	0.090356dm	0.002701cm	0.159864mm
Orientation Error	0.014519°	0.030057°	0.000000°	0.015687
Iterations	9	8	8	5
Computation time	0.149s	0.136s	0.130s	0.087s

Table 4.39: Experimental results obtained for Motion 1 of the 4DoF robot with  $\alpha = 1$  and UC Inverse.

Also as before, Figure 4.47 shows the paths of the end-effector when the units of the linear joints in the 4DoF robot are varied from *m* to *dm*, *cm* and finally to *mm*, still for Motion 1. Not surprisingly, the behavior of the robot is quite different and unpredictable when the units are varied. That is, the UC inverse is mishandling variables that are not affected by the change of units of the linear joint. As it can be seen, a simple change of units causes the robot to follow quite different paths.



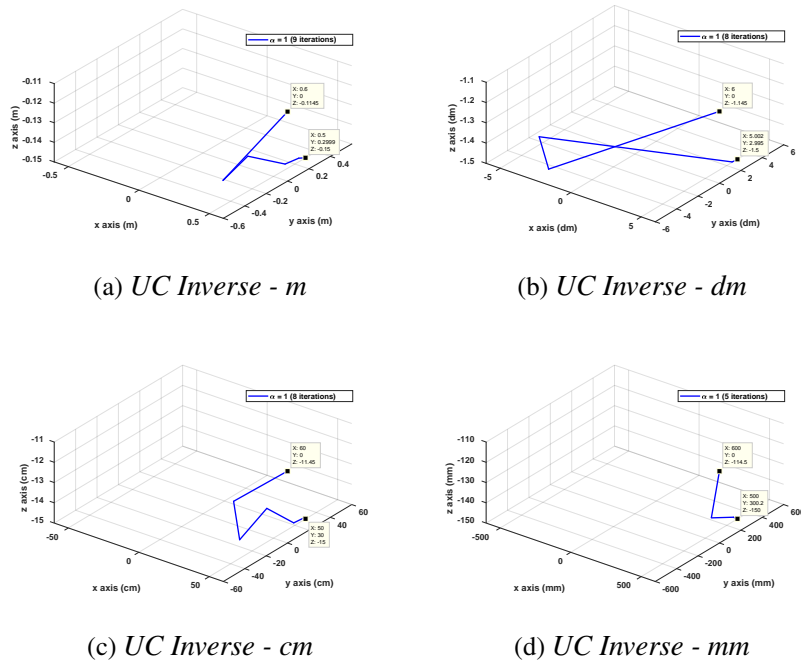


Figure 4.47: Behavior of the trajectories of the end-effector for Motion 1 of the 4DoF robot when varying the units while using the *UC Inverse* and  $\alpha = 1$ .

Finally, we studied the effects of the attenuation parameter on the path followed by the end-effector under different units and still using the *UC inverse*. Figure 4.48 shows these results. Now, also as expected, the attenuation parameter can not correct the negative effect on the path followed by the robot since the the *UC* is unable to provide consistency when the units are changed.

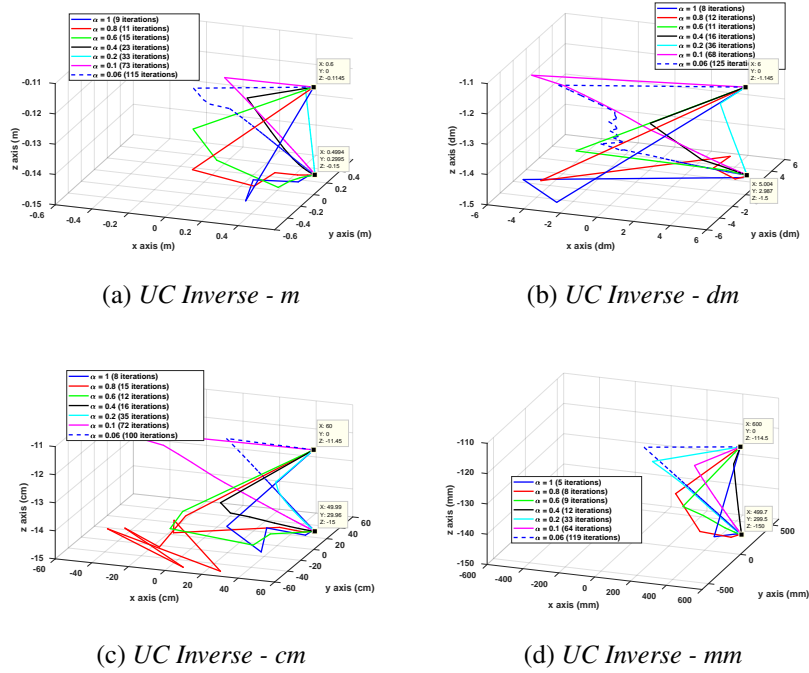


Figure 4.48: Behavior of the trajectories of the end-effector for Motion 1 of the 4DoF robot when varying the units while using the UC Inverse, for multiple values of  $\alpha$ .

• 4-DoF Robot using the MX Generalized Inverse

The last generalized inverse used was the Mixed inverse to the same Motion 1 of the 4DoF robot. As we explained at the beginning of this section, the *MX* inverse reduces to the *MP* inverse. Indeed, we observe the exact same results obtained for the *MP*. Table 4.40, and Figures 4.31 and 4.32 below summarize those results for the *MX* inverse.

Inverse used	MX Inverse			
Results for:	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 90 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.6000 \\ 0 \\ -0.1145 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 6.000 \\ 0 \\ -1.145 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 60.00 \\ 0 \\ -11.45 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 600.0 \\ 0 \\ 114.5 \\ -90 \\ 0 \\ 0 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} 0.5 \\ 0.3 \\ -0.15 \\ -85 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 5 \\ 3 \\ -1.5 \\ -85 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 50 \\ 30 \\ -15 \\ -85 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 500 \\ 300 \\ -150 \\ -85 \\ 0 \\ 0 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} 0.5000 \\ 0.3000 \\ -0.1500 \\ -85.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 4.9997 \\ 2.9998 \\ -1.5000 \\ -85.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 49.9970 \\ 29.9984 \\ -15.0000 \\ -85.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 499.9699 \\ 299.9841 \\ -150.0000 \\ -85.0000 \\ 0 \\ 0 \end{bmatrix}$
Position Error	0.000034m	0.000341dm	0.003407cm	0.034073mm
Orientation Error	0.000000°	0.000000°	0.000000°	0.000000°
Iterations	5	5	5	5
Computation time	0.071s	0.071s	0.073s	0.079s

Table 4.40: *Experimental results obtained for Motion 1 of the 4DoF robot with  $\alpha = 1$  and MX Inverse.*

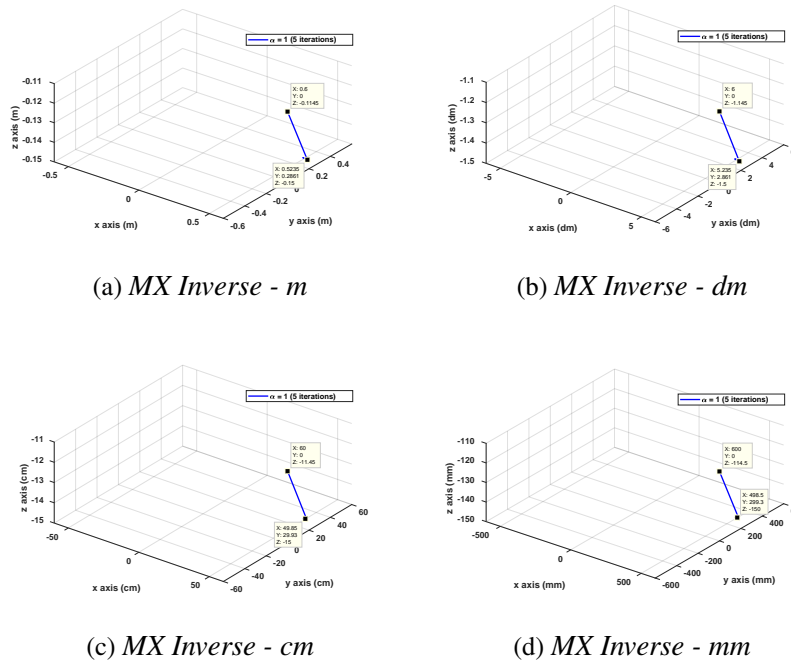


Figure 4.49: Behavior of the trajectories of the end-effector for Motion 1 of the 4DoF robot when varying the units while using the MX Inverse and  $\alpha = 1$ .

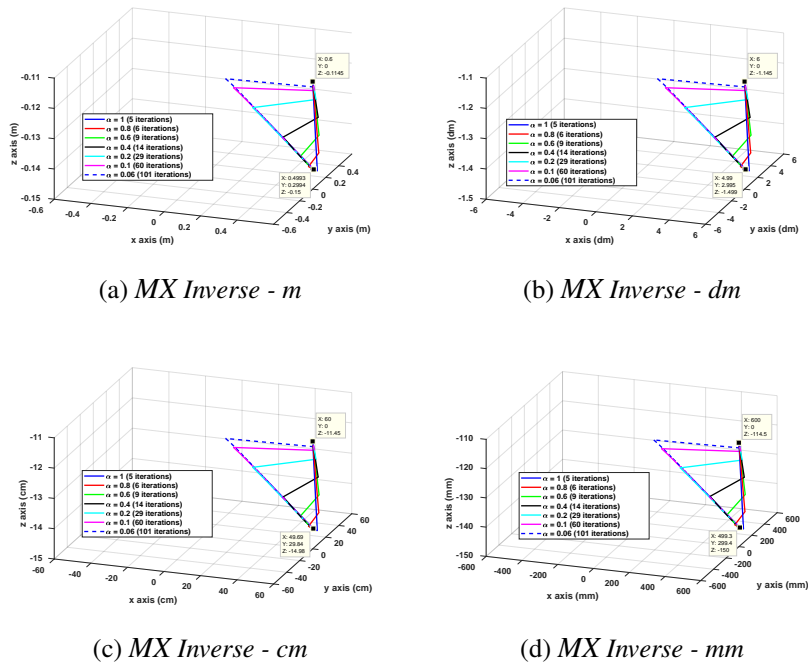


Figure 4.50: Behavior of the trajectories of the end-effector for Motion 1 of the 4DoF robot when varying the units while using the MX Inverse, for multiple values of  $\alpha$ .

## 4-DoF using Motions 2 and 3

- 4-DoF Robot using the MP Generalized Inverse

Next, we repeated the tests above – i.e. to investigate: 1) the effects of changing units, from  $m$  to  $dm$ ,  $cm$  and finally to  $mm$ , on the trajectory of the end-effector; and 2) the effects of different attenuation parameters on the same trajectories – for two additional and arbitrary motions of the 4-DoF robot: Motion 2 and Motion 3.

Table 4.41 and Figures 4.51 and 4.52 summarize the results and demonstrate the same conclusions reached above for Motion 1, now with Motion 2 using the  $MP$  inverse. While Table 4.42 and Figures 4.53 and 4.54 summarize the results and demonstrate the same conclusions for Motion 3 also using the  $MP$  inverse.

Inverse used	MP Inverse			
	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ 0 \\ 90 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.3915 \\ 0.4281 \\ -0.1145 \\ -30 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 3.915 \\ 4.281 \\ -1.145 \\ -30 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 39.15 \\ 42.81 \\ -11.45 \\ -30 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 391.5 \\ 428.1 \\ 114.5 \\ -30 \\ 0 \\ 0 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} -0.5 \\ 0.3 \\ 1.7 \\ 57 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -5 \\ 3 \\ 17 \\ 57 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -50 \\ 30 \\ 170 \\ 57 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -500 \\ 300 \\ 1700 \\ 57 \\ 0 \\ 0 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} -0.5000 \\ 0.3000 \\ 1.7000 \\ 57.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -4.9996 \\ 2.9998 \\ 17.0000 \\ 57.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -49.9964 \\ 29.9976 \\ 170.0000 \\ 57.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -500.0000 \\ 300.0000 \\ 170.0000 \\ 57.0000 \\ 0 \\ 0 \end{bmatrix}$
Position Error	0.000044m	0.000439dm	0.004390cm	0.043902mm
Orientation Error	0.000000°	0.000000°	0.000000°	0.000000°
Iterations	9	9	9	9
Computation time	0.130s	0.134s	0.132s	0.135s

Table 4.41: Experimental results obtained for Motion 2 of the 4DoF robot with  $\alpha = 1$  and  $MP$  Inverse.

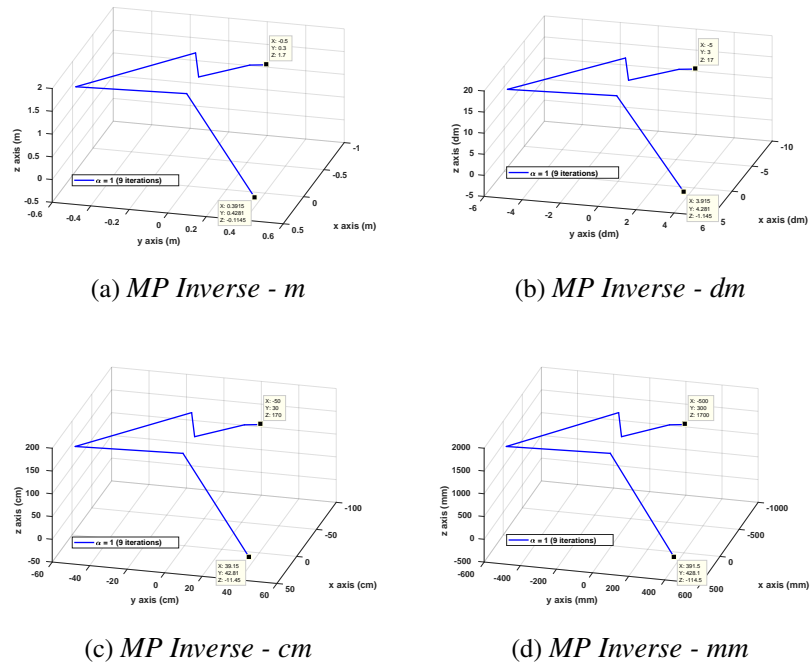


Figure 4.51: Behavior of the trajectories of the end-effector for Motion 2 of the 4DoF robot when varying the units while using the MP Inverse and  $\alpha = 1$ .

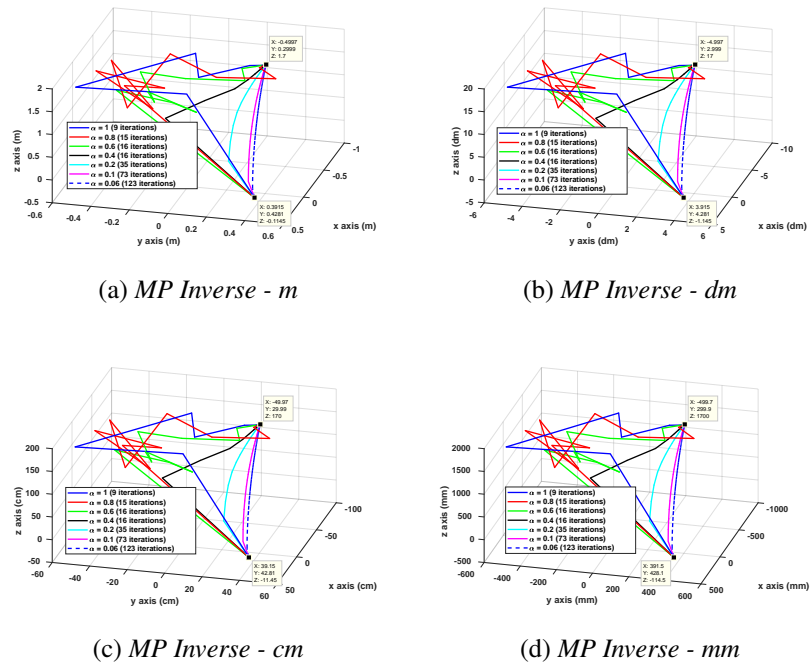


Figure 4.52: Behavior of the trajectories of the end-effector for Motion 2 of the 4DoF robot when varying the units while using the MP Inverse, for multiple values of  $\alpha$ .

