

CRANFIELD UNIVERSITY

DAVID JAMES GALVÃO WALL

A GRAPH-THEORY-BASED C-SPACE PATH PLANNER FOR MOBILE
ROBOTIC MANIPULATORS IN CLOSE-PROXIMITY ENVIRONMENTS

CRANFIELD DEFENCE AND SECURITY

PhD Thesis

Academic Year 2015-2016

Supervisor: Dr. John Economou

January 2016

CRANFIELD UNIVERSITY

CRANFIELD DEFENCE AND SECURITY

PhD Thesis

Academic Year 2015-2016

DAVID JAMES GALVÃO WALL

A graph-theory-based C-space path planner for mobile robotic
manipulators in close-proximity environments

Supervisor: Dr. John Economou

January 2016

© Cranfield University 2016, All rights reserved. No part of this
publication may be reproduced without the written permission of the
copyright owner.

To my beautiful wife Silvia and our expected twins.

In loving memory of my Grandad Tom.

ABSTRACT

In this thesis a novel guidance method for a 3-degree-of-freedom robotic manipulator arm in 3 dimensions for Improvised Explosive Device (IED) disposal has been developed. The work carried out in this thesis combines existing methods to develop a technique that delivers advantages taken from several other guidance techniques. These features are necessary for the IED disposal application. The work carried out in this thesis includes kinematic and dynamic modelling of robotic manipulators, T-space to C-space conversion, and path generation using Graph Theory to produce a guidance technique which can plan a safe path through a complex unknown environment. The method improves upon advantages given by other techniques in that it produces a suitable path in 3-dimensions in close-proximity environments in real time with no a priori knowledge of the environment, a necessary precursor to the application of this technique to IED disposal missions.

To solve the problem of path planning, the thesis derives the kinematics and dynamics of a robotic arm in order to convert the Euclidean coordinates of measured environment data into C-space. Each dimension in C-space is one control input of the arm. The Euclidean start and end locations of the manipulator end effector are translated into C-space. A three-dimensional path is generated between them using Dijkstra's Algorithm. The technique allows for a single path to be generated to guide the entire arm through the environment, rather than multiple paths to guide each component through the environment. The robotic arm parameters are modelled as a quasi-linear parameter varying system. As such it requires gain scheduling control, thus allowing compensation of the non-linearities in the system. A Genetic Algorithm is applied to tune a set of PID controllers for the dynamic model of the manipulator arm so that the generated path can then be followed using a conventional path-following algorithm. The technique proposed in this thesis is validated using numerical simulations in order to determine its advantages and limitations.

Keywords:

Robotic Manipulator, Guidance, Control, C-space, Graph Theory, Path Generation.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. John Economou for his ongoing support, encouragement during this research project. His enthusiasm and excitement to understand the work carried out was inspirational and motivated me to drive the research forwards.

I would like to thank Prof. Kevin Knowles for his contributions to discussion as part of the thesis committee. I am very grateful for his support as well as his attention to detail. I would especially like to thank Dr. Hugh Goyder, whose advice was invaluable. I would also like to thank Bill Staffner for the thought provoking discussions.

My thanks go to Dr. Peter Silson, Prof. Antonios Tsourdos and Dr. Seungkeun Kim for their input as Supervisor and thesis committee at the start of this research. Their suggestions and advice helped to lay the foundations for the work. A great deal of thanks go to my various officemates, Hyon-dong Oh, Jiyoung Choi and Rodrigo Felix Moreno for their support and discussions.

Finally, my thanks go to my family, who have supported me completely throughout the duration of the research, but especially Silvia, who has been a rock of support since the day I met her, and has been patient, understanding and helpful throughout both the research and writing up of this thesis, even when the nights have been sleepless.

TABLE OF CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENTS	iii
TABLE OF CONTENTS	v
NOMENCLATURE	xi
LIST OF TABLES	xix
TABLE OF FIGURES	xxi
TABLE OF EQUATIONS	xxxvii
1 INTRODUCTION.....	1
1.1 Background to Research.....	1
1.2 Research Question	5
1.3 Aims and Objectives	5
1.4 Project Outline.....	7
1.5 Assumptions and Bounds	9
1.6 Contributions	11
1.7 Summary.....	12
2 REVIEW OF LITERATURE.....	15
2.1 Modelling of Robotic Manipulators	15
2.2 Control of Robotic Manipulators	18
2.3 Guidance.....	21
2.3.1 Mapping.....	21
2.3.2 Path Generation and Following	26

2.4	Summary of Literature.....	50
3	KINEMATIC MODELLING OF ROBOTIC MANIPULATORS	53
3.1	Overview	54
3.2	Trigonometric Method	55
3.2.1	Forward Kinematics	55
3.2.2	Inverse Kinematics	58
3.3	Denavit-Hartenberg (Matrix Transform) Method.....	60
3.3.1	The parameters of the robotic arm	60
3.3.2	Derivation of the transformation matrices for each arm link.....	64
3.3.3	The forward kinematics of the robotic manipulator arm.	68
3.3.4	The inverse kinematics of the robotic manipulator arm.	70
3.4	Simultaneous Geometric Equations for Inverse Kinematics.....	76
3.5	Summary of Kinematic Modelling.....	80
4	DYNAMIC MODELLING OF A ROBOTIC MANIPULATOR	83
4.1	Manipulator Arm Parameters	84
4.2	Dynamic Equation Formulation	85
4.2.1	First Link	86
4.2.2	Second Link.....	89
4.2.3	Third Link.....	97
4.2.4	Angular Acceleration	100
4.2.5	Qualitative Validation	105
4.2.6	Summary of Dynamic Modelling.....	136
4.3	Servo Model.....	137
4.4	Summary of Dynamic Modelling.....	141

5	CONTROL OF 3-DOF MANIPULATOR ARM	143
5.1	Implementation.....	144
5.1.1	Genetic Algorithm	145
5.1.2	Robotic Manipulator Arm Tuning	149
5.1.3	Fitness Functions	150
5.2	Genetic Algorithm Validation.....	160
5.2.1	Validation of the GA Using Standard Optimisation Problems	160
5.2.2	Comparison of Optimisation Methods on the Robotic Arm Tuning Problem.....	169
5.3	Use of GA to Optimise Arm Gains.....	171
5.3.1	Assumptions.....	171
5.3.2	Optimisation Process.....	174
5.4	Gain Scheduler for Robotic Manipulator PID Controller	176
5.5	Random Step Sequence Testing and Validation.....	187
5.6	Summary of GA PID Gain Tuning Method	196
6	ENVIRONMENT MODELLING AND MAPPING IN C-SPACE	197
6.1	Implementation.....	198
6.1.1	Obtain Obstacle Data	199
6.1.2	Convert Obstacle Data into C-Space.....	206
6.1.3	Expand Impermissible Region to Permissible Boundary.	227
6.1.4	Node Graph Formation.....	230
6.1.5	Mapping of End Effector Start and Required Joint Positions	243
6.2	Summary of Environment Modelling	244
7	PATH GENERATION USING GRAPH THEORY	247

7.1	Path Generation in C-Space	248
7.2	Comparison of Path Generation Techniques	251
7.2.1	Bellman-Ford	257
7.2.2	Breadth-first	260
7.2.3	Dijkstra.....	263
7.3	Path to Trajectory Conversion.....	265
7.4	Summary of Robot Arm Guidance	272
8	VALIDATION OF GUIDANCE METHOD BY SIMULATION.....	277
8.1	Simulation Design	277
8.2	Results	278
8.2.1	Single Obstacle	278
8.2.2	Two Obstacles.....	282
8.2.3	Three Obstacles	288
8.2.4	Four Obstacles	294
8.2.5	Single Obstacle with Hole.....	298
8.2.6	Narrow Passage Between Two Long Obstacles.....	303
8.3	Summary.....	307
9	CONCLUSIONS.....	309
9.1	Further Conclusions	310
9.2	Research Contributions.....	312
9.3	Advantages and Limitations of the Technique.....	314
10	FUTURE RESEARCH WORK.....	317
11	CLOSING SUMMARY	319
	REFERENCES.....	xlv

APPENDICES	xiii
Appendix A Genetic Algorithm Supplementary Information and Results	xiii
Appendix B Extension of a 3-DoF Path Planning Algorithm to 9-DoF.....	xxviii
Appendix C List of Publications	xxx
C.1 Published Works	xxx
C.2 Submitted for Publication	xxxi
C.3 Submitted Abstracts.....	xxxi
C.4 Under Preparation.....	xxxi

NOMENCLATURE

Table of Abbreviations

CoG	Centre of gravity
CPC	Completeness and parametric continuity
C-Space	Configuration space
D-H	Denavit-Hartenberg
DoF	Degrees-of-freedom
GA	Genetic algorithm
IED	Improvised explosive device
IR	Infra-red
Latax	Lateral acceleration
L-E	Lagrange-Euler
LIDAR	Light detection and ranging
MIMO	Multi-input multi-output
N-E	Newton-Euler
P	Proportional feedback control
PD	Proportional derivative feedback control
PH	Pythagorean hodograph
PI	Proportional integral feedback control
PID	Proportional integral derivative feedback control

RADAR	Radio detection and ranging
RBF	Radial Basis Function
ROV	Remotely Operated Vehicle
RRT	Rapidly exploring random tree
SDRE	State-dependant Riccati equation
SLAM	Simultaneous localisation and mapping
SONAR	Sound navigation and ranging
ToF	Time of flight
T-Space	Task space
UAV	Uninhabited aerial vehicle
US	Ultra-sonic
DCM	Direction cosine matrix

Symbols

Variable	Description	Units
d	Light and sound based transmitter receiver spacing.	m
d	Stereoscopic sensor disparity	m
x	Light and sound based sensor distance to detected object	m
V	No. of node graph vertices	
E	No. of node graph edges	
O	Big-O notation running time	
α	Robotic manipulator joint 1 angle (manipulator frame)	$^{\circ}$ or c

β	Robotic manipulator joint 2 angle (manipulator frame)	$^{\circ}$ or c
γ	Robotic manipulator joint 3 angle (manipulator frame)	$^{\circ}$ or c
l_0	Robotic manipulator link 1 length (manipulator frame)	m
l_1	Robotic manipulator link 2 length (manipulator frame)	m
l_2	Robotic manipulator link 3 length (manipulator frame)	m
P_0	Robotic manipulator base/joint 1 location	m
P_1	Robotic manipulator joint 2 location	m
P_2	Robotic manipulator joint 3 location	m
P_f	Robotic manipulator end effector location	m
ζ	Robotic manipulator joint 3 intermediate angle	$^{\circ}$ or c
ε	Robotic manipulator joint 3 intermediate angle	$^{\circ}$ or c
δ	Robotic manipulator joint 3 intermediate angle	$^{\circ}$ or c
A	Direction cosine matrix	
R	Direction cosine matrix rotation cosine matrix	$^{\circ}$ or c
P	Direction cosine matrix position translation	m
f	Direction cosine matrix axis perspective change	
W	Direction cosine matrix scale	m
X	Direction cosine matrix transform X-axis	m
Y	Direction cosine matrix transform Y-axis	m
Z	Direction cosine matrix transform Z-axis	m
T_a^b	Denavit-Hartenberg transform matrix	
n	Denavit-Hartenberg transform matrix parameter	
o	Denavit-Hartenberg transform matrix parameter	

a	Denavit-Hartenberg transform matrix parameter	
p	Denavit-Hartenberg transform matrix parameter	
θ_r	Manipulator end effector roll angle	$^{\circ}$ or c
θ_y	Manipulator end effector yaw angle	$^{\circ}$ or c
θ_p	Manipulator end effector pitch angle	$^{\circ}$ or c
A	Simultaneous geometric equation inverse kinematic solution parameter	
B	Simultaneous geometric equation inverse kinematic solution parameter	
C	Simultaneous geometric equation inverse kinematic solution parameter	
D	Simultaneous geometric equation inverse kinematic solution parameter	
E	Simultaneous geometric equation inverse kinematic solution parameter	
F	Simultaneous geometric equation inverse kinematic solution parameter	
α	Robotic manipulator joint 1 angle (earth frame)	
σ	Robotic manipulator joint 2 angle (earth frame)	
η	Robotic manipulator joint 3 angle (earth frame)	
l_1	Robotic manipulator link 1 length (earth frame)	m
l_2	Robotic manipulator link 2 length (earth frame)	m
l_3	Robotic manipulator link 3 length (earth frame)	m
c_1	Robotic manipulator link 1 centre of mass (earth frame)	m
c_2	Robotic manipulator link 2 centre of mass (earth frame)	m

c_3	Robotic manipulator link 3 centre of mass (earth frame)	m
m_1	Robotic manipulator link 1 mass (earth frame)	kg
m_2	Robotic manipulator link 2 mass (earth frame)	kg
m_3	Robotic manipulator link 3 mass (earth frame)	kg
ΣT	Total torque about a joint excluding static friction	Nm
ΣT_{in}	Total torque input to a joint	Nm
ΣM_m	Total moments about a joint caused by mass	Nm
ΣM_f	Total moments about a joint caused by friction	Nm
ΣM_i	Total moments about a joint caused by inertia	Nm
ΣM_c	Total moments about a joint caused by centripetal effects	Nm
τ_s	Torque produced by static friction	Nm
τ_k	Torque produced by kinetic friction	Nm
$\Sigma \tau_f$	Total torque produced by frictional effects	Nm
τ	Torque	Nm
c_s	Coefficient of static friction	
c_k	Coefficient of kinetic friction	
θ	Generic joint angle	$^\circ$ or c
$\dot{\theta}$	Generic angular velocity	$^\circ s^{-1}$ or $^c s^{-1}$
$\ddot{\theta}$	Generic angular acceleration	$^\circ s^{-2}$ or $^c s^{-2}$
$\dot{\alpha}$	Angular velocity of joint 1	$^\circ s^{-1}$ or $^c s^{-1}$
$\ddot{\alpha}$	Angular acceleration of joint 1	$^\circ s^{-2}$ or $^c s^{-2}$
$\dot{\sigma}$	Angular velocity of joint 2	$^\circ s^{-1}$ or $^c s^{-1}$

$\ddot{\sigma}$	Angular acceleration of joint 2	$^{\circ} s^{-2}$ or $^c s^{-2}$
$\dot{\eta}$	Angular velocity of joint 3	$^{\circ} s^{-1}$ or $^c s^{-1}$
$\ddot{\eta}$	Angular acceleration of joint 3	$^{\circ} s^{-2}$ or $^c s^{-2}$
g	Acceleration due to gravity (9.81)	$m s^{-2}$
F_c	Force caused by centripetal effects	N
$\Sigma T'$	Total torque about a joint including frictional effects	Nm
$Y_{I\alpha}$	Ideal response to an input for joint 1	$^{\circ}$ or c
$Y_{I\sigma}$	Ideal response to an input for joint 2	$^{\circ}$ or c
$Y_{I\eta}$	Ideal response to an input for joint 3	$^{\circ}$ or c
$Y_{A\alpha}$	Actual response to an input for joint 1	$^{\circ}$ or c
$Y_{A\sigma}$	Actual response to an input for joint 2	$^{\circ}$ or c
$Y_{A\eta}$	Actual response to an input for joint 3	$^{\circ}$ or c
f	Fitness function	
t_0	Step response initial time	s
t_1	Step response rise time or peak time	s
t_2	Step response settling time	s
t_f	Step response final time	s
Y	Output of an optimisation problem	
n	Number of objectives in an optimisation problem	
r_s	Resolution of spacing between measured points in an obstacle	m
r_{θ}	Angular resolution of the LIDAR sensor	$^{\circ}$ or c
e_{ss}	Steady state error of each joint	$^{\circ}$ or c

e_l	Angular error of the LIDAR sensor	° or °
e_s	Measurement error of servo encoders	° or °

LIST OF TABLES

Table 2-1 Extract from a computational complexity comparison of Lagrangian and Newton-Euler mechanics from Turney et al. (1980)	16
Table 2-2 Comparison of Lagrangian and Newton-Euler methods in relation to a 3-DoF robotic manipulator arm.....	16
Table 2-3 Comparison of algorithm running times.....	37
Table 2-4 Comparison of algorithm running times with values.	37
Table 2-5 Change of path across the environment depending on the size of the weighting factor added to the risk associated with each vertex.....	49
Table 3-1 Parameters of the links in the manipulator arm.....	56
Table 3-2 Parameters of the links in the manipulator arm.....	63
Table 3-3 Parameters of links in a 3-DoF manipulator arm.....	77
Table 4-1 Numerical parameters of the Digital Vanguard 3-DoF manipulator arm.	106
Table 4-2 List of scenarios used for the qualitative validation of the manipulator dynamic model.	106
Table 4-3 Servo motor parameters for use in the manipulator arm dynamic model.	139
Table 5-1 Summary of the algorithmic process for a GA.....	149
Table 5-2 Summary of the variables used in the formation of the fitness function for the robotic manipulator tuning problem.	151
Table 5-3 List of standard optimisation problems used to test optimisation functions.	160
Table 5-4 Graphical representation of the surfaces generated by the optimisation problems listed in Table 5-3.	163

Table 5-5 Numerical results of optimisation method comparison on the robotic manipulator tuning problem for an optimisation with random initial estimated solution.	170
Table 5-6 Numerical data illustrating the possible change in moment values as a result of angle.....	173
Table 5-7 Results of the optimisation process of the manipulator arm for a single angle range displaying the change in best fitness and response to a unit step over successive generations.	174
Table 5-8 Gain Plateaus and Surfaces for the PID controller in joint α	178
Table 5-9 Gain Plateaus and Surfaces for the PID controller in joint σ	183
Table 5-10 Gain Plateaus and Surfaces for the PID controller in joint η	185
Table 6-1 Numerical data showing the adjacency matrix size difference between node graphs of different node spacings.	240
Table 6-2 Change in domain from the Euclidean domain to the Control domain for the end effector start and required end points used in the path generation example for method 1.....	243
Table 7-1: Joint angle combinations to guide the robotic manipulator in such a way as to drive the end effector from a starting position to a required position while providing effective avoidance for the entire manipulator.	250
Table 7-2 Results from a preliminary investigation of the three selected pathing methods for different numbers of obstacles.	253
Table 7-3 List of processes carried out in the guidance algorithm.	272
Table A-1 Full list of optimisation problems used to validate the developed GA.	xiii
Table A-2 Graphical representation of all tested optimisation problems.....	xviii
Table B-3 Parameters for the simulated C-space generation for use in the n-DoF path planning experimentation.	xxix

TABLE OF FIGURES

Figure 1-1 Digital Vanguard ROV (Photo courtesy of Allen Vanguard™).....	2
Figure 1-2 Defender ROV (Photo courtesy of Allen Vanguard™).	2
Figure 1-3 Flow Diagram showing the breakdown of autonomous functions for control of the ROV in question.	3
Figure 1-4 Breakdown of the processes required for an autonomous robotic manipulator arm.	9
Figure 1-5 Breakdown of the processes required for an autonomous robotic manipulator arm, including assumptions made and colour coded to match the remaining chapters of the thesis.	10
Figure 1-6 Project outline of the research presented in this thesis.....	14
Figure 2-1 Wave-based sensor operation using ToF to calculate the distance to an object.....	21
Figure 2-2 Monocular ranging using known vehicle motion to cause disparity between successive image frames.....	23
Figure 2-3 Monocular ranging using two mirrors at a known distance from each other to provide two images, the disparity of which can be compared to calculate object distance.....	24
Figure 2-4 Monocular ranging using a pair of double-sided half-mirror plates a known distance apart can be used to provide two images of the object, the disparity of which can be compared to calculate distance.	25
Figure 2-5 Binocular ranging using two cameras a known distance apart, and measuring the disparity between the image taken by each to calculate the object's distance.....	25
Figure 2-6 Comparison of a Dubins based path with a P-H based path that start and end at the same position and pose.....	31

Figure 2-7 Example of a node graph (not to scale). Each node is numbered and the vertices between them are labelled with letters.	34
Figure 2-8 Example of an environment where graph theory can be used to generate a trajectory.	39
Figure 2-9 A simple node graph.	40
Figure 2-10 A node graph with more complexity than that of Figure 2-9.	41
Figure 2-11 A node graph with three-dimensional coordinated.	43
Figure 2-12 The three-dimensional node graph from Figure 2-11 flattened out into a two-dimensional node graph.	44
Figure 2-13 Example of an environment where graph theory can be used to generate a trajectory with energy costs as well as distance.	45
Figure 2-14 Example of an environment where graph theory can be used to generate a trajectory with risk associated costs.	47
Figure 3-1 Manipulator arm kinematics (red) in relation to the overall guidance method.	53
Figure 3-2 Schematic of the variables and dimensions that make up the simple model of the Digital Vanguard ROV manipulator arm.	56
Figure 3-3 Graphical representation of the robotic arm. The red dotted line it's the representation of the arm when fully extended.	57
Figure 3-4 Diagram of the calculation of α using Pf and Po	58
Figure 3-5 Diagram of the calculation of β and γ using Pf and $P1$	59
Figure 3-6 Configuration of the first link in the manipulator arm.	61
Figure 3-7 Configuration of the second and third links in the manipulator arm.	61
Figure 3-8 Schematic of the variables and dimensions that make up the model of the manipulator arm.	62
Figure 3-9 Rotation of the X_0 and Y_0 axes about the Z_0 axis by α to transform the axes in to the intermediate axes X_0' , Y_0' and Z_0'	64

Figure 3-10 Rotation of the Y_0' and Z_0' axes about the X_0' axis by 90° into the X_1 , Y_1 and Z_1 axes.	65
Figure 3-11 Rotation of the $Xa Ya$ axes around the Za axis by β or γ into the Xb, Yb and Zb axes.	67
Figure 3-12: Schematic of the variables and dimensions that make up the simple model of the Digital Vanguard ROV manipulator arm.....	77
Figure 4-1 Manipulator arm dynamics (green) in relation to the overall guidance method.	83
Figure 4-2 Schematic of a 3-DoF manipulator arm showing the relevant parameters for the development of a dynamic model.....	85
Figure 4-3 Effect of angular velocity on static and kinetic friction.	87
Figure 4-4 Schematic of the weight of link 2 acting on link 2.....	89
Figure 4-5 Schematic of the weight of link 3 acting on link 2.....	90
Figure 4-6 Schematic of the centripetal force which acts on the 2 nd link due to its rotation about the 1 st joint.	92
Figure 4-7 Schematic of the centripetal force which acts on the 2 nd link due to the rotation of the 3rd link about the 1 st joint.	93
Figure 4-8 Schematic of the centripetal force caused by the rotation of the 3 rd link about the 2 nd joint.	94
Figure 4-9 Schematic of the centripetal force caused by the rotation of link 3 about joint 3.....	95
Figure 4-10 Schematic of the moment about joint 3 caused by the weight of link 3. 97	
Figure 4-11 Schematic of the centripetal force caused by the rotation of link 3 about joint 1.....	98
Figure 4-12 Schematic of the centripetal force caused by the rotation of the 3rd link about the 2nd joint.....	99
Figure 4-13 Simulink Control block diagram for the 3-DoF arm dynamic model. ...	101

Figure 4-14 Moments about σ	102
Figure 4-15 Moments about η	102
Figure 4-16 Angular position of each joint.	102
Figure 4-17 Effect of torque based saturation on friction in a joint.	104
Figure 4-18 Angular position of each joint in the arm model for scenario 1.....	108
Figure 4-19 Resultant joint angle locations for scenario 1 in relation to the starting geometry.	109
Figure 4-20 Angular position of each joint in the arm model for scenario 1 with static friction removed.....	109
Figure 4-21 Moment terms about σ (excluding input torque) for scenario 1.....	110
Figure 4-22 Moment terms about η (excluding input torque) for scenario 1.....	111
Figure 4-23 Angular position of each joint in the arm model for scenario 2.....	112
Figure 4-24 Angular position of σ for scenario 2.	112
Figure 4-25 Angular position of η for scenario 2.	112
Figure 4-26 Moment in σ (excluding input torque) for scenario 2.....	113
Figure 4-27 Moment in η (excluding input torque) for scenario 2.....	113
Figure 4-28 Angular position over time of all joints in scenario 3.	114
Figure 4-29 Angular position over time of all joints in scenario 4.	115
Figure 4-30 Angular position of all joints in scenario 4 over 15 seconds.....	116
Figure 4-31 Diagram illustrating the angular velocity over time of all joints in scenario 4.....	117
Figure 4-32 Angular velocity of joint 2 in scenario 4.....	117
Figure 4-33 Angular velocity of joint 3 in scenario 4.....	117
Figure 4-34 Moment terms about α (excluding input torque) for the scenario 4.	118

Figure 4-35 Moment terms about σ (excluding input torque) for scenario 4.	118
Figure 4-36 Moment terms about η (excluding input torque) for scenario 4.	119
Figure 4-37 Input torque to all joints for a modified version of scenario 4.	120
Figure 4-38 Angular position of all joints for a modified version of scenario 4.	120
Figure 4-39 Angular position of joint α for a modified version of scenario 4.	121
Figure 4-40 Angular position of joints σ and η for a modified version of scenario 4.	121
Figure 4-41 Angular position over time of all joints in scenario 5.	122
Figure 4-42 Moment terms about α (excluding input torque) for scenario 5.	123
Figure 4-43 Moment terms about σ (excluding input torque) for scenario 5.	124
Figure 4-44 Moment terms about η (excluding input torque) for scenario 5.	124
Figure 4-45 Angular position of each joint in scenario 6.	126
Figure 4-46 Angular acceleration of each joint in the arm during scenario 6.	126
Figure 4-47 Angular position of each of the joints during scenario 7.	127
Figure 4-48 Moment terms about σ (excluding input torque) for scenario 7.	128
Figure 4-49 Moment terms about η (excluding input torque) for scenario 7.	128
Figure 4-50 Angular position of each joint during scenario 8.	130
Figure 4-51 Moment terms about σ (excluding input torque) for scenario 8.	131
Figure 4-52 Moment terms about η (excluding input torque) for scenario 8.	131
Figure 4-53 Angular position of each joint during scenario 9.	132
Figure 4-54 Angular velocity of each joint during scenario 9.	133
Figure 4-55 Angular acceleration of each joint during scenario 9.	133
Figure 4-56 Moment terms about σ for scenario 9.	134
Figure 4-57 Moment terms about η for scenario 9.	134

Figure 4-58 Moment terms about σ for scenario 4.	134
Figure 4-59 Moment terms about η for scenario 4.	134
Figure 4-60 Input torques to each joint for scenario 10.	135
Figure 4-61 Angular position of each joint for scenario 10.	135
Figure 4-62 Angular position of σ for scenario 10.	136
Figure 4-63 Angular position of η for scenario 10.....	136
Figure 4-64 Circuit diagram of an electric motor drive connected to a load via a gearbox.	137
Figure 4-65 Architecture of Robotic Manipulator Joint Servos with Non-Linear Dynamic Model as the Load.....	139
Figure 4-66 Simulink control block diagram for the dynamic model embedded into a servo loop with the specified motor model.	140
Figure 4-67 Architecture of the PID controlled servo motors surrounding the manipulator arm dynamic model.	141
Figure 5-1 Manipulator arm control and path following (blue) in relation to the overall guidance method.....	143
Figure 5-2 A single cell containing two chromosomes.....	145
Figure 5-3 A cell undederging out asexual reproduction by mitosis.	146
Figure 5-4 Crossover of genetic material from two chromosomes during the first stage of meiosis.	146
Figure 5-5 Crossover in two parent cells during the process of meiosis. The two parents have different coloured chromosomes.	147
Figure 5-6 Production of reproductive cells in both parents during meiosis.	147
Figure 5-7 Formation of children cells by combination of reproductive cells from each parent cell.....	147

Figure 5-8 Result of the optimisation of a joint angle range in the arm using the sum squared error fitness function.....	153
Figure 5-9 Selection of the value t_1 in relation to the transient and steady state regions of a step response.	154
Figure 5-10 Result of the optimisation of a joint angle range in the arm using the Weighted Sum of Errors-Squared (Transient and Steady-State) fitness function...	155
Figure 5-11 Selection of the values t_1 and t_2 in relation to the rise time, settling time and steady state regions of a step response.....	156
Figure 5-12 Result of the optimisation of a joint angle range in the arm using the Weighted Sum of Errors-Squared (Rise Time, Overshoot and Steady-State) fitness function.....	157
Figure 5-13 Shape of the weighting function for the Gaussian and Time based fitness function.....	158
Figure 5-14 Result of the optimisation of a joint angle range in the arm using the Gaussian and Time Base Weighting fitness function.	159
Figure 5-15 Linear results for achieved fitness and runtime for three optimisation methods over all of the optimisation problems.	166
Figure 5-16 Logarithmic results for achieved fitness and runtime for three optimisation methods over all of the optimisation problems.	167
Figure 5-17 Graphical results of optimisation method comparison on the robotic manipulator tuning problem.....	170
Figure 5-18 Control block diagram illustrating the implementation of PID control into the dynamic model of robotic manipulator and servo drive. This block diagram also includes a torque correction factor for moments caused by weight on the arm.....	172
Figure 5-19 Moments of inertia about joint α given changes in σ and η	179
Figure 5-20 Manipulator arm geometries which generate the largest moments of inertia about α	179
Figure 5-21 PID Control gains for joint α overlaid.	180

Figure 5-22 Moments of inertia and PID gains for α .	181
Figure 5-23 Comparison of moments of inertia of joint σ with the PID control gains for the same joint.	184
Figure 5-24 Comparison of moments of inertia of joint η with the PID control gains for the same joint.	186
Figure 5-25 Control block diagram for the controlled servo and dynamic model system with the implemented gain scheduler.	187
Figure 5-26 Results of the random step sequence testing of the controlled dynamic model. Plot 1 is joint angle against time for each joint and plot 2 is the angular error against time for each joint.	188
Figure 5-27 PID control gains for each joint over the random step test sequence carried out above.	189
Figure 5-28 Zoomed in plot of joint angles against time and error against time for the random step test sequence.	190
Figure 5-29 PID control gains of joint α in the robotic manipulator during the random step testing of the manipulator arm.	191
Figure 5-30 PID control gains of joint σ in the robotic manipulator during the random step testing of the manipulator arm.	192
Figure 5-31 PID control gains of joint η in the robotic manipulator during the random step testing of the manipulator arm.	193
Figure 5-32 Results of the random step sequence testing of the controlled dynamic model. Plot 1 is joint angle against time for each joint and plot 2 is the angular error against time for each joint.	194
Figure 5-33 Zoomed in plot of joint angles against time and error against time for the random step test sequence.	194
Figure 5-34 Position and velocity change for a system at different points in time, especially when tending towards steady-state.	195

Figure 6-1 Environment and mapping in C-Space (purple) in relation to the overall guidance method.....	198
Figure 6-2 Change of measured point spacing with sensor angular resolution and range.....	199
Figure 6-3 Change of distance between measured points with distance from origin and angular resolution.....	201
Figure 6-4 Distribution of inspection points across one polygon. The red points represent the polygon corners, the green point represents the polygon centre and the blue points represent the other inspection points.	203
Figure 6-5: Different cases of intersection of polygons with the plane of operation of the 1 st link in the manipulator.....	210
Figure 6-6: Geometries of (a) two-link and (b) three-link arm configurations generated by random demand end effector positions as inspection points.	217
Figure 6-7: Positions of the randomly-generated points used to calculate arm geometries in Figure 6-6. The green points fall within the accessible range of each arm configuration, whereas the red points are inaccessible.....	217
Figure 6-8 Arm geometry for the calculation of σ	220
Figure 6-9 Arm geometry for the calculation of the σ and η joint angle combination.	221
Figure 6-10: Impermissible region for a triangular polygon situated in the arm extension range for all three links.....	225
Figure 6-11: Impermissible region formed by inspection of a single polygon, as seen from various directions.	226
Figure 6-12 Effect of resolution of the LIDAR sensor on the detection of the corners of an obstacle.....	228
Figure 6-13: Expansion of the impermissible region to create permissible boundary around the impermissible region seen from various directions.....	230

Figure 6-14 A 2-D pattern of inspection points for illustration of the convex hull technique for bounding a region of points.	232
Figure 6-15 The same 2-D patter of inspection points has now been bounded using the convex hull technique.....	232
Figure 6-16 Method by which the complex hull technique works.	233
Figure 6-17 Limitations of the convex hull technique and the result of using the alpha hull technique.	233
Figure 6-18 Limitations and compromise required when using the alpha volume technique.....	233
Figure 6-19: Permissible boundary surface converted into a wire frame representing the vertices in a node graph in the three servo angle dimensions.....	237
Figure 6-20 A sphere in T-space in relation to the base point of the manipulator arm.	239
Figure 6-21 C-space representation of the sphere in relation to the manipulator arm.	239
Figure 6-22 Sphere in C-space (red) overlaid over the unmodified node graph (black).	241
Figure 6-23 Nodes which collide with the sphere in C-space (red) to be removed from the remainder of the node graph (blue).....	241
Figure 6-24 Surface plot of a triangulation showing the complete node graph with the colliding nodes removed.....	242
Figure 7-1 Flowchart showing the context of the path generation (orange) in relation to the overall guidance method.	247
Figure 7-2: Generated path (red line) from an end effector start point (green circle) in the servo control domain to a desired end effector end point (blue circle) in the servo control domain.....	248
Figure 7-3: Generated path between the end effector start point (green circle) to the desired end effector end point (blue circle).	249

Figure 7-4 Spherical object in C-space with a safe path generated around it.	251
Figure 7-5 Algorithm run time for the selected path planning algorithms for different numbers of obstacles	255
Figure 7-6 Results of the path generation around two obstacles with 0.47 m, 0.48m and 0.49m radii using the Bellman-Ford Algorithm.	257
Figure 7-7 Results of the path planning around 4 obstacles with 0.475 m radius using the Bellman-ford algorithm.	259
Figure 7-8 Results of the path generation around two obstacles with 0.47 m, 0.48m and 0.49m radii using the Breadth-first Algorithm.	261
Figure 7-9 Results of the path planning around 4 obstacles with 0.475 m radius using the Breadth-first algorithm.	262
Figure 7-10 Results of the path generation around two obstacles with 0.47 m, 0.48m and 0.49m radii using Dijkstra's Algorithm.	263
Figure 7-11 Results of the path planning around 4 obstacles with 0.475 m radius using Dijkstra's algorithm.	264
Figure 7-12 Time response of 3 different transfer functions with time constants of 0.5, 0.25 and 0.167 seconds.	266
Figure 7-13 Time response of systems G1 , G2 and G3 given an input which is equivalent to the step response of system R	266
Figure 7-14 Time response of 3 different transfer functions with similar time constants of 0.4 , 0.3 and 0.286 seconds.	267
Figure 7-15 Time response of systems G4 , G5 and G6 given an input which is equivalent to the step response of system R	268
Figure 7-16 Path through space and the trajectory which represents the same path but with time information included.	269
Figure 7-17 Behaviour of systems G4 , G5 and G6 as dimension components of a path tracker when given the path as an input or the trajectory as an input.	270

Figure 7-18 Time response of each dimension to a demanded path and demanded trajectory.	271
Figure 8-1 Results of simulation 1 showing the planned (a) and actual (b) path of the manipulator through T-space, the planned and actual path through C-space (b) and the locations of each joint in time (d).....	280
Figure 8-2 Demand and actual C-space paths with a waypoint tolerance of 0.5° in scenario 1.....	281
Figure 8-3 Demand path and trajectory, actual path and tracking error over time for a waypoint tolerance of 0.5° in scenario 1.	282
Figure 8-4 Results of simulation 2 showing the planned (a) and actual (b) path of the manipulator through T-space, the planned and actual path through C-space (b) and the locations of each joint in time (d).....	284
Figure 8-5 Demand and actual C-space paths with a waypoint tolerance of 0.5° in scenario 2.....	285
Figure 8-6 Demand path and trajectory, actual path and tracking error over time for a waypoint tolerance of 0.5° in scenario 2.	285
Figure 8-7 Time response of demand path and trajectory and actual manipulator path.	286
Figure 8-8 Tracking error of manipulator against trajectory.....	286
Figure 8-9 Tracking error of manipulator joints in scenario 2. The highlighted areas show where the three joint errors do not match up in time, predicting divergence from the demand path.	287
Figure 8-10 C-space demand path for scenario 2 with the actual path plotted only for the time ranges highlighted in Figure 8-9.	287
Figure 8-11 Results of simulation 3 showing the planned (a) and actual (b) path of the manipulator through T-space, the planned and actual path through C-space (b) and the locations of each joint in time (d).....	290

Figure 8-12 Tracking error of manipulator joints in scenario 3. The highlighted areas show where the three joint errors do not match up in time, predicting divergence from the demand path.	291
Figure 8-13 C-space demand path for scenario 2 with the actual path plotted only for the time ranges highlighted in Figure 8-12.	291
Figure 8-14 Plot of planned C-space trajectory and actual path over time. Blue at the start and red at the end.	292
Figure 8-15 Tracking error over the time range shown in Figure 8-14.....	292
Figure 8-16 Demand and actual C-space paths with a waypoint tolerance of 0.5° in scenario 3.....	292
Figure 8-17 Demand path and trajectory, actual path and tracking error over time for a waypoint tolerance of 0.5° in scenario 3.	293
Figure 8-18 Tracking error of manipulator joints in scenario 3. The highlighted areas show where the three joint errors do not match up in time, predicting divergence from the demand path.	293
Figure 8-19 C-space demand path for scenario 2 with the actual path plotted only for the time ranges highlighted in Figure 8-18.	294
Figure 8-20 Results of simulation 4 showing the planned (a) and actual (b) path of the manipulator through T-space, the planned and actual path through C-space (b) and the locations of each joint in time (d).	296
Figure 8-21 Tracking error of manipulator joints in scenario 4. The highlighted areas show where the three joint errors do not match up in time, predicting divergence from the demand path.	297
Figure 8-22 C-space demand path for scenario 2 with the actual path plotted only for the time ranges highlighted in Figure 8-21.	297
Figure 8-23 Demand and actual C-space paths with a waypoint tolerance of 0.5° in scenario 2.....	298

Figure 8-24 Demand path and trajectory, actual path and tracking error over time for a waypoint tolerance of 0.5° in scenario 2.	298
Figure 8-25 Results of simulation 5 showing the planned (a) and actual (b) path of the manipulator through T-space, the planned and actual path through C-space (b) and the locations of each joint in time (d).	300
Figure 8-26 Tracking error of manipulator joints in scenario 5. The highlighted areas show where the three joint errors do not match up in time, predicting divergence from the demand path.	300
Figure 8-27 C-space demand path for scenario 2 with the actual path plotted only for the time ranges highlighted in Figure 8-26.	301
Figure 8-28 Actual path of manipulator arm in scenario 5 with a waypoint tolerance of 0.5°	302
Figure 8-29 Time based plots for planned path and trajectory, actual path and tracking errors in scenario 5 with a waypoint tolerance of 0.5°	302
Figure 8-30 Planned and actual C-space paths for the manipulator arm in scenario 5 with a waypoint tolerance of 0.5°	303
Figure 8-31 Results of simulation 6 showing the planned (a) and actual (b) path of the manipulator through T-space, the planned and actual path through C-space (b) and the locations of each joint in time (d).	304
Figure 8-32 Tracking error of manipulator joints in scenario 6. The highlighted areas show where the three joint errors do not match up in time, predicting divergence from the demand path.	305
Figure 8-33 C-space demand path for scenario 2 with the actual path plotted only for the time ranges highlighted in Figure 8-32.	306
Figure 8-34 Actual path of manipulator arm in scenario 5 with a waypoint tolerance of 0.5°	306
Figure 8-35 Time based plots for planned path and trajectory, actual path and tracking errors in scenario 5 with a waypoint tolerance of 0.5°	307

Figure 8-36 Planned and actual C-space paths for the manipulator arm in scenario 5 with a waypoint tolerance of 0.5° .	307
Figure A-1 Alternate representation of optimisation technique comparison (linear).	xxii
Figure A-2 Alternate representation of optimisation technique comparison (logarithmic).	xxiii
Figure A-3 Further PID controller testing results.	xxiv
Figure A-4 Further PID controller testing results.	xxiv
Figure A-5 Further PID controller testing results.	xxiv
Figure A-6 Further PID controller testing results.	xxiv
Figure A-7 Further PID controller testing results.	xxiv
Figure A-8 Further PID controller testing results.	xxiv
Figure A-9 Further PID controller testing results.	xxv
Figure A-10 Further PID controller testing results.	xxv
Figure A-11 Further PID controller testing results.	xxv
Figure A-12 Further PID controller testing results.	xxv
Figure A-13 Further PID controller testing results.	xxv
Figure A-14 Further PID controller testing results.	xxv
Figure A-15 Further PID controller testing results.	xxvi
Figure A-16 Further PID controller testing results.	xxvi
Figure A-17 Further PID controller testing results.	xxvi
Figure A-18 Further PID controller testing results.	xxvi
Figure A-19 Further PID controller testing results.	xxvi
Figure A-20 Further PID controller testing results.	xxvi
Figure A-21 Further PID controller testing results.	xxvii

Figure A-22 Further PID controller testing results. xxvii

Figure A-23 Further PID controller testing results. xxvii

Figure A-24 Further PID controller testing results. xxvii

Figure A-25 Further PID controller testing results. xxvii

Figure B-26 Results from a number of runs of the path planning algorithm for 2 to 9 degrees-of-freedom..... xxix

Figure B-27 Enlarged view of Figure B-26 to display only the results for path planning in 2 to 6 degrees-of-freedom.....xxx

TABLE OF EQUATIONS

(2.1).....	35
(2.2).....	40
(2.3).....	41
(2.4).....	42
(2.5).....	42
(2.6).....	43
(2.7).....	46
(2.8).....	48
(2.9).....	48
(2.10).....	48
(2.11).....	48
(3.1).....	56
(3.2).....	57
(3.3).....	58
(3.4).....	59
(3.5).....	59
(3.6).....	60
(3.7).....	60
(3.8).....	60
(3.9).....	63
(3.10).....	65

(3.11).....	65
(3.12).....	66
(3.13).....	66
(3.14).....	66
(3.15).....	67
(3.16).....	67
(3.17).....	68
(3.18).....	68
(3.19).....	68
(3.20).....	68
(3.21).....	69
(3.22).....	69
(3.23).....	69
(3.24).....	70
(3.25).....	70
(3.26).....	71
(3.27).....	71
(3.28).....	71
(3.29).....	72
(3.30).....	72
(3.31).....	72
(3.32).....	72
(3.33).....	72
(3.34).....	72

(3.35).....	73
(3.36).....	73
(3.37).....	73
(3.38).....	73
(3.39).....	73
(3.40).....	74
(3.41).....	74
(3.42).....	74
(3.43).....	74
(3.44).....	75
(3.45).....	75
(3.46).....	75
(3.47).....	75
(3.48).....	75
(3.49).....	76
(3.50).....	77
(3.51).....	78
(3.52).....	78
(3.53).....	78
(3.54).....	78
(3.55).....	78
(3.56).....	78
(3.57).....	79
(3.58).....	79

(3.59).....	79
(3.60).....	79
(3.61).....	80
(3.62).....	80
(3.63).....	80
(4.1).....	85
(4.2).....	86
(4.3).....	87
(4.4).....	87
(4.5).....	87
(4.6).....	88
(4.7).....	88
(4.8).....	88
(4.9).....	88
(4.10).....	88
(4.11).....	89
(4.12).....	89
(4.13).....	89
(4.14).....	90
(4.15).....	90
(4.16).....	91
(4.17).....	91
(4.18).....	91
(4.19).....	91

(4.20).....	92
(4.21).....	92
(4.22).....	93
(4.23).....	94
(4.24).....	94
(4.25).....	95
(4.26).....	96
(4.27).....	96
(4.28).....	97
(4.29).....	97
(4.30).....	97
(4.31).....	98
(4.32).....	98
(4.33).....	99
(4.34).....	99
(4.35).....	100
(4.36).....	100
(4.37).....	100
(4.38).....	101
(4.39).....	103
(4.40).....	103
(4.41).....	103
(4.42).....	103
(4.43).....	105

(4.44).....	125
(4.45).....	127
(4.46).....	127
(4.47).....	137
(4.48).....	137
(4.49).....	138
(4.50).....	138
(4.51).....	138
(4.52).....	138
(5.1).....	152
(5.2).....	154
(5.3).....	156
(5.4).....	157
(5.5).....	158
(5.6).....	166
(6.1).....	200
(6.2).....	200
(6.3).....	200
(6.4).....	200
(6.5).....	200
(6.6).....	202
(6.7).....	203
(6.8).....	203
(6.9).....	204

(6.10).....	204
(6.11).....	204
(6.12).....	205
(6.13).....	205
(6.14).....	207
(6.15).....	207
(6.16).....	207
(6.17).....	208
(6.18).....	208
(6.19).....	208
(6.20).....	208
(6.21).....	211
(6.22).....	211
(6.23).....	211
(6.24).....	212
(6.25).....	212
(6.26).....	213
(6.27).....	213
(6.28).....	213
(6.29).....	214
(6.30).....	214
(6.31).....	214
(6.32).....	214
(6.33).....	215

(6.34).....	215
(6.35).....	218
(6.36).....	219
(6.37).....	219
(6.38).....	220
(6.39).....	220
(6.40).....	221
(6.41).....	222
(6.42).....	222
(6.43).....	222
(6.44).....	222
(6.45).....	223
(6.46).....	223
(6.47).....	229
(6.48).....	236
(7.1).....	253
(7.2).....	253
(7.3).....	265

1 INTRODUCTION

In the field of Defence Engineering, there are applications which require the use of highly accurate robotic manipulator arms in close-proximity environments, especially those of Improvised Explosive Device (IED) disposal (Beltran-Gonzalez, et al., 2007), (Dubowsky & Vance, 1989), (Nguyen & Bott, 2000), (Goldenberg, et al., 2000). In this thesis a method of path generation for a 3-degree-of-freedom (3-DoF) robotic manipulator arm with the purpose of IED disposal is developed that can in real-time plan a path through a close-proximity environment, avoiding collisions with obstacles by all parts of the manipulator arm. Environment data are not required to be known a priori and the technique has the potential to generate paths for n-DoF manipulator arms. Since the application of the technique is for IED disposal, and IEDs tend to be installed in concealed locations with very little access space and are designed to be very unstable, the technique has the capability of approaching and tracking the edges of obstacles and environments rather than completely avoiding them.

1.1 Background to Research

This research carried out in this thesis was partially sponsored by the defence-solutions company Allen Vanguard™. Allen Vanguard™ researches and manufactures countermeasures against hazardous threats. These threats include various explosive devices and chemical, biological, radiological and nuclear agents. The company produces various products to counter these hazards. These include two remotely operated vehicles (ROVs): Digital Vanguard (Figure 1-1) and Defender (Figure 1-2). Digital Vanguard is a 56 kg tracked vehicle with an electronic servo-controlled robotic arm and three cameras installed in various locations. Defender is a 275 kg six-wheeled, skid-steer vehicle, with a heavier duty hydraulic-powered robotic arm and six cameras with varied functionality.



Figure 1-1 Digital Vanguard ROV (Photo courtesy of Allen Vanguard™).

These robots require a highly skilled operator to control them as they do not have installed any sensors other than cameras and so provide very little information to the operator other than the visual feeds. In the case of the smaller robot, Digital Vanguard, without any sensors indicating its position, the robotic arm can be unintentionally operated outside of its designed envelope, to the point where it is able to cut the cables controlling it. For this reason the operator has to divert a lot of their time and attention to monitoring the state of the robot rather than carrying out the primary mission of IED detection and disposal.



Figure 1-2 Defender ROV (Photo courtesy of Allen Vanguard™).

The diagram in Figure 1-3 illustrates the breakdown of the functions required to provide autonomous control of the Digital Vanguard ROV. This control can be split into two main areas, control of the skid-steer drive unit, and control of the manipulator arm. In each case simultaneous localisation and mapping (SLAM) must take place, and also path planning to avoid any obstacles. This path must then be followed by the vehicle in order to reach the required target.

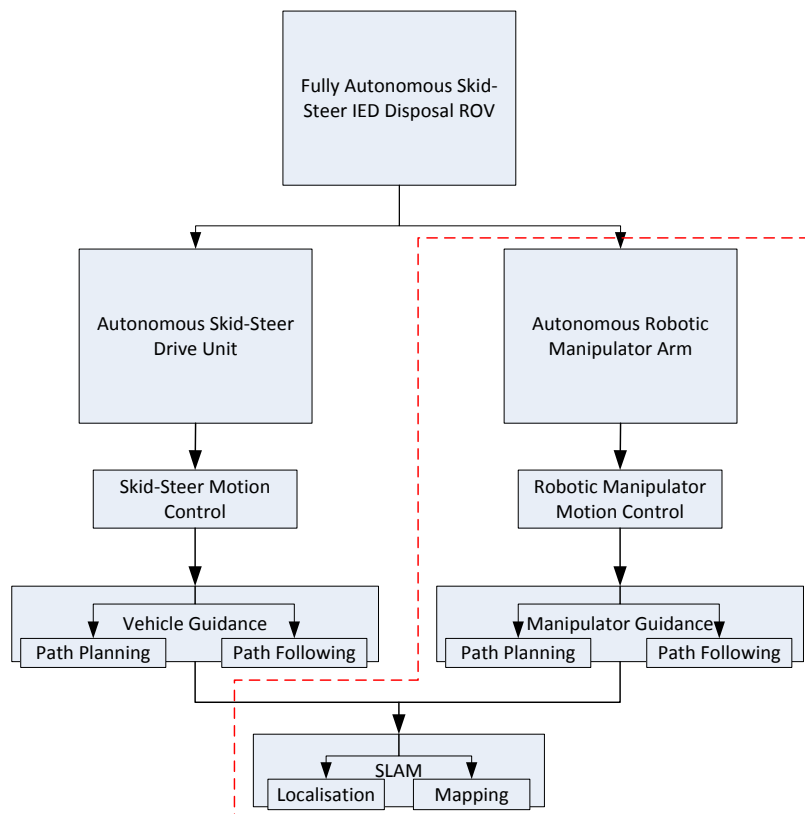


Figure 1-3 Flow Diagram showing the breakdown of autonomous functions for control of the ROV in question.

Achieving fully autonomous navigation of a vehicle of this type would require a significant amount of work, far in excess of the scope of this project. Therefore the work presented in this thesis will focus on the subsection of Figure 1-3 which is highlighted in red to bring autonomous navigation to the robotic manipulator arm.

This project will focus solely on the autonomous navigation of the robotic manipulator which is attached to the vehicle. The one aspect of the drive unit that will be taken into consideration is that the motion of the vehicle means that the arm will not remain in the same location, and therefore the environment will change. This means that knowledge of the environment may not be known a priori and the guidance algorithm for the arm will have to work in real time to continually provide a safe path for the manipulator arm to follow around obstacles in the environment.

Autonomous operation of the robotic manipulator requires enough accuracy to be able to navigate safely through a close-proximity environment without the risk of collision. For operation of the manipulator to carry out the disarmament of an IED, the environment is required to be mapped in much more detail than is needed for navigation with the vehicle platform. This can be done for a much smaller area since the vehicle must navigate over hundreds of metres of terrain, but the arm is only needed to disarm the IED in the final stage of the mission. The area in which the manipulator can move needs to be mapped, and obstacle avoidance needs to be carried out effectively so that collisions are very unlikely to take place, and this is imperative when sensitive explosive material may be present. Path planning through the environment to place the manipulator in the correct location to carry out its function is required, and subsequent following of the planned path needs to have a near zero number of errors.

Path planning techniques exist for robotic manipulators in various configurations and sizes, and in many different applications, but there is very little work reported in the literature that could be applicable to a scenario where the arm needs real-time path planning in a changing environment due to its mobility, and requires the ability to safely navigate in a new, heavily constrained space every single time it is used. This means that in the application of robotics for IED detection and disposal there is a need to develop a technique which can satisfy these requirements. Given that one of these requirements is for the robotic arm to be manoeuvred through potentially very tight spaces, consideration must be given to the position and trajectory of each link in the arm, not just the end effector.

1.2 Research Question

Having considered the background laid out in section 1.1, and the narrowing of scope to an achievable subset of the overall problem of navigational autonomy in the vehicles presented, the following research question is proposed:

“Is it possible and feasible to implement a path-generation algorithm that is capable of guiding a robotic manipulator arm through a close-proximity environment with the aim of carrying out Improvised Explosive Device disposal missions?”

To adequately answer this question, the aim and objectives discussed in section 1.3 have been determined.

1.3 Aims and Objectives

The aim of the work reported in this thesis is to develop a method of path planning for mobile 3-DoF manipulators that can plan a safe path in real-time through an environment around obstacles and towards a target object. This path will consider not just for the end effector but the manipulator arm in its entirety.

To satisfy this aim, several objectives must be met. In order to test any guidance method, either a software or hardware model must be implemented so that any planned path through an environment can be tested for feasibility, in essence validating the guidance method. Since a simulated system should be fully predictable given that it is governed by a set of known equations, a dynamic model of a 3-DoF manipulator arm will be developed for use in the majority of the work carried out in this thesis.

Once a predictable dynamic model has been developed, a suitable control schema will be selected and tuned to provide system behaviour which is capable of following a planned path. It is also important to know the limitations of the controlled system as any path generation technique must be designed to take into account such limitations.

The main focus of this thesis is to develop a guidance method which is capable of planning a safe path through close-proximity environments. Obstacle data are required to be modelled in an accurate way, taking into account any assumptions made for ease of simulation. This ensures that a usable and realistic data set is available for use in the mapping and path planning stages of the arm guidance.

Once data about obstacles in the environment have been obtained, a usable map of such data must be built in an appropriate way for the path planning to be carried out. Finally, a method of planning safe paths through the environment must be investigated, taking into account the limitations of the arm and sensor data.

Following the development of a guidance method the developed method must be validated and assessed for its limitations. This gives the aim and a series of objectives as follows:

Aim:

- Develop a method of planning a safe path for a 3-DoF robotic manipulator arm in 3-D close-proximity environments.

Objectives:

- Derivation of a dynamic model of a 3-DoF robotic manipulator arm.
- Implementation of a suitable control schema for such a dynamic system.
- Development of a guidance method for safe navigation of the controlled dynamic model through close-proximity environments to include
 - realistic obstacle data of increasing complexity;
 - creation of a suitable environment map;
 - development of a suitable path-generation method, taking into account arm and mapping and localisation sensor limitations.
- Validation of the guidance method to assess its strengths and limitations.

The aim and objectives set out in this section will be achieved by combining together in a novel way methods of robotic manipulator kinematic and dynamic modelling, manipulator control, environment mapping and path planning and following as outlined in section 1.4.

1.4 Project Outline

With the above research question and the suggested aim and objectives in mind a novel concept for navigational autonomy in the manipulator arm is developed. This concept uses localisation sensor data to create a map of the environment. It then uses a method of path generation and following to guide the manipulator arm through this environment, avoiding obstacles in the way, to reach the mission objective. In the case of the Digital Vanguard ROV, which has a 3 degree-of-freedom manipulator arm, path generation is required for each of the three arm links to guide the entire arm through the environment. This means that three 3-D paths must be generated. In the case of much larger robotic arms, with higher degrees of freedom, this would require a 3-D path to be generated for each link in the arm. In many high-degree-of-freedom robotic arms the solution to the multiple 3-D path generation problem is to plot a path for the first link in the arm limiting the area in which the entire arm operates to avoid obstacles. Every successive link then follows the same path therefore only one path is generated through space. The technique proposed in this thesis has the advantage of only having to generate one path that will drive each link to follow a safe path to the goal. In close-proximity environments the tight-volume-constraints that exist during the mission could mean that there may not always be a solution that allows the arm to manoeuvre in such a way as to achieve its objective, and therefore the proposed technique or any other may fail to find a safe path through the environment.

To calculate the range of joint angle combinations of the Digital Vanguard manipulator arm that cause collisions with obstacles in the environment, the Euclidean coordinates of the obstacles are passed through an inverse kinematic model of the arm. In the case of the three-degree-of-freedom arm, this provides a three-dimensional set of data specifying the joint angles where collisions occur. This 3-D data set contains the collision range of the arm with the environment in terms of the joint angles, which are the direct control requirements of the arm. The data set can be considered to be a map of the environment with the obstacles specified in terms of the control requirements of the arm.

The Euclidean coordinates of the manipulator end effector at its starting position and at the required end position can also be passed through the kinematic model to calculate these two locations in terms of the newly acquired map. With these two points in the control domain, a path can be plotted from the end effector starting point to the end location, avoiding all obstacles. This path is now represented in terms of the direct control requirements of the robotic arm, which is useful for guiding the end effector to its mission objective.

This method allows for the possible implementation of obstacle avoidance for the entire arm, and provides enough information for path planning of the arm in very strict volume constraints, allowing the end effector to be placed in the desired location, having been manoeuvred through very tight spaces to get there. This method only requires one path to be generated for the entire manipulator arm.

The flowchart in Figure 1-4 shows a breakdown of the processes required to get from the initial sensor and user inputs to a path which can then be followed by the arm implementation, which in this case will be a dynamic model of a 3-DoF manipulator arm.

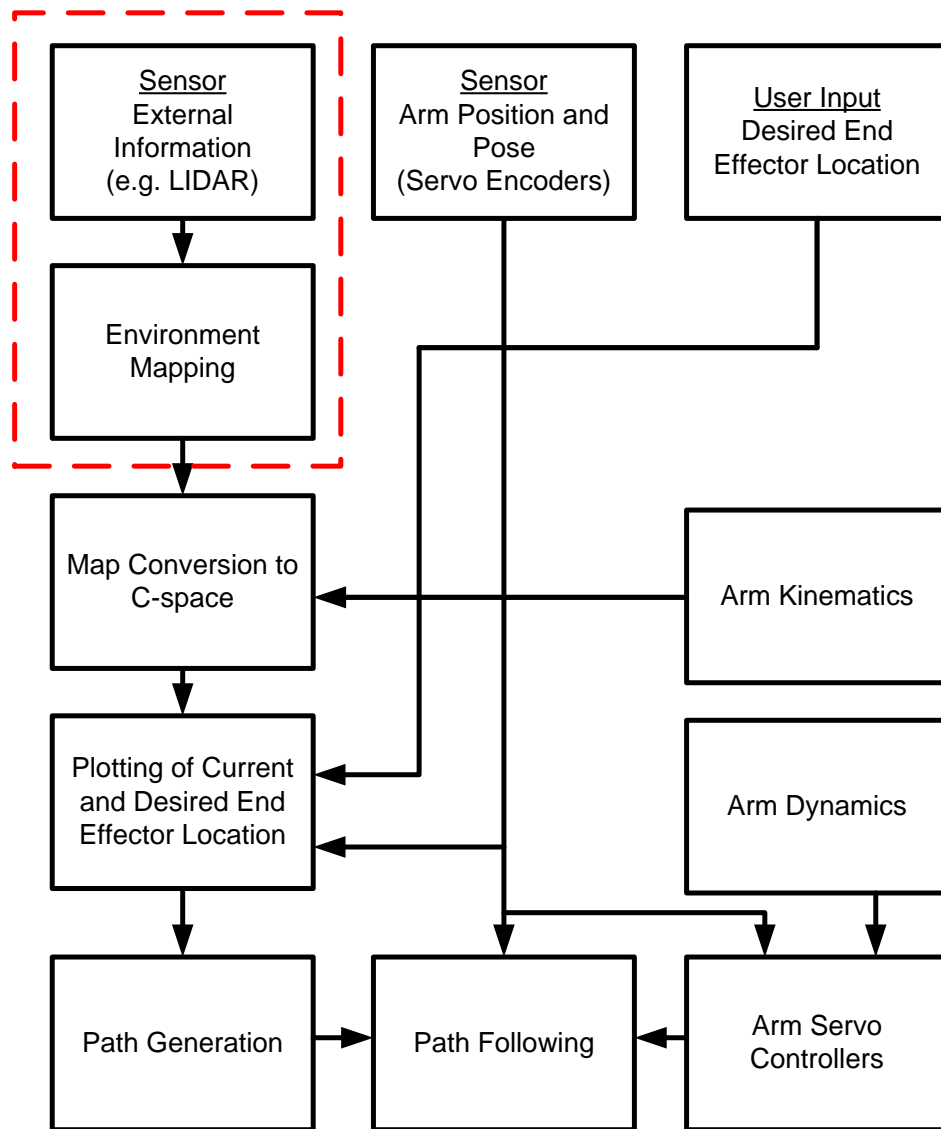


Figure 1-4 Breakdown of the processes required for an autonomous robotic manipulator arm.

1.5 Assumptions and Bounds

In the context of this project, the problem will be simplified and assumptions will be made in order to provide constraints that allow for a solution to be developed within the given timescale. The assumptions that have been made to narrow down the research and development of the proposed method are as follows:

- It will be assumed that the location data and dimensional parameters of objects in the environment have already been obtained by a suite of sensors,

whether these be the visual sensors already employed by the ROV, or other types of sensor such as IR or laser rangefinders, LIDAR, RADAR or SONAR.

- Since the location data and dimensional parameters of objects in the environment are assumed to be obtained by sensors, the environment data can be simulated and input into the guidance method.

Having made these assumptions the previously displayed flowchart in Figure 1-4 has been updated to illustrate how these assumptions affect the implementation of the overall solution in Figure 1-5.

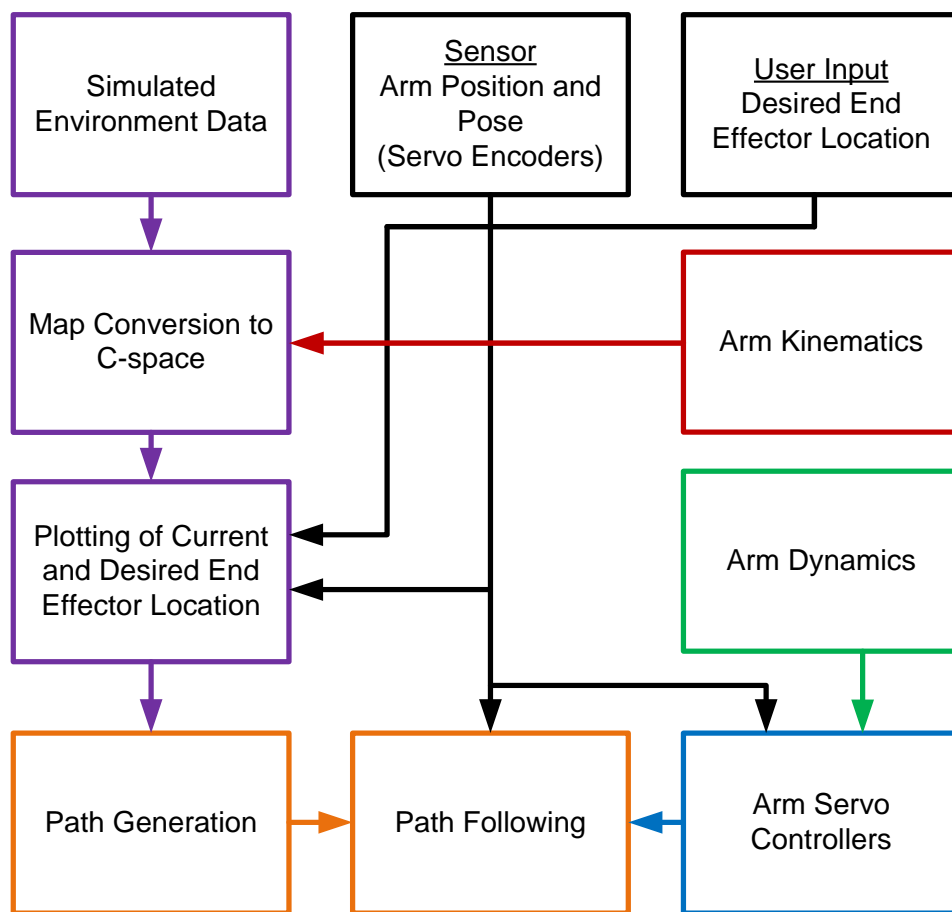


Figure 1-5 Breakdown of the processes required for an autonomous robotic manipulator arm, including assumptions made and colour coded to match the remaining chapters of the thesis.

The steps highlighted by the dashed red box in Figure 1-4 have been replaced with a single box that is responsible for simulated environment data in a useable format rather than data obtained by sensors. The coloured blocks in this figure represent the work carried out in Chapters 3-7. The arm kinematics block, outlined in red, is detailed in Chapter 3; the green block representing the arm dynamics is detailed in Chapter 4; the control of the servo motors which will drive the arm along a path, illustrated by the blue blocks in the figure, is dealt with in Chapter 5; the control domain mapping process outlined in purple in the figure is dealt with in Chapter 6 and the path generation method, orange, is detailed in Chapter 7.

1.6 Contributions

There are several contributions to knowledge made in the work presented in this thesis. The work presented here includes the implementation of a path-generation and following algorithm for a manipulator arm that is designed for use on a skid-steer vehicle with the main purpose being IED disposal. This means that there is no a priori knowledge of the environment at the start of every new mission, and the algorithm has to generate a map in real-time of the environment in C-space and generate a safe path around obstacles in order to reach the target end-effector position.

While techniques exist in the literature on manipulator guidance that seeks to solve the problem of path planning for the entire manipulator or attempt to solve the problem of path planning for an end effector in real-time, no technique exists which is able to successfully combine all of these factors, especially in a completely unknown environment and in such close proximity to obstacles.

The technique presented here uses graph theory in configuration space (C-space), which has never been previously applied in this way. Its implementation to manipulator guidance in C-space highlights its power in such a control domain since it has the capability to handle path planning in an unlimited number dimensions providing that an adjacency matrix can be calculated for n-dimensional points in a space.

The major contribution of the research presented in this thesis is that existing methods in the areas of robotic manipulator guidance and control, environment mapping and path planning have been drawn together and combined in such a way as to develop a guidance technique that is capable of satisfying all of the following attributes.

- The developed technique is capable of path planning in high complexity environments in real-time (i.e. less than 0.1 seconds).
- It has the potential to be applicable to n-DoF manipulator arms for 3-D environments.
- This method is capable of dealing with unknown environments as the manipulator arm is installed on a mobile vehicle therefore the environment is not a permanent reachable space that can be mapped a priori.
- Joints are not decoupled for path generation and so there is only one trajectory, therefore trajectories do not need to be resynchronised.
- This method is able to track around obstacles in the control domain which translates into edge following in Euclidean space.

When considering the problems of task space (T-space) to C-space conversion and path generation for graph theory, different solutions have been investigated in order to find those which are appropriate for real-time applications and so a comparison of techniques has been presented. Thus a trigonometric approach to T-space to C-space conversion and Dijkstra's algorithm as the solution to the pathing in T-space have been presented as the best options in the case of this application.

1.7 Summary

This chapter introduces the problem of path planning for a robotic manipulator in dangerous close-proximity environments and discusses the background surrounding the research question. The aims and objectives and novelty of the work are discussed as well as the constraining factors. It also outlines a possible solution that will be further explored in the following chapters.

Chapter 2 reviews the literature in the areas of modelling, control, mapping and path planning in the context of robotic manipulator arms. This review of literature serves as the basis for the further investigation and design decisions made in Chapters 3-7 of this thesis. Chapter 3 describes the forward and inverse kinematics of a 3-DoF arm to provide a method of localisation of the robotic manipulator in space, and also the Euclidean environment to control domain map conversion. In Chapter 4, the derivation of a dynamic model of the same robotic manipulator is carried out in order to build a predictable test bed for use in the validation of the guidance method. Chapter 5 deals with the selection, implementation and tuning of a control schema which allows the dynamic model to follow a planned path and inform the development of a mapping and path planning algorithm when considering the physical and practical limitations of the system. Chapter 6 develops a method of converting obstacle data from the Euclidean domain to a domain which is representative of the direct control requirements of the manipulator arm and uses that method to form a suitable map of the environment. Chapter 7 concentrates on a method of generating a path through the map and discusses how to include time based information to assist the manipulator arm to follow the path whilst remaining within its physical boundaries. In Chapter 8 the complete technique is integrated with the controlled robotic manipulator dynamic model and simulated environment data is used to validate the effectiveness of the technique. Finally, Chapter 9 draws all of the findings of the work together and provides a series of conclusions about the guidance method proposed in this thesis and suggest future work to enhance and improve the developed technique. The overall scope of the work carried out in this thesis is summarised by the diagram in Figure 1-6.

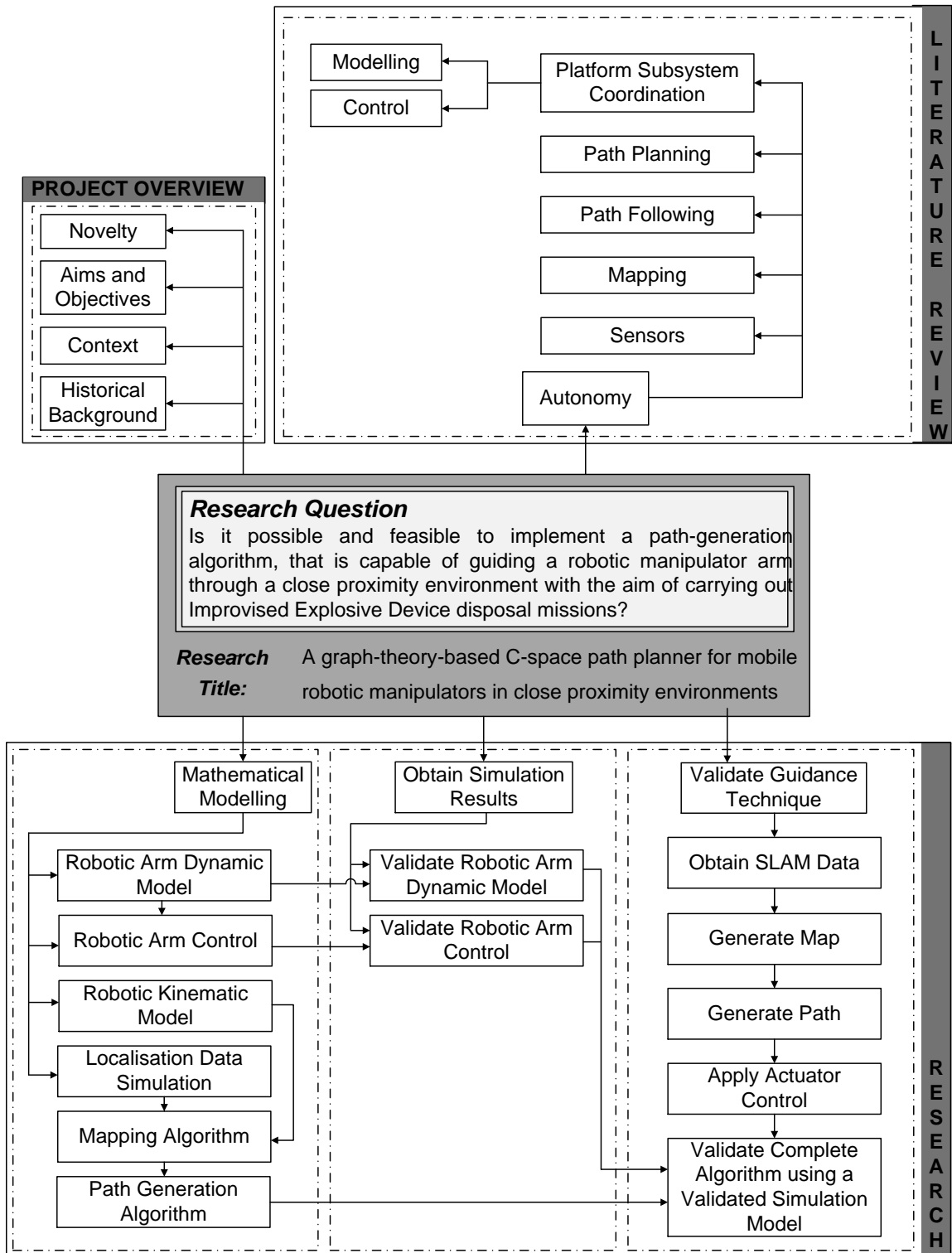


Figure 1-6 Project outline of the research presented in this thesis.

2 REVIEW OF LITERATURE

In the Introduction to this thesis the background and scope of the research to be carried out have been discussed. This has determined the areas of literature to be explored in this chapter which provide the necessary information for the investigation and technique development carried out in the remainder of the thesis to allow the overall aim and objective laid out in the Introduction to be satisfied.

There are four main areas of literature that are reviewed in this chapter. Techniques for modelling of the forward and inverse kinematics of a robotic manipulator are investigated to provide a basis for the work carried out in Chapters 3 and 6. Literature regarding the dynamic modelling of a robotic manipulator arm is also investigated since it will inform the work carried out in Chapter 4, where a dynamic model of a 3-DoF robotic manipulator will be derived for use as the main system for the remainder of the research. In this chapter appropriate methods of feedback compensation are also investigated to provide the necessary information for the selection of a suitable control schema in Chapter 5 to provide adequate control the manipulator arm such that it is able to follow any generated path without any unsafe deviation. In chapters 6 and 7 a suitable guidance method is developed which involves the areas of environment mapping, including sensors, path generation and path following.

2.1 Modelling of Robotic Manipulators

The following section of this chapter presents a review of literature in the area of robotic manipulator kinematic and dynamic modelling to provide a basis for the derivation of forward and inverse kinematics and a predictable dynamic model of a 3-DoF manipulator arm.

Turney et al. (1980) develop two formulations for robot arm dynamics. One of these is based on Lagrangian mechanics, and the other based on Newton-Euler (N-E) mechanics. The authors then show that the two approaches are mathematically

equivalent and the computational complexity of the methods is compared. Finally, a modified formulation of the Newton-Euler method is developed which is then proved to be less computationally complex and that allows more parallelism in its computation than the original two formulations. Table 2-1 and Table 2-2 display the results presented by Turney et al. and assess which method has the smallest computational overheads for a 3-DoF manipulator arm. In this case n is the number of degrees of freedom of the manipulator arm.

Table 2-1 Extract from a computational complexity comparison of Lagrangian and Newton-Euler mechanics from Turney et al. (1980)

Approach	Multiplications	Additions
Lagrange	$\frac{81}{6}n^3 + \frac{165}{2}n^2 + 5n$	$\frac{40}{3}n^3 + 58n^2 - \frac{64}{3}n$
Newton-Euler	$108n - 12$	$100n - 9$

Table 2-2 Comparison of Lagrangian and Newton-Euler methods in relation to a 3-DoF robotic manipulator arm.

Approach	Multiplications	Additions
Lagrange	1122	818
Newton-Euler	312	240

From these results it is clear that the Lagrangian mechanics have a computational complexity which is approximately 3 times as large as the Newton-Euler method for a 3-DoF manipulator arm.

Lee (1982) uses vector geometry and rotational matrices to describe the kinematics of a robotic arm, and uses the Denavit-Hartenberg (D-H) representation to describe the relationship between linkages in the arm. The author then uses the N-E, Lagrange-Euler (L-E) and Generalised d'Alembert Equations of Motion

representations to describe the dynamics of the robotic arm, and concludes that all three are useful depending on the required specification of the user.

Everett and Suryohadiprojo (1988) present work attempting to prove that regardless of the kinematic model that is chosen for a robotic manipulator there is a maximum number of parameters that must be determined. This characteristic implies that model accuracy cannot be improved by adding “extra” parameters. The paper also shows how to model a manipulator so that a minimum Jacobian is used, which reduces the computation required for calibration.

Mooring and Padavala (1989) describe a measurement system which collects data about the pose of a robot manipulator. The data is then used to identify the parameters which can then be used to identify the parameters of a manipulator.

Zhuang et al. (1992) propose a kinematic modelling convention for robotic manipulators. The modelling convention, named the CPC model because of the completeness and parametric continuity properties it displays, uses a singularity-free line representation which consists of four line parameters. The model works by transforming between the world axes and axes of the robotic manipulator base, and also the axes of final link and the axes of the installed tool, allowing for the location of each part to be described both in terms of the world axes or the base axes. All of the redundant parameters can be systematically eliminated from the model allowing for a linearised robot error model to be constructed. In this model all error parameters are independent and span the entire geometric error space, which makes the model useful for robot calibration. The authors focus on model construction, mappings between the D-H model (Denavit & Hartenberg, 1955), (Hartenberg & Denavit, 1964), the study of the model properties and its application to robot kinematic calibration.

Kostic et al. (2004) carry out a case study which explains a procedure for getting models of robot kinematics and dynamics that are appropriate for robot design. The authors concentrate heavily on the design of identification experiments and online reconstruction of state coordinates, which influence the quality of the estimation process. They state that the modelling of friction and the estimation of friction parameters are important and so consider them in detail. The method uses rigid

body dynamics so does not take into consideration compression and bending of joints and linkages.

2.2 Control of Robotic Manipulators

The following section of this chapter presents a review of literature in the area of robotic manipulator control in order to select an appropriate control schema to provide adequate performance for the manipulator arm.

D. E. Whitney (Whitney, 1969) analyses the kinematics of remote manipulators and human prostheses to derive resolved motion rate control. The approach taken suggests solutions to problems of coordination, motion under task constraints, and appreciation of forces encountered by a controlled hand. The author concludes that working with rates means that the problem remains linear, regardless of the arm configurations. The paper shows that the operator can obtain control of motion easily along the “world coordinates” if the control actions are modified by the inverse of the arm’s Jacobian Matrix. This allows for a choice of several different coordinate systems in which to control.

In Saridis and Lee (1979), the authors develop a theoretical procedure for comparing the performance of arbitrarily selected admissible controls with each other and with the optimal solution of a nonlinear optimal control problem. The authors propose a recursive algorithm for sequential improvement of the control law which converges to optimal. The approach is applied to the approximately optimal control of a trainable manipulator with seven degrees-of-freedom, where the controller is used for motion coordination and optimal execution of object-handling tasks.

Luh et al. (1980) state that a manipulator is very difficult to control due to the nonlinearity of the system and the high level of coupling between the joints. They present a technique which adopts the idea of an “inverse problem” and extend the results of resolved-motion-rate controls. The method deals directly with the position and orientation of the end effector. The approach taken by the authors is to specify acceleration and execute the feedback control at the hand level.

Koivo and Guo (1981) present an approach to the position and velocity control of a manipulator by using an adaptive (self-tuning) controller. The system is modelled by a set of time varying differential equations and the parameters of the system are determined by an on-line recursive algorithm based on the least squares error criterion. These differential equations and the chosen parameters are then used as the basis for the design of an adaptive controller. The controller is calculated online using the model with estimated values of the system parameters.

Lee and Chung (1982) also focus on the study of an adaptive control method. The approach taken is based on the perturbation equations in the vicinity of a desired trajectory. The highly coupled nonlinear dynamic equations of a manipulator are expanded in the vicinity of a pre-planned joint trajectory to obtain the perturbation equations. The adaptive control strategy reduces the manipulator control to that of a linear system about a desired trajectory. The authors carry out numerical simulations on a three-jointed robot arm and the results illustrate that the proposed adaptive control algorithm performs better for various loading conditions than a simple PD controller based on a computed torque technique. They conclude that a clear advantage of the proposed formulation is that the nominal torques and the variational torques can be completed separately and simultaneously.

Lea et al. (1993) investigate the feasibility of applying fuzzy logic based control for robotic systems. They develop fuzzy logic based algorithms for semi-automatic control of a robotic arm to eliminate the problem of inversion of Jacobian matrices in conventional control. In the controller the difference between the desired location and current location is fed in as an input vector and joint rate commands are generated.

Jung and Hsia (1995) present a neural network control technique for non-model based PD control of robot manipulators. The proposed technique compensates for the robot dynamic uncertainties outside the control loop by modifying the desired input trajectory. The authors use two neural network training signals to develop two different control algorithms. One of the algorithms is comparable to that of the Feedback Error Learning (FEL) technique proposed by Kawato et al. in (Kawato, et al., 1987) and the other involves the Jacobian of the Proportional Derivative (PD) controlled robot dynamic system.

Xin et al. use the State Dependent Riccati Equation (SDRE) Technique (Cloutier, 1997) to build a robust controller for robotic manipulators in (Xin, et al., 2001). The authors formulate the control problem as a nonlinear optimal regulator problem to treat the high level of nonlinearity in the problem. The SDRE technique is used to synthesise an optimal controller for the robot control problem. A neural network based controller was also synthesised in order to achieve the robustness in the presence of the parameter uncertainties.

Yamada et al. (1998) present a robot manipulator control method using a disturbance observer in task space and a simple coordinate transformation using the transposed Jacobian matrix. The authors state that the control method can realise a simple control algorithm with smaller computational overheads for large degree-of-freedom robot manipulators, however, the inertial variation depending on manipulator configuration is large due to the simplifications of the control algorithm. This means that the decoupled error of the system influences the accuracy of the system. Therefore, the authors introduce a multi-input multi-output (MIMO) disturbance observer to the control of the system so that the non-diagonal elements of the inertia matrix can be regarded as a control parameter. The authors also introduce a design method for this observer which considers the quadratic stability of the system.

Nahapetian et al. (2008) present a tuning method for a PID controller that uses a Genetic Algorithm (GA) as a main gain estimator and Fuzzy Logic as a ranking basement for the GA.

Tang et al. (2010) propose a self-adaptive Proportional Integral Derivative (PID) controller based on a Radial Basis Function (RBF) neural network online identification for a robot manipulator. This approach addresses the strong nonlinearity and parameter uncertainty in manipulator control and solves the weak adaptive ability and poor robustness of the conventional PID control. The approach uses a self-adaptive single neuron network to tune the parameters of the PID controller, while and RBF neural network identifies the manipulator online and simultaneously obtains the Jacobian transformation for the controller.