Inverse used	MP Inverse			
	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 0.2 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 2 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 20 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 200 \\ 90 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.6 \\ 0 \\ -0.3145 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 6 \\ 0 \\ -3.145 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 60 \\ 0 \\ -31.45 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 600 \\ 0 \\ -314.5 \\ -90 \\ 0 \\ 0 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} 0.496 \\ -0.18 \\ 0.45 \\ 77 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 4.96 \\ -1.8 \\ 4.5 \\ 77 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 49.6 \\ -18 \\ 45 \\ 77 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 496 \\ -180 \\ 450 \\ 77 \\ 0 \\ 0 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} 0.4960 \\ -0.1800 \\ 0.4500 \\ 77.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 4.9598 \\ -1.8000 \\ 4.5000 \\ 77.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 49.5978 \\ -18.0001 \\ 45.0000 \\ 77.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 495.9784 \\ -180.0009 \\ 450.0000 \\ 77.0000 \\ 0 \\ 0 \end{bmatrix}$
Position Error	0.000022m	0.000216dm	0.002162cm	0.021624mm
Orientation Error	0.000000°	0.000000°	0.000000°	0.000000°
Iterations	7	7	7	7
Computation time	0.115s	0.114s	0.106s	0.109s

Table 4.42: Experimental results obtained for Motion 3 of the 4DoF robot with  $\alpha = 1$  and MP Inverse.

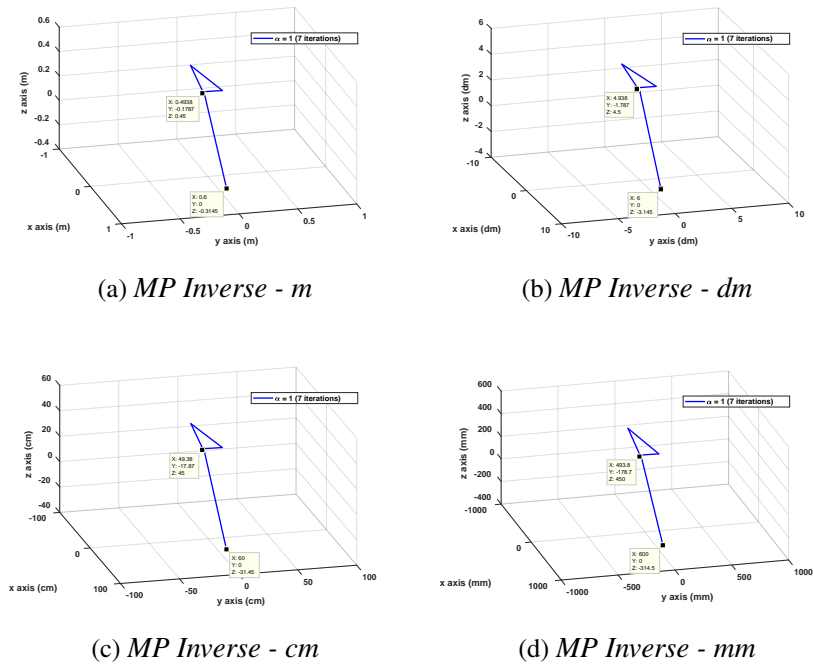


Figure 4.53: Behavior of the trajectories of the end-effector for Motion 3 of the 4DoF robot when varying the units while using the MP Inverse and  $\alpha = 1$ .

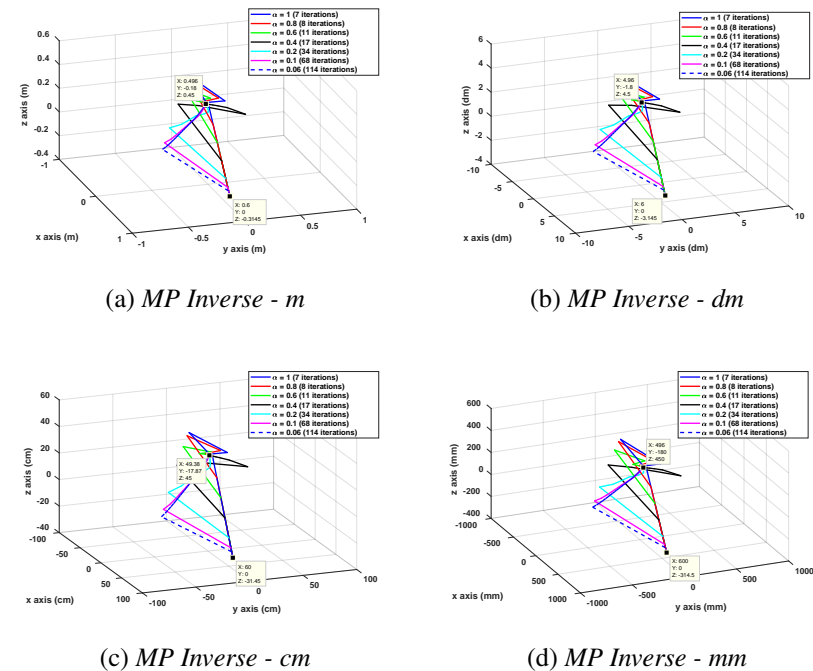


Figure 4.54: Behavior of the trajectories of the end-effector for Motion 3 of the 4DoF robot when varying the units while using the MP Inverse, for multiple values of  $\alpha$ .

• 4-DoF Robot using the UC Generalized Inverse



Once again, the Unit Consistency inverse was applied to these two additional motions of the robot and the results are presented in Table 4.43 and Figures 4.55 and 4.56 for Motion 2, and Table 4.44 and Figures 4.57 and 4.58 for Motion 3. Here, unlike what happened with Motion 1, the *UC* inverse does provide consistent paths in the presence of unit changes for Motion 2, while it still fails to do so for Motion 3. The reason for the *UC* inverse to succeed in the specific case of Motion 2 is likely to be related to specific values of the angles in the path (e.g. exact or close to 0 or 90 degree angles that could lead to *Null* terms in the Jacobian matrix), but a formal proof must be developed in the future.

Inverse used	UC Inverse			
Results for:	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ 0 \\ 90 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.3915 \\ 0.4281 \\ -0.1145 \\ -30 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 3.915 \\ 4.281 \\ -1.145 \\ -30 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 39.15 \\ 42.81 \\ -11.45 \\ -30 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 391.5 \\ 428.1 \\ 114.5 \\ -30 \\ 0 \\ 0 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} -0.5 \\ 0.3 \\ 1.7 \\ 57 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -5 \\ 3 \\ 17 \\ 57 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -50 \\ 30 \\ 170 \\ 57 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -500 \\ 300 \\ 1700 \\ 57 \\ 0 \\ 0 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} -0.5000 \\ 0.3000 \\ 1.7000 \\ 57.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -4.9996 \\ 2.9998 \\ 17.0000 \\ 57.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -49.9964 \\ 29.9976 \\ 170.0000 \\ 57.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -500.0000 \\ 300.0000 \\ 170.0000 \\ 57.0000 \\ 0 \\ 0 \end{bmatrix}$
Position Error	0.000044m	0.000439dm	0.004390cm	0.043902mm
Orientation Error	0.000000°	0.000000°	0.000000°	0.000000°
Iterations	9	9	9	9
Computation time	0.145s	0.147s	0.140s	0.139s

Table 4.43: Experimental results obtained for Motion 2 of the 4DoF robot with  $\alpha = 1$  and UC Inverse.

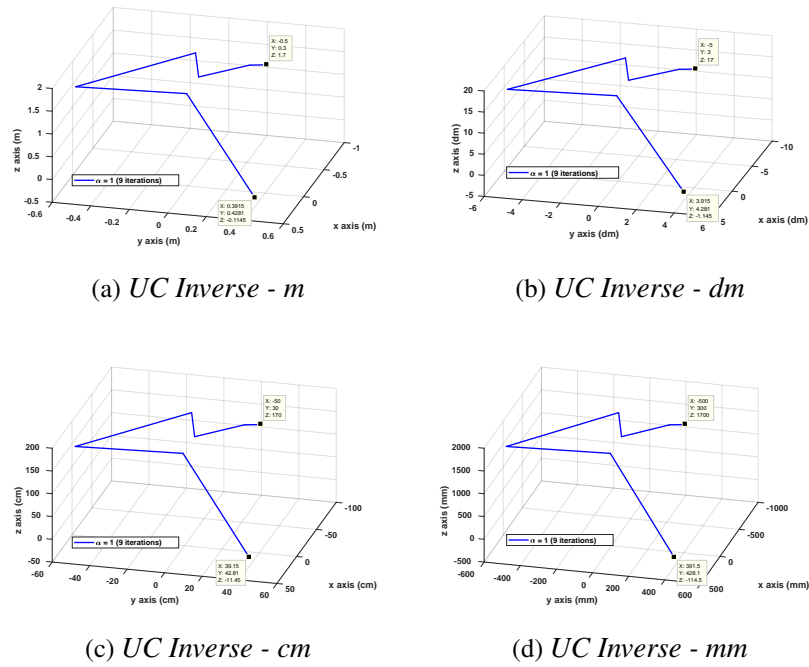


Figure 4.55: Behavior of the trajectories of the end-effector for Motion 2 of the 4DoF robot when varying the units while using the UC Inverse and  $\alpha = 1$ .

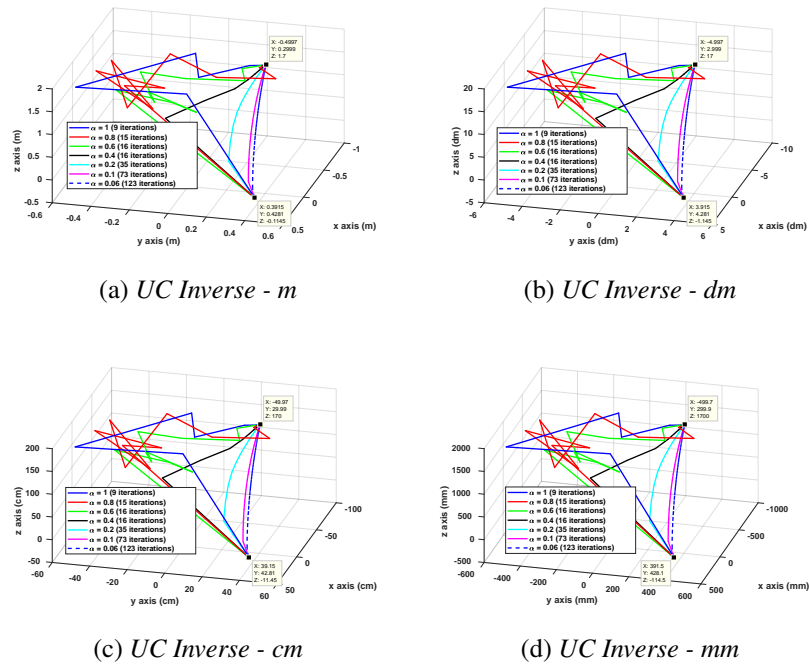


Figure 4.56: Behavior of the trajectories of the end-effector for Motion 2 of the 4DoF robot when varying the units while using the UC Inverse, for multiple values of  $\alpha$ .

Inverse used	UC Inverse			
	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 0.2 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 2 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 20 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 200 \\ 90 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.6 \\ 0 \\ -0.3145 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 6 \\ 0 \\ -3.145 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 60 \\ 0 \\ -31.45 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 600 \\ 0 \\ -314.5 \\ -90 \\ 0 \\ 0 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} 0.496 \\ -0.18 \\ 0.45 \\ 77 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 4.96 \\ -1.8 \\ 4.5 \\ 77 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 49.6 \\ -18 \\ 45 \\ 77 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 496 \\ -180 \\ 450 \\ 77 \\ 0 \\ 0 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} 0.4955 \\ -0.1798 \\ 0.4500 \\ 77.0741 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 4.9581 \\ -1.8020 \\ 4.5000 \\ 76.9554 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 49.5848 \\ -17.9977 \\ 45.0000 \\ 77.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 495.9580 \\ -180.1348 \\ 450.0000 \\ 76.9831 \\ 0 \\ 0 \end{bmatrix}$
Position Error	0.000491m	0.002748dm	0.015327cm	0.141211mm
Orientation Error	0.024701°	0.014858°	0.000000°	0.005639°
Iterations	9	11	7	7
Computation time	0.153s	0.179s	0.112s	0.118s

Table 4.44: *Experimental results obtained for Motion 3 of the 4DoF robot with  $\alpha = 1$  and UC Inverse.*

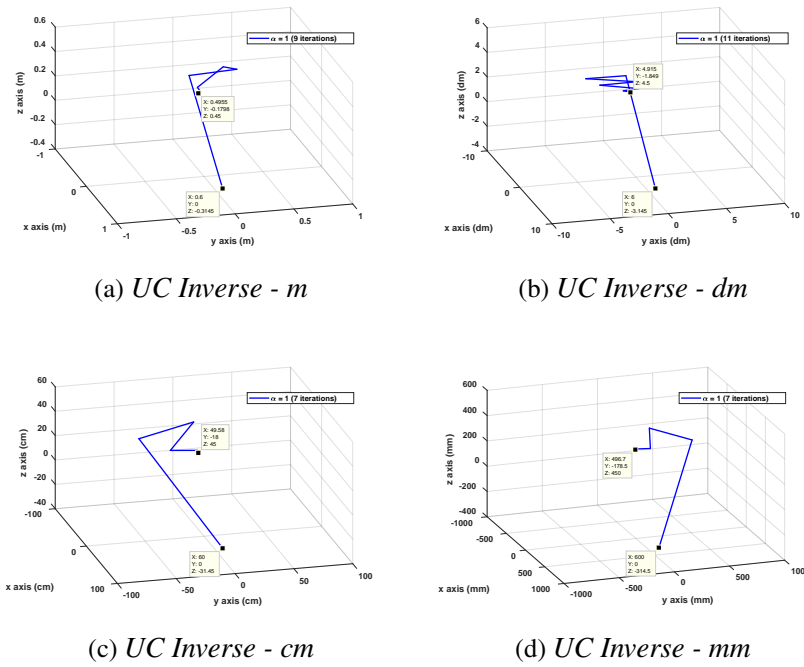


Figure 4.57: Behavior of the trajectories of the end-effector for Motion 3 of the 4DoF robot when varying the units while using the UC Inverse and  $\alpha = 1$ .

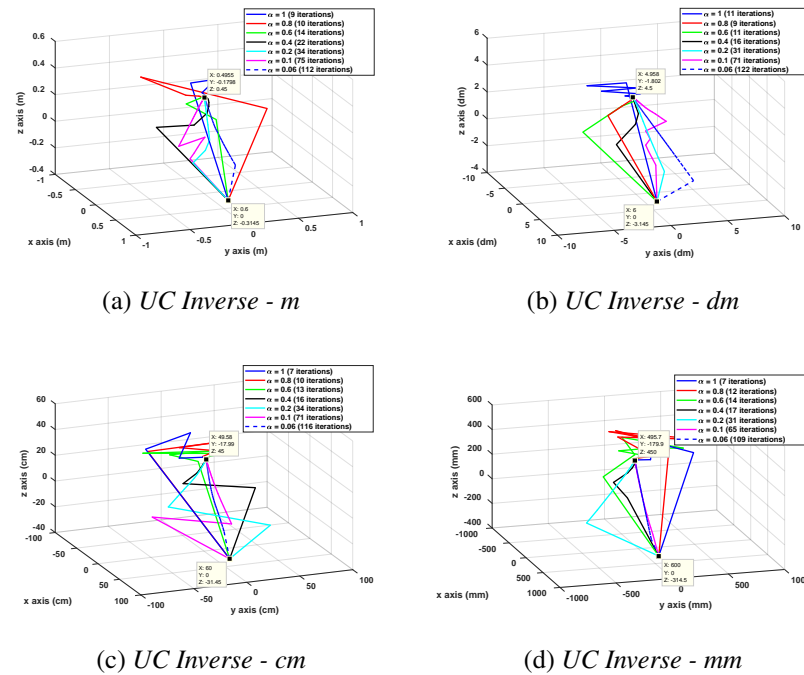


Figure 4.58: Behavior of the trajectories of the end-effector for Motion 3 of the 4DoF robot when varying the units while using the UC Inverse, for multiple values of  $\alpha$ .

• 4-DoF Robot using the MX Generalized Inverse

Finally, the Mixed inverse was also applied to the same two additional motions of the robot and the results are presented in Table 4.45 and Figures 4.59 and 4.60 for Motion 2, and Table 4.46 and Figures 4.61 and 4.62 for Motion 3. As it was the case with Motion 1, with Motions 2 and 3 the  $MX$  inverse also works consistently despite the units or the attenuation factor.

Inverse used	MX Inverse				
	Results for:	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ 0 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 30 \\ 30 \\ 0 \\ 90 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.3915 \\ 0.4281 \\ -0.1145 \\ -30 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 3.915 \\ 4.281 \\ -1.145 \\ -30 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 39.15 \\ 42.81 \\ -11.45 \\ -30 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 391.5 \\ 428.1 \\ 114.5 \\ -30 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 391.5 \\ 428.1 \\ 114.5 \\ -30 \\ 0 \\ 0 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} -0.5 \\ 0.3 \\ 1.7 \\ 57 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -5 \\ 3 \\ 17 \\ 57 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -50 \\ 30 \\ 170 \\ 57 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -500 \\ 300 \\ 1700 \\ 57 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -500 \\ 300 \\ 1700 \\ 57 \\ 0 \\ 0 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} -0.5000 \\ 0.3000 \\ 1.7000 \\ 57.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -4.9996 \\ 2.9998 \\ 17.0000 \\ 57.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -49.9964 \\ 29.9976 \\ 170.0000 \\ 57.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 499.9993 \\ 299.9995 \\ -150.0000 \\ -85.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 499.9993 \\ 299.9995 \\ -150.0000 \\ -85.0000 \\ 0 \\ 0 \end{bmatrix}$
Position Error	0.000044m	0.000439dm	0.004390cm	0.043902mm	0.043902mm
Orientation Error	0.000000°	0.000000°	0.000000°	0.000000°	0.000000°
Iterations	9	9	9	9	9
Computation time	0.138s	0.133s	0.139s	0.142s	0.142s

Table 4.45: Experimental results obtained for Motion 2 of the 4DoF robot with  $\alpha = 1$  and  $MX$  Inverse.

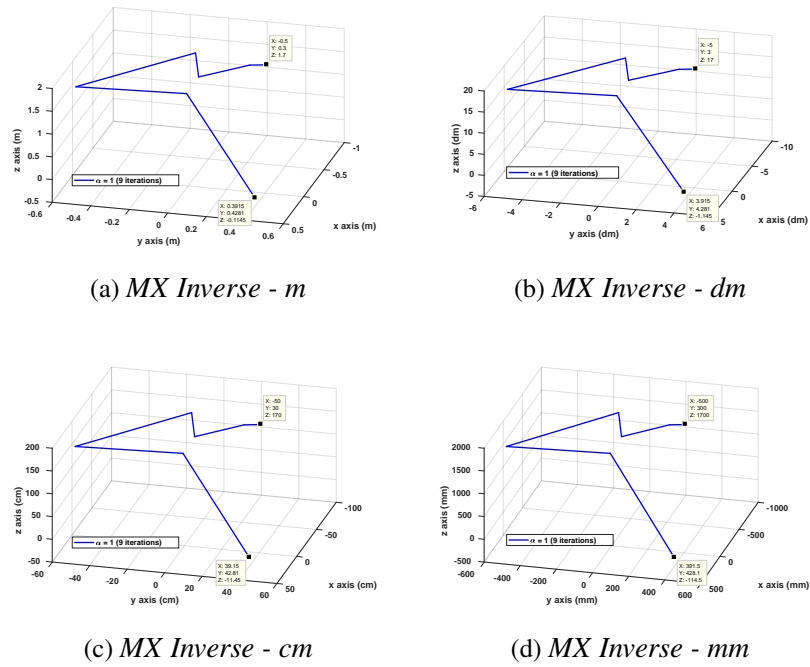


Figure 4.59: Behavior of the trajectories of the end-effector for Motion 2 of the 4DoF robot when varying the units while using the MX Inverse and  $\alpha = 1$ .

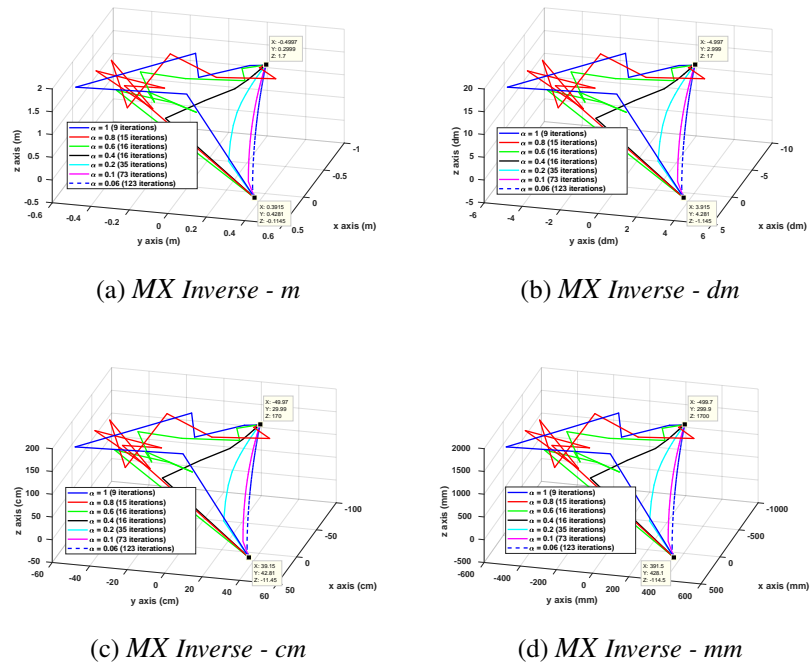


Figure 4.60: Behavior of the trajectories of the end-effector for Motion 2 of the 4DoF robot when varying the units while using the MX Inverse, for multiple values of  $\alpha$ .

Inverse used	MX Inverse			
	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 0.2 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 2 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 20 \\ 90 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 0 \\ 0 \\ 200 \\ 90 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.6 \\ 0 \\ -0.3145 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 6 \\ 0 \\ -3.145 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 60 \\ 0 \\ -31.45 \\ -90 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 600 \\ 0 \\ -314.5 \\ -90 \\ 0 \\ 0 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} 0.496 \\ -0.18 \\ 0.45 \\ 77 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 4.96 \\ -1.8 \\ 4.5 \\ 77 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 49.6 \\ -18 \\ 45 \\ 77 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 496 \\ -180 \\ 450 \\ 77 \\ 0 \\ 0 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} 0.4960 \\ -0.1800 \\ 0.4500 \\ 77.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 4.9598 \\ -1.8000 \\ 4.5000 \\ 77.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 49.5978 \\ -18.0001 \\ 45.0000 \\ 77.0000 \\ 0 \\ 0 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 495.9784 \\ -180.0009 \\ 450.0000 \\ 77.0000 \\ 0 \\ 0 \end{bmatrix}$
Position Error	0.000022m	0.000216dm	0.002162cm	0.021624mm
Orientation Error	0.000000°	0.000000°	0.000000°	0.000000°
Iterations	7	7	7	7
Computation time	0.115s	0.114s	0.106s	0.109s

Table 4.46: Experimental results obtained with the 4DoF with  $\alpha = 1$  and MX Inverse.

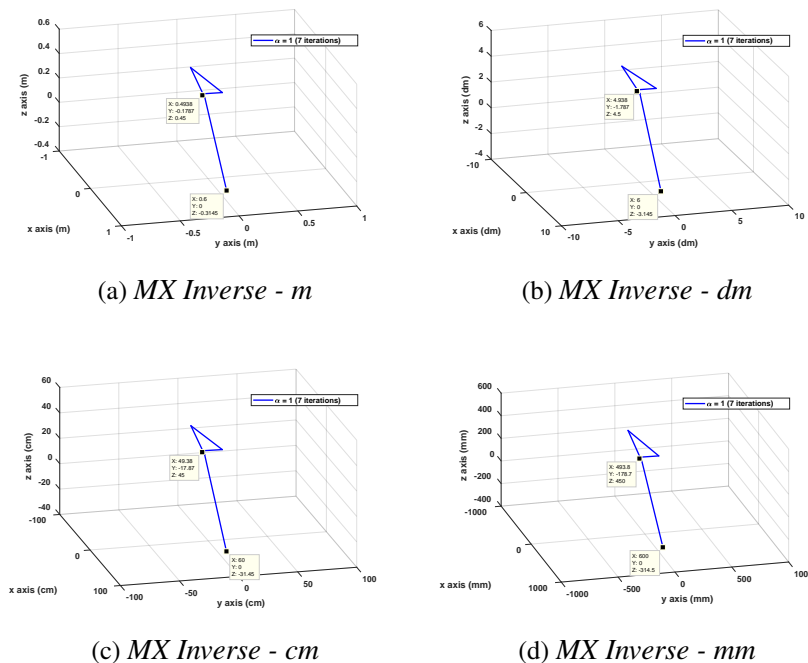


Figure 4.61: Behavior of the trajectories of the end-effector for Motion 3 of the 4DoF robot when varying the units while using the MX Inverse and  $\alpha = 1$ .

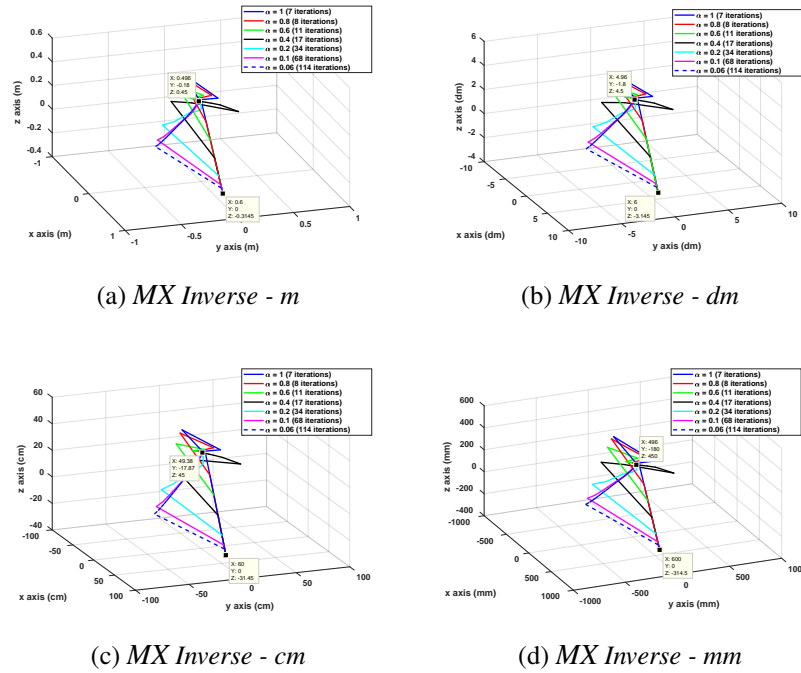


Figure 4.62: Behavior of the trajectories of the end-effector for Motion 3 of the 4DoF robot when varying the units while using the MX Inverse, for multiple values of  $\alpha$ .

#### 4.2.2.3 6 DoF Serial Robot

##### Matrix block-partition for the application of the Mixed Inverse (MX) to the 6DoF (RRPRRR + End-Effector Pose)

Here, an inspection of the two early rotations of the 6DoF serial manipulator with a RRPRRR configuration shows that both  $\Theta_1$  and  $\Theta_2$  affect the linear (prismatic) joint  $d_3$ , which in turn affects  $X$ ,  $Y$ , and  $Z$ . So, the block partitioning of the MX inverse applied to the Jacobian of this 6DoF robot becomes as follows.

Given the Jacobian,

$$J = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix} = \begin{bmatrix} \frac{\partial X}{\partial \theta_1} & \frac{\partial X}{\partial \theta_2} & \frac{\partial X}{\partial d_3} & \frac{\partial X}{\partial \theta_4} & \frac{\partial X}{\partial \theta_5} & \frac{\partial X}{\partial \theta_6} \\ \frac{\partial Y}{\partial \theta_1} & \frac{\partial Y}{\partial \theta_2} & \frac{\partial Y}{\partial d_3} & \frac{\partial Y}{\partial \theta_4} & \frac{\partial Y}{\partial \theta_5} & \frac{\partial Y}{\partial \theta_6} \\ \frac{\partial Z}{\partial \theta_1} & \frac{\partial Z}{\partial \theta_2} & \frac{\partial Z}{\partial d_3} & \frac{\partial Z}{\partial \theta_4} & \frac{\partial Z}{\partial \theta_5} & \frac{\partial Z}{\partial \theta_6} \\ \frac{\partial R_o}{\partial \theta_1} & \frac{\partial R_o}{\partial \theta_2} & \frac{\partial R_o}{\partial d_3} & \frac{\partial R_o}{\partial \theta_4} & \frac{\partial R_o}{\partial \theta_5} & \frac{\partial R_o}{\partial \theta_6} \\ \frac{\partial P_i}{\partial \theta_1} & \frac{\partial P_i}{\partial \theta_2} & \frac{\partial P_i}{\partial d_3} & \frac{\partial P_i}{\partial \theta_4} & \frac{\partial P_i}{\partial \theta_5} & \frac{\partial P_i}{\partial \theta_6} \\ \frac{\partial Y_a}{\partial \theta_1} & \frac{\partial Y_a}{\partial \theta_2} & \frac{\partial Y_a}{\partial d_3} & \frac{\partial Y_a}{\partial \theta_4} & \frac{\partial Y_a}{\partial \theta_5} & \frac{\partial Y_a}{\partial \theta_6} \end{bmatrix}$$

and the fact that the variables requiring consistency with respect to the change of units



need to be in  $W$  – so it is handled by the  $UC$  inverse – we have:

$$\begin{aligned}
 w &= \begin{bmatrix} \frac{\partial X}{\partial \theta_1} & \frac{\partial X}{\partial \theta_2} & \frac{\partial X}{\partial d_3} \\ \frac{\partial Y}{\partial \theta_1} & \frac{\partial Y}{\partial \theta_2} & \frac{\partial Y}{\partial d_3} \\ \frac{\partial Z}{\partial \theta_1} & \frac{\partial Z}{\partial \theta_2} & \frac{\partial Z}{\partial d_3} \end{bmatrix} \text{ is a } 3 \times 3 \text{ matrix} \\
 X &= \begin{bmatrix} \frac{\partial X}{\partial \theta_4} & \frac{\partial X}{\partial \theta_5} & \frac{\partial X}{\partial \theta_6} \\ \frac{\partial Y}{\partial \theta_4} & \frac{\partial Y}{\partial \theta_5} & \frac{\partial Y}{\partial \theta_6} \\ \frac{\partial Z}{\partial \theta_4} & \frac{\partial Z}{\partial \theta_5} & \frac{\partial Z}{\partial \theta_6} \end{bmatrix} \text{ is a } 3 \times 3 \text{ matrix} \\
 Y &= \begin{bmatrix} \frac{\partial R_o}{\partial \theta_1} & \frac{\partial R_o}{\partial \theta_2} & \frac{\partial R_o}{\partial d_3} \\ \frac{\partial P_i}{\partial \theta_1} & \frac{\partial P_i}{\partial \theta_2} & \frac{\partial P_i}{\partial d_3} \\ \frac{\partial Y_a}{\partial \theta_1} & \frac{\partial Y_a}{\partial \theta_2} & \frac{\partial Y_a}{\partial d_3} \end{bmatrix} \text{ is a } 3 \times 3 \text{ matrix} \\
 Z &= \begin{bmatrix} \frac{\partial R_o}{\partial \theta_4} & \frac{\partial R_o}{\partial \theta_5} & \frac{\partial R_o}{\partial \theta_6} \\ \frac{\partial P_i}{\partial \theta_4} & \frac{\partial P_i}{\partial \theta_5} & \frac{\partial P_i}{\partial \theta_6} \\ \frac{\partial Y_a}{\partial \theta_4} & \frac{\partial Y_a}{\partial \theta_5} & \frac{\partial Y_a}{\partial \theta_6} \end{bmatrix} \text{ is a } 3 \times 3 \text{ matrix}
 \end{aligned}$$

Then, the  $Mx$  inverse can be computed as follows:

$$J^{-M} = \begin{bmatrix} (W - XZ^{-P}Y)^{-U} & -W^{-U}X(Z - YW^{-U}X)^{-P} \\ -Z^{-P}Y(W - XZ^{-P}Y)^{-U} & (Z - YW^{-U}X)^{-P} \end{bmatrix}$$

The resulting  $J^{-M}$  is a  $6 \times 6$  matrix. Here, it is important to mention that this is a square Jacobian, and whenever it is non-singular we will have  $J^{-1} = J^{-P} = J^{-U} = J^{-M}$  which makes the use of generalized inverses unnecessary unless the 6DoF robotic arm reaches a singular configuration in its joint space (i.e. the  $\det(J) = 0$ ).