2.3 Guidance

Two processes are required to provide guidance for the robotic arm. The first is to provide information about the environment in the form of a map. The second is to generate the path through the map from the starting location to the desired location.

2.3.1 Mapping

Before being able to plan a path for the arm, data about obstacles in the operating space must first be obtained. In reality this data must be gathered using some form of sensor. Several types of sensors are available for use in constructing a map of the environment. A simple method of constructing a map is by using range data from objects surrounding the vehicle. Range data can be obtained by measuring the time between the transmission of a pulse of a waveform, either of light or sound, and its subsequent arrival after reflection off a surface as seen in Figure 2-1. This is known as time-of-flight (ToF). Knowing the time taken between the transmission and receipt of the wave, and the speed of the wave in question, the distance to the object and back can be calculated, and the distance to the object is half this.

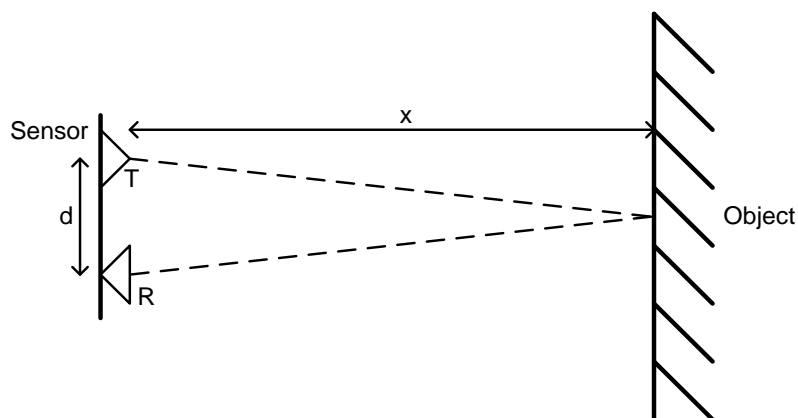


Figure 2-1 Wave-based sensor operation using ToF to calculate the distance to an object.

In this diagram T is the transmitter, R is the receiver, x is the distance to the object, and d is the distance between T and R. This distance is often so small that it can be considered negligible and treated as zero, simplifying the calculation.

Infrared (IR) sensors can be used for distance measurements as shown in (Benet, et al., 2002). Infrared sensors are low cost and have a very quick response time, on the order of 10 to 100 wave emission and detections per second (Akai, et al., 2006). The disadvantage of using infrared sensors is that they exhibit highly non-linear behaviour and depend heavily on the reflectance of the surface in question. This causes environment maps made with measurements based on the intensity of back-scattered IR light to be of poor quality. It is for this reason the IR sensors are almost exclusively used as proximity detectors in mobile robots. There are ways of estimating the properties of a surface based on its reflectance and subsequently the distance from sensor to surface and its angle of orientation, such as the use of the Phong Illumination Model (Novotny & Ferrier, 1999), which tends to increase the accuracy of the range measurement.

Benet et al. (2002) also explain how ultrasonic (US) sensors can also be used for distance measurement, with a precision of less than one centimetre over a distance of six metres. These sensors are also relatively inexpensive; however they also pose a very crucial disadvantage when being used for real time data gathering. Most US sensor range finding is based on ToF measurement. With an object 6 m away there is a total flight distance of 12 m. With a pulse speed of approximately 340 m s^{-1} the ToF is approximately 0.04 s. When compared to light based waveforms with a pulse speed of approximately $3 \times 10^8 \text{ m s}^{-1}$ and ToF is 4×10^{-8} seconds, US has a very long response time, making range finding very slow in comparison to light based ranging sensors. Other examples of US ranging and mapping include (Audenaert, et al., 1992), (Rencken, 1993) and (Mohammad, 2009).

Another sensor which provides range information is a laser rangefinder. Laser rangefinder sensors are highly capable of obtaining data in real time with a high degree of accuracy. However, the sensors are very expensive, so using an array of laser range-finding sensors to build up a map of the environment is impractical. An alternative is to use a single rapidly scanning laser rangefinder to build a picture of the environment. Buchberger et al. (Buchberger, et al., 1993) use a combination of laser-radar and a sonar sensor so that a world model for obstacle avoidance could be built.

A similar approach by Zhang et al. (Zhang, et al., 2005) uses a Light Detection and Ranging (LIDAR) sensor in conjunction with a digital map and video image sequence to build a 3-D model of a city landscape. Other, much older approaches have been taken using sonar sensors both for mapping and localisation, and examples include (Moravec & Elfes, 1985). Likewise, improvements in radar over the decades have allowed it to be used not only for mapping and object detection, but to the point where it can be used to track multiple moving objects (Mobus & Kolbe, 2004), (Drumbeller, 2009) and (Elfes & Matthies, 2007)

LIDAR can also be utilised in a slightly different way to gather information about the environment for the purposes of mapping. A technique called flash LIDAR imagery can be implemented which provides highly detailed data for the construction of an accurate map of the environment. A pulse from a wide beam laser can be sent out, and the resultant reflection recorded using a high resolution camera. Each pixel can be analysed individually to obtain useful information. Knowing the frequencies in use by the LIDAR, ToF can be used to calculate the range to the object that can be found in each pixel. Given that an emitted sensor pulse travels at the speed of light, this technique can be considered to be almost instantaneous, therefore real-time, working within μs for distances of kilometres. Another form of data that can be collected from this technique is the intensity of the light that is reflected of the frequencies in use. A large spike indicates that the majority of the light in that pixel was reflected; therefore the object it reflected off was flat. If the reflected light is of a lower intensity then less of the light has been reflected, and information about the shape of the surface in that pixel can be inferred. Examples of flash LIDAR in use are given in (Gelbart, et al., 2003) and (Hanna, et al., 2005).

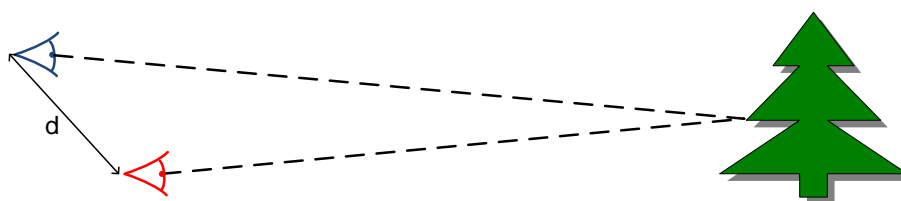


Figure 2-2 Monocular ranging using known vehicle motion to cause disparity between successive image frames.

Another method of gathering range data and building environment maps is to use visual information from camera images. This can be done using a single camera in one of several ways. The first is to take multiple images of the same object as the vehicle moves along its path, knowing the motion of the vehicle hence the distance between each image position, and measuring the disparity between images of the same object to calculate the distance to the object as shown in Figure 2-2. The biggest drawback to this approach is that the distance to the object can only be calculated from successive images so is not instantaneous. In a similar way a complete visual map of an object can be made by taking several images at known location and orientations about the object. This technique is carried out Niem and Wingbermuehle (Niem & Wingbermuehle, 1997).

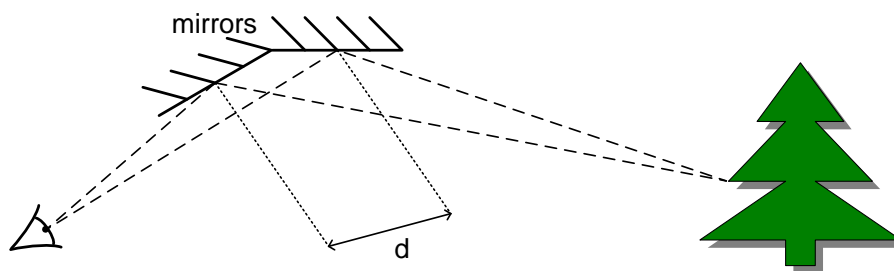


Figure 2-3 Monocular ranging using two mirrors at a known distance from each other to provide two images, the disparity of which can be compared to calculate object distance.

Another method used is to reflect the image of the object from two slightly different positions into one camera, knowing the difference in the position of each reflector and measuring the disparity between the reflections to calculate object range as shown in Figure 2-3. This principle effectively simulates stereoscopic vision and provides the range information in real time rather than over successive frames. This is the case in (Shimizu & Okutomi, 2007) where Shimizu and Okutomi use a double-sided half-mirror plate to provide two reflected images, as shown in Figure 2-4. This approach is novel but not practical in a case where multiple cameras are already installed on the vehicle as there is no need to add extra hardware to the vehicle for stereoscopic ranging.

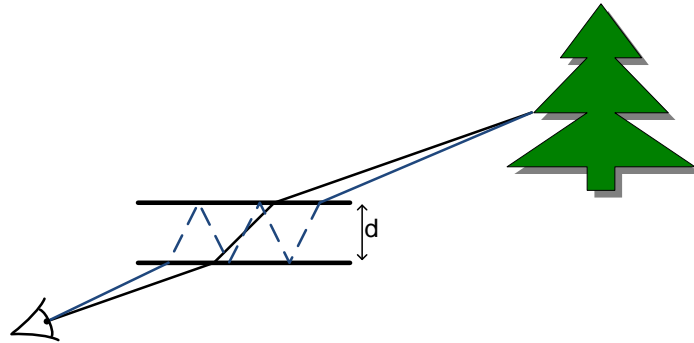


Figure 2-4 Monocular ranging using a pair of double-sided half-mirror plates a known distance apart can be used to provide two images of the object, the disparity of which can be compared to calculate distance.

A third approach to monocular ranging is to identify the object in the image from an image memory and with this information gauge the approximate size of the object. This parameter, along with the size of the object in pixels on the image can be used to calculate the range of the object. This approach is taken by Menegatti et al. in (Menegatti, et al., 2004). The greatest limitation with this approach is the high memory requirement needed to store enough images to provide an adequate library for object identification.

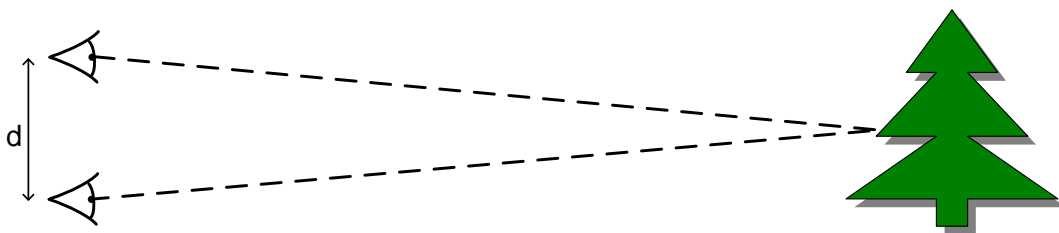


Figure 2-5 Binocular ranging using two cameras a known distance apart, and measuring the disparity between the image taken by each to calculate the object's distance.

A simpler approach to visual ranging and mapping is to use multiple cameras to provide stereoscopic images, as illustrated in

Figure 2-5. This approach does not require multiple image frames to provide a disparity between images, which reduces the calculation time for range information. This approach is taken by Murray and Little (2000) to provide real-time stereo vision information. This information can be used for ranging, mapping, velocity estimation (Ab-Rahman, et al., 2005) and 3-D object recognition (Stasse, et al., 2006). Object recognition for localisation and map building can be enhanced by using edge recognition to more easily identify object shapes. This type of approach is taken in Rosenfield & Thurston (2006) and Tomono (2010).

Some of these sensor methods are impractical for this application since they would be too large for the arm or base of the manipulator, but there are sensors of small enough size and high enough accuracy for the majority of the above techniques. In order to provide an accurate map of the environment in real-time with a large number of data points, it will be assumed that a LIDAR sensor will be used, and data obtained from objects will follow this assumption.

2.3.2 Path Generation and Following

Prior to the review of literature in the area of robotic arm guidance it is important to consider some terms that are referred to in great frequency in the literature. The physical world, which is the simplest to visualise occurs in the Euclidean domain. This domain is also where tasks are carried out by manipulator arms and so is often referred to in the literature as task space or T-space. However it can be useful to visualise the world in terms of the manipulator joint combinations that would cause a collision between the object and the manipulator arm, as in this visualisation, the dimensions of space are the manipulator joint ranges, making the path that is planned a single path through this space. The number of dimensions that are present in this space is dependent on the number of degrees-of-freedom of the manipulator arm. For example a 3-DoF arm would create a 3-D space, and an n-DoF arm would create an n-D space. This space is often referred to as joint space or configuration space (C-Space).

Path generation for manipulator arms is often carried out in the Euclidean operating space with the path then being transformed into a trajectory in joint space to more

easily allow for the control of the manipulator (Gasparetto & Zanotto, 2007). The trajectory generation can be done to satisfy different requirements, such as a time-optimal solution (Gasparetto & Zanotto, 2007), (Ramos, et al., 2013), (Haschke, et al., 2008); optimal-pose (Zha, 2002), which is especially useful in surgical robotics, minimum energy or optimal power (Gasparetto & Zanotto, 2007), or minimal jerk (Gasparetto & Zanotto, 2007), (Gasparetto & Zanotto, 2008), (Haschke, et al., 2008). The solution to the problem of trajectory generation in robotic manipulators is sufficiently complex that the manipulator dynamics are frequently removed to decouple the manipulator joints from one another and calculate for each one individually (Gasparetto & Zanotto, 2007). This paper presents a solution to the path and trajectory generation problem which removes the need to decouple joints to reduce complexity. The process of converting obstacles into the joint space (which is no more complex than converting a pre-generated path into joint space) allows for the manipulator arm to be considered a point mass, and a single path can be generated to guide it through the joint space.

Leven and Hutchinson (2002) use random sampling in C-Space to create a probabilistic roadmap of the obstacle-free space. This method is capable of planning a path for a serial-link manipulator with 20 joints in 2 or 3 dimensions, but the method is not able to cope with fine motion planning or narrow passages between obstacles, which is an inherent drawback with random sampling approaches to this type of problem. This makes this technique unsuitable for the application of path planning and obstacle avoidance in close-proximity environments where the likelihood of narrow passages is very high.

Lu and Chung (2005) present a method of path planning based on collision detection which is safe for operation in close-proximity to humans. The method involves driving the arm to its desired location, while ensuring that if a collision occurs it is detected and an emergency stop command is enabled which controls the force level to ensure it is below the human pain threshold to prevent injury. This method has many benefits, especially since it allows robotic manipulators to be operated safely in human presence, but is not useable in the application of IED disposal since the lightest touch could trigger an explosive detonator.

Lin et al (2005) carry out potential field path planning in C-Space rather than the Euclidean space. In order to solve the issue of local minima, the technique employs a method of adjusting the potential gradient locally. This method is tested only in 2-D, but can be used for 9-DoF, since the potential field will have an effect on all joints and links in the arm. The paper concludes that the method can be extended to 3-D, however, since this method acts on each specified point on the arm, it still effectively has to generate a path for each link in the manipulator separately. Wei and Shimin (2010) also use potential fields combined with neural networks to allow for path planning in dynamic environments. The technique can be used to generate a path which guarantees obstacle avoidance at a safe distance while planning the shortest possible route through the environment while also having the benefit of low computational load. This work utilises neural networks to provide optimisation of the path. This method also suffers from the effects of local minima. Padula and Perdereau (2011) use potential fields to control joint velocities in Euclidean space. The benefit of controlling joint velocity is that it does not consider dynamics, and therefore removes the requirement to deal with joint torques in control. This method is effectively generating a path for each joint in the arm, and has the drawback of suffering from the inherent problem of potential fields in that it may converge on local minima in cluttered environments, making it unsuitable for the application of close-proximity environments. Nakamura (2013) uses a 2-D dipole field to generate a path for the end effector of a robotic manipulator. This paper states that the conventional method used to prevent local minima settling often causes oscillations and that the author's proposed method seeks to prevent that occurring. The method is able to provide a smooth path for the end effector in 3-D space, but the technique is carried out offline and only deals with the end effector.

Ryu et al (2007) use inverse kinematic analysis of humanoid robots and a rapidly exploring random tree (RRT) to generate a path in C-Space for the robot. This method reduces the need to find several thousand unnecessary configurations, which reduces computational complexity, but requires an initial and goal configuration, which could be unsolvable if one or both of these configurations are unsolvable. D'Silva and Miikkulainen (2009) have used neural networks RRT methods to avoid obstacles in an environment with a 6-DoF manipulator arm. The method of obstacle avoidance can occur in real time, but the learning by the neuro-

controllers takes place in an environment where the number of objects does not change, but may move around, and the nodes in the network cannot relearn once they have carried out their learning process. This makes the technique unsuitable for use in a function where the environment will be different with every use of the arm.

Ding et al (2009) use mixed-integer linear programming to plan an optimal path through an environment with dynamic obstacles. The technique uses rectangular approximation of object to simplify the computational requirements, but it requires trajectories to be known a priori. The simplifying of obstacles to rectangular approximations means that any concavity in the objects will be removed, and so potentially navigable valleys where objects form a narrow channel will not be considered by the path planning algorithm. This drawback removes the ability of the algorithm to handle the close-proximity environments which are commonplace in IED disposal applications.

Korayem et al (2009) attempt to overcome the non-linearity in flexible robot arm dynamic equations using finite element analysis to plan a path in task space for the end effector of a 2-DoF manipulator. The advantage of this method is that it removes the need to linearize the system in order to implement optimal control methods. Since this method only solves the path planning problem for the end effector of a 2-DoF arm, it is not applicable to the problem of manipulator planning in 3-DoF for obstacle avoidance of the entire arm.

Chetty and Pomambalam (2012) employ a heuristic approach towards the planning of a path for a manipulator which operates in a 2-D plane. The method uses an iterative approach of particle swarm optimisation to search for random joint combinations that will enable the arm to reach the desired end effector location without colliding with obstacles at any point on the arm throughout the process. The manipulator that is used is a planar 5-DoF manipulator, though this technique could be applied to 3-D. The method is only applicable, however, in a static environment with fixed obstacles since any motion would render potential solutions already found ineffective or unachievable.

Chen et al (2012) use fuzzy logic to plan a path for a 2-DoF fixed pedestal manipulator arm. Their approach is advantageous since it does not require inverse

kinematics, but only deals with obstacle avoidance for the end effector and only for a 2-DoF manipulator arm.

Singh and Leu (1987) carry out an offline trajectory planning algorithm for a manipulator arm which it can then follow. They attempt to reduce the problem to a trajectory search for a single link, but the inability to work in real time prevents this algorithm from being useful for the application investigated in this project. Kubota et al. (1997) attempt to solve this problem using a virus evolutionary algorithm. They carry this process out by generating intermediate locations for the arm that are free from collision and then plot trajectories between these different positions to achieve their goal. Gasparetto and Zanotto (2007) use B-spheres to generate a smooth trajectory rather than the jerk-bounded approach that has previously been used. Macfalane and Croft (2003) use an online method to obtain jerk-bounded trajectories using a concatenation of fifth order polynomials, and achieves the goal but requires up to a maximum of eight points per trajectory waypoint. Dos Santos et al. (2008) use joint velocity control to plan a path through a constrained workspace. This technique is highly applicable in industrial environments where time-optimal solutions are highly sought after. The method analyses the distance between points on the entire manipulator and any obstacle to determine a safe path. This technique is run offline however, and since the path is pre-generated, it cannot handle dynamic environments or moving obstacles.

Koren and Borenstein discuss the use of the potential field method and its strengths and limitations (Koren & Borenstein, 2002). The potential field method gained popularity in the early 1990s for obstacle avoidance, especially when applied robot manipulators. The potential field method is attractive because of its elegance and simplicity, but has several substantial shortcomings such as trap situations due to local minima, the inability to pass between closely spaced obstacles and oscillation in the presence of obstacles and in narrow passages. It is these drawbacks that make this method unsuitable for the guidance of the manipulator arm due to the tight constraints that are imposed as a result of the nature of the function to be carried out.

Hota and Ghose (2010) take a novel approach to path planning. Instead of planning an optimal solution for a constant speed and turn rate constrained

Uninhabited/Unmanned Aerial Vehicle (UAV), they propose using a sub-optimal solution. They compare their sub-optimal solution with an optimal one and conclude that the optimal solution is very computation intensive and the sub-optimal solution generates a solution close to the optimal one in a shorter time, with a lower computational load, making it more efficient and more easily applicable to a UAV with limited computational power.

Work has been carried out by the former Autonomous Systems Group at Cranfield University involving guidance of UAVs. From 2007 onwards, Shanmugavel et al. carried out work looking at two different geometric path planning algorithms for the guidance for simultaneous arrival at a target of multiple UAVs.

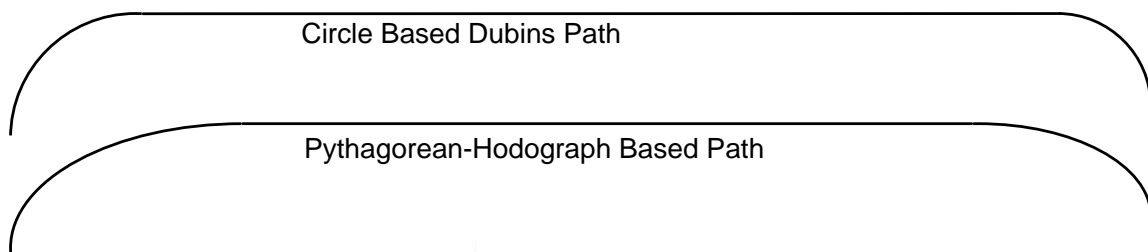


Figure 2-6 Comparison of a Dubins based path with a P-H based path that start and end at the same position and pose.

In Shanmugavel, et al. (2007) the authors propose a technique for path following based on Dubins Paths (Dubins, 1957). The technique aims to ensure simultaneous arrival of all vehicles at the target by enforcing a constraint whereby the path length of each UAV is the same. The authors state that the transition between arc and line segments, which is required in Dubins path following, entails discontinuous changes in lateral acceleration (latax) for fixed-wing UAVs. Because of this the authors make the choice to replace the Dubins-based solution with one based on quintic Pythagorean Hodograph (PH) curves as shown in Figure 2-6, with which the latax demand is continuous. They design the PH paths to be similar in length to the paths generated in the Dubins-based solution to maintain a near minimum time solution. The solution meets the required specification of minimum curvature and produces multiple paths of equal length by plotting paths for each UAV and then lengthening

those shorter than the others. The solution also satisfies the safety constraints and avoids inter-collision by maintaining a minimum distance between UAVs and maintaining a minimum distance between UAVs and maintaining non-intersection at equal path distance.

In Shanmugavel, et al. (2009) the authors describe the cooperative path planning of a group of UAVs, again with the constraint of simultaneous arrival of the UAVs at the target. In this case the authors elect to use Dubins paths with Clothoid shaped arc segments, where the curvature decreases gradually, rather than the instantaneous change between zero curvature of the line segment and a much larger curvature of a curved path segment. The paths are produced using the principles of differential geometry used by White et al. in (White, et al., 2007).

Further work by the Autonomous Systems Group is carried out by Kim et al. (2010). The authors propose the use of a decision making algorithm that mainly relies on waypoint generation and path planning based on Dubins' Theory to guarantee communication between a ground control station and a swarm of UAVs. The algorithm looks at various constraints such as maximum speed, minimum curvature radius and no fly zones that arise in the mission operative scenario.

These methods are not directly applicable to manipulator arms in T-space since there are multiple connected systems to plan paths for simultaneously, but if the path were to be generated in the control domain, C-space, then the robotic manipulator would become, in effect, a point mass. This would then allow the above techniques to be applied to the problem of path planning for robotic manipulators.

Yao and Gupta (2007) plot a kinematic roadmap in C-Space with end-effector constraints before using an RRT to plan a path. The technique uses task space for end effector planning to narrow down the C-Space search. This method does not guarantee a solution since selecting end effector start and end locations does not ensure that an achievable path is possible, and this will only be discovered when the RRT search is exhausted. Conversely, there may be a large number of possible solutions that require searching in order to find the best route, be it time or energy-optimum. Kunz et al (2010) implement real-time path planning in a changing environment by building a roadmap of the environment and then generating a path using a RRT method. This technique handles dynamic obstacles by blocking parts

of the roadmap so that the RRT algorithm cannot use them when an obstacle is in that region of the roadmap. This technique is applicable to high-DoF and can generate a path on the roadmap online and in real-time, but it is implemented in a quasi-dynamic environment where the roadmap does not change, and so can be calculated off-line prior to the operation of the arm. This saves time when online, but makes it difficult to apply when the roadmap would need to be generated prior to every new mission.

Lahaouar et al (2005) employ a grid based method for path planning of manipulators in C-Space, whereby the grid is developed for a path directly from the start configuration to the end configuration, but does not consider any obstacle unless it crosses the path, whereby a grid search takes place to find a safe path around the obstacle. This significantly reduces the computational cost of the path generation and this method has merit for the application of IED disposal since it involves edge following. This method, however, is dependent on the resolution of the grid since a grid square is classed as occupied if it is 1% full or 100% full. This method also has to search in different directions to find the shortest path around an obstacle. This doubled up search is time-consuming. Klanke et al (2006) use a genetic algorithm to plot a path through a grid in C-Space. Their algorithm runs in real-time, and the local node complexity does not depend on the dimensionality of the space. This is done by decoupling the degrees of freedom to generate a path for each decoupled set.

For the grid-based methods of path planning, graph theory is a well-established technique (Euler, 1766), (Biggs, et al., 1976). Graph theory is, in essence, the study of graphs, and these can be used to examine, investigate and model relationships between data or objects (Walther, 2012). They are used throughout science, computing, finance, mathematics, and many other disciplines. Prime examples of this are the search and cataloguing algorithms used by large databases owned by companies like Google (Langville & Meyer, 2006) or Facebook (CBS & McCarthy, 2010). Graph theory is used to create connections between pieces of data. Graph theory is also used to carry out energy optimisation (Kladis, et al., 2008). The use that is most similar to that implemented here is for satellite navigation units, including those used by the consumer market. Graph theory is used to plot a path between destinations via road junctions, and can be carried out using the shortest distance, quickest time, or best fuel consumption.

In this case a graph will be used to model the relationship between points in space. The graph contains “vertices” which represent the points in space and “links” which represent the paths between nodes, and connect them to each other. The length of these vertices could represent a whole plethora of information, but for simplicity this will be the distance between the nodes. This now forms a graph similar to that shown in Figure 2-7.

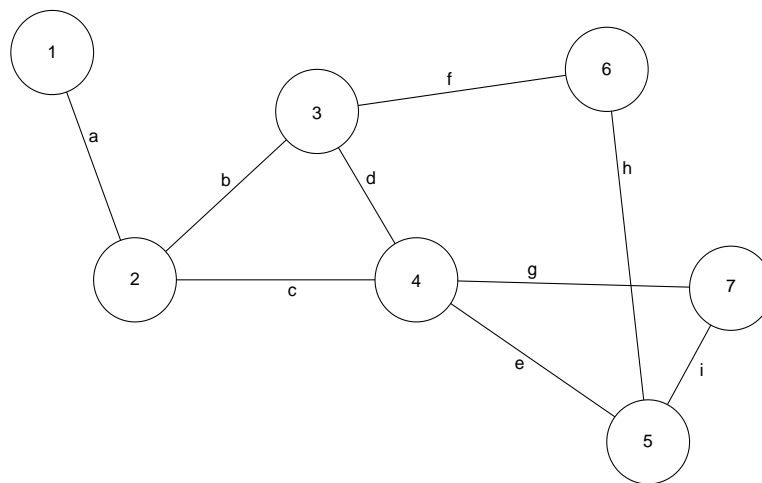


Figure 2-7 Example of a node graph (not to scale). Each node is numbered and the vertices between them are labelled with letters.

Each of the nodes are numbered and each vertex has length, in this case these lengths are defined by lower case a-i. This graph can also be represented by an $n \times n$ matrix, where n is the number of nodes in the graph. In the example given previously, this is a 7×7 matrix (A). Each element in the array represents the relationship or vertex between the nodes that are equivalent to the horizontal and vertical indices of the element.

For example the link between node 4 and node 7 is represented by $A_{4,7}$ and $A_{7,4}$. The value of the distance between the nodes is the value recorded by the corresponding elements in the array. For nodes that are connected this is the length of the vertex. For values that are not connected, this length is infinite. For the elements that represent the distance between a node and itself, the value is zero. For the above example, this gives matrix A as seen in Equation (2.1):

$$A = \begin{bmatrix} 0 & a & \infty & \infty & \infty & \infty & \infty \\ a & 0 & b & c & \infty & \infty & \infty \\ \infty & b & 0 & d & \infty & f & \infty \\ \infty & c & d & 0 & e & \infty & g \\ \infty & \infty & \infty & e & 0 & h & i \\ \infty & \infty & f & \infty & h & 0 & \infty \\ \infty & \infty & \infty & g & i & \infty & 0 \end{bmatrix} \quad (2.1)$$

This matrix is known as an adjacency matrix. It is worth noting that the lengths of the vertices could mean something completely different. For example, if instead of requiring the distance between the nodes, the energy requirement to travel between them were needed, then the data presented as the vertex “length” would be the energy cost. It is also interesting to note that if all of the links between nodes on the graph can be traversed in both directions, then the adjacency matrix will be a diagonal matrix as is the one above.

An advantage to graph theory is that any node, regardless of where it is situated, is given an identifier which corresponds to a row and a column in the adjacency matrix, which means that regardless of the number of variables needed to define a node (e.g. x, y, z which corresponds to three-dimensional coordinates), the node can always be represented in relation to all of the others by including it in a two-dimensional adjacency matrix. This is also true when more than three dimensions are required. This will be shown later in this chapter.

In the case of the technique being developed to provide autonomous path generation and guidance for a robotic arm, several dimensions are needed, ranging from 5 to 9 or 10 dimensions. This presents a problem where the solution is one of high dimensionality, and there are several methods that can be used for path planning in multiple dimensions. The method chosen here has the ability to reduce the dimensionality of the path planning problem from n to 2.

The map of the environment in Euclidean space is obtained from an array of sensors and converted into the control domain. In the case of robot arm control this could constitute the control requirements of each of the servos that move the arm in one degree-of-freedom. In the case of a high degree-of-freedom arm the number of

dimensions could also be high. The nodes on this map, which follow the boundaries of objects can be analysed to find the adjacency matrix, forming a node graph which can be represented in 2 dimensions.

Here a path planning algorithm that employs graphs must be utilised to find the node sequence with the smallest cost. This will be the most optimum path in that variable (distance, time, energy, etc).

There are a number of algorithms that could be used to generate a path through the graph. These include:

- Ford-Fulkerson Algorithm (Ford & Fulkerson, 1956)
- Kruskal's Algorithm (Kruskal, 1956)
- Nearest Neighbour Algorithm (Gutin, et al., 2002)
- Prim's Algorithm (Prim, 1957)
- Depth-first Algorithm (Anon., 2001)
- Breadth-first Algorithm (Anon., 2001)
- Bellman-Ford Algorithm (Bellman, 1958)
- Dijkstra's Algorithm (Dijkstra, 1959) (Anon., 2001)

These algorithms are all designed to find paths through a graph, but for different functions. The Ford-Fulkerson Algorithm is designed to find maximum flow in a flow network. Kruskal's Algorithm finds a minimal spanning tree which connects all of the nodes together by the shortest paths possible. Prim's Algorithm is designed to solve the same problem as Kruskal's Algorithm. The Nearest Neighbour Algorithm performs a similar function in that it attempts to solve the travelling salesman problem and visits each node at least once, by visiting the nearest node to the current one.

The Breadth-first and Depth-first Algorithms are designed to search through a tree to find a specified node. The Depth-first search completely explores one branch of the tree to its tip before exploring the sub-branches of that branch back towards the root of the tree. Once all of the sub-branches on that branch are explored it moves on to the next branch of the tree. The Breadth-first Algorithm searches across the entirety of the second level of the tree before moving on the third level. This achieves a search which covers all branches simultaneously. These searches would find a solution to the problem of finding a path from the start node to the destination node,

but except by chance would not necessarily be an optimal path or be found in the shortest time.

Dijkstra's Algorithm is designed to solve a single source, shortest path problem for a node graph. This produces a shortest path tree, and the algorithm can be stopped when the tree reaches the desired node. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined. The algorithm works by assigning a value to the distance from the start node to every node that it can connect to in the graph, and attempts to improve upon the distances at every step. The Bellman-Ford Algorithm is very similar to Dijkstra's Algorithm in that it is also designed to compute the shortest path from one node to all the others, and can be stopped when it reaches the desired node. The Bellman-Ford Algorithm has the added advantage that it is capable of handling negative link costs as well as non-negative ones. This advantage is not necessary in the required application as all distances must be positive, and the Bellman-Ford Algorithm is also theoretically slower than Dijkstra's Algorithm. The performance of the two algorithms are shown in Table 2-3 and Table 2-4.

Table 2-3 Comparison of algorithm running times.

Algorithm	Worst case running time
Dijkstra	$O(E + V \log V)$
Bellman-Ford	$O(V E)$

Where E is the number of edges (links) and V the number of vertices (nodes). For the above example graph, which has 7 nodes and 9 links, the following is the case. $E = 9$ and $V = 7$:

As can be seen from Table 2-4, Dijkstra's Algorithm is significantly faster than the Bellman-Ford Algorithm, and based on the formula for calculation time, this difference in calculation time will only increase for larger, more complex graphs.

Table 2-4 Comparison of algorithm running times with values.

Algorithm	Worst case running time	
Dijkstra	$O(9 + 7\log 7)$	$= O(22.6214)$
Bellman-Ford	$O(7 \times 9)$	$= O(63)$

The following section gives an explanation of the method by which Dijkstra's Algorithm works. It is implemented by the carrying out the following iterative process. The starting node will be termed the initial node (I), and the destination node will be denoted by J. Dijkstra's Algorithm assigns distance values to the path and attempts to improve them step by step:

1. Assign an initial distance value to each node, zero for the initial node and infinite for all of the others.
2. All nodes are marked as unvisited, and the initial node is set as the current node. All unvisited nodes are included in a set (unvisited set), which at this point consists of all the nodes except for the current node.
3. Consider all of the nodes connected to the current node by a link, calculate their tentative distances from the current node based on the cost of the link between these nodes and the current node. For example, if current node A is marked with a distance of 5, and the link connecting it with neighbour B has length 4, then the distance to B via A will be the sum of tentative distance at node A and the link length from A to B. If this distance is less than any previously recorded tentative distance at B (for example via another route), then overwrite that distance with the new tentative distance. Even when the neighbour B has been examined, it is not marked as visited and remains in the unvisited set.
4. When all of the neighbours of the current node have been examined, mark the current node as visited and remove it from the unvisited set. A visited node can never be checked again once it has been marked as visited.

5. If the destination node has been marked as visited or the smallest tentative distance among the nodes is set as infinite, then stop as the algorithm has finished.
6. Select the unvisited node with the smallest tentative distance out of all nodes in the unvisited set and set it as the current node.
7. Repeat algorithm from step 3.

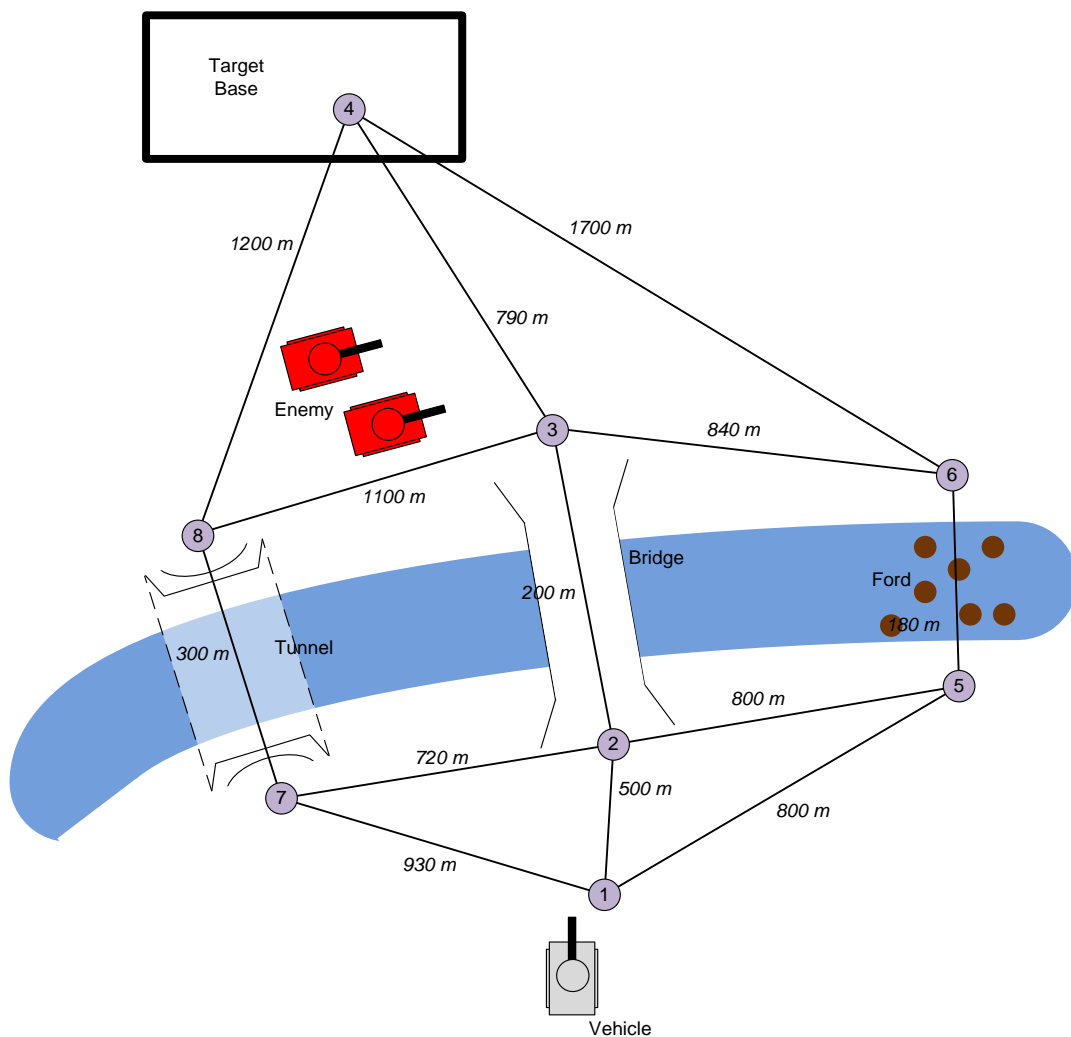


Figure 2-8 Example of an environment where graph theory can be used to generate a trajectory.

To illustrate the Dijkstra's Algorithm as simple example of a tank traversing an environment towards a target over a river with four obstacles, a ford in the river, a bridge over it, a tunnel under it and two enemy tanks. The purple circles represent

nodes in the graph and the vertices are labelled with their costs (in this case the distances between them). The following diagram shows the scenario and with a graph overlaid on top. The graph contains 8 nodes and 13 links and the cost of each link is displayed in the diagram. The vehicle at node one is attempting to reach node 4. This graph show in Figure 2-8 produces the adjacency matrix found in Equation (2.2):

$$A = \begin{bmatrix} 0 & 500 & \infty & \infty & 800 & \infty & 930 & \infty \\ 500 & 0 & 200 & \infty & 800 & \infty & 720 & \infty \\ \infty & 200 & 0 & 790 & \infty & 840 & \infty & 1100 \\ \infty & \infty & 790 & 0 & \infty & 1700 & \infty & 1200 \\ 800 & 800 & \infty & \infty & 0 & 180 & \infty & \infty \\ \infty & \infty & 840 & 1700 & 180 & 0 & \infty & \infty \\ 930 & 720 & \infty & \infty & \infty & \infty & 0 & 300 \\ \infty & \infty & 1100 & 1200 & \infty & \infty & 300 & 0 \end{bmatrix} \quad (2.2)$$

By inspection of the map (which is not to scale), the path that appears to be the shortest would be 1 → 2 → 3 → 4, with a distance of 1490 m. By passing the matrix through Dijkstra's Algorithm, the result is identical.

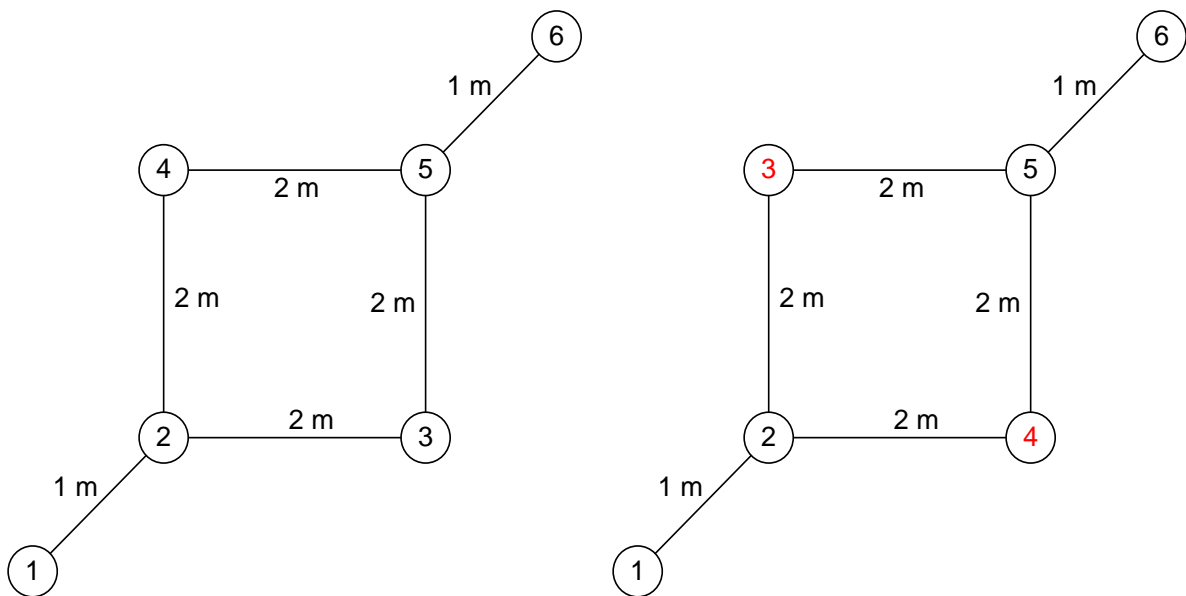


Figure 2-9 A simple node graph.

It is worth noting what occurs when two paths are the same length. In the example shown in Figure 2-9, the two possible routes are $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 6$ or $1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$. However, Dijkstra's Algorithm will always travel by the shortest path with the lowest node identifiers as the algorithm searches sequentially through the nodes which are the nearest neighbours of the current node. To illustrate this nodes 3 and 4 have been switched in the second diagram, indicated by the red node identifiers, however, as the adjacency matrix is identical, the algorithm will still always pick the first of the two routes ($1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 6$). Equation (2.3) shows the adjacency matrix representing the node graphs from Figure 2-9.

$$A = \begin{bmatrix} 0 & 1 & \infty & \infty & \infty & \infty \\ 1 & 0 & 2 & 2 & \infty & \infty \\ \infty & 2 & 0 & \infty & 2 & \infty \\ \infty & 2 & \infty & 0 & 2 & \infty \\ \infty & \infty & 2 & 2 & 0 & 1 \\ \infty & \infty & \infty & \infty & 1 & 0 \end{bmatrix} \quad (2.3)$$

A more complex example which shows this to be the case is given in Figure 2-10.

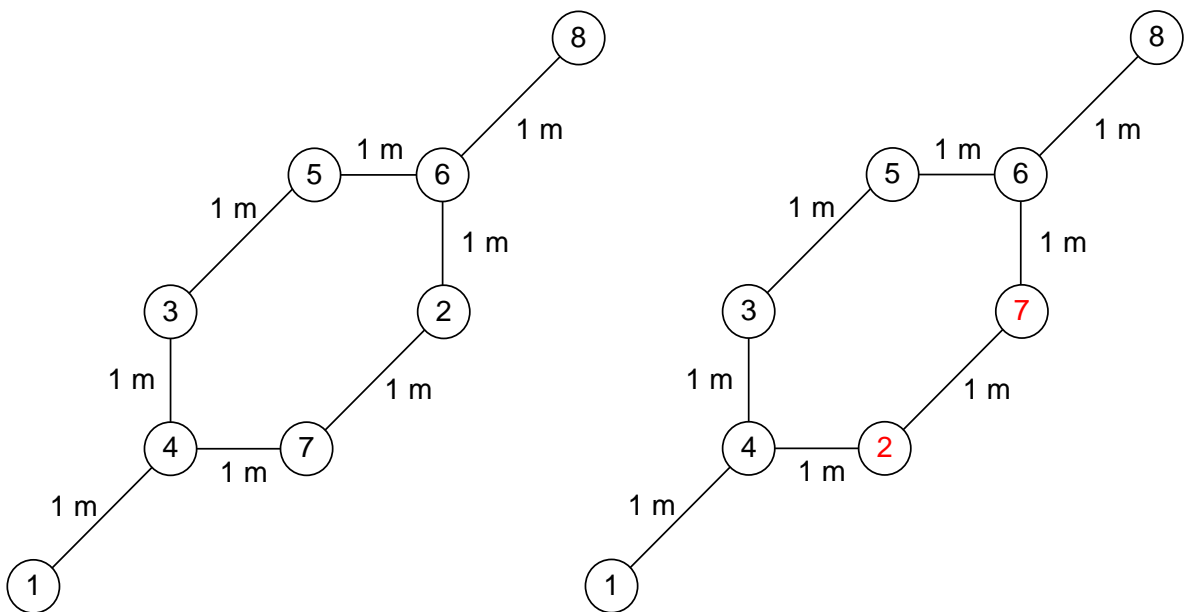


Figure 2-10 A node graph with more complexity than that of Figure 2-9.

The adjacency matrices for the node graphs show in Figure 2-10 are found in Equations (2.4) and (2.5), which represent the left and right node graphs in the figure respectively.

$$A = \begin{bmatrix} 0 & \infty & \infty & 1 & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty & 1 & 1 & \infty \\ \infty & \infty & 0 & 1 & 1 & \infty & \infty & \infty \\ 1 & \infty & 1 & 0 & \infty & \infty & 1 & \infty \\ \infty & \infty & 1 & \infty & 0 & 1 & \infty & \infty \\ \infty & 1 & \infty & \infty & 1 & 0 & \infty & 1 \\ \infty & 1 & \infty & 1 & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & 1 & \infty & 0 \end{bmatrix} \quad (2.4)$$

$$A = \begin{bmatrix} 0 & \infty & \infty & 1 & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & 1 & \infty & \infty & 1 & \infty \\ \infty & \infty & 0 & 1 & 1 & \infty & \infty & \infty \\ 1 & 1 & 1 & 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & 1 & \infty & 0 & 1 & \infty & \infty \\ \infty & \infty & \infty & \infty & 1 & 0 & 1 & 1 \\ \infty & 1 & \infty & \infty & \infty & 1 & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & 1 & \infty & 0 \end{bmatrix} \quad (2.5)$$

Having run Dijkstra's Algorithm on these two adjacency matrices, the results are as follows. For the original graph, the path taken is $1 \rightarrow 4 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 8$, but for the second graph, the path taken is $1 \rightarrow 4 \rightarrow 2 \rightarrow 7 \rightarrow 6 \rightarrow 8$. This again shows that the algorithm searches the tree with the lowest node identifier first.

As was previously mentioned in this chapter, graph theory allows for higher dimensionality path generation by reducing the dimensionality to 2 while calculating the nodes in the path. For example, Figure 2-11 maps the boundaries of a cube.

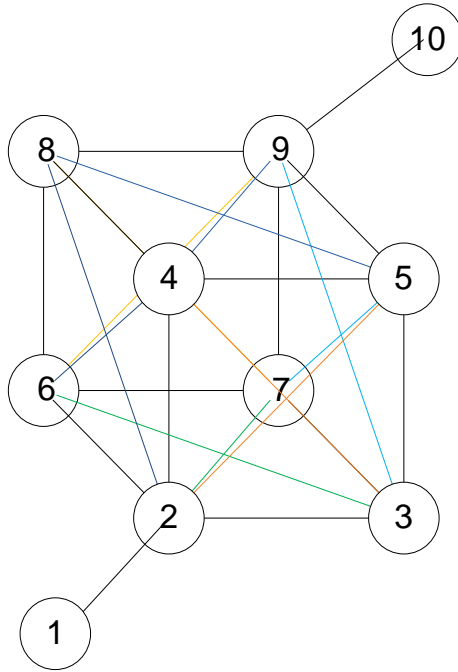


Figure 2-11 A node graph with three-dimensional coordinated.

In this graph all of the black lines have 1m cost and all of the coloured lines have $\sqrt{2}$ m cost. As can be seen in the graph, the nodes labelled 2 to 5, which could be considered to be on one plane and are in that sense a 2-D graph, have links to, and so are adjacent to the nodes labelled 6 to 9. There are 10 nodes on the graph, and this produces the 10x10 adjacency matrix shown in Equation (2.6).

$$A = \begin{bmatrix} 0 & 1 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ 1 & 0 & 1 & 1 & \sqrt{2} & 1 & \sqrt{2} & \sqrt{2} & \infty & \infty \\ \infty & 1 & 0 & \sqrt{2} & 1 & \sqrt{2} & 1 & \infty & \sqrt{2} & \infty \\ \infty & 1 & \sqrt{2} & 0 & 1 & \sqrt{2} & \infty & 1 & \sqrt{2} & \infty \\ \infty & \sqrt{2} & 1 & 1 & 0 & \infty & \sqrt{2} & \sqrt{2} & 1 & \infty \\ \infty & 1 & \sqrt{2} & \sqrt{2} & \infty & 0 & 1 & 1 & \sqrt{2} & \infty \\ \infty & \sqrt{2} & 1 & \infty & \sqrt{2} & 1 & 0 & \sqrt{2} & 1 & \infty \\ \infty & \sqrt{2} & \infty & 1 & \sqrt{2} & 1 & \sqrt{2} & 0 & 1 & \infty \\ \infty & \infty & \sqrt{2} & \sqrt{2} & 1 & \sqrt{2} & 1 & 1 & 0 & 1 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 1 & 0 \end{bmatrix} \quad (2.6)$$

This adjacency matrix has the ability to generate the graph shown in Figure 2-12, where the red lines have a cost of 1m and the black lines a cost of $\sqrt{2}$ m.

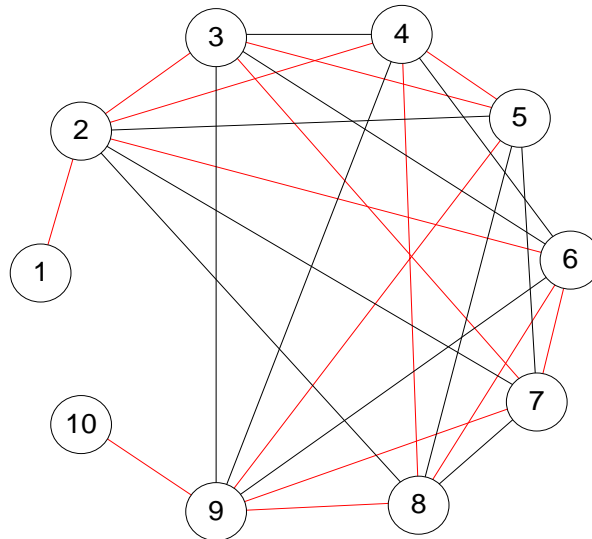


Figure 2-12 The three-dimensional node graph from Figure 2-11 flattened out into a two-dimensional node graph.

This adjacency matrix, when inputted into Dijkstra’s Algorithm, gives the optimum distance path from 1 to 10 as $1 \rightarrow 2 \rightarrow 3 \rightarrow 9 \rightarrow 10$, which by inspection of the cube graph is one of the shortest paths, with a cost of 4.4142 m. It also follows the convention of moving to the smallest node identifier first.

There are, of course, other variables that could be used to calculate the optimum path between nodes. In the example of the tank scenario, instead of the distance from one node to another, the energy requirement to get between the nodes could be used to find an energy optimum path. In Figure 2-13 the distance is denoted by the black value, and the energy is denoted by the red number.

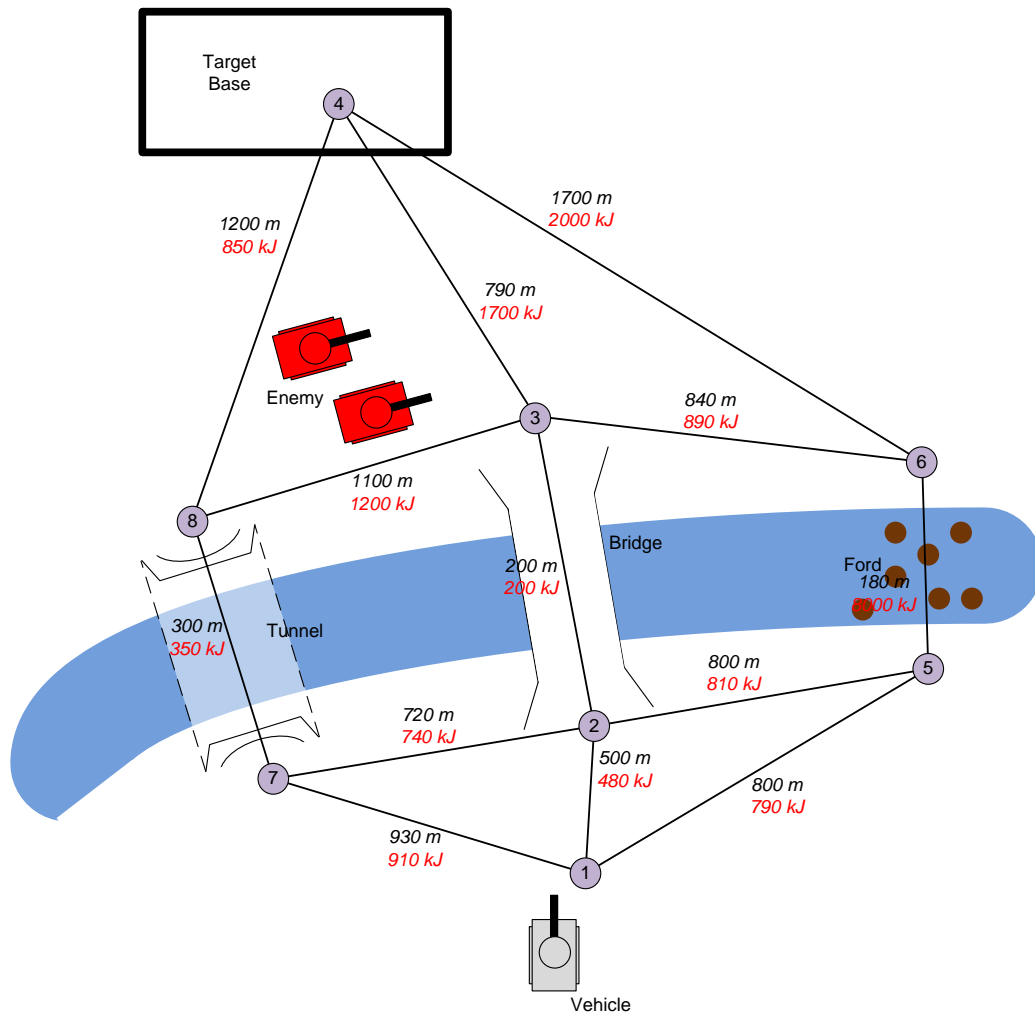


Figure 2-13 Example of an environment where graph theory can be used to generate a trajectory with energy costs as well as distance.

These values are not designed to necessarily be realistic, but follow a general rule of being similar in proportion to the lengths (a longer path will require more energy). The ford is drastically different as it would require significantly more energy to wade through the water. The target base is designed to be slightly uphill, and the lower bank of the river slopes gently downwards away from the vehicle start point towards the water. These values produce the adjacency matrix seen in Equation (2.7):

$$A = \begin{bmatrix} 0 & 480 & \infty & \infty & 790 & \infty & 910 & \infty \\ 480 & 0 & 200 & \infty & 810 & \infty & 740 & \infty \\ \infty & 200 & 0 & 1700 & \infty & 890 & \infty & 1200 \\ \infty & \infty & 1700 & 0 & \infty & 2000 & \infty & 850 \\ 790 & 810 & \infty & \infty & 0 & 8000 & \infty & \infty \\ \infty & \infty & 890 & 2000 & 8000 & 0 & \infty & \infty \\ 910 & 740 & \infty & \infty & \infty & \infty & 0 & 300 \\ \infty & \infty & 1200 & 850 & \infty & \infty & 300 & 0 \end{bmatrix} \quad (2.7)$$

This adjacency matrix produces a path from node 1 to 4 of $1 \rightarrow 7 \rightarrow 8 \rightarrow 4$, and a cost of 2060 kJ, which varies from the distance graph, which produced a path of $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$, with a cost of 1490 m.

Jun and D'Andrea (2002) (Butenko, et al., 2002) investigate the use of graph theory for path planning in UAV applications, but add the concept of decision-making. They create a probability map of the environment, which maps the probability of encountering threats and then integrate the path planning algorithm to generate the safest path through the environment.

A simple way of carrying this out would be to calculate the probability of a threat on each link of the graph, and add these probabilities, scaled by some calibrated weighting factor, to the cost of the link. In this way those paths with a higher probability of threat will have a significantly higher cost, and so would be more likely to be avoided. It would be important to calibrate the weighting factor correctly as there could be a fine balance between choosing a short route on the basis of it being the most optimum route in terms of energy consumption and it being a non-optimal route in terms of the risk to the mission completion or vehicle safety, or both. The example of the tank crossing a river will again be used to illustrate this implementation.

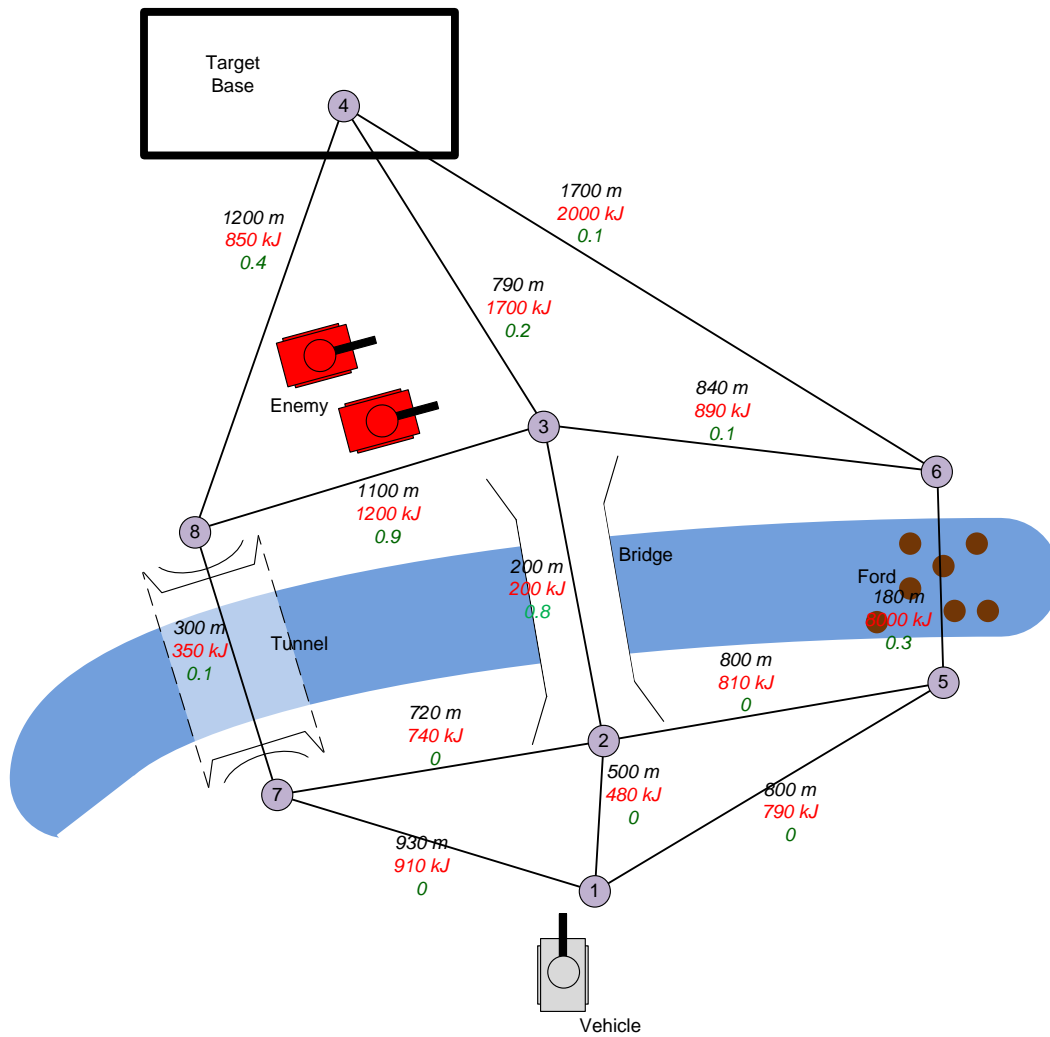


Figure 2-14 Example of an environment where graph theory can be used to generate a trajectory with risk associated costs.

The black and red values in Figure 2-14 represent the distance and energy costs of the links, respectively, and the green values represent the probability that a link will be broken. The adjacency matrices for distance and energy cost are identical to those previously outlined. The adjacency matrix for the distance is seen in Equation (2.8).

$$A = \begin{bmatrix} 0 & 500 & \infty & \infty & 800 & \infty & 930 & \infty \\ 500 & 0 & 200 & \infty & 800 & \infty & 720 & \infty \\ \infty & 200 & 0 & 790 & \infty & 840 & \infty & 1100 \\ \infty & \infty & 790 & 0 & \infty & 1700 & \infty & 1200 \\ 800 & 800 & \infty & \infty & 0 & 180 & \infty & \infty \\ \infty & \infty & 840 & 1700 & 180 & 0 & \infty & \infty \\ 930 & 720 & \infty & \infty & \infty & \infty & 0 & 300 \\ \infty & \infty & 1100 & 1200 & \infty & \infty & 300 & 0 \end{bmatrix} \quad (2.8)$$

The adjacency matrix for the energy cost is seen in Equation (2.9).

$$A = \begin{bmatrix} 0 & 480 & \infty & \infty & 790 & \infty & 910 & \infty \\ 480 & 0 & 200 & \infty & 810 & \infty & 740 & \infty \\ \infty & 200 & 0 & 1700 & \infty & 890 & \infty & 1200 \\ \infty & \infty & 1700 & 0 & \infty & 2000 & \infty & 850 \\ 790 & 810 & \infty & \infty & 0 & 8000 & \infty & \infty \\ \infty & \infty & 890 & 2000 & 8000 & 0 & \infty & \infty \\ 910 & 740 & \infty & \infty & \infty & \infty & 0 & 300 \\ \infty & \infty & 1200 & 8500 & \infty & \infty & 300 & 0 \end{bmatrix} \quad (2.9)$$

The adjacency matrix for the probabilities of link breakage is seen in Equation (2.10).

$$A = \begin{bmatrix} 0 & 0 & \infty & \infty & 0 & \infty & 0 & \infty \\ 0 & 0 & 0.8 & \infty & 0 & \infty & 0 & \infty \\ \infty & 0.8 & 0 & 0.2 & \infty & 0.1 & \infty & 0.9 \\ \infty & \infty & 0.2 & 0 & \infty & 0.1 & \infty & 0.4 \\ 0 & 0 & \infty & \infty & 0 & 0.3 & \infty & \infty \\ \infty & \infty & 0.1 & 0.1 & 0.3 & 0 & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty & \infty & 0 & 0.1 \\ \infty & \infty & 0.9 & 0.4 & \infty & \infty & 0.1 & 0 \end{bmatrix} \quad (2.10)$$

By using Equation (2.11) the probability of a break in one or more links can be taken into account when the path is generated.

$$B = A + wA_p \quad (2.11)$$

Where B is the new adjacency matrix, A is the adjacency matrix that is being used to calculate the optimum path (e.g. distance, energy cost, etc), and A_p is the adjacency matrix with the probability of link breakages. When Equation (2.11) is applied with a variety of different weighting factors, the results in Table 2-5 are generated:

As can be seen from these results, depending on the probability of a break in a link, when the weighting factor is large enough the path with the best probability of success is chosen. In the case of the distance graph, this path is different to the optimum distance path. In the case of the energy cost graph, this path is the same as the optimum energy path.

Table 2-5 Change of path across the environment depending on the size of the weighting factor added to the risk associated with each vertex.

Probability	Distance	Energy
0	1 → 2 → 3 → 4 (1490 m)	1 → 7 → 8 → 4 (2060 kJ)
1	1 → 2 → 3 → 4 (1491 m)	1 → 7 → 8 → 4 (2060.5 kJ)
10	1 → 2 → 3 → 4 (1500 m)	1 → 7 → 8 → 4 (2065 kJ)
100	1 → 2 → 3 → 4 (1590 m)	1 → 7 → 8 → 4 (2110 kJ)
1000	1 → 2 → 3 → 4 (2490 m)	1 → 7 → 8 → 4 (2560 kJ)
10000	1 → 5 → 6 → 4 (6680 m)	1 → 7 → 8 → 4 (7060 kJ)

2.4 Summary of Literature

In the literature, the most popular method of robotic manipulator control is to use a PID controller, but since systems of this type display highly non-linear characteristics and joints in a manipulator arm are dynamically linked the control has to be adaptive to provide adequate control in many different geometries and angular velocities. Several of the investigated literature attempt to solve the problem of gain selection online during manipulator operation, and different optimisation methods are used to solve this problem.

Regarding navigation through environments for robotic manipulators, the most commonly used principle is to carry out the path generation in C-Space rather than T-space. Working in C-Space is advantageous since any generated path is produced in terms of the direct control requirements of the manipulator arm rather than requiring to be converted into its control requirements following the generation of the path. The second great advantage of working in C-Space rather than T-space relates to the need to consider the entire arm for collision avoidance and not just the end effector.

Objects in T-space can be considered to be regions of space where no part of the arm can traverse. To prevent any collisions between the arm and these impermissible regions of space and the entire arm, the locations of at least all of the joints in the arm, and possibly the entire arm must be considered to generate a series of arm configurations which must then be resolved into the control parameters (i.e. joint angles).

Conversely, any impermissible regions in T-space must correspond with a set of joint angles which are also forbidden. Given that each object is continuous and also the arm is continuous, each solution set for the impermissible angle ranges caused by an object must also be continuous. By converting obstacles from T-space into C-Space, all collision information from the entire arm with each obstacle is now contained in the C-Space version of the obstacles in an n-dimensional space, where n is the number of control parameters (joint angles) of the arm, in this case 3. This allows the entire arm to be considered as a point mass in C-Space and only 1 path has to be generated from the initial arm configuration to the desired arm configuration which already exists in the control domain rather than multiple paths in

T-space which must then be converted into a single path in C-Space following the path generation.

With the decision to operate in C-Space, the path planning problem becomes one of a standard 3-dimensional obstacle avoidance problem. This allows for the use of a lot of established techniques which would have been impractical when generating a path in T-space.

Having carried out a survey of available techniques, the decision has been taken to implement path generation for a robotic manipulator using graph theory. This is because the node graph generation can be carried out very simply given the way in which the impermissible regions and permissible boundaries of obstacles are calculated using triangulation. Also, graph theory allows for the path planning problem to become two-dimensional which will reduce the computational requirements of the trajectory generation and hence speed up the run time of the technique.

Based upon the conclusions made in this section the following decisions have been made. The guidance method will be carried out in C-Space and graph theory will be used to generate a path through C-Space. Simulated sensor data about obstacles will be assumed to be noiseless; therefore the data inputted to the guidance method will be the exact location of the measured points in space. Derivation of the dynamic model of a manipulator arm will be carried out using Newtonian mechanics owing to its smaller computational overheads.

In the following chapters, the decisions made as a result of the review of literature will be implemented in order to carry out the objectives determined in Chapter 1. Chapters 3 and 4 derive the kinematic and dynamic models of the robotic manipulator arm respectively and Chapter 5 implements a suitable control schema for the dynamic model. Chapter 6 deals with the formation of a map in C-space, given T-space obstacle data. Chapter 7 is responsible for the implementation of a path planning method which will use the C-space map to safely plan a route for the manipulator arm through the environment. Chapter 8 presents the results of a series of simulations which are then used to validate the effectiveness of the combined dynamic system and guidance method and Chapter 9 presents the findings of the research undertaken in this thesis.

3 KINEMATIC MODELLING OF ROBOTIC MANIPULATORS

In the Introduction chapter to this thesis, the problems surrounding guidance and control of a 3-DoF robotic manipulator arm in real time in close-proximity environments has been discussed. The remainder of this thesis deals with the development of a technique which will satisfy this goal. A starting point in achieving the aims and objectives laid down in the Introduction would be to investigate the development of kinematic and dynamic models of a 3-DoF manipulator arm for use both as a test bed for the control and path tracking ability of the arm model, and also as a means of developing a guidance method which is suitable for use in this type of navigation problem. In this chapter a forward and inverse kinematic model is developed for use in the remainder of the thesis. This chapter deals with the block highlighted in red in Figure 1-4, which is displayed again here with all of the other processes greyed out.

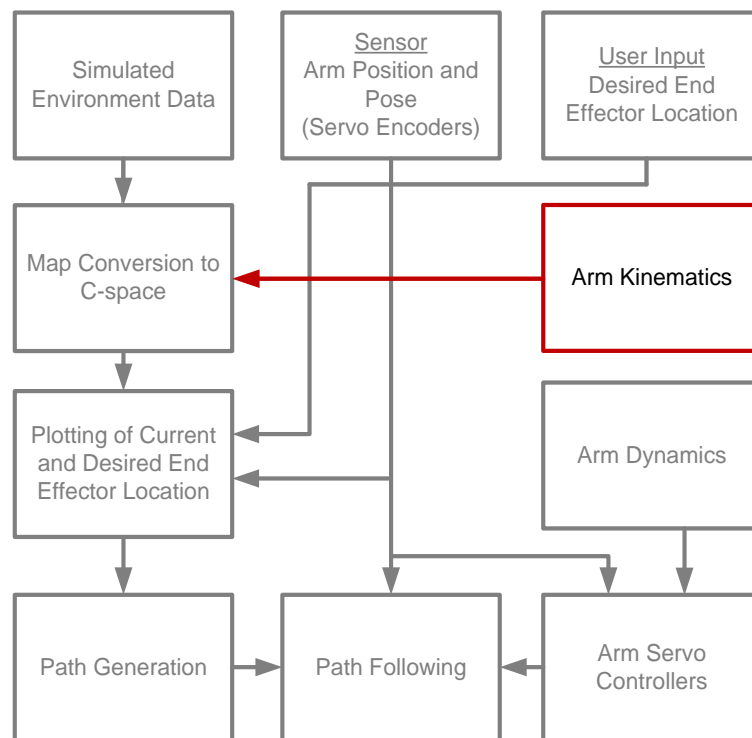


Figure 3-1 Manipulator arm kinematics (red) in relation to the overall guidance method.

3.1 Overview

Having investigated literature on kinematic modelling of robotic manipulator arms, some of the information contained there can be used to develop forward and inverse kinematic models of the robotic manipulator, which can then be used in Chapter 6 for the conversion of T-space obstacle data into a C-space environment map. The kinematic modelling is carried out in three ways, the first by trigonometry, the second by Denavit-Hartenberg Parameters and the third by solution to simultaneous equations.

The development of the forward kinematics allows for the modelling of the arm so that given the angles between each link in the arm (provided by servo encoders) the states of the arm can be calculated, while the development of the inverse kinematics allows for the control of the arm given a demand position of each pivot point by calculating the required angles between each of the links to do so. There are three methods of carrying out these calculations that are dealt with in this chapter, and these kinematic models will be used in Chapter 3 when converting the environment in T-space into C-space, and one of the methods will be selected for use. The first method deals with the kinematics using trigonometry and geometry. The second method, described by Rosales and Gan (2002), uses matrix transforms to derive the forward kinematics and then manipulates them into simultaneous equations to solve the inverse kinematics problem. The main advantage of the trigonometric method is that it requires much less calculation than the matrix transform method, and so is more memory efficient, but the matrix transform method deals with the orientation of the axes at each of the joints dealt with, providing more information, which may be useful in terms of the manipulator control. The third method uses the geometric parameters of the robotic manipulator arm to form a series of simultaneous equations that can be solved to calculate the Euclidean positions of each joint given the manipulator base point and end effector location. The vectors between each point can then be used to calculate the joint angles.

The arm modelled in this chapter has three links, which can be seen from Figure 3-2. The first link rotates about the Z-axis at a pivot at its base. The second link is attached to the first link at a joint 0.090 m from the base of the first link, which operates in the X-Y plane. The second link rotates about an axis which is parallel to

the X-Y plane and perpendicular to the direction of the first link. The third link is attached to the second link at a joint 0.332 m from the joint between the first and second links. The third link rotates about an axis which is parallel to the X-Y plane and the axis of rotation of the second joint, and perpendicular to the plane in which the second and third links operate. The second and third links operate in a plane which is formed by the X-Y direction of the first link and the Z-axis. The third link is 0.538 m long and the end effector is located at the end of this link.

3.2 Trigonometric Method

The manipulator arm used for this research consists of three straight links of fixed length, the positions of which are controlled by the angles between them at their joints. Because all of the lengths and angles are known, it is very simple to calculate the position of each joint and the end effector in relation to the origin of the arm by using trigonometry.

3.2.1 Forward Kinematics

This method requires the arm to be defined with a series of parameters, the length of each link and the angles between them, and the position of the origin of the arm. The positions of each of the joints, and the end effector are the outputs of the calculations. The joint angles will be defined as α , β and γ and the link lengths will be defined as l_0 , l_1 and l_2 . The position of the origin is defined as P_0 and the positions of the intermediate joints are P_1 and P_2 . The position of the end effector is defined as P_f . This is shown in Figure 3-2.

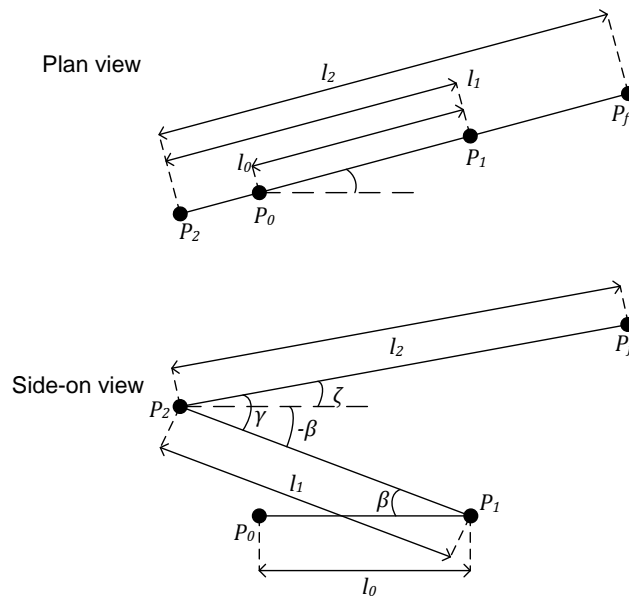


Figure 3-2 Schematic of the variables and dimensions that make up the simple model of the Digital Vanguard ROV manipulator arm.

The parameters are outlined in Table 3-1.

Table 3-1 Parameters of the links in the manipulator arm.

Link Label	Joint	Joint Angle	Link Length
0	P_0	α	$l_0 = 0.09$
1	P_1	β	$l_1 = 0.332$
2	P_2	γ	$l_2 = 0.09$

Angle γ is defined as the angle between links 1 and 2, and must be given in terms of link 2 and the horizontal. This can be represented in terms of angle β as:

$$\zeta = \gamma + (-\beta) \tag{3.1}$$

The position vectors of P_1 , P_2 and P_f are calculated in Equation (3.2).

$$P_1 = P_0 + \begin{bmatrix} l_0 \cos \alpha \\ l_0 \sin \beta \\ 0 \end{bmatrix} \quad (3.2)$$

$$P_2 = P_0 + \begin{bmatrix} (l_0 + l_1 \cos \beta) \cos \alpha \\ (l_0 + l_1 \cos \beta) \sin \beta \\ l_1 \sin \beta \end{bmatrix} \quad P_f = P_0 + \begin{bmatrix} (a_0 + l_1 \cos \beta + l_2 \cos \zeta) \cos \alpha \\ (a_0 + l_1 \cos \beta + l_2 \cos \zeta) \sin \beta \\ l_1 \sin \beta + l_2 \sin \zeta \end{bmatrix}$$

These equations have been derived using trigonometry, with P_1 being calculated with respect to P_0 . P_2 has been calculated with respect to P_1 and then converted to be with respect to P_0 by substitution. The same is the case for P_f . A figure showing a graphical representation of the simple manipulator model is shown in Figure 3-3.

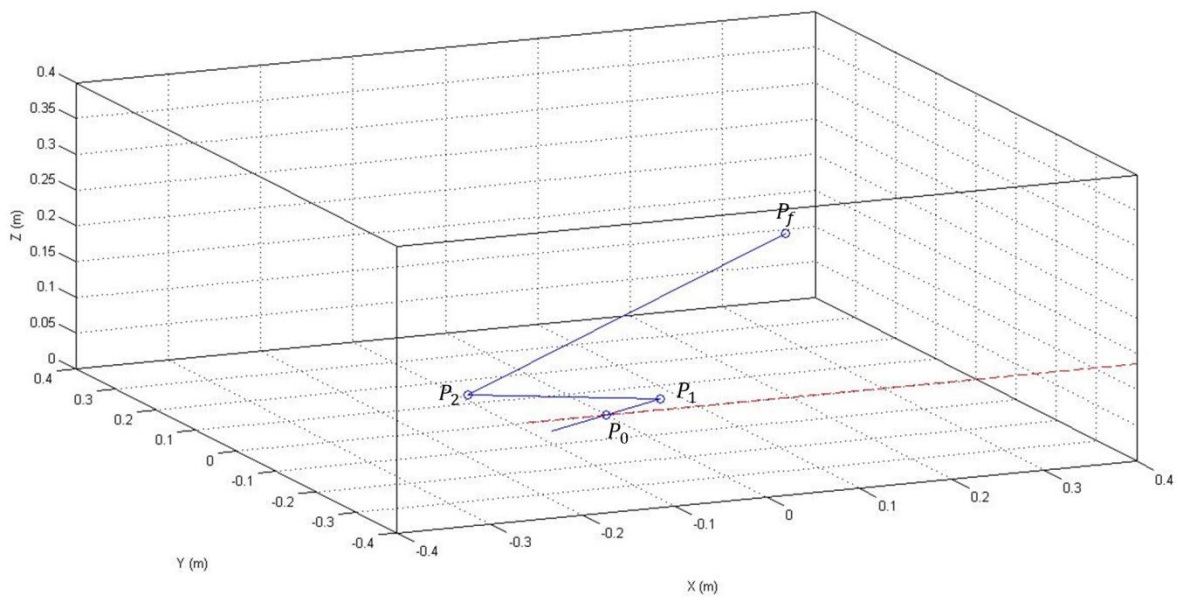


Figure 3-3 Graphical representation of the robotic arm. The red dotted line is the representation of the arm when fully extended.

3.2.2 Inverse Kinematics

The inverse kinematics can also be calculated very easily using trigonometry and geometry. The control of the end effector position can be envisaged as controlling the direction of the end point in the X-Y plane by varying angle α , and the distance of the end effector position from the origin in the X-Y plane and the Z-direction using β and γ . Because of this the angle α can be calculated using the vector from the origin to the end effector in the X-Y plane. This then simplifies the problem of the inverse kinematics to a two-dimensional one which is solvable using the cosine rule.

The first part of the derivation is to calculate angle α . This is carried out using the vector between P_0 and P_f in the X-Y plane, as shown in Figure 3-4.

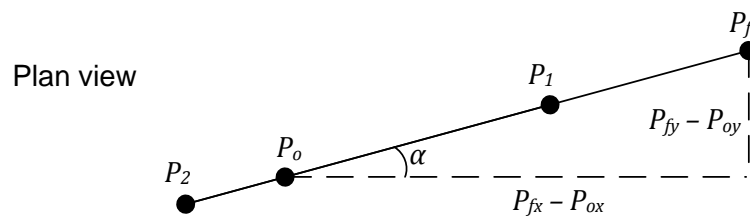


Figure 3-4 Diagram of the calculation of α using P_f and P_o

The calculation of α requires only the tangent function as shown in Equation (3.3):

$$\alpha = \arctan2\left(\frac{P_{fy} - P_{oy}}{P_{fx} - P_{ox}}\right) \quad (3.3)$$

where P_{fx} and P_{fy} are the X and Y coordinates of position P_f respectively, and P_{ox} and P_{oy} are the X and Y coordinates of position P_o respectively. This leaves the problem requiring the solution to the two remaining joint angles, with known positions of the end manipulator P_f and the first joint P_1 . These unknowns can all be solved using geometry and trigonometry, as shown in Figure 3-5.

Side-on view

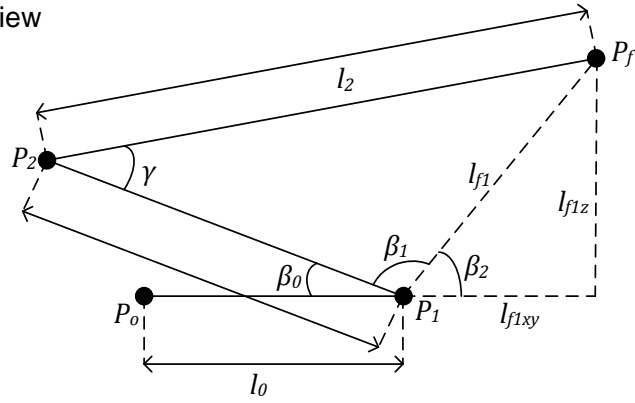


Figure 3-5 Diagram of the calculation of β and γ using P_f and P_1

In order to calculate β_2 and γ the cosine rule must be used, and both require the length l_{f1} to be known. The length l_{f1} and angle β_2 require the lengths l_{f1xy} and l_{f1z} to be calculated by subtraction of the point P_1 from P_f as:

$$l_{f1xy} = \sqrt{(P_{fx} - P_{1x})^2 + (P_{fy} - P_{1y})^2} \quad (3.4)$$

$$l_{f1z} = P_{fz} - P_{1z}$$

The angle β_1 can be calculated using the inverse tangent of these two lengths and l_{f1} can be calculated using Pythagoras' Theorem for these two lengths as:

$$\beta_1 = \arctan2\left(\frac{l_{f1z}}{l_{f1xy}}\right) \quad (3.5)$$

$$l_{f1} = \sqrt{l_{f1xy}^2 + l_{f1z}^2}$$

The cosine rule can now be applied to find the angles β_2 and γ , shown in Equations (3.6) and (3.8):

$$\beta_2 = \arccos\left(\frac{l_1^2 + l_{f1}^2 - l_2^2}{2l_1l_{f1}}\right) \quad (3.6)$$

$$\gamma = \arccos\left(\frac{l_1^2 + l_2^2 - l_{f1}^2}{2l_1l_{f1}}\right) \quad (3.7)$$

Finally angle β_0 can be calculated in Equation (3.8).

$$\beta_0 = \pi - \beta_1 - \beta_2 \quad (3.8)$$

3.3 Denavit-Hartenberg (Matrix Transform) Method

The kinematics of the robotic manipulator can also be represented in terms of the rotation and of the axes at each pivot into the next pivot. When all the transforms are combined, a transform representing the rotation and translation of the reference axes to the axes at the end effector is derived. The advantage of this is that both the position and the orientation of the end effector are described.

3.3.1 The parameters of the robotic arm

Before the kinematics of the arm are derived, the body axes at each joint must be defined. As seen below in Figure 3-6 and Figure 3-7 the axis about which the rotation is made to move the link is defined as the Z-axis. The X-axis is defined as being in the direction of the length of the link between the first and second joints of the arm, and the Y-axis is defined using the other two axes in a right-handed set.

The robotic arm in use in the project uses three links. The first rotates about the axis perpendicular to the body frame, while the second two rotate about an axis which is rotated by 90° and perpendicular to the longitudinal length of the first link. This is shown in Figure 3-6 and Figure 3-7, which give an illustration of the axes used at each pivot on the link.

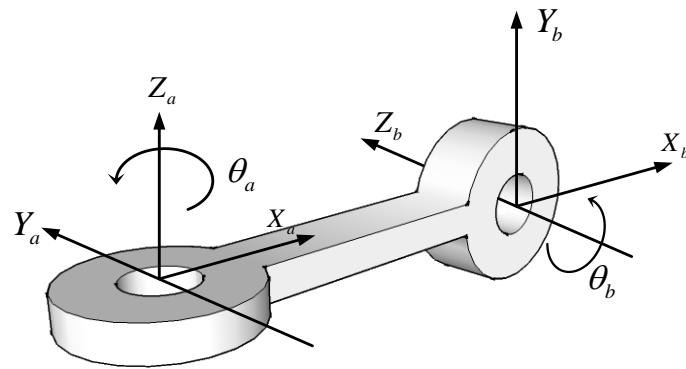


Figure 3-6 Configuration of the first link in the manipulator arm.

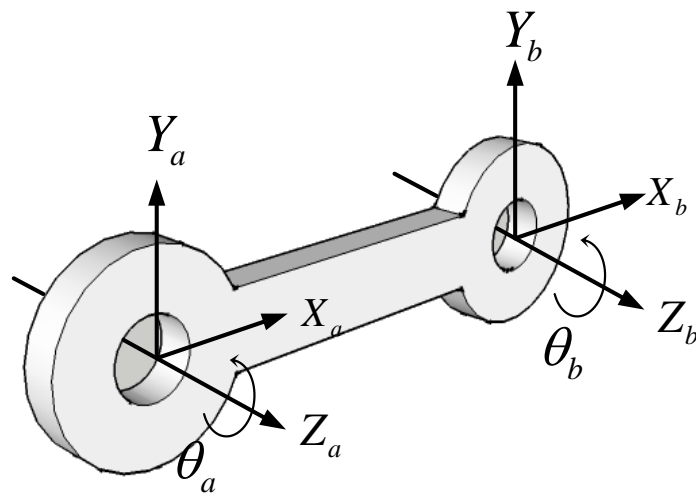


Figure 3-7 Configuration of the second and third links in the manipulator arm.

The transformation from axis-a of each link to axis-b of each link can be described using a transformation matrix $A_0^1 = \begin{bmatrix} R_{3 \times 3} & P_{3 \times 1} \\ f_{1 \times 3} & W_{1 \times 1} \end{bmatrix}$, where R is the rotation cosine

matrix that describes the rotation of the axes, P is the translation which is a vector describing the change in position of the origin of the axes, f is the change in perspective of the axes, and W is the change in scale of the axes. In the case of the manipulator arm, the scale of the axes does not change, so W is set as 1, and the perspective of the axes remains the same, therefore $f = [0 \ 0 \ 0]$. This means that the parameters needed for each link to calculate the position and orientation of each pivot and the end effector are the lengths of each of the pivots, the rotation of the axes from one end of the link to the other caused by the shape of the link, and the rotation about Z_0 by angle θ_0 .

It is important to recognise the direction and reference point of each of the angles as they may not be defined as the angle between the physical parts of the links being connected by that joint. This is shown in Figure 3-8.

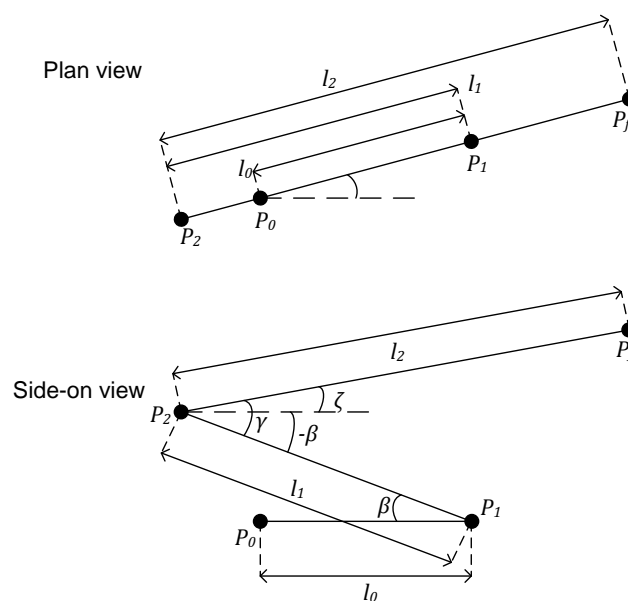


Figure 3-8 Schematic of the variables and dimensions that make up the model of the manipulator arm.

The angle of rotation about the Z -axis is defined in the kinematic model as being an anti-clockwise rotation starting in the direction of the length of the previous link. This has no effect on α but has a significant effect on the angles that are represented by β and γ when compared to those specified previously in Section 3.3.3. Angles δ and

ε are the angles which would preferably be used to control the state of each of the links, but the way in which each of the angles are specified mean that β and γ are the angles used in the kinematics. Angle γ shown on the diagram is in the negative direction, hence the negative sign. The angles δ and ε can be represented in terms of β and γ using the following geometric relationships:

$$\delta = 180 - \beta \tag{3.9}$$

$$\varepsilon = \gamma + 180$$

The parameters used to derive the kinematics of the manipulator arm can be outlined more easily in the Table 3-2.

Table 3-2 Parameters of the links in the manipulator arm.

Link Number	Joint	Joint Angle	Link Length (m)	θ
1	P_0	α	$a_0 = 0.09$	90°
2	P_1	$\beta = 180 - \delta$	$a_1 = 0.332$	0°
3	P_2	$\gamma = \varepsilon - 180$	$a_2 = 0.538$	0°

Where the joint angle is the angle of rotation about the Z_a -axis, the link length is the length of the link (taken from the intersection between the a-axes, to the intersection between the b-axes) and θ is the rotation of the axes about the X-axis due to the geometry of the link.

This information can be used to construct the transformation matrices to find the position of each pivot and the end manipulator in terms of the reference frame and the rotation angle of each pivot.

3.3.2 Derivation of the transformation matrices for each arm link

This section describes the derivation of the transformation matrices which provide the forward kinematics for the manipulator arm. This is done by constructing a transformation matrix for each link in the arm separately.

To transform the axes of the first link from the reference frame to that of the end pivot point requires two separate rotation transforms due to the rotation of the axes by 90° about the X-axis. The axes X_0 , Y_0 and Z_0 are transformed into a set of intermediate axes, X_0' , Y_0' and Z_0' by the rotation about Z_0 by angle α , which can then be rotated by 90° about the X_0' -axis into the axes X_1 , Y_1 and Z_1 . This is shown in

Figure 3-9 and

Figure 3-10.

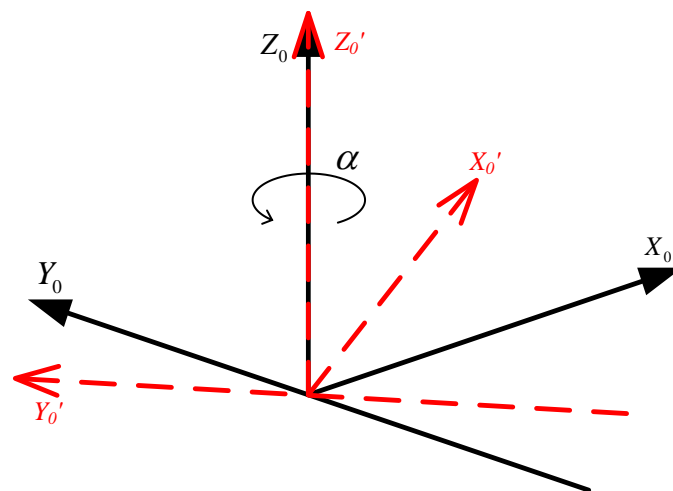


Figure 3-9 Rotation of the X_0 and Y_0 axes about the Z_0 axis by α to transform the axes into the intermediate axes X_0' , Y_0' and Z_0' .

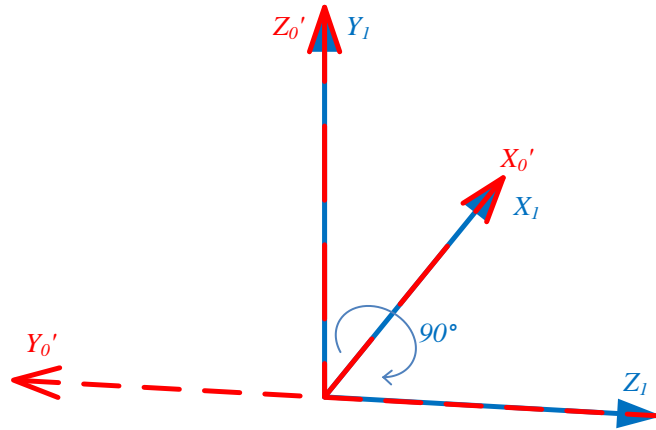


Figure 3-10 Rotation of the Y_0' and Z_0' axes about the X_0' axis by 90° into the X_1 , Y_1 and Z_1 axes.

The rotation shown in

Figure 3-9 can be described by the direction cosine matrix in Equation (3.10).

$$\begin{bmatrix} X_0' \\ Y_0' \\ Z_0' \end{bmatrix} = \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} \quad (3.10)$$

The rotation shown in

Figure 3-10 can be described by the direction cosine matrix in Equation (3.11).

$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos 90^\circ & \sin 90^\circ \\ 0 & -\sin 90^\circ & \cos 90^\circ \end{bmatrix} \begin{bmatrix} X_0' \\ Y_0' \\ Z_0' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} X_0' \\ Y_0' \\ Z_0' \end{bmatrix} \quad (3.11)$$

The complete rotation from the 0-axes to the 1-axes can be carried out using Equation (3.12).

$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} \quad (3.12)$$

$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \begin{bmatrix} \cos \alpha & 0 & \sin \alpha \\ \sin \alpha & 0 & -\cos \alpha \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix}$$

The translation of the axes from the 0-axes to the 1-axes follows simple trigonometry.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} l_0 \cos \alpha \\ l_0 \sin \alpha \\ 0 \end{bmatrix} \quad (3.13)$$

This allows for the construction of the transformation matrix seen in Equation (3.14).

$$A_0^1 = \begin{bmatrix} \cos \alpha & 0 & \sin \alpha & l_0 \cos \alpha \\ \sin \alpha & 0 & -\cos \alpha & l_0 \sin \alpha \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.14)$$

The transformation of the axes of the second and third links from their original frames to that of their end pivots requires only a single rotation, where the X_a and Y_a axes are rotated about the Z_a axis by β or γ for links 2 and 3 respectively into the X_b , Y_b and Z_b axes. This is shown in Figure 3-11.

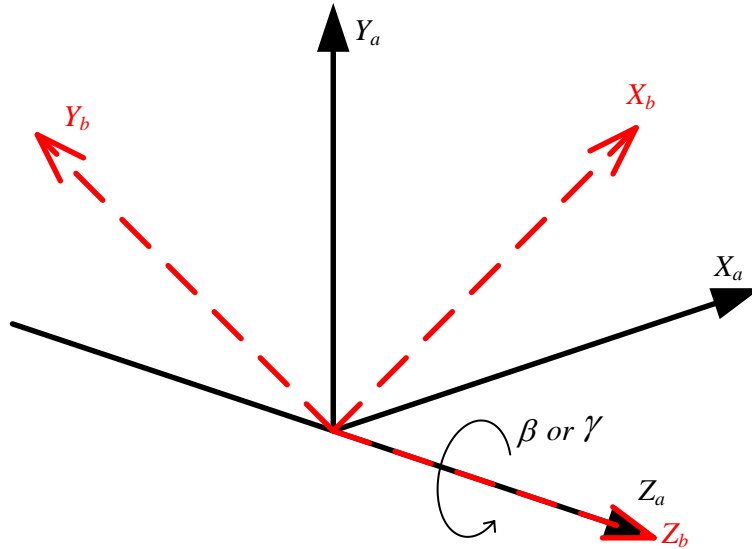


Figure 3-11 Rotation of the $X_a Y_a$ axes around the Z_a axis by β or γ into the X_b, Y_b and Z_b axes.

The rotation shown in Figure 3-11 can be described by the direction cosine matrix found in Equation (3.15).

$$\begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_a \\ Y_a \\ Z_a \end{bmatrix} \quad (3.15)$$

where θ is the angle of rotation about the Z-axis in either link. The translation of the axes from the a-axes to the b-axes again follows simple trigonometry.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} l \cos \theta \\ l \sin \theta \\ 0 \end{bmatrix} \quad (3.16)$$

where l is the length of the link. This allows for the construction of the transformation matrix in Equation (3.17).

$$A_a^b = \begin{bmatrix} \cos \theta & 0 & -\sin \theta & l \cos \theta \\ \sin \theta & 0 & \cos \theta & l \sin \theta \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.17)$$

This transformation matrix can be implemented for links 2 and 3, producing the transformation matrices seen in Equations (3.18) and (3.19) respectively.

$$A_1^2 = \begin{bmatrix} \cos \beta & -\sin \beta & 0 & l_1 \cos \beta \\ \sin \beta & \cos \beta & 0 & l_1 \sin \beta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.18)$$

$$A_2^f = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 & l_2 \cos \gamma \\ \sin \gamma & \cos \gamma & 0 & l_2 \sin \gamma \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.19)$$

3.3.3 The forward kinematics of the robotic manipulator arm.

To model the complete forward kinematics of the robotic manipulator arm, these transformation matrices must be combined to find the position and orientation of each of the pivots in terms of the original reference frame. This is done by multiplying each of the transforms together:

$$T_0^1 = A_0^1 = \begin{bmatrix} \cos \alpha & 0 & \sin \alpha & l_0 \cos \alpha \\ \sin \alpha & 0 & -\cos \alpha & l_0 \sin \alpha \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.20)$$

$$\begin{aligned}
T_0^2 &= A_0^1 A_1^2 \\
&= \begin{bmatrix} \cos \alpha \cos \beta & -\cos \alpha \sin \beta & \sin \alpha & \cos \alpha (l_0 + l_1 \cos \beta) \\ \sin \alpha \cos \beta & -\sin \alpha \sin \beta & -\cos \alpha & \sin \alpha (l_0 + l_1 \cos \beta) \\ \sin \beta & \cos \beta & 0 & l_1 \sin \beta \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.21)
\end{aligned}$$

$$\begin{aligned}
T_0^f &= A_0^1 A_1^2 A_2^f \\
&= \begin{bmatrix} \cos \alpha \cos(\beta + \gamma) & -\cos \alpha \sin(\beta + \gamma) & \sin \alpha & \cos \alpha (l_0 + l_1 \cos \beta + l_2 \cos(\beta + \gamma)) \\ \sin \alpha \cos(\beta + \gamma) & -\sin \alpha \sin(\beta + \gamma) & -\cos \alpha & \sin \alpha (l_0 + l_1 \cos \beta + l_2 \cos(\beta + \gamma)) \\ \sin(\beta + \gamma) & \cos(\beta + \gamma) & 0 & l_1 \sin \beta + l_2 \sin(\beta + \gamma) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.22)
\end{aligned}$$

In transforms T_0^2 and T_0^f , β and γ can be replaced by δ and ε using the relationships described in equation (4.14). The forward kinematics can be used in numerical simulation to calculate the position and orientation of each of the joints. The transform T_0^n can be equated to a generic transform T_H^R which is given in Equation (3.23).

$$T_H^R = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.23)$$

By equating each transform to the T_H^R matrix, a set of nine equations is created for the variables in the T_H^R matrix. Following a set of rules given in (Rosales & Gan, 2002), the orientation of each of the joints can be calculated as a series of angles representing roll, yaw and pitch. The rules used to calculate the orientation of the joints are provided in Equation (3.24).

$$\text{if } (n_x = 0 \wedge n_y = 0) \Rightarrow \begin{cases} \theta_r = \arctan2(o_x, o_y) \\ \theta_y = 90^\circ \\ \theta_p = 0^\circ \end{cases} \quad (3.24)$$

$$\text{if } (n_x \neq 0 \vee n_y \neq 0) \Rightarrow \begin{cases} \theta_r = \arctan2(o_z, a_z) \\ \theta_y = \arctan2(n_y, n_x) \\ \theta_p = \arctan2(-n_z, \sqrt{n_x^2 + n_y^2}) \end{cases}$$

where θ_r, θ_y and θ_p are the angles of roll, yaw and pitch respectively.

3.3.4 The inverse kinematics of the robotic manipulator arm.

The inverse kinematics allow for the calculation of the required joint angles given a demand manipulator end position and orientation. This position and pose is defined in the X, Y and Z directions and roll, yaw and pitch, represented by $P_x, P_y, P_z, \theta_r, \theta_y$ and θ_p respectively. This, however, is not in a form that is compatible with the T-matrix, and so must be converted to the format of the T_H^R matrix. This is done by combining direction cosine matrices representing the rotations about each of the axes as:

$$\begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_r & \sin \theta_r \\ 0 & -\sin \theta_r & \cos \theta_r \end{bmatrix} \begin{bmatrix} \cos \theta_y & 0 & -\sin \theta_y \\ \sin \theta_y & 0 & \cos \theta_y \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \cos \theta_p & -\sin \theta_p & 0 \\ \sin \theta_p & \cos \theta_p & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_a \\ Y_a \\ Z_a \end{bmatrix} \quad (3.25)$$

$$\begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix} = \begin{bmatrix} \cos \theta_y \cos \theta_p & -\cos \theta_y \sin \theta_p & -\sin \theta_y \\ \sin \theta_r \sin \theta_p + \cos \theta_r \sin \theta_y \cos \theta_p & \sin \theta_r \cos \theta_p - \cos \theta_r \sin \theta_y \sin \theta_p & \cos \theta_r \cos \theta_y \\ \cos \theta_r \sin \theta_p - \sin \theta_r \sin \theta_y \cos \theta_p & \cos \theta_r \cos \theta_p + \sin \theta_r \sin \theta_y \sin \theta_p & -\sin \theta_r \cos \theta_y \end{bmatrix} \begin{bmatrix} X_a \\ Y_a \\ Z_a \end{bmatrix}$$

The direction cosine matrix and the demand position calculated using Equation (3.25) can then be combined for the demand T-matrix in Equation (3.26).

$$T_H^R = \begin{bmatrix} \cos \theta_y \cos \theta_p & -\cos \theta_y \sin \theta_p & -\sin \theta_y & P_x \\ \sin \theta_r \sin \theta_p + \cos \theta_r \sin \theta_y \cos \theta_p & \sin \theta_r \cos \theta_p - \cos \theta_r \sin \theta_y \sin \theta_p & \cos \theta_r \cos \theta_y & P_y \\ \cos \theta_r \sin \theta_p - \sin \theta_r \sin \theta_y \cos \theta_p & \cos \theta_r \cos \theta_p + \sin \theta_r \sin \theta_y \sin \theta_p & -\sin \theta_r \cos \theta_y & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.26)$$

To derive the inverse kinematics, the T_H^R matrix in Equation (3.23) is equated to the forward transforms derived in Equations (3.20), (3.21) and (3.22). This forms a series of simultaneous equations in each case which can be solved to find the required joint angles.

Inverse kinematics for joint P_1

Equation (3.27) deals with the kinematics of the joint at the end of the first link:

$$T_H^R = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos \alpha & 0 & \sin \alpha & a_0 \cos \alpha \\ \sin \alpha & 0 & -\cos \alpha & a_0 \sin \alpha \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.27)$$

From Equation (3.27) each of the elements can be equated to find several simultaneous equations. The only simultaneous equations necessary for calculating α are either n_x and n_y or a_x and a_y . In this case the latter have been used and rearranged to find α in Equation (3.28):

$$\begin{aligned} a_x &= \sin \alpha & a_y &= -\cos \alpha \\ \frac{a_x}{-a_y} &= \frac{\sin \alpha}{\cos \alpha} = \tan \alpha & \alpha &= \arctan2(a_x, -a_y) \end{aligned} \quad (3.28)$$

The demand a_x and a_y can be substituted by those derived in Equation (3.26) to find angle α in terms of roll, yaw and pitch, as in Equation (3.29).

$$a_x = -\sin \theta_y \qquad a_y = \cos \theta_r \cos \theta_y \qquad (3.29)$$

$$\alpha = \arctan2(-\sin \theta_y, -\cos \theta_r \cos \theta_y)$$

Inverse kinematics for joint P₂

Equation (3.30) deals with the kinematics of the joint at the end of the second link.

$$T_H^R = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos \alpha \cos \beta & -\cos \alpha \sin \beta & \sin \alpha & \cos \alpha (a_0 + a_1 \cos \beta) \\ \sin \alpha \cos \beta & -\sin \alpha \sin \beta & -\cos \alpha & \sin \alpha (a_0 + a_1 \cos \beta) \\ \sin \beta & \cos \beta & 0 & a_1 \cos \beta \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.30)$$

In this case α can still be solved using a_x and a_y , giving the same result as Equation (3.29). Solving for β can be done using p_z , a_y and p_x as shown in Equations (3.31) to (3.37):

$$p_x = a_1 \sin \beta \Rightarrow \sin \beta = \frac{p_x}{a_1} \qquad (3.31)$$

$$a_y = -\cos \alpha \Rightarrow \cos \alpha = -a_y \qquad (3.32)$$

Substituting $\cos \alpha$ for a_y removes α giving,

$$p_x = -a_y(a_0 + a_1 \cos \beta) \qquad (3.33)$$

which can then be rearranged to make $\cos \beta$ the subject as:

$$\cos \beta = \frac{\left(\frac{p_x}{-a_y} - a_0\right)}{a_1} \qquad (3.34)$$

Equations (3.31) and (3.40) can be combined to find β .

$$\frac{\sin \beta}{\cos \beta} = \frac{p_z a_1}{a_1 \left(\frac{p_x}{-a_y} - a_0 \right)} = \tan \beta \quad (3.35)$$

$$\tan \beta = \frac{p_z}{\left(\frac{p_x}{-a_y} - a_0 \right)} \frac{-a_y}{-a_y} = \frac{-p_z a_y}{p_x + a_0 a_y} \quad (3.36)$$

$$\beta = \arctan2(-p_z a_y, p_x + a_0 a_y) \quad (3.37)$$

Again, the relationships seen in Equation (3.26) can be substituted into the above equation to find β in terms of roll, yaw and pitch.

$$\beta = \arctan2(-p_z \cos \theta_r \cos \theta_y, p_x + a_0 \cos \theta_r \cos \theta_y) \quad (3.38)$$

Inverse kinematics for joint P_f

As has been done in Sections 0 and 0, the process can be repeated for the inverse kinematics of the arm given the demand position and orientation of P_f .

$$T_H^R = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos \alpha \cos(\beta + \gamma) & -\cos \alpha \sin(\beta + \gamma) & \sin \alpha & \cos \alpha (l_0 + l_1 \cos \beta + l_2 \cos(\beta + \gamma)) \\ \sin \alpha \cos(\beta + \gamma) & -\sin \alpha \sin(\beta + \gamma) & -\cos \alpha & \sin \alpha (l_0 + l_1 \cos \beta + l_2 \cos(\beta + \gamma)) \\ \sin(\beta + \gamma) & \cos(\beta + \gamma) & 0 & l_1 \sin \beta + l_2 \sin(\beta + \gamma) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.39)$$

Again angle α can be found using a_x and a_y to give the same result as in Equation (3.29). To find β , p_z and p_y are used and n_z , o_x and o_z are substituted in for various

variables to reduce the unknowns to $\cos \beta$ and $\sin \beta$, as shown in Equations (3.40) and (3.41).

$$\begin{aligned}
 p_z &= l_1 \sin \beta + l_2 \sin(\beta + \gamma) \\
 n_z &= \sin(\beta + \gamma) \\
 p_z &= l_1 \sin \beta + l_2 n_z
 \end{aligned} \tag{3.40}$$

$$\begin{aligned}
 p_z &= \sin \alpha (l_0 + l_1 \cos \beta + l_2 \cos(\beta + \gamma)) \\
 a_x &= \sin \alpha \\
 o_z &= \cos(\beta + \gamma) \\
 p_y &= a_x (l_0 + l_1 \cos \beta + l_2 o_z)
 \end{aligned} \tag{3.41}$$

The equations shown in Equations (3.40) and (3.41) can be rearranged to make $\cos \beta$ and $\sin \beta$ the subjects in Equation (3.42).

$$\begin{aligned}
 \sin \beta &= \frac{p_z - l_2 n_z}{l_1} \\
 \cos \beta &= \frac{\frac{p_y}{a_x} - l_0 - l_2 o_z}{l_1}
 \end{aligned} \tag{3.42}$$

These equations can be combined and manipulated to find β , as shown in Equations (3.43) to (3.45).

$$\frac{\sin \beta}{\cos \beta} = \frac{(p_z - l_2 n_z) l_1}{l_1 \left(\frac{p_y}{a_x} - l_0 - l_2 a_z \right) a_x} = \tan \beta \tag{3.43}$$

$$\tan \beta = \frac{p_z a_x - l_2 a_x}{p_y - l_0 a_x - l_2 o_z a_x} \quad (3.44)$$

$$\beta = \arctan2(p_z a_x - l_2 n_z a_x, p_y - l_0 a_x - l_2 o_z a_x) \quad (3.45)$$

The relationships seen in Equation (3.26) are once again substituted into Equation (3.45) to find β in terms of roll, yaw and pitch.

$$\begin{aligned} \beta \\ = \arctan2 \left(\begin{array}{l} l_2 \sin \theta_y (\cos \theta_r \sin \theta_p - \sin \theta_r \sin \theta_y \cos \theta_p) - p_z \sin \theta_y, \\ p_y + l_0 \sin \theta_y + l_2 \sin \theta_y (\cos \theta_r \cos \theta_p + \sin \theta_r \sin \theta_y \sin \theta_p) \end{array} \right) \end{aligned} \quad (3.46)$$

Finally, having found β , γ can be calculated using n_z and o_z .

$$n_z = \sin(\beta + \gamma)$$

$$o_z = \cos(\beta + \gamma) \quad (3.47)$$

$$\therefore \frac{n_z}{o_z} = \frac{\sin(\beta + \gamma)}{\cos(\beta + \gamma)} = \tan(\beta + \gamma)$$

Rearranging to make γ the subject gives,

$$\gamma = \arctan2(n_z, o_z) - \beta \quad (3.48)$$

where n_z and o_z can be substituted out using the relationships from Equation (3.26) to find γ in terms of roll, yaw and pitch.

$$\gamma = \arctan2(\cos \theta_r \sin \theta_p - \sin \theta_r \sin \theta_y \cos \theta_p, \cos \theta_r \cos \theta_p + \sin \theta_r \sin \theta_y \sin \theta_p) - \beta \quad (3.49)$$

With all of the joint angles calculated, δ and ε can be calculated using the relationships in Equation (3.9). In order to remove dependence on pose, a fourth link can be added with 0 m length with axes of freedom and no limitations on freedom. The equations have a high degree of dependency on one another and become unsolvable.

3.4 Simultaneous Geometric Equations for Inverse Kinematics

This method for calculating the robotic manipulator arm inverse kinematics uses the geometric parameters of the arm to form a set of simultaneous equations. These can be solved to determine the joint positions given the manipulator base point and end effector positions. Figure 3-12 illustrates the manipulator arm configuration and defines the parameters to be used in the simultaneous equations.

In this diagram l_0 , l_1 , and l_2 are the length of each of the links respectively, xy_0 , xy_1 , and xy_2 are the xy-component of each of the links respectively and z_1 and z_2 are the z-components of the second and third links respectively. P_f is the position of the end effector on the third link, and has two components, P_{fxy} and P_{fz} which are the x-y and z coordinates of the end effector respectively. P_0 is the position of the first joint and the origin of the arm. The parameters of the manipulator arm are outlined in Table 3-3:

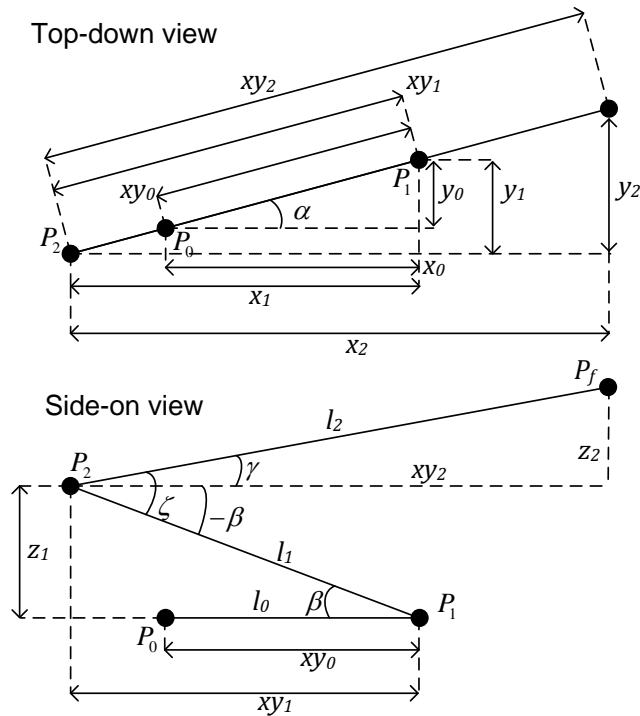


Figure 3-12: Schematic of the variables and dimensions that make up the simple model of the Digital Vanguard ROV manipulator arm.

Table 3-3 Parameters of links in a 3-DoF manipulator arm.

Link Label	Joint	Joint Angle	Link Length
l_1	$P_0 - P_1$	α	0.090 m
l_3	$P_1 - P_2$	β	0.332 m
l_3	$P_2 - P_{ef}$	γ	0.538 m

The geometric relationships describing the manipulator arm are seen in Equations (3.50) to (3.55).

$$xy_1^2 + z_1^2 = l_1^2 \quad (3.50)$$

$$xy_2^2 + z_2^2 = l_2^2 \quad (3.51)$$

$$xy_3^2 + z_3^2 = l_3^2 \quad (3.52)$$

$$P_{xy1} = xy_1 \quad (3.53)$$

$$P_{z1} = 0$$

$$P_{xy2} = xy_1 + xy_2 \quad (3.54)$$

$$P_{z2} = z_2$$

$$P_{fxy} = xy_1 + xy_2 + xy_3 \quad (3.55)$$

$$P_{fz} = z_2 + z_3$$

The unknowns in these relationships are x_2 , y_2 , x_3 and y_3 from xy_2 and xy_3 respectively and z_2 and z_3 and can be solved simultaneously to form Equations (3.56) to (3.62).

$$xy_2 = -\frac{AB \pm P_{fz}\sqrt{-CD} + l_1^3 - P_{fxy}^3}{E} \quad (3.56)$$

where,

$$\begin{aligned}
A &= l_1(l_2^2 - l_3^2 + P_{fz}^2 + 3P_{fxy}^2) - P_{fxy} \\
B &= 3l_1^2 + l_2^2 - l_3^2 + P_{fz}^2 \\
C &= (l_1^2 - 2l_1P_{fxy} - l_2^2 + 2l_2l_3 - l_3^2 + P_{fz}^2 + P_{fxy}^2) \\
D &= (l_1^2 - 2l_1P_{fxy} - l_2^2 - 2l_2l_3 - l_3^2 + P_{fz}^2 + P_{fxy}^2) \\
E &= 2(l_1^2 - 2l_1P_{fxy} + P_{fz}^2 + P_{fxy}^2)
\end{aligned}
\tag{3.57}$$

$$xy_3 = \mp \frac{A \mp P_{fxy}B + P_{efz}\sqrt{C} + l_1^3 \mp P_{fxy}^3}{F}
\tag{3.58}$$

where,

$$\begin{aligned}
A &= l_1(l_3^2 - l_2^2 \pm P_{efz}^2 \pm 3P_{fxy}^2) \\
B &= 3l_1^2 + l_2^2 - l_3^2 + P_{fz}^2 \\
C &= -DE \\
D &= l_1^2 + 2(l_2l_3 - l_1P_{fxy}) - l_2^2 - l_3^2 + P_{fz}^2 + P_{fxy}^2 \\
E &= l_1^2 + 2(l_2l_3 + l_1P_{fxy}) - l_2^2 - l_3^2 + P_{fz}^2 + P_{fxy}^2 \\
F &= 2(l_1^2 - 2l_1P_{fxy} + P_{fz}^2 + P_{fxy}^2)
\end{aligned}
\tag{3.59}$$

$$z_2 = \frac{P_{fz}}{2} + \frac{P_{fz} \frac{l_2^3 - l_3^3 \pm l_1\sqrt{A} \mp P_{fxy}\sqrt{A}}{2}}{D}
\tag{3.60}$$

and,

$$z_3 = \frac{P_{fz}}{2} - \frac{P_{fz} \frac{l_2^3 - l_3^3 \pm l_1 \sqrt{A} \mp P_{fxy} \sqrt{A}}{2}}{D} \quad (3.61)$$

where,

$$\left. \begin{aligned} A &= -BC \\ B &= l_1^2 + 2l_1 P_{fxy} - l_2^2 + 2l_2 l_3 - l_3^2 + P_{fz}^2 + P_{fxy}^2 \\ C &= l_1^2 + 2l_1 P_{fxy} - l_2^2 + 2l_2 l_3 - l_3^2 + P_{fz}^2 + P_{fxy}^2 \\ D &= l_1^2 + 2l_1 P_{fxy} + P_{fz}^2 + P_{fxy}^2 \end{aligned} \right\} \quad (3.62)$$

These equations give the x-y and z positions of each joint which can then be used with Equation (3.63) to calculate the angle of each.

$$\alpha = \arctan2(P_z, P_{xy}) \quad (3.63)$$

3.5 Summary of Kinematic Modelling

The reason behind the development of the forward and inverse kinematic models presented in this chapter is to have accessible a method of mapping the environment in terms of the joint angle ranges where collisions occur between the arm and obstacles in the Euclidean domain, with the dimensions of the map corresponding to each of the joint angles. The procedure involved in this method is to determine an array of points along all of the exposed surfaces and edges of the object/obstacle and then run each point on the obstacle through the arm kinematics, using the kinematics for each of P_1 , P_2 and P_f at intervals along these arm links of 0 m to l_1 m, l_2 m and l_3 m respectively. This means that collisions at any point along the arm are recorded, and not only the end point for each link. By inspection of these three methods it is clear that the inverse solution for the Denavit-Hartenberg matrices

requires a lot more calculations than the other two suggested methods, therefore this method shall be discounted at this point.

The third line of development for this type of mapping used basic geometry. In the robotic manipulator arm, each link is moved by rotation in only one axis. This means that, given a fixed starting point for the link, any collisions with that link must occur within a circle on the same plane as the link, with a radius of the link length, where the centre of the circle is at the pivot point of the link. If there is a collision then the cosine of the joint angle that causes said collision can be found using the dot product of two vectors divided by the multiplication of the magnitudes of the two vectors, where the two vectors describe the link in its zero or initial position and the line from the pivot point to the collision point.

If the object is larger than a point mass, then there will be a continuous range of angles that would cause a collision with the object, providing that the object is a regular shape. This allows for a simplification to the algorithm as only the maximum and minimum collision angles for that object need to be found. For a regular shape any angle in between these two limits must cause a collision between the arm link and the object.

Having looked at a way of finding the collision angles for one link, it becomes very easy to investigate collision angles for a series of connected links. By starting with the lowest link in the chain and finding all of the collision angles, the next link can be investigated. The starting position for this link is the end position of the previous one so any starting position that occurs in the range of angles for the previous link that causes a collision with the object does not need to be investigated. Only starting positions that are available from accessible angles of the previous link must be investigated.

This method has a significant drawback however. For a small number of small objects with a low number of defined inspection points, the algorithm will execute quite quickly as the number of calculations is relatively small, but as soon as the size and number of objects and the number of inspection points increases, the algorithm run time will also increase. This problem is compounded further by increasing the number of points along the robotic arm used for inspection for collisions. Depending on how many inspection points on both the obstacle and robotic arm are required to

make a usable collision map of the object, especially given the complexity of the real world environments to be explored, this method may be unviable as it may be too slow to be used to navigate in real time. This process can then be extended for any further links, and the number of iterations increases by an order of magnitude for each extra link. This increase in calculations by an order of magnitude for each extra link can be seen as a potential problem for higher number degrees-of-freedom in the arm as the time taken for the mapping calculation will increase dramatically, potentially making the method unfeasible.

As described in section 3.2 of this chapter, there is a third method considered which is very similar to the second, but which very quickly finds all solutions to a single end effector position. If this is carried out for a series of inspection points along the arm for all of the inputted data points from the simulated obstacles then the algorithm will run very quickly due to its low computational load.

The inverse trigonometric method and the simultaneous geometric equation method will both be implemented in Chapter 6 in order to test the feasibility of use of each of them as part of the mapping algorithm. The following chapter deals with modelling of the dynamics of a 3-DoF robotic manipulator arm to provide a set of constraints with which to develop the guidance method and also provide a test bed for validation of the complete technique.

4 DYNAMIC MODELLING OF A ROBOTIC MANIPULATOR

In the previous chapter the forward and inverse kinematics of a 3-DoF manipulator arm have been derived for use in the work carried out in Chapter 6 in the development of a guidance method for the manipulator arm. The present chapter deals with the dynamic modelling of a generic 3-DoF manipulator arm with the same degrees-of-freedom as previously specified. Applying values to the numerical parameters in the dynamic model will allow for qualitative validation of the developed model, which can then be used in future chapters to design and tune a controller providing sufficient performance for the arm. This chapter deals with the block highlighted in green in Figure 1-4, which is displayed again here with all of the other processes greyed out.

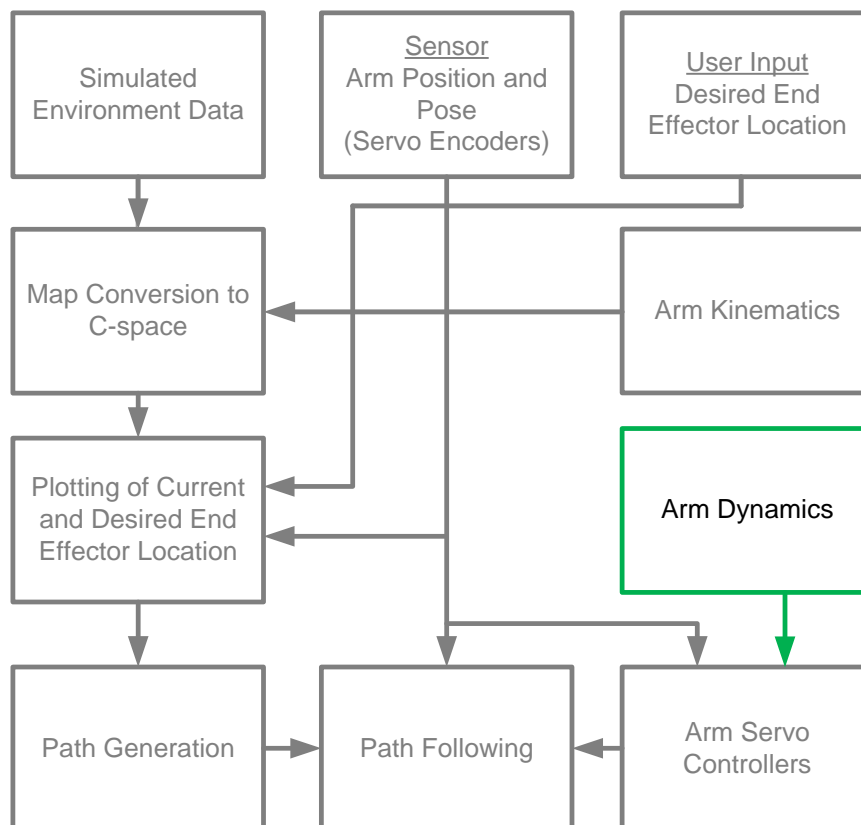


Figure 4-1 Manipulator arm dynamics (green) in relation to the overall guidance method.

Having investigated literature regarding the dynamic modelling of robotic manipulator arms in Chapter 2, some of the information contained there can be used for the development and validation of a dynamic model, including a simple servo model, for use as the test bed for the control and guidance parts of the thesis.

The dynamic model of the manipulator arm will be derived using Newtonian mechanics since (Turney, et al.) showed that the Lagrangian derivation has a higher computational overhead, the results of which are presented in Table 2-1, and Table 2-2 for a 3-DoF problem.

4.1 Manipulator Arm Parameters

It is important to recognise at this point a change in the joint angle usage, and therefore its nomenclature. In the kinematic modelling of the manipulator arm, the joint angles in use referred to the grey shaded angles in Figure 4-2, β and γ , which corresponded to the angles starting at 0^c when the arm was fully retracted and the positive direction for each joint was as follows. The angle α had exactly the same starting location and direction as in the following figure, γ and η have the same direction, but $\beta = -\sigma$. The reason for the change in angles to α, σ, η rather than the original α, β, γ is that for the dynamic modelling, maintaining all of the angles with the same starting orientation and direction ensures that all of the moments about joints operate in the same directions and there is no need to resolve directions. The difference is that the starting pose for the arm will be different. In the $[\alpha, \beta, \gamma]$ coordinate system, the starting pose would be $[0,0,0]$. In the $[\alpha, \sigma, \eta]$ coordinate system, the starting pose would be $[0, \pi, 0]$.

The manipulator arm shown in Figure 4-2 has 3 degrees-of-freedom, with link 1 rotating about the Z-axis at P_0 , link 2 rotating about an axis on a horizontal plane at P_1 and link 3 rotating about an axis on a horizontal plane at P_2 . The lengths l_1, l_2 and l_3 are the lengths of links 1, 2 and 3 respectively. The mass of each link is represented by a point mass at the centre of mass of the link. The masses m_1, m_2 and m_3 are the masses of each link. The position of each of these centres of mass are described by the distance from the lower end of the respective link, therefore c_1, c_2 and c_3 are the distances from the lower joint of links 1, 2 and 3 respectively to their centre of mass.

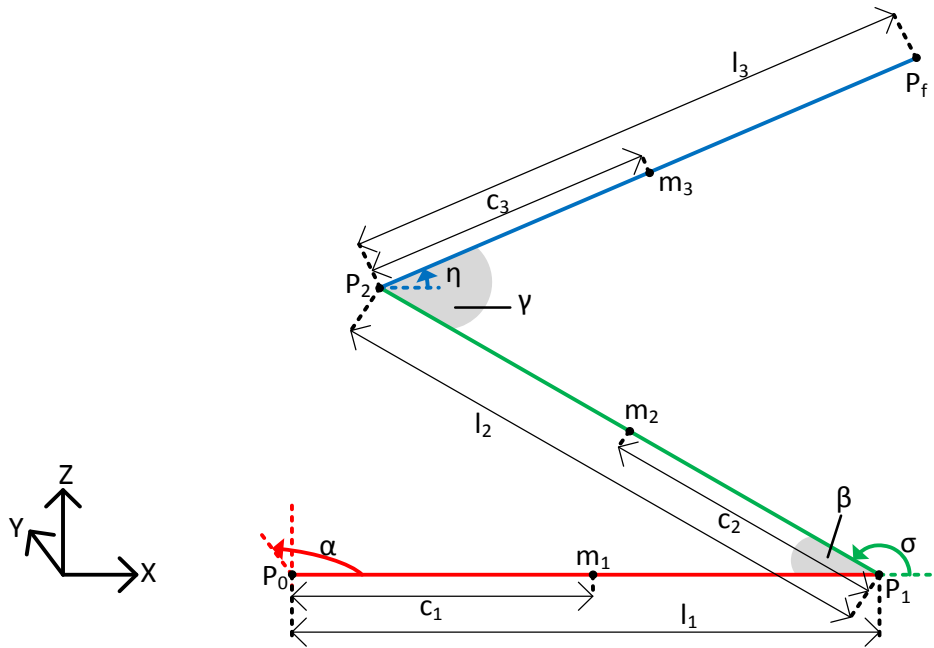


Figure 4-2 Schematic of a 3-DoF manipulator arm showing the relevant parameters for the development of a dynamic model.

The angles α , σ and η are the angle of each joint from the horizontal. The positive direction of each axis of rotation is anti-clockwise. Based on the above diagram this means that α will rotate into the page, and σ and η will rotate anti-clockwise as positive. For the purposes of the model, 0 radians in each case is when the joint angle is pointing horizontally to the right of the page, therefore in the case of the diagram, $\alpha = 0^\circ$, $\sigma \approx 140^\circ$, and $\eta \approx 30^\circ$.

4.2 Dynamic Equation Formulation

Having laid out the parameters of a generic 3-DoF manipulator arm, the equations governing its dynamic properties can be explored. By using Newtonian mechanics and d'Alembert's principle, the dynamics can be developed. For each link the relationship shown in Equation (4.1) is the case.

$$\Sigma T = T_{in} - \Sigma M_m - \Sigma M_f - \Sigma M_i - \Sigma M_c \quad (4.1)$$

Where ΣT is the total torque about a joint, T_{in} is the input torque to the joint, ΣM_m is the total moment about the joint caused by the weight of the joint and any joints further up the arm, ΣM_f is the total moment about the joint caused by static and friction, ΣM_i is the total moment caused by moments of inertia of the link and any links further up the arm about the joint and ΣM_c is the total moment about the joint caused by the reaction to centripetal forces generated by circular motion in the joint, or any caused by circular motion in joints both below and above the joint.

4.2.1 First Link

In the first link, the rotation occurs about the Z-axis, so with the exception of any shearing forces causing friction in the joint, which will be assumed to be negligible, the weight of each link will have no effect on the total torque about the joint, therefore ΣM_m is zero. Equally, any reaction forces to the centripetal forces generated due to circular motion of any of the links acts in a direction parallel to the length of the first link in the horizontal plane, therefore there is no moment generated by them about the first joint, hence ΣM_c is also zero. This reduces Equation (4.1) to that of Equation (4.2).

$$\Sigma T = T_{in} - \Sigma M_f - \Sigma M_i \quad (4.2)$$

A non-conservative torque which is present in the system is that which is caused by friction. This model will deal with the effects of two forms of friction, static and kinetic. Kinetic friction can be modelled very easily since torque produced by kinetic friction is calculated by $\tau_k = c_k \dot{\theta}$, where c_k is the coefficient of kinetic friction. Static kinetic friction is more complex to model since it only exists when the angular velocity is small or zero. For this to be the case the static friction term needs to reduce to zero as the angular velocity increases. This is done using the expression given in Equation (4.3).

$$\tau_s = \text{sign}(\dot{\theta})e^{-|\dot{\theta}|}c_s \quad (4.3)$$

Where, c_s is the coefficient of static friction. This forms the expression for the total torque about the joint due to friction shown in Equation (4.4).

$$\Sigma\tau_f = c_k\dot{\theta} + \text{sign}(\dot{\theta})e^{-|\dot{\theta}|}c_s \quad (4.4)$$

The coefficient of kinetic friction can be modelled using this expression, which displays a shape which matches that of a Coulomb and Viscous friction model. Using values taken from (Meriam & Kraige, 2012) for steel against steel, the static friction coefficient is 0.6 and the kinetic friction coefficient is 0.4. The reason behind these choices of coefficient of friction is because the friction terms in the gearboxes and servo motors for each joint have not been modelled, and this friction term allows for their effects on the system as well. In this case the moment about the joint caused by friction is given in Equation (4.5).

$$\Sigma M_f = 0.4\dot{\alpha} + 0.6\text{sign}(\dot{\alpha})e^{-|\dot{\alpha}|} \quad (4.5)$$

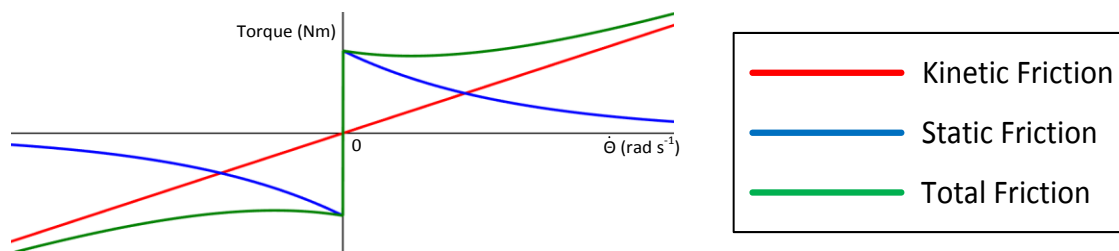


Figure 4-3 Effect of angular velocity on static and kinetic friction.

Figure 4-3 show how the expression for static and kinetic friction behaves with varying angular velocity. The red line is the kinetic friction, the blue line is the static friction and the green line the summation of the two. It is clear from the figure that

static friction has the most effect early at low angular velocities, which requires an initial amount of energy to overcome. As angular velocity increases the static friction had a depreciating effect and kinetic friction becomes the principal frictional effect acting on the system. The friction model is symmetrical in that the friction has the exact same effect at negative velocities, only in the reverse direction.

Since moment of inertia, $I = mr^2$, and torque, $\tau = I\ddot{\theta}$, and in the case of joint 1 the radius is the horizontal distance of each mass from the same joint the moment of inertia of each of the links about joint 1 can be calculated using Equations (4.6) to (4.8).

$${}^1_1\tau = m_1\ddot{a}r_1^2 = m_1\ddot{a}c_1^2 \quad (4.6)$$

$${}^2_1\tau = m_2\ddot{a}r_2^2 = m_2\ddot{a}(l_1 + c_2 \cos \sigma)^2 \quad (4.7)$$

$${}^3_1\tau = m_3\ddot{a}r_3^2 = m_3\ddot{a}(l_1 + l_2 \cos \sigma + c_3 \cos \eta)^2 \quad (4.8)$$

Therefore the total moment caused by the moments of inertia is calculated using Equation (4.9).

$$\Sigma M_i = m_1\ddot{a}c_1^2 + m_2\ddot{a}(l_1 + c_2 \cos \sigma)^2 + m_3\ddot{a}(l_1 + l_2 \cos \sigma + c_3 \cos \eta)^2 \quad (4.9)$$

Using this relationship the total torque about joint 1 can be by Equation (4.10).

$$\begin{aligned} \Sigma T_1 = T_{1in} - (0.4\dot{a} + 0.6\text{sign}(\dot{a})e^{-|\dot{a}|}) \\ -m_1\ddot{a}c_1^2 - m_2\ddot{a}(l_1 + c_2 \cos \sigma)^2 - m_3\ddot{a}(l_1 + l_2 \cos \sigma + c_3 \cos \eta)^2 \end{aligned} \quad (4.10)$$

4.2.2 Second Link

In the case of the second link, the weights of the 2nd and 3rd links have an effect on the moments about joint 2. The moment about joint 2 caused by the weight of link 2 can be calculated using the schematic shown in Figure 4-4.

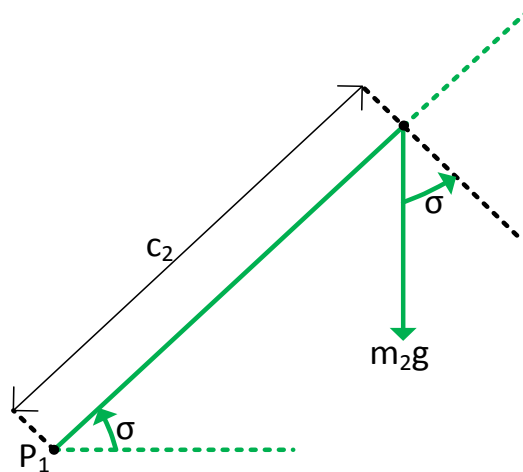


Figure 4-4 Schematic of the weight of link 2 acting on link 2.

In this case the moment acting about P_1 due to the weight of the 2nd link is given by Equation (4.11).

$$\frac{2}{2}M = m_2 c_2 g \cos \sigma \quad (4.11)$$

For the moment acting about P_1 due to the weight of the 2nd link, a similar process can take place. In this case the effect of the weight of link 3 is calculated using θ and l_θ . To do this Equations (4.12) and (4.13) are used.

$$\theta = \text{atan} \left(\frac{l_2 \sin \sigma + c_3 \sin \eta}{l_2 \cos \sigma + c_3 \cos \eta} \right) \quad (4.12)$$

$$l_\theta = \sqrt{(l_2 \sin \sigma + c_3 \sin \eta)^2 + (l_2 \cos \sigma + c_3 \cos \eta)^2} \quad (4.13)$$

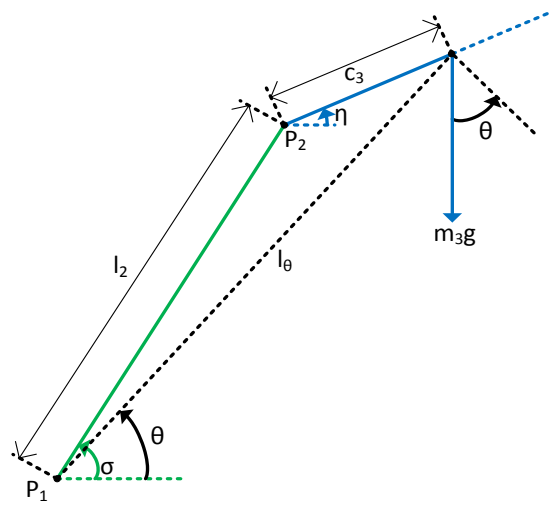


Figure 4-5 Schematic of the weight of link 3 acting on link 2.

Equation (4.13), which calculates the value for l_θ can be simplified using trigonometric identities to that of Equation (4.14).

$$l_\theta = \sqrt{l_2^2 + c_3^2 + 2l_2c_3 \cos(\sigma - \eta)} \quad (4.14)$$

By combining Equations (4.13) and (4.14), the moment caused by the weight of link 3 about the second joint can be calculated using Equation (4.15).

$${}^3_2M = m_3g \cos\left(\text{atan}\left(\frac{l_2 \sin \sigma + c_3 \sin \eta}{l_2 \cos \sigma + c_3 \cos \eta}\right)\right) \sqrt{l_2^2 + c_3^2 + 2l_2c_3 \cos(\sigma - \eta)} \quad (4.15)$$

therefore,

$$\begin{aligned}
& \Sigma M_m \\
& = m_2 c_2 g \cos \sigma \\
& + m_3 g \cos \left(\text{atan} \left(\frac{l_2 \sin \sigma + c_3 \sin \eta}{l_2 \cos \sigma + c_3 \cos \eta} \right) \right) \sqrt{l_2^2 + c_3^2 + 2l_2 c_3 \cos(\sigma - \eta)} \quad (4.16)
\end{aligned}$$

Kinetic friction about the 2nd joint occurs in exactly the same way as that of the first, therefore,

$$\Sigma M_f = 0.4\dot{\sigma} + 0.6\text{sign}(\dot{\sigma})e^{-|\dot{\sigma}|} \quad (4.17)$$

Moments of inertia about the 2nd joint can be calculated similarly to that of the first. Equation (4.18) calculates the moment of inertia caused by the second link about the 2nd joint.

$${}^2_2\tau = m_2 \ddot{\sigma} c_2^2 \quad (4.18)$$

For the moment of inertia of the 3rd link acting about the 2nd joint, l_θ is used as the radius, hence,

$$\begin{aligned}
{}^3_2\tau & = m_3 \ddot{\sigma} \sqrt{l_2^2 + c_3^2 + 2l_2 c_3 \cos(\sigma - \eta)}^2 \\
& = m_3 \ddot{\sigma} (l_2^2 + c_3^2 + 2l_2 c_3 \cos(\sigma - \eta)) \quad (4.19)
\end{aligned}$$

Therefore the total torque generated by moments of inertia about joint 2 can be described by Equation (4.20).

$$\Sigma M_i = m_2 \ddot{\sigma} c_2^2 + m_3 \ddot{\sigma} (l_2^2 + c_3^2 + 2l_2 c_3 \cos(\sigma - \eta)) \quad (4.20)$$

There are five reaction forces caused by circular motion which act on the 2nd link. Rotation of the 2nd link about the 2nd joint causes a centripetal force which acts axially along the link, and therefore causes no moment about joint 2. However, reactions to the centripetal forces caused by the motion of the 2nd and 3rd links about the 1st joint act through the 2nd joint, as do the reactions to the centripetal forces caused by the rotation of the 3rd link about the 2nd and 3rd joints.

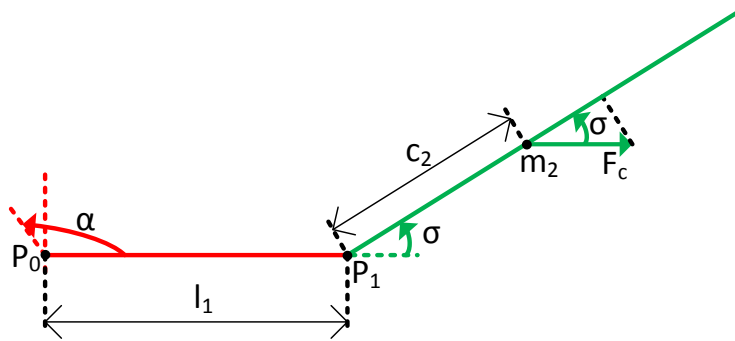


Figure 4-6 Schematic of the centripetal force which acts on the 2nd link due to its rotation about the 1st joint.

For the moment generated about the 2nd joint by the reaction force caused by the rotation of the 2nd link about the first joint, the Figure 4-6 assists with the derivation of the equation. Since centripetal force $F = mr\omega^2$, in this case, the radius would be the horizontal distance between joint 1 and the position of m_2 . The component of this force that causes a moment about joint 2 can be calculated by trigonometry, forming Equation (4.21).

$$\frac{1}{2}M_{2c} = c_2 m_2 (l_1 + c_2 \cos \sigma) \dot{\sigma}^2 \sin \sigma \quad (4.21)$$

In this case the reaction to the centripetal force caused by the rotation of the 3rd link about the 1st joint acts at the end of link 2 in the horizontal direction and so can be calculated with trigonometry.

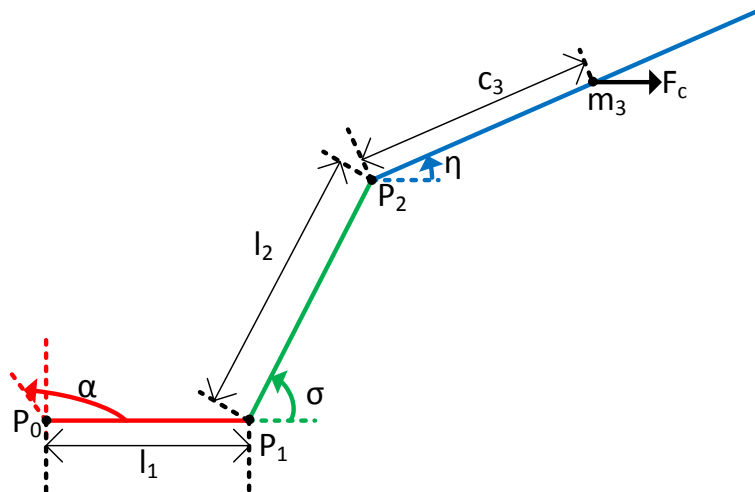


Figure 4-7 Schematic of the centripetal force which acts on the 2nd link due to the rotation of the 3rd link about the 1st joint.

$$\frac{1}{3}M_{2c} = l_2 m_3 (l_1 + l_2 \cos \sigma + c_3 \cos \eta) \dot{\alpha}^2 \sin \sigma \quad (4.22)$$

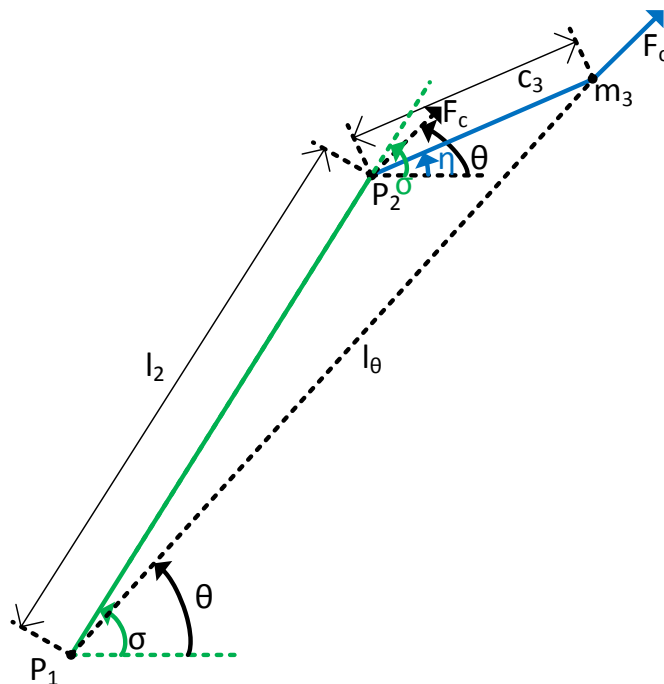


Figure 4-8 Schematic of the centripetal force caused by the rotation of the 3rd link about the 2nd joint.

In this case the centripetal force can be calculated using Equation (4.23).

$$F_c = m_3 l_3 \dot{\theta}^2 = m_3 \dot{\sigma}^2 \sqrt{l_2^2 + c_3^2 + 2l_2 c_3 \cos(\sigma - \eta)} \quad (4.23)$$

The moment about joint 2 can be calculated by finding the component of this force perpendicular to link 2 and multiplying it by l_2 . This forms Equation (4.24).

$$\begin{aligned} {}^2_3M_{2c} &= l_2 \sin(\sigma - \theta) m_3 \dot{\sigma}^2 \sqrt{l_2^2 + c_3^2 + 2l_2 c_3 \cos(\sigma - \eta)} \\ &= l_2 \sin\left(\sigma - \arctan\left(\frac{l_2 \sin \sigma + c_3 \sin \eta}{l_2 \cos \sigma + c_3 \cos \eta}\right)\right) m_3 \dot{\sigma}^2 \sqrt{l_2^2 + c_3^2 + 2l_2 c_3 \cos(\sigma - \eta)} \end{aligned} \quad (4.24)$$

Finally, the moment about joint 2 caused by the centripetal reaction force generated by the rotation of link 3 about joint 3 can be calculated by using Figure 4-9 as a reference.

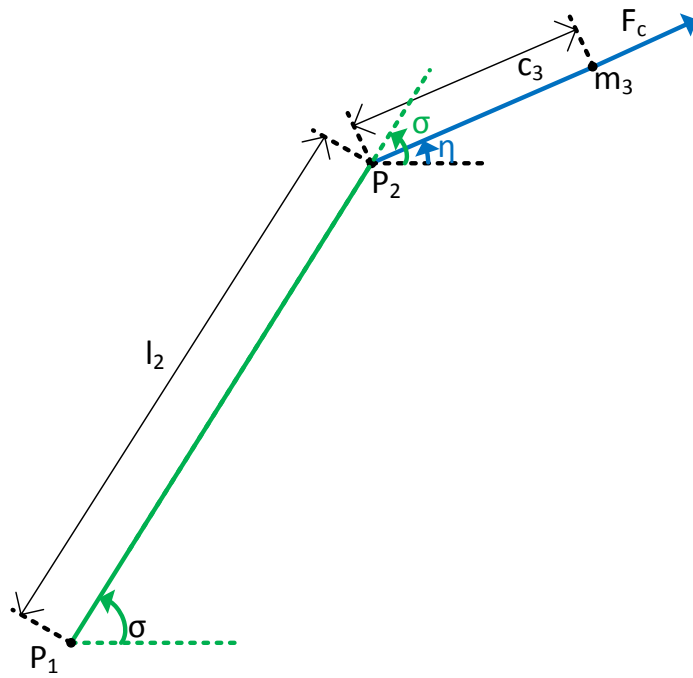


Figure 4-9 Schematic of the centripetal force caused by the rotation of link 3 about joint 3.

In this case the moment of centripetal force caused by the rotation of link 3 about joint three which acts about joint 2 is calculated by Equation (4.25).

$${}^3_3M_{2c} = l_2 m_3 c_3 \dot{\eta}^2 \sin(\sigma - \eta) \quad (4.25)$$

This means that the sum of moments caused by centripetal forces calculated by Equation (4.26).

$$\begin{aligned}
& \Sigma M_c \\
& = l_2 m_3 (l_1 + l_2 \cos \sigma + c_3 \cos \eta) \dot{\alpha}^2 \sin \sigma + m_2 c_2 (l_1 + c_2 \cos \sigma) \dot{\alpha}^2 \sin \sigma \\
& + \left(l_2 \sin \left(\sigma \right. \right. \\
& \left. \left. - \arctan \frac{l_2 \sin \sigma + c_3 \sin \eta}{l_2 \cos \sigma + c_3 \cos \eta} \right) m_3 \dot{\sigma}^2 \sqrt{l_2^2 + c_3^2 + 2l_2 c_3 \cos(\sigma - \eta)} \right) \\
& + l_2 m_3 c_3 \dot{\eta}^2 \sin(\sigma - \eta)
\end{aligned} \tag{4.26}$$

When the above equations are summed to calculate the total torque about joint 2, Equation (4.27) is formed:

$$\begin{aligned}
& \Sigma T_2 \\
& = T_{2in} - c_2 m_2 g \cos \sigma \\
& - \left(m_3 g \cos \left(\arctan \frac{l_2 \sin \sigma + c_3 \sin \eta}{l_2 \cos \sigma + c_3 \cos \eta} \right) \sqrt{l_2^2 + c_3^2 + 2l_2 c_3 \cos(\sigma - \eta)} \right) \\
& - (0.4 \dot{\sigma} + 0.6 \text{sign}(\dot{\sigma}) e^{-|\dot{\sigma}|}) - m_2 \ddot{\sigma} c_2^2 - m_3 \ddot{\sigma} (l_2^2 + c_3^2 + 2l_2 c_3 \cos(\sigma - \eta)) \\
& - l_2 m_3 (l_1 + l_2 \cos \sigma + c_3 \cos \eta) \dot{\alpha}^2 \sin \sigma - m_2 c_2 (l_1 + c_2 \cos \sigma) \dot{\alpha}^2 \sin \sigma \\
& - \left(l_2 \sin \left(\sigma \right. \right. \\
& \left. \left. - \arctan \frac{l_2 \sin \sigma + c_3 \sin \eta}{l_2 \cos \sigma + c_3 \cos \eta} \right) m_3 \dot{\sigma}^2 \sqrt{l_2^2 + c_3^2 + 2l_2 c_3 \cos(\sigma - \eta)} \right) \\
& - l_2 m_3 c_3 \dot{\eta}^2 \sin(\sigma - \eta)
\end{aligned} \tag{4.27}$$

4.2.3 Third Link

In the case of the third link, the weight of the third link has an effect on the total moment about the joint.

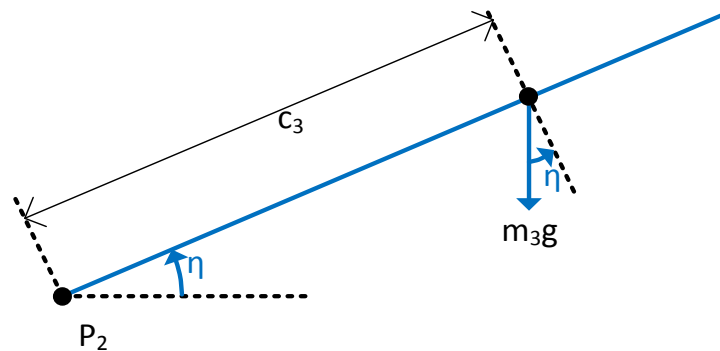


Figure 4-10 Schematic of the moment about joint 3 caused by the weight of link 3.

In this case.

$$M_m = m_3 c_3 g \cos \eta \quad (4.28)$$

The friction about joint 3 also has an impact, therefore,

$$M_f = 0.4\dot{\eta} + 0.6\text{sign}(\dot{\eta})e^{-|\dot{\eta}|} \quad (4.29)$$

In the case of the third link its inertia has an impact:

$$M_i = m_3 \ddot{\eta} c_3^2 \quad (4.30)$$

Centripetal forces caused by rotation have an effect on the moments about joint 3. Rotation of link 3 about joint 3 causes a centripetal reaction force which acts axially along the length of link 3, and therefore does not impact the moments about joint 3. However rotation of link 3 about joints 1 and 2 do have an effect. For the rotation of link 3 about joint 1 Figure 4-11 is the case.

The centripetal force caused by the acceleration about joint 1 can be calculated using Equation (4.31).

$$F_c = m_3(l_1 + l_2 + c_3 \cos \eta)\ddot{\alpha}^2 \quad (4.31)$$

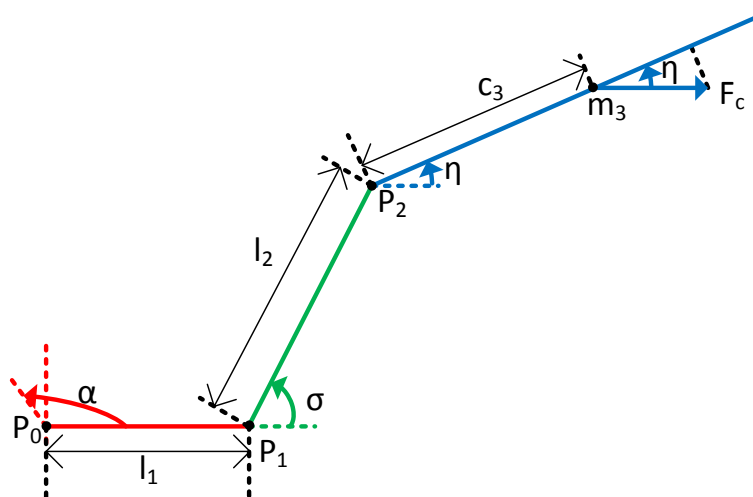


Figure 4-11 Schematic of the centripetal force caused by the rotation of link 3 about joint 1.

The component of this force which acts perpendicular to the link can be calculated using Equation (4.32).

$$\frac{1}{3}M_{3c} = m_3(l_1 + l_2 + c_3 \cos \eta)\ddot{\alpha}^2 \sin \eta \quad (4.32)$$

As before $l_\theta = \sqrt{l_2^2 + c_3^2 + 2l_2c_3 \cos(\sigma - \eta)}$ and $\theta = \text{atan}\left(\frac{l_2 \sin \sigma + c_3 \sin \eta}{l_2 \cos \sigma + c_3 \cos \eta}\right)$. Since this is the case the centripetal force can be calculated using Equation (4.33).

$$F_c = m_3 \dot{\sigma}^2 \sqrt{l_2^2 + c_3^2 + 2l_2c_3 \cos(\sigma - \eta)} \quad (4.33)$$

For the rotation of link 3 about joint 2, Figure 4-12 shows the direction of the centripetal force acting on the link.

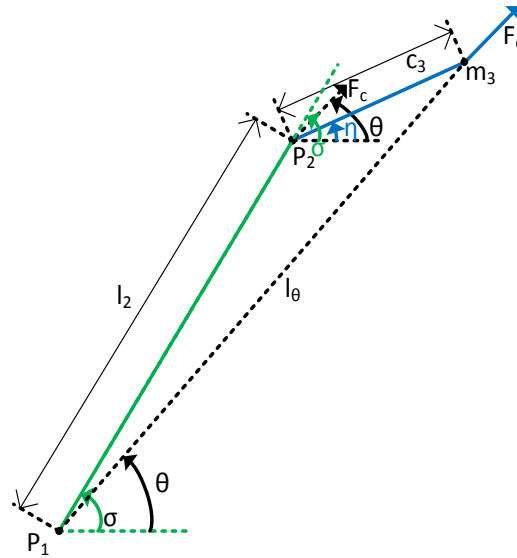


Figure 4-12 Schematic of the centripetal force caused by the rotation of the 3rd link about the 2nd joint.

The component of this force that acts perpendicular to link 3 will have an impact on the total moment about joint 3, therefore,

$$\begin{aligned} \frac{2}{3}M_{3c} = m_3 \dot{\sigma}^2 \sin \left(\text{atan} \left(\frac{l_2 \sin \sigma + c_3 \sin \eta}{l_2 \cos \sigma + c_3 \cos \eta} \right) \right. \\ \left. - \eta \right) \sqrt{l_2^2 + c_3^2 + 2l_2c_3 \cos(\sigma - \eta)} \end{aligned} \quad (4.34)$$

The resultant expression formed by collecting all of the moment terms about link 3 gives the total torque about the link. This is displayed in Equation (4.35).

$$\begin{aligned}
\Sigma T_3 = & T_{3in} - m_3 c_3 g \cos \eta - (0.4\dot{\eta} + 0.6 \text{sign}(\dot{\eta}) e^{-|\dot{\eta}|}) - m_3 \ddot{\eta} c_3^2 \\
& - m_3 (l_1 + l_2 \cos \sigma + c_3 \cos \eta) \dot{\alpha}^2 \sin \eta \\
& - \left(m_3 \dot{\sigma}^2 \sin \left(\text{atan} \left(\frac{l_2 \sin \sigma + c_3 \sin \eta}{l_2 \cos \sigma + c_3 \cos \eta} \right) \right. \right. \\
& \left. \left. - \eta \right) \sqrt{l_2^2 + c_3^2 + 2l_2 c_3 \cos(\sigma - \eta)} \right)
\end{aligned} \tag{4.35}$$

4.2.4 Angular Acceleration

Having derived the expressions for (4.10), (4.27) and (4.35), the resultant torques about each joint, the angular acceleration can be derived next. Since $\tau = mr^2 \ddot{\theta}$, the angular acceleration of each joint can be calculated by $\ddot{\theta} = \frac{\tau}{mr^2}$. For joint 1, all three links have an impact on the moments of inertia, therefore the sum of all three moments of inertia about joint 1 are used.

$$\Sigma I_1 = m_1 c_1^2 + m_2 (l_1 + c_2 \cos \sigma)^2 + m_3 (l_1 + l_2 \cos \sigma + c_3 \cos \eta)^2 \tag{4.36}$$

For joint 2, the moments of inertia of the second and third about the second joint have an impact, therefore,

$$\Sigma I_2 = m_2 c_2^2 + m_3 (l_2^2 + c_3^2 + 2l_2 c_3 \cos(\sigma - \eta)) \tag{4.37}$$

For joint 3, only the moment of inertia of the third link about the third joint has an impact on the acceleration, therefore,

$$\Sigma I_3 = m_3 c_3^2 \quad (4.38)$$

With these equations developed, the entire set of mechanical equations for the dynamic model has been derived, and the model can be built.

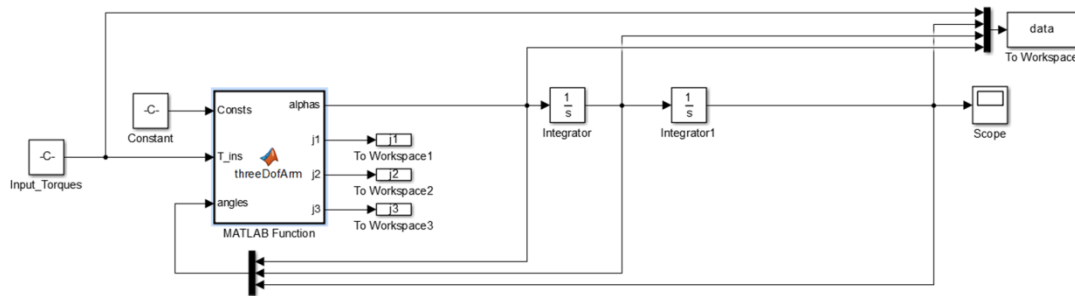


Figure 4-13 Simulink Control block diagram for the 3-DoF arm dynamic model.

In the model shown in Figure 4-13, the torques and arm parameters are inputted into the system and the dynamic equations are used to calculate the joint angular accelerations as Matlab code inside the ‘threeDofArm’ MATLAB Function, which are then fed out of the block as the signal ‘alphas’. These accelerations are integrated twice to calculate the joint angles. The joint angular positions, velocities and accelerations are fed back into the ‘threeDofArm’ MATLAB Function to carry out the next iteration of calculations.

Upon simulating scenarios using the dynamic model, the friction term in each of the joint equations displayed a high frequency oscillation at the magnitude of ± 0.6 when the system was within the range of angular velocities where static friction has more of an effect on the system than kinetic friction. Figure 4-14 and Figure 4-15 are the resulting moments about the 2nd and 3rd joints for the 1st scenario from Section 4.2.5 with the friction model described in Equation (4.4).

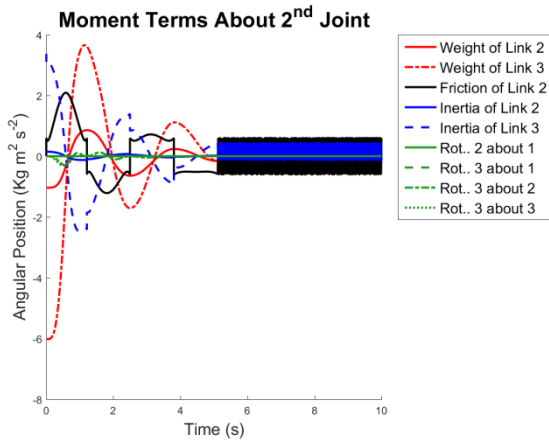


Figure 4-14 Moments about σ .

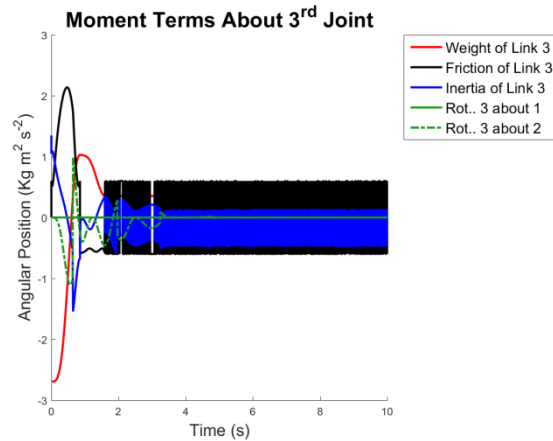


Figure 4-15 Moments about η .

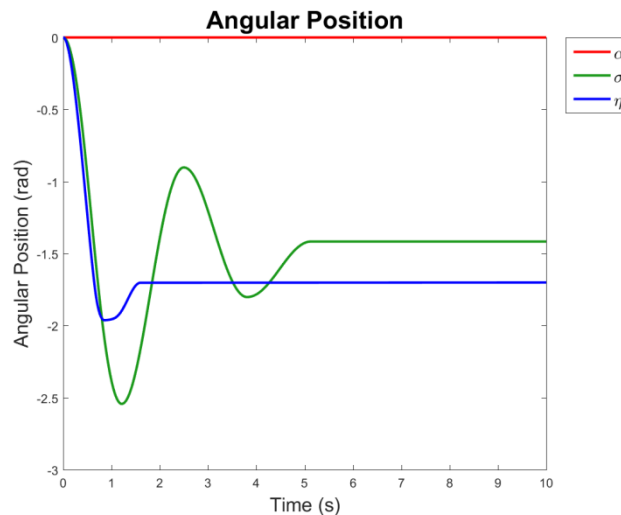


Figure 4-16 Angular position of each joint.