Since for this experiment we considered three arbitrary motions that happened to contain no singular configurations, all three inverses produced the exact same paths: all consistent despite changes in units, and all affected in the same way by the attenuation parameter. So, the results for all three motions are presented together.

## 6-DoF using Motions 1, 2 and 3

### • 6-DoF Robot using the MP Generalized Inverse

First, we applied the Moore-Penrose generalized inverse to Motions 1, 2 and 3 of the 6DoF robot. Tables 4.47, 4.48 and 4.49 summarize the overall results for Motions 1, 2 and 3 respectively. The tables present the input parameters provided to the algorithm versus the number of iterations and final errors in the position of the end-effector with

respect to the desired position. It is important to mention that the attenuation parameter was kept constant and equal to 1 for all three motions, unless for the third experiment for all three motions, where the effects of  $\alpha$  on the  $MP$  generalized inverse was investigated.

Inverse used	MP Inverse			
Results for:	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 10 \\ 20 \\ 0.5 \\ 30 \\ 40 \\ 50 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 10 \\ 20 \\ 5 \\ 30 \\ 40 \\ 50 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 10 \\ 20 \\ 50 \\ 30 \\ 40 \\ 50 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 10 \\ 20 \\ 500 \\ 30 \\ 40 \\ 50 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.1501 \\ 0.1714 \\ 0.4743 \\ -73.6939 \\ -23.7725 \\ 54.6519 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 1.5019 \\ 1.71418 \\ 4.74347 \\ -73.6939 \\ -23.7725 \\ 54.6519 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 15.019 \\ 17.1418 \\ 47.4347 \\ -73.6939 \\ -23.7725 \\ 54.6519 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 150.19 \\ 171.418 \\ 474.347 \\ -73.6939 \\ -23.7725 \\ 54.6519 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} -0.25 \\ 0.36 \\ 0.43 \\ -10 \\ 25 \\ -65 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -2.5 \\ 3.6 \\ 4.3 \\ -10 \\ 25 \\ -65 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -25 \\ 36 \\ 43 \\ -10 \\ 25 \\ -65 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -250 \\ 360 \\ 430 \\ -10 \\ 25 \\ -65 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} -0.2500 \\ 0.3600 \\ 0.43000 \\ -9.9996 \\ 24.9997 \\ -65.0003 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -2.5000 \\ 3.6000 \\ 4.3000 \\ -9.9996 \\ 24.9997 \\ -65.0003 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -25.0000 \\ 36.0000 \\ 43.0000 \\ -9.9996 \\ 24.9997 \\ -65.0003 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -250.0000 \\ 360.0001 \\ 429.9999 \\ -9.9996 \\ 24.9997 \\ -65.0003 \end{bmatrix}$
Position Error	0.000575m	0.000575dm	0.000575dm	0.000584mm
Orientation Error	0.000326°	0.000326°	0.000326°	0.000326°
Iterations	15	15	15	15
Computation time	0.258s	0.272s	0.269s	0.271s

Table 4.47: Experimental results obtained for Motion 1 of the 6DoF robot with  $\alpha = 1$  and MP Inverse.

Inverse used	MP Inverse			
Results for:	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 0.7 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 7 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 70 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 700 \\ 45 \\ 35 \\ 60 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.1260 \\ 0.1985 \\ 0.6819 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 1.2600 \\ 1.9851 \\ 6.8198 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 12.6007 \\ 19.8516 \\ 68.1981 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 126.0078 \\ 198.5166 \\ 681.9813 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} 0.42 \\ 0.16 \\ 0.21 \\ -60 \\ 11 \\ 66 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 4.2 \\ 1.6 \\ 2.1 \\ -60 \\ 11 \\ 66 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 42 \\ 16 \\ 21 \\ -60 \\ 11 \\ 66 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 420 \\ 160 \\ 210 \\ -60 \\ 11 \\ 66 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} 0.4200 \\ 0.1600 \\ 0.2100 \\ -60.0000 \\ 10.9999 \\ 66.0001 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 4.2000 \\ 1.6000 \\ 2.1000 \\ -60.0000 \\ 10.9999 \\ 66.0001 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 42.0000 \\ 16.0000 \\ 21.0000 \\ -60.0000 \\ 10.9999 \\ 66.0001 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 420.0000 \\ 160.0000 \\ 210.0000 \\ -60.0000 \\ 10.9999 \\ 66.0001 \end{bmatrix}$
Position Error	0.000140m	0.000140dm	0.000140cm	0.000141mm
Orientation Error	0.000063°	0.000063°	0.000063°	0.000063°
Iterations	10	10	10	10
Computation time	0.155s	0.157s	0.166s	0.166s

Table 4.48: Experimental results obtained for Motion 2 of the 6DoF robot with  $\alpha = 1$  and MP Inverse.

Inverse used	MP Inverse			
Results for:	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 0.7 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 7 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 70 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 700 \\ 45 \\ 35 \\ 60 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.1260 \\ 0.1985 \\ 0.6819 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 1.2600 \\ 1.9851 \\ 6.8198 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 12.6007 \\ 19.8516 \\ 68.1981 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 126.0078 \\ 198.5166 \\ 681.9813 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} -0.61 \\ -0.04 \\ 0.02 \\ 1 \\ 20 \\ 23 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -6.1 \\ -0.4 \\ 0.2 \\ 1 \\ 20 \\ 23 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -61 \\ -4 \\ 2 \\ 1 \\ 20 \\ 23 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -610 \\ -40 \\ 20 \\ 1 \\ 20 \\ 23 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} -0.6100 \\ -0.0400 \\ 0.0200 \\ 1.0891 \\ 20.0798 \\ 22.9885 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -6.0997 \\ -0.3998 \\ 0.1999 \\ 1.0891 \\ 20.0798 \\ 22.9885 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -60.9969 \\ -3.9983 \\ 0.1989 \\ 1.0891 \\ 20.0798 \\ 22.9885 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -609.9688 \\ -39.9829 \\ 19.9888 \\ 1.0891 \\ 20.0798 \\ 22.9885 \end{bmatrix}$
Position Error	0.000037m	0.000373m	0.003730cm	0.037297mm
Orientation Error	0.060113°	0.060113°	0.060113°	0.060113°
Iterations	14	14	14	14
Computation time	0.175s	0.172s	0.181s	0.193s

Table 4.49: Experimental results obtained for Motion 3 of the 6DoF robot with  $\alpha = 1$  and MP Inverse.

Next, Figures 4.63, 4.64 and 4.65 show the paths of the end-effector when the units of the linear joint in the 6DoF robot are varied from  $m$  to  $dm$ ,  $cm$  and finally to  $mm$ , still for Motions 1, 2, and 3 respectively. The behavior of the robot is consistent whichever units are used. That is, as pointed out earlier, the  $MP$  generalized inverse for a full-rank Jacobian provides consistent end-effector path despite the units change.

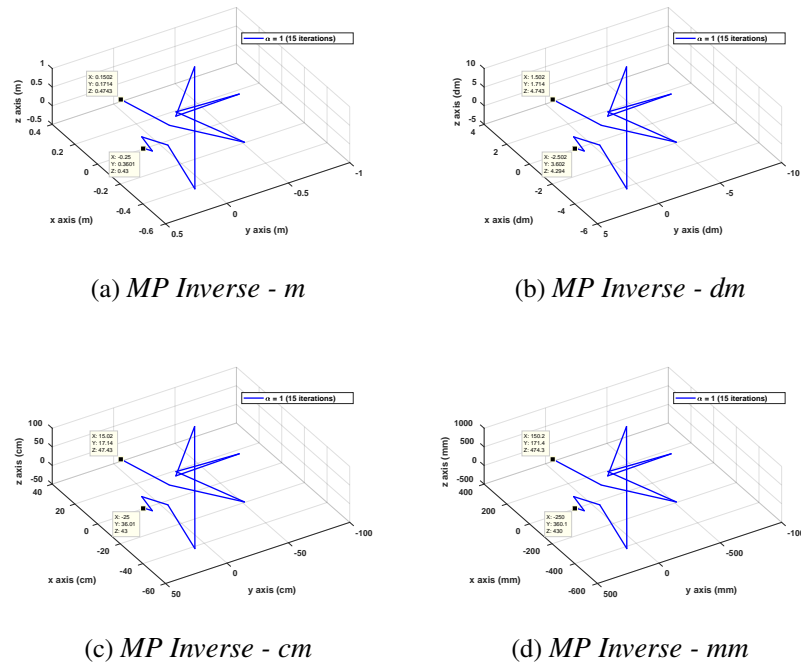


Figure 4.63: Behavior of the trajectories of the end-effector for Motion 1 of the 6DoF robot when varying the units with the MP Inverse for  $\alpha = 1$ .

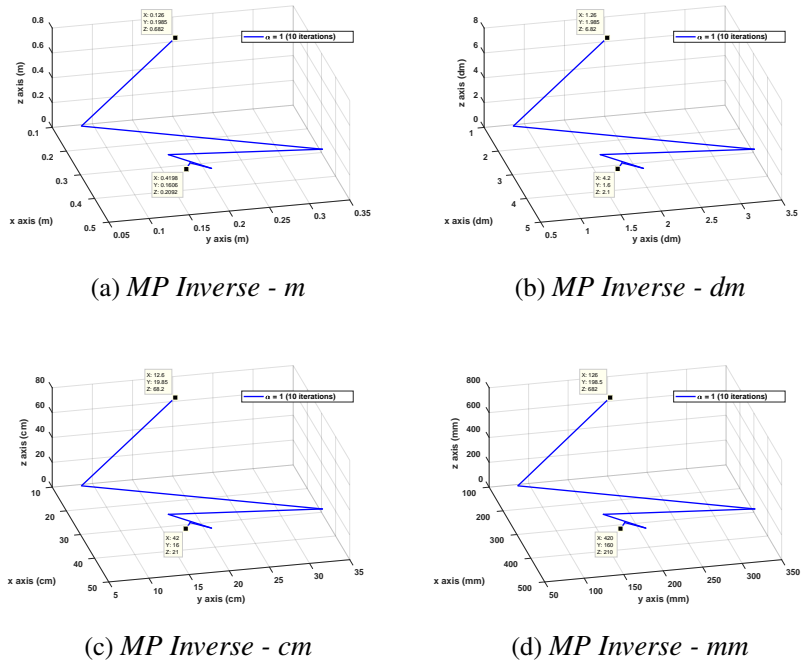


Figure 4.64: Behavior of the trajectories of the end-effector for Motion 2 of the 6DoF robot when varying the units while using the MP Inverse and  $\alpha = 1$ .

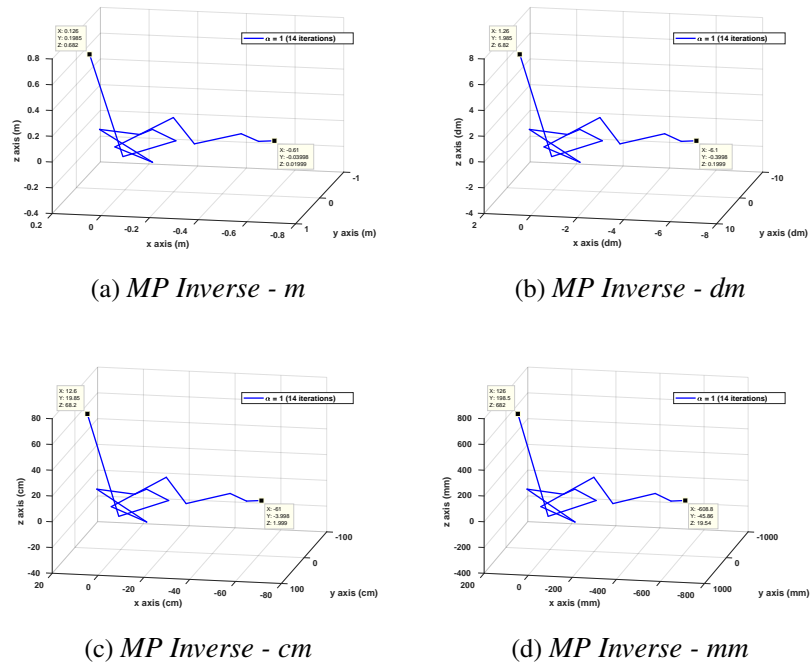


Figure 4.65: Behavior of the trajectories of the end-effector for Motion 3 of the 6DoF robot when varying the units while using the MP Inverse and  $\alpha = 1$ .

Finally, we studied the effects of the attenuation parameter on the path followed by the end-effector under different units, but still using the MP inverse. Figures 4.28, 4.67 and 4.68 show these results for Motions 1, 2 and 3 respectively. As it can be observed, some choices for the attenuation parameter can provide smooth path for any specific unit, but the paths followed by the robot are exactly the same despite the unit used.

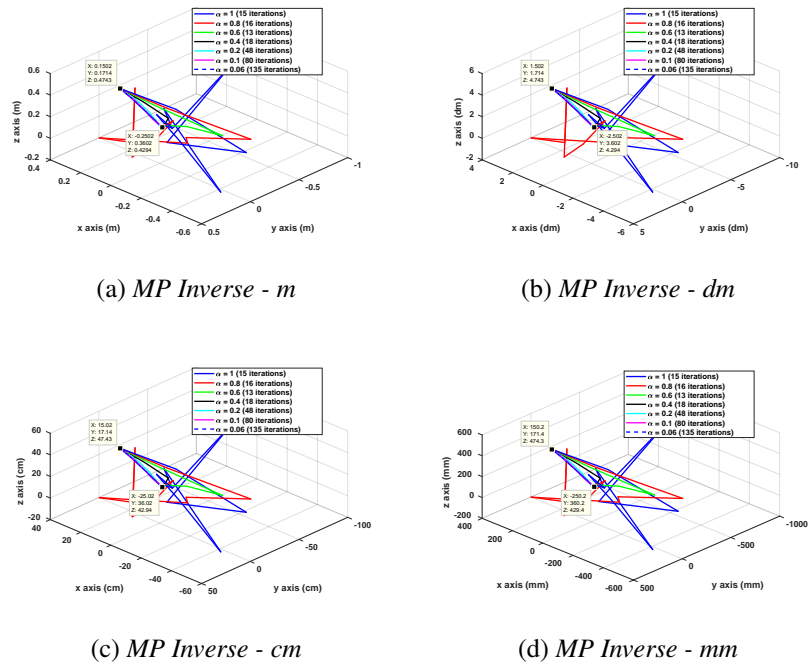


Figure 4.66: Behavior of the trajectories of the end-effector for Motion 1 of the 6DoF robot when varying the units while using the MP Inverse, for multiple values of  $\alpha$ .

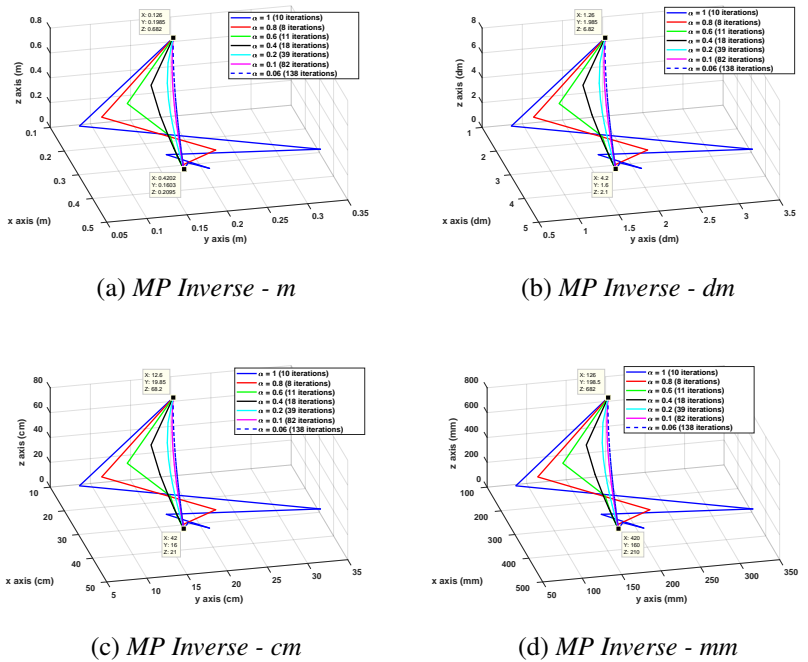


Figure 4.67: Behavior of the trajectories of the end-effector for Motion 2 of the 6DoF robot when varying the units while using the MP Inverse, for multiple values of  $\alpha$ .

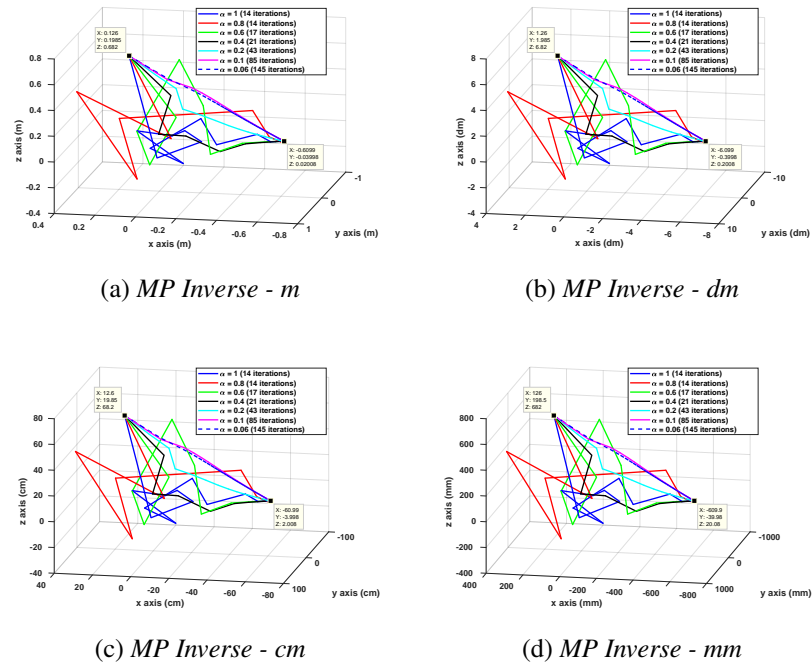


Figure 4.68: Behavior of the trajectories of the end-effector for Motion 3 of the 6DoF robot when varying the units while using the MP Inverse, for multiple values of  $\alpha$ .

- **6-DoF Robot using the UC Generalized Inverse**

Next, we applied the Unit Consistency generalized inverse to all three motions of the 6DoF robot. As before, Tables 4.50, 4.51 and 4.52 summarize the overall results for Motion 1, 2 and 3 respectively under the *UC* inverse. Once again, the attenuation parameter was kept constant and equal to 1, unless specified otherwise.



Inverse used	UC Inverse			
Results for:	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 10 \\ 20 \\ 0.5 \\ 30 \\ 40 \\ 50 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 10 \\ 20 \\ 5 \\ 30 \\ 40 \\ 50 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 10 \\ 20 \\ 50 \\ 30 \\ 40 \\ 50 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 10 \\ 20 \\ 500 \\ 30 \\ 40 \\ 50 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.1501 \\ 0.1714 \\ 0.4743 \\ -73.6939 \\ -23.7725 \\ 54.6519 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 1.5019 \\ 1.7141 \\ 4.7434 \\ -73.6939 \\ -23.7725 \\ 54.6519 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 15.0198 \\ 17.1417 \\ 47.4347 \\ -73.6939 \\ -23.7725 \\ 54.6519 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 150.19 \\ 171.418 \\ 474.347 \\ -73.6939 \\ -23.7725 \\ 54.6519 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} -0.25 \\ 0.36 \\ 0.43 \\ -10 \\ 25 \\ -65 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -2.5 \\ 3.6 \\ 4.3 \\ -10 \\ 25 \\ -65 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -25 \\ 36 \\ 43 \\ -10 \\ 25 \\ -65 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -250 \\ 360 \\ 430 \\ -10 \\ 25 \\ -65 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} -0.2500 \\ 0.3600 \\ 0.43000 \\ -9.9996 \\ 24.9997 \\ -65.0003 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -2.5000 \\ 3.6000 \\ 4.3000 \\ -9.9996 \\ 24.9997 \\ -65.0003 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -25.0000 \\ 36.0000 \\ 43.0000 \\ -9.9996 \\ 24.9997 \\ -65.0003 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -250.0000 \\ 360.0001 \\ 429.9999 \\ -9.9996 \\ 24.9997 \\ -65.0003 \end{bmatrix}$
Position Error	0.000575m	0.000575dm	0.000575dm	0.000584mm
Orientation Error	0.000326°	0.000326°	0.000326°	0.000326°
Iterations	15	15	15	15
Computation time	0.277s	0.270s	0.275s	0.293s

Table 4.50: Experimental results obtained for Motion 1 of the 6DoF robot with  $\alpha = 1$  and UC Inverse.

Inverse used	UC Inverse			
Results for:	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 0.7 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 7 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 70 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 700 \\ 45 \\ 35 \\ 60 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.1260 \\ 0.1985 \\ 0.6819 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 1.2600 \\ 1.9851 \\ 6.8198 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 12.6007 \\ 19.8516 \\ 68.1981 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 126.0078 \\ 198.5166 \\ 681.9813 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} 0.42 \\ 0.16 \\ 0.21 \\ -60 \\ 11 \\ 66 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 4.2 \\ 1.6 \\ 2.1 \\ -60 \\ 11 \\ 66 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 42 \\ 16 \\ 21 \\ -60 \\ 11 \\ 66 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 420 \\ 160 \\ 210 \\ -60 \\ 11 \\ 66 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} 0.4200 \\ 0.1600 \\ 0.2100 \\ -60.0000 \\ 10.9999 \\ 66.0001 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 4.2000 \\ 1.6000 \\ 2.1000 \\ -60.0000 \\ 10.9999 \\ 66.0001 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 42.0000 \\ 16.0000 \\ 21.0000 \\ -60.0000 \\ 10.9999 \\ 66.0001 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 420.0000 \\ 160.0000 \\ 210.0000 \\ -60.0000 \\ 10.9999 \\ 66.0001 \end{bmatrix}$
Position Error	0.000140m	0.000140dm	0.000140cm	0.000141mm
Orientation Error	0.000063°	0.000063°	0.000063°	0.000063°
Iterations	10	10	10	10
Computation time	0.172s	0.175s	0.173s	0.172s

Table 4.51: Experimental results obtained for Motion 2 of the 6DoF robot with  $\alpha = 1$  and UC Inverse.

Inverse used	UC Inverse			
Results for:	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 0.7 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 7 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 70 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 700 \\ 45 \\ 35 \\ 60 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.1260 \\ 0.1985 \\ 0.6819 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 1.2600 \\ 1.9851 \\ 6.8198 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 12.6007 \\ 19.8516 \\ 68.1981 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 126.0078 \\ 198.5166 \\ 681.9813 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} -0.61 \\ -0.04 \\ 0.02 \\ 1 \\ 20 \\ 23 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -6.1 \\ -0.4 \\ 0.2 \\ 1 \\ 20 \\ 23 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -61 \\ -4 \\ 2 \\ 1 \\ 20 \\ 23 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -610 \\ -40 \\ 20 \\ 1 \\ 20 \\ 23 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} -0.6100 \\ -0.0400 \\ 0.0200 \\ 1.0891 \\ 20.0798 \\ 22.9885 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -6.0997 \\ -0.3998 \\ 0.1999 \\ 1.0891 \\ 20.0798 \\ 22.9885 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -60.9969 \\ -3.9983 \\ 0.1989 \\ 1.0891 \\ 20.0798 \\ 22.9885 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -609.9688 \\ -39.9829 \\ 19.9888 \\ 1.0891 \\ 20.0798 \\ 22.9885 \end{bmatrix}$
Position Error	0.000037m	0.000373m	0.003730cm	0.037297mm
Orientation Error	0.060113°	0.060113°	0.060113°	0.060113°
Iterations	14	14	14	14
Computation time	0.191s	0.199s	0.196s	0.181s

Table 4.52: Experimental results obtained for Motion 3 of the 6DoF robot with  $\alpha = 1$  and UC Inverse.

Also as before, Figures 4.69, 4.70 and 4.71 show the paths of the end-effector when the units of the linear joint in the 6DoF robot are varied from  $m$  to  $dm$ ,  $cm$  and finally to  $mm$ , for all three motions using the UC inverse. As expected, the path followed by the robot is exactly the same, no matter what units are used. That is, being a full rank Jacobian makes the UC generalized inverse also numerically equivalent to the actual matrix inverse.

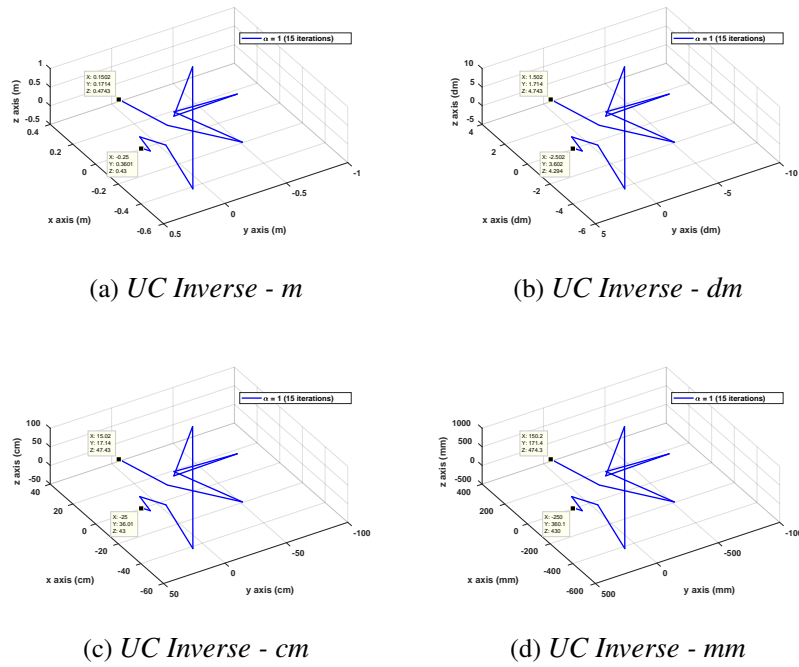


Figure 4.69: Behavior of the trajectories of the end-effector for Motion 1 of the 6DoF robot when varying the units with the UC Inverse for  $\alpha = 1$ .

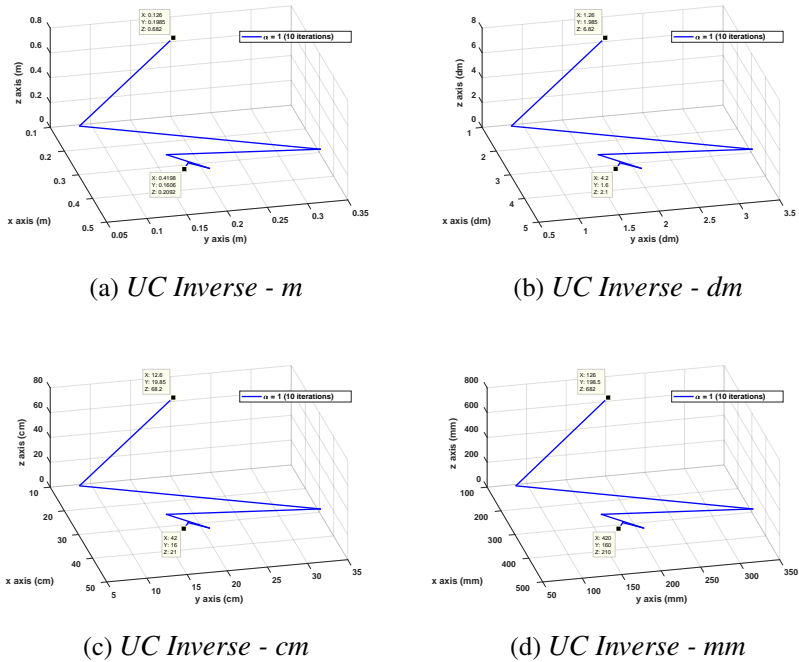


Figure 4.70: Behavior of the trajectories of the end-effector for Motion 2 of the 6DoF robot when varying the units while using the UC Inverse and  $\alpha = 1$ .

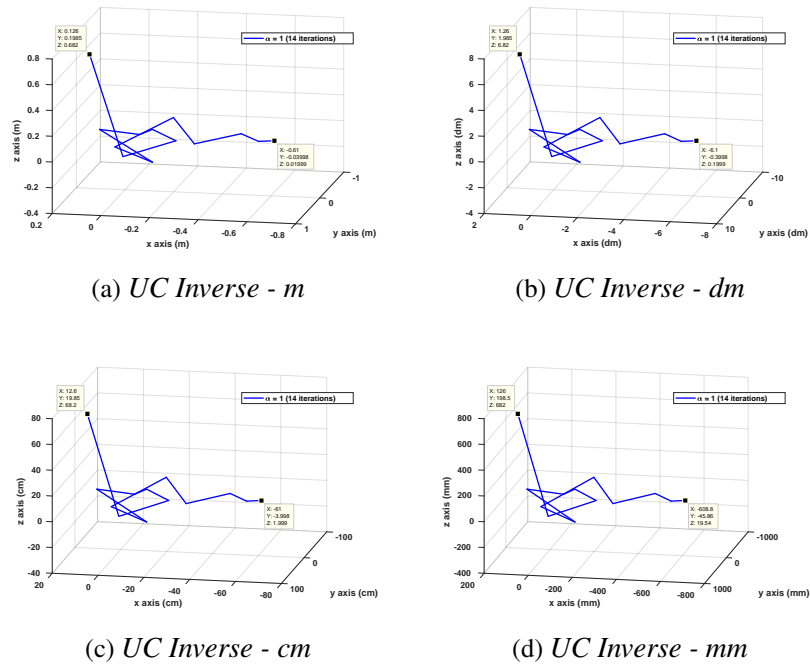


Figure 4.71: Behavior of the trajectories of the end-effector for Motion 3 of the 6DoF robot when varying the units while using the UC Inverse and  $\alpha = 1$ .

Finally, we studied the effects of the attenuation parameter on the path followed by the end-effector under different units while still using the UC inverse. Figures 4.72, 4.73 and 4.74 show these results for Motion 1, 2, and 3 respectively. As also expected, the attenuation parameter behaves as expected, providing smoothness to some choices of units, but all paths are the same despite the unit used.

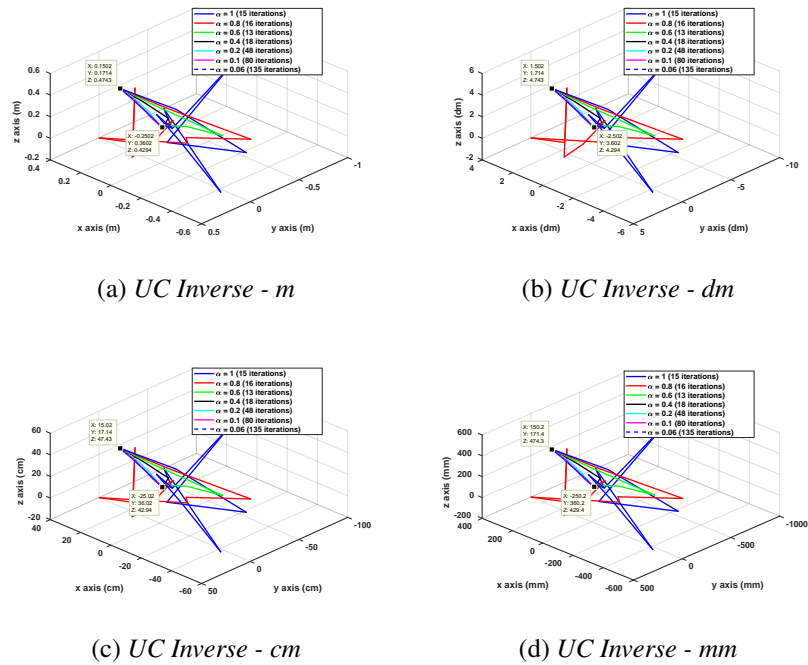


Figure 4.72: Behavior of the trajectories of the end-effector for Motion 1 of the 6DoF robot when varying the units while using the UC Inverse, for multiple values of  $\alpha$ .