The time at which the high frequency moment occurs is when the system has settled to a final value, i.e. the joints are stationary. This can be observed by inspecting the time range in Figure 4-16 for the same scenario where the angular position does not change. It is clear that the moments caused by friction are inserting energy into the system, and the resultant change in velocity is affecting the acceleration of the system, therefore the inertial terms are also affected.

The reason behind this phenomenon is that the static friction term is being driven by the angular velocity of the joint, whereas static friction can only provide a force or, in

this case, torque to oppose another force or torque. A static friction torque is produced even when the system does not have velocity, since the exponent of 0 is 1. Therefore the current static friction model is required to be altered to reflect its dependence on an opposing torque. The resultant model is displayed in Equations (4.39) to (4.42). The general principle of the model is correct, Equation (4.39) will still form the basis of the model.

$$\tau_s = \text{sign}(\dot{\theta})e^{-|\dot{\theta}|}c_s \quad (4.39)$$

Since the static friction will work in the opposite direction to the resultant moment acting on the system from all other moment terms in the model, the relationship shown in (4.40) can be used.

$$\Sigma T = T_{in} - \Sigma M_m - \Sigma M_i - \Sigma M_c - \tau_k \quad (4.40)$$

Using this relationship the direction that the static friction acts can be implemented.

$$\tau_s = \text{sign}(-\Sigma T)e^{-|\dot{\theta}|}c_s \quad (4.41)$$

Finally the maximum magnitude of the static friction torque that can be applied must not exceed the value of the magnitude of the other moments about the joint.

$$\tau_s = \text{sat}_{-\Sigma T}^{\Sigma T} \text{sign}(-\Sigma T)e^{-|\dot{\theta}|}c_s \quad (4.42)$$

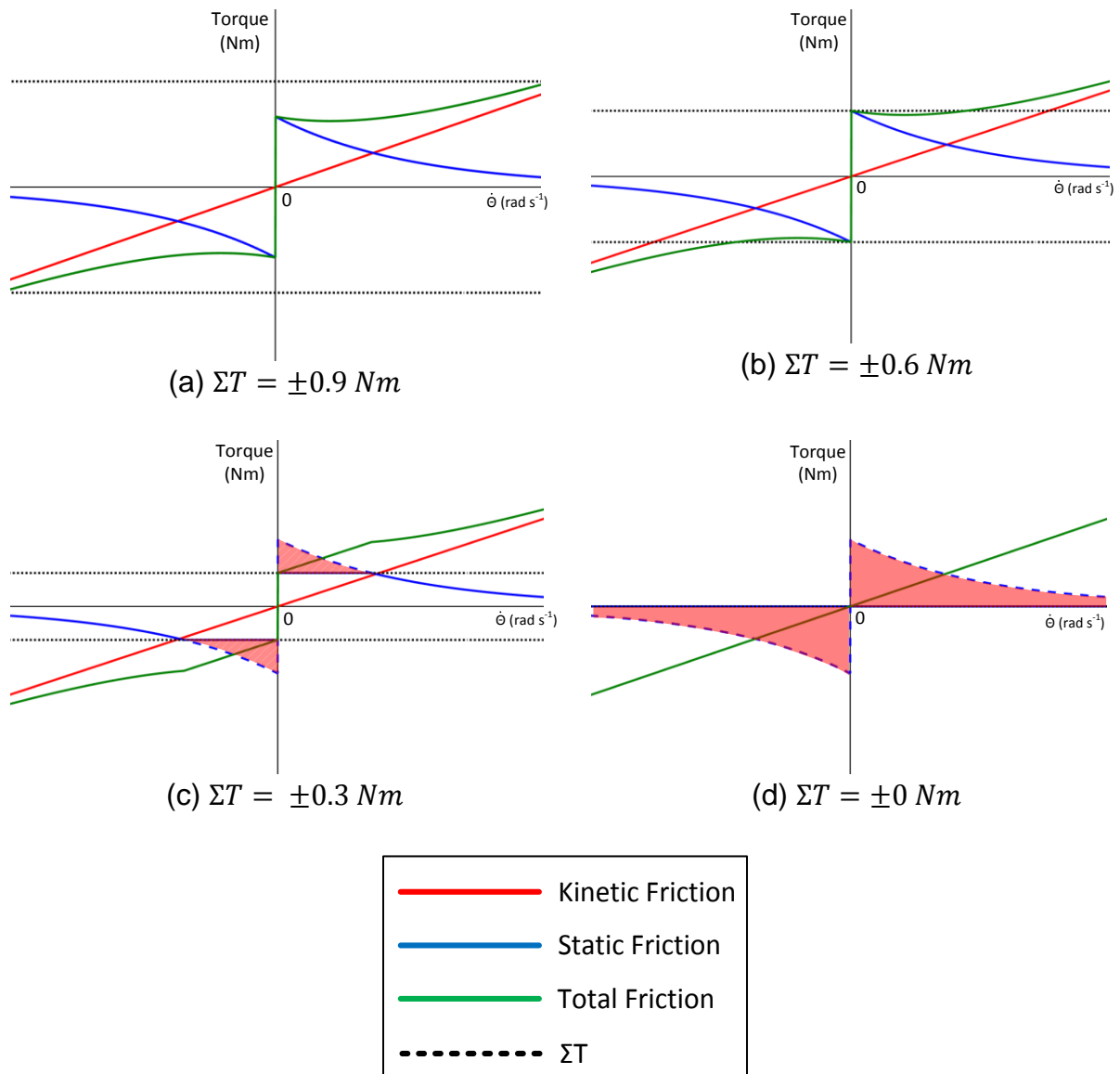


Figure 4-17 Effect of torque based saturation on friction in a joint.

Figure 4-17 shows how the static friction term is affected by different resultant torques about each joint. In figure (a) the resultant torque about the joint is larger than the maximum static friction therefore the reaction torque produced by friction is not affected. In figure (b) the resultant torque about the joint is equal to the maximum static friction so the total friction is still unaffected. In figure (c) the resultant torque is less than the maximum static friction, therefore this term is subject to a saturation which limits it to the same magnitude as the resultant torque in the case of zero angular velocity. The red shaded area indicates the region of static friction which is no longer applied. In figure (d) the resultant torque about the joint is

0 Nm, and this means that no static friction is generated as a reaction torque and only the kinetic friction has any effect on the system.

The application of the static friction term to the dynamic model is given by equation (4.43).

$$\Sigma T' = \Sigma T - \tau_s \quad (4.43)$$

4.2.5 Qualitative Validation

In this section of the chapter, a series of simulated quantitative experiments will be carried out with the dynamic model developed previously in order to validate it. For each experiment an assumption will be made about what the system is expected to do and then a comparison of the numerical results to the qualitative expectation will be carried out to validate whether the system is performing as expected. In order to carry out any validation, numerical values must be applied to all of the variables specified in the dynamic model derivation. For this case several assumptions have been made:

- Each link is a hollow steel tube of diameter 0.02 m and wall thickness of 0.002 m and the end of each link is capped with a piece of steel of the same thickness. The density of steel is also taken from Meriam and Kraige Dynamics (2012), and is stated as 7830 Kg m^{-3} .
- The servo motors that would control the arm currently have not been modelled and, as such, for the purposes of these simulations are assumed to have zero mass and inertia.
- The mass and inertia of each link is distributed uniformly through the material; therefore the centre of mass of each link occurs half way along the length of the link and occurs in the centre of the circle.
- This allows for all of the parameters to be calculated or decided upon. This robot arm will have the same parameters as the Allen Vanguard™ Digital Vanguard robotic manipulator arm, as shown in Table 4-1.

Table 4-1 Numerical parameters of the Digital Vanguard 3-DoF manipulator arm.

Parameter	Value	Parameter	Value	Parameter	Value
l_1	0.09 m	c_1	0.045 m	m_1	0.184 Kg
l_2	0.332 m	c_2	0.166m	m_2	0.637 Kg
l_3	0.538 m	c_3	0.269 m	m_3	1.021 Kg

Table 4-2 List of scenarios used for the qualitative validation of the manipulator dynamic model.

Sim No.	$\tau_1 (Nm)$	$\tau_2 (Nm)$	$\tau_3(Nm)$	$\alpha (rad)$	$\sigma (rad)$	$\eta (rad)$
1	0	0	0	0	0	0
2	0	0	0	0	$\frac{\pi}{2}$	$\frac{\pi}{2}$
3	0	0	0	0	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$
4	10	0	0	0	0	0
5	7	0	0	0	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$
6	0	7.06	0	0	0	0
7	0	7.06	2.69	0	0	0
8	0	5.74	1.91	0	0	0
9	Ramp	0	0	0	0	0
10	Ramp	0	0	0	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$

In the following scenarios, the initial angle of each joint will be specified and then either constant torques will be applied to the joints or, in the case of the final two scenarios, a ramp input will be applied to the first joint to assess whether the system

behaves in the expected way. Table 4-2 gives the input torque and joint angle initial states for each simulation. The initial conditions for the angular velocities and angular accelerations in each of the following simulations for all joints is 0 rad s^{-1} and 0 rad s^{-2} respectively.

Scenario 1

The first scenario involves initial angles of $\alpha = \sigma = \eta = 0 \text{ rad}$ and input torques of 0 Nm for all joints. In this case it is expected that α will remain at 0 rad since there is no input torque and the axis of rotation does not allow for any moment to be produced by the weight of the arm. The other two joints do allow weight to produce a moment which will cause them to rotate. Given that this is the case, both the 2nd and 3rd links will rotate in the negative direction. Both of these links will rotate until there is no force acting in the direction which will cause a moment about the joint. Given that there are no limits to the rotation of the joints currently modelled in to the simulation, all three joints are able to rotate freely in any direction, and in this case the 2nd and 3rd links will rotate until they are pointing straight downwards ($-\frac{\pi}{2} \text{ rad}$). This point is also the position where these links have the least potential energy and so is likely to be the settling point. The inertia in the system is likely to cause an overshoot from this minimum potential energy position, and so the direction of the moment caused by gravity will reverse each link passes the $-\frac{\pi}{2} \text{ rad}$ point. The kinetic friction and static friction models in the system will damp down the oscillations as they will always act against the angular velocity, hence the motion of the link, and this effect will occur at high angular velocities due to the kinetic friction model and when the angular velocity tends to zero due to the static friction model.

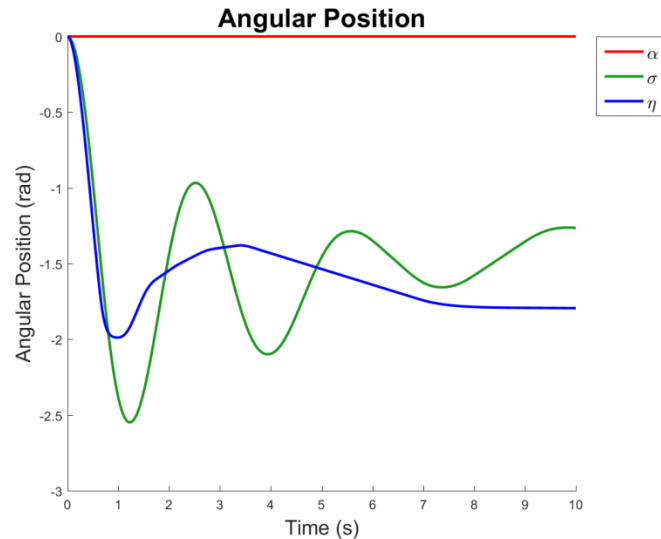


Figure 4-18 Angular position of each joint in the arm model for scenario 1.

As can be seen from Figure 4-18, α remains at 0 rad for the entirety of the simulation, but both σ and η decrease towards $-\frac{\pi}{2} \text{ rad}$. There is oscillation about the settling point in both of the 2nd and 3rd joints, and this oscillation is different for both links. The 3rd link is longer than the 2nd, but the 2nd link has the inertia of both 2nd and 3rd links. This means that the friction terms take longer to overcome the inertia as there is more mass to slow down, and so the 2nd link has a longer settling time than the 3rd. Neither the 2nd or 3rd links have settled at exactly $-\frac{\pi}{2} \text{ rad}$, but this is due primarily to the effect of static friction as the angular velocity of the system has to cross past the 0 rad s^{-1} at the maxima and minima of each oscillation, and this extra kinetic friction from having a low angular velocity helps to remove energy from the system until the point where the static friction overcomes the moments caused by weight, and the system decelerates to a halt. The final position of each joint is displayed in Figure 4-19.

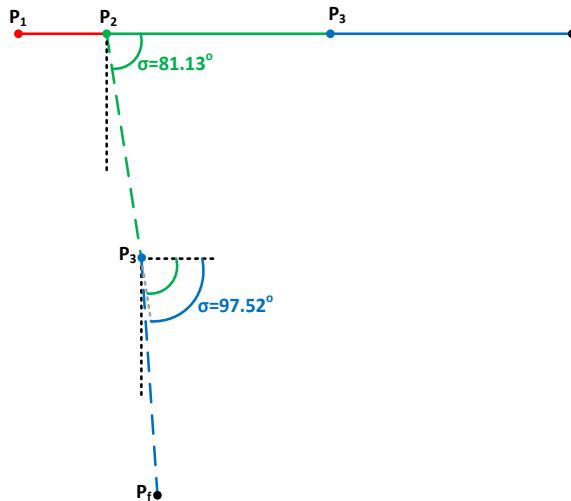


Figure 4-19 Resultant joint angle locations for scenario 1 in relation to the starting geometry.

The effect of static friction can be verified by simulating the same system with static friction removed and determining the steady-state values of each joint angle.

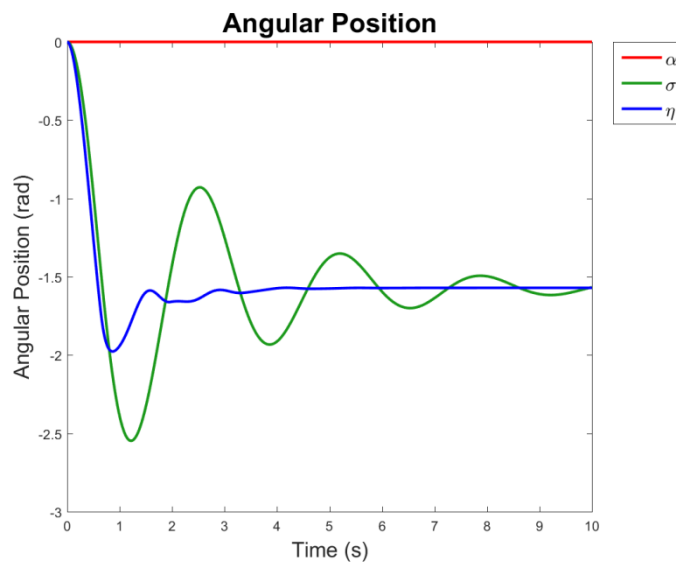


Figure 4-20 Angular position of each joint in the arm model for scenario 1 with static friction removed.

The results shown in Figure 4-20 provide the final value for each of the joint angles σ and η when static friction is removed from the model. Both of these joints have a

final value of $-\frac{p_i^c}{2}$ which is the predicted location. This shows that as the 2nd and 3rd joints settle towards their final value and their angular velocity reduces so that static friction becomes larger than the kinetic friction term. Since the joints are converging on a vertical angle, the cosine of the angle tends to 0, hence the moments about the joints cause by weight also tend to 0. As the static friction term increases, it becomes larger than the moments caused by weight. In reality, the point at which the moment cause by gravity and the static friction are equal for each joint will be the final value for the system. The effect of the friction model can be further investigated by inspecting each term in the mechanical equation separately.

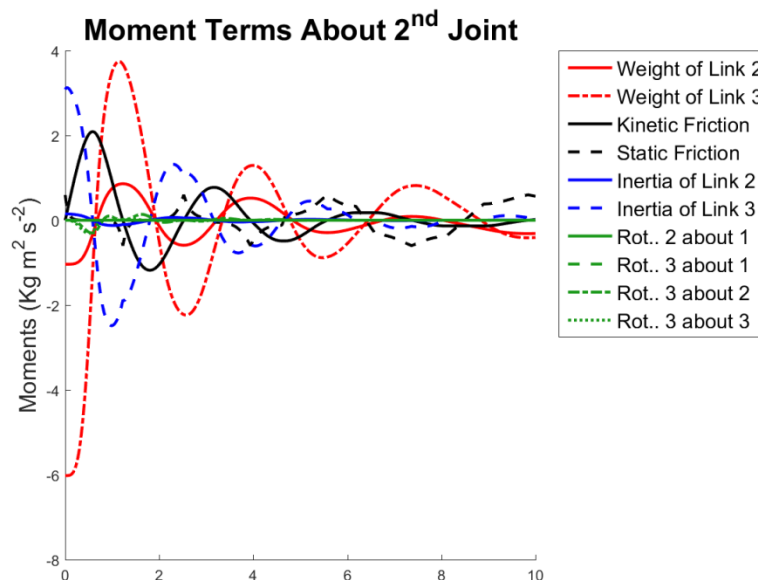


Figure 4-21 Moment terms about σ (excluding input torque) for scenario 1.

As can be seen from Figure 4-21, the weight of the links has a large effect on the overall torque acting on each joint. As each joint converges on $-\frac{\pi}{2}$ rad the component of weight action perpendicular to the link tends to zero, and as the angular velocity of each link tends to zero, the link enters the static friction range and the kinetic friction torque becomes less prominent. This is clear from the reduction in the magnitude of the sinudoidal shape which is present in the solid black line in the figure. As the angular velocity of the joint reduces, the static friction moment (dotted

black line) becomes more apparent. Once the value for the total moment acting on the joint becomes smaller than the static friction term, the static friction is only generated at the same magnitude since it will only counteract another force, not generate motion by itself. The model has performed in this scenario in a manner which is consistent with the prediction of its behaviour prior to the running of the simulation, and therefore for this scenario it can be considered to be accurate.

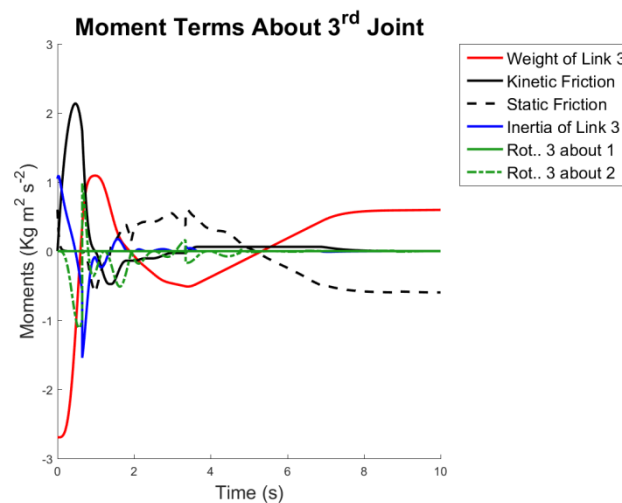


Figure 4-22 Moment terms about η (excluding input torque) for scenario 1.

Scenario 2

The second scenario involves joint angles of $\alpha = 0 \text{ rad}$, $\sigma = \eta = \frac{\pi}{2} \text{ rad}$, and has torque inputs of 0 Nm for each joint. Given these joint angles of the 2nd and 3rd links should be pointing directly upwards in the vertical direction. Since this means that weight has no components acting perpendicular to the links, all of the other terms are proportional to angular velocity and acceleration and there are no external disturbances, there should be no motion whatsoever, and the joint angles should remain constant at their initial values.

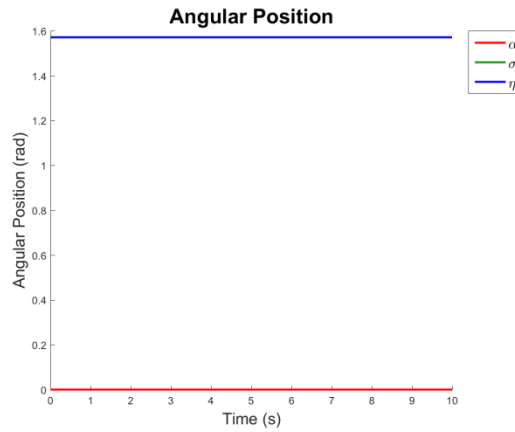


Figure 4-23 Angular position of each joint in the arm model for scenario 2.

Figure 4-23 clearly confirms that the prediction made for this scenario correct since α remains constant at 0 rad and σ and η remain approximately constant at $\frac{\pi}{2} \text{ rad}$. Figure 4-24 and Figure 4-25 show σ and η separately since they overlap.

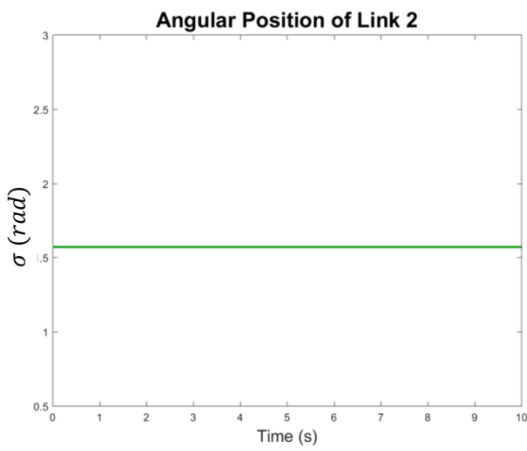


Figure 4-24 Angular position of σ for scenario 2.

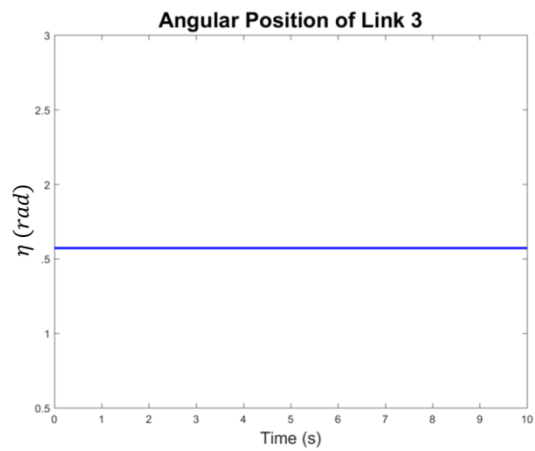


Figure 4-25 Angular position of η for scenario 2.

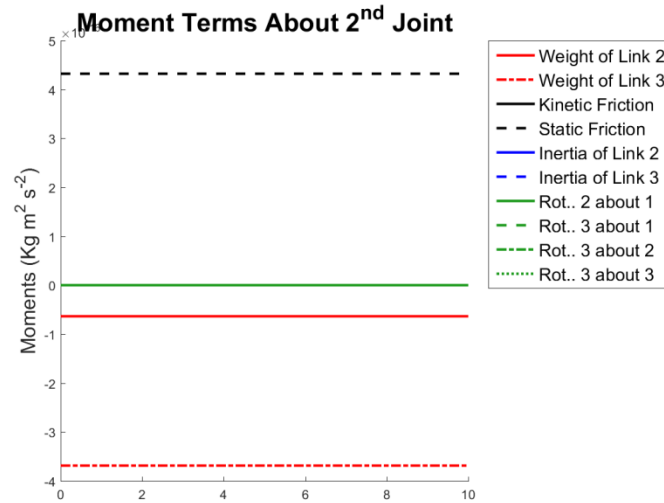


Figure 4-26 Moment in σ (excluding input torque) for scenario 2.

Figure 4-26 and Figure 4-27, show that there is negligible moments caused by any of the terms in the dynamics acting on the system. The order of magnitude of the moments displayed in these figures is 10^{-16} so the moments can be considered to be 0 Nm. The reason for this offset is numerical rounding errors in Simulink. In this scenario the model has performed in a manner which is consistent with the predicted behaviour therefore it can be considered to be relatively accurate with regards to this set of inputs and initial conditions.

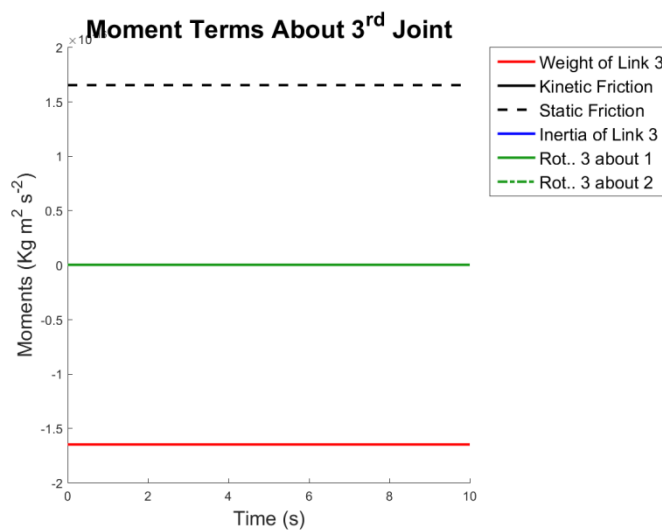


Figure 4-27 Moment in η (excluding input torque) for scenario 2.

Scenario 3

The third scenario involves joint angles of $\alpha = 0$, $\sigma = -\frac{\pi}{2}$ and $\eta = -\frac{\pi}{2}$. Again, the input torques in each of the joints is 0 Nm . This scenario should behave similarly to scenario 2. However, in this scenario the system starts at its minimum potential energy point rather than its highest, as in the previous scenario, which means that any disturbance would not cause the system to diverge from this set of angles, in fact it should converge on this set of angles again should it be disturbed. The reason that the system starts at its minimum potential energy point is that once again the components of weight acting perpendicular to the 2nd and 3rd links are 0, and any motion away from this angle combination would generate a moment caused by weight which acts to rotate the system towards this angle set. This means that the only movement in the system would be caused by numerical simulation error.

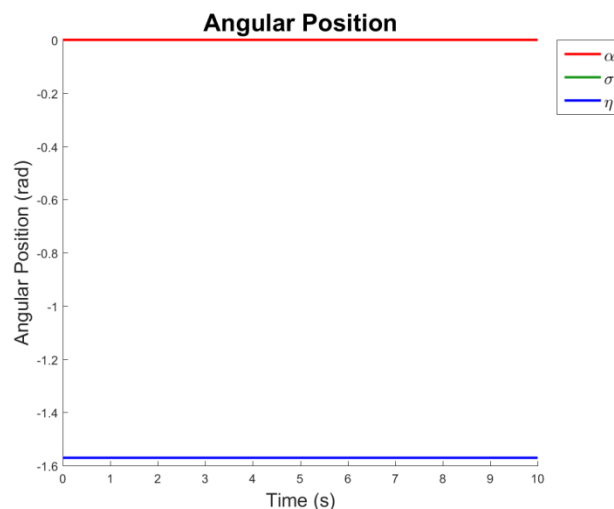


Figure 4-28 Angular position over time of all joints in scenario 3.

Figure 4-28 shows that the system behaves as expected given the initial angular positions of $\alpha = 0$, $\sigma = -\frac{\pi}{2}$ and $\eta = -\frac{\pi}{2}$, and input torques of 0 Nm .

Scenario 4

The fourth scenario involves joint angles of $\alpha = \sigma = \eta = 0 \text{ rad}$, and an input torque on joint 1 of 10 Nm . In this case the input torque will cause link 1 to accelerate until the kinetic frictional torque about joint 1 increases enough to counterbalance the input torque, and then the link will maintain a constant angular velocity. The centripetal effects caused by the rotation of the 2nd and 3rd links about joint 1 will have an impact on the angular positions of σ and η . Since there are no input torques on these two joints, they will initially tend towards the $-\frac{\pi}{2} \text{ rad}$ points, but as the centripetal acceleration increases due to the increase in the angular velocity of joint 1, their tendency will be to move radially away from joint 1 and this will provide a moment which rotates them towards 0 rad . Since the links have inertia, there is likely to be oscillation about the 0 rad point for both σ and η .

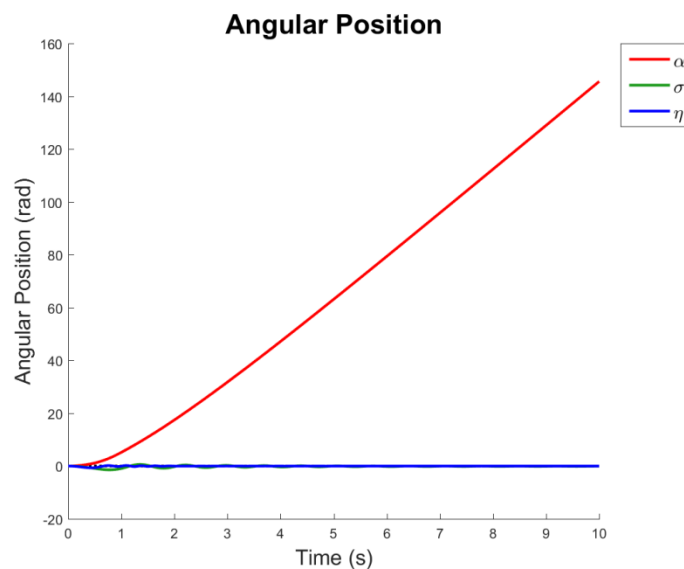


Figure 4-29 Angular position over time of all joints in scenario 4.

As can be seen in Figure 4-29, the angular velocity of α increases for just over 1 second, and then becomes constant, which can be observed from the straight line for joint angle α . Both σ and η begin to drop before α has enough time to accelerate to an angular velocity high enough for the centripetal terms in the mode to have any appreciable effect. Once $\dot{\alpha}$ has increase for centripetal effects to outweigh the

moments due to weight, joint angles σ and η are forced towards 0 rad. This is shown in Figure 4-30. In this 15 second run of the system for scenario 4, the angle range shown on the figure is smaller in order to visibly see the motion of σ and η . In this case it is clear that these two joint angles tend towards $-\frac{\pi}{2}$ initially, but then are driven back towards 0° as the momentum of joint 1 increases. The predicted oscillation occurs as the links approach and pass the 0 radian point, but the frictional terms in the model damp down the oscillations over time.

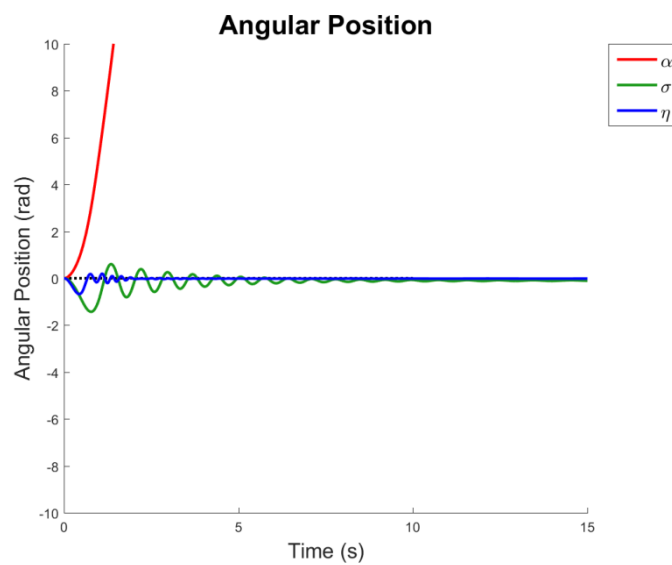


Figure 4-30 Angular position of all joints in scenario 4 over 15 seconds.

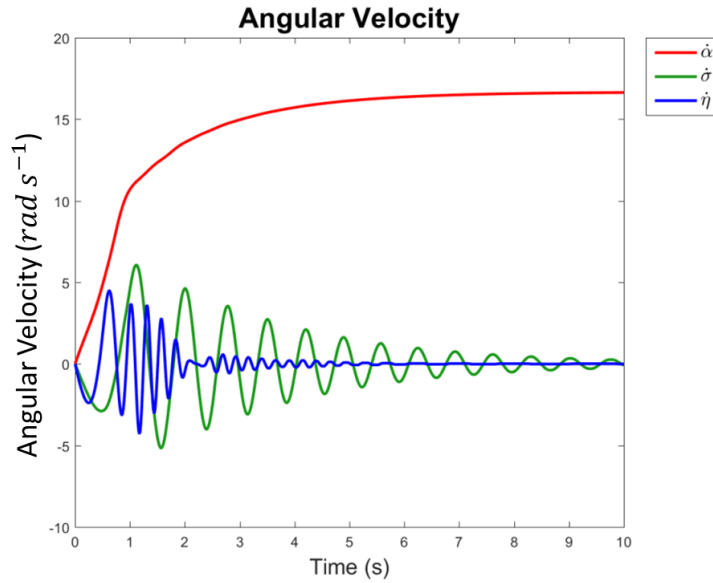


Figure 4-31 Diagram illustrating the angular velocity over time of all joints in scenario 4.

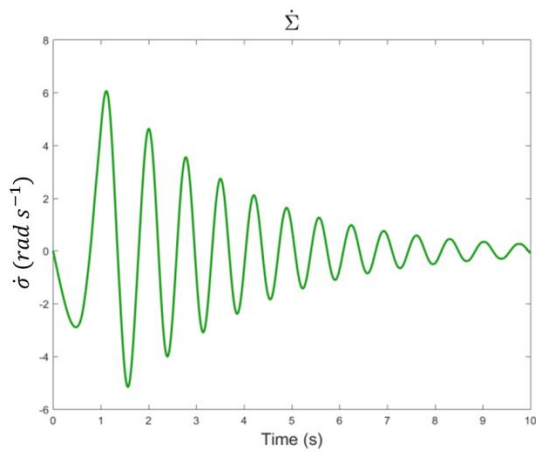


Figure 4-32 Angular velocity of joint 2 in scenario 4.

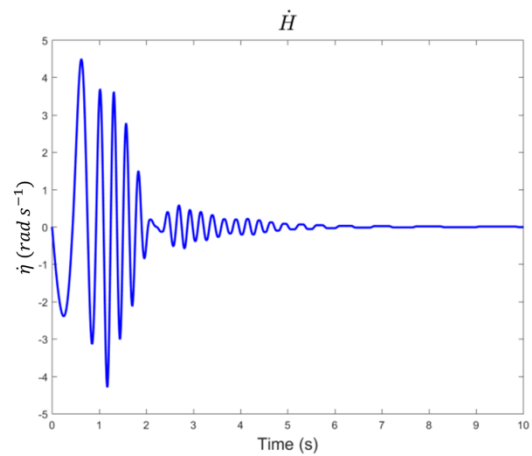


Figure 4-33 Angular velocity of joint 3 in scenario 4.

Figure 4-31 and Figure 4-33 illustrate the motion of the 2nd and 3rd joints over time. As can be seen here, both joints display highly oscillatory behaviour as they initially head towards the $-\frac{\pi}{2}$ rad point but, before 0.5 s for η , and less than 1 s for σ , as the angular velocity of joint 1 ($\dot{\alpha}$) becomes large enough such that the centripetal moment caused by this rotation becomes larger than the moments caused by weight on the 2nd and 3rd joints, and these links are forced to change direction back towards the 0 rad point, and the oscillation begins. As predicted the inertia in the system

causes the 2nd and 3rd joints to oscillate about 0 rad until the oscillations die down due to friction.

These angular velocity results show that $\dot{\alpha}$ increases during the first phase of the simulation, and the acceleration increases initially, but then decreases to 0 rad s⁻¹, as the velocity levels off and becomes constant over time. The uneven shape that occurs on the $\dot{\alpha}$ line is caused by the change in inertia of the system as σ and η tend towards 0 rad.

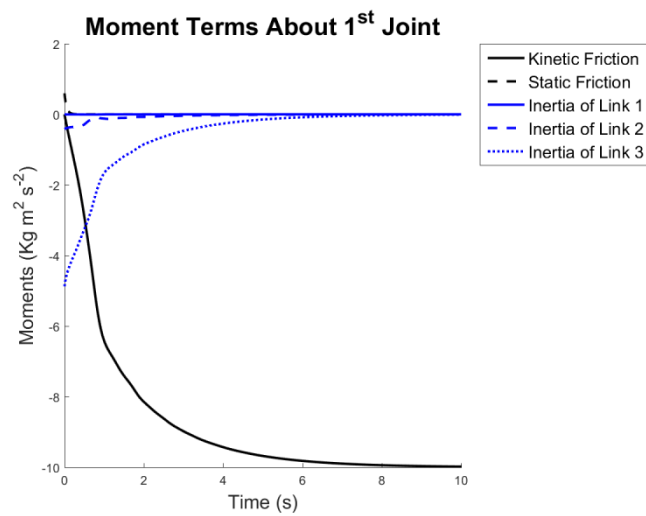


Figure 4-34 Moment terms about α (excluding input torque) for the scenario 4.

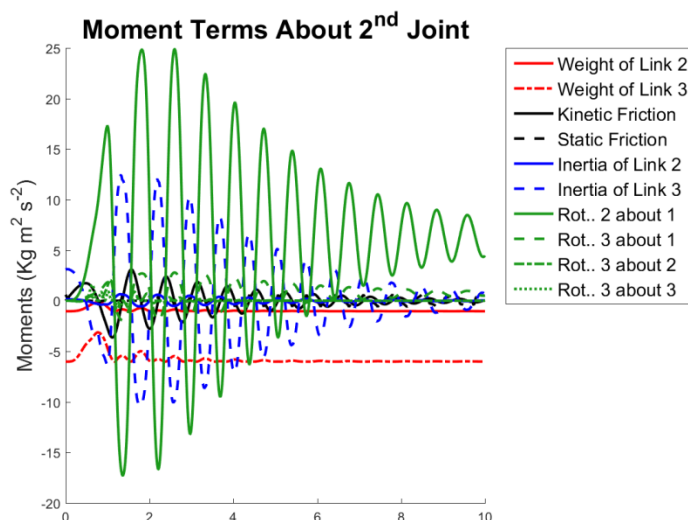


Figure 4-35 Moment terms about σ (excluding input torque) for scenario 4.

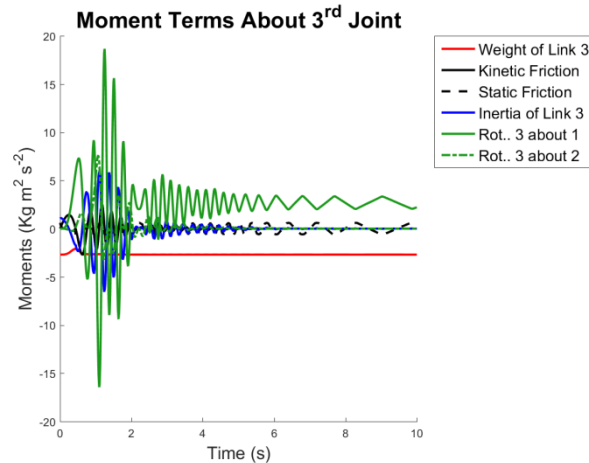


Figure 4-36 Moment terms about η (excluding input torque) for scenario 4.

As can be seen from Figure 4-34 to Figure 4-36, which show the breakdown of moments in the 1st, 2nd and 3rd links, the centripetal terms in the model have the largest effect on the system in this scenario. What can also be observed is that the inertial effects and friction have an effect on the system as well, and it is these terms which cause the oscillations to die down towards the 0 rad point. As can be seen from the breakdown of moment terms about α , kinetic friction increases over time, which is the term that brings the angular acceleration of the joint to 0 rad s^{-2} and stabilises the angular velocity. As described earlier in this section, the inertia of links 2 and 3 have a substantial effect between 0 and 2 seconds, and this correlates precisely to the irregular shape of $\dot{\alpha}$ during the same time.

A further experiment carried out in this section is to extend the simulation to 30 seconds and cut the input torque to joint α back to 0 Nm after 15 seconds. The rotation of link 1 should decelerate to stationary, and as the centripetal effects reduce on the 2nd and 3rd links, their position should tend towards the $-\frac{\pi}{2}$ radian point again. The input torque signal and the resultant position of the joints are shown in Figure 4-37 and Figure 4-38.

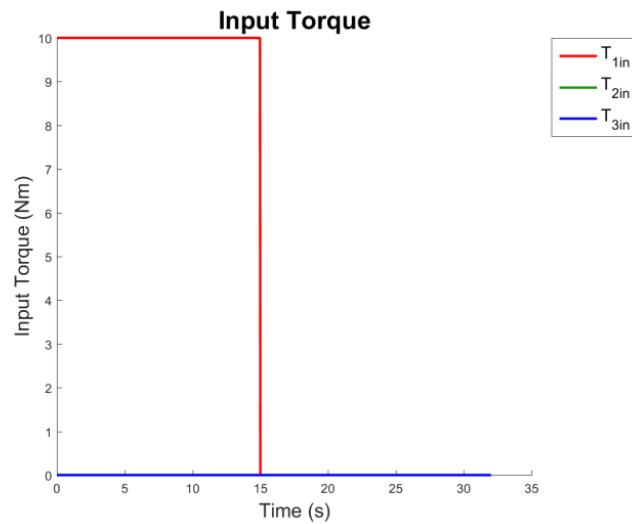


Figure 4-37 Input torque to all joints for a modified version of scenario 4.

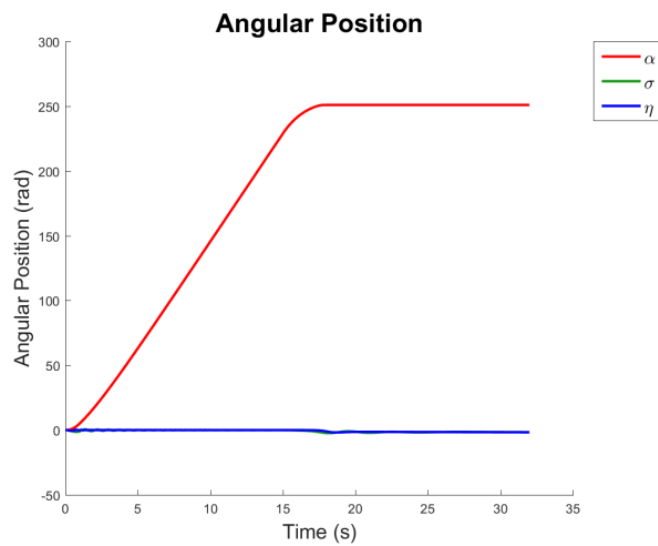


Figure 4-38 Angular position of all joints for a modified version of scenario 4.

There is a visible dip in the positions of σ and η at the time of approximately 17 seconds which corresponds to the torque removal from and deceleration of α . By inspecting the range of time from this point to the end of the simulation, the result is more visible.

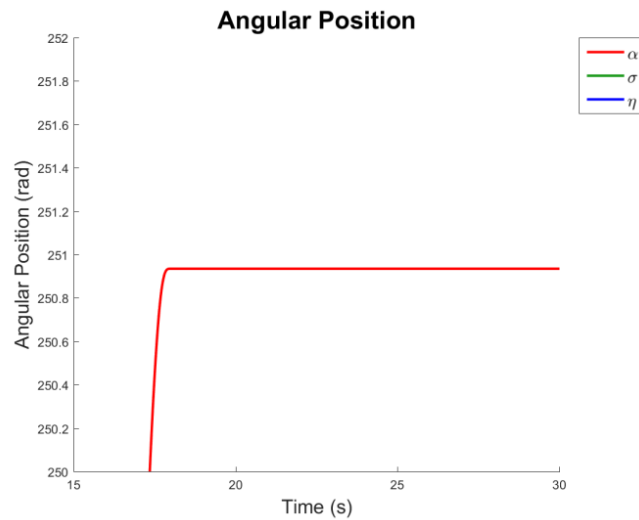


Figure 4-39 Angular position of joint α for a modified version of scenario 4.

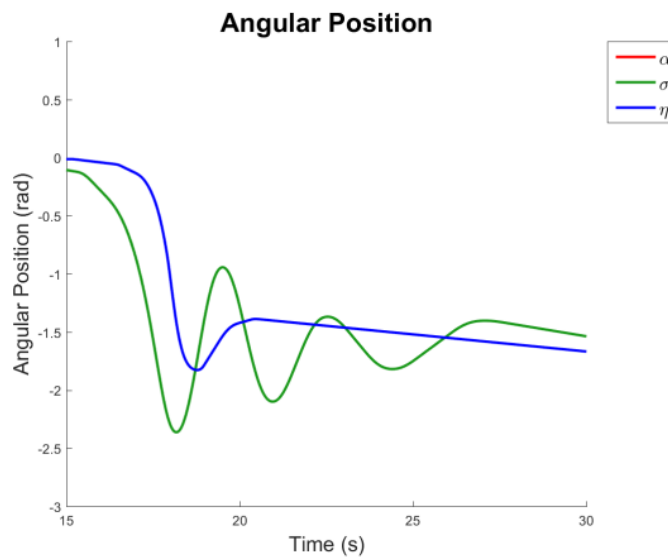


Figure 4-40 Angular position of joints σ and η for a modified version of scenario 4.

Figure 4-39 clearly shows the deceleration of α to a final value. The time taken to stop following the step change in input torque at a simulation time of 15 seconds, is due to the inertia of the arm, especially since the arm is completely extended. Figure 4-40 shows how joint angles η and σ drop to the $-\frac{\pi}{2}$ radian point. Due to the inertia of these links, the angle at which these joints angles settle from oscillation is more negative than $-\frac{\pi}{2}$, but the links are tending to a final value at the end of the simulation.

The behaviour of the system during this simulation is consistent with the behaviour during previous scenarios and the prediction of its behaviour during this scenario, as such the model can be considered accurate with regards to this set of inputs and outputs.

Scenario 5

In the fifth scenario, the initial angular positions are $\alpha = 0 \text{ rad}$, $\sigma = -\frac{\pi}{2} \text{ rad}$ and $\eta = -\frac{\pi}{2} \text{ rad}$ and an input torque to joint 1 of 7 Nm , and input torques of 0 Nm for each of the 2nd and 3rd joints. In this scenario the increase in angular velocity is predicted to have the same effect as that of the previous scenario. The angular positions σ and η should tend towards 0 rad , with a larger amount of oscillation than in the previous scenario, since they are likely to have a higher angular velocity during their approach to the 0 rad point.

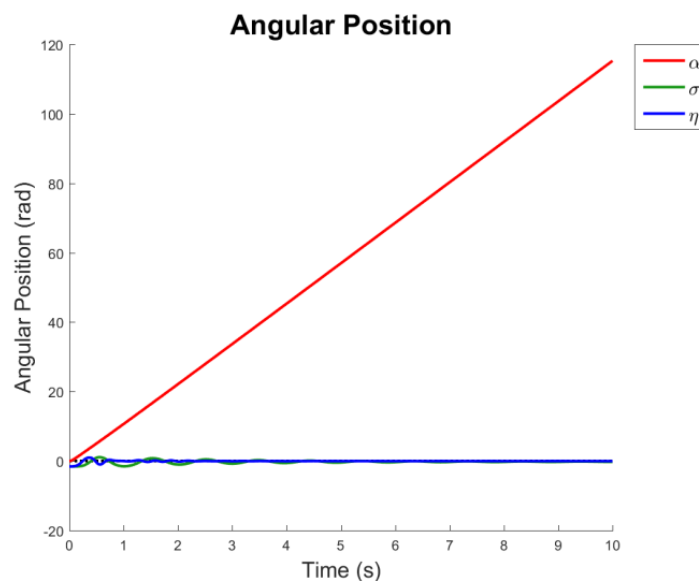


Figure 4-41 Angular position over time of all joints in scenario 5.

As predicted, the same effect has occurred as in the previous scenario. However, the reason for presenting the results to an input torque of 7 Nm to joint 1, rather than the same input torque of 10 Nm as in the previous scenario is because for a larger

torque than about $7 Nm$, the system tends to become unstable and the simulation collapses.

In the case of an input to joint 1 of $10 Nm$, the 2nd and 3rd links are given enough energy by the rotation of α to drive them over the $\frac{\pi}{2} rad$ point. Once this happens, the moments caused by the component of weight perpendicular to each joint works alongside the moments caused by the centripetal acceleration of the links about α to accelerate $\dot{\sigma}$ and $\dot{\eta}$. This process continues as joints 2 and 3 rotate completely by 2π , and complete the entire circle. These joints continue to accelerate towards ∞ . However, for torques less than $7 Nm$, the system settles to a finite value, in this case approximately $0 rad$ for both σ and η .

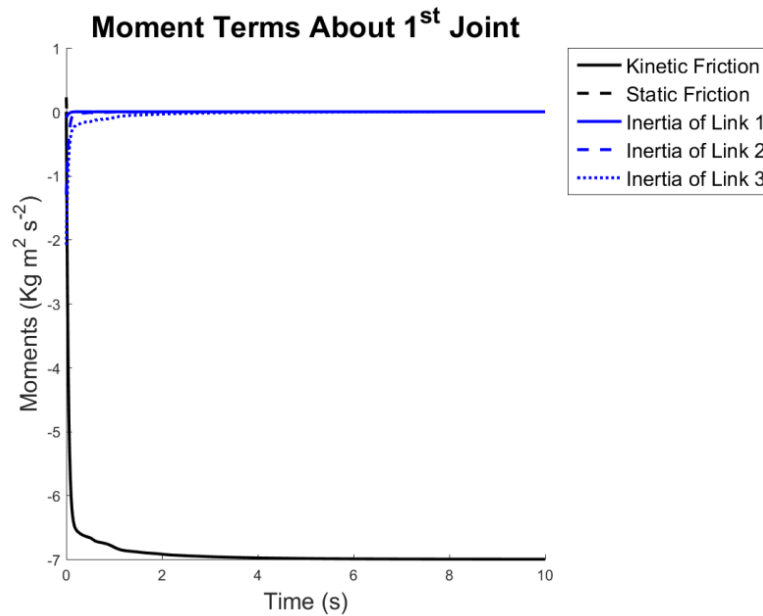


Figure 4-42 Moment terms about α (excluding input torque) for scenario 5.

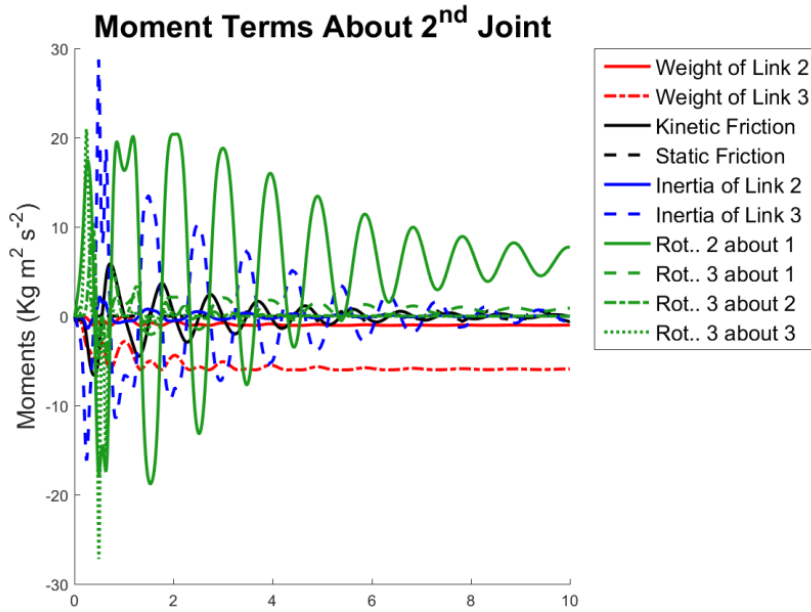


Figure 4-43 Moment terms about σ (excluding input torque) for scenario 5.

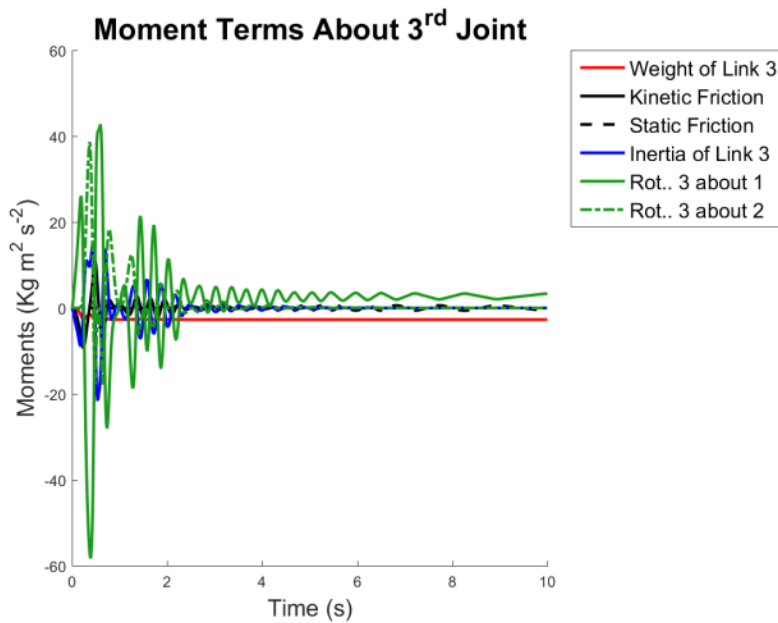


Figure 4-44 Moment terms about η (excluding input torque) for scenario 5.

As Figure 4-42 to Figure 4-44 (which show the breakdown of moment terms in each joint) illustrate the centripetal terms have a large effect on the positions of the 2nd and 3rd joints, as these terms tend to cancel out the moments caused by weight. Once

again the system has behaved as expected, and all of the observed phenomena can be explained by the same set of results.

Scenario 6

At this point, having investigated the effect on the system of inputs torques of $0 Nm$ on the 2nd and 3rd joints, investigating whether the model behaves in an expected manner to different input torques in these joints is also important. To that end this scenario will apply an input torque to joint 2 which is designed to counterbalance the effect of weight of the 2nd and 3rd links on the same joint. The holding torque for joint 2 at $\sigma = 0 rad$ and $\eta = 0 rad$ can be calculated by Equation (4.44).

$$\tau_{holding_2} = c_2 m_2 g + (l_2 + c_3) m_3 g \quad (4.44)$$

The resultant torque is $7.06 Nm$, and this will be inputted into joint 2, along with torques of $0 Nm$ to joints 1 and 3, and the initial angular positions of all three joints will be at $0 rad$.

Since η is free to move it will accelerate downwards due to the effect of its weight, and its position will still tend to $-\frac{\pi}{2} rad$. As the 3rd link drops down from $0 rad$ the changing geometry of the system will affect the overall centre of gravity and drive it closer to the point of rotation about σ . This will reduce the total moment caused by weight about joint 2, and also reduce the moments of inertia acting about the 2nd joint. As this process occurs, the torque required to hold joint 2 at $0 rad$ will decrease, becoming less than the constant input torque into the joint, and so $\dot{\sigma}$ will accelerate therefore σ will increase.

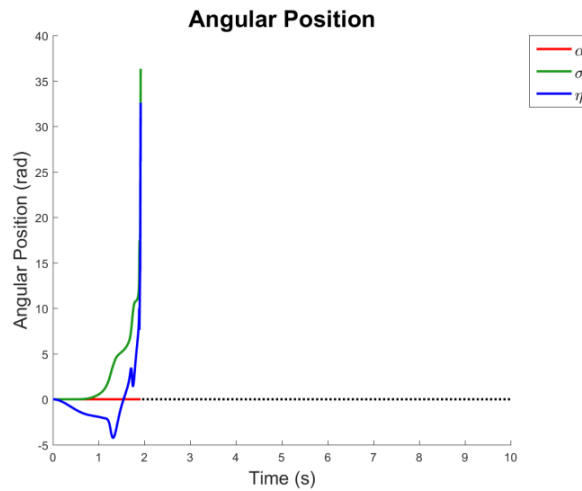


Figure 4-45 Angular position of each joint in scenario 6.

As can be observed from Figure 4-45, the prediction is accurate since η begins to drop towards $-\frac{\pi}{2}$ rad immediately as the simulation starts. As the weight moment and inertia acting about σ decrease, the input torque is able to accelerate σ and its magnitude increases. As $\dot{\sigma}$ increases and σ gets larger, the effect on η is dramatic, and the result is that both the 2nd and 3rd joint angles begin to increase rapidly until the effect of the rotation of link 3 about σ caused by the increasing $\dot{\sigma}$ drives $\ddot{\sigma}$ and $\ddot{\eta}$ to ∞ and the simulation breaks down. This can be observed by inspection of Figure 4-46.

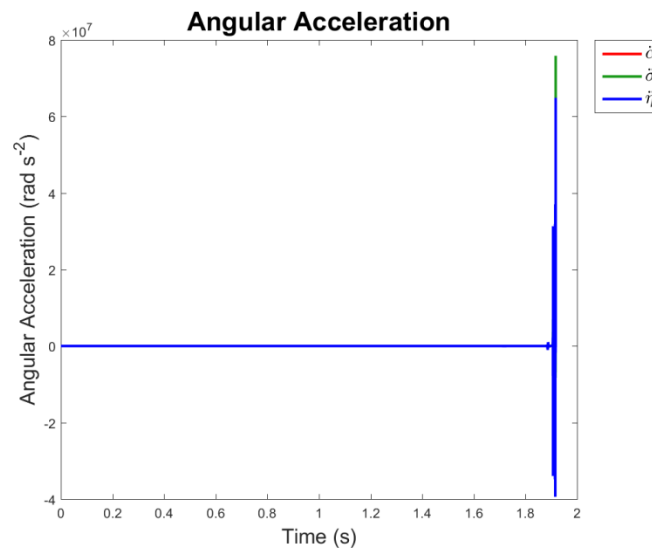


Figure 4-46 Angular acceleration of each joint in the arm during scenario 6.

Scenario 7

The 7th scenario deals with initial conditions of $\alpha = \sigma = \eta = 0 \text{ rad}$ and the joints 2 and 3 have holding torque inputs which are calculated using Equations (4.45) and (4.46) as follows.

$$\tau_{\text{holding}_2} = c_2 m_2 g + (l_2 + c_3) m_3 g = 7.06 \text{ Nm} \quad (4.45)$$

$$\tau_{\text{holding}_3} = c_3 m_3 g = 2.69 \text{ Nm} \quad (4.46)$$

The input torque to joint 1 is 0 Nm . In this scenario, the prediction is that the holding torques will counter the moments generated by weight acting on the system. At zero angular velocity the static friction model generates a maximum coefficient of friction of 0.6 which will act against any other moments present in the joint, but without velocity the kinetic friction moment is 0 Nm . The resultant response of the system should be one of zero angular velocity, hence no movement.

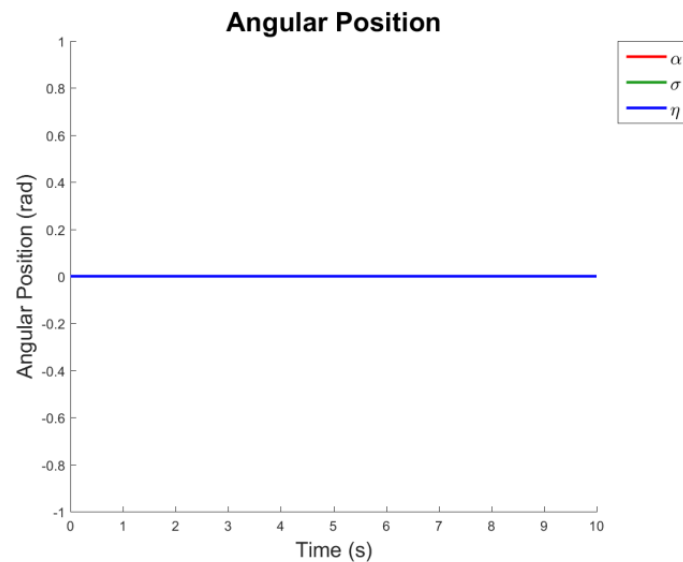


Figure 4-47 Angular position of each of the joints during scenario 7.

As can be seen Figure 4-47 there is no motion in the arm therefore the calculated holding torque has provided the correct moments about each joint to counteract the moments due to weight. An inspection of the moment terms about joints 2 and 3 will confirm this.

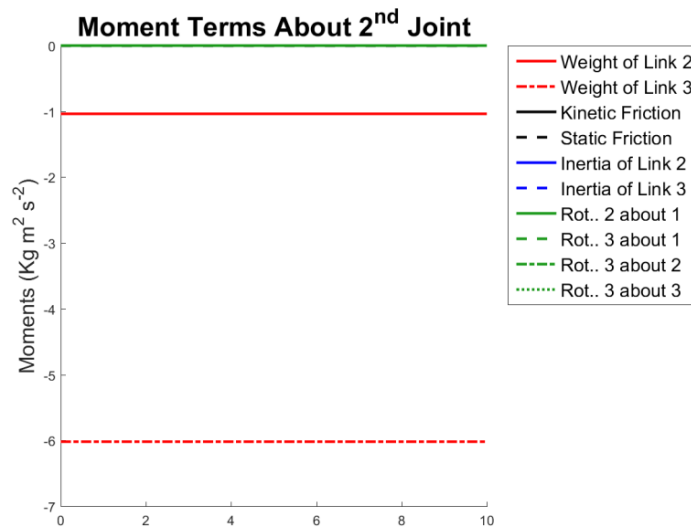


Figure 4-48 Moment terms about σ (excluding input torque) for scenario 7.

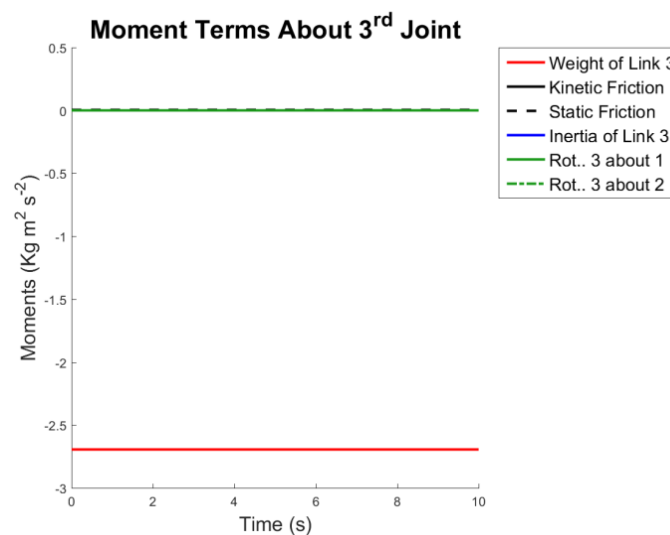


Figure 4-49 Moment terms about η (excluding input torque) for scenario 7.

As can be seen from the moment terms about the 2nd and 3rd joints, all of the moments but those related to weight have a magnitude of 0 Nm. The moments caused by the weights of links 2 and 3 add up to the inputted holding torque, and for η the moment caused by the weight of link 3 is equal to the inputted holding torque.

Scenario 8

For the 8th scenario, the input torques selected are to hold σ and η at an angle of $\frac{\pi}{4} \text{ rad}$. Since the initial angles of the joints are $\alpha = \sigma = \eta = 0 \text{ rad}$, these torques will not be large enough to overcome the moments caused by the weight of each link and as such both joints will decelerate towards $-\frac{\pi}{2} \text{ rad}$. However, since the rotation of each link towards $-\frac{\pi}{2} \text{ rad}$ will reduce the magnitude of the component of weight which acts perpendicular to each link. Eventually the input torque will be equal to the moments caused by weight and the system will no longer decelerate. As the system passes this point the input torques will be larger than the weight moments and the system will accelerate and move back towards the equilibrium point. As the velocity oscillates about 0 rad s^{-1} kinetic friction will have a larger impact and will reduce the magnitude of the oscillations until the system slows to a final value for each joint. Given that the input torque is equal to the moment cause by weight when the system is at $\frac{\pi}{4} \text{ rad}$, and this moment has the same magnitude at $-\frac{\pi}{4} \text{ rad}$, the prediction is that the system will settle at a final value of approximately $-\frac{\pi}{4} \text{ rad}$ for σ and η .

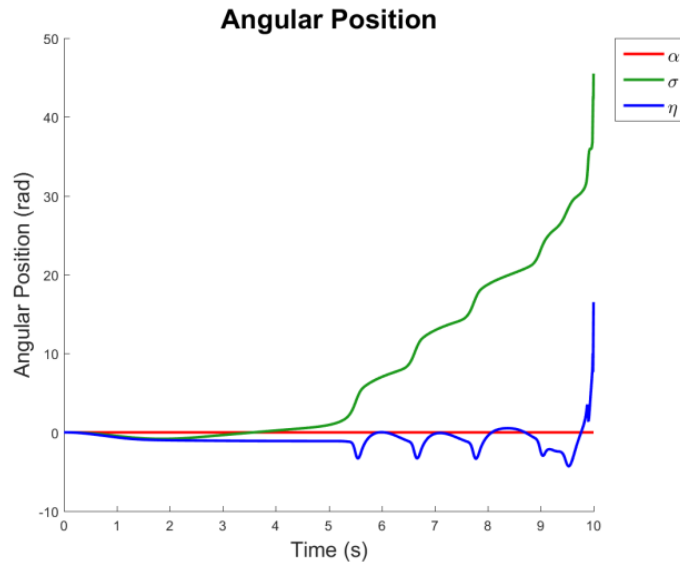


Figure 4-50 Angular position of each joint during scenario 8.

Figure 4-50 shows that the prediction is correct for the first 2 seconds, with joint angles σ and η converging on $-\frac{\pi}{4}$ rad. The system does not settle at a value of $-\frac{\pi}{4}$ for the 2nd and 3rd joints however. This is because the differences in dynamics for each joint means that they fall at different rates and overshoot the $-\frac{\pi}{4}$ radian point. Once the two joints are no longer falling at the same rate the centre of mass of each moves and the moments caused by weight will no longer have the same magnitude as the inputted torques. Once the moments due to weight for σ are less than the input torque for joint 2, the link accelerates, ensuring that the centre of mass will not return to the expected location when the input torque was calculated. The resultant system spins exponentially out of control and the system becomes unstable. Figure 4-51 and Figure 4-52 show that the centripetal and inertial terms increase exponentially to ∞ .

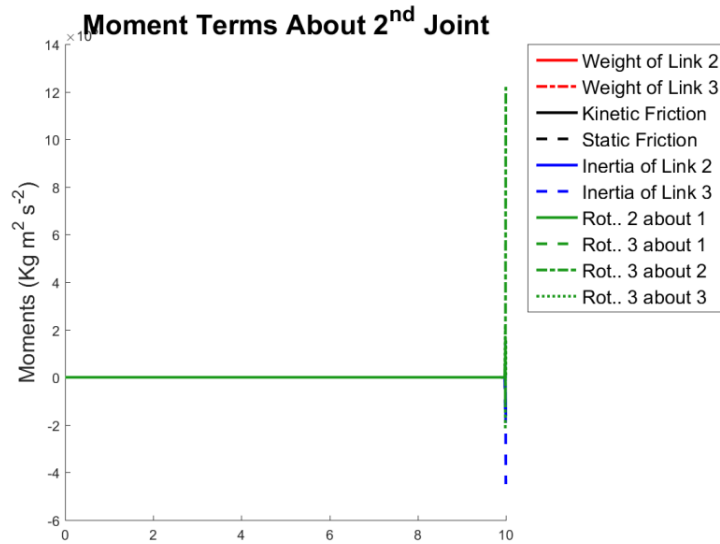


Figure 4-51 Moment terms about σ (excluding input torque) for scenario 8.

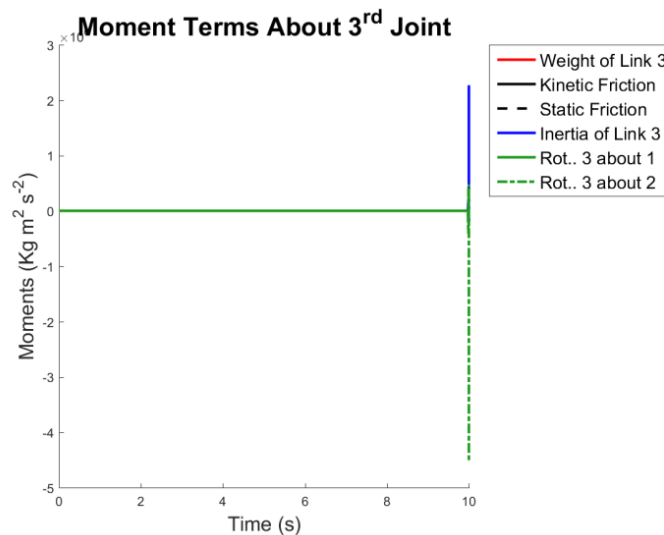


Figure 4-52 Moment terms about η (excluding input torque) for scenario 8.

In this case, the system has not behaved as expected, but the resultant instability has resulted from an inadequate open loop control of the joint angles. The input torque did not take into account the different dynamics of each joint so the system was driven to instability. The result shows why adequate control is required for this type of system.

Scenario 9

It is also useful to investigate how the system responds to a more gradual increase in input torque, rather than the instantaneous step as seen in the previous scenarios, which is not very realistic when considering that the input torque will be provided by servo motors with their own dynamics. For the final set of scenarios, the system will be inspected for its response to a ramp input. The first of these scenarios will have a ramp input to joint 1, with initial value of 0 Nm and a gradient of 1 Nm s^{-1} for initial joint angles of $\alpha = \sigma = \eta = 0 \text{ rad}$.

The prediction here is that σ and η will tend towards $-\frac{\pi}{2} \text{ rad}$ and will drop by a larger magnitude than in scenario 4 since the input torque to α is initially less, therefore $\ddot{\alpha}$ is smaller and $\dot{\alpha}$ takes longer to increase and cause centripetal effects large enough to drive σ and η to 0 rad . The expected result is that this will occur, but that the overshoot for both the 2nd and 3rd joints will be much smaller and the magnitude of the oscillations will die down much more quickly than for the step input approach in scenario 4.

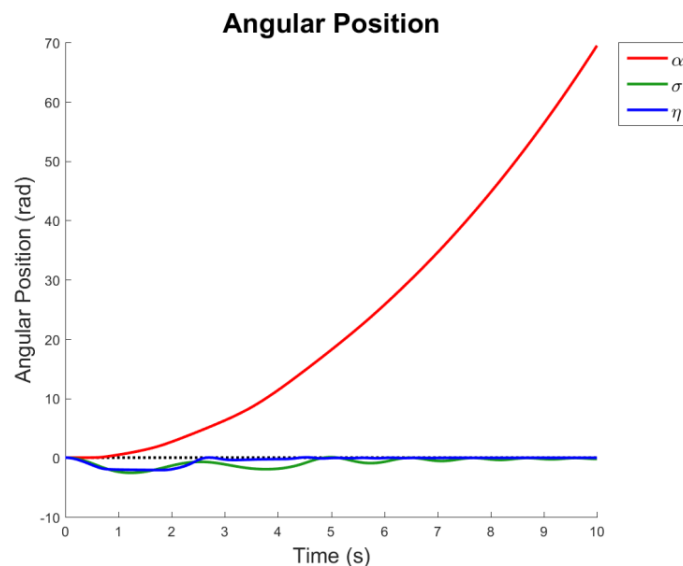


Figure 4-53 Angular position of each joint during scenario 9

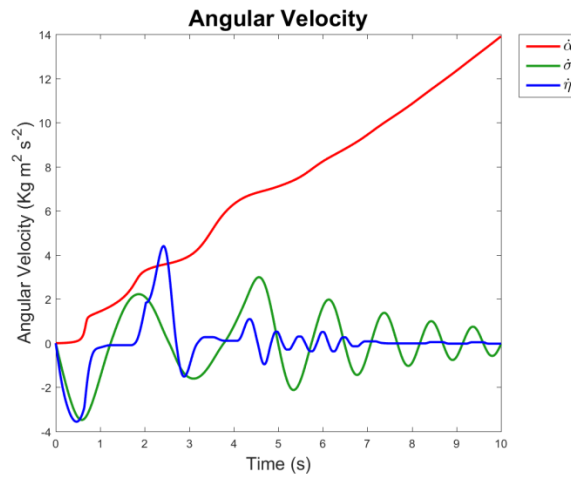


Figure 4-54 Angular velocity of each joint during scenario 9.

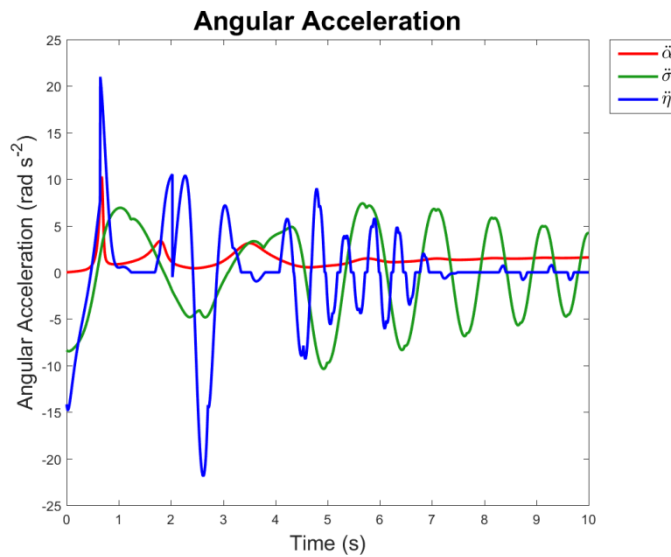


Figure 4-55 Angular acceleration of each joint during scenario 9.

As can be seen in Figure 4-53 to Figure 4-55, the prediction is correct, and the reason for the lower oscillation is that the rate of energy input into the system is not constant, and ramps up from zero, meaning that the centripetal moments increase at a slower rate, reducing the acceleration of σ and η . This can be seen by comparing the moment terms for the 2nd and 3rd links for both this scenario and scenario 4.

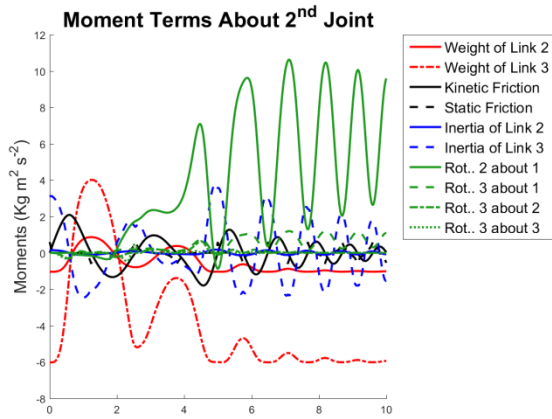


Figure 4-56 Moment terms about σ for scenario 9.

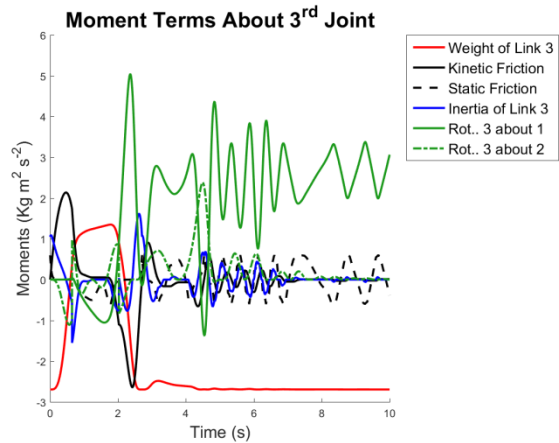


Figure 4-57 Moment terms about η for scenario 9.

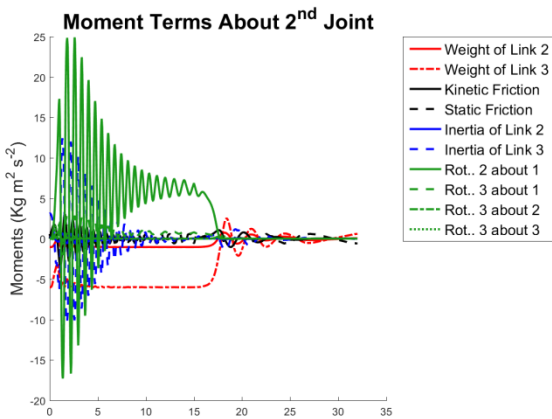


Figure 4-58 Moment terms about σ for scenario 4.

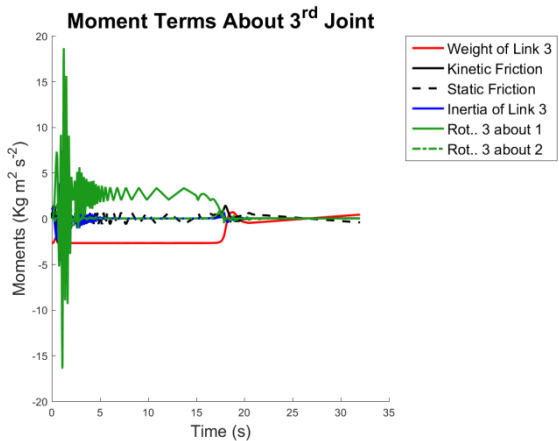


Figure 4-59 Moment terms about η for scenario 4.

In this scenario the centripetal moments ramp up slowly rather than the large increase that occurs earlier in the simulation for scenario 4.

Scenario 10

In this final scenario the stepped input torque to joint 1 for scenario 5 will be compared to a ramp input. Similarly to scenario 5, the initial joint angles will be

$\alpha = 0 \text{ rad}$, $\sigma = \eta = -\frac{\pi}{2} \text{ rad}$, and input torques to the 2nd and 3rd joints of 0 Nm . In scenario 5 this system accelerated to ∞ for any input torque larger than approximately 7 Nm . In this scenario a ramp input of 1 Nm s^{-1} with an initial torque of 0 Nm is used for joint 1 to assess if there is an improvement in the system. The prediction is that this case, similarly to scenario 9, the lower rate of energy input will allow σ and η to settle at 0 rad , rather than accelerating to $\infty \text{ rad}$.

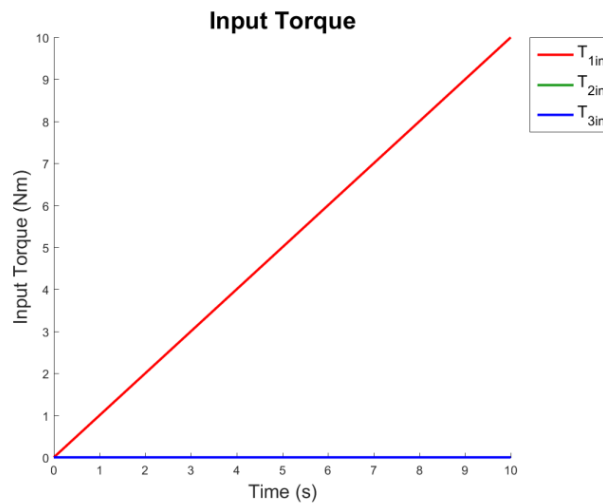


Figure 4-60 Input torques to each joint for scenario 10.

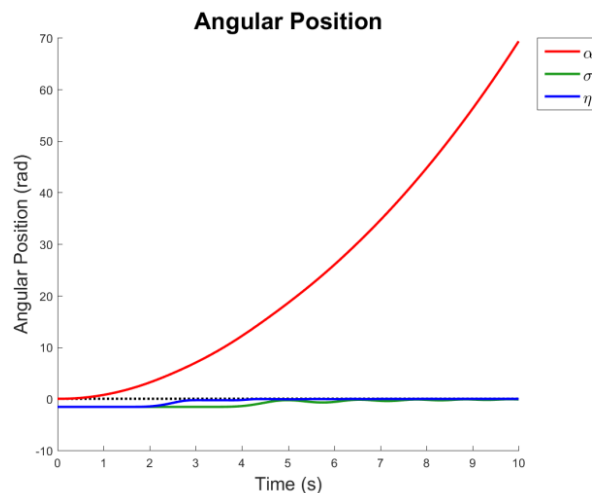


Figure 4-61 Angular position of each joint for scenario 10.

In this case the torque is able to pass the 7 Nm point without causing the system to become unstable, and the σ and η angles rise slowly to 0 rad. Both display a small amount of oscillation, which can be more easily observed in Figure 4-62 and Figure 4-63. In this case η converges more quickly than σ , which displays more oscillatory behaviour. In any case, the magnitude of oscillation is very small, within 1 Nm.

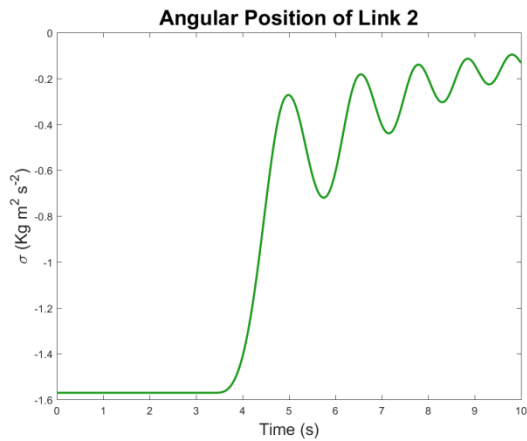


Figure 4-62 Angular position of σ for scenario 10.

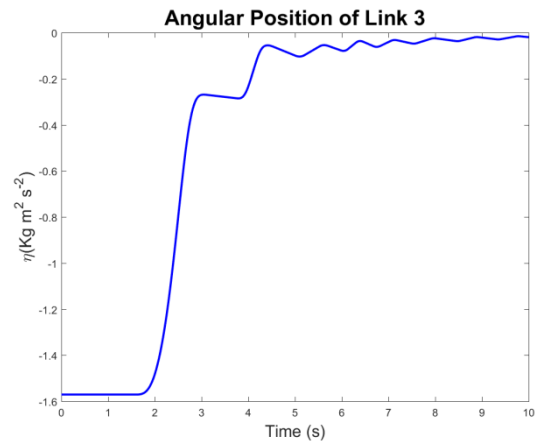


Figure 4-63 Angular position of η for scenario 10.

Again in this case, due to the increasing velocity of α , the centripetal terms have a high impact on the motion of σ and η , which is what would be expected from this system.