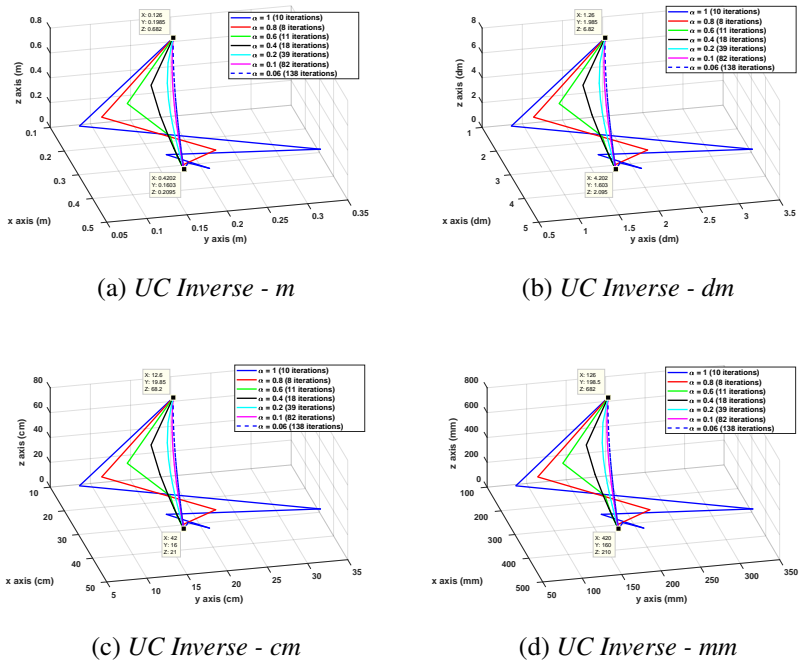


Figure 4.73: Behavior of the trajectories of the end-effector for Motion 2 of the 6DoF robot when varying the units while using the UC Inverse, for multiple values of  $\alpha$ .

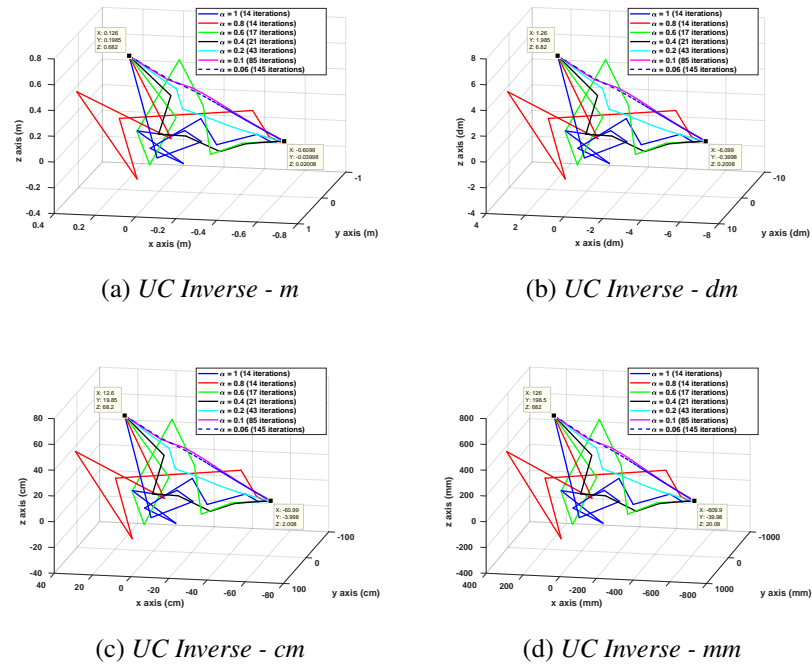


Figure 4.74: Behavior of the trajectories of the end-effector for Motion 3 of the 6DoF robot when varying the units while using the UC Inverse, for multiple values of  $\alpha$ .

- **6-DoF Robot using the MX Generalized Inverse**

Finally, the Mixed inverse was applied to all three motions of the robot and the results are presented in Table 4.53, and Figures 4.75 and 4.76 for Motion 1; Table 4.54 and Figures 4.77 and 4.78 for Motion 2, and Table 4.55 and Figures 4.79 and 4.80 for Motion 3. As it was the case for the other two generalized inverses, the *MX* inverse also works consistently despite the units or the attenuation factor.

Inverse used	MX Inverse			
Results for:	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 10 \\ 20 \\ 0.5 \\ 30 \\ 40 \\ 50 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 10 \\ 20 \\ 5 \\ 30 \\ 40 \\ 50 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 10 \\ 20 \\ 50 \\ 30 \\ 40 \\ 50 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 10 \\ 20 \\ 500 \\ 30 \\ 40 \\ 50 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.1501 \\ 0.1714 \\ 0.4743 \\ -73.6939 \\ -23.7725 \\ 54.6519 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 1.5019 \\ 1.7141 \\ 4.7434 \\ -73.6939 \\ -23.7725 \\ 54.6519 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 15.0198 \\ 17.1417 \\ 47.4347 \\ -73.6939 \\ -23.7725 \\ 54.6519 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 150.19 \\ 171.418 \\ 474.347 \\ -73.6939 \\ -23.7725 \\ 54.6519 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} -0.25 \\ 0.36 \\ 0.43 \\ -10 \\ 25 \\ -65 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -2.5 \\ 3.6 \\ 4.3 \\ -10 \\ 25 \\ -65 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -25 \\ 36 \\ 43 \\ -10 \\ 25 \\ -65 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -250 \\ 360 \\ 430 \\ -10 \\ 25 \\ -65 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} -0.2500 \\ 0.3600 \\ 0.43000 \\ -9.9996 \\ 24.9997 \\ -65.0003 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -2.5000 \\ 3.6000 \\ 4.3000 \\ -9.9996 \\ 24.9997 \\ -65.0003 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -25.0000 \\ 36.0000 \\ 43.0000 \\ -9.9996 \\ 24.9997 \\ -65.0003 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -250.0000 \\ 360.0001 \\ 429.9999 \\ -9.9996 \\ 24.9997 \\ -65.0003 \end{bmatrix}$
Position Error	0.000575m	0.000575dm	0.000575dm	0.000584mm
Orientation Error	0.000326°	0.000326°	0.000326°	0.000326°
Iterations	15	15	15	15
Computation time	0.296s	0.287s	0.288s	0.295s

Table 4.53: Experimental results obtained for Motion 1 of the 6DoF robot with  $\alpha = 1$  and MX Inverse.



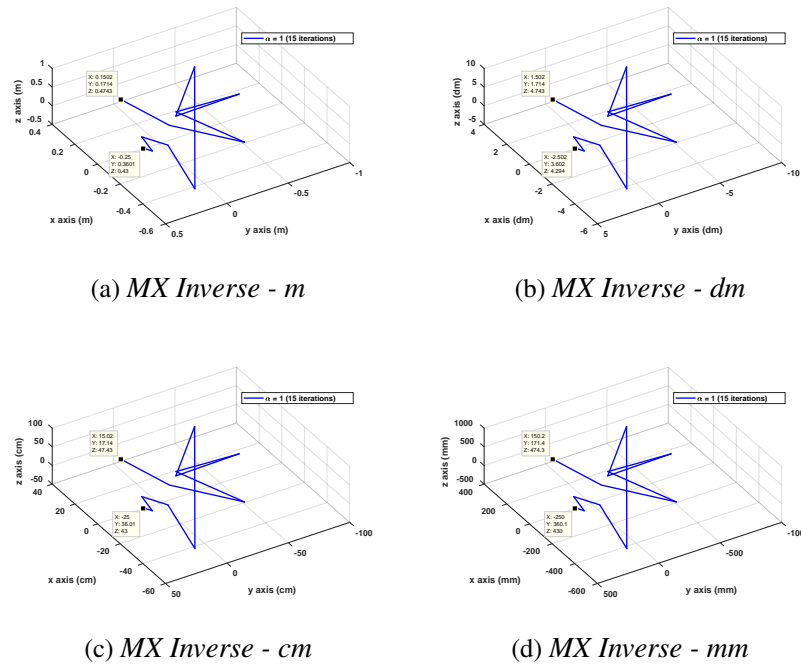


Figure 4.75: Behavior of the trajectories of the end-effector for Motion 1 of the 6DoF robot when varying the units while using the MX Inverse and  $\alpha = 1$ .

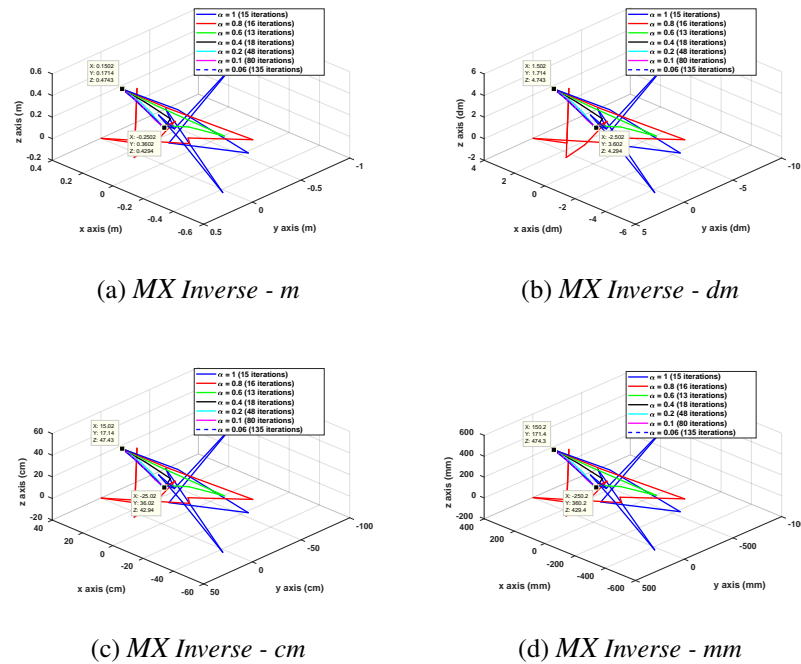


Figure 4.76: Behavior of the trajectories of the end-effector for Motion 1 of the 6DoF robot when varying the units while using the MX Inverse, for multiple values of  $\alpha$ .

Inverse used	MX Inverse			
	Results for:	m case	dm case	cm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 0.7 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 7 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 70 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 700 \\ 45 \\ 35 \\ 60 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.1260 \\ 0.1985 \\ 0.6819 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 1.2600 \\ 1.9851 \\ 6.8198 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 12.6007 \\ 19.8516 \\ 68.1981 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 126.0078 \\ 198.5166 \\ 681.9813 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} 0.42 \\ 0.16 \\ 0.21 \\ -60 \\ 11 \\ 66 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 4.2 \\ 1.6 \\ 2.1 \\ -60 \\ 11 \\ 66 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 42 \\ 16 \\ 21 \\ -60 \\ 11 \\ 66 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 420 \\ 160 \\ 210 \\ -60 \\ 11 \\ 66 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} 0.4200 \\ 0.1600 \\ 0.2100 \\ -60.0000 \\ 10.9999 \\ 66.0001 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 4.2000 \\ 1.6000 \\ 2.1000 \\ -60.0000 \\ 10.9999 \\ 66.0001 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 42.0000 \\ 16.0000 \\ 21.0000 \\ -60.0000 \\ 10.9999 \\ 66.0001 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 420.0000 \\ 160.0000 \\ 210.0000 \\ -60.0000 \\ 10.9999 \\ 66.0001 \end{bmatrix}$
Position Error	0.000140m	0.000140dm	0.000140cm	0.000141mm
Orientation Error	0.000063°	0.000063°	0.000063°	0.000063°
Iterations	10	10	10	10
Computation time	0.166s	0.155s	0.154s	0.153s

Table 4.54: Experimental results obtained for Motion 2 of the 6DoF robot with  $\alpha = 1$  and MX Inverse.

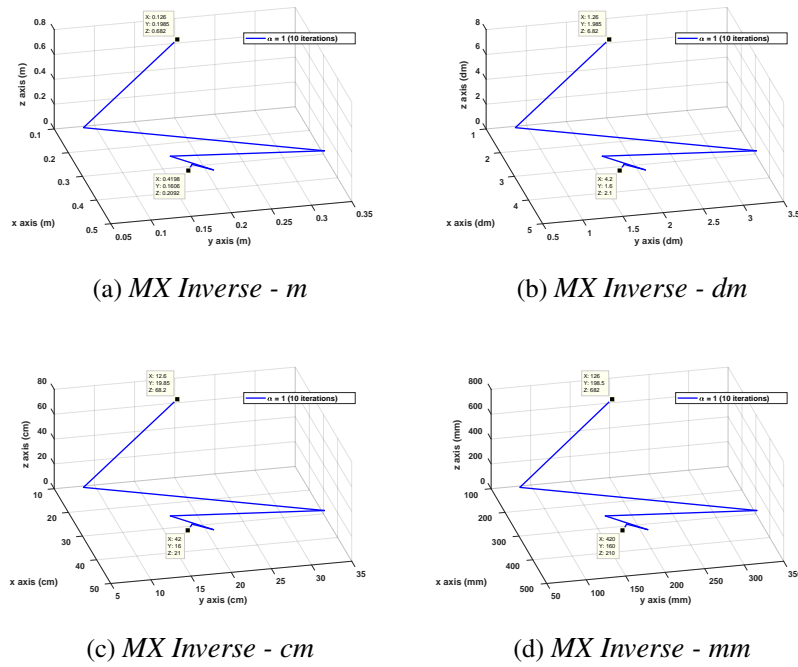


Figure 4.77: Behavior of the trajectories of the end-effector for Motion 2 of the 6DoF robot when varying the units while using the MX Inverse and  $\alpha = 1$ .

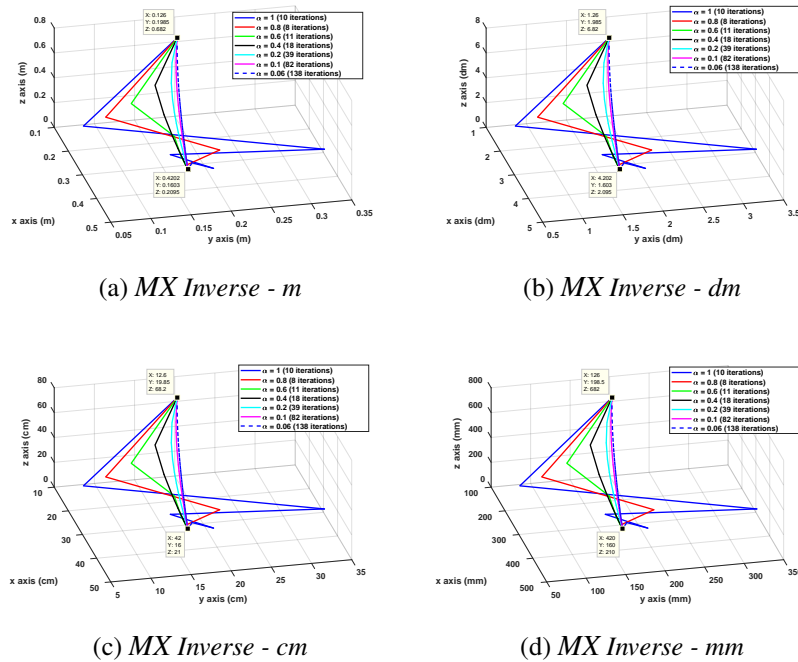


Figure 4.78: Behavior of the trajectories of the end-effector for Motion 2 of the 6DoF robot when varying the units while using the MX Inverse, for multiple values of  $\alpha$ .