4.2.6 Summary of Dynamic Modelling

In the scenarios presented in the qualitative validation, the dynamic model behaved as expected, indicating that the model is qualitatively performing as expected given the assumptions and bounds made. The model is an open loop system which can only be driven by an input torque. To properly drive and control the system a set of three motors is needed, along with feedback on the joint angles to form servo systems which control the arm. The following section of the chapter deals with the modelling of a servo system for use in the model.

4.3 Servo Model

Figure 4-64 shows a circuit diagram of a motor to be implemented in the arm model. The motor is connected to the load by a gearbox. Since the inertia of the arm is large, the inertia of the motor and gearbox will be assumed to be negligible. The frictional part of the motor and gearbox are already modelled inside the arm dynamics.

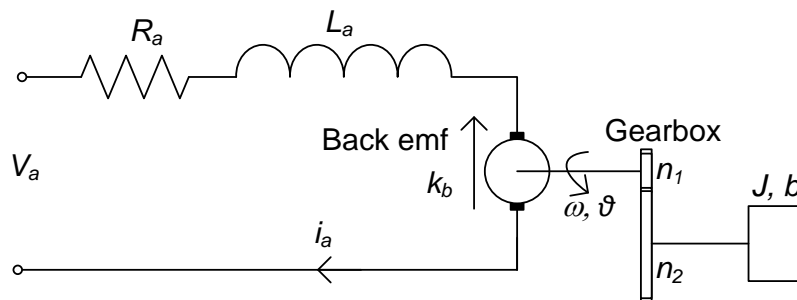


Figure 4-64 Circuit diagram of an electric motor drive connected to a load via a gearbox.

In this case the following can be said:

$$V_a(t) = R_a I_a(t) + L \frac{dI_a}{dt} + k_b \omega_c(t)$$

$$\mathcal{L}\{V_a(t)\} = V_a(s), \quad \text{therefore} \tag{4.47}$$

$$V_a(s) = R_a I_a(s) + L I_a s(s) + k_b \omega_c(s)$$

Equation (4.47) can be rearranged to form Equation (4.48).

$$V_a(s) = (R_a + Ls) I_a(s) + k_b \omega_c(s) \tag{4.48}$$

It can be said that $\tau(s) = k_t I_a(s)$, therefore, in rearranged form this can be substituted into Equation (4.48) to form Equation (4.49).

$$V_a(s) = (R_a + Ls) \frac{\tau(s)}{k_t} + k_b \omega_c(s) \quad (4.49)$$

This can be rearranged further.

$$V_a(s) - k_b \omega_c(s) = (R_a + Ls) \frac{\tau(s)}{k_t} \quad (4.50)$$

The expression on the left hand side of Equation (4.50) can be represented as a single variable $V_i(s)$.

$$V_a(s) - k_b \omega_c(s) = V_i(s) \quad \therefore \quad (4.51)$$

$$V_i(s) = (R_a + Ls) \frac{\tau(s)}{k_t}$$

Rearranging Equation (4.51) once more gives the transfer function displayed in Equation (4.52).

$$\frac{\tau(s)}{V_i(s)} = \frac{k_t}{(R_a + Ls)} \quad (4.52)$$

This can be represented in the system architecture shown in Figure 4-65.

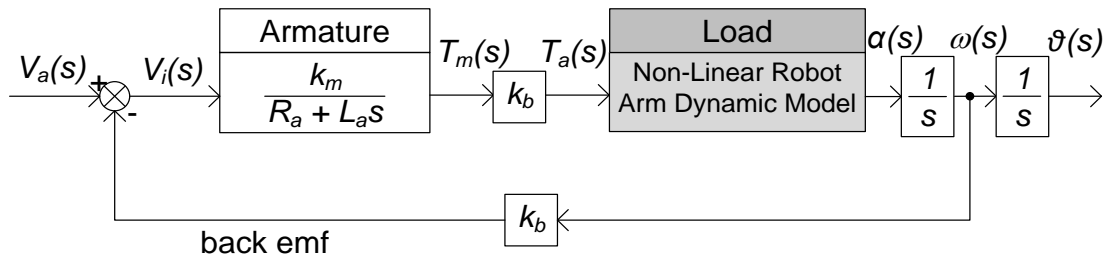


Figure 4-65 Architecture of Robotic Manipulator Joint Servos with Non-Linear Dynamic Model as the Load.

The servo motors have been modelled using the architecture shown in Dorf & Bishop, Modern Control Systems (2006). Realistic values have been chosen from a product datasheet with servo motors that can fulfil the torque requirements. In order to satisfy the requirement a McLennan Servo Supplies P05D, M542E servomotor has been selected for its parameters (McLennan Servo Supplies, Ltd, 2014). This information is given in Table 4-3.

Table 4-3 Servo motor parameters for use in the manipulator arm dynamic model.

Description	Parameter	Value	Conversion
Motor Inertia	J_s	0.438 kg cm^2	0.00438 kg m^2
Motor Friction	B	0.028 Nm	Reduce to zero
Armature Resistance	R_a	1.75Ω	-
Motor Constant	k_m	0.105 Nm/A	-
Armature Inductance	L_a	5 mH	$5 \times 10^{-3} \text{ H}$
Armature PD	V_a	$50 \times 1.1 \text{ V}$	-

The friction in the motor has been reduced to zero to compensate for the high friction coefficient in the dynamic arm model. The nominal voltage has been increased by 10% since voltage can vary by $\pm 10\%$.

While the mass and volume of the motor are not specified in the data sheet, the assumption in this case is that these motors add no extra mass or volume to the manipulator arm, since this could be modelled by increasing length and mass of the manipulator links, and centre of mass could be adjusted accordingly. Also, the torque provided by this motor is not large enough for the maximum required holding torque in the manipulator arm model, but since larger motors would also have larger mass and volume a more realistic approach would be to add a gearbox to the system.

In this case the gear ratio required would be to increase the torque output by 6, therefore a scaling gain is used to simulate the gearbox and scale up the output torque. Again, since the modelled kinetic friction in the manipulator arm is significantly higher than the realistic value, no friction is modelled in the gearbox. Finally, the maximum voltage that can be provided to the motor has been limited to $\pm 10\%$ of the nominal voltage, which in this case is 55 V, so a saturation of 55 V and -55 V has been added to the input of the servo motor so that the voltage in cannot exceed these limits.

Figure 4-66 shows a block diagram illustrated the final architecture of a single servo motor:

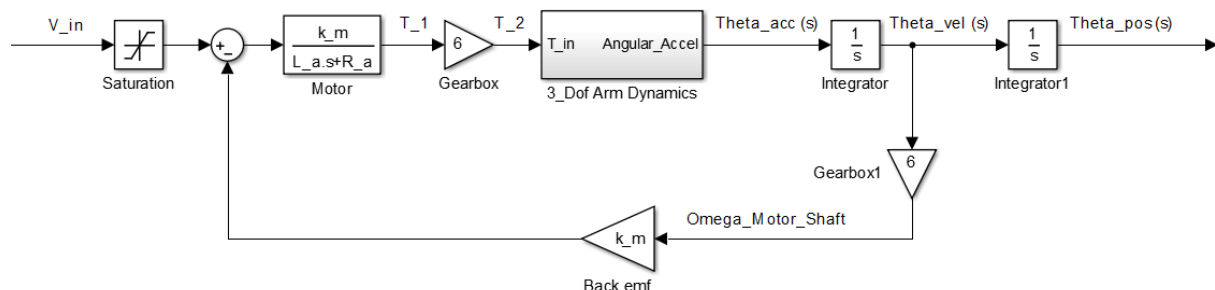


Figure 4-66 Simulink control block diagram for the dynamic model embedded into a servo loop with the specified motor model.

This allows for the following architecture shown in Figure 4-67 to be implemented.

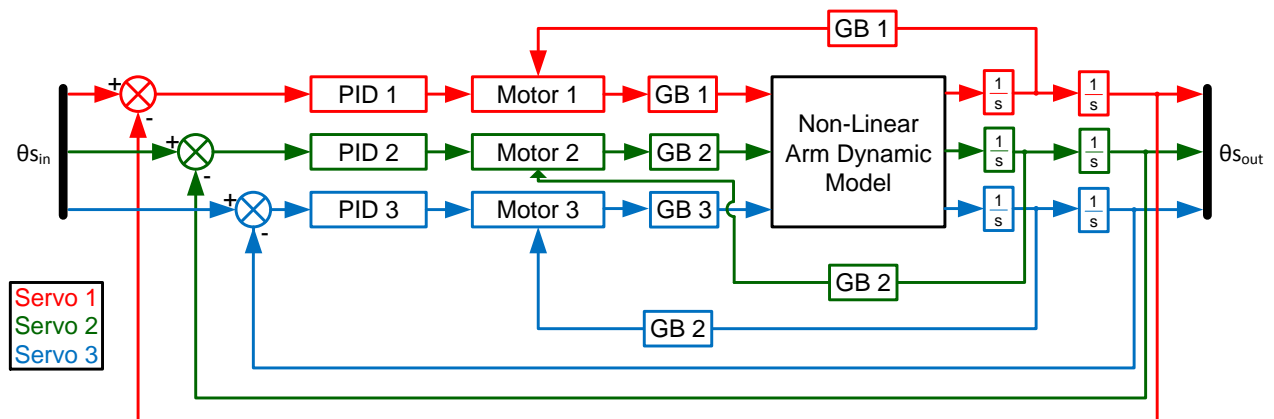


Figure 4-67 Architecture of the PID controlled servo motors surrounding the manipulator arm dynamic model.

Having implemented this system architecture, the PID controllers are required to be tuned. However, as can be seen from the validation results in the previous chapter, this system is non-linear, and the holding moments and inertia change with joint angle. This means that a single controller on each joint will be unable to handle the control requirements of the system over the entire range of joint angles, and so multiple controllers with the same architecture must be used for different angle regions of the joint (in other words the control gains in the controller must be tuned for different angle regions).

4.4 Summary of Dynamic Modelling

This chapter has developed a dynamic model of a 3-DoF manipulator arm including servo model for use as a test-bed for the control and guidance parts of the thesis. In Chapter 5 an appropriate control schema will be selected and tuned to ensure adequate performance for the arm to allow it to track a path. In Chapters 6 and 7 the guidance method is considered, and a path generation technique will be implemented to guide the arm through a close-proximity environment without colliding with any obstacles.

Following the selection of a suitable control schema, the remainder of the chapter deals with the tuning of the control method and validation of the robotic manipulator performance after the tuned controller has been implemented.

5.1 Implementation

There is only one set of physical parameters of the arm which changes during operation. This is the angle of each joint, which is affected only by each joint angle velocity. The second of these can be removed from the problem by maintaining a small rate of change of angle, therefore the change in moments cause by centripetal effects will be small. This means that the only parameters that change the dynamics of the arm are the joint angles themselves. Once this is the case the required gains for the controller on each joint can be calculated offline. This can be done using one of many optimisation methods, and three have been selected based upon their availability in Matlab, genetic algorithms, least squares minimisation and the Nelder-Mead optimisation.

A genetic algorithm (GA) (Goldberg, 1989) is a form of evolutionary algorithm designed to solve both constrained and unconstrained optimization problems based on a natural selection process that mimics biological evolution. The algorithm iteratively modifies a population of individual solutions. At each step, the genetic algorithm selects the fittest individuals from the current population and uses them as parents to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution.

The Least Squares Minimisation Method (Bjork, 1996) is standard approach which is taken in regression analysis, where the approximate solution of a set of equations in which there are more equations than unknowns is searched for. The overall solution to the Least Squares method is an attempt to minimize the sum of the squares of the errors made in the results of every single equation. Since the optimisation problem presented in this chapter is non-linear, the Non-linear least squares method is the form of least squares analysis used to fit a set of m observations with a model that is non-linear in n unknown parameters ($m > n$). It is used in some forms of non-linear regression. This method circumvents the non-linearity of a problem by approximating the model by a linear one and refining the parameters by successive iterations.

The Nelder–Mead method, (McKinnon, 1998) downhill simplex method or amoeba method is a commonly applied numerical method used to find the minimum or maximum of an objective function in a multidimensional space. It is applied to nonlinear optimization problems for which derivatives may not be known. The Nelder–Mead method is a heuristic search method that can converge to non-stationary points on problems that can be solved by alternative methods. In Matlab, this method is known as `fminsearch`.

In order to better understand how optimisation methods work, a Genetic Algorithm has been implemented in the following section.

5.1.1 Genetic Algorithm

Genetic algorithms are naive attempts to mimic evolution in nature. This allows for an iterative change to variables in a system to make an improvement by only selecting the best changes and combining them to create a new set of variables.

Figure 5-2 shows an illustration of a cell with special emphasis on its nucleus. Inside can be seen a simplification of the genetic material as a pair of chromosomes, each one contributed by a different parent.

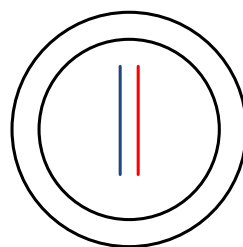


Figure 5-2 A single cell containing two chromosomes.

In normal cell division (during growth), mitosis, the chromosomes are duplicated as a pair as the nucleus splits and as the cell splits into two cells a nucleus is found in each.

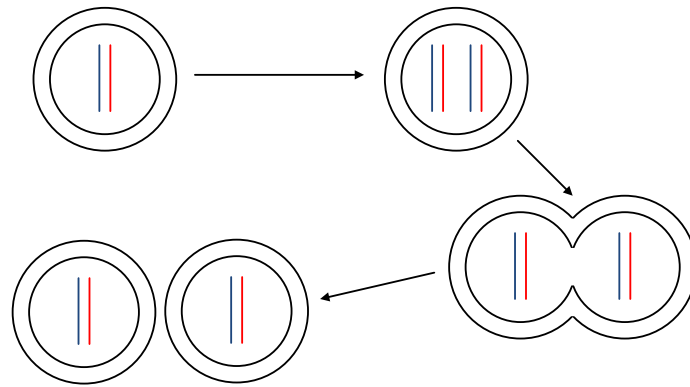


Figure 5-3 A cell undergoing asexual reproduction by mitosis.

This leaves two identical cells which are also identical to the original cell. During reproduction, meiosis, when forming the single chromosome reproductive cells, the chromosomes get broken up into pieces and the genetic material from each chromosome can be recombined into a different combination of characteristics. For example:

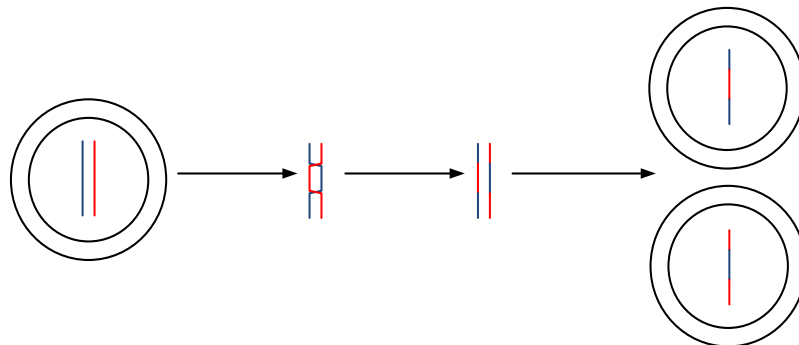


Figure 5-4 Crossover of genetic material from two chromosomes during the first stage of meiosis.

This process is known as crossover. To illustrate the effect this has on the offspring of two parents, the following figure shows two individuals each with different characteristics in their chromosomes. The first individual has the original red and blue chromosomes. The second individual has green and orange chromosomes. As the two produce their reproductive cells, crossover occurs randomly, which is illustrated in the figure by different lengths of chromosome fragments in each individual.

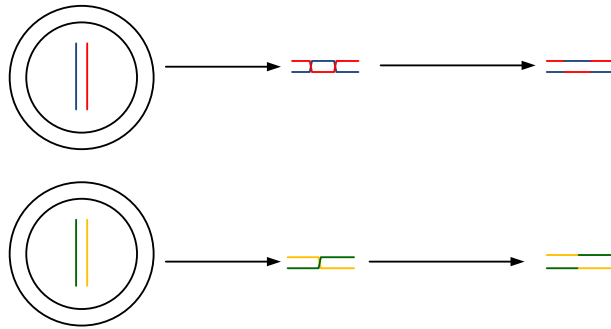


Figure 5-5 Crossover in two parent cells during the process of meiosis. The two parents have different coloured chromosomes.

These four new chromosomes produce 4 reproductive cells, two in each individual (1 and 2). Each reproductive cell has information from the parents of our two individuals in the previous generation, but the information from their parents has been jumbled up.

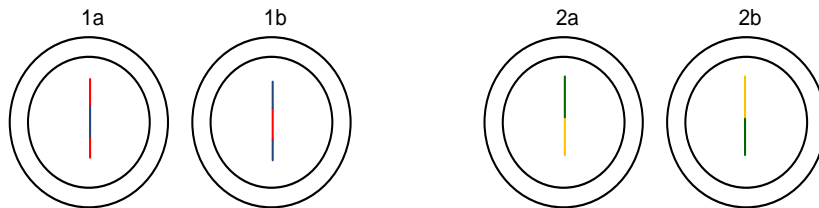


Figure 5-6 Production of reproductive cells in both parents during meiosis.

The reproductive cells from individual 1 can be combined with reproductive cells from individual 2 to form 2 new offspring.

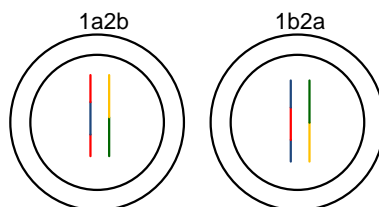


Figure 5-7 Formation of children cells by combination of reproductive cells from each parent cell.

Now there are two new individuals with a different combination of characteristics to their two parents and their four grandparents. The processes of crossover in the production of reproductive cells and combination of reproductive cells from each parent occur randomly, so there can be any number of crossovers (including 0) and any reproductive cell from one parent can combine with any reproductive cell from the other. This helps to produce a very large number of possible combinations of the parents in the offspring.

There is another mechanism occurring in the process of meiosis which causes minute changes in the genetic material. When the chromosomes are mixed up during reproductive cell generation, bits of information can be changed slightly. For example, an 8 bit binary number such as 00101010 (42), could be changed slightly to 11001011. In this case the 1st, 2nd, 3rd and 8th bits have been changed to the opposite state, turning the number 42 into 203. This is known as mutation. Again, the mutation of a bit occurs randomly, and in an 8 bit word, there could be 0 to 8 mutations. To summarise, there are three main processes occurring in genetic reproduction that can be emulated in a genetic algorithm to help solve an optimisation problem:

- Mutation.
- Crossover.
- Genotype to phenotype combination.

This allows for new individuals to be created from the existing population. In nature the individual with the best chances of survival will survive long enough to pass on their genetic material by reproduction. As the population competes to survive and only the fittest survive, eventually those with the worst ability to survive will die off and the whole population will have the characteristics that best aid survival. This is known as survival of the fittest and it is the primary mechanism by which evolution occurs.

As the number of generations increases, mutation and crossover gives chances for slightly better versions of characteristics and better combinations of characteristics than in the previous generation. When this is the case, the offspring with these new characteristics out-compete the other individuals in the population, and the overall

fitness of the population will slowly head to an 'optimal' fitness over successive generations. This is the process that GAs attempt to replicate.

In the case of the use of GAs to solve an optimisation problem, two crucial parts are required. There is a need to specify the problem to be solved as an objective function, i.e. a function which specified the objective(s) to be met. This function is usually set up in such a way that the aim is to find its maximum or minimum value. The other part of the GA is the fitness function, which is designed to give a measure of an individual's suitability to satisfy the goals of the objective function.

5.1.2 Robotic Manipulator Arm Tuning

In the case of the robotic arm dynamic model, there are three controllers that must be tuned to provide acceptable performance. Since the system is non-linear, multiple sets of gains must be determined for different arm states based on the angular position of its joints. To do this objective and fitness functions must be developed to optimise the gains for this system.

An initial starting point is to write a GA which will attempt to solve the non-linear multi-objective problem which is posed in the need to tune a 3-dof robotic manipulator arm. Prior to carrying out the gain optimisation for a problem of this size, validation of the GA against well-known optimisation problems will be carried out to validate whether the GA is able to find solutions to this kind of problem. Also, a comparison of the effectiveness of the implemented GA against other optimisation methods is also carried out to assess the feasibility of use of the implemented GA. Table 5-1 summarises the algorithmic process for a Genetic Algorithm.

Table 5-1 Summary of the algorithmic process for a GA.

Initialisation

- Take initial estimates for the controller gains as inputs (these could be specified by the user or generated randomly).
- Using the initial estimates form an individual. In the case of the robotic manipulator arm, which has three controllers, the gains for each controller will be bundled into groups to form chromosomes, so each controller represents one chromosome of three genes.

Mitosis	<ul style="list-style-type: none"> • Duplicate the individual with a small amount of random mutation to each gene in order to generate the required sample size.
Natural Selection	<ul style="list-style-type: none"> • Run the arm model for the specified scenario with each of the individuals providing the controller gains and assess the fitness of each individual based on how close to a specified system behaviour the manipulator arm operates with.
Meiosis (Including Crossover and Mutation)	<ul style="list-style-type: none"> • Take the top several individuals with the best fitness and select them for reproduction (the number of individuals selected will aim to maintain the specified population size). • Pair off the individuals and carry out reproduction by creating all combinations of chromosomes. (i.e. for two individuals with three chromosomes each, there are eight possible combinations). Mutation will occur randomly for each gene in the pool as the chromosomes are recombined. • The parent individuals that were responsible for reproduction will survive into the next generation of individuals. This is important since the recombination and mutation of the fittest individuals may not provide a better individual, therefore keeping the fittest individuals from the previous generation will prevent a reduced fitness in the population. • There are now ten individual for each two parents in the previous generation, which can then be used to provide controller gains for the arm model and each individual is tested to find their fitness.
Iteration	<ul style="list-style-type: none"> • The process is repeated until an individual is found which satisfies the fitness criterion which is specified by the user (this would be calculated based on the fitness function and what the outcome of the fitness function means in real terms).

Given that the robotic manipulator system displays a large degree of non-linearity, a single set of controller gains for the whole range of joint angles will be inappropriate since the moments about each joint vary by a large amount depending on the joint angle. To account for this, the GA gain tuning can be carried out for discrete joint ranges to provide a series of controller gains over the range of joint angles.

5.1.3 Fitness Functions

In order to select which individuals are the fittest in the population, a fitness function is required which analyses the values of the error between the ideal response of each joint to an input and the actual response of each joint. For the fitness function, several options were tested. To describe the different fitness functions, the nomenclature defined in Table 5-2 is used.

Table 5-2 Summary of the variables used in the formation of the fitness function for the robotic manipulator tuning problem.

Variable	Description	Variable	Description
$Y_{I\alpha}(t)$	The set of ideal output magnitudes over time given a step input to the joint α .	$Y_{A\alpha}(t)$	The set of output magnitudes over time of the joint α for the modelled robotic manipulator arm with PID controller.
$Y_{I\sigma}(t)$	The set of ideal output magnitudes over time given a step input to the joint σ .	$Y_{A\sigma}(t)$	The set of output magnitudes over time of the joint σ for the modelled robotic manipulator arm with PID controller.
$Y_{I\eta}(t)$	The set of ideal output magnitudes over time given a step input to the joint η .	$Y_{A\eta}(t)$	The set of output magnitudes over time of the joint η for the modelled robotic manipulator arm with PID controller.

Where $t \in [t_0, t_f]$. This allows for the following fitness functions to be implemented. Note that $Y = Y(t)$ for all of the sets of outputs in the following equations. Four different fitness functions have been implemented and investigated, all which are intended to reduce the error between the ideal output to a step input and the output of each joint to the same step input. Since the system being tuned is a dynamical system, there exists both a transient stage (which consists of rise, overshoot and settling) and a steady-state stage to the response over time, therefore the following fitness functions have been designed with the intention of providing the best compromise when tuning the system over the whole range of stages.

Sum-Squared Error

The first fitness function designed takes the error over time $e(t)$ for each joint and squares the set to make it positive, forming $e^2(t)$. The set of error squared for each joint is then summed to provide a scalar value, $\sum e^2(t)$. The mean of the sum squared error for all three joints is then taken as the fitness value. The complete fitness function is displayed in Equation (5.1).

$$f = \frac{\sum(Y_{A\alpha} - Y_{I\alpha})^2 + \sum(Y_{A\sigma} - Y_{I\sigma})^2 + \sum(Y_{A\eta} - Y_{I\eta})^2}{3} \quad (5.1)$$

The square of this function has two effects on the fitness value. Firstly it ensures that all fitness values are positive. This is useful since a large negative error would be found to be smaller than a small positive error, and this effect needs to be removed for a useful comparison. Secondly, squaring a number larger than one increases the magnitude, whereas, squaring a fraction decreases the magnitude. This means that errors larger than 1 will be penalised significantly more heavily than those which are less than 1. The fitness for each joint is taken with equal weight in the mean since all three joint performances are equally important. This function worked well in optimising the PID gains to produce a sensible response but took a large number of generations to converge. It was noted by inspection of the responses over successive generations that the area of the response causing the problem was the transient, specifically the overshoot and settling region of the transient.

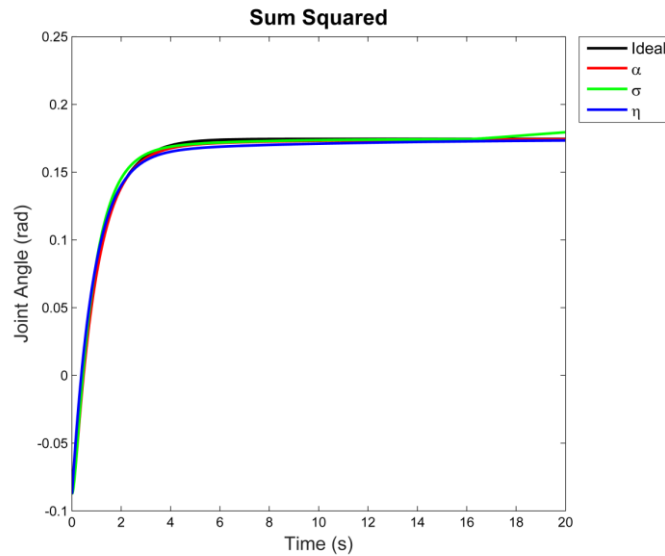


Figure 5-8 Result of the optimisation of a joint angle range in the arm using the sum squared error fitness function.

Weighted Sum of Errors-Squared (Transient and Steady-State)

Since the sum-squared error fitness function does not take into account any weighting between the transient and steady state, and the GA struggled to find gains which reduced overshoot and settling time, finding a way of weighing the errors in certain parts of the response would allow the GA to put more emphasis on solutions with smaller overshoot and settling time. This method splits the time response into two phases, the transient and the steady-state, and a single time is used to split the two phases. This time is denoted as t_1 . When the simulation time is smaller than or equal to t_1 then the square of the error between the actual and ideal outputs are weighted with one value, and when the time of the simulation is larger than t_1 then the square of the error between the actual and ideal output for each joint is weighed by a second value. In order to put emphasis on the transient phase of the response, w_1 is selected to be larger than w_2 . In this case the mean fitness between the three joints is not considered, only the sum since the two parameters are the same in essence, with only a scaling factor changing them. This gives a complete expression which is displayed in Equation (5.2).

$$\begin{aligned}
f = & w_1 \sum_{t_0}^{t_1} (Y_{A\alpha} - Y_{I\alpha})^2 + w_2 \sum_{t_1}^{t_f} (Y_{A\alpha} - Y_{I\alpha})^2 + w_1 \sum_{t_0}^{t_1} (Y_{A\sigma} - Y_{I\sigma})^2 \\
& + w_2 \sum_{t_1}^{t_f} (Y_{A\sigma} - Y_{I\sigma})^2 + w_1 \sum_{t_0}^{t_1} (Y_{A\eta} - Y_{I\eta})^2 \\
& + w_2 \sum_{t_1}^{t_f} (Y_{A\eta} - Y_{I\eta})^2
\end{aligned} \tag{5.2}$$

Where $w_1 = \mathbb{R}^+$ when $t \in [t_0, t_1]$, $w_1 = 0$ when $t \in (t_1, t_f]$ and $w_2 = 0$ when $t \in [t_0, t_1]$, $w_2 = \mathbb{R}^+$ when $t \in (t_1, t_f]$.

Figure 5-9 illustrates how the selection of the value for t_1 allows for the splitting of the response in to the transient and steady-state phases of the response. When selecting weightings it is important to consider how long the system has been run for, as a longer run time will imply a much longer steady-state phase, and therefore this phase of the response is being automatically weighted based on run time.

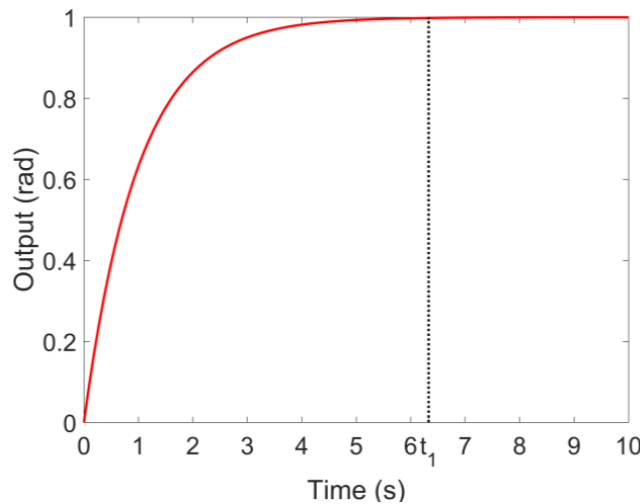


Figure 5-9 Selection of the value t_1 in relation to the transient and steady state regions of a step response.

This fitness function improved the transient part of response, shown in Figure 5-10, in that it reduced the overshoot but at the expense steady-state error, therefore the

aim of this fitness function was achieved but to the detriment of the other parts of the response.

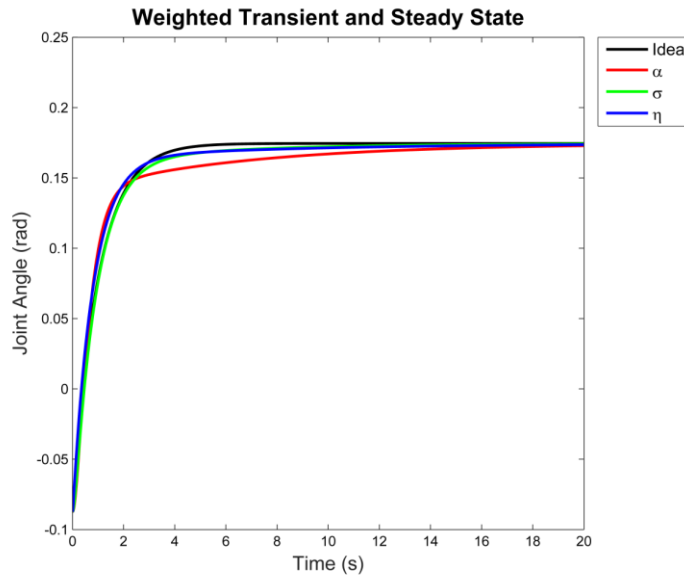


Figure 5-10 Result of the optimisation of a joint angle range in the arm using the Weighted Sum of Errors-Squared (Transient and Steady-State) fitness function.

Weighted Sum of Errors-Squared (Rise Time, Overshoot and Steady-State)

To get a better amount of control over all of the response, the run time of the response can be split up further into several smaller phases. In this case the best split would be to separately weight the rise time, overshoot and steady-state to attempt to find the best balance between them. In this case the rise time section is treated exactly as before, and to provide a better overshoot and steady-state error, the power of these sections has been increased from 2 to 4 in the case of the overshoot and from 2 to 6 in terms of the steady-state error. This has been done so that large errors in these areas are heavily penalised and small errors in these areas are almost rewarded. Again the weighting and times for each phase have to be selected appropriately. This produces the fitness function shown in Equation (5.3).

$$\begin{aligned}
f = w_1 \sum_{t_0}^{t_1} & \left((Y_{A\alpha} - Y_{I\alpha})^2 + (Y_{A\sigma} - Y_{I\sigma})^2 + (Y_{A\eta} - Y_{I\eta})^2 \right) \\
& + w_2 \sum_{t_2}^{t_3} \left((Y_{A\alpha} - Y_{I\alpha})^6 + (Y_{A\sigma} - Y_{I\sigma})^6 + (Y_{A\eta} - Y_{I\eta})^6 \right) \\
& + w_3 \sum_{t_2}^{t_f} \left((Y_{A\alpha} - Y_{I\alpha})^4 + (Y_{A\sigma} - Y_{I\sigma})^4 + (Y_{A\eta} - Y_{I\eta})^4 \right)
\end{aligned} \tag{5.3}$$

Figure 5-11 illustrates how the t_1 and t_2 values can be chosen to split the response into its separate phases.

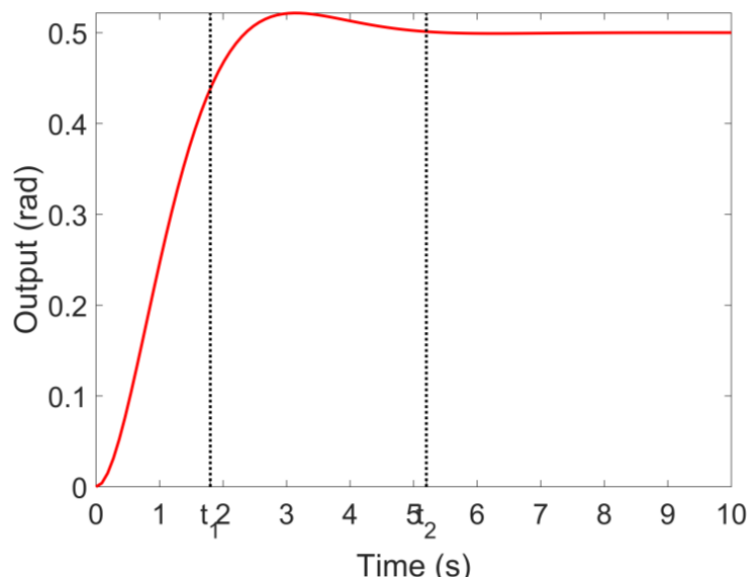


Figure 5-11 Selection of the values t_1 and t_2 in relation to the rise time, settling time and steady state regions of a step response.

This fitness function performed similarly to the previous one in terms of overall fitness, but the higher powers of error in the overshoot and steady-state phases of the response mean that the GA focuses on minimising the error in these phases at the expense of the rise time. The response of each of the joints when tuned using this fitness function are displayed in Figure 5-12.

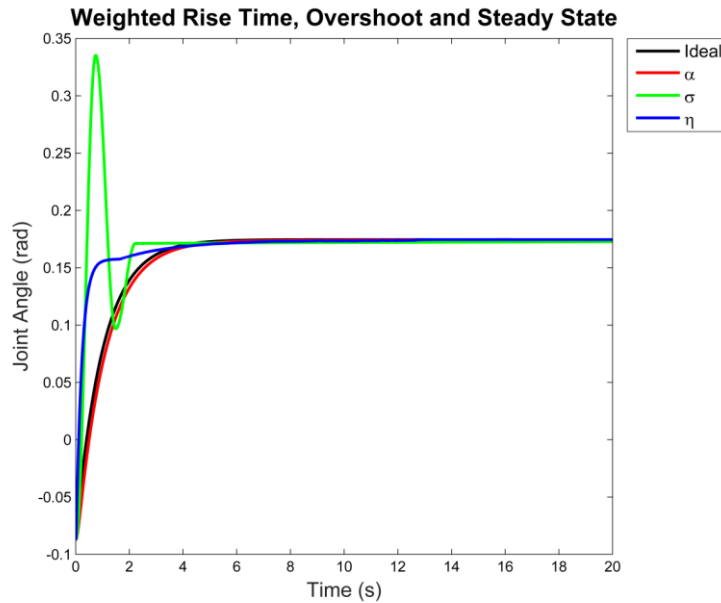


Figure 5-12 Result of the optimisation of a joint angle range in the arm using the Weighted Sum of Errors-Squared (Rise Time, Overshoot and Steady-State) fitness function.

Gaussian and Time Based Weighting

This fitness function attempts to solve the same issues as the previous function. Instead of using higher powers, the function introduces a time-squared element to the function, where an error that remains in the response over long times is penalised more heavily the longer that it exists in the response. In order to reduce the overshoot a Gaussian weighting is introduced at the peak time of the first peak in the system. The intention is to increase the weighting the closer to the peak that the response is and then decrease as the response moves away from the peak. The centre and width of the Gaussian weighting is selected using Equation (5.4).

$$g(t) = e^{-\frac{(t-b)^2}{2c^2}} \quad (5.4)$$

Where, t is the time of the response, b is the centre of the peak, to be selected at t_p of the response and c is the standard deviation of the Gaussian bell, or the RMS

width. This allows for a selection of when this weighting starts and ends as a function of time. The height or maximum magnitude of the weighting would be given by a scaling factor which the whole function is multiplied by, and this is given by w_1 in the fitness function. The time part of the fitness function is carried out by multiplying the error of each joint over time, by the integral of time from t_0 to t_i , where t_i is the time index of that value of error. This part of the function is also weighted and the weighting given by w_2 . This produces the fitness function shown in Equation (5.5).

$$f = \left(w_1 g + w_2 \int_{t_0}^{t_i} t \right) \left(\sum (Y_{A\alpha} - Y_{I\alpha})^2 + \sum (Y_{A\sigma} - Y_{I\sigma})^2 + \sum (Y_{A\eta} - Y_{I\eta})^2 \right) \quad (5.5)$$

Given values, the weighting which the sum of the error-squared is multiplied by can be visualised in Figure 5-13. Using the values $w_1 = 0.5$, $w_2 = 1 \times 10^{-4}$, $b = 2.5$ and $c = 1$. This shows how any error about the peak time of 2.5 seconds will be heavily penalised and any error as the time increases will be heavily penalised.

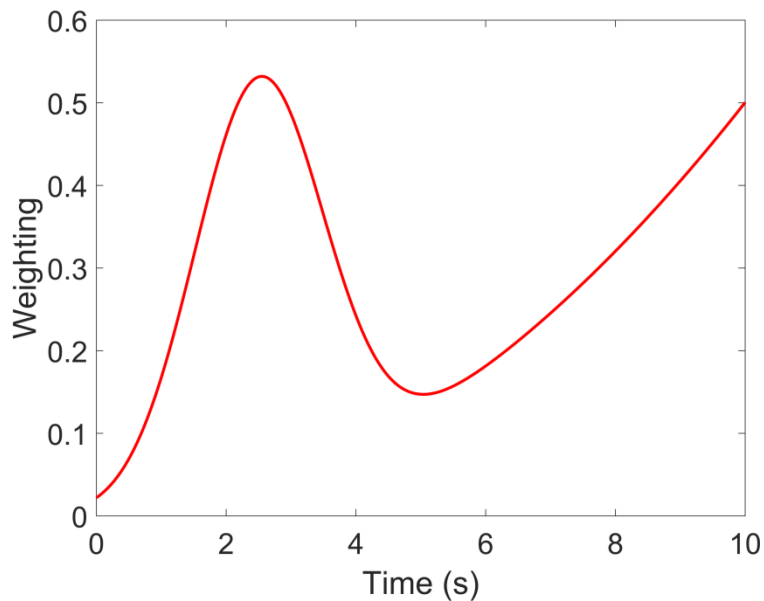


Figure 5-13 Shape of the weighting function for the Gaussian and Time based fitness function.

The responses of each of the joints using this fitness function are shown in Figure 5-14. This function had a better performance in terms of optimising the gains, but still was unable to match the performance of the Sum Squared Error method presented first.

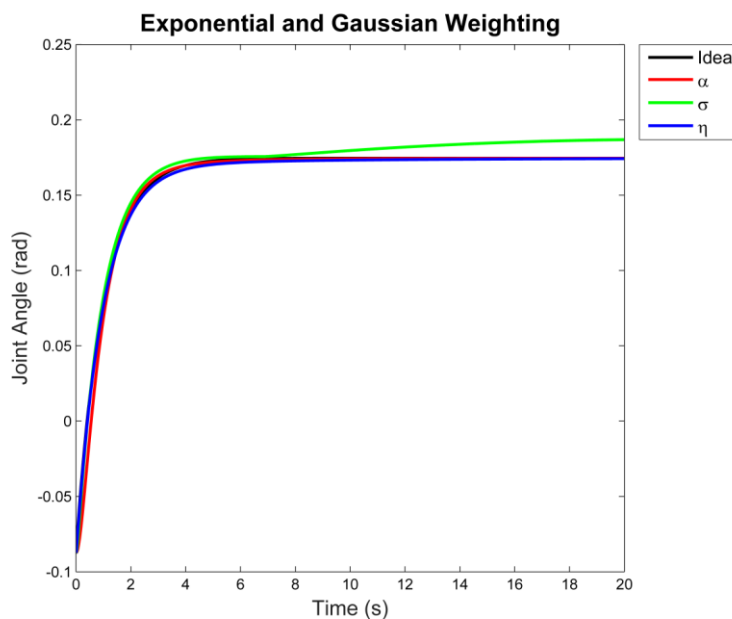


Figure 5-14 Result of the optimisation of a joint angle range in the arm using the Gaussian and Time Base Weighting fitness function.

This is due to the inclusion of parameters which attempt to specify the rise, peak and settling times of the responses. Given that the optimisation is attempting to tune the gains of three PID controllers, any change in gains will change pole locations of the linearized version of the system, hence changing the rise, peak and settling times. This means that any prediction of the location of these points will only ever be an estimate, thus the weighting for each phase of the response will overlap into other phases. Therefore the Sum Squared Error fitness function will be used to carry out the optimisation of the PID gains.

5.2 Genetic Algorithm Validation

Having developed a Genetic Algorithm to optimize the PID gains in the robotic manipulator servo controllers, the algorithm should be validated to test its effectiveness at solving the problem. This can be done by comparing it against the performance of the other optimization methods established as options at the end of the literature review in this chapter. The validation of the developed GA takes two forms. The first is a comparison of its performance against the other methods for standard optimization problems, and the second is a comparison of its performance against the other methods on the PID gain tuning problem for the robotic manipulator dynamic model developed in Chapter 4.

5.2.1 Validation of the GA Using Standard Optimisation Problems

Having implemented a genetic algorithm which has been designed for the tuning of a nine gain 3-DoF manipulator arm PID controller, it is important to validate the effectiveness of the GA against other optimisation methods for both the problem in question and other, well known optimisation problems. This will give a benchmark for the feasibility of use of the GA for the robotic manipulator gain scheduling problem. To carry this out the GA is modified to optimise a list of different functions (Back, 1995), (Haupt & Ellen, 2004), (Deb, 2002), (Binh & Korn, 1997), (Binh, 1999), (Simionescu, 2014). The functions displayed in Table 5-3 are a small selection of those used, and the remainder are found in the appendices. Those functions labelled in blue are single objective problems and those labelled in pink are multi-objective problems.

Table 5-3 List of standard optimisation problems used to test optimisation functions.

Function Name	Function	Search Domain
Rosenbrock function	$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$-\infty \leq x_i \leq \infty,$ $1 \leq i \leq n$

Beale's function	$f(x, y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$	$-4.5 \leq x, y \leq 4.5$
Easom Function	$f(x, y) = -\cos(x) \cos(y) e^{-((x-\pi)^2 + (y-\pi)^2)}$	$-100 \leq x, y \leq 100$
Cross-in-tray Function	$f(x, y) = -0.0001 \left(\sin(x) \sin(y) e^{\left 100 - \frac{\sqrt{x^2 + y^2}}{\pi} \right } + 1 \right)^{0.1}$	$-10 \leq x, y \leq 10$
Eggholder Function	$f(x, y) = -(y + 47) \sin \left(\sqrt{\left y + \frac{x}{2} + 47 \right } \right) - x \sin \left(\sqrt{ x - (y + 47) } \right)$	$-512 \leq x, y \leq 512$
Hölder table Function	$f(x, y) = - \left \sin(x) \cos(y) e^{\left -1 - \frac{\sqrt{x^2 + y^2}}{\pi} \right } \right $	$-10 \leq x, y \leq 10$
Kursawe Function	$\min \begin{cases} f_1(x) = \sum_{i=1}^2 -10e^{-0.2\sqrt{x_i^2 + x_{i+1}^2}} \\ f_2(x) = \sum_{i=1}^3 x_i ^{0.8} + 5 \sin(x_i^3) \end{cases}$	$-5 \leq x_i \leq 5,$ $1 \leq i \leq 3$
Schaffer Function No. 2	$\min \begin{cases} f_1(x) = \begin{cases} -x, & \text{if } x \leq 1 \\ x - 2, & \text{if } 1 < x \leq 3 \\ 4 - x, & \text{if } 3 < x \leq 4 \\ x - 4, & \text{if } x > 4 \end{cases} \\ f_2(x) = (x - 5)^2 \end{cases}$	$-5 \leq x \leq 10$
Poloni's Two Objective Function	$\min \begin{cases} f_1(x, y) = 1 + (A_1 - B_1(x, y))^2 + (A_2 - B_2(x, y))^2 \\ f_2(x, y) = (x + 3)^2 + (y + 1)^2 \end{cases}$ <p style="text-align: center;"><i>where,</i></p> $\begin{cases} A_1 = 0.5 \sin(1) - 2 \cos(1) + \sin(2) - 1.5 \cos(2) \\ A_2 = 1.5 \sin(1) - \cos(1) + 2 \sin(2) - 0.5 \cos(2) \\ B_1(x, y) = 0.5 \sin x - 2 \cos x + \sin y - 1.5 \cos y \\ B_2(x, y) = 1.5 \sin x - \cos x + 2 \sin y - 0.5 \cos y \end{cases}$	$-\pi \leq x, y \leq \pi$

Zitzler-Deb-
Thiele's
Function
No.1

$$\min \begin{cases} f_1(x) = x_1 & 0 \leq x_i \leq 1, \\ f_2(x) = g(x)h(f_1(x), g(x)) & \\ g(x) = 1 + \frac{9}{29} \sum_{i=2}^{30} x_i & 1 \leq i \leq 30 \\ h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} \end{cases}$$

Zitzler-Deb-
Thiele's
Function
No.2

$$\min \begin{cases} f_1(x) = x_1 & 0 \leq x_i \leq 1, \\ f_2(x) = g(x)h(f_1(x), g(x)) & \\ g(x) = 1 + \frac{9}{29} \sum_{i=2}^{30} x_i & 1 \leq i \leq 30 \\ h(f_1(x), g(x)) = 1 - \left(\frac{f_1(x)}{g(x)}\right)^2 \end{cases}$$

Zitzler-Deb-
Thiele's
Function
No.3

$$\min \begin{cases} f_1(x) = x_1 & 0 \leq x_i \leq 1, \\ f_2(x) = g(x)h(f_1(x), g(x)) & \\ g(x) = 1 + \frac{9}{29} \sum_{i=2}^{30} x_i & 1 \leq i \leq 30 \\ h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} - \left(\frac{f_1(x)}{g(x)}\right) \sin(10\pi) \end{cases}$$

Zitzler-Deb-
Thiele's
Function
No.4

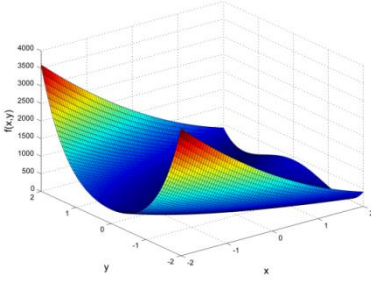
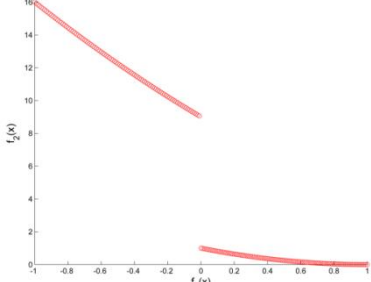
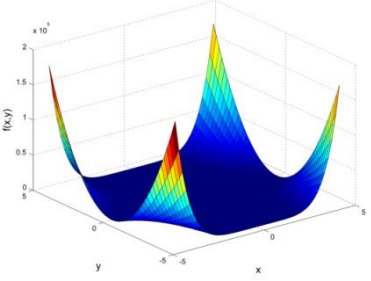
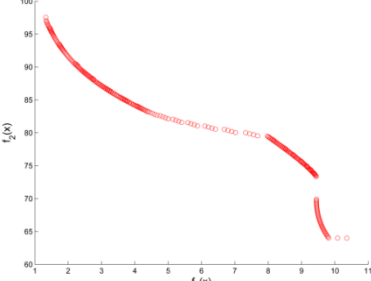
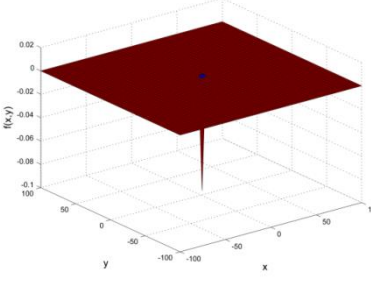
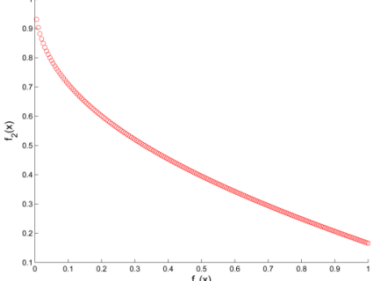
$$\min \begin{cases} f_1(x) = x_1 & 0 \leq x_1 \leq 1, \\ f_2(x) = g(x)h(f_1(x), g(x)) & \\ g(x) = 91 + \sum_{i=2}^{10} (x_i^2 - 10 \cos 4\pi x_i) & -5 \leq x_i \leq 5, \\ h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} & 2 \leq i \leq 10 \end{cases}$$

Zitzler-Deb-
Thiele's
Function
No.6

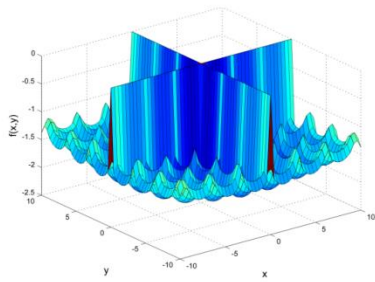
$$\min \begin{cases} f_1(x) = 1 - e^{-4x_1} \sin^6(6\pi x_1) & 0 \leq x_i \leq 1, \\ f_2(x) = g(x)h(f_1(x), g(x)) & \\ g(x) = 1 + 9 \left(\frac{\sum_{i=2}^{10} x_i}{9}\right)^{0.25} & 1 \leq i \leq 10 \\ h(f_1(x), g(x)) = 1 - \left(\frac{f_1(x)}{g(x)}\right)^2 \end{cases}$$

The 14 different optimisation problems displayed here, along with the rest which are found in the appendices have been used to compare the optimisation methods, since they all have uses in testing the performance of each method. Some of them have similar properties therefore the optimisation methods should behave similarly with them, but to illustrate why these functions have been used, several of them have been described in detail. Table 5-4 presents the surfaces of each of the optimisation problems presented in Table 5-3.

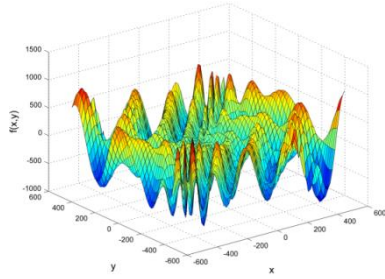
Table 5-4 Graphical representation of the surfaces generated by the optimisation problems listed in Table 5-3.

Function Name	Function Surface	Function Name	Function Surface
Rosenbrock function		Schaffer Function No. 2	
Beale's function		Poloni's Two Objective Function	
Easom Function		Zitzler-Deb-Thiele's Function No.1	

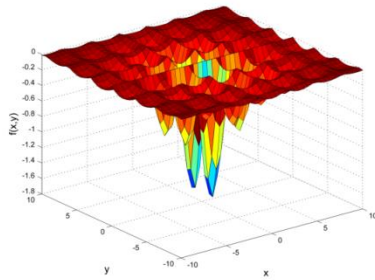
Cross-in-tray Function



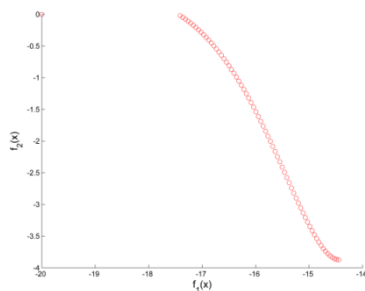
Eggholder Function



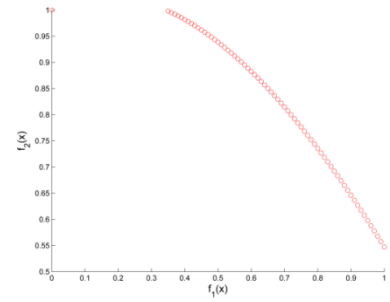
Hölder table Function



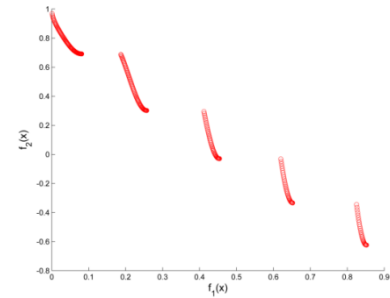
Kursawe Function



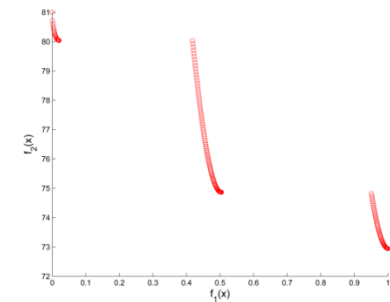
Zitzler-Deb-Thiele's Function No.2



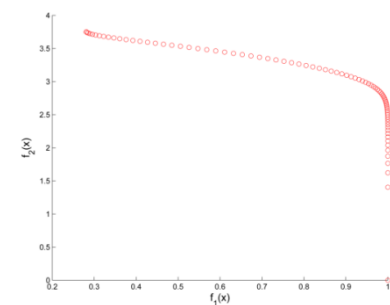
Zitzler-Deb-Thiele's Function No.3



Zitzler-Deb-Thiele's Function No.4



Zitzler-Deb-Thiele's Function No.6



The Rosenbrock Function and Beale's Function have been used since they have very steep peaks for the optimisation functions to diverge away from, but the global minimum lies on a flat plain or valley, which makes it difficult for an optimiser to converge on a single point. The Easom Function has been used since it is a function

which has a flat plain with a single inverse peak, which could be difficult to find depending on how well the optimiser is able to search a flat area and spread out.

The Cross-in-Tray Function provides a surface with many local minima, the surface of which is intersected by a cross shaped ridge. This means that the optimiser can easily converge on one local minimum, but it may not be the global minimum and the optimisers may struggle to traverse the cross shaped ridge. The Eggholder Function has lots of local minima and is good to test whether the optimisers can find the global minimum among the set of local minima.

The Hölder Table Function is a very interesting optimisation problem as it consists of a plateau containing many local minima and has four groups of inverse peaks in the XY corners of the space. These groups of inverse peaks contain the global minima but also several large local minima which could be difficult to differentiate from the global minima. Since the majority of space in this problem is taken up by a large plateau of shallow local minima, it could be very difficult to find the large inverse peaks in the corners of the space.

The Kusawe Function, Schaffer Number 2 Function, Zitzler-Deb-Thiele Function and Poloni's Function are all multi-objective functions with a general trend of decreasing Y values with increasing X values. There exists discontinuities in the output of these functions which means that there are several local minima in the space with no output beyond them for a distance in both the X and Y directions. This means that the optimisation functions will output one of the local minima as the solution unless they can traverse the spaces where discontinuities occur.

Three other optimisers were chosen to provide a benchmark in performance to compare the implemented GA against. The chosen optimisation methods were Least Squares Optimisation and a comparable Genetic Algorithm, both of which exist in the Matlab optimisation toolbox. In the initial investigation all four optimisation methods were run on each of the above problems with the same fitness function and the fitness they achieved, along with the run time of the each on the same PC is compared for each problem.

For each of the above optimisation methods, the fitness function used to minimise the output of each of the test functions was the mean-squared value of the outputs to the function. This fitness function is given in Equation (5.6).

$$f = \frac{\sum Y^2}{n}, \tag{5.6}$$

where $Y = \{y_1, \dots, y_n\}$,

Where, Y is the set of outputs from the function and F is the fitness. The results of the optimisation of all of these problems using the GA presented in this chapter, the GA implementation in MATLAB and the Least Squares Minimisation implementation in MATLAB are shown in Figure 5-15 and Figure 5-16. Figure 5-15 presents the results with fitness and run time as linear values and Figure 5-16 presents the results with fitness and run time as logarithmic values in order to provide a more visible comparison between techniques.

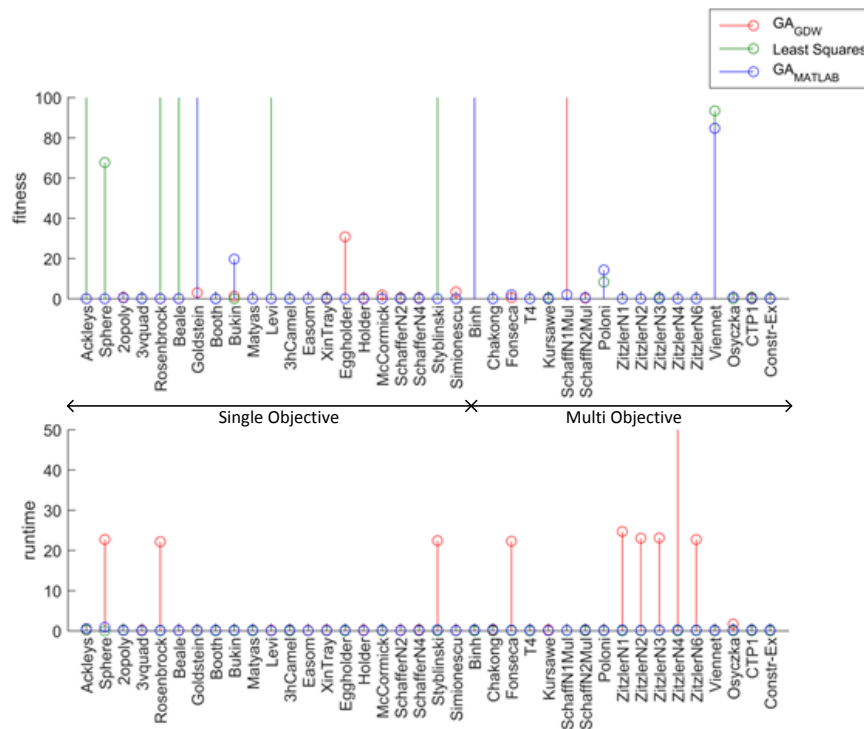


Figure 5-15 Linear results for achieved fitness and runtime for three optimisation methods over all of the optimisation problems.

These two figures present the results both with linear scales for fitness and runtime on the y-axis and also with logarithmic scales for fitness and runtime. This is because for some problems, there were cases where the fitness values produced by one or more optimisation methods, or the time it took for the optimisation to be run, were found to be values on the order of 10^3 seconds for time, and the fitness values were found to range between the order of 10^{-204} and 10^9 in magnitude. The lines on these plots are not used to designate trends but to aid visibility of each of the points on the figures. There are also places on the logarithmic plots where the line is broken, indicating points which have not been plotted on the figure. This is because those values have a magnitude of $-\infty$. This occurs when the value of fitness or time is 0, since $\log_{10} 0 = -\infty$.

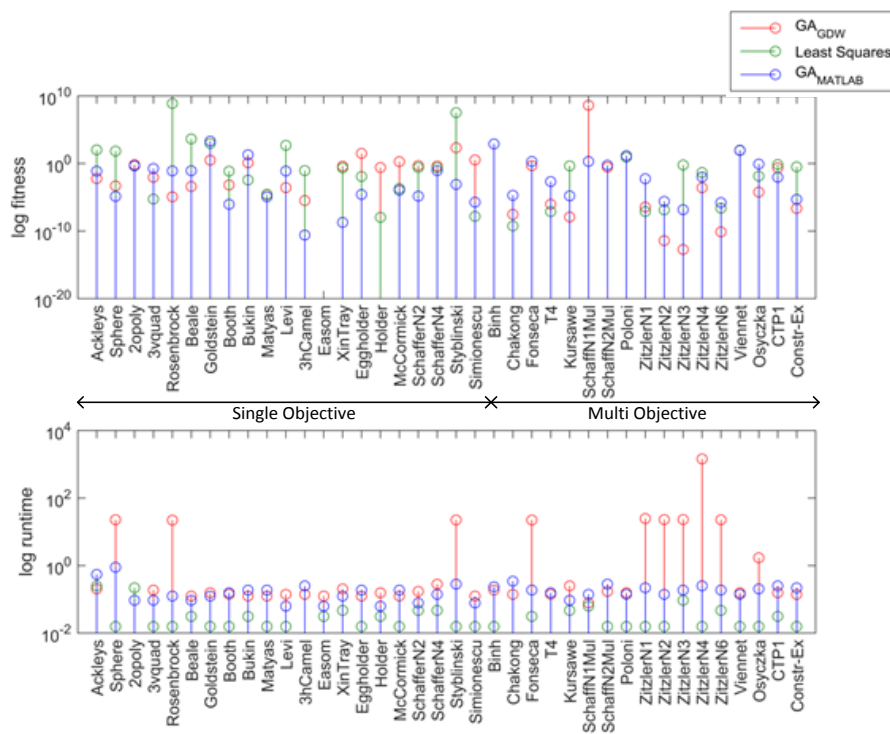


Figure 5-16 Logarithmic results for achieved fitness and runtime for three optimisation methods over all of the optimisation problems.

It can be observed from the logarithmic figure that over the range of different optimisation problems, the optimisation method which achieves the best fitness is

varied, but the method which takes the least amount of time to converge on a solution is consistently the least squares method, however, the least squares method also tends to be the method which provides the worst fitness result. The simplest reason for this fast convergence on a value which is clearly not the global minima is that the method is getting stuck in local minima. A quick inspection of the shape of each of the optimisation problems shows that the problems whereby this method displays a fast run time with a large fitness value, the problems that are being solved for have lots of local minima very close to the global minima. This indicates that the method very quickly converges on the minimum region in the search domain of the problem, but then is very easily satisfied by the fitness values that it achieves as it approaches the global minima and does not continue to search.

With regards to the two Genetic Algorithm methods, the performance is varied. There are some problems whereby the two different methods perform similarly in terms of run time, fitness or both, and some problems whereby the performance of one of the methods is better than the other by a large margin. The better performing algorithm is not the same one consistently, which indicates that both methods have strengths and weaknesses, allowing them to perform better in solving different problems. Again, an inspection of the shape of the search domain of the problems in which each performs better shows that the GA built into MATLAB performed better than the implemented method in problems where the global minimum of the problem existed in a very small range of the search domain and the area around the global minimum had very large changes in values, whereas the implemented method achieved better fitness values where the search domain had very small changes in the range around the global minimum and a lot of the search domain had a large area of very low values surrounding the global minimum. In these cases the implemented method took a long time to converge on the solution however. Since both GA methods were instructed to find a solution which satisfies a certain fitness value within a set number of generations, which was 300 generations, and were instructed to maintain the same population size over generations, a population of 50, it is clear that the rate of convergence on the global minimum with respect to increase in generations for the MATLAB GA is smaller than that of the implemented GA. This meant that it took more generations to achieve a comparable fitness for the

Matlab GA to that of the implemented GA. However, the implemented GA took longer to carry out each generation in terms of time.

Since the PID controller gain tuning of the robotic manipulator is a multi-objective problem, it is appropriated to investigate more closely the performance of each optimisation method for the multi-objective optimisation problems. In these cases, with the exception of one optimisation problem, the Schaffer Number 1 multi-objective problem, where the implemented GA displays the worst performance, it consistently matches or outperforms the other methods for fitness value. Since the PID gains are tuned off-line, the longer run time for this method when optimising these problems is not considered an issue.

5.2.2 Comparison of Optimisation Methods on the Robotic Arm Tuning Problem

Having compared these optimisation methods against each other for a series of different problems and assessed them as being comparable, it is also important to investigate their performance against each other for tuning the robotic arm gains for a single scenario. To directly assess the feasibility of using the implemented GA to provide a set of solutions for the PID controller gains required to operate the robotic manipulator arm, each of the aforementioned optimisation methods were used to find controller gains for the arm in a single scenario (range of angle motions). This allows for a direct comparison of the run time of each method and the achievable fitness value using the same fitness function, and will determine how well the implemented method behaves in comparison to existing methods.

In order to do this each method will be used to tune the gains to the arm through a motion where joints σ and η are required to move through an angle of 0.175 radians from -0.087 radians to 0.087 radians. The joint angle α will move through an angle of 0.35 radians. The initial estimates for the gains will be randomly generated, but will be identical for each method used. The results are displayed in Table 5-5.

Table 5-5 Numerical results of optimisation method comparison on the robotic manipulator tuning problem for an optimisation with random initial estimated solution.

Optimisation Method	Fitness Value	Run Time (s)	Run Time
Implemented GA	0.0227	13723	≈ 3.8 hrs
MatLab GA	0.0152	12409	≈ 3.4 hrs
Nelder-Mead (Fminsearch)	1.47806	529.3	≈ 8.8 mins
Least Squares	3.1279	51.02	≈ 0.85 mins

As can be observed from the information presented above and in Figure 5-17, over the four optimisation methods used, there is a large variation in fitness and run time in seconds. The least squares optimisation method converges on a solution the fastest in terms of time, with Nelder-Mead Method (Fminsearch) converging on a solution next in time.

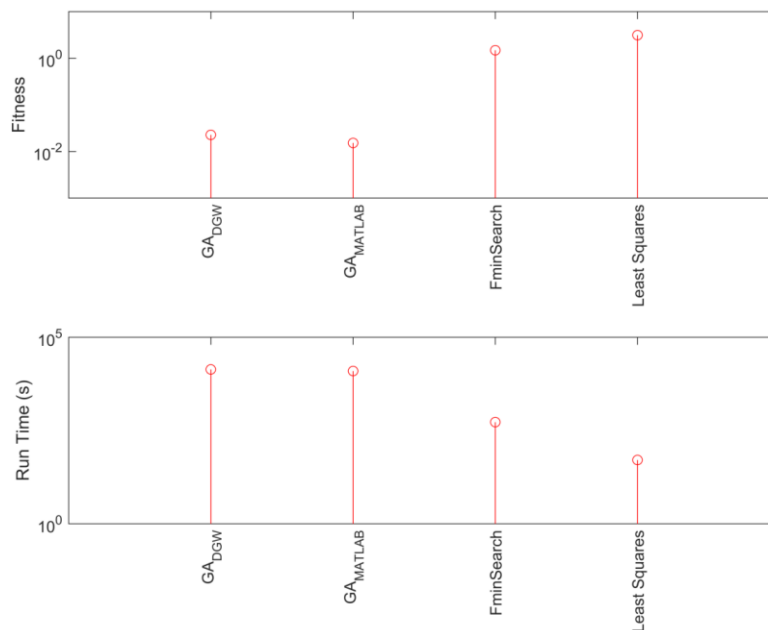


Figure 5-17 Graphical results of optimisation method comparison on the robotic manipulator tuning problem.

Both of the GA optimisation methods display similar performance in terms of time, giving run times in the order of 10^5 seconds per optimisation; however they are able to find solutions with fitnesses in the order of 10^{-2} , whereas the other two optimisation methods are only able to converge on solutions with a fitness value in the order of 10^1 . By using the numerical values rather than the graph the fitness of the two GAs can be broken down into their physical meaning. Since the fitness function takes the mean over the three joints of the sum of the error squared, the following is the case. For the MATLAB GA, the value of 0.0152 means that over the three joints over the entire run time, there was a total error of 0.2135 radians and for the implemented GA, the value of 0.0227 gives a total error of 0.261 radians over all three joints for the entire run time of the system. Given that this is the case, the difference between them in terms of error is very small.

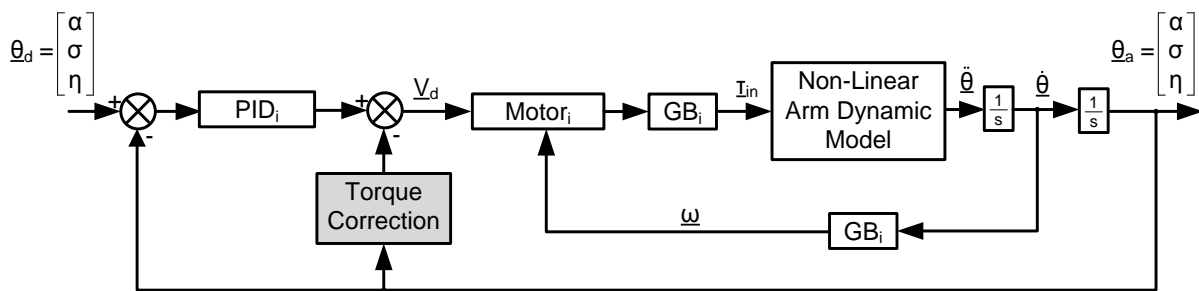
5.3 Use of GA to Optimise Arm Gains

Having validated the performance of the implemented GA, it can be used to optimise the PID gains in the controller for the robotic manipulator arm. Since the moments about the arm vary non-linearly with changing joint angle it is appropriated to tune the PID controller for different ranges of angles. While the time taken to find a solution to this problem is not an issue since the gains only need to be found once and this can be carried out offline, practical time constraints exist. Either of the GA methods will take 10^5 seconds to run one optimisation, which is in the order of a four hours. Given that this is the case, any efforts to minimise the number of discrete angle ranges which are optimised for would be useful in reducing the time taken to carry out the optimisation. To do this the following factors and assumptions are taken into consideration.

5.3.1 Assumptions

The only terms which change their direction relative to the motion of the arm are those concerning gravity. Since there is only one gravitational term in joint η and two

gravitational terms in joint σ , and these relate to the components of weight perpendicular to the two upper links, these terms are only affected by cosines of the respective joint angles and so can be calculated easily. Removing this term during the optimisation and compensating for it by equation with an extra torque input will mean that the optimisation only has to take place for 1 angular direction in each joint, therefore effectively reducing the number of required optimisations by a factor of 4. The architecture change to the servo system by adding this torque correction is shown in Figure 5-18.



Where $i = \{1,2,3\}$ or $i = \{\alpha, \sigma, \eta\}$

Figure 5-18 Control block diagram illustrating the implementation of PID control into the dynamic model of robotic manipulator and servo drive. This block diagram also includes a torque correction factor for moments caused by weight on the arm.

The lowest link in the arm has an effect on those above it which relates to angular velocity, not position, therefore provided that $\dot{\alpha}$ is small, its effect can be considered negligible. This means that the motion of α can be kept constant for every single optimisation.

To determine the range of each discrete angle step, the moment terms which are affected by changing angle are considered. With gravitational terms excluded, this leaves only the moments of inertia of each joint, which change depending on the extension of the links, hence joint angle. Given that these moments of inertia terms contain the cosine of angles, it is useful to consider the values of these cosines. Table 5-6 shows that for the cosine of angles between 0 and 90 degrees, the angle changes the result of the cosine in the order of 10^{-2} until approximately 20° , after which the order of the change increases to the order of 10^{-1} , therefore each angle

range has been selected to be 20° or 0.3491^c , which gives a total variation in value for each of the cosine terms of a maximum of 7.03%.

Table 5-6 Numerical data illustrating the possible change in moment values as a result of angle.

Angle	0°	10°	20°	30°	40°	50°	60°	70°	80°	90°
Cosine	1	0.9848	0.9397	0.866	0.766	0.6428	0.5	0.342	0.1736	0

Another factor which will reduce the number of discrete angle ranges necessary to optimise the PID controller is to consider that operating range of the arm. Assumptions can be made about the accessible range of the arm so that the entire $-\pi$ to π range of each joint does not have to be considered. As such the operating range of the arm is assumed to be as follows:

$$-180^\circ \leq \alpha \leq 180^\circ$$

$$-40^\circ \leq \sigma \leq 180^\circ$$

$$-180^\circ \leq \eta \leq 140^\circ$$

Since the change in angle for each joint per scenario is $\Delta\alpha = \Delta\sigma = \Delta\eta = 20^\circ$, and α will always move through the same 20° , there will be 1 discrete range in α , 11 discrete ranges in σ and 15 discrete ranges in η , therefore a total of 165 scenarios to optimise for.

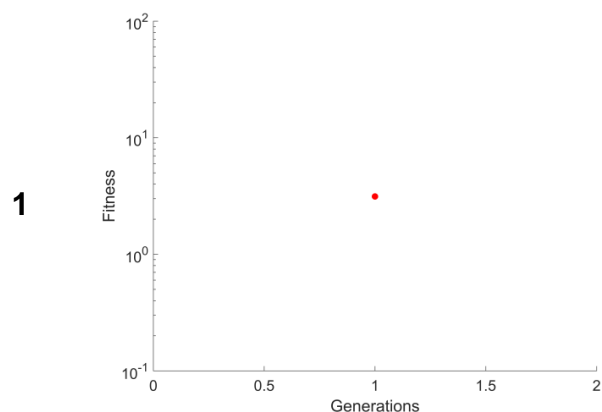
To potentially speed up the time taken to carry out each optimisation, a further assumption will be made. Given that the moments about each joint vary non-linearly, but continuously with angle, the idea that the gains required to control the arm will also vary non-linearly but continuously can be considered. If this is the case then the solution to a previous scenario could be quite close to a solution to the next scenario, and the solution set from the previous scenario should then be used as the initial estimate for the gains to the next scenario, rather than starting from scratch with a new set of random numbers for the PID gains. If this is so then the first scenario will take the longest in terms of the number of required generations and thus the run time, and all scenarios after that will take less time to find a solution for.

5.3.2 Optimisation Process

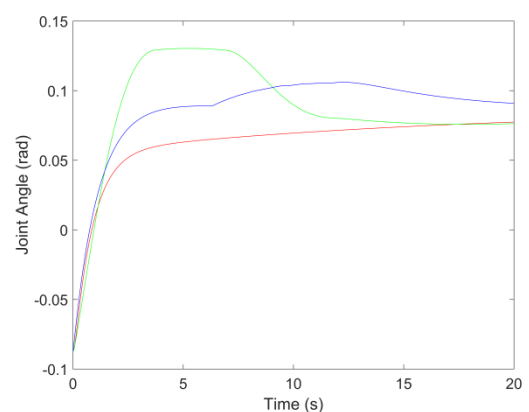
Table 5-7 shows the progression of a the optimisation process for a single scenario, starting with the first generation and showing the optimisation progress every 50 generations to the maximum allows number of generations in this case, which is 300. It can be observed that the fitness decreases exponentially as a function of generation, and converges on a fitness which is in the order of 10^{-2} . As can be seen from the step response of each joint over the generations, the shape of the responses initially starts with overshoot and even oscillation like a higher order system, but the shape of the response gradually gets closer to that of a 1st order step response as the number of generations increases. As the optimisation process drives the system closer to behaving in the desired manner a strange effect begins to appear whereby one or more of the joints display a secondary rise in their joint angle magnitude. This is due to the coupling effects between joints but disappears again as the optimisation process compensates for it by altering the control gains.

Table 5-7 Results of the optimisation process of the manipulator arm for a single angle range displaying the change in best fitness and response to a unit step over successive generations.

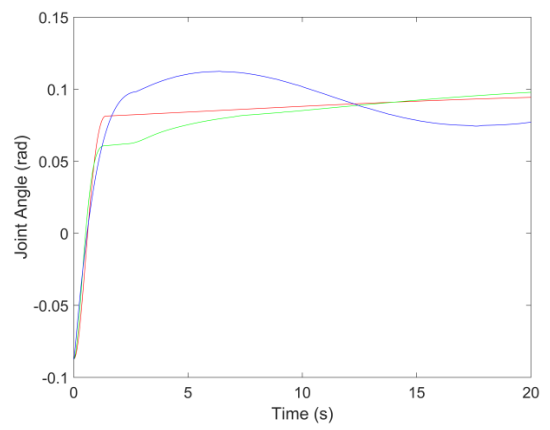
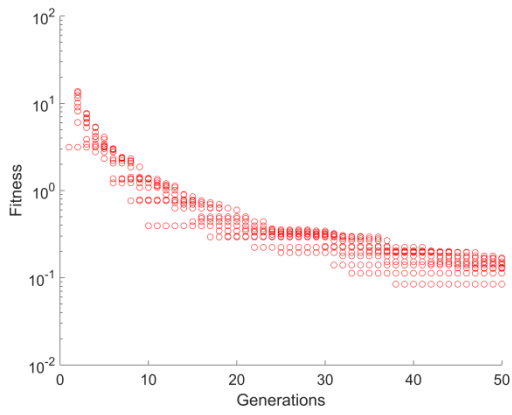
G Fitness vs. Generations



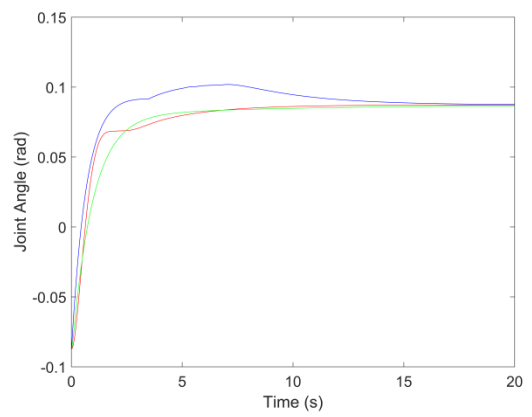
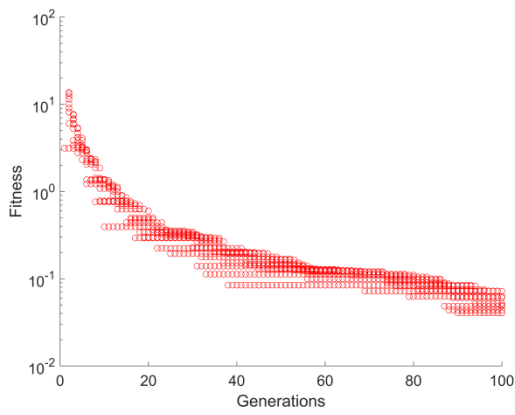
Joint Angles (rad) vs. Time (s)



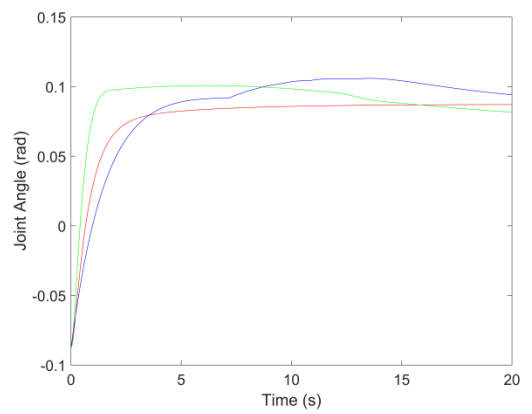
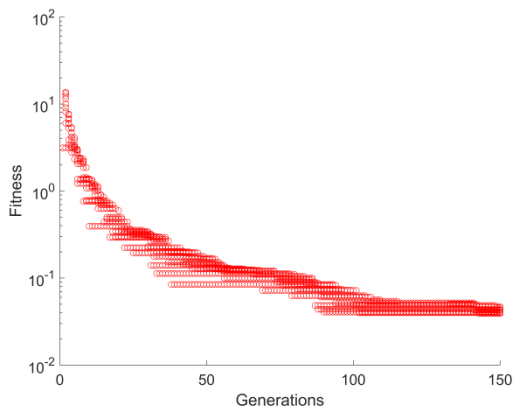
50



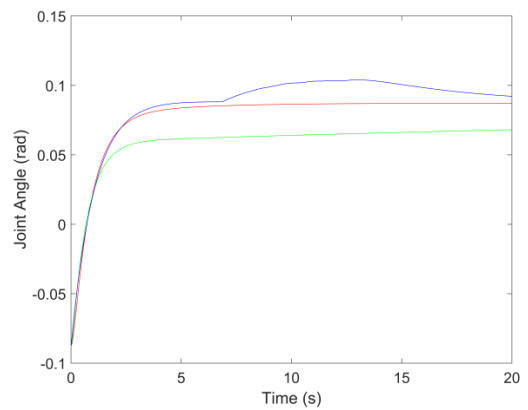
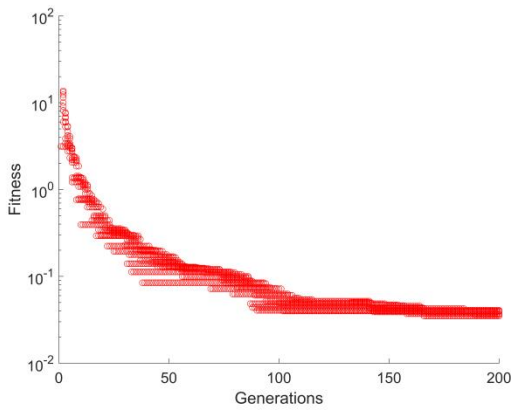
100



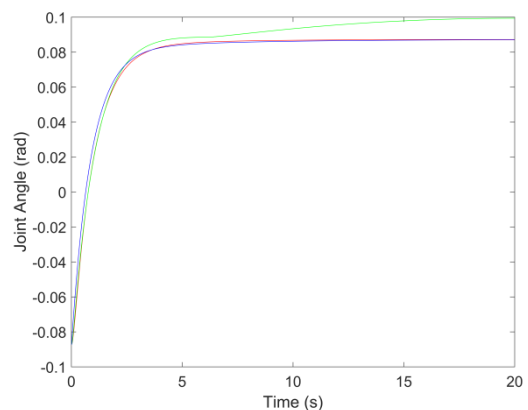
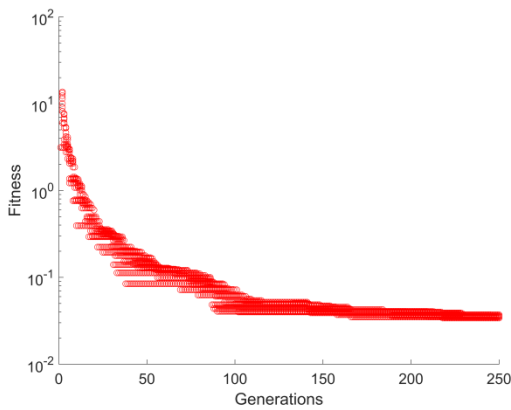
150



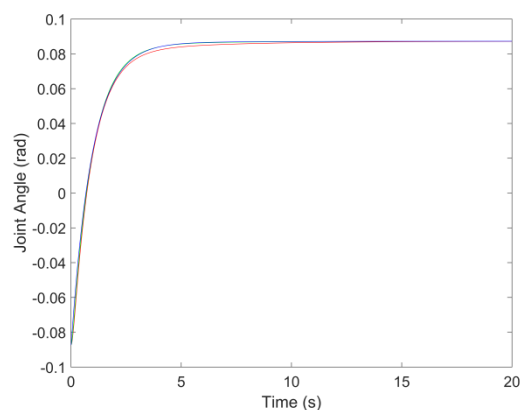
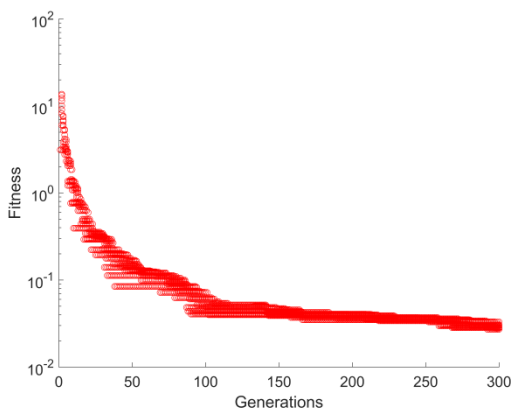
200



250



300



5.4 Gain Scheduler for Robotic Manipulator PID Controller

Having carried out the optimisation, the results which have been obtained are 9 sets of gains, three for each PID controller in the arm. The results as they appear at this point are a series of 9 sets of 165 square surfaces which represent the proportional, integral and derivative gains for the servo controller implemented on each link in each of the scenarios in question. For the gain plateaus $\{link \alpha, link \sigma, link \eta\} =$

$\{(K_{p\alpha}^1, K_{i\alpha}^1, K_{d\alpha}^1), (K_{p\sigma}^1, K_{i\sigma}^1, K_{d\sigma}^1), (K_{p\eta}^1, K_{i\eta}^1, K_{d\eta}^1)\}$. These can be plotted as a set of 9 plots containing plateaus representing the gains in each of these scenarios.

While the gain profiles as a set of plateaus is useful and already distinct shapes are clearly observable in each of the gain profiles for the PID controller, these gain plateaus are discontinuous by virtue of being obtained in a discontinuous manner. Using MATLABs curve fitting toolbox, these surfaces can be converted to splines which can then be used to implement gain schedulers for the PID controller. To do this the 'centre of gravity' of each of the planes in the profiles displayed in the coming pages was taken to determine an approximate point at which each of the gains could be assigned. The resultant data sets were passed through the curve fitting toolbox to produce a series of continuous surfaces which are presented in the following figures. In this form it is clearly visible that the gains are continuous since the surfaces obtained from the optimisation have produced a series of correlated shapes. In the case of the gain surfaces

$$\{link \alpha, link \sigma, link \eta\} = \{(K_{p\alpha}^2, K_{i\alpha}^2, K_{d\alpha}^2), (K_{p\sigma}^2, K_{i\sigma}^2, K_{d\sigma}^2), (K_{p\eta}^2, K_{i\eta}^2, K_{d\eta}^2)\}.$$

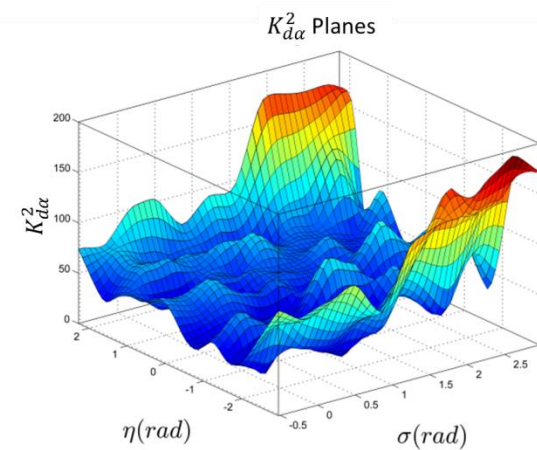
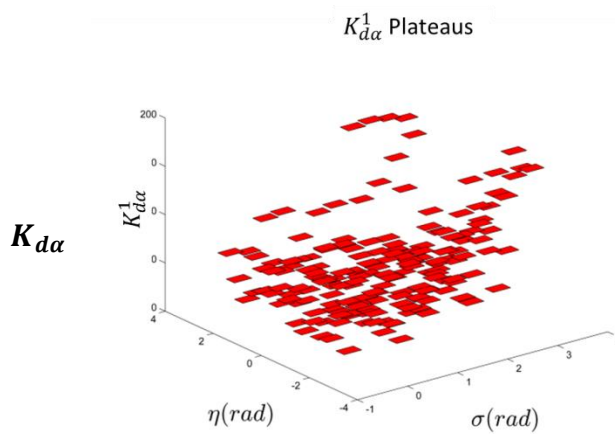
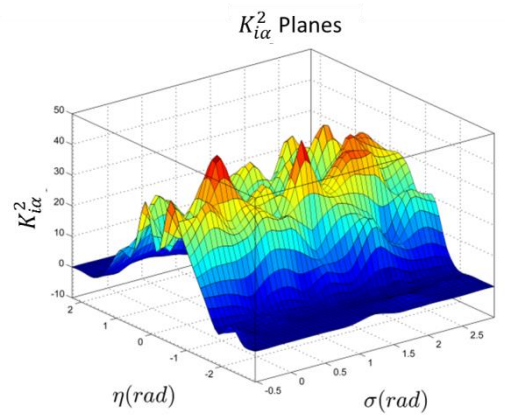
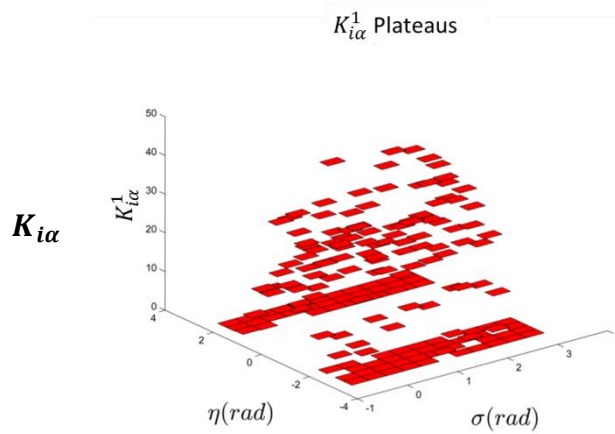
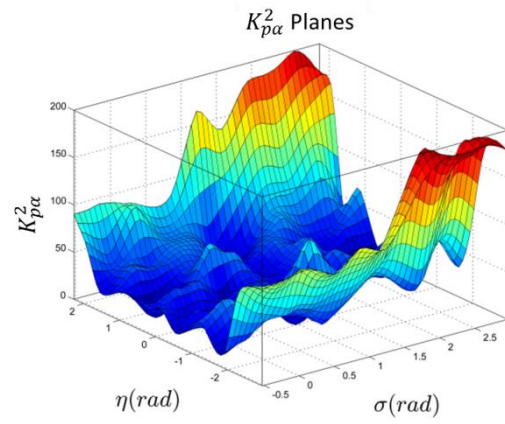
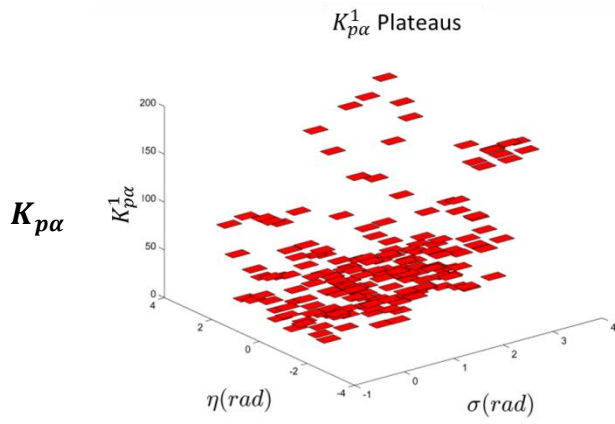
Table 5-8 shows the gain profile for each term in the set of controllers as both a profile of gain plateaus and also a profile as a single continuous surface. A comparison of these gain profiles against the changes in moments about joint α with varying σ and η can be carried out in order to investigate whether the gain profiles have any relationship to these moments. Given that each gain plateau found during the optimisation process was obtained for an angle range of the same size, and the motion for each was designed to be identical, all of the terms in the dynamic model which relate to $\dot{\alpha}$ and $\ddot{\alpha}$ will behave in the same way for each angle range in the set of gain profiles. With this being the case, the gains will only be compared against moments about each joint which are affected by α . A modification has already been made to remove the need to compensate for changes in moments due to weight with joint angle. This means that the only moment which is directly affected by angle is the moment of inertia, therefore only the effect of angle on moments of inertia has been investigated.

Table 5-8 Gain Plateaus and Surfaces for the PID controller in joint α .

K

Gain Plateaus

Gain Surfaces



Moments of Inertia about Joint α

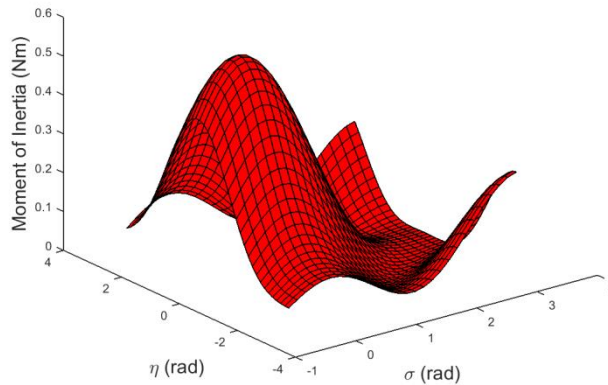


Figure 5-19 Moments of inertia about joint α given changes in σ and η .

As would be expected, Figure 5-19 shows that the moments of inertia are largest when the manipulator arm is fully extended, i.e. σ and η are 0 radians. All of the angle combinations which result in a large moment of inertia involve extension of the mass of the arm away from the axis of rotation about α . This provides three peaks in the figure where the manipulator is extended. These geometries are shown in Figure 5-20.

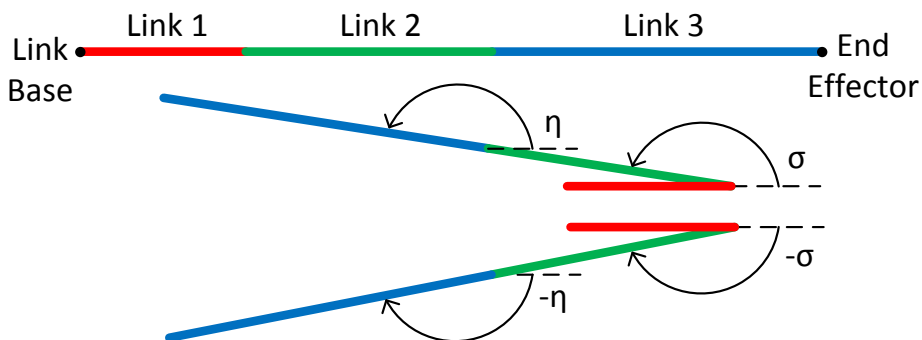


Figure 5-20 Manipulator arm geometries which generate the largest moments of inertia about α .

Investigation of the gain profiles from Table 5-8 shows that the $K_{\alpha p}^2$ and $K_{i\alpha}^2$ gains partially reflect this relationship. In the angle ranges of 2 to π radians and -2 to $-\pi$ radians, this relationship appears to be reflected in the gains, but the large peak at σ

and η of 0 radians is missing from these gain profiles. In Figure 5-21, which shows all three sets of optimised PID control gains for the joint α , it can be observed that for the proportional gain $K_{p\alpha}$ and derivative gain $K_{d\alpha}$, the trend is of a low gain for low σ and η angles, and increasing gain as these two angles increase.

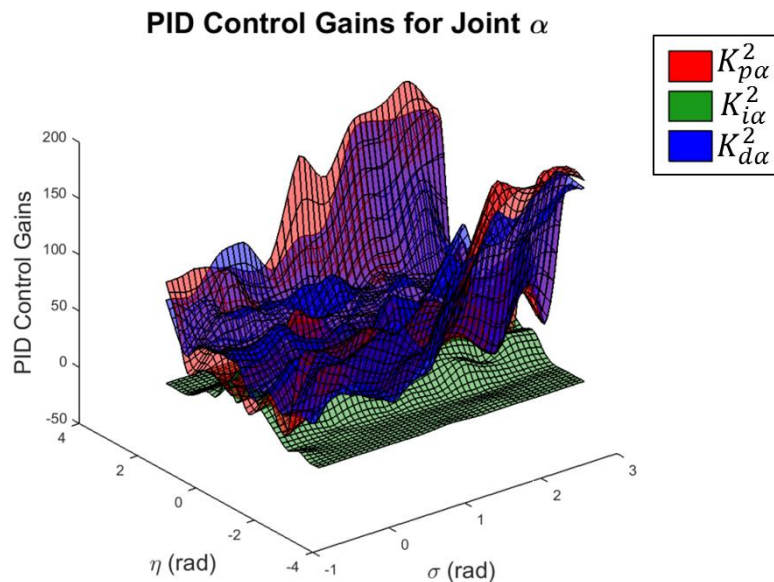


Figure 5-21 PID Control gains for joint α overlaid.

The expected result would be that for σ and η joint angles where the arm has less extension in the XY plane, and therefore joint α would experience lower moments of inertia generated by the 2nd and 3rd links in the arm. For α and σ values where there is a larger extension of the manipulator in the XY plane, the higher proportional gain would be expected to reflect the larger moments of inertia experienced by joint α as a direct consequence of a longer arm extension. The same would be expected for the derivative gain. When the arm is extended, there is a greater moment of inertia about joint α , which means that the joint requires more energy to slow down or reverse the motion of the arm rotation about α , and this would lead to a larger overshoot if uncompensated. Since derivative gain affects the damping of the system, hence reducing overshoot, it makes sense that the optimisation would produce a larger derivative gains at arm geometries of higher extension. This appears to be the case for the larger angles, where σ and η are both in the range of

2 to π or -2 to $-\pi$ radians, but not for the range of joint angles for around 0 radians for both σ and η . An inspection of the sum of all three PID gains in comparison with the moments of inertia about α in Figure 5-22 may provide extra information.

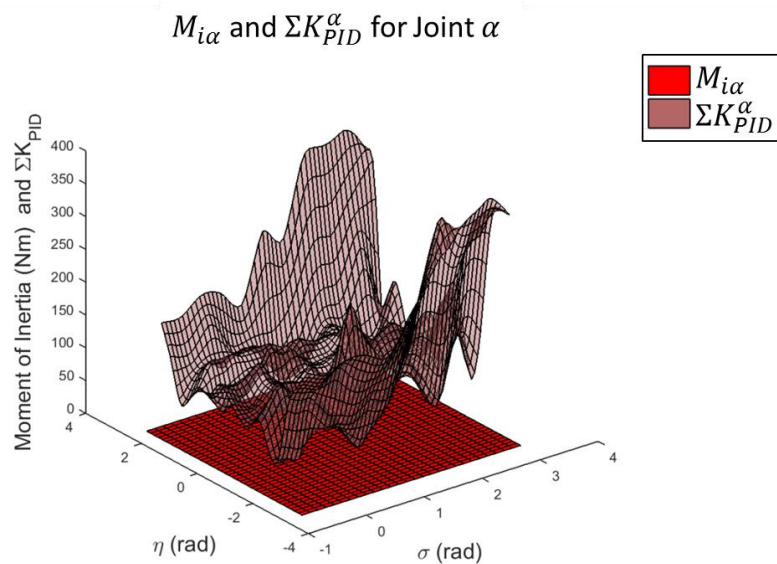


Figure 5-22 Moments of inertia and PID gains for α .

It can be seen from this figure that the sum of the three gain profiles is significantly larger than the moments of inertia about α . This shows that the relationship between proportional and derivative gain and moments of inertia is not as clear as expected. The magnitude of the gains means that any need for proportional and derivative gains in order to provide the required response is satisfied across the whole range of σ and η . The other trend that is observable in these results is the opposite trend for K_i in the gain profile. Since the integral gain is present predominantly when the other two gains are not, this implies a limit on the gains that is reached to stop the integral gain from being used. Since there is a saturation on the servo motors of $\pm 55 V$, the gains in the PID controller will amplify the input signal and must be limited in some way in order to prevent them from causing the input to the servo from exceeding this saturation. Given that proportional and derivative gains are required in order to accelerate and decelerate the arm, these two gains are more likely to be larger since they are needed to control the dynamics of the system. The integral

gain is responsible for reducing the error signal over time, therefore an integration of a persistent error over a long period of time may result in a very large input voltage which could exceed the saturation limit, therefore the optimisation will provide a much smaller integral gain than the other two, and this gain will be reduced significantly in the regions of the gain profile where there is a need for proportional and derivative gain.

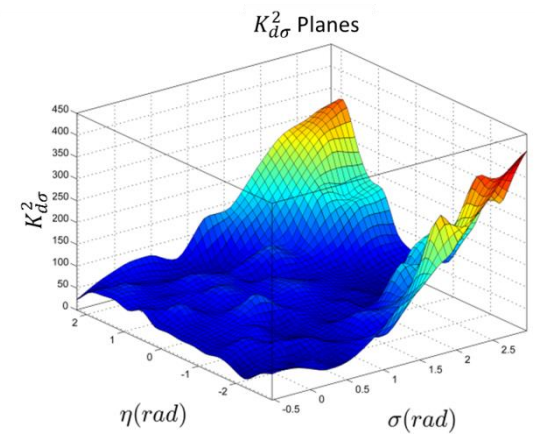
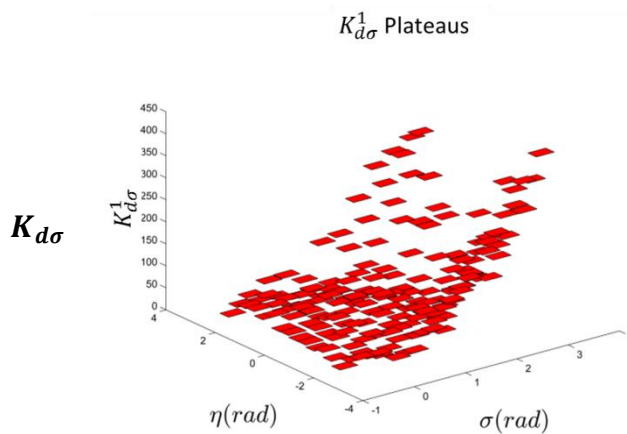
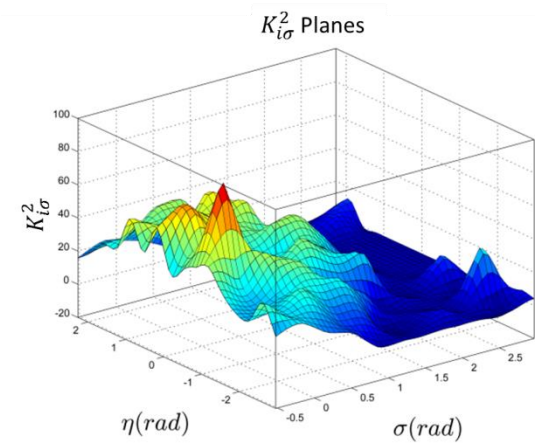
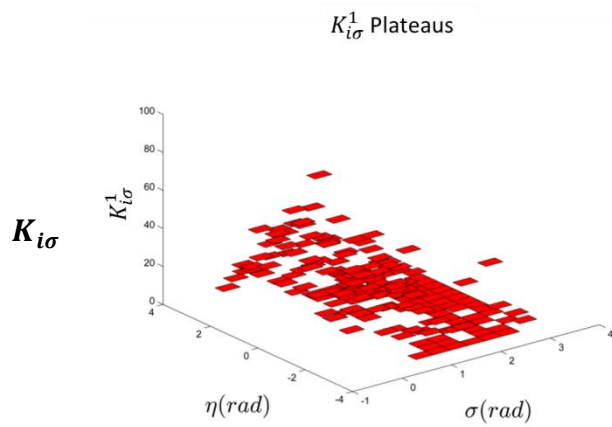
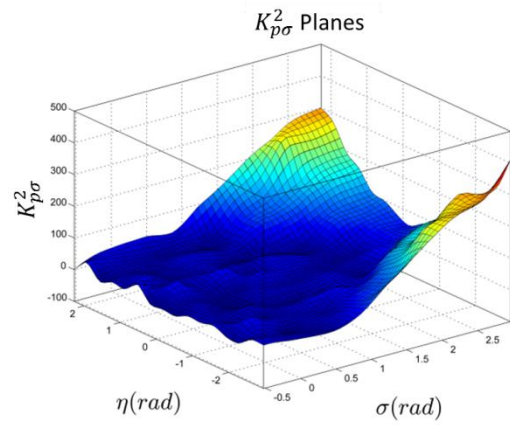
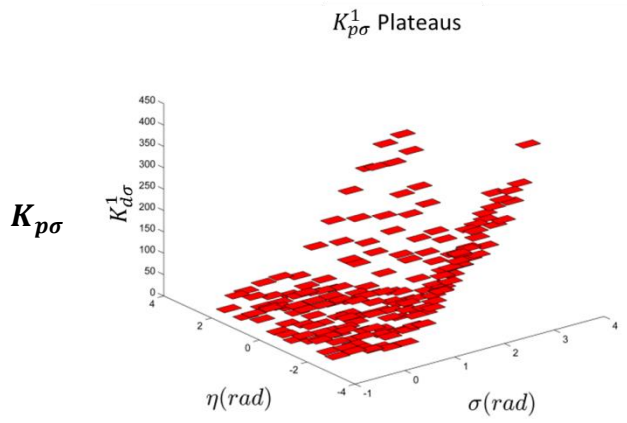
Table 5-9 presents the gain profiles for joint σ . By observing the PID control gains for joint σ , it can be seen that the same set of trends is apparent as was the case in joint α . This illustrates that the optimiser has tuned the PID gains for joint σ with the joint experiencing a similar pattern of moments and torques about the joint over the same range of joint angle combinations. The one change in the pattern in this case is that the integral gain dies down with larger values of σ . Figure 5-23 (c) shows that the valley in the sum of gains which occurs between the two peaks at the $\pm 2 \leftrightarrow \pm \pi$ radian ranges has a greater magnitude than the plain in the region of 0 radians. This indicates that the limit of saturation on the voltage input to the servo would be exceeded for a larger integral gain in this region. It can be seen that the gain profiles are being forced to the shape which is seen in these figures by another driver other than the moments of inertia about the joints. In the case of σ the moments of inertia display a completely different shape to that of the PID control gain pattern, but again the magnitude of the gain profiles is significantly larger than that of the moments of inertia.

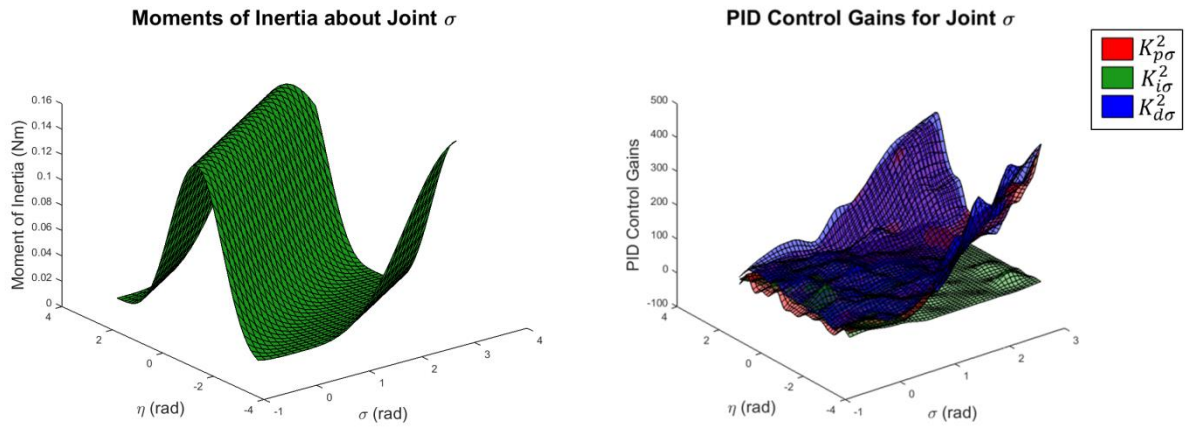
Table 5-9 Gain Plateaus and Surfaces for the PID controller in joint σ .

K

Gain Plateaus

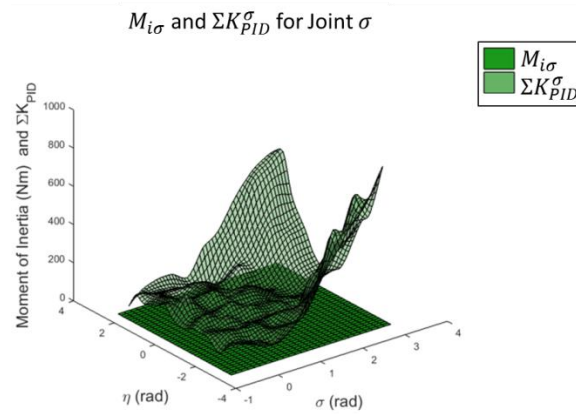
Gain Surfaces





(a) Moments of inertia about σ .

(b) PID control gains for σ .

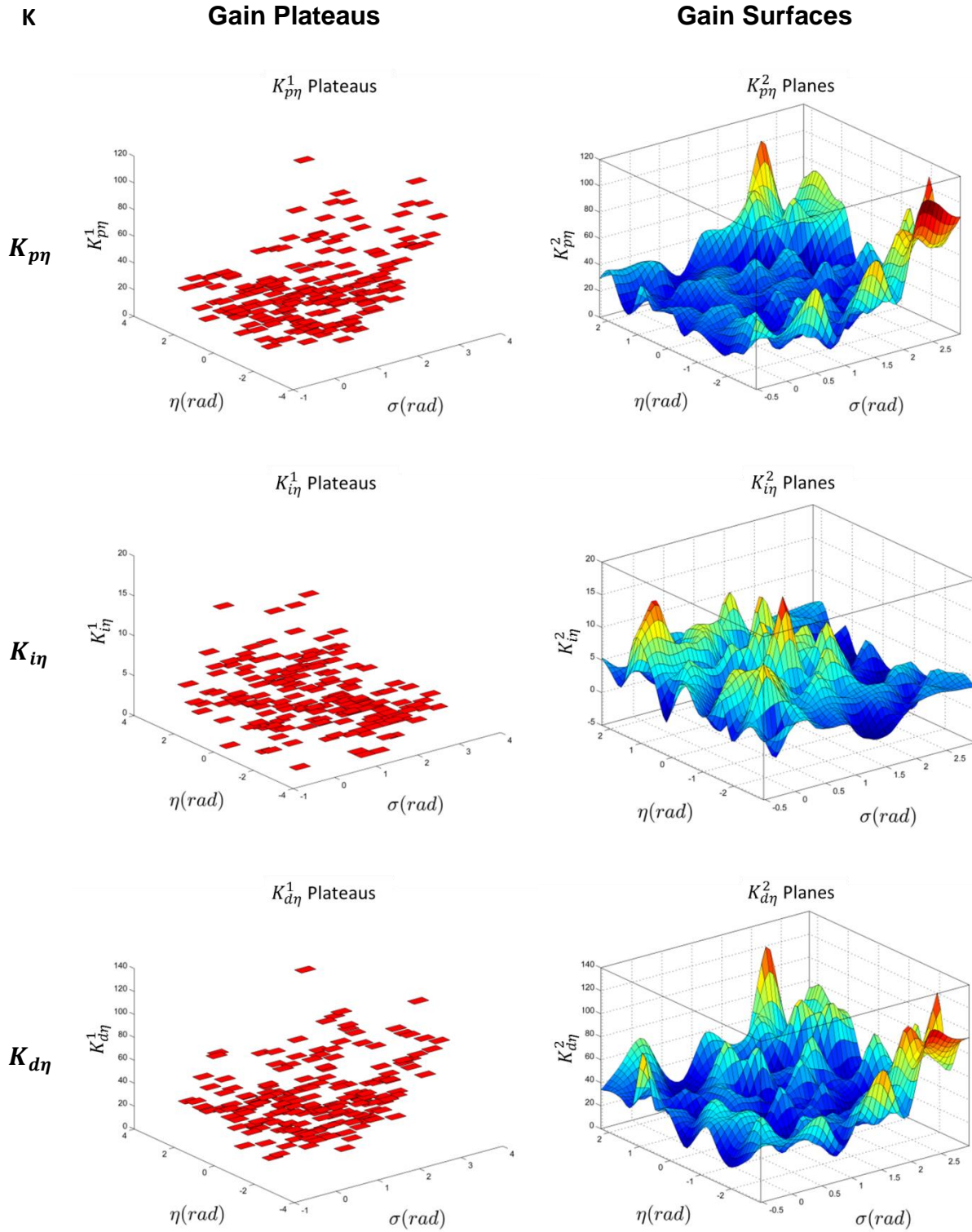


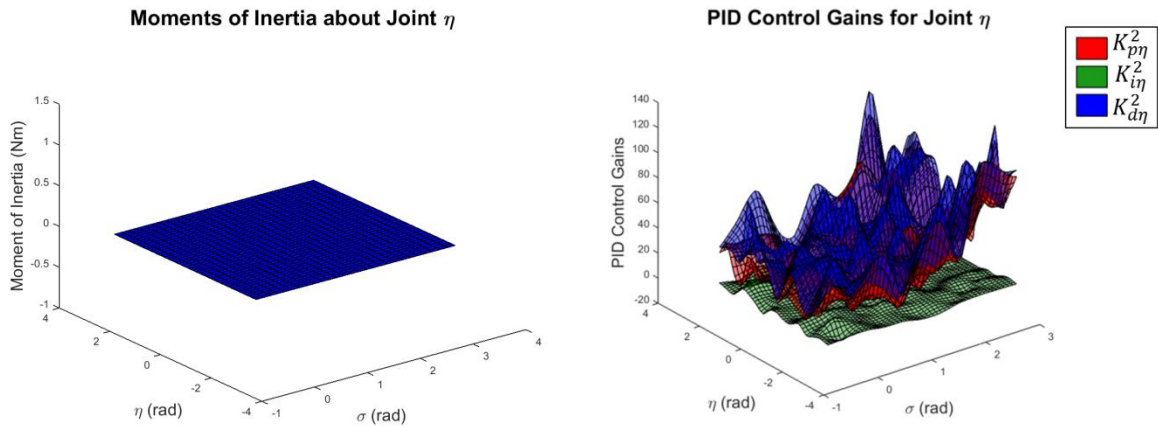
(c) Moments of inertia and sum of PID gains for σ .

Figure 5-23 Comparison of moments of inertia of joint σ with the PID control gains for the same joint.

Figure 5-10 presents the gain profiles for joint η .

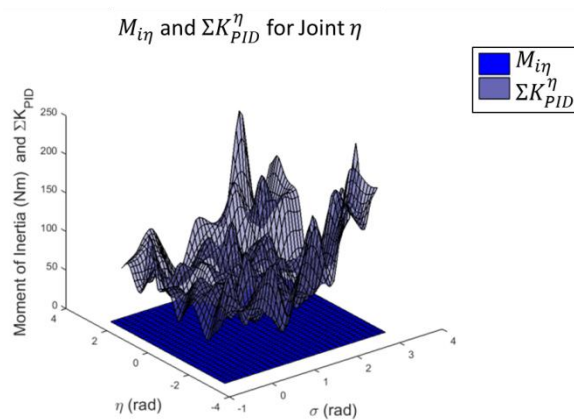
Table 5-10 Gain Plateaus and Surfaces for the PID controller in joint η .





(a) Moments of inertia about η .

(b) PID control gains for η .



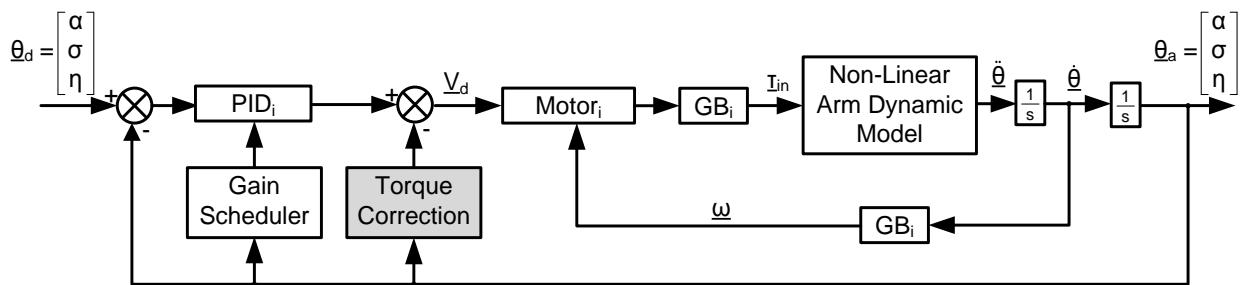
(c) Moments of inertia and sum of PID gains for η .

Figure 5-24 Comparison of moments of inertia of joint η with the PID control gains for the same joint.

In the case of the PID control gains for joint η , the same trend is visible once again, though with a much smaller correlation. This is a more predictable result since the same fundamental principles are in effect. The reason for the reduced correlation in the case of joint η is simple because this joint is the last in the chain, therefore the previous links have much less of an effect on the moments experienced by this joint both opposing and assisting the demanded torque into the joint.

Having determined a series of gain profiles in the form of splines, these splines can be converted into look up tables and used to select gains by gain scheduling. Figure

5-25 shows the architecture of the servo system with the gain scheduler implemented.



Where $i \in \{1,2,3\}$ or $i \in \{\alpha, \sigma, \eta\}$.

Figure 5-25 Control block diagram for the controlled servo and dynamic model system with the implemented gain scheduler.

5.5 Random Step Sequence Testing and Validation

To test the effectiveness of the PID tuning method, the fully tuned system is required to be tested against some input. To do this a series of paths will be generated using uniformly distributed random number within the assumed operating angle range that has been specified for each link in the arm. Each path will consist of a series of three sets of 20 random numbers representing a set of 20 three-dimensional waypoints in the configuration space of the manipulator arm. Effectively each waypoint is an angle demand on each of the three joint angles, and the paths are a series of 20 of these demands. When all three of the joints reach a steady state of 2° (0.0349 radians) or less from the demanded angles, the arm will be considered to have arrived at the waypoint and the next waypoint in the path will be inputted into the arm as a set of demanded joint angles. This method allows multiple different paths to be generated and tested very quickly. A series of twenty paths was generated and tested, and the results of two of these paths are displayed in the following pages. The remaining paths are displayed in 11Appendix A.

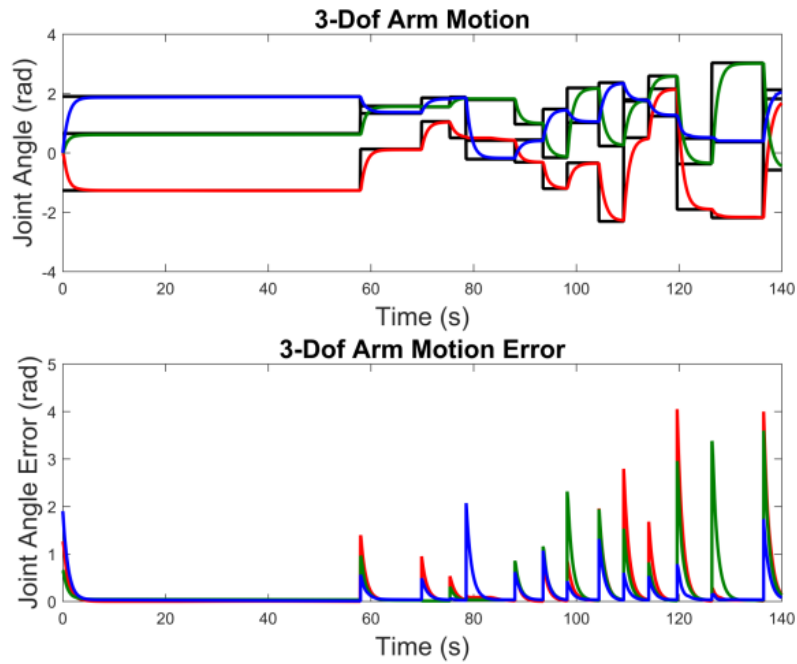
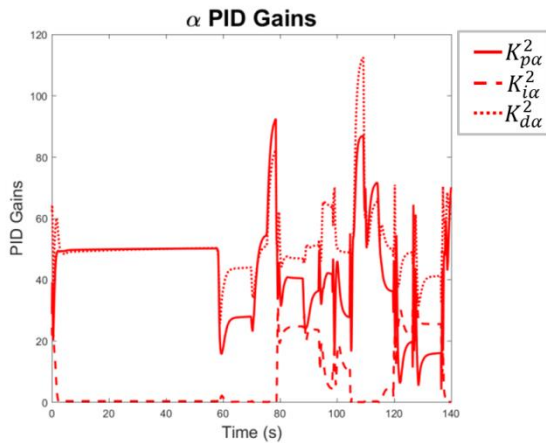
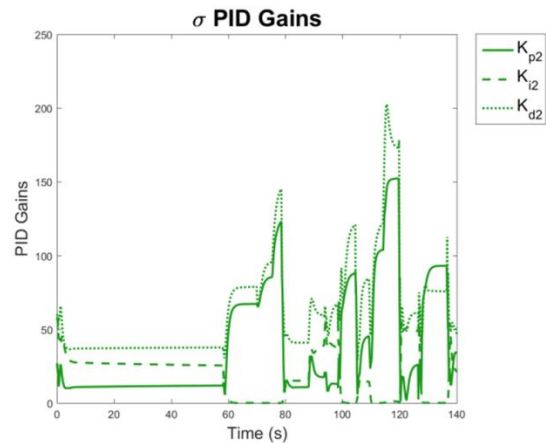


Figure 5-26 Results of the random step sequence testing of the controlled dynamic model. Plot 1 is joint angle against time for each joint and plot 2 is the angular error against time for each joint.

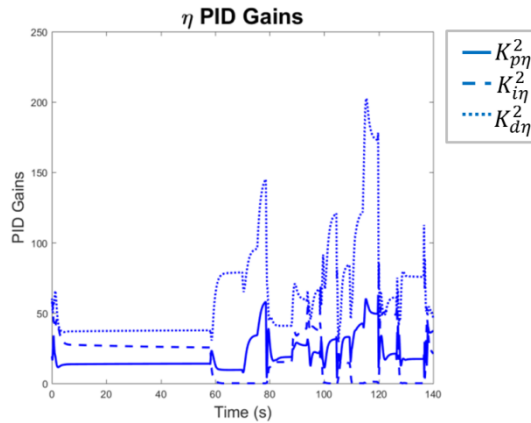
In this scenario, the system was able to reach 13 waypoints. As can be seen in the figure, every time all three joints achieve a value within 2° a new waypoint containing a new demand angle for each joint is given, causing the spikes in joint angle error for each joint. As can be observed from the figure, the spikes in angle error for each joint occur together in time. The biggest observation that can be made from the figure is that the frequency of the changes in waypoint varies through the running of the system over time. Since the waypoint is considered to be achieved when all three joints are within 2° of the demanded angle, this suggests that the system takes varying time to achieve this goal. Investigating further by zooming in to two regions of the above figure illustrates that this is the case. The horizontal dotted line in each of the error subplots illustrates the threshold which all three joint angles must be within for the path to switch to the next waypoint. The vertical dotted lines, which are primarily visible in the second of the two error subplots shows the exact moments in time when the path moves to the next waypoint.



(a) Joint α PID Gains



(b) Joint σ PID Gains

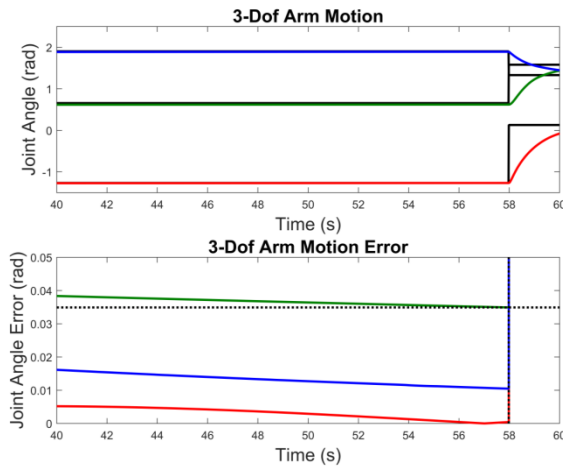


(c) Joint η PID Gains

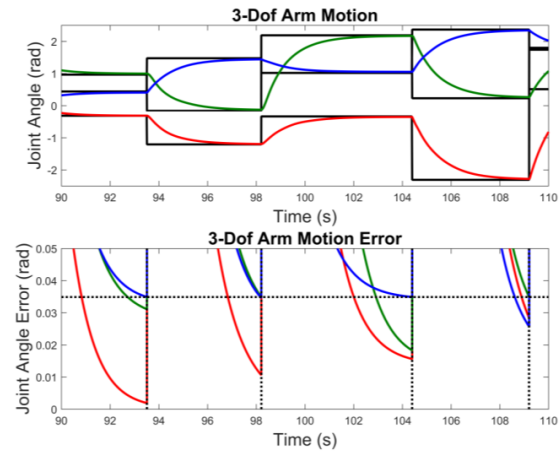
Figure 5-27 PID control gains for each joint over the random step test sequence carried out above.

In Figure 5-28a, the output of the system is focussed on the region of the response between 40 seconds and 60 seconds, which is the final part of the motion to the first waypoint. As can be observed from the figure, the α and η joint angles are well within the 2° threshold (signified by the horizontal dotted line on the figure) for the entire time range which is visible on the figure before the waypoint change. In this case the waypoint changes when σ reaches the 2° threshold. By inspecting a different time range in Figure 5-28b for this scenario, where the frequency of waypoint changes is much higher, given by the time range of 90 to 110 seconds in the second figure, it can be seen that all three joint angles reach the 2° threshold

much more quickly, though it is clear from the figure that they do not reach that threshold at the exact same time.



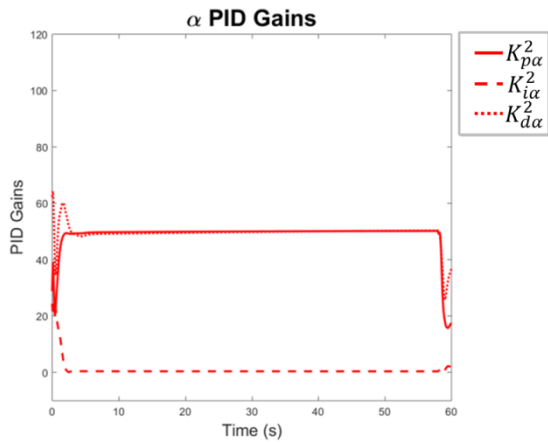
(a) Plot of 40 to 60 seconds.



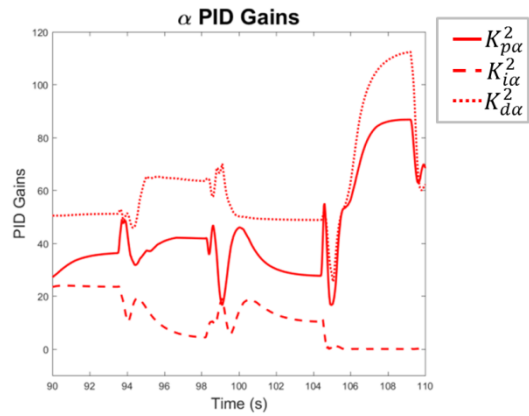
(b) Plot of 90 to 110 seconds.

Figure 5-28 Zoomed in plot of joint angles against time and error against time for the random step test sequence.

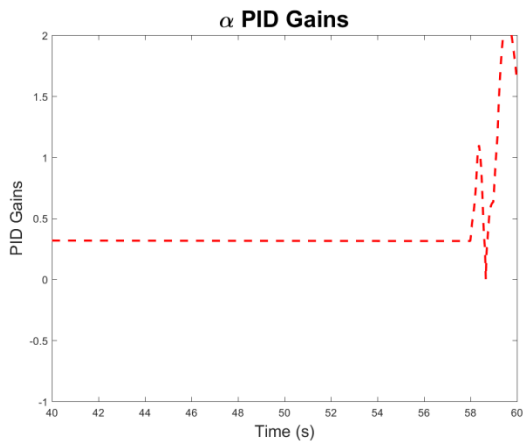
Given that there is a large amount of interconnectivity between the dynamics for each link, which is caused by the change in the moments about each joint which changing joint angle, it is likely that the different inertias at different joint angles has an effect on the ability of the system to reach zero steady state error. Also, given that for these different angle ranges, the system is effectively a different dynamic system for each different joint angle, this means that the ability of the GA to find a solution in each angle range will be different. Also, given that the gain profile for the gain scheduler has been obtained for discrete angle ranges and then interpolated to create a continuous surface, the final gains are an estimate. These two actualities mean that the gains may not provide an exact ideal response as was optimised to. In the case of the above scenario, σ does not converge on the final value therefore the integral gain for the second joint is not strong enough, but for the later parts of the scenario, the integral gain is strong enough since all three joint angles converge on the final value much more quickly. This can be seen by investigating the PID gains over the periods presented in the above figures. Figures 4-28 to 4-31 show the gains on each of the joints.



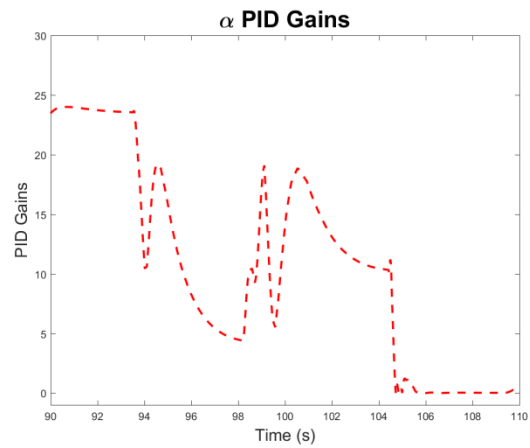
(a) PID gains of joint α over the time range of 40 to 60 seconds.



(b) PID gains of joint α over the time range of 90 to 110 seconds.

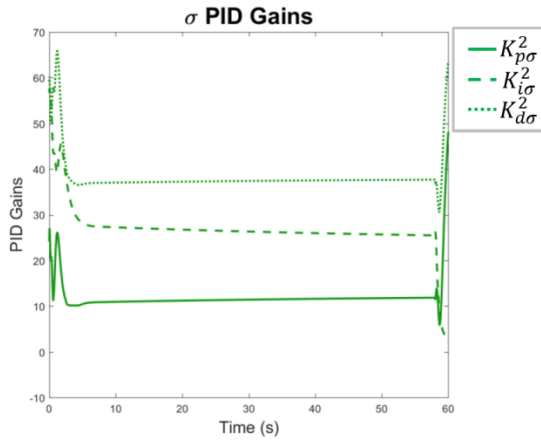


(c) K_i of joint α over the time range of 40 to 60 seconds.

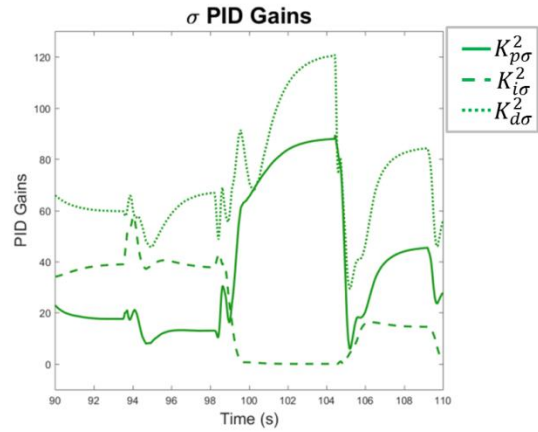


(d) K_i of joint α over the time range of 90 to 110 seconds.

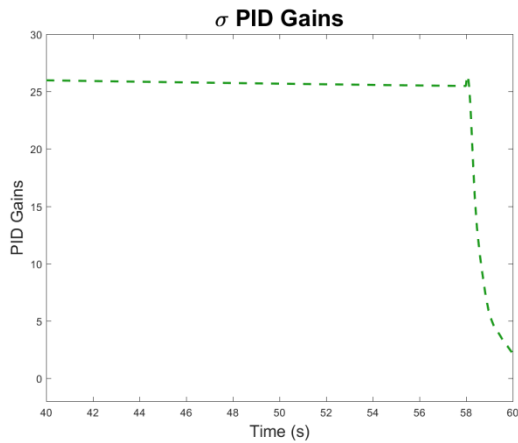
Figure 5-29 PID control gains of joint α in the robotic manipulator during the random step testing of the manipulator arm.



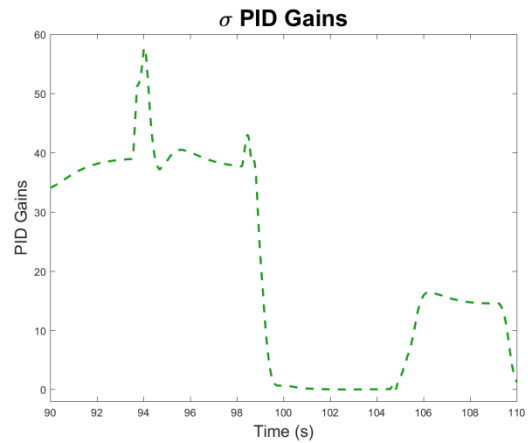
(a) PID gains of joint σ over the time range of 40 to 60 seconds.



(b) PID gains of joint η over the time range of 90 to 110 seconds.

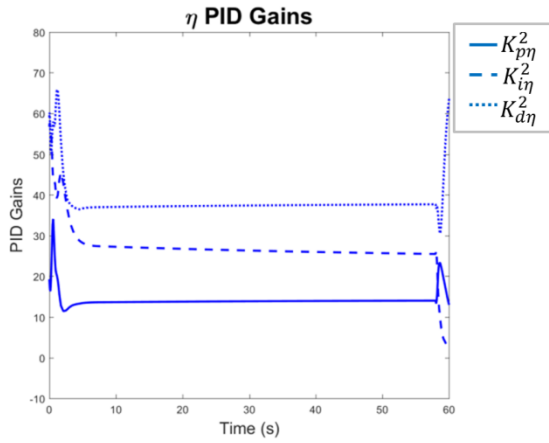


(c) K_i of joint σ over the time range of 40 to 60 seconds.

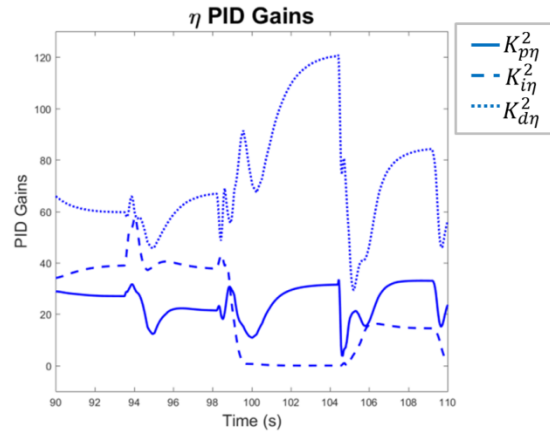


(d) K_i of joint α over the time range of 90 to 110 seconds.

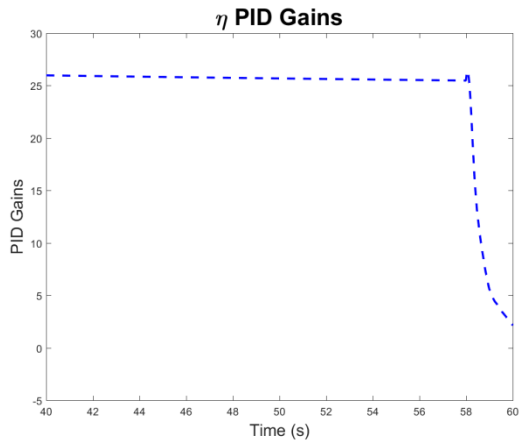
Figure 5-30 PID control gains of joint σ in the robotic manipulator during the random step testing of the manipulator arm.



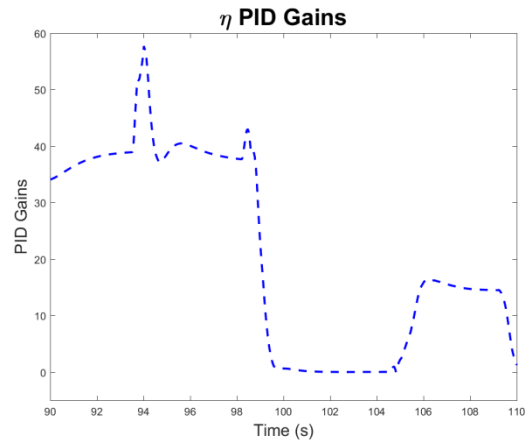
(a) PID gains of joint η over the time range of 40 to 60 seconds.



(b) PID gains of joint η over the time range of 90 to 110 seconds.



(c) K_i of joint η over the time range of 40 to 60 seconds.



(d) K_i of joint η over the time range of 90 to 110 seconds.

Figure 5-31 PID control gains of joint η in the robotic manipulator during the random step testing of the manipulator arm.

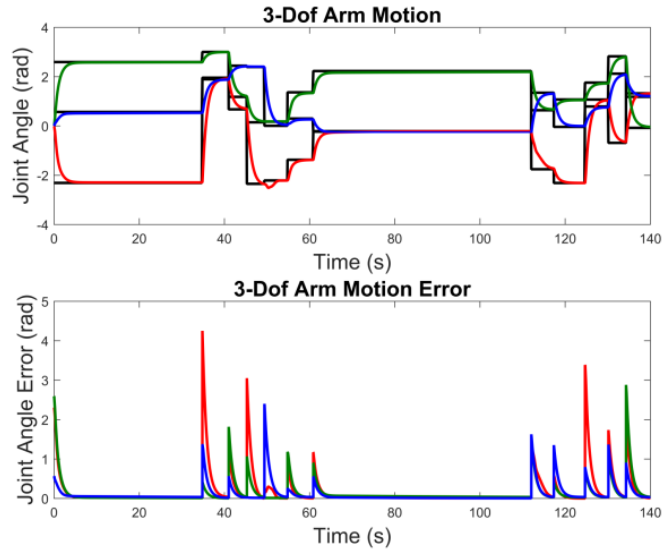
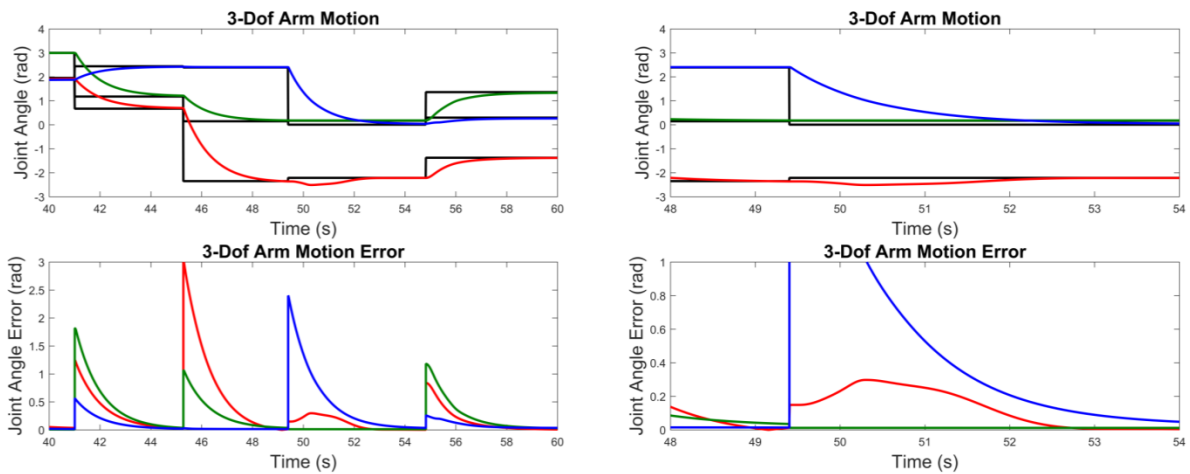


Figure 5-32 Results of the random step sequence testing of the controlled dynamic model. Plot 1 is joint angle against time for each joint and plot 2 is the angular error against time for each joint.

In the second scenario presented here, shown in Figure 5-32, there is a range of time where the α joint appears to travel in the opposite direction to the demand. This is clearly problematic and the reasons behind this must be discussed.



(a) Plot of 40 to 60 seconds.

(b) Plot of 48 to 54 seconds.

Figure 5-33 Zoomed in plot of joint angles against time and error against time for the random step test sequence.

In Figure 5-34 the step response to a unit input of the system $\frac{1}{s+1}$. The velocity of the system is also given. If the required maximum error before the system is allowed to switch to the next waypoint is set as 0.3, then the waypoint would switch when the system had a position of 0.7 and hence a velocity of 0.3. If this required maximum error were set to 0.1, then the waypoint would switch when the system had a position of 0.9 and a velocity of 0.1

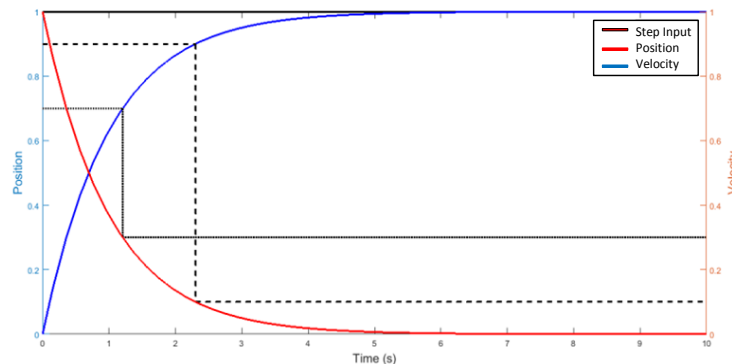


Figure 5-34 Position and velocity change for a system at different points in time, especially when tending towards steady-state.

It is this characteristic which means that the system has a higher velocity when switching between waypoints when the tolerance to the waypoint that the joint angles have to be within is larger. For the robotic manipulator links, there is a significant amount of inertia involved, especially when the arm is fully extended or σ and η tend towards 0° , which in the case of the second scenario is the location that the 2nd and 3rd joints are heading to. This means that the first joint is experiencing the largest range of moments of inertia during this time period, and the link reaches the 2^o waypoint threshold before it has decelerated fully. This means that the joint still has angular velocity in the opposite direction to the input from the new waypoint which the controller has to reduce before accelerating in the correct direction. This is what causes this overshoot effect. The first joint experiences the most inertia of all three therefore the highest susceptibility to this issue. The solution would be to reduce the waypoint threshold and this will be investigated further in the validation of the guidance method in Chapter 7.

5.6 Summary of GA PID Gain Tuning Method

This chapter has dealt with the selection of a suitable control schema for the operation of a 3-DoF robotic manipulator arm. The selection made was a set of three gain scheduled PID controllers. These controllers required tuning; therefore gain selection was carried out by optimisation using a genetic algorithm to find a series of 9 gain profiles. The resultant gain profiles were then used in a gain scheduled PID controller to drive the arm in an acceptable manner. The controller produced in this way was able to satisfy a steady-state error requirement of 2° on each joint.

Having developed the dynamic model of a 3-DoF robotic manipulator arm in Chapter 4, and successfully tuned a PID controller to drive the model within an acceptable performance range in this chapter, the next step in the process of autonomous motion of the arm is to design and implement a guidance algorithm which will enable the arm to plan a safe path through a close-proximity environment in real time given sensor data about the surroundings and then follow the path. This stage of the process will be investigated in Chapters 6 and 7.

6 ENVIRONMENT MODELLING AND MAPPING IN C-SPACE

Chapter 4 of this thesis have developed a dynamic model of a 3-DoF robotic manipulator arm and its corresponding control and tuning. In chapter 5, a control schema was applied to the developed model to allow it to be driven and controlled in a stable and predictable manner in Euclidean space. This control schema was a gain scheduled PID controller, tuned using a Genetic Algorithm and the robotic arm and its controller were assessed in their performance.

Having tuned and validated system is important since it gives a set of numeric parameters which any guidance method can be designed and fitted around. In this chapter existing literature in the area of robotic arm guidance is investigated to assess the possibility of use of existing techniques and to inform the development of a technique for use in the context of guidance in close-proximity environments.

This chapter will outline a method of generating simulated environment data and a method of generating environment data to satisfy the requirement of simulated environment data in the algorithm. Secondly, this chapter will deal with the investigation of how the environment can be mapped to provide a method for obstacle avoidance for the entire robotic arm. The basis for this technique stems from the kinematics described in the previous chapters. If all of the robotic joint angles can be calculated when the position of any point along the robotic arm is specified, then the joint angles that cause a collision between a point along the arm and an obstacle in the real world can be calculated. This creates an impermissible region in the control angle domain which defines the joint angle combinations that the manipulator arm cannot use, otherwise a collision will occur. The impermissible region can be expanded to provide a permissible boundary layer which, along with the permissible boundary layers to all other obstacles in the environment, can be used to form a node graph. The remainder of this chapter will detail the validation of the developed guidance method for use with the simulation model developed in

Chapters 4 and 5. The work carried out in this chapter is outlined in purple in Figure 6-1, with all of the other processes greyed out.

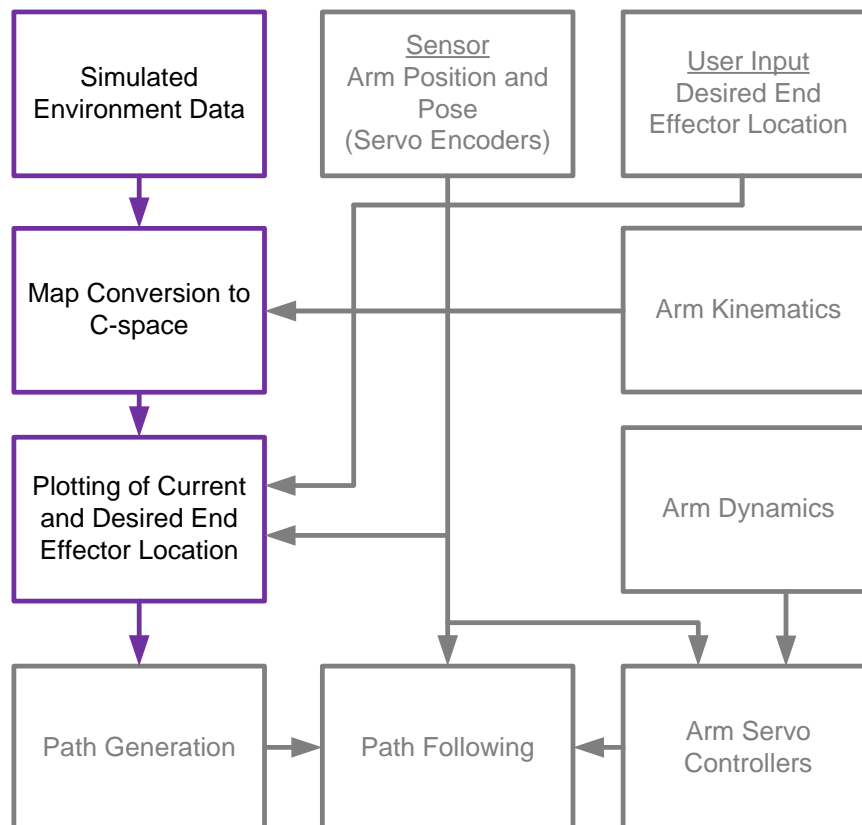


Figure 6-1 Environment and mapping in C-Space (purple) in relation to the overall guidance method.

6.1 Implementation

In order to implement a guidance method using the two above techniques, several steps are involved:

1. Obtain obstacle data.
2. Convert obstacle data into C-Space.
3. Expand C-Space obstacle angle data by the steady-state error of the arm determined in chapter 2 in order to prevent any collisions due to error from the demand angles.

4. Create a node graph in the available range of C-Space which contains the obstacle information.
5. Map the end effector start and end points in C-Space.
6. Carry out a pathing algorithm to generate a path through C-Space.

6.1.1 Obtain Obstacle Data

Before being able to convert any obstacles into C-Space their T-space data must first be obtained. Since in this case the system is a simulated system, simulated data will also be used. Since the selection for sensor was a LIDAR, then simulated data must be obtained which follows this assumption. A second assumption that will be made is that the LIDAR is an ideal sensor, and so displays no measurement noise or dynamics. These sensors measure data in terms of heading and range, but this can be very quickly transformed into Euclidean coordinates.

The distance between consecutive measured points on an obstacle is dependent on two factors, the angular resolution of the sensor and the range from the sensor to the obstacle.

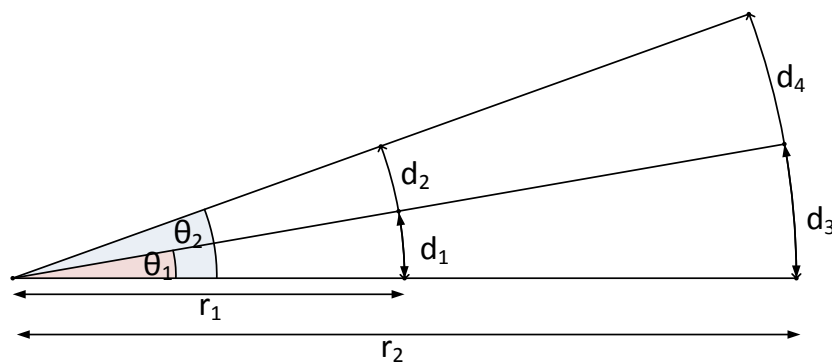


Figure 6-2 Change of measured point spacing with sensor angular resolution and range.

This relationship is governed by Equations (6.1) to (6.5).

$$d = r\theta \quad (6.1)$$

$$d_1 = r_1\theta_1 \quad (6.2)$$

$$d_2 = r_2\theta_1 \quad (6.3)$$

$$d_3 = r_3\theta_2 \quad (6.4)$$

$$d_4 = r_4\theta_2 \quad (6.5)$$

Since the distance between measured points in an object will change with range to the object, the first place to start is to investigate realistic angular resolutions of LIDAR sensors to obtain a range of realistic angles between measured points for this type of sensor. Several datasheets for commercially available LIDAR sensors were investigated and a realistic range of angular resolutions for this type of sensors was between 0.09° and 0.36° , (Velodyne LIDAR, 2016), (Hokuyo Automatic Co., Ltd., 2016), (Technical Avenue Sdn Bhd, 2016), therefore this is the range over which distance between measured points will be investigated. The range over which distance between measured points changes that is required to be investigated is from 0 m to the maximum extension range of the robotic arm, which is 0.96 m. Therefore the distance between each consecutive measured point in this range of angles and range from sensor can be displayed in Figure 6-3.

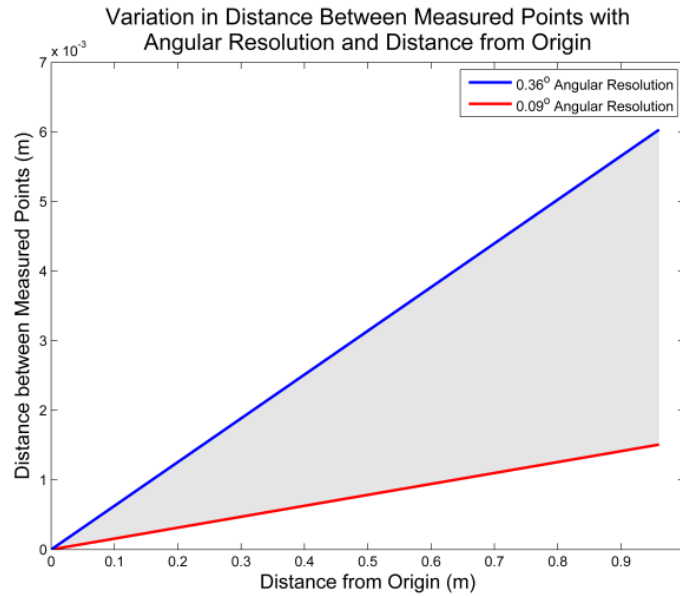


Figure 6-3 Change of distance between measured points with distance from origin and angular resolution.

The maximum measured point spacing is very small, on the order of $10^{-3} m$. This resolution will produce a very large number of measured points per object since the spacing is between 1.5 and 6 mm depending on the angular resolution of the sensor. Given that this is the case a selection of the mean of these two limits will be used at the maximum arm extension range, which is $2.92 mm \approx 3 mm$. This will be the spacing between each point. To quickly generate the measured points in objects a method must be developed which allows for an object shape to be specified and then the points which are to be measured in that object very quickly generated. These sensors also have an accuracy of less between 0.02 m at 25 m. This translates to an accuracy of $8 \times 10^{-4}^\circ$. For the maximum extension range of the arm this gives an accuracy of $1.3 \times 10^{-5} m$ or 0.134 mm. In this case of the work carried out in this thesis the LIDAR is assumed to be noiseless and 100% accurate, but in a reality, this accuracy would impact the accuracy of the measured points. This will be further discussed in section 6.1.2 of this chapter.

Triangular Polygon Object Formation

In computer graphics, objects are generated into any shape by approximating them using a series of small triangular polygons. To generate a more accurate approximation of the object, a larger number of smaller polygons is used. However, the drawback is that the computing power required for more polygons is also larger. This means that one method of quickly generating sensor data is to build an object from triangular polygons to form the shape of the object in question and then automatically generating points of the correct spacing in each polygon.

An object formed by n polygons can be described by a matrix n -by-9 in size. Each row represents one polygon that makes up the object. The elements in each row are, $x_a, y_a, z_a, x_b, y_b, z_b, x_c, y_c$ and z_c , which represent the x, y and z coordinates of each corner of the polygon in Euclidean space. This creates a matrix O in (6.6), which describes an entire object.

$$O = \begin{bmatrix} x_a & y_a & z_a & x_b & y_b & z_b & x_c & y_c & z_c \\ \vdots & & & & & & & & \vdots \\ \vdots & & & & & & & & \vdots \\ \vdots & & & & & & & & \vdots \\ x_{an} & y_{an} & z_{an} & x_{bn} & y_{bn} & z_{bn} & x_{cn} & y_{cn} & z_{cn} \end{bmatrix} \quad (6.6)$$

Each row in the O matrix can be used to generate a series of inspection points which can be used to calculate the range of possible collisions with the manipulator arm. This is done by scattering inspection points across the polygon in a regular distribution as shown in Figure 6-4.

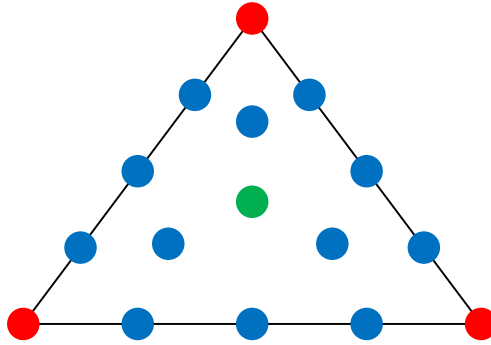


Figure 6-4 Distribution of inspection points across one polygon. The red points represent the polygon corners, the green point represents the polygon centre and the blue points represent the other inspection points.

This spread is created by calculating the centre point of the circle in Equation (6.7).

$$P_c = \frac{P_1 + P_2 + P_3}{3}$$

where,

(6.7)

$$P_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}, P_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}, P_3 = \begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix}$$

The vector from the centre to each corner can then be calculated:

$$\vec{V}_c^1 = P_1 - P_c$$

$$\vec{V}_c^2 = P_2 - P_c$$

(6.8)

$$\vec{V}_c^3 = P_3 - P_c$$

The inspection points inside the polygon can then be calculated in Equation (6.9).

$$\begin{aligned}
P_{i1} &= P_c + D\vec{V}_c^1 \\
P_{i2} &= P_c + D\vec{V}_c^2 \\
P_{i3} &= P_c + D\vec{V}_c^3
\end{aligned} \tag{6.9}$$

$$\text{where, } D_{k=0}^{\frac{\|P_{n+1}-P_n\|}{d}} = P_n + kd$$

And d is the distance between measured points. The vector from corner to corner is calculated in Equation (6.10):

$$\begin{aligned}
\vec{V}_1^2 &= P_2 - P_1 \\
\vec{V}_2^3 &= P_3 - P_2 \\
\vec{V}_3^1 &= P_1 - P_3
\end{aligned} \tag{6.10}$$

The inspection points along the sides of the polygon can be found in the same way as the inspection points inside the polygon using Equation (6.11).

$$\begin{aligned}
P_{en,1} &= P_n + D\vec{V}_n^{n+1} \\
P_{en,2} &= P_n + D\vec{V}_n^{n+1} \\
P_{en,3} &= P_n + D\vec{V}_n^{n+1}
\end{aligned} \tag{6.11}$$

Where D is the same vector as previously specified. All of the points are stored in transpose in a single matrix, creating a matrix n-by-3 in dimensions which represents all of the inspection points in a single polygon, where n is the number of points in the polygon. Because of the nature of the method that the inspection points are

calculated, the polygons have no requirement to be equilateral triangles, which is useful for constructing entire obstacles.

Sphere Generation

The previous method of generating measured points in objects is very useful for building highly detailed objects very quickly, but these objects must be modelled in terms of their polygons before generation and this is very time consuming. A faster method in terms of very quickly generating objects is to provide a centre point and generate a sphere of measured points around it.

Firstly the azimuth and elevation angles must be calculated to generate each point on the surface of the sphere with the correct spacing between them. This is once again carried out using Equation (6.12).

$$\begin{aligned}
 d &= r\theta \\
 \therefore \\
 \theta &= \frac{d}{r}
 \end{aligned}
 \tag{6.12}$$

Where d is the spacing between points and r is the radius of the sphere. Having carried this calculation out, Equation (6.13) is used.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = C + \begin{bmatrix} r\cos(A)\cos(E) \\ r\sin(A)\cos(E) \\ r\sin(E) \end{bmatrix}
 \tag{6.13}$$

where $A \in \{-\pi, \theta, \pi\}, E \in \{-\pi, \theta, \pi\}$

This generates a set of points which form the surface of a sphere. These points can then be converted from T-space to C-Space.

6.1.2 Convert Obstacle Data into C-Space

Two methods have been used to convert the obstacles into C-Space. Both methods decouple each of the links into its own problem and then convert the obstacle into C-Space for each joint. Any collision in the first joint will occur over the whole range of the 2nd and 3rd angles any collision in the second joint will occur over the whole range of the 3rd angle. The first method uses trigonometry to calculate the angles for collisions with the first and second joints in the arm and a series of simultaneous equations to calculate the angles for collisions with the third joint. The second method uses trigonometry to calculate the angle solutions to collisions along all three links.

Method 1 – Trigonometry and Simultaneous Equation Solution

Collisions with obstacles can occur in this case with one of the three links in the 3-DoF manipulator arm, and the links are temporarily decoupled to calculate collision angles for each of them. Should the collision occur with the first link then the angles responsible for the position of the other two links are irrelevant and so for the angles of the first link that cause a collision, α , the other angles, β and γ , will collide regardless of angle from $-\pi$ to π . The same occurs if the second link does not collide but the third link does. The angle responsible for the position of the second link is irrelevant so for a collision of the first link at a set combination of α and β there will be a collision for all angles of γ between $-\pi$ and π radians.

Collision Range of First Link

Any collisions with the first link must occur if a polygon passes through the plane of operation of the first link, inside a circle on the plane with its centre at the base point of the manipulator and radius of the length of the first link. This circle represents the maximum range of the first link.

Line:Plane Intersection

For each polygon in an object, the first thing that is calculated is whether or not it crosses the plane of operation of the first link. This is done by calculating the line-plane intersection of each of the edges of the polygon, which can be considered to be lines of infinite length which intersect with one another. The plane of operation of the first link is defined as a plane in the X-Y directions with $z = 0$.

In vector notation a plane can be expressed as Equation (6.14).

$$(p - p_0) \cdot n = 0 \quad (6.14)$$

The point p_0 and p are points on the plane. In this case p_0 is the origin point of the first link in the arm and p is the point on the plane where a line intersects it. In this case p is the unknown. The variable n is a vector perpendicular to the plane, which in this case is required to be equal to n in Equation (6.15):

$$n = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (6.15)$$

From vector mathematics, the dot product of two vectors is the cosine of the angle between them. For a vector which is perpendicular to another the angle would be $\pi/2$ radians, therefore the cosine would be 0, hence the 0 in Equation (6.14). The vector equation for a line is shown in Equation (6.16).

$$p = l_1 + dL \quad (6.16)$$

The variable p in this equation is again the intersection between the line and the plane. The variable l_1 is a point on the line, in this case one of the corners of the polygon, and L is the direction unit vector towards another point on the line, in this case the other polygon corner which bounds the line. The variable d is the distance

along the vector to the point p . By substituting the line vector equation for p into the plane vector equation, Equation (6.17) is achieved:

$$(l_1 + dL - p_0) \cdot n = 0 \quad (6.17)$$

This can be rearranged using the process in Equation (6.18).

$$\begin{aligned} dL \cdot n + (l_1 - p_0) \cdot n &= 0 \\ dL \cdot n &= (p_1 - l_0) \cdot n \\ d &= \frac{(p_1 - l_0) \cdot n}{L \cdot n} \end{aligned} \quad (6.18)$$

This provides the distance d which can be re-entered into the vector line equation to find the line-plane intercept if d is a real number. It is required to check that p lies between the two line edges. This is done by using the magnitudes of the vector between the two corners, and the vectors from the corners to the intersection point, displayed in Equation (6.19).

$$\begin{aligned} m_0^1 &= |l_1| \\ m_0^p &= |l_p - l_1| \\ m_p^1 &= |l_1 - l_p| \end{aligned} \quad (6.19)$$

If the point lies inside the two corners then Equation (6.20) is used.

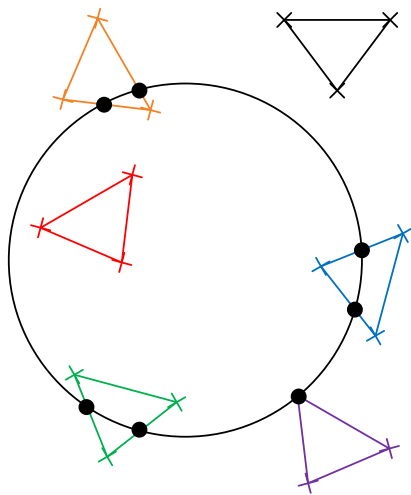
$$m_0^p + m_p^1 = m_0^1 \quad (6.20)$$

Line:Circle Intersection

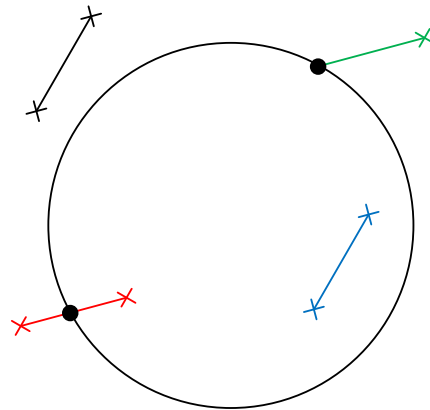
Once it is determined how many of the edges of the polygon intersect with the plane, they can be checked for intersections with the circle of maximum range of the first link.

For the first link, any of the three lines of a polygon that pass through the plane of operation of the link are inspected for their intersection with that plane. This leads to one of four cases. 1) The entire polygon lies on the plane, in which case all three lines intersect the plane at all points in the lines. 2) The polygon could intersect the plane, in which case two of the lines would intersect with the plane, giving two intersection points. 3) The polygon could be just touching the plane, in which case two of the lines would intersect the plane at the point at which they intersect each other, i.e. the corner of the polygon. 4) The polygon does not intersect with the plane at all, in which case there are no intersection points. In the cases of intersections with the plane, any intersection which occurs inside the circle whose radius is the length of the first link, and whose origin is the origin of the first link (the first arm joint) is inspected for the maximum angle range between them. This inspection relies on the assumption that a continuous object will have a continuous range of collision angles. This angle range when defined in terms of distance from the zero angle of the joint is the range which will cause a collision between the obstacle and the first link.