Inverse used	MX Inverse			
Results for:	m case	dm case	cm case	mm case
Initial Joints	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 0.7 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 7 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 70 \\ 45 \\ 35 \\ 60 \end{bmatrix}$	$\vec{Q} = \begin{bmatrix} 20 \\ 30 \\ 700 \\ 45 \\ 35 \\ 60 \end{bmatrix}$
Initial Position	$\vec{D} = \begin{bmatrix} 0.1260 \\ 0.1985 \\ 0.6819 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 1.2600 \\ 1.9851 \\ 6.8198 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 12.6007 \\ 19.8516 \\ 68.1981 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} 126.0078 \\ 198.5166 \\ 681.9813 \\ -46.8325 \\ -11.1559 \\ 45.6142 \end{bmatrix}$
Desired Final Position	$\vec{D} = \begin{bmatrix} -0.61 \\ -0.04 \\ 0.02 \\ 1 \\ 20 \\ 23 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -6.1 \\ -0.4 \\ 0.2 \\ 1 \\ 20 \\ 23 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -61 \\ -4 \\ 2 \\ 1 \\ 20 \\ 23 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -610 \\ -40 \\ 20 \\ 1 \\ 20 \\ 23 \end{bmatrix}$
Calculated Position	$\vec{D} = \begin{bmatrix} -0.6100 \\ -0.0400 \\ 0.0200 \\ 1.0891 \\ 20.0798 \\ 22.9885 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -6.0997 \\ -0.3998 \\ 0.1999 \\ 1.0891 \\ 20.0798 \\ 22.9885 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -60.9969 \\ -3.9983 \\ 0.1989 \\ 1.0891 \\ 20.0798 \\ 22.9885 \end{bmatrix}$	$\vec{D} = \begin{bmatrix} -609.9688 \\ -39.9829 \\ 19.9888 \\ 1.0891 \\ 20.0798 \\ 22.9885 \end{bmatrix}$
Position Error	0.000037m	0.000373m	0.003730cm	0.037297mm
Orientation Error	0.060113°	0.060113°	0.060113°	0.060113°
Iterations	14	14	14	14
Computation time	0.187s	0.173s	0.189s	0.191s

Table 4.55: Experimental results obtained for Motion 3 of the 6DoF robot with  $\alpha = 1$  and MX Inverse.

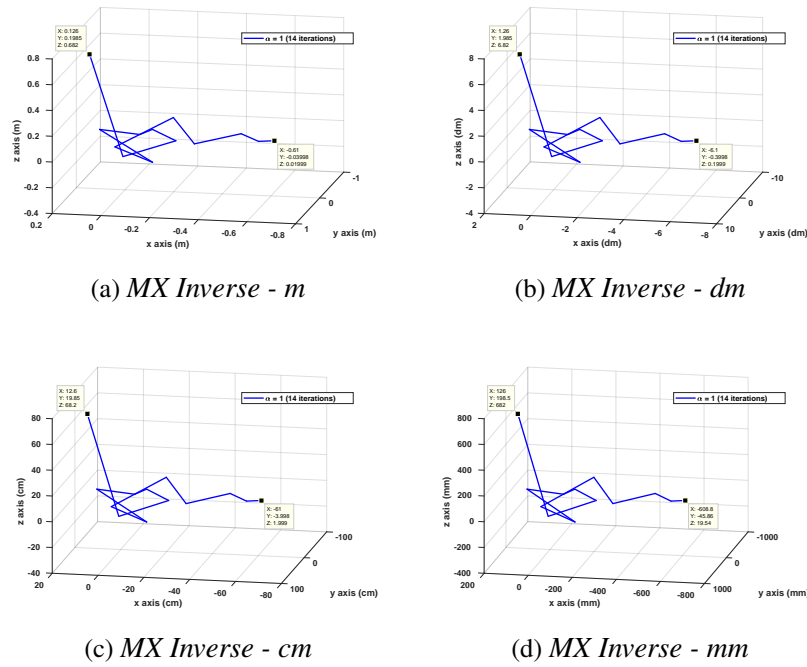


Figure 4.79: Behavior of the trajectories of the end-effector for Motion 3 of the 6DoF robot when varying the units while using the MX Inverse and  $\alpha = 1$ .

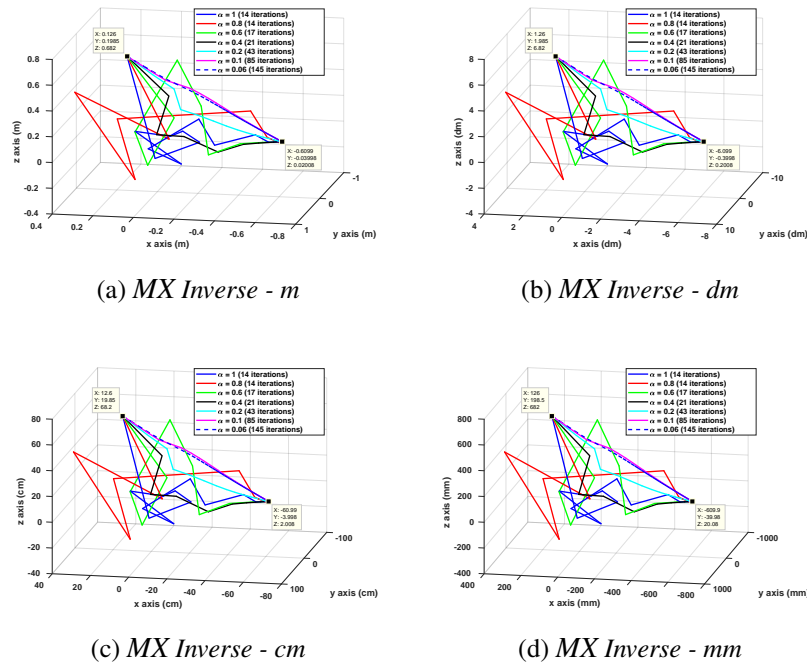


Figure 4.80: Behavior of the trajectories of the end-effector for Motion 3 of the 6DoF robot when varying the units while using the MX Inverse, for multiple values of  $\alpha$ .

Comparative factors	Investigated approximate IK methods	
	Data-driven	Numerical
Implementation Difficulty	Difficult	Easy
Dataset	Constrained workspaces	N/A
Model Tuning	more than 6 hours / robot / dataset	N/A
Final Accuracy	While reasonable from a NN point of view, not all IK solutions are acceptable for some robotics applications	IK solutions are all within range (errors less than 1mm for the position and less than 1° for the orientation)

Table 4.56: *Qualitative comparison between data-driven and numerical approaches where (N/A) means not applicable.*

### 4.2.3 Observations on the Experiments on Numeric Methods

In the above experiments, we presented the results for three arbitrary motions performed on three robots with 3, 4 and 6 DoF. The experiments show that despite the path followed, all robots achieved sub-millimeters and sub-degrees level of accuracy with respect to the desired final pose of the end-effector. The computation time reported is the average time across five runs of the proposed numerical approach. For each run, the algorithm is robust and produces quick solution to the IK problem.

## 4.3 Comparison

In this chapter, we compared two approximate methods for the IK problem: data-driven and numerical. Upon analysis of the results above and based on four main criteria: implementation difficulty, need for preprocessing – i.e. dataset creation, training, and network architecture search – and final accuracy, it seems reasonable to say that numerical methods are much better than data-driven ones. Indeed, the investigated data-driven methods (All-Joints-ANN, Individual-Joints-ANN, and Individual-Joints-ANN) were time-consuming due to their requirement to determine the joint constraints, generate the dataset, search for an appropriate architecture before training and evaluating the network. For example, the network architecture search for the random dataset (10000 data points) of the 4DoF manipulator with the All-Joints-ANN run for approximately 6 hours on a desktop with an Intel(R) Core(TM) i7-8700CPU and NVIDIA GTX 2060 GPU. Moreover, the best

architectures obtained for all investigated ANNs provided in average much higher MSE values than the numeric methods. On the other hand, the proposed numerical approach was easily implemented and applicable to any manipulators.

# Chapter 5

## Conclusion and Future Work

This thesis investigated the use of approximate solutions for the inverse kinematics. Two main groups of approximate IK methods were presented — data-driven and numerical — and used for the computation of IK solutions in task-independent workspaces. In the case of the data-driven methods, Multi-Layer Perceptron (MLP) and Adaptive Neuro-Fuzzy Inference System (ANFIS) were trained and evaluated for task-free workspaces depicted by two types of datasets — structured and random. Different architectures of MLPs (All-joints-ANN, Individual-Joints-ANN) and an ANFIS (Individual-Joints-ANFIS) were employed using four robotic arms with 4, 5, 6 and 7 DoF, and the errors measured in terms of predicted joint configurations and reconstructed poses, under a challenging task-independent scenario. The datasets were generated by varying all manipulator joints within specific ranges, in an effort to confine the learned workspace and hence improve the chances of better results.

From the experiments conducted, MLPs and ANFIS are apparently not the best options for solving task-independent IK problems because of the high error values obtained for some of the predictions. The vast combination of highly pattern-free input vectors makes it quite difficult for neural networks to capture any similarity in the inputs and consequently, to learn the high-dimension mapping function between end-effector poses and required joint values.

For a better level of accuracy, a robust and accurate numerical method was investigated. This numerical approach uses the Jacobian and inverse Jacobian matrices to iterate between an initial and final end-effectors poses. Based on different inverse meth-



ods Moore-Penrose (MP), Unit-Consistent (UC) and Mixed (MX) inverses; the proposed method was evaluated on three incommensurate robotic manipulators with 3, 4, and 6 DoF. Furthermore, we investigated the effects of the attenuation parameter both on the accuracy of the final IK solution and the smoothness between the initial and final poses. From the experiments conducted, the numerical approach reaches sub-millimeters accuracy in the estimation of the end-effector position and sub-degrees accuracies in the estimation of the end-effector orientation. And the variation of the attenuation parameter plays an important in provided stable solutions. Overall, our study shows that the proposed numerical approach is simpler to implement, less time consuming, more robust and accurate than data-driven IK approaches.

The study conducted on the approximate IK solutions gave several directions for future work. In the case of data-driven methods, the use of an ensemble approach may be investigated. In such an attempt, multiple neural networks would be specialized in learning subspaces of a specific manipulator workspace. Moreover, with the growing interest of Neural Architectural Search (NAS) [54–56] methods that create a correlation between a specific dataset and an optimized network architecture, finding the appropriate network architecture based on the IK task to conduct may also be explored under advanced NAS paradigms. In the specific case of the proposed numerical approach, decaying the attenuation parameter or investigating the use of multiple values of the attenuation parameter in a parallel setting when searching for IK solutions may lead to faster convergence of the algorithm.

# Bibliography

- [1] S. B. Niku, *Introduction to robotics: analysis, control, applications*. John Wiley & Sons, 2010. (document), 1, 2.1, 2.1
- [2] V. V. Fjodor. (2016) The neural network zoo. [Online]. Available: <https://www.asimovinstitute.org/author/fjodorvanveen/> (document), 3.1
- [3] J.-S. Jang, “Anfis: adaptive-network-based fuzzy inference system,” *IEEE transactions on systems, man, and cybernetics*, vol. 23, no. 3, pp. 665–685, 1993. (document), 2.3.1.1, 3.3, 3.1.1.2, 3.4, 4.1.3.3
- [4] S. Farzan and G. N. DeSouza, “From dh to inverse kinematics: A fast numerical solution for general robotic manipulators using parallel processing,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 2507–2513. (document), 2.3.2.2, 3.2, 3.6, 3.2, 3.1, 4.1.5, 4.2.1, 4.2.2
- [5] F. Chollet *et al.*, “Keras,” <https://keras.io>, 2015. (document), 4.5, 4.11
- [6] R. R. Yager and D. P. Filev, “Generation of fuzzy rules by mountain clustering,” *Journal of Intelligent & Fuzzy Systems*, vol. 2, no. 3, pp. 209–219, 1994. (document), 4.6, 4.1.3.3, 4.7, 4.8, 4.9
- [7] A. Aristidou and J. Lasenby, “Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver,” 2009. 1, 2.1
- [8] J. J. Craig, *Introduction to robotics: mechanics and control, 3/E*. Pearson Education India, 2009. 1, 2.1, 2.2

- [9] A. Aristidou, J. Lasenby, Y. Chrysanthou, and A. Shamir, "Inverse kinematics techniques in computer graphics: A survey," in *Computer Graphics Forum*, vol. 37, no. 6. Wiley Online Library, 2018, pp. 35–58. 1, 2.1, 2.2, 2.3.2
- [10] R. S. Hartenberg and J. Denavit, "A kinematic notation for lower pair mechanisms based on matrices," 1955. 1, 2.1
- [11] B. B. Choi and C. Lawrence, "Inverse kinematics problem in robotics using neural networks," 1992. 1, 2.3.1.1
- [12] P. Jha and B. Biswal, "A neural network approach for inverse kinematic of a scara manipulator," *IAES International Journal of Robotics and Automation*, vol. 3, no. 1, p. 52, 2014. 1, 2.3.1.1, A.2a
- [13] A. Csiszar, J. Eilers, and A. Verl, "On solving the inverse kinematics problem using neural networks," in *2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*. IEEE, 2017, pp. 1–6. 1, 2.3.1.1, A.2d
- [14] P. Srisuk, A. Sento, and Y. Kitjaidure, "Inverse kinematics solution using neural networks from forward kinematics equations," in *2017 9th International Conference on Knowledge and Smart Technology (KST)*. IEEE, 2017, pp. 61–65. 1, 2.3.1.1
- [15] K. D. Polyzos, P. P. Groumpos, and E. Dermatas, "Solving the inverse kinematics of robotic arm using autoencoders," in *Conference on Creativity in Intelligent Technologies and Data Science*. Springer, 2019, pp. 288–298. 1, 2.3.1.1
- [16] J. Demby's, Y. Gao, and G. N. DeSouza, "A study on solving the inverse kinematics of serial robots using artificial neural network and fuzzy neural network," in *2019 International Conference on Fuzzy Systems (FuzzIEEE)*. IEEE, 2019, pp. 1433–1438. 1, 2.3.1
- [17] B. Daya, S. Khawandi, M. Akoum *et al.*, "Applying neural network architecture for inverse kinematics problem in robotics," *Journal of Software Engineering and Applications*, vol. 3, no. 03, p. 230, 2010. 1
- [18] Z. Zhou, H. Guo, Y. Wang, Z. Zhu, J. Wu, and X. Liu, "Inverse kinematics solution for robotic manipulator based on extreme learning machine and sequential mutation

- genetic algorithm,” *International Journal of Advanced Robotic Systems*, vol. 15, no. 4, p. 1729881418792992, 2018. 1, 2.3.1.1, A.2c, A.2
- [19] A. El-Sherbiny, M. A. Elhosseini, and A. Y. Haikal, “A comparative study of soft computing methods to solve inverse kinematics problem,” *Ain Shams Engineering Journal*, 2017. 1, 2.3.1.1, A.2b, A.2
- [20] R. Penrose, “A generalized inverse for matrices,” in *Mathematical proceedings of the Cambridge philosophical society*, vol. 51, no. 3. Cambridge University Press, 1955, pp. 406–413. 1
- [21] B. Zhang and J. Uhlmann, “Applying a unit-consistent generalized matrix inverse for stable control of robotic systems,” *Journal of Mechanisms and Robotics*, vol. 11, no. 3, p. 034503, 2019. 1, 2.3.2.3, 4.2.1, 4.26a, 4.27a
- [22] J. Uhlmann, “A generalized matrix inverse that is consistent with respect to diagonal transformations,” *SIAM Journal on Matrix Analysis and Applications*, vol. 39, no. 2, pp. 781–800, 2018. 1, 2.3.2.2, 2.3.2.2, 2.3.2.2, 2.3.2.3
- [23] E. Schwartz, R. Manseur, and K. Doty, “Noncommensurate systems in robotics,” *International Journal of Robotics and Automation*, vol. 17, no. 2, pp. 86–92, 2002. 1, 2.3.2.3
- [24] E. M. Schwartz, R. Manseur, and K. L. Doty, “Non-commensurate manipulator jacobian.” in *Robotics and Applications*, 2003, pp. 112–115. 1, 2.3.2.2, 2.3.2.3
- [25] D. Isbell, M. Hardin, and J. Underwood, “Mars climate orbiter team finds likely cause of loss,” *NASA news release*, 1999. 1, 2.3.2.3
- [26] R. Lloyd and C. I. S. Writer, “Metric mishap caused loss of nasa orbiter,” *CNN Interactive*, 1999. 1, 2.3.2.3
- [27] A. Guez, “Solution to the inverse kinematics problem in robotics by neural network,” in *Proc. Int. Conf. on Neural Networks, Skovde, 1988*, 1988, pp. II617–II624. 2.3.1.1