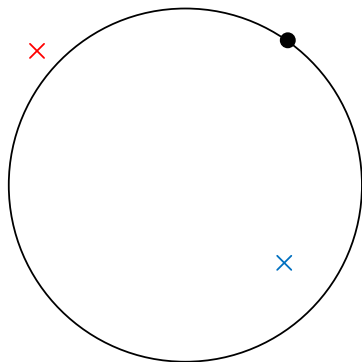
Figure 6-5 illustrates the possible cases of intersections between a polygon and the circle which bounds the range of the first link in the manipulator arm. In cases 1 and 3 the crosses represent corners of the polygon. In case 2 the crosses represent intersections between edges of the polygon and the plane of operation. In all cases the black dots represent intersections between the polygon and the edge of the reachable range of the link.



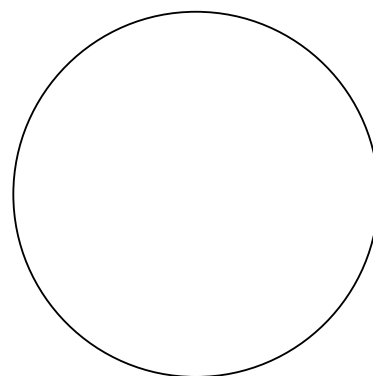
(a) Case 1: Polygon lies completely on plane of operation of 1st link



(b) Case 2: Polygon intersects with plane of operation of 1st link.



(c) Case 3: Polygon corner intersects with plane of operation of 1st link



(d) Case 4: Polygon does not intersect with plane of operation of 1st link

Figure 6-5: Different cases of intersection of polygons with the plane of operation of the 1st link in the manipulator.

In case 1 each line is checked for a line-circle intersection. In case 2, the two line-plane intersections are connected to form a line which is checked for a line-circle

intersection. In case 3, the distance from the centre of circle to the line-plane intersection point is calculated. If it is less than the radius of the circle then the point lies inside the circle. In case 4 there is no intersection between the polygon and the plane, so there is no collision angle range for the first link.

For each line that occurs, it must be investigated for a line-circle intersection. This is done by using the Euclidean line and circle equations in Equation (6.21).

$$y = mx + c$$

$$(x - a)^2 + (y - b)^2 = r^2 \quad (6.21)$$

Where a and b are the x and y coordinates of the centre of the circle respectively and in this case are both 0. The line equation can be substituted into the circle equation and the resulting quadratic equation solved to find both possible intersections (if any occur).

$$x^2 + (mx + c)^2 = r^2$$

$$\therefore \quad (6.22)$$

$$(m^2 + 1)x^2 + 2mxc + c^2 + r^2 = 0$$

This quadratic equation is solved using equation (6.23).

$$y = \frac{-mc \pm \sqrt{-c^2 + m^2r^2 + r^2}}{(m^2 + 1)x} \quad (6.23)$$

where the solutions are the y values of the line-circle intersections. The calculated solutions can then be used to find the range of the first joint angle that causes a collision.

Collision Range of Second Link

To find instances of collisions with the second joint is simpler than that of the first. The direction in the X-Y plane that the polygon inspection point lies, which corresponds to the joint angle of the first link (α), can be calculated as follows. If

$P_i = \begin{bmatrix} I_x \\ I_y \\ I_z \end{bmatrix}$ then the angle of the first link α can be calculated using Equation (6.24).

$$\alpha = \arctan2\left(\frac{I_y - P_{0y}}{I_x - P_{0x}}\right) \quad (6.24)$$

Using this angle the joint between the first and second links can be found.

$$P_1 = P_0 + l_1 \begin{bmatrix} \cos \alpha \\ \sin \alpha \\ 0 \end{bmatrix} \quad (6.25)$$

This point relates to the base point of the second link. The angle direction from this point to the inspection point can now be calculated in Equation (6.26).

$$\vec{V}_1^I = P_i - P_1$$

$$|\vec{V}_{1xy}^I| = \sqrt{(\vec{V}_{1x}^I)^2 + (\vec{V}_{1y}^I)^2} \quad (6.26)$$

$$\beta = \arctan2\left(\frac{\vec{V}_{1z}^I}{|\vec{V}_{1xy}^I|}\right)$$

If the magnitude of \vec{V}_1^I is smaller than or equal to l_2 then the second link will collide with the inspection point at that combination of α and β joint angles.

Due to the geometry of the arm, and the length of the second and third links being longer than that of the first, it is possible for collisions between the second or third links and an object when $\alpha = \alpha \pm \pi$ rad (i.e. pointing in the opposite direction to the object), therefore the process must be carried out again for that case.

Collision Range of Third Link

To investigate collisions between inspection points and the third link, the kinematic equations that have been described in the previous chapter must be used. The angle α is calculated in the same way as it is for collisions with the second link. Equations (6.27) to (6.34) show how the joint angle ranges of collisions with the third link are calculated.

$$P_{xy1} = |\vec{V}_{1xy}^I| \quad (6.27)$$

$$P_{z1} = 0$$

$$xy_2 = -\frac{AB \pm P_{fz}\sqrt{-CD} + l_1^3 - P_{fxy}^3}{E} \quad (6.28)$$

where,

$$\left. \begin{aligned}
 A &= l_1(l_2^2 - l_3^2 + P_{fz}^2 + 3P_{fxy}^2) - P_{fxy} \\
 B &= 3l_1^2 + l_2^2 - l_3^2 + P_{fz}^2 \\
 C &= (l_1^2 - 2l_1P_{fxy} - l_2^2 + 2l_2l_3 - l_3^2 + P_{fz}^2 + P_{fxy}^2) \\
 D &= (l_1^2 - 2l_1P_{fxy} - l_2^2 - 2l_2l_3 - l_3^2 + P_{fz}^2 + P_{fxy}^2) \\
 E &= 2(l_1^2 - 2l_1P_{fxy} + P_{fz}^2 + P_{fxy}^2)
 \end{aligned} \right\} \quad (6.29)$$

$$xy_3 = \mp \frac{A \mp P_{fxy}B + P_{fz}\sqrt{C} + l_1^3 \mp P_{fxy}^3}{F} \quad (6.30)$$

where,

$$\left. \begin{aligned}
 A &= l_1(l_3^2 - l_2^2 \pm P_{fz}^2 \pm 3P_{fxy}^2) \\
 B &= 3l_1^2 + l_2^2 - l_3^2 + P_{fz}^2 \\
 C &= -DE \\
 D &= l_1^2 + 2(l_2l_3 - l_1P_{fxy}) - l_2^2 - l_3^2 + P_{fz}^2 + P_{fxy}^2 \\
 E &= l_1^2 + 2(l_2l_3 + l_1P_{fxy}) - l_2^2 - l_3^2 + P_{fz}^2 + P_{fxy}^2 \\
 F &= 2(l_1^2 - 2l_1P_{fxy} + P_{fz}^2 + P_{fxy}^2)
 \end{aligned} \right\} \quad (6.31)$$

$$z_2 = \frac{P_{fz}}{2} + \frac{P_{fz} \frac{l_2^3 - l_3^3 \pm l_1\sqrt{A} \mp P_{fxy}\sqrt{A}}{2}}{D} \quad (6.32)$$

and,

$$z_3 = \frac{P_{fz}}{2} - \frac{P_{fz} \frac{l_2^3 - l_3^3 \pm l_1 \sqrt{A} \mp P_{fxy} \sqrt{A}}{2}}{D} \quad (6.33)$$

where,

$$\left. \begin{aligned} A &= -BC \\ B &= l_1^2 + 2l_1 P_{fxy} - l_2^2 + 2l_2 l_3 - l_3^2 + P_{fz}^2 + P_{fxy}^2 \\ C &= l_1^2 + 2l_1 P_{fxy} - l_2^2 + 2l_2 l_3 - l_3^2 + P_{fz}^2 + P_{fxy}^2 \\ D &= l_1^2 + 2l_1 P_{fxy} + P_{fz}^2 + P_{fxy}^2 \end{aligned} \right\} \quad (6.34)$$

Summary of Calculations

This set of calculations produces a set of three-dimensional points with dimensions that are equivalent to the angle ranges of each link. For the first joint α , β and γ have a range of $-\pi$ to π radians. This is because a collision with the first link makes the positions of the second and third links irrelevant. The same is true with γ for collisions with the third link. The set of points generated for a collision with a polygon provides a comprehensive scatter across the entire range of angles which the manipulator arm cannot enter without colliding with the inspected polygon. This represents the impermissible region caused by the range of collision angles for that polygon.

Two-Link Equation Analysis

A series of 200 random two-dimensional points were generated and the resulting geometries calculated using the inverse kinematics for a two-link arm configuration derived in Chapter 3 are plotted in Figure 6-6.

In Figure 6-6 (a), the green geometries in the upper graph represent the arm geometries where the randomly-generated point is achievable. The red geometries

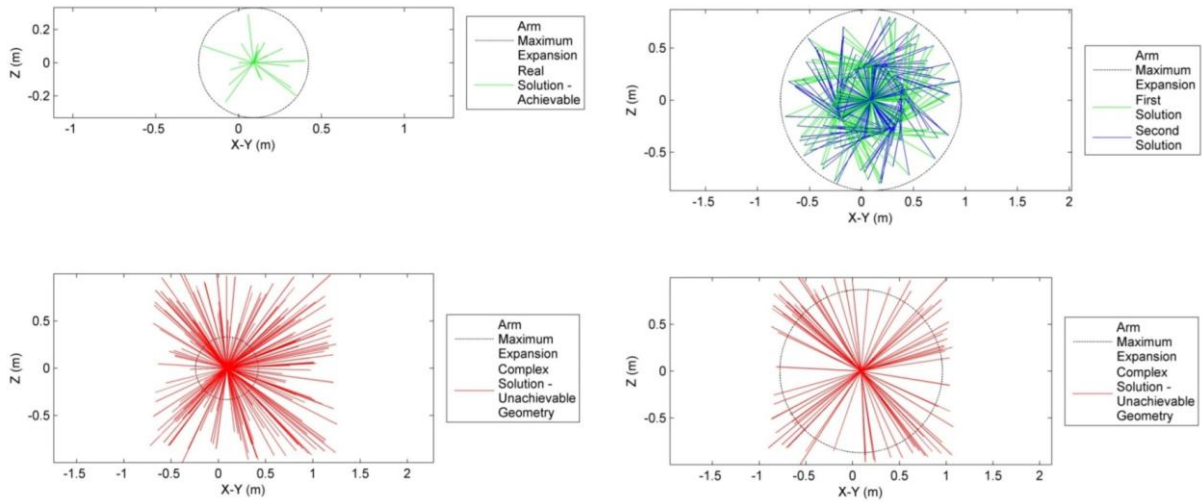
in the lower graph represent the arm geometries where the randomly-generated point is unachievable. Figure 6-7 (a) shows the positions of the randomly-generated points used to calculate arm geometries in Figure 6-6 (a). As seen in the diagram, the red points fall outside the maximum arm extension range, and correspond to the unachievable arm geometries in the lower graph of Figure 6-6 (a). The green points fall inside the maximum arm extension range and correspond to the achievable arm geometries in the upper graph of Figure 6-6 (a). This shows that these equations are viable for a two-link arm.

Three-Link Equation Analysis

The equations developed to calculate the arm geometry for a given collision point have been tested using Monte Carlo Simulation. For the two-link and three-link equation sets a series of random points in a range that includes the arm extension range have been generated to test the effectiveness of the equations.

All of the points in the series of random two-dimensional points that fall outside the achievable arm geometries when using the two-link arm configuration equations were then recalculated using the inverse kinematics for the three-link arm configuration derived in Chapter 3 and were plotted in Figure 6-6 and Figure 6-7.

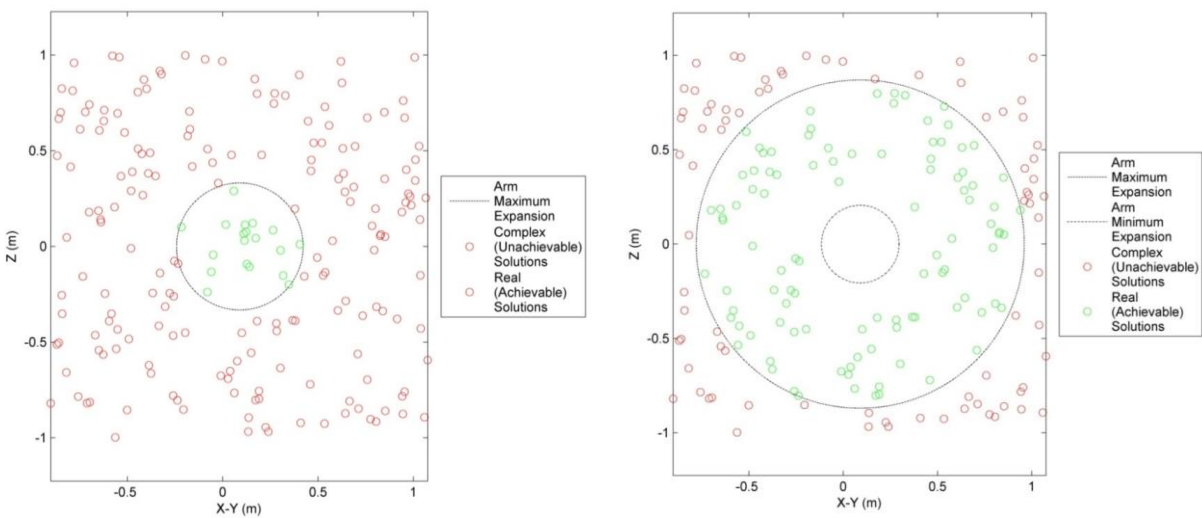
In Figure 6-6 (b) the green and blue geometries in the upper graph represent the arm geometries where the randomly-generated point is achievable. In a three-link configuration such as this one, there are two possible geometric solutions for each randomly-generated inspection point. The blue and green geometries correspond to each of the two solutions. The red geometries in the lower graph represent the arm geometries where the randomly generated point is unachievable. Figure 6-7 (b) shows the positions of the randomly-generated points used to calculate arm geometries in Figure 6-6 (b). As seen in the diagram, the red points fall outside the maximum arm extension range, and correspond to the unachievable arm geometries in the lower graph of Figure 6-6 (b). The green points fall inside the maximum arm extension range and correspond to the achievable arm geometries in the upper graph of Figure 6-7 (b). This shows that these equations are viable for a three-link arm.



(a) 2nd link configuration of a two-link arm.

(b) 2nd and 3rd link configuration of a three-link arm.

Figure 6-6: Geometries of (a) two-link and (b) three-link arm configurations generated by random demand end effector positions as inspection points.



(a) Two-link arm analysis.

(b) Three-link arm analysis.

Figure 6-7: Positions of the randomly-generated points used to calculate arm geometries in Figure 6-6. The green points fall within the accessible range of each arm configuration, whereas the red points are inaccessible.

As can be seen from Figure 6-6 and Figure 6-7, a large proportion of the points that were unachievable with the two-link equations become achievable when calculated using the three-link equations. It is also worth noting that the accessible range of the three-link equations has a circular hole in the centre. Any inspection point that falls inside this hole will be achievable by the two-link series of equations, as the lower limit characteristic displayed by the first link is not present for the two link configuration because the second link is longer than the first. In Figure 6-6 (a) the green lines and in (b) the green and blue lines represent the geometries required when the random points have fallen within the reachable range, of the arm. In both diagrams the red lines represent the cases where the random points fall outside of the reachable range.

Method 2 – Trigonometry Only

This method uses only the trigonometric inverse kinematics so find solutions to collisions with all of the points measured points on an obstacle and points along the entire range of each joint. For the first and second joints, this solution only requires finding the angle between the measured point and the joint in question and then the range from the measured point to the joint is compared with the length of the link. If the Euclidean distance between the joint and the measured point is smaller than or equal to the length of the link then there is a collision with the obstacle at that joint angle. For the third joint the inverse trigonometric equations of end effector are used. A vector of points representing inspection points along the third link are inputted as the total length of the third link.

For joint α , the vectors P_x and P_y are the vectors which contain the X and Y coordinates of each of the measured points in the obstacle. For this calculation only the points where P_z (the Z coordinates of the measured points) are 0, hence they lie on the plane of the first link. The calculation of the α angles of collisions between the first link and any obstacles is given with Equation (6.35).

$$A = \tan^{-1} \frac{P_y}{P_x} \quad (6.35)$$

Where A is the vector of joint angles found by the inverse tangent of the quotient of the vectors P_y and P_x . For each of these angles, the Euclidean distance to the point from the origin is found using Equation (6.36) and for all those points where the distance is less than or equal to the length of the link, l_1 , then the angle is kept as a collision angle.

$$R = \sqrt{P_x^2 + P_y^2} \quad (6.36)$$

For joint σ , the same method is used but in this case the location of the joint has to be calculated. For each measured point, joint angle α is calculated using the above equation, and then the location of the joint between links 1 and 2 can be found using the forward kinematics.

Since, in the case of the robotic arm presented in this thesis, the second joint is larger than the first, there is the potential for two solutions to the joint angles α and σ . There may be a solution when $-\pi^c \leq \alpha \leq \pi^c$, but also when $\pi^c \leq \alpha \leq 2\pi^c$. This second possible range of α angles is given by Equation (6.37).

$$A_1 = A \text{ and } A_2 = A - \pi \quad (6.37)$$

Given that this is the case, the two sets of solutions Σ can be found using the schematic shown in Figure 6-8.

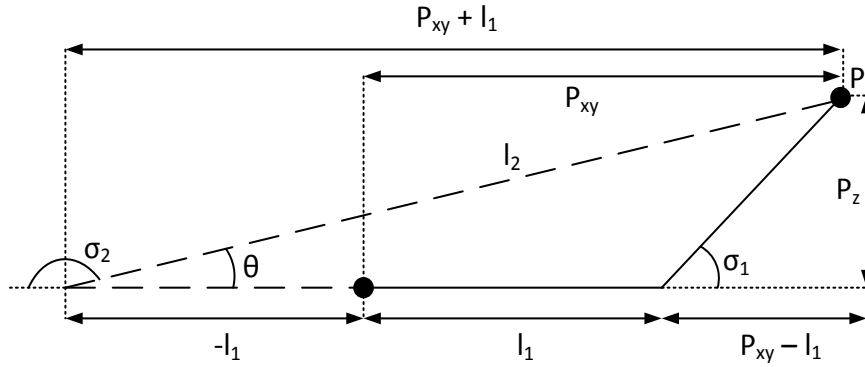


Figure 6-8 Arm geometry for the calculation of σ .

In this case the σ angles can be calculated using Equation (6.38).

$$\Sigma_1 = \tan^{-1} \frac{P_z}{\sqrt{P_x^2 + P_y^2} - l_1} \quad (6.38)$$

$$\Sigma_2 = \tan^{-1} \frac{P_z}{\sqrt{P_x^2 + P_y^2} + l_1}$$

Again the Euclidean distance from the joint to the measured points is compared with the length of the link, in this case l_2 , and if the Euclidean distance, calculated using Equation (6.39), is smaller than or equal to l_2 then there is a collision between the link and the measured points.

$$R_1 = \sqrt{(\sqrt{P_x^2 + P_y^2} - l_1)^2 + P_z^2}, \text{ for } \sigma_1. \quad (6.39)$$

$$R_2 = \sqrt{(\sqrt{P_x^2 + P_y^2} + l_1)^2 + P_z^2}, \text{ for } \sigma_2.$$

For joint 3, the entirety of the 3rd link is taken into consideration since there are multiple solutions for collisions with the 3rd link and an obstacle. To carry this set of calculations out, the length of link 3, l_3 is inputted as a vector from 0 to l_3 . The schematic illustrating the collision geometries for the third link is given in Figure 6-9.

In this case O is the vector coordinates of the base of the manipulator arm. P is the vector coordinates of a measured point (or all the measured points in the obstacle). Again, the joint angles α_1 and α_2 can be calculated using the inverse tangent of the XY vector between the measured point and the base of the 1st link in the manipulator arm, shown in Equation (6.40).

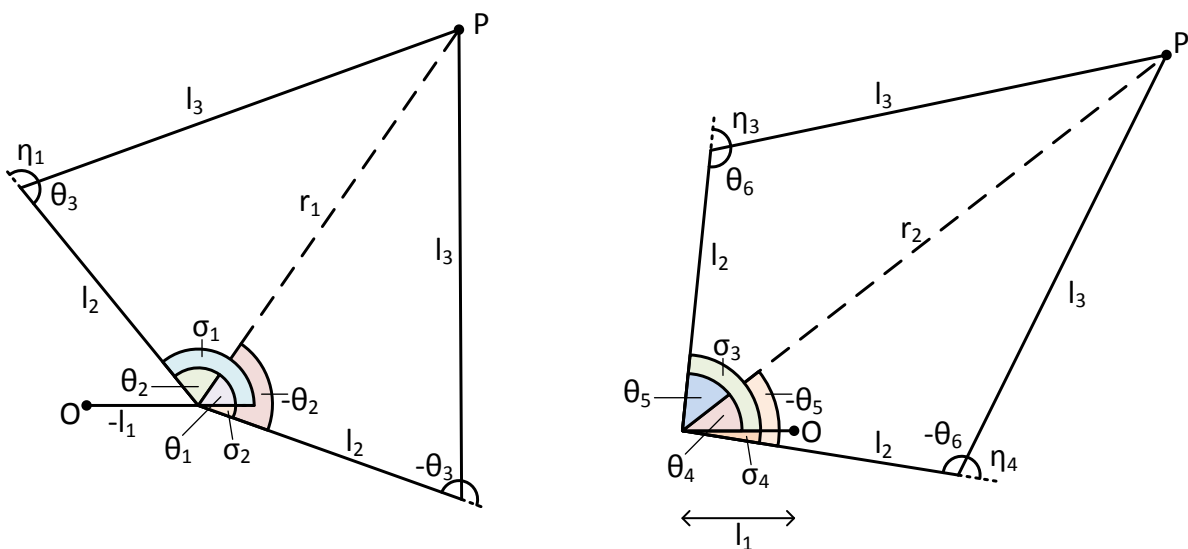


Figure 6-9 Arm geometry for the calculation of the σ and η joint angle combination.

$$A_1 = \tan^{-1} \frac{P_y}{P_x} \tag{6.40}$$

$$A_2 = \pi + \tan^{-1} \frac{P_y}{P_x}$$

The Euclidean distance (h_1 and h_2) between the measured points and the base of the manipulator can be used along with the lengths of links 2 and 3 to find all of the

internal angles of the triangle made by these three lengths. Equations (6.41) to (6.44) provide calculate the internal angles of the geometries shown in Figure 6-9.

$$R_1 = \sqrt{(\sqrt{P_x^2 + P_y^2} - l_1)^2 + P_z^2}, \text{ for } \sigma_1, \sigma_2 \text{ and } \eta_1, \eta_2. \quad (6.41)$$

$$R_2 = \sqrt{(\sqrt{P_x^2 + P_y^2} + l_1)^2 + P_z^2}, \text{ for } \sigma_3, \sigma_4 \text{ and } \eta_3, \eta_4.$$

Given the length of link three as a vector of points along the length of the link, L_3 , the following set of equations will give the angles θ_1 to θ_6 .

$$\theta_1 = \tan^{-1} \frac{P_z}{\sqrt{P_x^2 + P_y^2} - l_1} \quad (6.42)$$

$$\theta_4 = \tan^{-1} \frac{P_z}{\sqrt{P_x^2 + P_y^2} + l_1}$$

$$\theta_2 = \cos^{-1} \frac{l_2^2 + R_1^2 - L_3^2}{2l_2R_1} \quad (6.43)$$

$$\theta_5 = \cos^{-1} \frac{l_2^2 + R_2^2 - L_3^2}{2l_2R_2}$$

$$\theta_3 = \cos^{-1} \frac{l_2^2 + L_3^2 - R_1^2}{2l_2L_3} \quad (6.44)$$

$$\theta_6 = \cos^{-1} \frac{l_2^2 + L_3^2 - R_2^2}{2l_2L_3}$$

This set of equations now provides the necessary parameters to calculate the σ and η values.

$$\begin{aligned}\Sigma_1 &= \Theta_1 + \Theta_2 \\ \Sigma_2 &= \Theta_1 - \Theta_2\end{aligned}\tag{6.45}$$

$$H_1 = \pi - \Theta_3$$

$$H_2 = \Theta_3 - \pi$$

$$\Sigma_3 = \Theta_4 + \Theta_5$$

$$\Sigma_4 = \Theta_4 - \Theta_5$$

$$H_3 = \pi - \Theta_6$$

$$H_4 = \Theta_6 - \pi$$

(6.46)

This provides the full list of α , σ and η for each point in the obstacle for all of the inspection points along the length of link 3. This set of calculations provides the entire range of joint angles which causes a collision between all of the measured points in the obstacle and the entire arm.

Summary of T-Space to C-Space Conversion Methods

Both of these methods provide a suitable method of converting points in T-Space into a C-Space map. However, the first method requires a large amount of sorting of information to find which angle combinations fall within the range of each of the links and this takes a large amount of calculation overheads and time. Even when both methods use vector operations to reduce the number of calculations necessary, method 1 takes 10 times as long to execute for an object of 250 points as method 2, which only takes approximately 3 seconds for an obstacle containing approximately 16000 measured points. Clearly, the second method is much more efficient at handling the T-Space/C-Space conversion; therefore this method shall be used for the path generation technique.

Analysis of C-Space Objects

To determine how obstacles appear in C-Space, a single polygon has been used to generate a C-Space object. Figure 6-10 illustrates the shape of an impermissible region for a single polygon. In the diagram there are two illustrations. The illustration on the left shows the position of a polygon in relation to the maximum arm expansion radii in the plane of the first link. The black circles represent the inspection points on the polygon, which is placed in the X-Z plane. The red circle illustrates the maximum radius of the first link, the blue circle the maximum radius of the first and second links and the green circle the maximum radius of the entire arm in the X-Y plane.

The diagram on the right of Figure 6-10 illustrates the components of the impermissible region formed by this polygon in the expansion range of the arm. There are two distinct shapes on the diagram. The shape made by the red, dark blue and dark green regions represents the impermissible region caused by collisions with the arm when the first link is pointing in a direction range towards the polygon. Mirroring the colours of the maximum expansion radii in the diagram on the left, the red shape is the impermissible region caused by collisions with the first link, the dark blue shape is the impermissible region caused by collisions with the second link, and the dark green shape is the impermissible region caused by collisions with the third link. The other distinct shape, made up of the light blue and light green impermissible regions represents the collisions between the polygon and arm when the first link is pointing in a direction range which is opposite to the direction towards the polygon. In this case, the first link will never collide, but the second and third links, which have a longer length than the first link in this application, are able to extend back in the direction of the polygon and collisions occur. The light blue shape illustrates the impermissible region caused by collisions between the polygon and the second link, and the light green shape illustrates the impermissible region caused by collisions between the polygon and the third link in the case of the first link pointing in a direction opposite to the direction of the polygon.

The reason that the impermissible regions of the first and second links extend entirely through one or two dimensions of the space is because if the first link collides with the polygon, then regardless of the direction of the other two links, the collision occurs. So for a given α angle, all β and γ angles cause a collision. The

same is the case for the second link. If the second link collides, then for the α and β angles that cause the collision, the direction of the third link is irrelevant and the collision will occur for all γ angles.

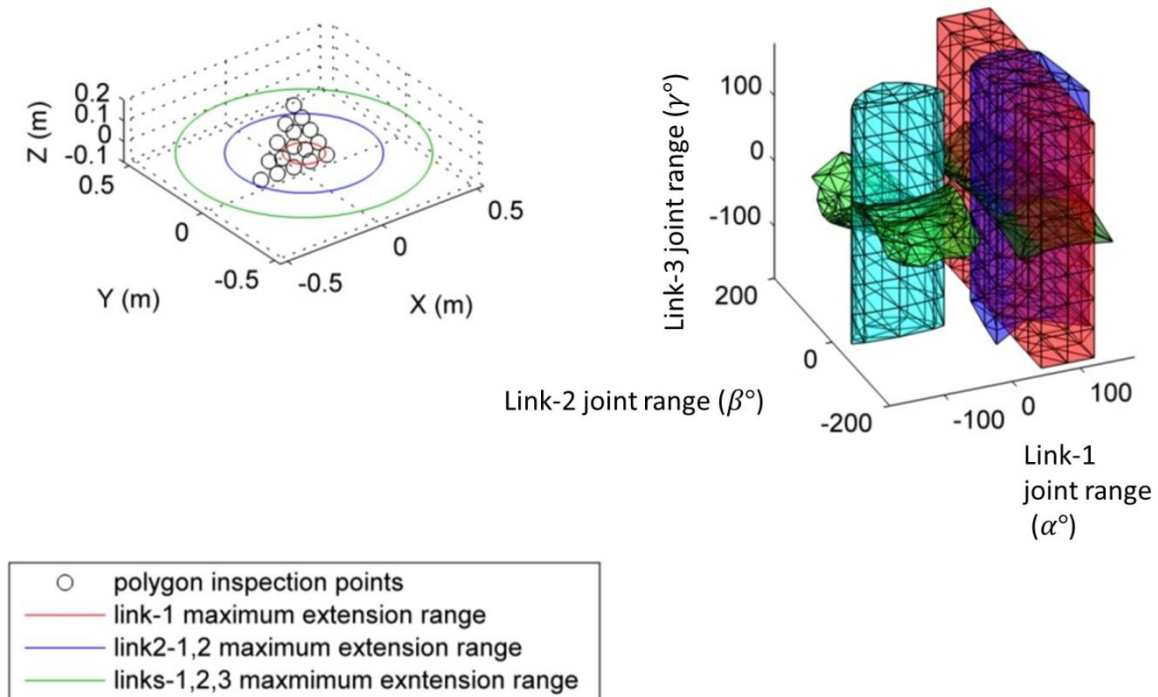
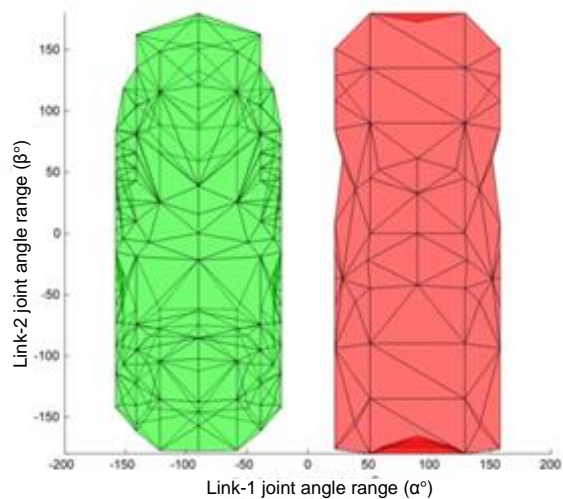
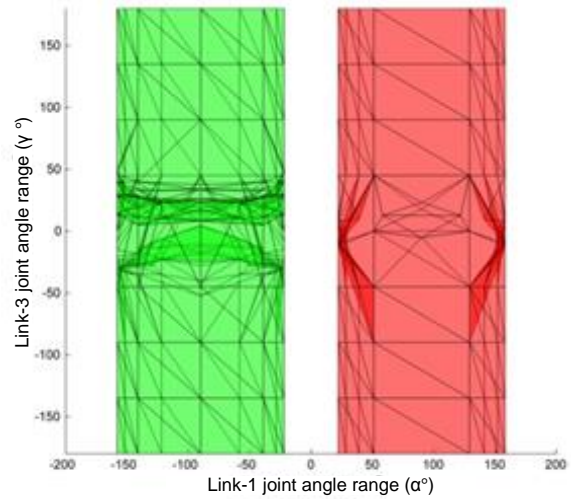


Figure 6-10: Impermissible region for a triangular polygon situated in the arm extension range for all three links.

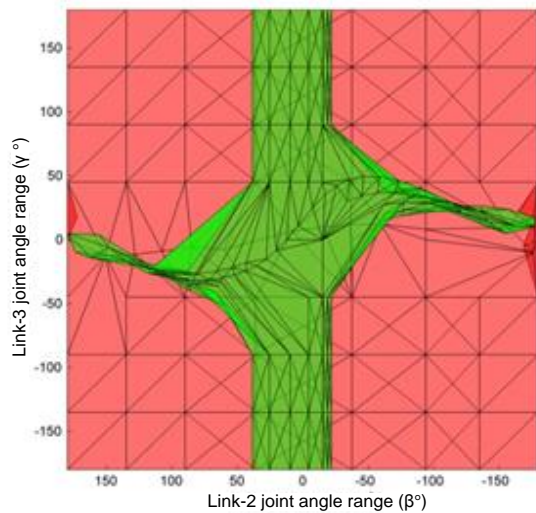
Figure 6-11 illustrates how the different impermissible regions for the different link lengths are combined to form the overall impermissible region for the polygon. These display the two distinct regions for α in the direction range of the polygon in red, and in the opposite direction range of the polygon in green. In this set of diagrams it is particularly clear that the regions completely fill the space in some dimensions. Figure 6-11 shows that the impermissible regions extend completely through the β angle dimension and the γ angle dimension for certain ranges of α .



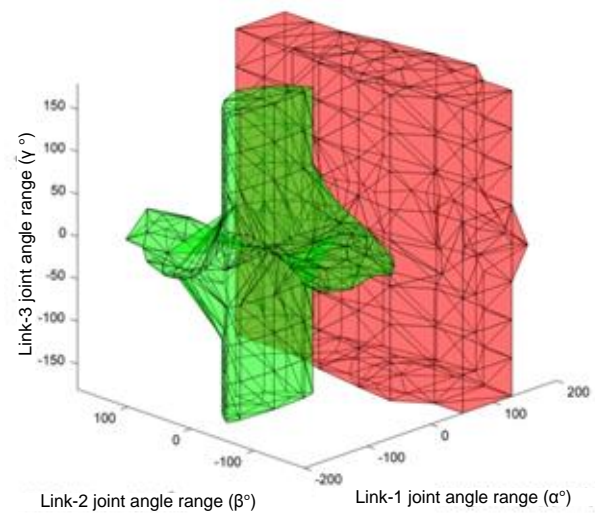
(a) View of the α - β plane.



(b) View of the α - γ plane.



(c) View of the β - γ plane.



(d) View in all three servo angle dimensions.

Figure 6-11: Impermissible region formed by inspection of a single polygon, as seen from various directions.

Given that collisions with the first link completely fill the C-Space in the σ and η dimensions, this will prevent the arm from navigating a large amount of the Task Space. For this reason, future obstacles used will fall outside of the range of link 1 to allow for navigation about the space.

6.1.3 Expand Impermissible Region to Permissible Boundary.

For each obstacle, a region in the C-Space exists that constitutes an impermissible region for the combinations of control inputs. These forbidden areas can be plotted as a three-dimensional graph, using the angle dimensions as the X, Y and Z dimensions of the graph. Moving the arm into a geometry whereby the joint angle combination falls inside this impermissible region would cause a physical collision. To prevent any collisions with the impermissible region formed by the C-Space obstacle, the impermissible region can be expanded to form a new shape which is slightly larger than the original C-Space shape.

By expanding the impermissible region slightly a new region is created, the boundary of which is unobstructed by the obstacle. When this new expanded shape is bounded it will form a boundary that the arm can touch that will not cause a collision provided that the arm does not cross inside it. To expand the impermissible region of the obstacle each point must be expanded to a sphere (or a hemisphere), or similar shape as will be explained, to provide a safe boundary around each point. To carry out this expansion, the required action is for duplicate versions of the C-Space points to be added to a set of vectors which contain a translation by the amount which corresponds to several limitations of the system.

The first limitation of the system is the error in the LIDAR sensor, which could cause the measurements to be off by a small amount (calculated to be in the region of 1.3×10^{-5} m or $1.35 \times 10^{-5^c}$ or $7.76 \times 10^{-4^o}$) which will form part of a boundary region which will be expanded around the impermissible region. The second limitation is the resolution of the sensor. Regardless of how small this resolution is, the sensor may miss the very edges of obstacles when scanning for points in range. This is shown in Figure 6-12. The red lines on the obstacle are areas where the LIDAR sensor may not detect that there is an obstacle; r_θ is the angular resolution of the LIDAR, r_s is the relative spatial resolution of the sensor and the red lines on the obstacle are the regions where the sensor has not detected any object.

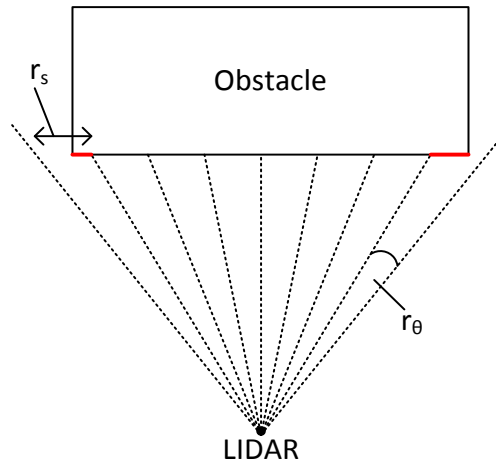


Figure 6-12 Effect of resolution of the LIDAR sensor on the detection of the corners of an obstacle.

To prevent this from becoming a problem, the permissible boundary expansion can also be expanded by the angular resolution of the sensor, since this will ensure that any undetected parts of an obstacle will not be collided with. The third limitation involves the accuracy of the joint servo encoders. If there is any noise or error in the measurements taken by the joint then the arm will not be located where the guidance method thinks that it is, and there may be a collision. A solution to this is to add a value to the permissible boundary around obstacles which is equal to the maximum measurement error of the servo sensors, hence avoiding a collision: The fourth limitation that must be taken care of is the steady state error of the joints in the dynamic model, which in Chapter 5 was shown to be 2° or 0.035^c . This means that even when the system displays the dynamics presented in Chapter 5, there will still be no collisions.

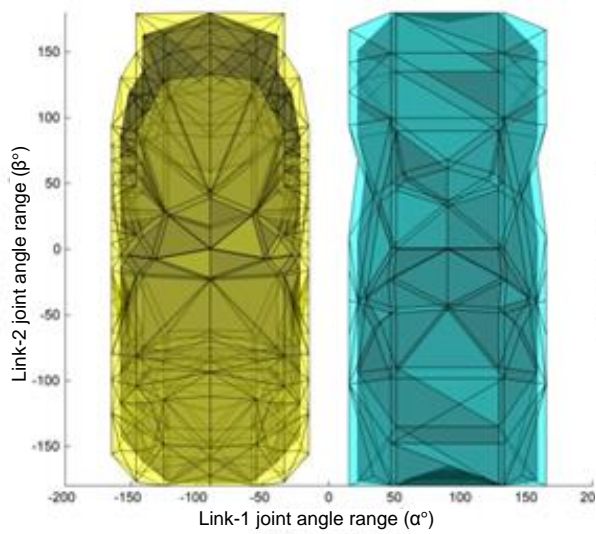
This vector appears as shown in Equation (6.47) where C_{points} is the vector of points in C-Space, $C_{points2}$ is the new vector of points in C-Space, e_l is the angular error in the LIDAR sensor, r_θ is the angular resolution of the LIDAR and e_s is the measurement error of the servo encoders. This new vector provides a cluster of points with which to create a node graph from. In this case the angular error in the LIDAR is negligible and the since the measured points in the obstacle are simulated, the entirety of the obstacle has been detected by the LIDAR. The servo encoders in

this case are also considered to be ideal. For this reason e_l , r_θ and e_s will be considered to be 0.

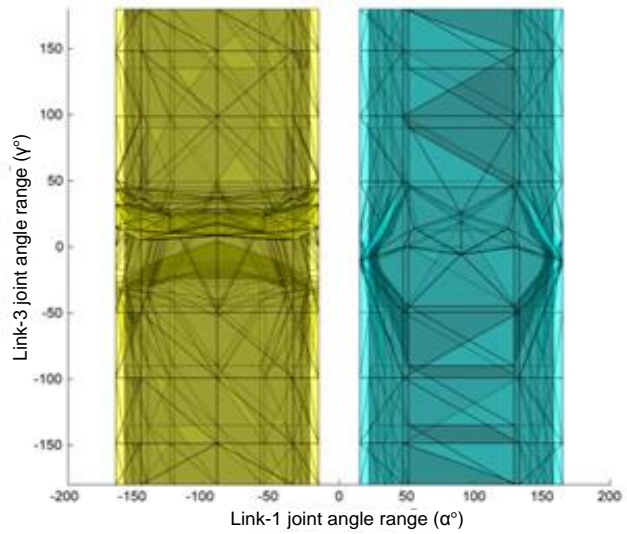
$$C_{points2} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \\ 0.7071 & 0.7071 & 0 \\ 0.7071 & -0.7071 & 0 \\ -0.7071 & 0.7071 & 0 \\ -0.7071 & -0.7071 & 0 \\ 0.5774 & 0.5774 & 0.5774 \\ 0.5774 & -0.5774 & 0.5774 \\ -0.5774 & 0.5774 & 0.5774 \\ -0.5774 & -0.5774 & 0.5774 \\ 0.5774 & 0.5774 & -0.5774 \\ 0.5774 & -0.5774 & -0.5774 \\ -0.5774 & 0.5774 & -0.5774 \\ -0.5774 & -0.5774 & -0.5774 \end{bmatrix} \quad (6.47)$$

$C_{points} + (e_{ss} + e_l + r_\theta + e_s)$

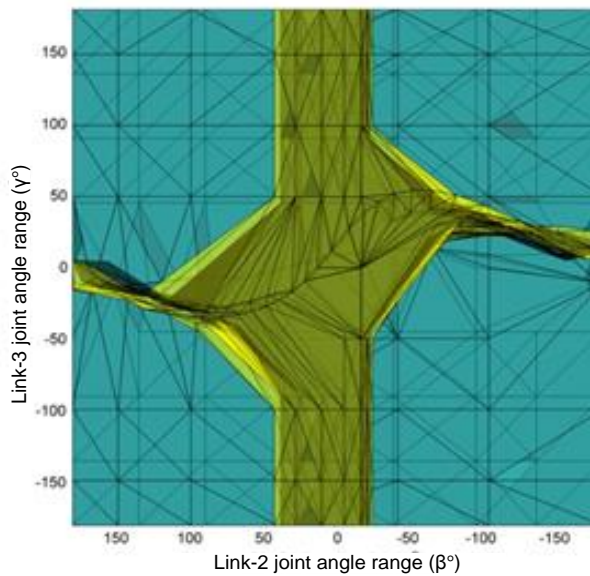
Figure 6-13 illustrates how the impermissible regions are expanded to create a permissible boundary around them. In the diagrams the blue shape represents the impermissible region for the arm with the first link in the direction of the polygon and the yellow shape the impermissible region for the arm with the first link in the opposite direction of the polygon. The darker region in the shapes are the impermissible regions and the lighter enclosing shapes are the permissible boundaries that have been created by expanding the impermissible regions.



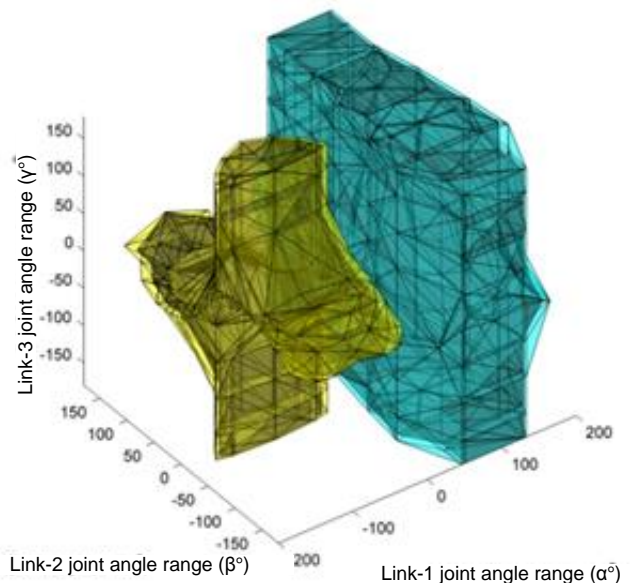
(a) View of the α - β plane.



(b) View of the α - γ plane.



(c) View of the β - γ plane.



(d) View in all three servo angle dimensions.

Figure 6-13: Expansion of the impermissible region to create permissible boundary around the impermissible region seen from various directions.

6.1.4 Node Graph Formation

This part of the method deals with the formation of a node graph in order to be able to execute a path generation algorithm and generate a path. Two methods are presented here. The first is to manipulate the permissible boundary regions to be able to use them to form a node graph and plot a path around them. The second

method is to generate a grid of nodes over the entire space and remove node from this graph which fall inside the permissible boundary regions.

In either case, the permissible boundary regions still contain all of the C-Space points, including those which fall inside the boundaries. These superfluous points can be removed in since they are not necessary. Also, the generation of surfaces that bound these points is necessary in both methods. The first requires the use of the edges of surfaces since they represent vertices connecting the nodes of the permissible boundary. The second method requires the surfaces since it analyses which points in the node graph of the entire space fall inside the closed shapes formed by the surfaces.

Convex Hulls and Alpha Volumes

Due to the extensive inspection of each obstacle, a lot of the collision points will find themselves inside the impermissible region for that obstacle. These points must be removed before a node graph is generated, otherwise they will be connected to the graph and this will allow the path generation algorithm the ability to use them in any path it generates, risking a collision. To solve this problem a technique called an alpha volume is used. This is the same as a convex hull, which is explained in the next paragraph, but with another extra function which is described immediately following the convex hull.

Convex Hull

A convex hull is a shape or volume which forms the boundary of a cloud of points. This can be done in two or three dimensions. In the case of the map generation technique described here, the method is used in three dimensions; however, for the purposes of outlining the method, an example in two dimensions will be used.

Consider a farmer's field with fence posts driven into it at random points across the field and a rope is tied to one at the outside edge. The farmer then walks with the other end of the rope all the way around the field and back to the first post. This creates a boundary with the rope that, if pulled tight will form a convex, irregular

polygon with a high number of sides that entirely bounds the posts in the field. The corners of this shape are the posts on the outside edge of the set of posts. This is outlined in Figure 6-14 and Figure 6-15.

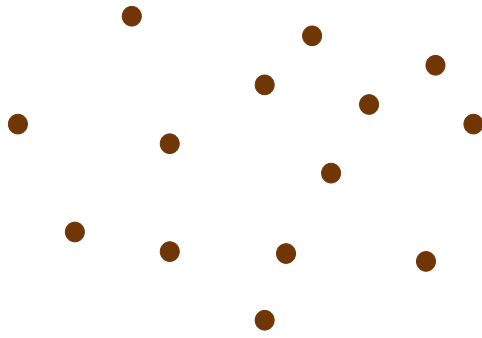


Figure 6-14 A 2-D pattern of inspection points for illustration of the convex hull technique for bounding a region of points.

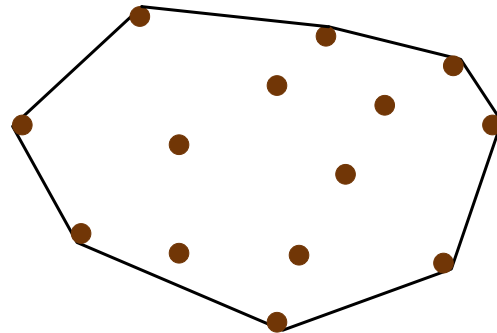


Figure 6-15 The same 2-D pattern of inspection points has now been bounded using the convex hull technique.

As can be seen by these figures, the black line has completely encircled the brown circles (our fence posts). This can be done by the following method:

- Start with the left-most point in the group and take a vector from it in a leftward direction.
- Find the point in the remainder of the set, the vector to which has the smallest angle from the original vector, when the angle is taken clockwise. This is the next point; store it in a list with the previous ones.
- Take the new vector as the starting direction vector and repeat step 2.
- Repeat step 2 and 3 until the next point is the original one.
- The points in the list represent all of the points that bound the set.

This can be seen in Figure 6-16

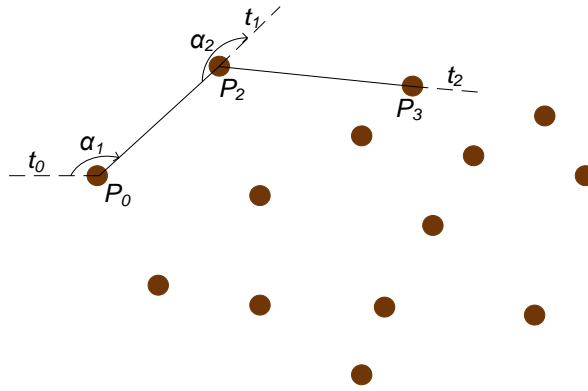


Figure 6-16 Method by which the complex hull technique works.

This is a very good start; however it is only useful if the distribution of points forms a convex shape. If there is a concave area to the shape, then the algorithm will ignore it. For example in Figure 6-17 the black line clearly ignores an area where the shape is concave, but the red line is a more accurate representation of the shape. This is done using an alpha shape, or volume in three dimensions.

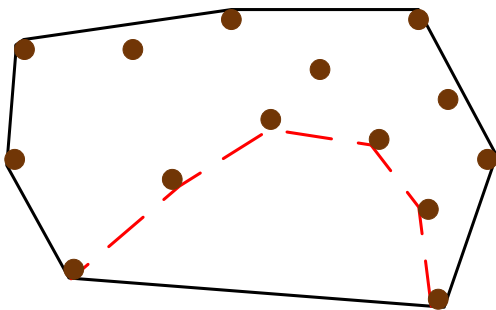


Figure 6-17 Limitations of the convex hull technique and the result of using the alpha hull technique.

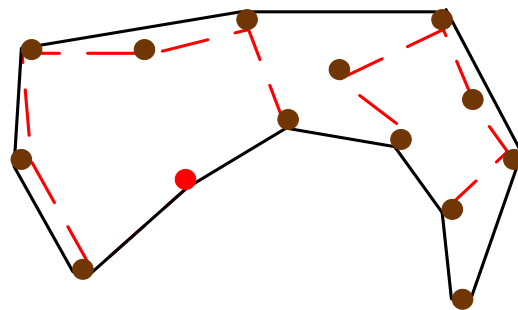


Figure 6-18 Limitations and compromise required when using the alpha volume technique.

Alpha Shapes/Volumes

An alpha shape follows the same principal as a convex hull; however it also has a tunnelling method to detect convex areas of a shape when a certain threshold is achieved. The extra addition to the method is to investigate whether the distance from the current point to the next one found by the concave hull method. If there is

another point which is closer with a larger direction angle it will use that as the next point and choose its next vector to be back along the previous line. This is called tunnelling and requires a tunnelling distance to check as the threshold value. There is a drawback to using this extra part of the algorithm however, depending on the required task. If the tunnelling distance is set too large then the algorithm will continue to function in the same way as the convex hull and not detect any concavity in the shape. If the tunnelling distance is set too short then the algorithm will begin to pick up gaps in the shape where none exist. This is shown in Figure 6-18.

As can be seen in the figure, the algorithm tunnels into areas that it should not, and even decides that one point is not connected to the rest. This is dangerous if the tunnelling distance is not calibrated correctly, especially when in the case of the map generation algorithm described in this thesis. Here the points are a spread of inspection points across a continuous shape. The alpha shape algorithm could conceivably decide that a surface on the shape does not exist in the control angle domain, and subsequently there are surfaces inside the object, which could lead to a potential collision.

This algorithm can be used in three dimensions, which allows it to generate surfaces around objects, forming a volume in the same way as it forms a shape. This is called an alpha volume. If it carries out this function to the two shortest angle points in three dimensions it can form a triangular polygon, which can be used later in the map generation.

Method 1 – Node Graph from Permissible Boundary Alpha Volumes

This method requires that each of the permissible boundaries are checked for overlapping areas, and if so then the permissible boundaries are merged with one another. Following this the remaining permissible boundaries are connected to one another by their closest nodes. Finally, the C-Space version of the robotic manipulator end effector start and desired end locations are calculated and connected to the resulting node graph.

Merging of Overlapping Permissible Boundaries

Due to the nature of the environment there may be places where the permissible boundary regions for two obstacles may be very close to one another. When the regions are expanded to form the permissible boundaries it may be that these regions overlap, and so they must be combined before a node graph is generated so that nodes on the edge of one permissible boundary that find themselves inside another are not included on the graph. This problem is solved simply by checking whether there are points from different impermissible regions within a certain distance of another impermissible region and if so then combining all of the points from the two regions into one array, effectively turning them into one region.

Conversion of Object Permissible Boundaries into Node Graphs

Each permissible boundary is required to be converted to a node graph for use with a pathing algorithm to find the shortest path between nodes in the graph. This is done very simply. The alpha volume technique described in the previous chapter finds polygon shaped surfaces around the edges of a scatter of points. These are stored in a format known as a triangulation, which has two arrays. The first array is an n -by-3 sized array which contains the coordinate data of the points in the triangulation for n points in the object. The first column in the array is the coordinate in the first dimension, normally x , but in the case of the 'permissible boundaries' this is α . The second column represents the β coordinates and the third column the γ coordinates. This matrix is known as the index matrix, and is identical to the one used by the node graph that is generated. The second matrix is an m -by-3 triangulation matrix, where m is the number of polygons that make up the surface of the 'permissible boundary'. Each of the elements in a row corresponds to a row of the index matrix, each specifying the coordinates of one point. The three points that are specified make up one of the m polygons in the boundary.

Converting the triangulation into a node graph is a very small, but potentially memory intensive process, which has been dealt with later in this chapter. An matrix is created of n -by- n dimensions where each element is infinite. This is to become an adjacency matrix for the node graph. Before it is populated with costs, all of the

nodes are set to have no connections. The (1,1) to (n,n) diagonal elements are set to 0 as the angular cost from one point to itself is zero. Each row of the triangulation matrix is then taken and the elements specified within it are connected to each other in the adjacency matrix. This is done by calculating the resultant angle change between each of the points and setting these as the cost to travel between those nodes in the adjacency matrix.

$$T_1 = [13 \quad 290 \quad 165] \tag{6.48}$$

For example, if a row in the triangulation matrix was as presented in (6.48), then elements (13,290), (290,13), (290,165), (165,290), (13,165) and (165, 13) would be filled with the resultant angle change between the points that these element identifiers refer to respectively.

This process is carried out for every ‘permissible boundary’ for the environment. It is at this point that a limitation of computer memory becomes a problem. The adjacency matrices for these permissible boundaries’ have the potential for several thousand rows and columns in them. In this case, there are limitations to various pieces of computer programming software, including Matlab, which cannot process extremely large matrices so there is a limit to the size that the adjacency matrices can become. This has the potential to create a problem when several adjacency matrices are combined, and a solution has been found for this issue. The solution is dealt with later in the chapter.

A second issue that only becomes clear at this point in the construction of the adjacency matrices is as follows. The space in which the map is built is circular. This means that where the impermissible region has dimensions of $-\pi$ to π radians, a continuous shape is formed that is circular in that dimension, exactly like the space. As $-\pi$ and π are both 180° , the region simply exits off one side of the map and re-enters on the other only to travel completely across that dimension. This in itself is not an issue as that means that there is a potential for some regions that are technically accessible to be inaccessible due to the impermissible region cutting off access completely to that area, which is physically possible in the real world. Take a

closed box for example. Theoretically the arm could be inside the box, but if all sides of the box are closed then the inside is inaccessible from the outside and vice-versa.

The issue comes in that the techniques used so far do not know that the space is circular and to the computer $-\pi \neq \pi$. This means that when the impermissible region stretch completely across one dimension, the alpha volume method connects up the sides at the ends of the shape where the coordinate in that dimension is either $-\pi$ or π radians.

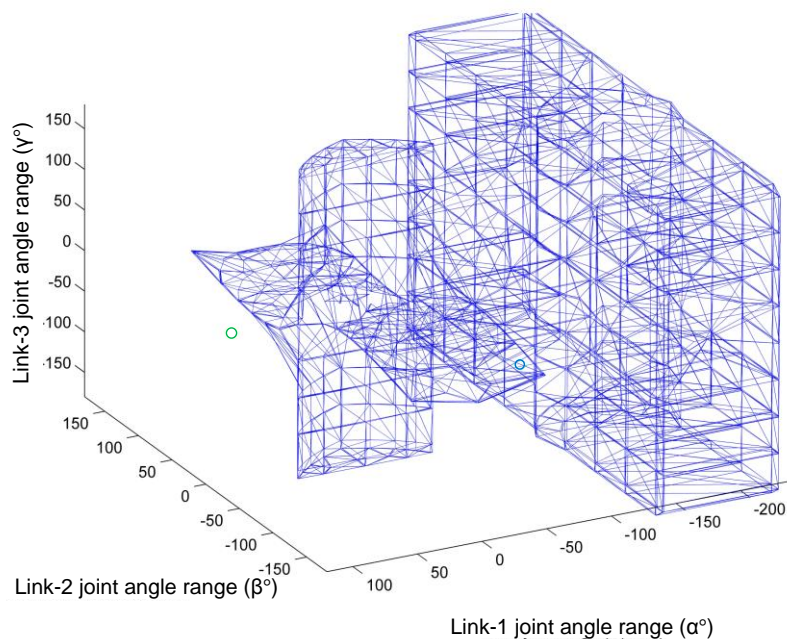


Figure 6-19: Permissible boundary surface converted into a wire frame representing the vertices in a node graph in the three servo angle dimensions.

The solution to this problem is as follows: If the shape covers one entire dimension then expand the shape by several radians at each end; i.e. if a point has a value of π , then duplicate it and add 0.1 radians to the duplicate in that dimension. If negative π then duplicate and subtract 0.1 radians to the duplicate.

- Carry out the Alpha Volume generation method again to get a new triangulation.
- For all points that are greater than π or less than $-\pi$, then do not add to the graph.

- If there is a point at π or $-\pi$ in one dimension and another the other value of π then connect them up in the node graph.
- The node graph for one 'permissible boundary' is shown in Figure 6-19.

The permissible boundary region is used to build connectivity and adjacency matrices which represent the node graph used to generate a path. The wireframe objects shown in Figure 6-19 are the representation of the vertices or connections between nodes in that graph.

Connection of Permissible Boundary Node Graphs for all Obstacles

Having generated a node graph for each 'permissible boundary and converted the end effector start and required locations into angular space, the map needs to be joined together. To reduce the computational load of large matrices this is done on a two level basis. Each of the 'permissible boundaries' has its centre point calculated and the boundaries are organised in order of increasing α position. This is because the largest variation in real world position is in the direction so the 'permissible boundaries' need to be connected in this direction to provide paths between them. Each boundary is connected to the next as a single node in the list in the top tier of the graph. The last in the list is also connected to the first in the same way as they are next to one another in the space.

Method 2 – Whole Space Node Graph with Removal by Alpha Volume Intersection

The above method is able to generate a node graph in the space, however, there are several operations that are required to be carried out which are computationally and time intensive. The following method aims to reduce a large amount of the computational overheads in the above method. The following sets of calculations are removed from this method:

- Checking for overlaps between permissible boundaries.
- Searching permissible boundaries for the closest points between them to connect them together.

- Searching the edges of the space for paths which exceed the limitations of the space and checking permissible boundaries in case they exceed the limitations of the space.
- Having to create 2 different levels of node graph, a global node graph and local node graphs for each permissible boundary to prevent memory problems.

To achieve all of the above, this method only requires two processes. The first is to create a node graph which covers the entirety of the space as though there are no obstacles. The spacing of the nodes in the graph will again match the steady state error values found in the previous chapter. The second is to check whether any of the nodes in this graph fall inside the permissible boundaries of the obstacles in C-Space and if so, removing them from both the list of indices and the adjacency matrix for the space. The largest advantage to this method is that the node graph of the unobstructed space can be generated a priori and only loaded when necessary, saving large amounts of memory. Also, the number of nodes in this graph will only reduce since nodes are removed when they are found to be obstructed by obstacles. Figures Figure 6-20 to 6-24 illustrate the effectiveness of this method.

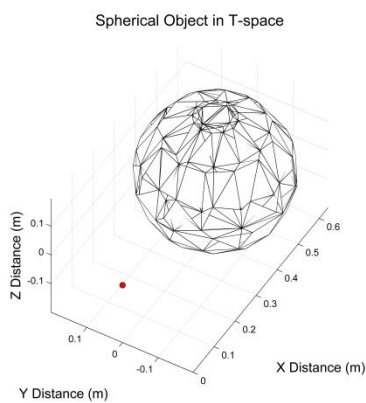


Figure 6-20 A sphere in T-space in relation to the base point of the manipulator arm.

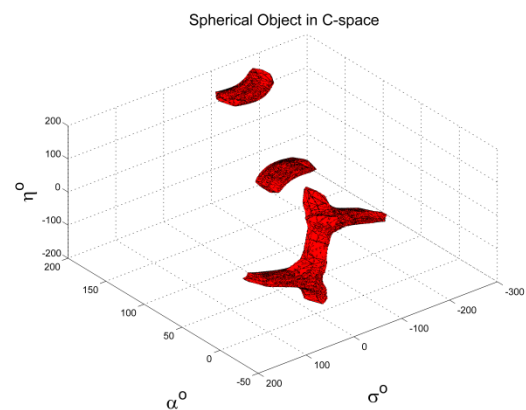


Figure 6-21 C-space representation of the sphere in relation to the manipulator arm.

Here the C-Space permissible boundary regions formed by a spherical object can be seen. The next stage in the process is to create a node graph for the accessible C-space. This is carried out by first creating a node graph for the space without any obstacles. As specified in Chapter 5, the accessible region of the manipulator arm is in the range of $-180 \leq \alpha \leq 180, -40 \leq \sigma \leq 180, -180 \leq \eta \leq 140$.

The choice of spacing between nodes is important and relates to two limitations, the first of which is related to the control limitations of the manipulator, the other is related to the memory capabilities of the computer carrying out the node graph construction. The first limitation is the steady state error of the joint angles. While this has been taken care of in the expansion of impermissible regions into permissible boundaries around which the arm can safely travel in section 6.1.3, the nodes in the graph which is being generated are direct control requirements of the joints and so if they have a smaller separation than the steady state error of the system then the arm will be situated at the location of two different nodes in reality and inside the guidance method. For this reason the spacing between nodes will be set as the steady state error of the joints as a minimum. In this case 2° or 0.035^c . The second limitation is that the computer which carries out the node graph generation is limited in terms of memory, and a node graph with a small spacing will have a very large number of nodes, increasing the size of the adjacency matrix in the computer memory, hence increasing overheads and run time. For example, with the angle range limitations for the manipulator arm used in this thesis, the size of the adjacency matrix is as follows for the angle spacing of $1^\circ, 2^\circ, 5^\circ$ and 10° .

Table 6-1 Numerical data showing the adjacency matrix size difference between node graphs of different node spacings.

Node spacing	1°	2°	5°	10°
Adjacency matrix size	79781×79781 $= 6.365 \times 10^9$	20091×20091 $= 403648281$	3285×3285 $= 10791225$	851×851 $= 724201$

Clearly, the smaller the spacing between nodes in the graph, the larger the adjacency matrix of the graph becomes, and this relationship is exponential. It is for this reason that the node spacing has been selected as 5° or 0.087^c to reduce the computational overhead significantly.

The obtained set of C-Space permissible boundaries can be overlaid over the node graph of the empty space. Figure 6-22 shows the It is clear that some of the angle combinations that cause collisions fall outside of this space.

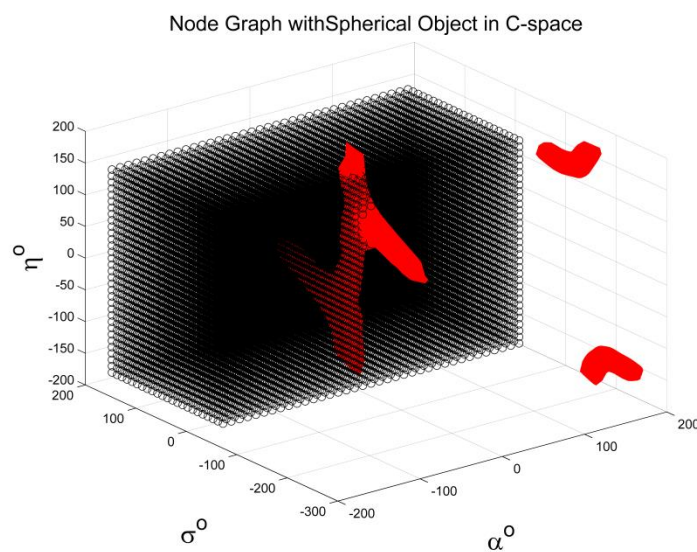


Figure 6-22 Sphere in C-space (red) overlaid over the unmodified node graph (black).

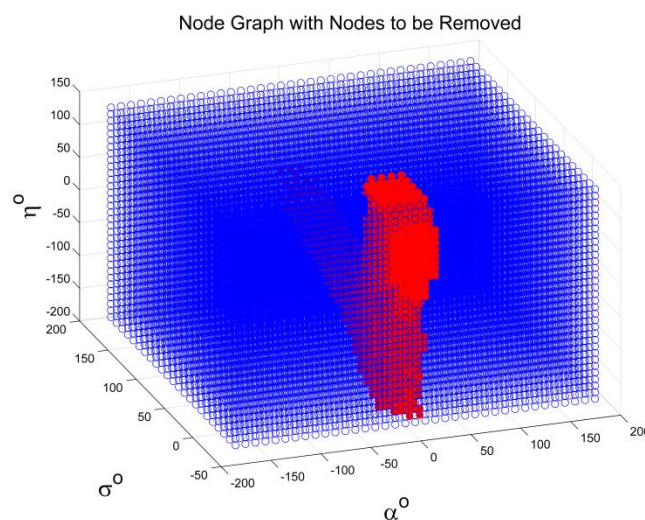


Figure 6-23 Nodes which collide with the sphere in C-space (red) to be removed from the remainder of the node graph (blue).

The triangulation shown in Figure 6-24 illustrates the space that the removal of the red nodes leaves behind. The triangulation itself is not used but displayed for illustrative purposes.

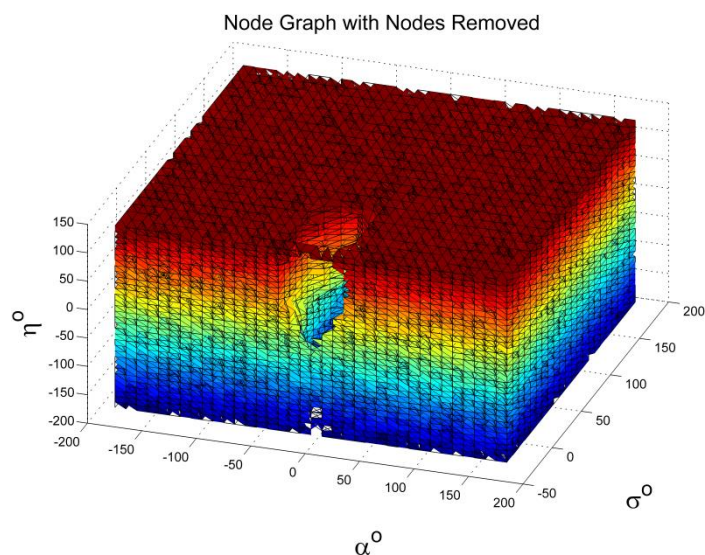


Figure 6-24 Surface plot of a triangulation showing the complete node graph with the colliding nodes removed.

The nodes in the graph of the empty space can be compared with the permissible boundary regions and any nodes that fall inside these regions can be removed. The red nodes in Figure 6-23 illustrate which nodes will be removed from the graph.

Summary of the Method

This method has advantages over the first method in that the preloaded node graph already defines the space and, therefore, there is a reduction in computing overheads because the algorithm does not have to search for areas of the C-Space obstacles which exceed the limitations of the operational space of the manipulator arm. There is also no computational overhead caused by the need to check each of the permissible boundaries for overlaps since any overlap will be taken care of when points are removed from the node graph. Again, the closest points between each of the permissible boundaries do not have to be found, further reducing the amount of

computation. Finally, there is no need to create two levels of node graph. This is because each individual node graph and list of index points stored separately cost more in terms of memory than a node graph for the entire space together.

It is for this reason that the second method will be used to calculate the node graph with which to generate paths through the T-Space.

6.1.5 Mapping of End Effector Start and Required Joint Positions

The end effector start and required positions have also to be mapped into the space to allow them to be connected to the node graph. As with all the other inspection points, when these points are mapped into the angle space, there are several potential solutions. The solutions with the smallest resultant angular change (i.e. closest together in the space) are chosen, provided that they do not appear inside one of the 'permissible boundaries'. If that is the case then that solution is inaccessible. If all solutions are inaccessible then the start or require point in Euclidean space is also inaccessible.

In Figure 6-19, the end effector start position and required end position have also been mapped. The green circle represents the angle combination of the start position and the red circle represents the angle combination of the required end position. Table 6-2 illustrates the change in domain from the Euclidean domain to the control domain.

Table 6-2 Change in domain from the Euclidean domain to the Control domain for the end effector start and required end points used in the path generation example for method 1.

End Effector Identifier	Point	Euclidean Domain	Control Domain
Start point (P_s)		$\begin{bmatrix} -0.0514 \text{ m} \\ -0.2915 \text{ m} \\ 0 \text{ m} \end{bmatrix}$	$\begin{bmatrix} 100^\circ \\ 0^\circ \\ 0^\circ \end{bmatrix}$

$$\text{Required end point } (P_r) \quad \begin{bmatrix} -0.0514 \text{ m} \\ 0.2915 \text{ m} \\ 0 \text{ m} \end{bmatrix} \quad \begin{bmatrix} -100^\circ \\ 0^\circ \\ 0^\circ \end{bmatrix}$$

The end effector start and required angle combinations are compared with the centre point of each 'permissible boundary' in the space to find which nodes in the graph they are closest to. These points are then connected to only the node that they are closest to. This completes the map of the environment, including the end effector start and required locations.

6.2 Summary of Environment Modelling

In this chapter a method of creating a navigable map of the environment has been presented which will allow the robotic manipulator system to safely pass around the obstacles in the environment. Once the obstacle data has been converted to C-space and the resultant impermissible regions expanded into safe boundaries, two methods of utilising the resultant information have been presented. The first was to create an alpha volume of each of the permissible boundaries around objects and then connect these alpha volumes together as a node graph. This method was very high in computational complexity due to the amount of manipulation that had to be carried out to make sure that it was contained in the region which is bounded by the limitation of the manipulator arm. Also the resultant node graph was limiting in terms of where in the space the manipulator could travel. The method was ensuring that the manipulator could track around the edges of obstacles but could only navigate by jumping between the edges of different obstacles. For this reason a second option was investigated. The second method uses a pre-calculated node graph of the empty space, which can then have any nodes which fall inside the alpha volumes for the accessible boundary regions removed to leave a node graph of the remaining accessible space. This method has an advantage over the other method in that the edges of the space are already bounded since there are no nodes located outside of the limitations of the manipulator arm, hence no extra manipulation of the node graph is required. Also there is no need to check each triangulation for any overlap since when two alpha volumes overlap, the first to be considered will remove all

nodes which intersect with it, and when the second alpha volume is considered, any nodes which would have been located in the overlap region of the two alpha volumes will have already been removed.

With a successful implementation of a node graph which contains the map of the environment in C-space, Chapter 7 will investigate methods of generating a path through the space before the complete method is validated by simulations in Chapter 8.

7 PATH GENERATION USING GRAPH THEORY

In the previous chapter of this thesis a method of converting obstacle data in T-Space into a safe, usable map in C-space has been presented. Having accessibility to a safe map of the environment is important since it provides a domain for a path to be generated through, allowing the manipulator arm to navigate from A to B without collision with obstacles. This chapter is responsible for the design decisions and implementation of a path generation method which will satisfy the requirement of a safe path through C-space. Figure 7-1 illustrates how the path generation method fits into the context of the guidance method as a whole.

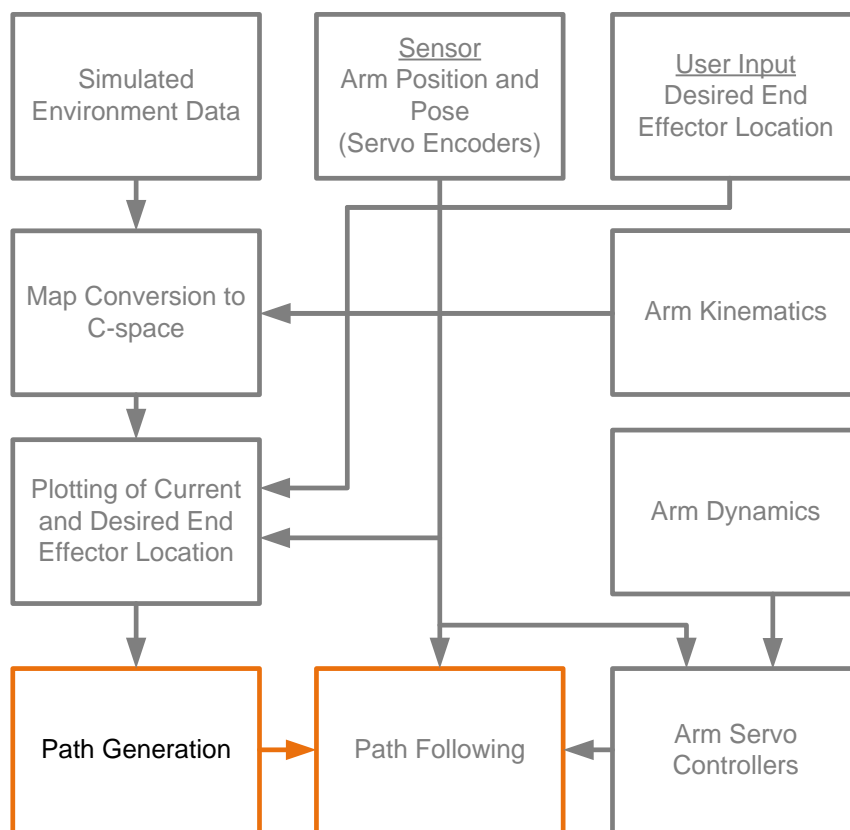


Figure 7-1 Flowchart showing the context of the path generation (orange) in relation to the overall guidance method.

7.1 Path Generation in C-Space

A path generation algorithm is required to find the path from the start to required angle combinations. Figure 7-2 shows how the node graph for one obstacle can be represented graphically in C-Space, with a resultant generated path. The path generation method will be detailed later in the chapter. This example is for the triangular polygon previously presented in this chapter.

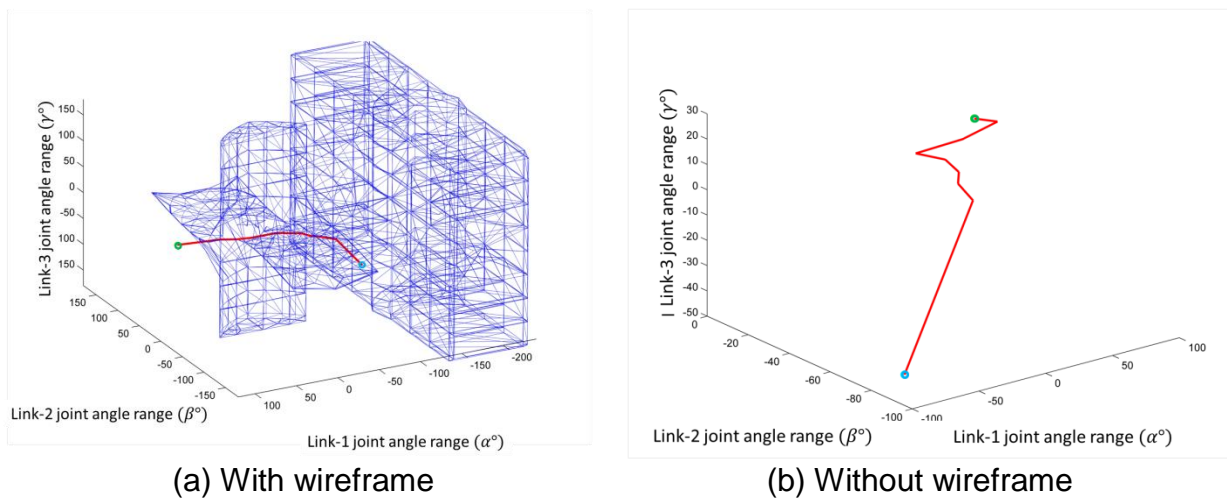
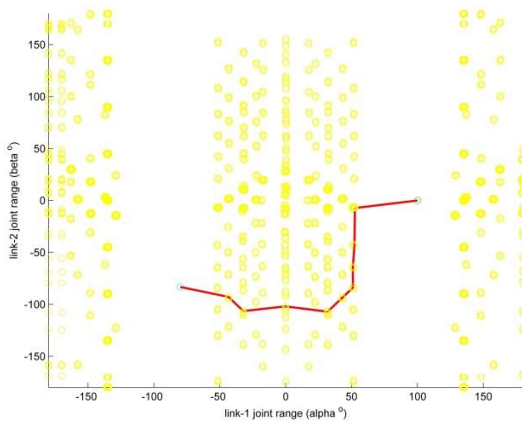
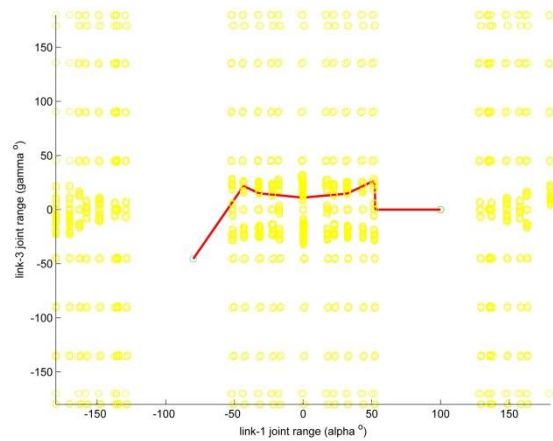


Figure 7-2: Generated path (red line) from an end effector start point (green circle) in the servo control domain to a desired end effector end point (blue circle) in the servo control domain.

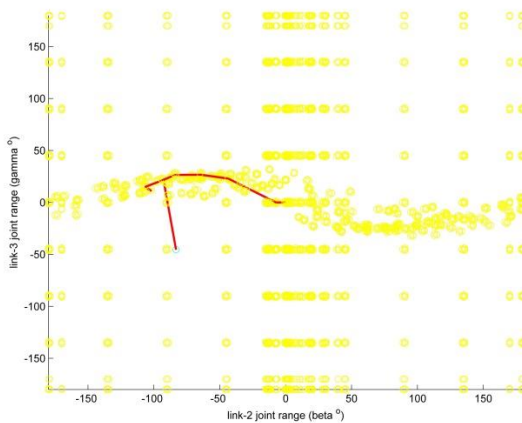
Figure 7-2 shows the shortest resultant angle change path from P_s to P_r through the servo control angle space. The red line represents the generated path which follows closely the edges of the permissible boundary layer.



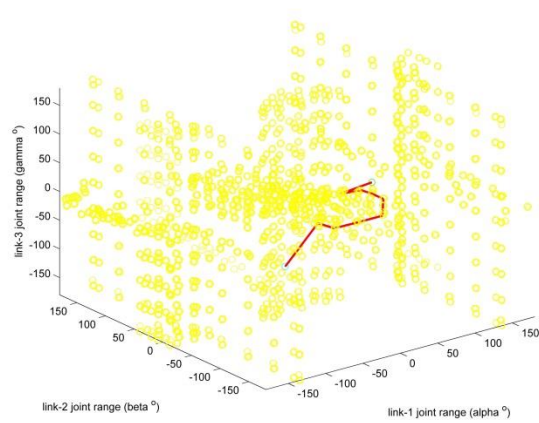
(a) View of the α - β plane.



(b) View of the α - γ plane.



(c) View of the β - γ plane.



(d) View in all three servo angle dimensions.

Figure 7-3: Generated path between the end effector start point (green circle) to the desired end effector end point (blue circle).

Figure 7-3 illustrates the path once the space has been normalised to a -180° to 180° space in α , β and γ . The red line represents the path, and the yellow circles are the nodes in the node graph of the permissible boundary around the impermissible regions. The dimensions of the servo control space have been normalised to show a -180° to 180° square space. As can be seen in the diagram, one of the two impermissible regions crossed the $-180^\circ/180^\circ$ boundary in the α -angle dimension and so half of the object appears greater than -180° and the other half as smaller

than 180°. This further illustrates the circularity of the space. This figure illustrates the complete path that the arm must take in terms of the joint angle combinations that will guide the end effector from its start location to its required end position, while avoiding the obstacle with the entire arm. The list of angle combinations for this path is given in Table 7-1.

Table 7-1: Joint angle combinations to guide the robotic manipulator in such a way as to drive the end effector from a starting position to a required position while providing effective avoidance for the entire manipulator.

Point ID	α°	β°	γ°
1	-100	0	0
2	-121.695	-14.1908	0.1828
3	-159.75	-42.4559	-49.5
4	165.3354	-27.9574	-49.5
5	144.0931	-10.6416	-49.5
6	124.7179	-4.6067	-11.838
7	100	0	0

Alternatively, a complete example using the sphere as previously presented. Figure 7-4 illustrates the path generated by path generation method around the obstacle in C-Space.