- [28] S. Alavandar and M. J. Nigam, “Neuro-fuzzy based approach for inverse kinematics solution of industrial robot manipulators,” *International Journal of Computers Communications & Control*, vol. 3, no. 3, pp. 224–234, Sept. 2008. 2.3.1.1
- [29] J. Narayan and A. Singla, “Anfis based kinematic analysis of a 4-dofs scara robot,” in *2017 4th International Conference on Signal Processing, Computing and Control (ISPCC)*. IEEE, 2017, pp. 205–211. 2.3.1.1
- [30] A. R. Almusawi, L. C. Dülger, and S. Kapucu, “A new artificial neural network approach in solving inverse kinematics of robotic arm (denso vp6242),” *Computational intelligence and neuroscience*, vol. 2016, 2016. 2.3.1.1, A.2b
- [31] J. Chen and H. Y. Lau, “Inverse kinematics learning for redundant robot manipulators with blending of support vector regression machines,” in *2016 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*. IEEE, 2016, pp. 267–272. 2.3.1.1
- [32] S. N. Raptis and E. S. Tzafestas, “Robot inverse kinematics via neural and neurofuzzy networks: architectural and computational aspects for improved performance,” *Journal of Information and Optimization Sciences*, vol. 28, no. 6, pp. 905–933, 2007. 2.3.1.1
- [33] Z. Bingul, H. Ertunc, and C. Oysu, “Comparison of inverse kinematics solutions using neural network for 6r robot manipulator with offset,” in *2005 ICSC congress on computational intelligence methods and applications*. IEEE, 2005, pp. 5–pp. 2.3.1.1, A.2c
- [34] P.-Y. Zhang, T.-S. Lü, and L.-B. Song, “Rbf networks-based inverse kinematics of 6r manipulator,” *The International Journal of Advanced Manufacturing Technology*, vol. 26, no. 1-2, pp. 144–147, 2005. 2.3.1.1, A.2b
- [35] R. Köker, T. Cakar, and Y. Sari, “A neural-network committee machine approach to the inverse kinematics problem solution of robotic manipulators,” *Engineering with Computers*, vol. 30, no. 4, pp. 641–649, 2014. 2.3.1.1

- [36] H. Hoffmann and R. Möller, “Unsupervised learning of a kinematic arm model,” in *Artificial neural networks and neural information processing - ICANN/ICONIP 2003*. Springer, 2003, pp. 463–470. 2.3.1.1
- [37] G. D. A. Barreto, A. F. Araújo, and H. J. Ritter, “Self-organizing feature maps for modeling and control of robotic manipulators,” *Journal of Intelligent and Robotic Systems*, vol. 36, no. 4, pp. 407–450, 2003. 2.3.1.1
- [38] S. Cavalcanti and O. Santana, “Self-learning in the inverse kinematics of robotic arm,” in *2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)*. IEEE, 2017, pp. 1–5. 2.3.1.1
- [39] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016. 2.3.2.2
- [40] A. Ben-Israel and A. Charnes, “Contributions to the theory of generalized inverses,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 3, pp. 667–699, 1963. 2.3.2.2
- [41] A. Albert *et al.*, “Regression and the moore-penrose pseudoinverse,” 1972. 2.3.2.2
- [42] G. H. Golub and C. Reinsch, “Singular value decomposition and least squares solutions,” in *Linear Algebra*. Springer, 1971, pp. 134–151. 2.3.2.2
- [43] C. A. Klein and C.-H. Huang, “Review of pseudoinverse control for use with kinematically redundant manipulators,” *IEEE Transactions on Systems, Man, and Cybernetics*, no. 2, pp. 245–250, 1983. 2.3.2.3, 4.2.1
- [44] E. M. Schwartz and K. L. Doty, “The weighted generalized-inverse applied to mechanism controllability,” in *6th Annual Conference on Recent Advances in Robotics, University of Florida, Gainesville, Florida. Pp. B*, vol. 1, 1993, pp. 1–6. 2.3.2.3
- [45] E. M. Schwartz, “Algebraic properties of noncommensurate systems and their applications in robotics,” Ph.D. dissertation, University of Florida, 1995. 2.3.2.3

- [46] E. M. Schwartz, R. Manseur, and K. L. Doty, “Algebraic properties of non-commensurate systems,” in *Florida Conference on Recent Advances in Robotics, University of Florida, Gainesville, Florida*. Citeseer, 1999. 2.3.2.3
- [47] K. L. Doty, C. Melchiorri, and C. Bonivento, “A theory of generalized inverses applied to robotics,” *The International Journal of Robotics Research*, vol. 12, no. 1, pp. 1–19, 1993. 2.3.2.3, 4.2.1
- [48] K. L. Doty, C. Melchiorri, E. M. Schwartz, and C. Bonivento, “Robot manipulability,” *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3, pp. 462–468, 1995. 2.3.2.3
- [49] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994. 3.1.1, 3.2a
- [50] J. M. Keller, D. Liu, and D. B. Fogel, *Fundamentals of computational intelligence: neural networks, fuzzy systems, and evolutionary computation*. John Wiley & Sons, 2016. 3.2b
- [51] R. Reed and R. J. MarksII, *Neural smithing: supervised learning in feedforward artificial neural networks*. Mit Press, 1999. 3.1.1
- [52] MATLAB, *version 9.2.0.538062 (R2017a)*. Natick, Massachusetts: The MathWorks Inc., 2017. 3.1.1.2
- [53] S. Farzan and G. N. DeSouza, “A parallel evolutionary solution for the inverse kinematics of generic robotic manipulators,” in *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 358–365. 3.2, 3.2, 4.2.2
- [54] T. Elsken, J. H. Metzen, and F. Hutter, “Neural architecture search: A survey,” *arXiv preprint arXiv:1808.05377*, 2018. 4.1.3.1, 5
- [55] T. K. Gupta and K. Raza, “Optimizing deep neural network architecture: a tabu search based approach,” *arXiv preprint arXiv:1808.05979*, 2018. 4.1.3.1, 5
- [56] T. Elsken, J. H. Metzen, and F. Hutter, “Neural architecture search: A survey,” *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019. 5

- [57] B. T. Polyak, “Some methods of speeding up the convergence of iteration methods,” *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964. A.1.2
- [58] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in *International conference on machine learning*, 2013, pp. 1139–1147. A.1.3
- [59] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011. A.1.4
- [60] LeCun Yann, Bengio Yoshua, and Hinton Geoffrey, “Deep learning,” *Nature*, vol. 521, p. 436, may 2015. A.1.4
- [61] G. Hinton, “Neural networks for machine learning. coursera,[video lectures],” 2012. A.1.5
- [62] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014. A.1.6



# Appendix A

## Optimization Techniques

### A.1 Optimization methods for gradient descent algorithm

The gradient descent algorithm used during the backward pass (back-propagation algorithm described in section 3.1.1) can be optimized with some common techniques that we present in this section. These well established techniques focus on the improvement of the update step of back-propagation to optimize a neural network.

#### A.1.1 Serial Gradient Descent

Serial Gradient Descent, used in equations (3.6) and (3.7) is the simplest update form to change parameters along the direction of a negative gradient. The effect of the momentum

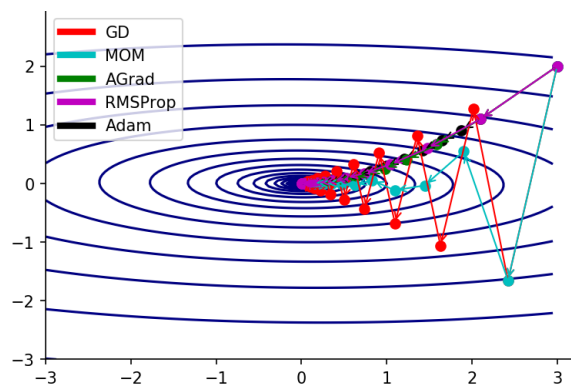


Figure A.1: An example representation of the effects of common optimization techniques for back-propagation algorithm.

is depicted in Figure A.1. Let us consider  $\theta$  a parameter that will be updated and  $d\theta$  the gradient evaluated at the location of  $\theta$ , the general form of the serial Gradient Descent update is expressed by:

$$\theta^{new} = \theta^{old} - (\alpha * d\theta) \quad (\text{A.1.1})$$

where:  $\alpha$  is the learning rate.

### A.1.2 Momentum

The momentum is designed to accelerate the learning process especially for high curvature, small but consistent gradients, or noisy gradients [57]. The name momentum comes from the analogy with which a negative gradient is a force moving a particle through space, based on Newton's law of motion. The effect of the momentum is depicted in Figure A.1. Let us consider  $\theta$  a parameter that will be updated and  $d\theta$  the gradient evaluated at the location of  $\theta$ , the momentum update form is expressed by:

$$\begin{aligned} v &= (\beta * v) - (\alpha * d\theta) \\ \theta^{new} &= \theta^{old} + v \end{aligned} \quad (\text{A.1.2})$$

where:  $\alpha$  is the learning rate,  $\beta$  is the momentum,  $v$  is the velocity parameter initialized to 0.

### A.1.3 Nesterov Momentum

Inspired from the Nesterov's accelerated gradient method, the Nesterov momentum was introduced as a slightly different variant of the momentum [58]. The difference between the Nesterov momentum and the momentum is where the gradient is evaluated. In fact, with Nesterov momentum, the gradient is computed after the current velocity is applied to the old position of the parameter to update. In that sense, it adds a correction factor to the standard momentum method. Let us consider  $\theta$  a parameter that will be updated and  $d\theta$  the gradient evaluated at the location of  $\theta$ . The correction is applied to  $\theta^{old}$  to find  $\theta^{corrected}$  and the gradient is instead evaluated at this corrected location to find  $d\theta^{corrected}$ . The Nesterov momentum update form is expressed by:

$$\begin{aligned}
\theta^{corrected} &= \theta^{old} + (\beta * v) \\
v &= (\beta * v) - (\alpha * d\theta^{corrected}) \\
\theta^{new} &= \theta^{old} + v
\end{aligned}
\tag{A.1.3}$$

where:  $\alpha$  is the learning rate,  $\beta$  is the momentum,  $v$  is the velocity parameter initialized to 0.

#### A.1.4 Adaptive Gradient (Adagrad)

Adagrad is a per parameter adaptive learning rate algorithm originally proposed by [59], that adapts the learning rates of model parameters by proportionally scaling them to the square root of the sum of all their historical squared values [60] later referred as  $\theta^{cached}$ . This is better captured by the update form expressed by equation (A.1.4). The parameters with high gradients (largest partial derivatives) will have their learning rate reduced while the the parameters with small gradients (small partial derivatives) will have their learning rate increased. That is, the progression path of the optimization toward the solution (global minimum) is smoothed in the search space. The effect of Adagrad is also shown in Figure A.1. Let us consider  $\theta$  a parameter that will be updated and  $d\theta$  the gradient evaluated at the location of  $\theta$ .

$$\begin{aligned}
\theta^{cached} &= (d\theta)^2 \\
\theta^{new} &= \theta^{old} - \frac{\alpha * d\theta}{\sqrt{\theta^{cached} + \epsilon}}
\end{aligned}
\tag{A.1.4}$$

where:  $\alpha$  is the learning rate,  $\epsilon$  is the smoothing term which avoids the division by 0.

#### A.1.5 RMSProp

RMSProp simplifies the Adagrad update to reduce its aggressive and monotonically decreasing learning rate to perform better when applied to non-convex functions [61]. In fact, the mathematical design of Adagrad makes it converge rapidly when applied to convex functions. RMSProp changes the gradient accumulation  $\theta^{cached}$  used by Adagrad to an exponentially weighted moving average. The RMSProp update form is expressed by:

$$\begin{aligned}\theta^{cached} &= \rho * \theta^{cached} + (1 - \rho) * (d\theta)^2 \\ \theta^{new} &= \theta^{old} - \frac{\alpha * d\theta}{\sqrt{\theta^{cached} + \epsilon}}\end{aligned}$$

where:  $\alpha$  is the learning rate,  $\epsilon$  is the smoothing term which avoids the division by 0,  $\rho$  is a hyper-parameter controlling the length scale of the moving average of squared gradients.

### A.1.6 Adaptive moments (Adam)

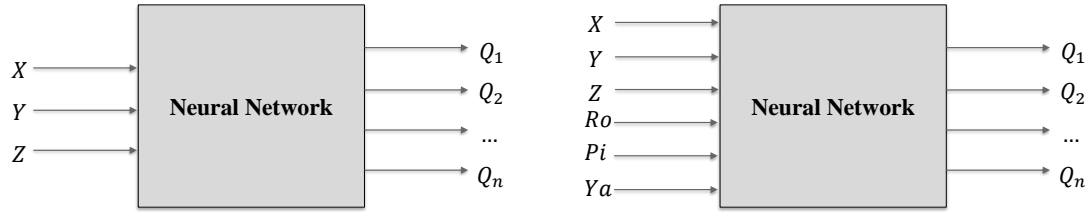
Adam is a variant of the combination of RMSProp and momentum by combining first and second order moments [62]. This update form takes into account exponentially weighted moving average of first and second order moments. Let us consider  $\theta$  a parameter that will be updated and  $d\theta$  the gradient evaluated at the location of  $\theta$ . The Adam update form is expressed by:

$$\begin{aligned}m &= \beta_1 * m + (1 - \beta_1) * d\theta \\ v &= \beta_2 * v + (1 - \beta_2) * (d\theta)^2 \\ \theta^{new} &= \theta^{old} - \frac{\alpha * d\theta}{\sqrt{\theta^{cached} + \epsilon}}\end{aligned}$$

where:  $\alpha$  is the learning rate,  $\epsilon$  is the smoothing term which avoids the division by 0,  $\rho$  is a hyper-parameter controlling the length scale of the moving average of squared gradients,  $\beta_1$  and  $\beta_2$  are hyper-parameters balancing how much the update takes into account the history,  $m$  and  $v$  are accumulation variables respectively for the first order momentum and the second order momentum.

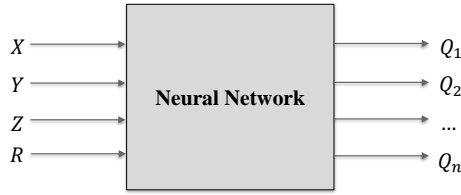
In practice, Adam is recommended as the default update form. It is also recommended to use the combination of Gradient Descent with Nesterov momentum. However, there is currently no best update form found to be applicable and successful to a wide range of learning tasks.

The above revisited neural networks concepts are exploited to approximate the IK of serial robots using MLP and ANFIS data-driven IK solvers investigated in this thesis.



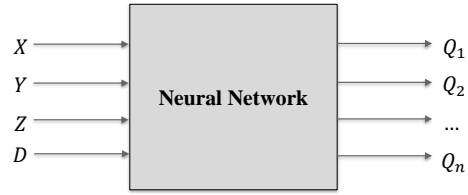
(a) A representation of the learning scheme used in [12] where the inputs represent the coordinates of the end-effector position.

(b) A representation of the learning scheme used in [19, 30, 34] where the inputs represent the position and orientation of the end-effector.



Where:  $R = R(\vec{n}, \delta, \vec{a}) = 9$  elements of the rotation matrix

(c) A representation of the learning scheme used in [18, 33] where the inputs represent the position and elements of the rotation matrix (orientation) of the end-effector. The network has 12 inputs in this case.



Where:  $D = \sqrt{X^2 + Y^2 + Z^2}$  is the Euclidean distance

(d) A representation of the learning scheme used in [13] where the inputs represent the coordinates of the end-effector position and their Euclidean distance.

Figure A.2: Overview of some of the learning schemes found in the literature for data-driven inverse kinematics solvers using all joints in the output layer.

## A.2 Optimization methods for the network initial predictions

When the predicted joints are less accurate than the desired ones; several optimization techniques can be used to improve the network responses. For example, in [19], an error minimization algorithm was used to improve the results predicted by a MLP and ANFIS. Also, in [18], a sequential mutation genetic algorithm was used to optimize preliminary results returned by an extreme learning machine.