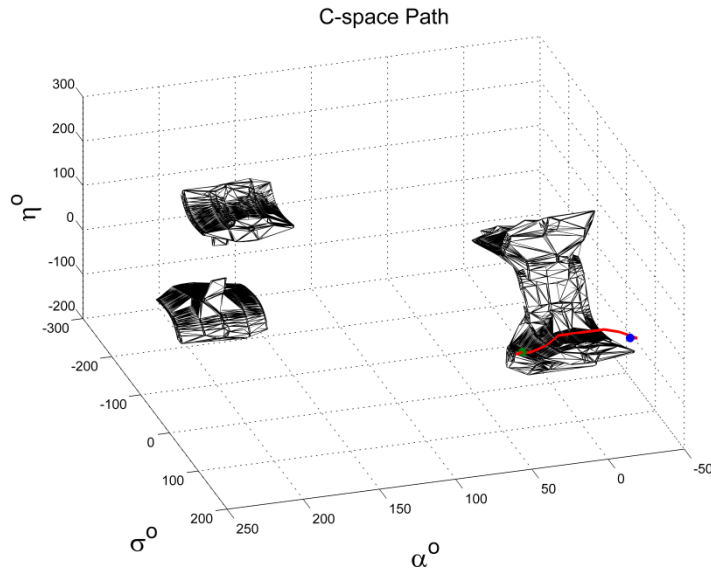
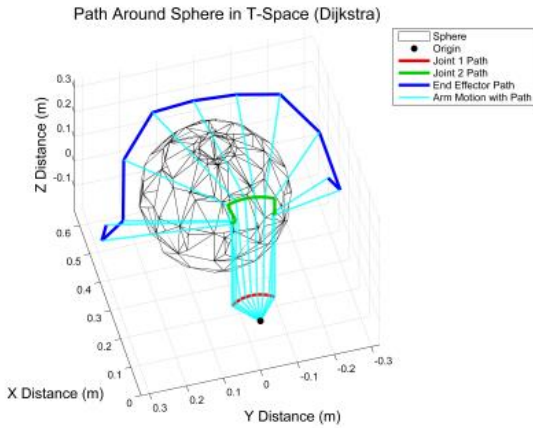


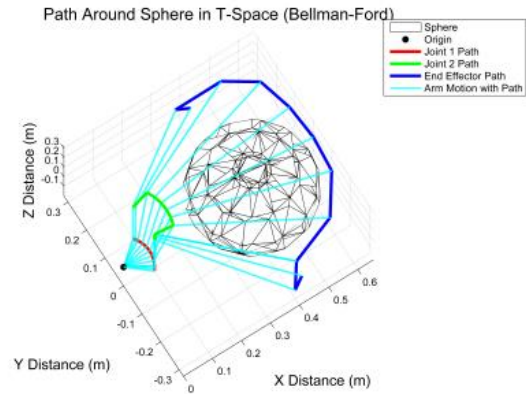
Figure 7-4 Spherical object in C-space with a safe path generated around it.

7.2 Comparison of Path Generation Techniques

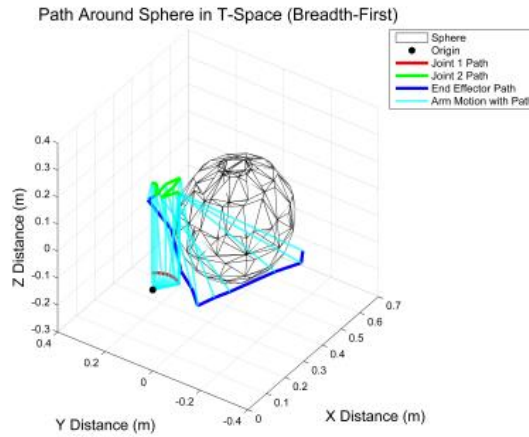
Plotting a path through C-Space in the quickest way possible, while still maintaining an accurate path around any obstacles is important. Based on the review of literature carried out in Chapter 2, three pathing methods have been selected for testing, Dijkstra's Algorithm, Bellman-Ford's Algorithm and a Breadth-first Search Algorithm. Figures 7-5 a to c show the same spherical obstacle from earlier in the chapter with a path generated around it for the same start and desired end effector locations using the three specified pathing methods, Dijkstra's Algorithm, the Bellman-Ford Algorithm and a Breadth-first Search of the node graph. In these figures there are 4 sets of lines. The red lines indicate the path made by the first joint, the green lines indicate the path made by the second joint and the dark blue line represents the path made by the end effector. The light blue lines show the geometry of the manipulator arm at each of the waypoints in the C-Space path.



(a) Dijkstra's Algorithm



(b) Bellman-Ford



(c) Breadth-first

Figure 7-5 Dijkstra's Algorithm (a), the Bellman-Ford Algorithm (b) and Breadth-First Algorithm (c) having plotted a path around the same spherical obstacle.

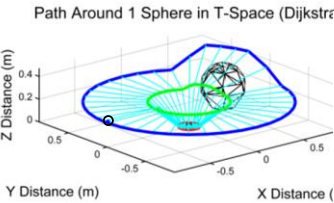
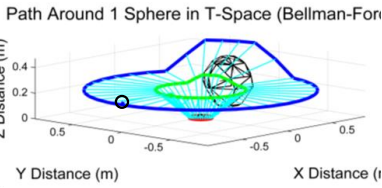
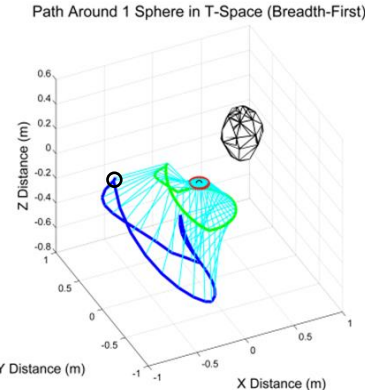
As can be seen from the figures, in this case all three pathing methods are able to generate a path around the obstacles. In order to make a choice of pathing method, all three must be compared in several scenarios. To compare the methods, they have been tested in several scenarios with the same start and end locations. Equation (7.1) shows the end effector starting location and Equation (7.2) the end effector demand location.

$$\begin{bmatrix} \alpha_s \\ \sigma_s \\ \eta_s \end{bmatrix} = \begin{bmatrix} -180^\circ \\ 45^\circ \\ 0^\circ \end{bmatrix} \quad (7.1)$$

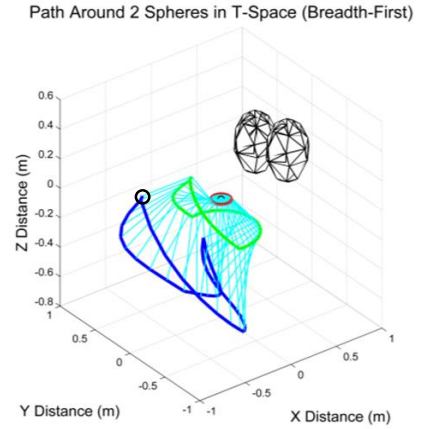
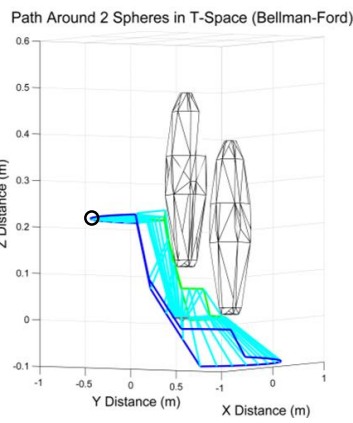
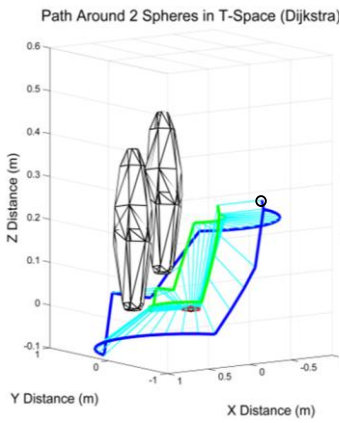
$$\begin{bmatrix} \alpha_d \\ \sigma_d \\ \eta_d \end{bmatrix} = \begin{bmatrix} 180^\circ \\ 45^\circ \\ 0^\circ \end{bmatrix} \quad (7.2)$$

This means that the general direction of the path will always be the same. Following this, obstacles will be added in the arc of the end effector motion with the unobstructed path in order to provide obstructions which the arm has to navigate about. Obstacles were added until there were 7 spheres obstructing the path from the starting pose to the end pose.

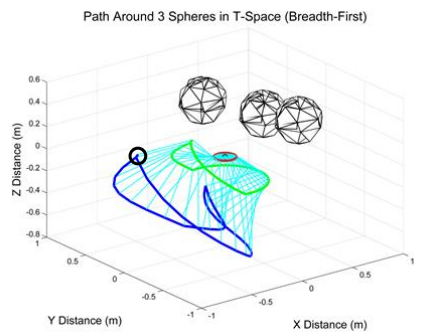
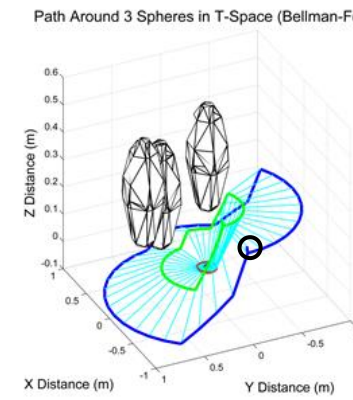
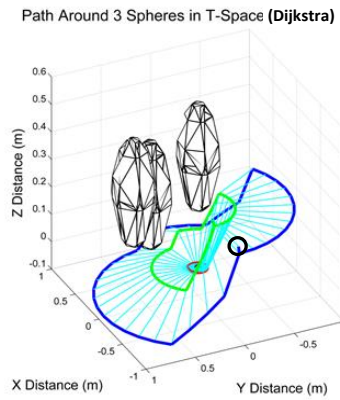
Table 7-2 Results from a preliminary investigation of the three selected pathing methods for different numbers of obstacles.

No. Obs.	Dijkstra's Algorithm	Bellman-Ford Algorithm	Breadth-first Algorithm
1			

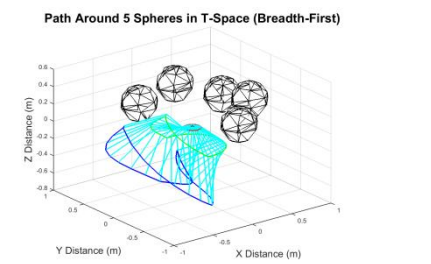
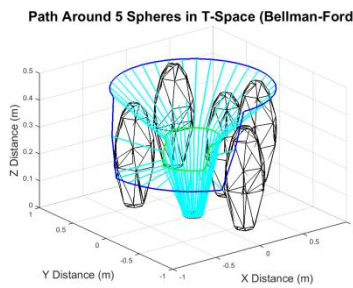
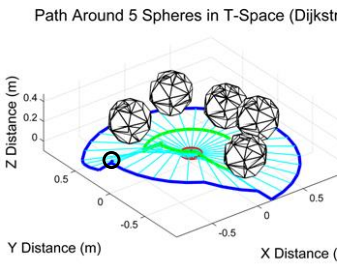
2



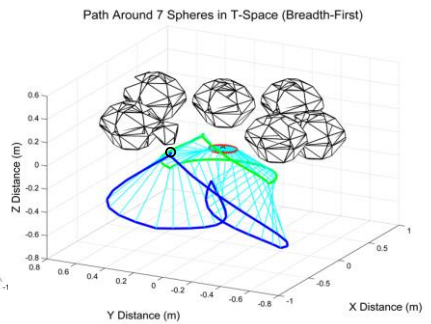
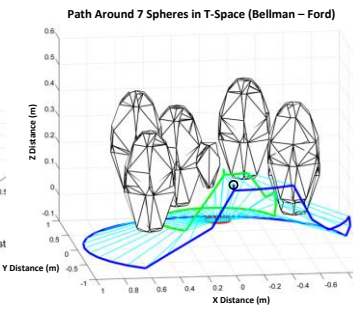
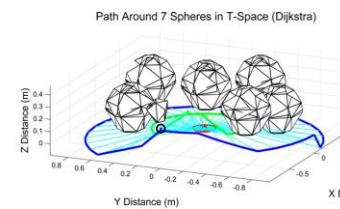
3



5



7



By inspection it can be seen that in all three of the above methods, the paths that are generated avoid collisions with the obstacles. Also, the general shape of the paths is similar for the Dijkstra and Bellman-Ford Algorithms, but a very different path shape is present for the Breadth-first Search method. An inspection of the run time for each of the methods given the number of obstacles provides interesting results. The figure below shows the run time of each algorithm for 1, 2, 3, 5 and 7 obstacles. It can be observed from the figure that the Breadth-first Algorithm consistently runs $2 \times 10^{-2} s$ to $4 \times 10^{-2} s$ faster than Dijkstra's Algorithm. The Bellman-Ford Algorithm was $2 \times 10^{-2} s$ to $6 \times 10^{-6} s$ slower again. This confirms the results presented shown by Table 2-3 and Table 2-4 in the review of literature in Chapter 2. There is a slight trend of all three algorithms speeding up with larger numbers of obstacles, and this can be explained by the smaller number of accessible nodes to explore when more of the nodes are removed from the graph because they lie inside an obstacle. The Breadth-first Algorithm is faster than the other two methods because it is designed to explore outwards from the starting point first and in this case will stop as soon as it finds the target node. This means that it will find the path with the smallest number of vertices, regardless of cost. The other two methods will continue to search the graph for paths to the goal with shorter path costs, even if the resultant path contains more vertices.

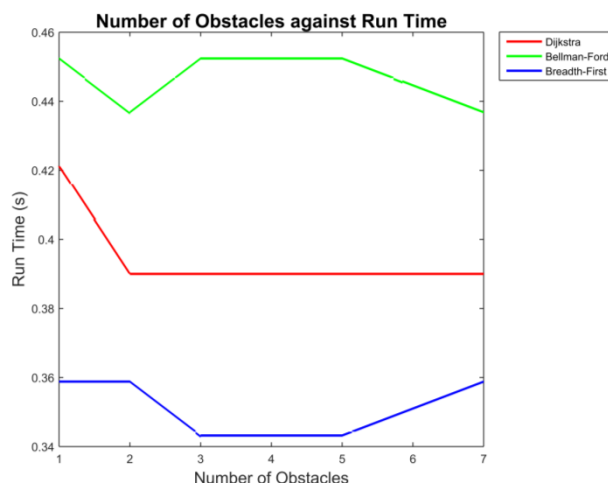


Figure 7-5 Algorithm run time for the selected path planning algorithms for different numbers of obstacles

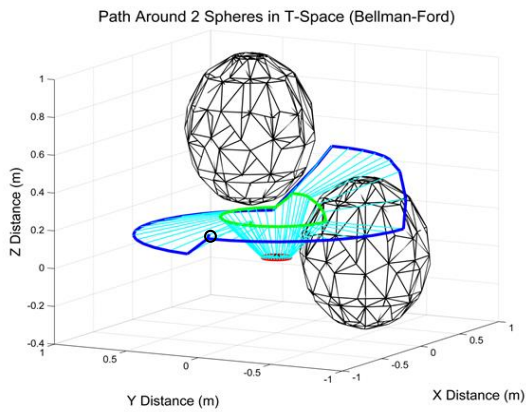
Regardless of the reasons why one method is faster than the others, all three methods find a path within 0.1 seconds of each other, and since only one path has to be generated, this difference is very small in regards to the run time of the entire algorithm. A further set of tests will increase the size of the obstacles to reduce the distance between them. It is also of note that in all of the above paths, the algorithms generated a path around the entire cluster of obstacles rather than through the cluster. Making the obstacles larger will increase the distance of the path if it navigates around the perimeter of the cluster rather than through the cluster, potentially encouraging the algorithms to travel through the smaller gaps between the obstacles.

Each of the three methods will be investigated with 2 and 4 spheres with a Euclidean distance between the origin of the manipulator arm and the centre of each sphere of 0.5657 m in the XY-plane, at coordinates of [0.4,0.4] m, [0.4,-0.4] m, [-0.4,0.4] m or [-0.4,-0.4] m and either 0.6 m or 0m in the Z-direction. This provides a central coordinate for each of the obstacles, which are 1 m apart from the adjacent obstacles, where the radius of each sphere can be specified to allow for collisions with both the 2nd and 3rd links and also provide a known separation between each obstacle.

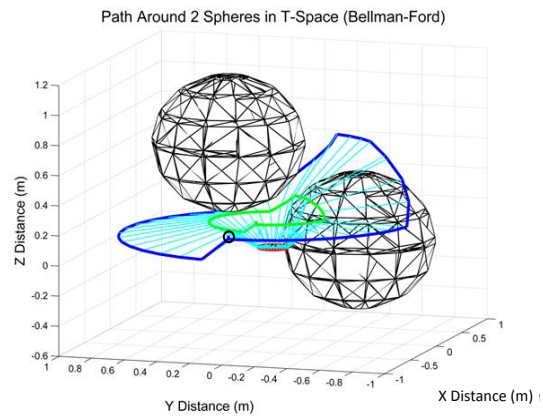
The investigation will consist of the same end effector start and desired geometries in terms of joint angle and there will be two obstacles with centres at [0.4,0.4,0.6] and [0.4,-0.4,0]. The radii of the two obstacles must be varied to change the spacing between the two obstacles. A mathematical analysis can be used to calculate the range of radii which should be investigated. At this point it is important to consider that the permissible boundary expansion of 2° gives a total of 4° between the two obstacles. For the maximum extension of the arm this translates into 0.067 m, and for a Euclidean distance of approximately 0.64 m, which is the distance to the halfway point between the two obstacles, the 4° safety margin translates to approximately 0.05 m. This means that for the arm to be able to pass, the maximum radius of each of the obstacles will be 0.475 m, since $0.05 + 2 \times 0.475$ is 1 m. In order to test this, each of the methods will be tested for obstacle radii of 0.47 m, 0.475 m and 0.48m.

For these radius selections the distance between the surfaces of each of the objects is 0.94 m, 0.95 m and 0.96 m, and with the expanded permissible boundary, the safe spacing between objects is 0.01, 0 m and -0.01 m respectively. This means that for the three object radii selections, the path generation method should be able to generate a safe path between the obstacles in the first case, theoretically generate a safe path between the obstacles in the second case since the safe boundaries touch but do not overlap, and in the third case, since the boundaries do overlap, there should not be a safe path between the obstacles.

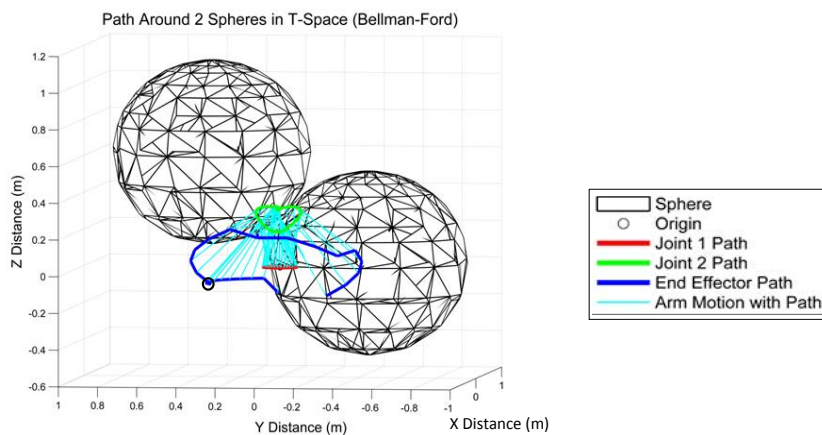
7.2.1 Bellman-Ford



(a) 0.47 m Obstacle Radius



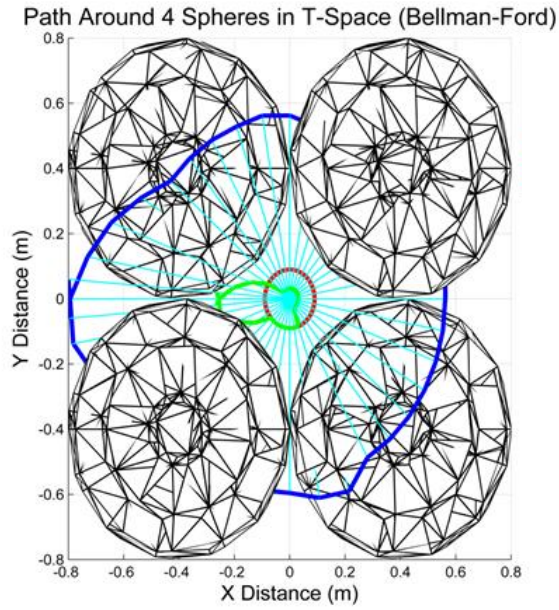
(b) 0.475 m Obstacle Radius



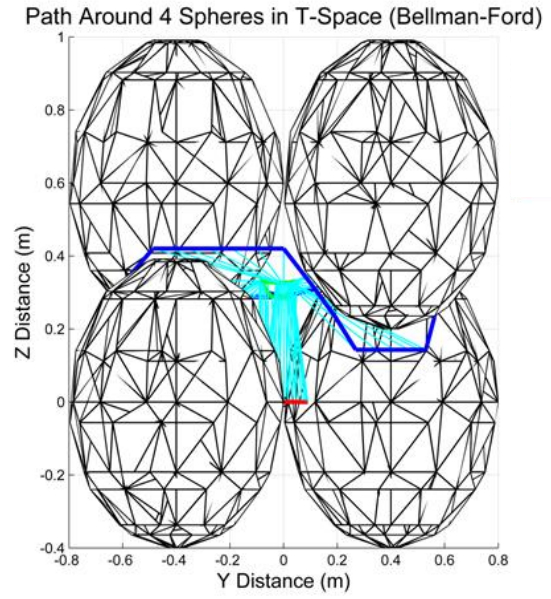
(c) 0.48 m Obstacle Radius

Figure 7-6 Results of the path generation around two obstacles with 0.47 m, 0.48m and 0.49m radii using the Bellman-Ford Algorithm.

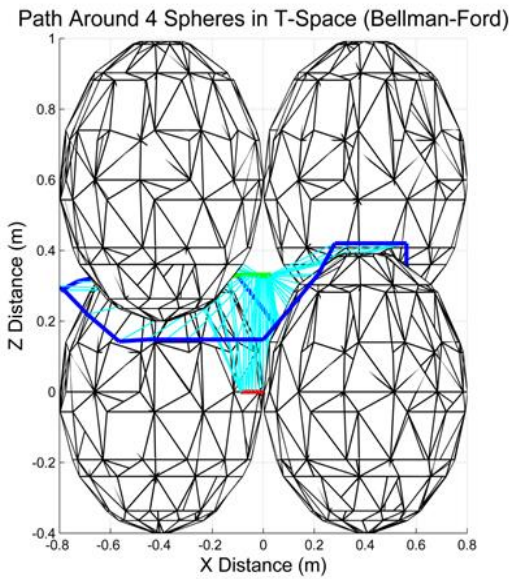
Figure 7-6 displays the results for the Bellman-Ford Method. This set of results shows that the Bellman-Ford method can find a suitable path with sphere radii of 0.47 m and 0.745 m. For the sphere radii of 0.48 m, the path generated collides with one of the obstacles. This can be observed where the dark blue line showing the path of the end effector disappears inside the sphere of an obstacle. This set of results agrees with the predicted outcome prior to the path generations. A further test of the ability of this method is to carry out the same path generation but for 4 obstacles with radius of 0.475 m rather than 2 obstacles. Figure 7-7 illustrates that the Bellman-Ford Algorithm is capable of generating a path around the obstacles.



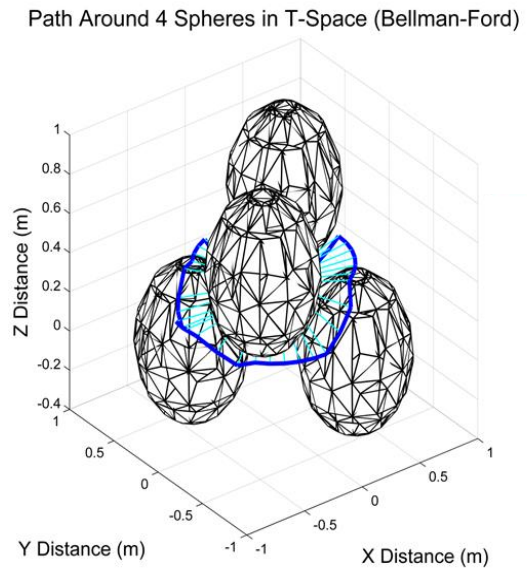
(a) X-Y View



(b) Y-Z View



(c) X-Z View



(d) 3-D View

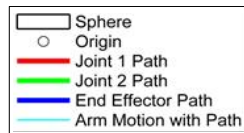
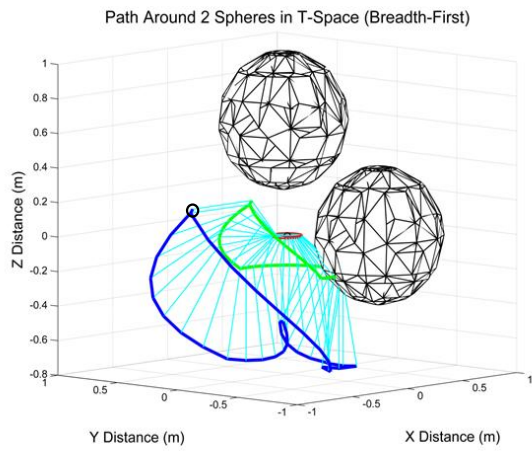
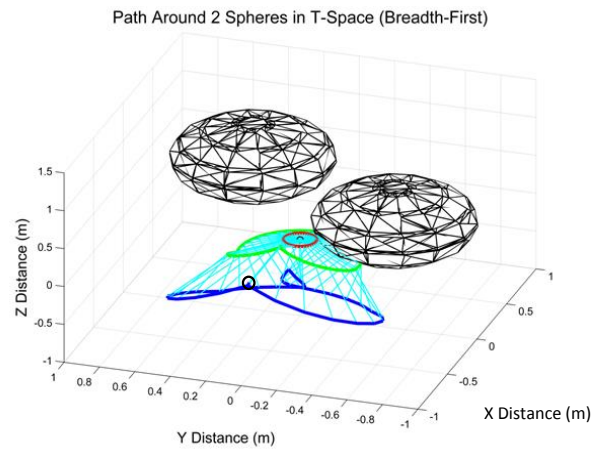


Figure 7-7 Results of the path planning around 4 obstacles with 0.475 m radius using the Bellman-ford algorithm.

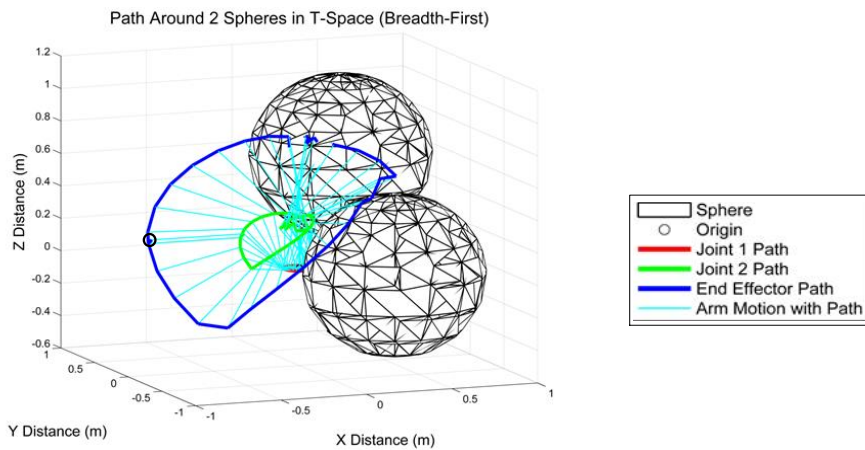
7.2.2 Breadth-first



(a) 0.47 m Object Radius

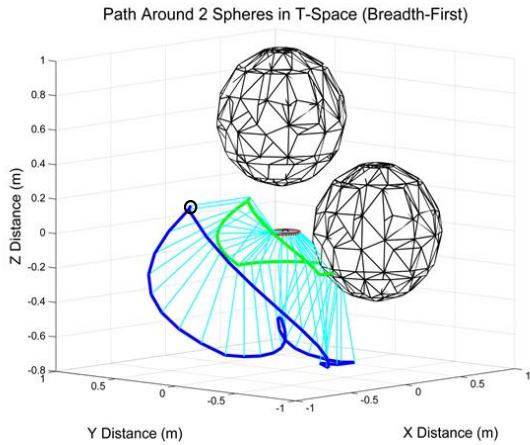


(b) 0.475 m Object Radius

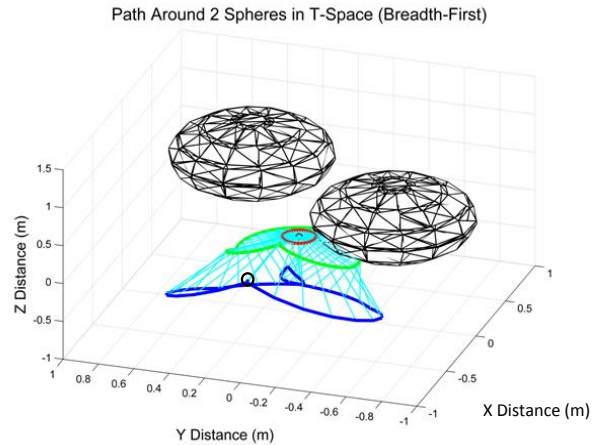


(c) 0.48 m Object Radius

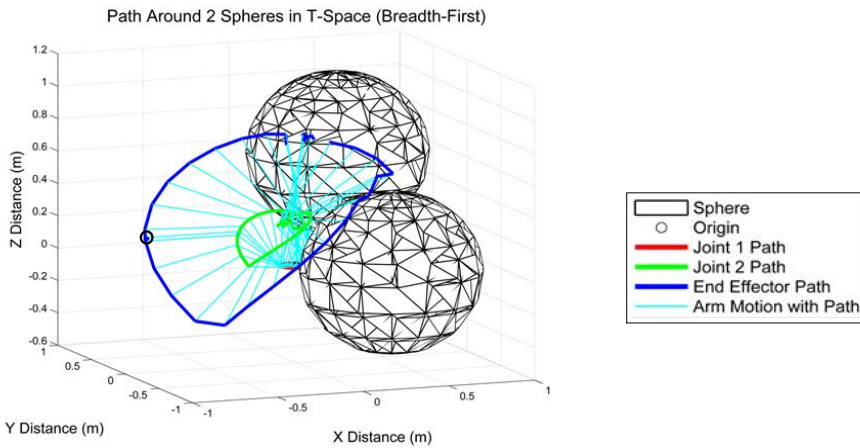
Figure 7-8 illustrates the results for the Breadth-first Method.



(c) 0.47 m Object Radius



(d) 0.475 m Object Radius

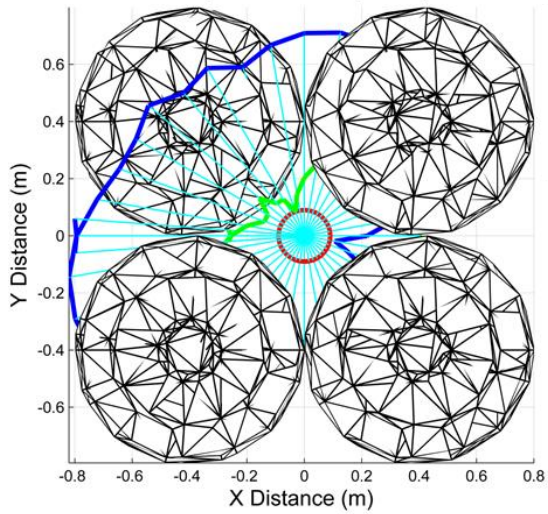


(c) 0.48 m Object Radius

Figure 7-8 Results of the path generation around two obstacles with 0.47 m, 0.48m and 0.49m radii using the Breadth-first Algorithm.

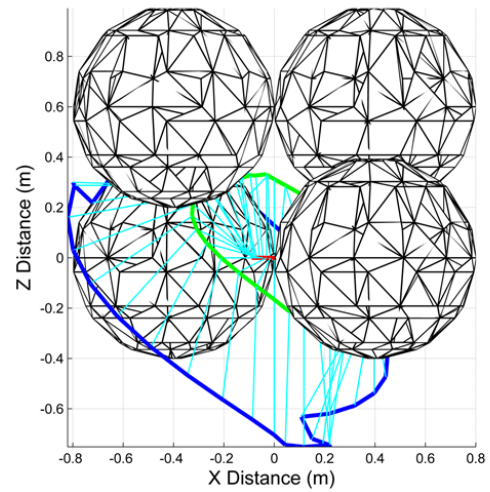
For the Breadth-first Method it can be seen from the results that the path generation around the obstacles does not produce a safe and traversable path in any of the above cases. The further path generation with 4 spheres further illustrates that this method is not able to generate a safe path.

Path Around 4 Spheres in T-Space (Breadth-First)



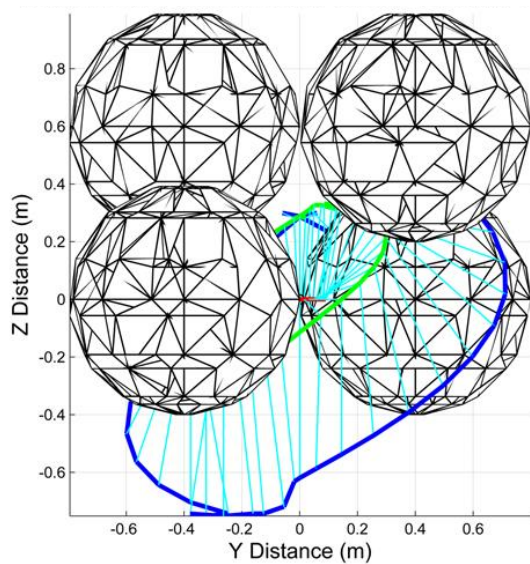
(a) X-Y View

Path Around 4 Spheres in T-Space (Breadth-First)



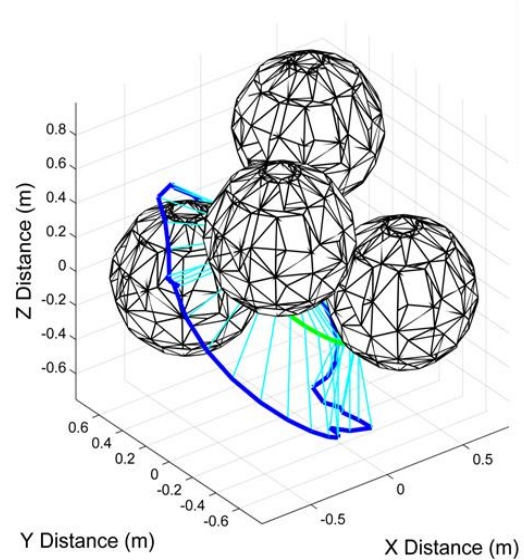
(a) X-Z View

Path Around 4 Spheres in T-Space (Breadth-First)



(c) Y-Z View

Path Around 4 Spheres in T-Space (Breadth-First)



(d) 3-D View

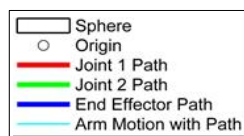
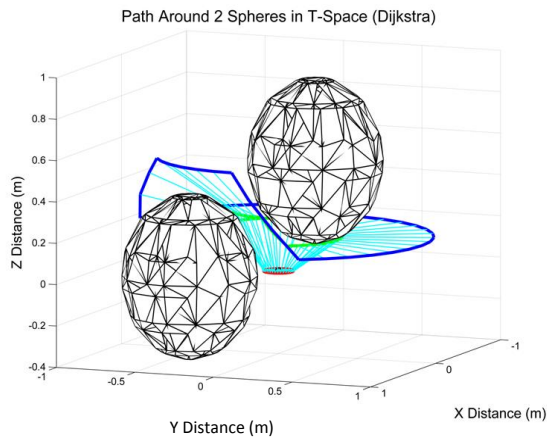


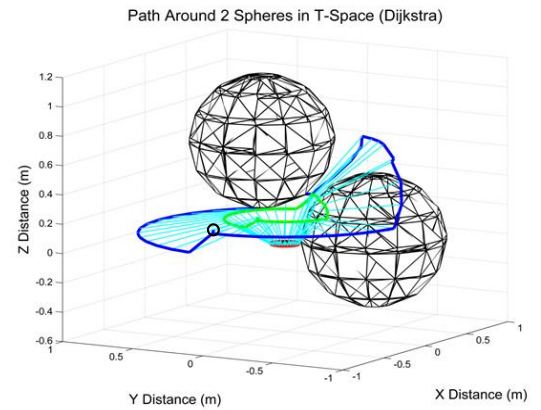
Figure 7-9 Results of the path planning around 4 obstacles with 0.475 m radius using the Breadth-first algorithm.

7.2.3 Dijkstra

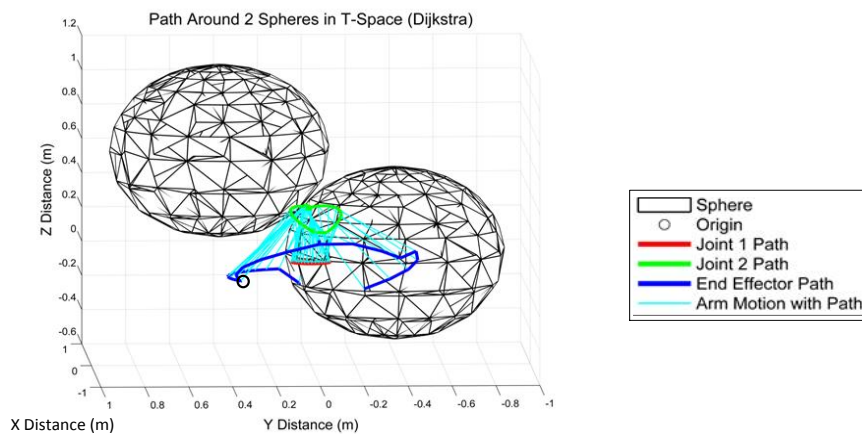
Figure 7-10 illustrates the results for the Dijkstra Path Generation Method.



(a) 0.47 m Sphere Radius



(b) 0.475 m Sphere Radius



(c) 0.48 m Sphere Radius

Figure 7-10 Results of the path generation around two obstacles with 0.47 m, 0.48m and 0.49m radii using Dijkstra's Algorithm.

In the case of Dijkstra's Algorithm a safe path has again been generated around the obstacles with 0.47 m radius and 0.475 m radius, but not the obstacles with 0.48 m radius, again confirming the prediction made upon selection of the scenarios to investigate. Further investigation by plotting a path through 4 obstacles yields the same results.

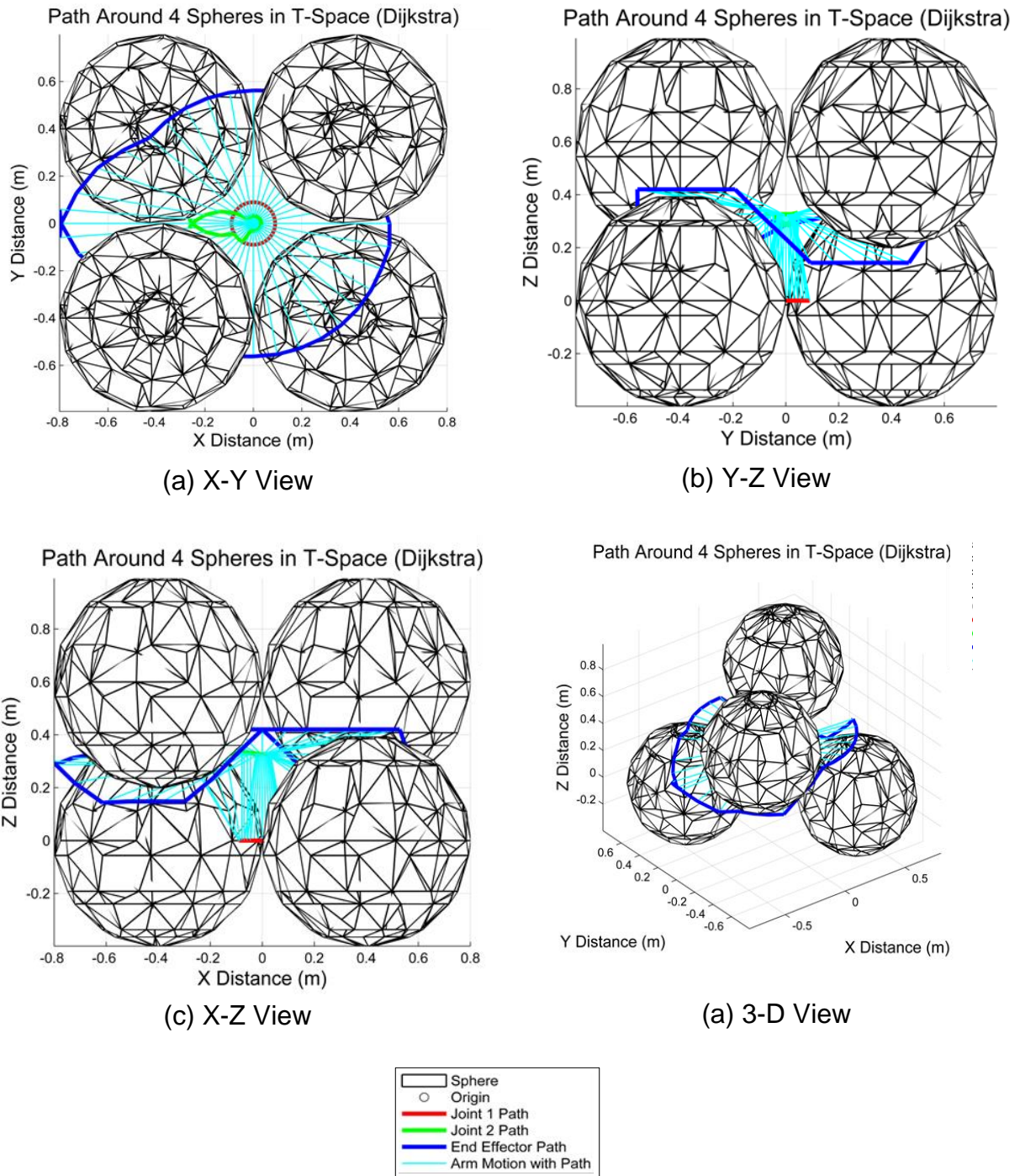


Figure 7-11 Results of the path planning around 4 obstacles with 0.475 m radius using Dijkstra's algorithm.

Given that Dijkstra's Algorithm and the Bellman-Ford Algorithm both provide safe paths around obstacles, but Dijkstra's Algorithm executes slightly faster than the

Bellman-Ford Algorithm, which is reflected in the literature, Dijkstra's Algorithm has been selected for use as the path method of choice.

7.3 Path to Trajectory Conversion

Currently the path which is generated by the guidance method is a list of demanded waypoints, but does not give any dynamic information for the arm to follow. Ideally, each joint in the arm has been designed with identical dynamics in mind by tuning the PID controller for each joint with an ideal step response matching the following system dynamic model, as shown in (7.3):

$$R_i(s) = \frac{1}{s + 1} \quad (7.3)$$

This means that each joint should reach approximately 63% of the final value in one second and have a zero steady-state error. In reality, this is not the case since the optimisation technique was not able to perfectly match this system behaviour in any scenario for any joint, but was able to maintain system performance close to this. Also, since the joints were tuned over discrete ranges, the selected gains will not be ideal for every single one of the infinite angle combinations, therefore the performance will never exactly match the ideal.

All of this means that the actual time response of each joint will not be identical and the three joints will not meet their demanded joint angle at the same time. What can be done in this case is to add dynamics to the path so that the demanded joint angle changes more slowly than all three of the joints can respond. If this is the case they should all be able to track the change in demand at the same time. If the change in demanded joint angle was faster than the joints could respond, then they would all respond at different speeds and so would reach the demanded angle at different times. This means there would be a deviation from the path as each of the joints would move in an uncoordinated way. This process is illustrated in Figure 7-12 and Figure 7-17.

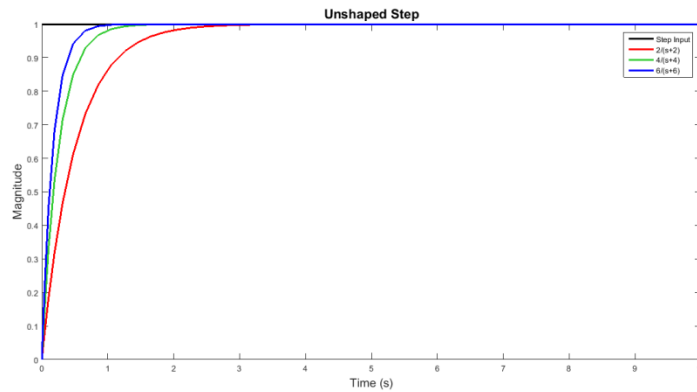


Figure 7-12 Time response of 3 different transfer functions with time constants of 0.5, 0.25 and 0.167 seconds.

In Figure 7-12, the response to a unit step of the transfer functions $G_1 = \frac{2}{s+2}$, $G_2 = \frac{4}{s+4}$ and $G_3 = \frac{8}{s+8}$ are shown. It is clear that none of the three systems reach steady-state at the same time. The following figure shows the response of the same three systems to a signal which displays the dynamics of $R = \frac{1}{s+1}$ to the unit step. In essence the three example systems are tracking the response of system R to the unit step input.

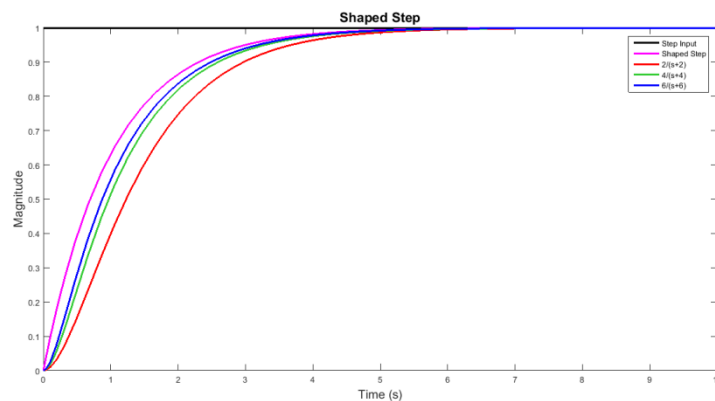


Figure 7-13 Time response of systems G_1 , G_2 and G_3 given an input which is equivalent to the step response of system R.

Figure 7-13 shows the response of the system R to the unit step in magenta, and the original three systems G_1 , G_2 , and G_3 track the magenta signal. It is clear that the speed of each system has an effect on how well it is able to track the response of R,

but as time increases all three of the systems converge on the shaped input R as it tends towards the steady-state, hence they all appear to settle at approximately the same time. Clearly there is still a difference between the responses of the system, but the dynamics of the systems chosen in this example differ by a large amount to exaggerate the difference between their responses to a unit step. In reality, all of three of the joints in the arm are tuned to display the same behaviour and as such the difference between them is much smaller. Figure 7-14 illustrates the effect if all three of the systems have very similar dynamic behaviours.

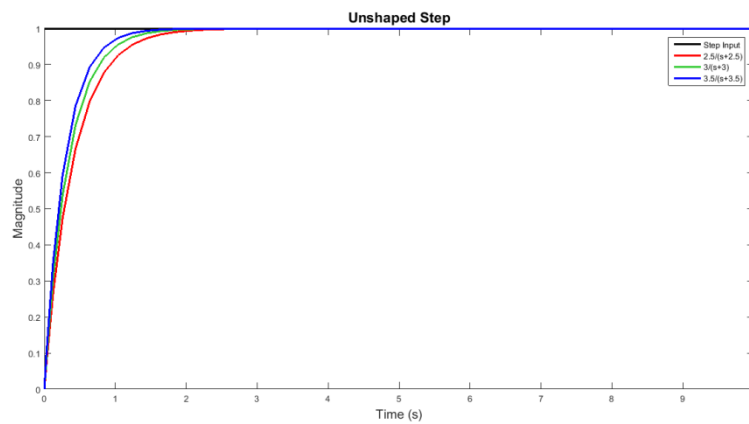


Figure 7-14 Time response of 3 different transfer functions with similar time constants of 0.4, 0.3 and 0.286 seconds.

In this case, the three systems have similar dynamics, with each system being described as follows; $G_4 = \frac{2.5}{s+2.5}$, $G_5 = \frac{3}{s+3}$ and $G_6 = \frac{3.5}{s+3.5}$. There is still a small disparity between the speed at which each responds and consequently the time at which each converges on the steady-state.

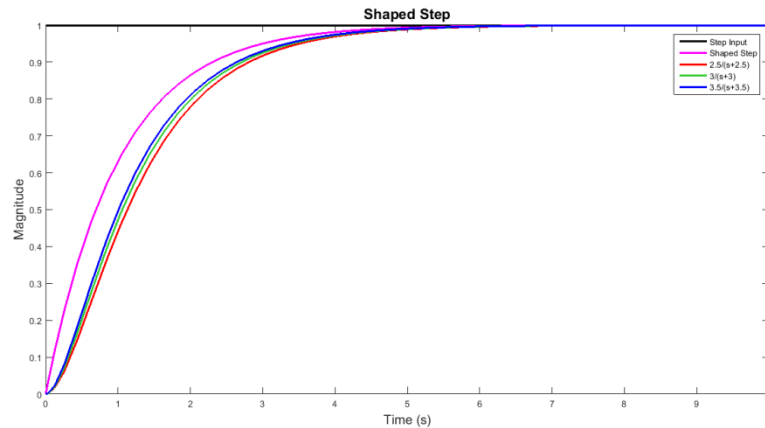


Figure 7-15 Time response of systems G_4 , G_5 and G_6 given an input which is equivalent to the step response of system R .

When inspecting the response of each of these systems to the shaped step it is clear that the disparity still exists, but the time at which each of the systems converges on the steady-state is almost identical. The distance between each of these responses, both in the transient and when converging on the steady-state, is important since any difference will correspond to a divergence from the desired path and the potential for a collision with an obstacle. If all three of the systems displayed identical dynamics, then it would not matter how fast or slow those dynamics were in terms of following the path since they would follow a linear motion along the desired path. To illustrate this a simple example of a single linear path will be simulated with three systems representing a joint each, and the divergence from the path will be shown. This investigation will be carried out with a step-based path and the same path with a shaping filter such as the low pass filter $\frac{1}{s+1}$ which has been used as the shaping function throughout this section. In this example, the X-direction will be controlled by a transfer function $G_4 = \frac{2.5}{s+2.5}$, the Y-direction controlled by transfer function $G_5 = \frac{3}{s+3}$, and the Z-direction controlled by $G_6 = \frac{3.5}{s+3.5}$, therefore the system dynamics are identical to those used in the above example. In this case the system will start at coordinates $[0,0,0]$ and be required to move to coordinates $[1,2,3]$, which will be shown in the figure below.

To illustrate the lack of difference between the path and trajectory once dynamic information is removed, Figure 7-16 shows the two plotted through the space in which they are operating.

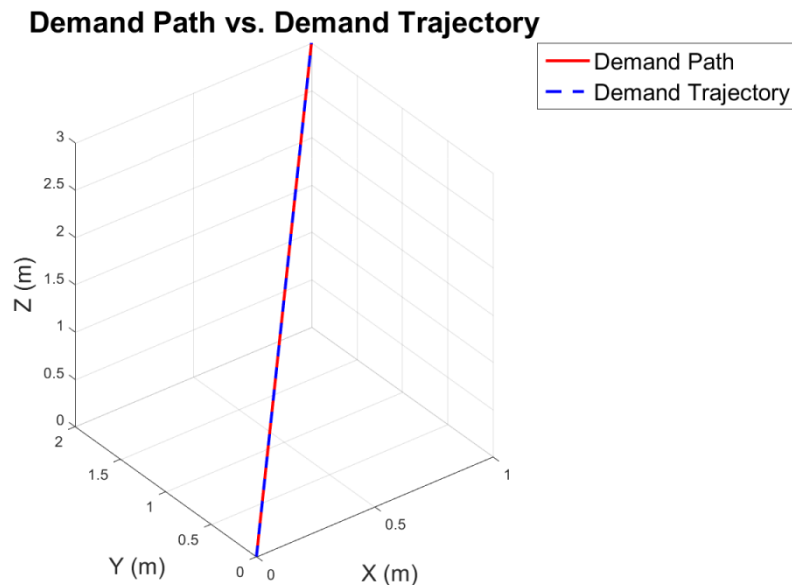


Figure 7-16 Path through space and the trajectory which represents the same path but with time information included.

As can be observed in the figure, the two lines lie directly on top of one another, therefore the spatial parameters of the path and trajectory are identical, only the time based information has changed. Figure 7-17 illustrates the difference in the system output between the system following the path and the system using the trajectory to follow the path. In Figure 7-17, the black line indicates the spatial demand of both the path and the trajectory, and this is the line that the system would ideally follow. The red line shows how the system tracks the path and the blue line shows how the system tracks the trajectory. It is clear just from this figure that there is an improvement in deviation from the desired path by using dynamic information as well as spatial.

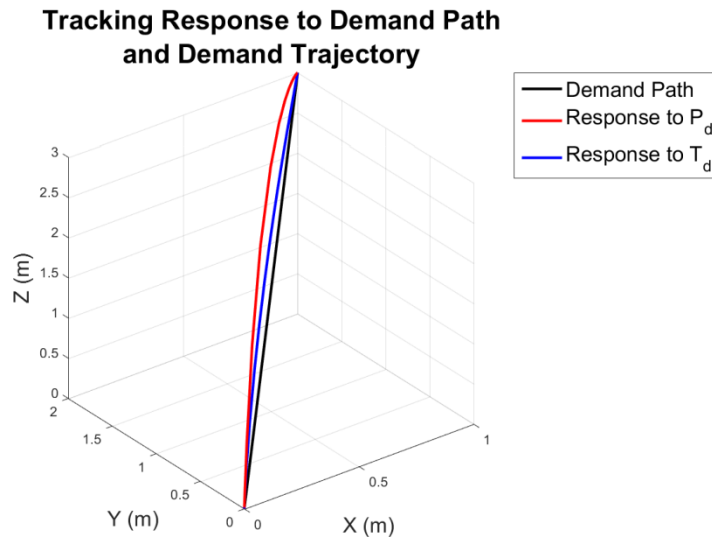


Figure 7-17 Behaviour of systems G_4 , G_5 and G_6 as dimension components of a path tracker when given the path as an input or the trajectory as an input.

To further illustrate this effect, the time response of each of the dimensions to the demand path and to the demand trajectory is plotted in Figure 7-18. There are two areas of each response which are of interest. The dashed lines between 0 and 1 seconds in the first plot and between 1 and 2 seconds in the second plot represent the time constant of each dimension, i.e. the point at which each dimension reaches 63% of the final value. In both cases these points occur very close together, so the initial response of each dimension is similar for both cases. The dotted lines which occur between 1 and 2 seconds in the first plot and at approximately 5 seconds on the second plot are the points at which the response reaches 1% of the final value, and so can be considered to have arrived at the desired location. When the system follows the demand path, the 'settling time' occurs at different times for each dimension, but when following the trajectory there is a simultaneous arrival to the demanded point.

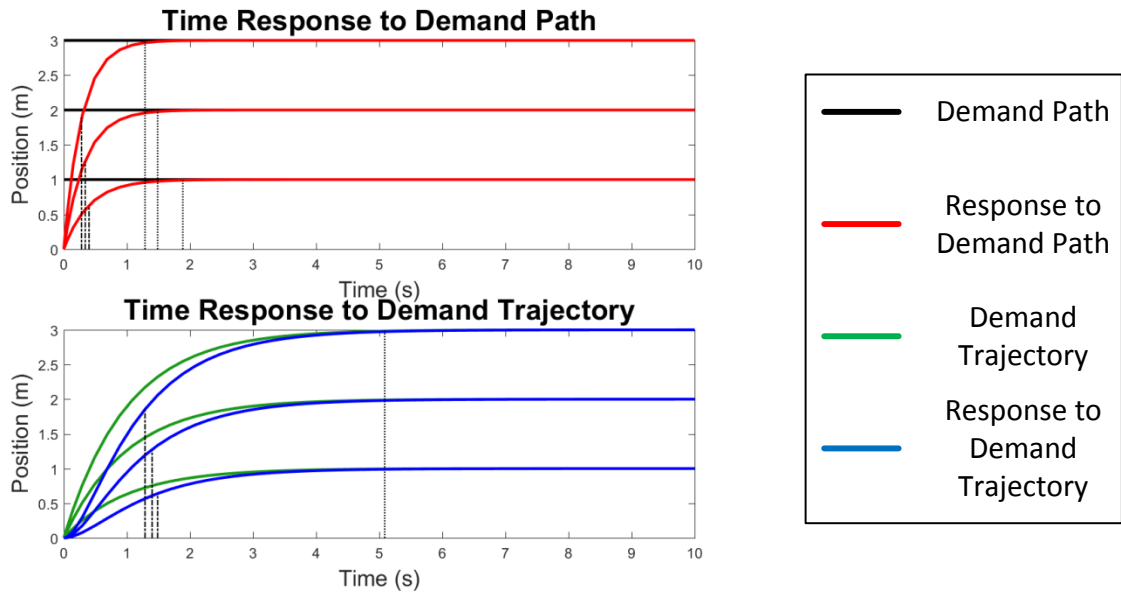


Figure 7-18 Time response of each dimension to a demanded path and demanded trajectory.

Since this is the case, the paths generated by the guidance method will be passed through a filter to give them dynamic information. The time constant of the filter will be designed such that the demanded change in position over time is slower than any of the system dynamics, helping to reduce any deviation from the desired path. In the case of the robotic manipulator dynamic model, the system is designed to behave with dynamic characteristics of $R = \frac{1}{s+1}$, and while the system almost achieves this in all of the different angle ranges, it is prudent to slow down the trajectory by this amount plus a small tolerance to account for any PID gain sets which are slower than this. By allowing the time constant of the trajectory to be 10% more than the designed time constant this should enable the system to track the trajectory through the majority of gain sets.

This means that the path will be passed through a low pass filter with the transfer function $F = \frac{0.9091}{s+0.9091}$ to generate a suitable trajectory.

7.4 Summary of Robot Arm Guidance

Having investigated multiple methods of obtaining simulated obstacle data, converting the obstacle data from T-Space to C-Space, creating a node graph which avoids the obstacles and planning a path from start to end points on this graph a selection for each can be made in order to combine them into a single algorithm which is capable of planning a safe path for an entire arm around obstacles in T-Space. The resultant algorithm carries out the above processes in the following way:

Table 7-3 List of processes carried out in the guidance algorithm.

Obstacle Data Simulation	<ol style="list-style-type: none">1. Using the selected LIDAR angular resolution calculate the spacing between points on the surface of an obstacle.2. Calculate the change in azimuth and elevation from the centre of the obstacles to each of the points on the surface.3. Using the azimuth and elevation between each point and the obstacle radius calculate the points on the surface of the spherical obstacle.
-------------------------------------	---

4. Check the Z-coordinates of each point and for those with a Z-coordinate of 0 m, indicating that the point lies on the XY-plane.
5. Using trigonometry calculate the angle between the base of the arm and each of the selected measured point on the obstacle surface in the XY-plan.
6. Calculate the XY-range between the base of the arm and the selected measured points. If the distance is smaller than or equal to the length of link 1, then there is a collision.
7. Using trigonometry calculate the elevation angle between the end of link 1 and the measured points. Calculate the angle if the α angle is π^c away from the original angle.
8. Calculate the Euclidean distance between the end of link 1 and the measured points in both the α and $\alpha \pm \pi$ cases. If the Euclidean distances to the point are less than or equal to the length of link 2, then there is a collision.
9. Using the same process as for link 2, calculate the length and angle between the end of link 1 and the measured points for both the α and $\alpha \pm \pi$ cases.
10. Using the cosine rule calculate the angles σ and η for a range of length 3 from 0 to the full length of l_3 . This gives the entire range of α , σ and η where collisions occur.
11. Add separate sets of points which are equivalent to the original but translated in each axis by the steady-state error of each of the joints found in Chapter 2.
12. Create the alpha-volume of each of the permissible boundaries.

Map End Effector Start And End Locations	Node Graph Creation
--	---------------------

13. Generate a blank node graph which is a grid of points where each dimension fills the operating range of each of the arm joints, but with spacing which corresponds to the steady-state error of the joint angles found in Chapter 2.
14. Join the nodes in the graph together with their adjacent nodes which all fall within the immediate horizontal adjacent nodes and all diagonally adjacent nodes.
15. Investigate which nodes fall inside the permissible boundaries and remove them from the adjacency matrix and list of indices.
16. Using the inverse kinematics calculate all possible angle solutions for the end effector start and desired locations.
17. Sort the angle solutions and remove those which fall outside of the operating range of the manipulator arm.
18. Find which of the remaining start and desired end effector locations are closest to each other and add them to the node graph.

19. Assign to every node an initial distance value. The distance to the starting node is set to zero and all the others to infinite
20. All the nodes are marked as unvisited, except for the starting node which is set as the current node. A set is created containing all the unvisited nodes, which initially contains all of the nodes but the starting one.
21. For the current node, consider all of its the nodes connected to it that are in the set of unvisited nodes and calculate their distance values. For example, if the current node is marked with a distance of 10, and the vertex connecting it with a neighbour has length of 5, then the distance the neighbour node through the current node will be $10 + 5 = 15$.
22. When the algorithm has considered all of the neighbour nodes of the current node, mark the current node as visited and remove it from the set of unvisited nodes. The current node will not require inspecting again.
23. If the destination node has been marked as visited or if the distance among the nodes in the set of unvisited nodes is infinity (when planning a path from one node to another this occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
24. Select the unvisited node that is marked with the smallest distance from the current node, and set it as the new "current node" then go back to step 3. If all neighbour nodes have been visited then go back down the path to the nearest node that has a neighbour which is unvisited.

The above table presents the algorithm which has been developed to plot a safe path through the environment. Alongside the Kinematics and Dynamic Model developed in Chapter 1, and the controller and PID tuning developed in Chapter 2, the path generation technique developed in this chapter can be combined to validate

the method. In the next chapter, the entire method is tested to validate the guidance method and the conclusions from the investigation are presented.

8 VALIDATION OF GUIDANCE METHOD BY SIMULATION

In Chapters 3 to 7 of this thesis a novel concept has been developed and presented which in real-time generates a safe path for a 3-DoF robotic manipulator through a close-proximity environment. In Chapter 3 a forward and inverse kinematic model of the manipulator was developed to provide knowledge of the location of the entire manipulator given a set of joint angles. Chapter 4 presented the development of a dynamic model of a 3-DoF manipulator to provide a test bed for use in the development of a guidance technique. Chapter 5 was responsible for the selection and tuning of a control schema for the robotic manipulator dynamic model to provide adequate system performance and also provide some of the design specifications for the guidance method, especially in terms of safe boundaries around obstacles. Chapters 6 and 7 detailed the design decisions and implementation of a technique which could satisfy the requirements of the manipulator arm dynamic model. In this chapter, several simulated scenarios of increasing obstacle complexity are presented and the performance of the combined controlled dynamic model and guidance method discussed.

8.1 Simulation Design

To assess the capability of the tuned arm and the guidance method together a series of simulated environments with obstacles contained within them must be decided upon. Each environment will get gradually more complex to test the limits of the arm-guidance combination.

The first simulation to be carried out is to plan a path around a single obstacle since this will show whether the arm can follow a simple path. Following this the number of obstacles should be increased to add complexity to the path. Ideally the obstacles should be placed such that the generated path is required to pass between them to assess the manipulator arm's ability to stay within the set confines of the accessible space. If the arm is able to maintain the performance displayed in Chapter 2, then

the design parameters of the guidance method should prevent the arm from colliding with any obstacle as it passes between them. In this case the arm displayed a performance such that it was able to maintain a steady-state error of 2° or less and no overshoot so the guidance method allows a safe region of 2° around each obstacle in C-Space to provide the extra leeway that the arm requires when navigating the environment. Therefore the obstacle locations will be designed such that the guidance method can only just pass between them. This will mean that when the dynamic model of the arm is driven along the demand path, any deviation from the path would cause a collision with an obstacle. To test that the guidance method and arm work together in the environment, any deviation larger than $2^\circ \pm 0.225^\circ$ may cause a collision with an obstacle.

The first four environments will contain 1, 2, 3 and 4 obstacles respectively, and the final two will increase the complexity further. The 5th scenario will contain 4 spheres which are close enough together to overlap, leaving a small hole through which the arm has to pass. The 6th scenario will contain a string of spheres close enough together to overlap. This scenario is designed to force the arm to have to navigate a narrow corridor where any deviation in any direction could cause a collision.

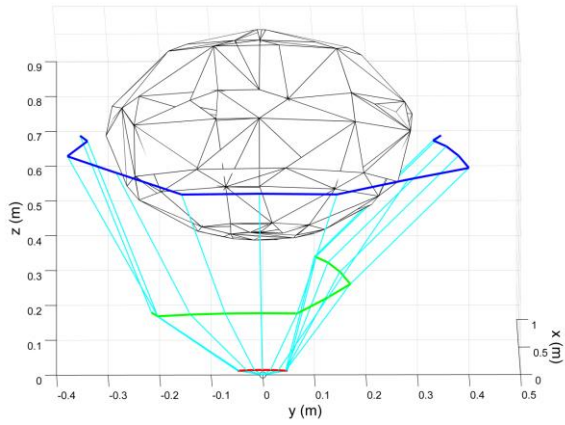
8.2 Results

The following section of this chapter presents the results from the simulated scenarios.

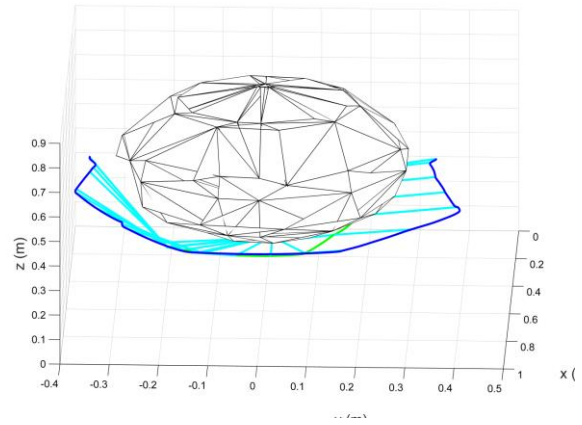
8.2.1 Single Obstacle

This scenario contains a single sphere as an obstacle. The centre of the sphere is located at [0.566,0,0] and the sphere radius is 0.3 m. The arm is required to move the end effector from one side of the obstacle to the other without a collision. Figure 8-1 shows four different results. Subfigure (a) shows the planned path of the manipulator around the spherical obstacle in T-space while subfigure (b) shows the actual path that the manipulator model takes when tracking the planned path.

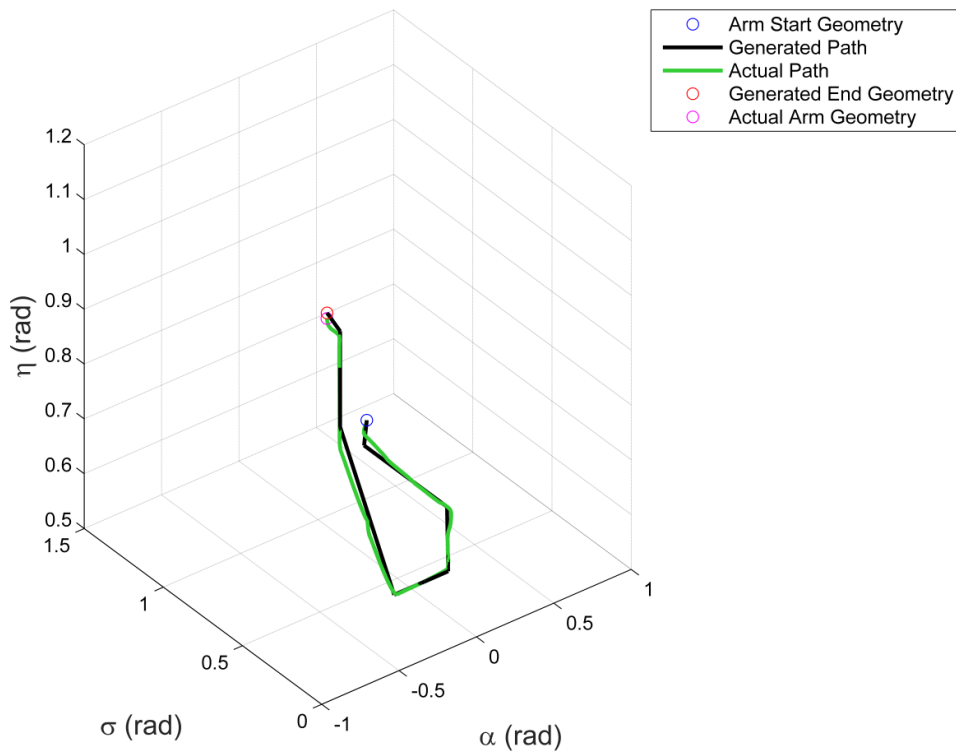
Subfigure (c) shows these two paths in contrast to one another in C-space which results in two tracks overlapping one another.



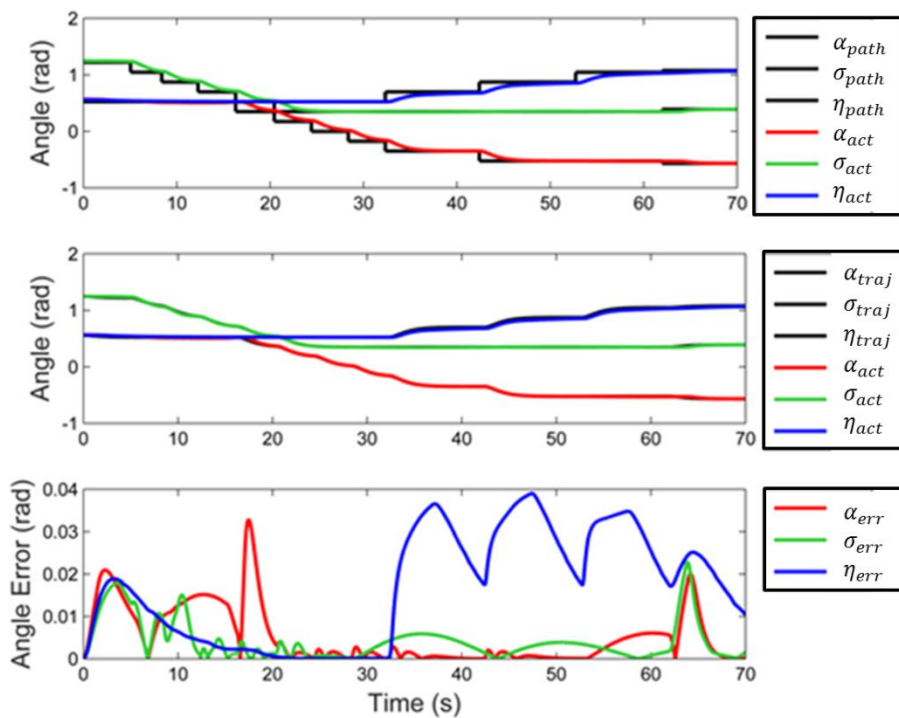
(a) Planned Path in T-space



(b) Actual Path in T-space



(c) Demand and Actual Paths in C-space



(d) Path and Trajectory Demands, Actual Path and Tracking Error.

Figure 8-1 Results of simulation 1 showing the planned (a) and actual (b) path of the manipulator through T-space, the planned and actual path through C-space (b) and the locations of each joint in time (d).

It can be observed from the plot that the actual path (green) diverges slightly from the planned path (black). Subfigure (d) has three sets of information. The first of the subplots shows the planned path as a series of step inputs against time, with the path also converted to a trajectory plotted on the same axes. The second of the subplots shows the planned trajectory against time with the actual joint angles of the arm plotted on the same axes and the third subplot shows the angular tracking error of each joint over time.

In this case the guidance algorithm is able to generate a path which safely navigates around the obstacle. When the arm is driven through the generated trajectory it is able to follow the path with a small amount of deviation. By inspection of the motion of the arm through T-Space, it would appear that the arm does not collide at any point since none of the joint tracks disappear into the sphere and none of the cyan

arm geometries collide with the sphere. By inspecting the angular error of each joint it is clear that none of the joints exceed an error of 0.04^c . In fact the largest error is experienced by joint η , which is just short of 0.039^c , or 2.235° . This value does exceed the allowed error by 0.01° . A further simulation with a smaller threshold of 0.05° to switch between waypoints was carried out following this simulation, and Figure 8-2 and Figure 8-3 present the results.

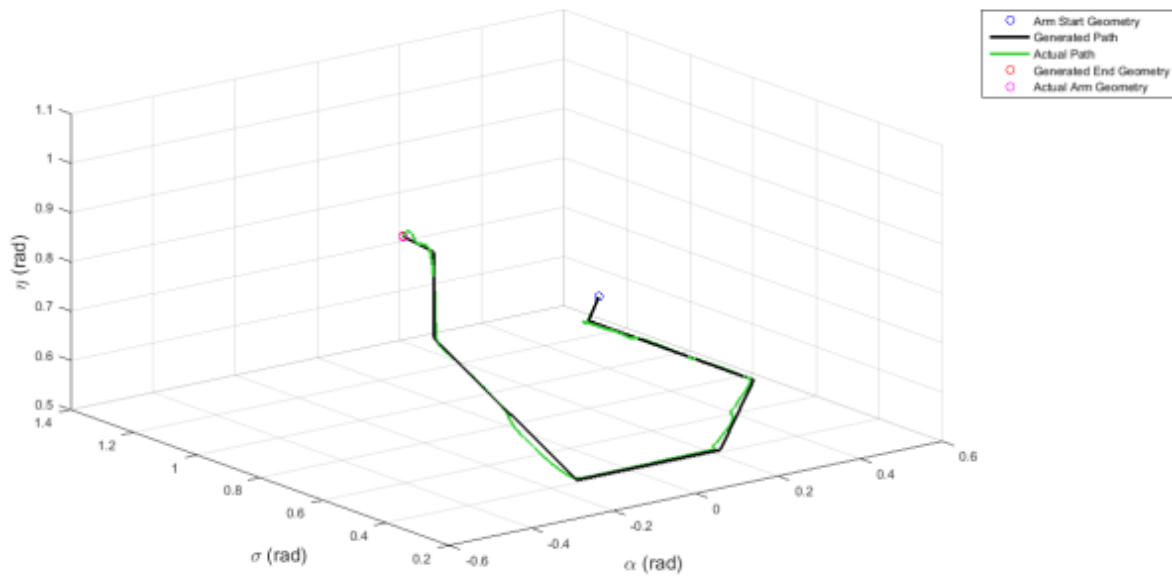


Figure 8-2 Demand and actual C-space paths with a waypoint tolerance of 0.5° in scenario 1.

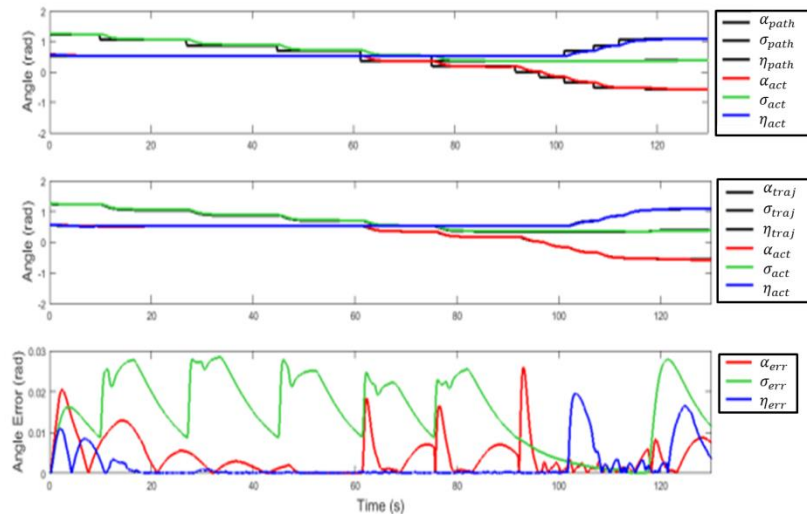
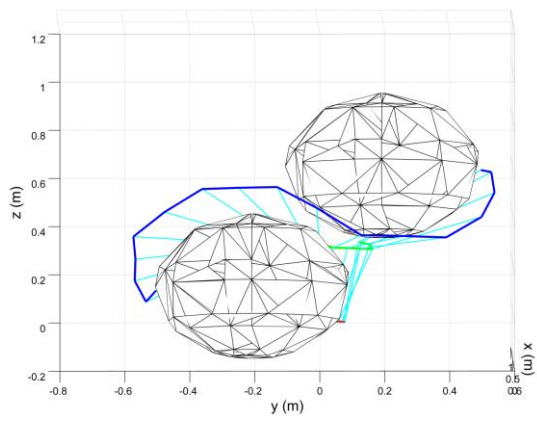


Figure 8-3 Demand path and trajectory, actual path and tracking error over time for a waypoint tolerance of 0.5° in scenario 1.

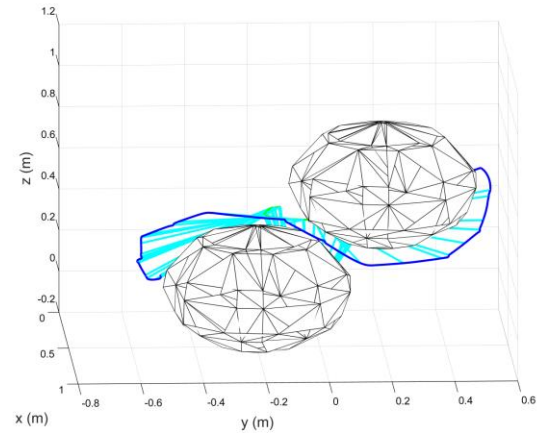
In the case of the smaller tolerance before the path following algorithm allows the system to move to the next waypoint, the arm was able to follow the path more smoothly, with a maximum error of close to 0.029^c , which is approximately 1.7° . This smaller error is well within the tolerances allowed by the guidance algorithm therefore the arm does not collide with the obstacle. The reason for the previous error value is that the threshold of 2° was large enough that the joints were still converging on their steady-state value and so the angular velocity was sufficiently high that the joints overshoot the next waypoint. This is especially true when multiple waypoints exist in the same direction since the arm does not have to slow down when the next waypoint is selected and so continues with a larger angular velocity.

8.2.2 Two Obstacles

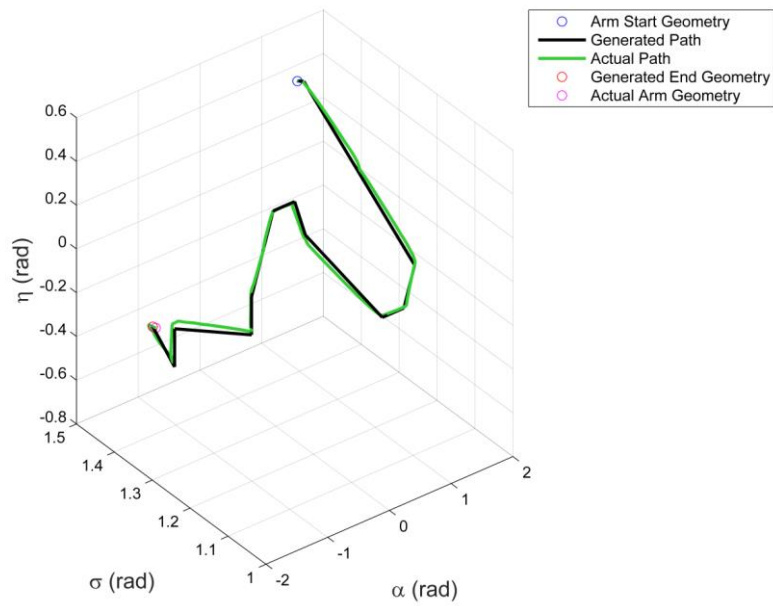
This scenario contains two spheres close together as obstacles in the environment. The centre of the spheres are located at $[0.53, 0.2, 0.7]$ and $[0.53, -0.2, 0.2]$, again with sphere radii of 0.3 m. The arm is required to move the end effector from one side of one of the obstacles to the opposite side of the other without a collision. Figure 8-4 contains the same series of results as Figure 8-1, but for scenario 2.



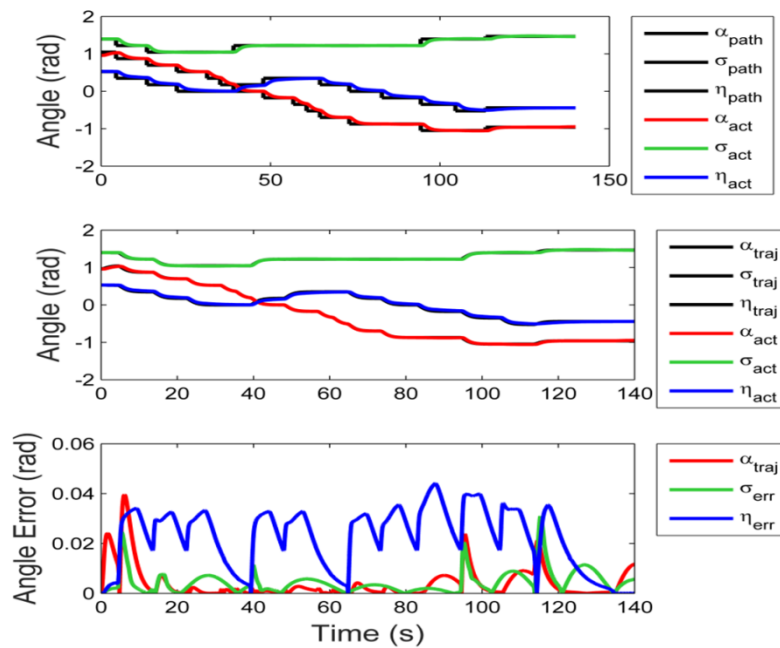
(a) Planned Path in T-space



(b) Actual Path in T-space



(c) Demand and Actual Paths in C-space



(d) Path and Trajectory Demands, Actual Path and Tracking Error.

Figure 8-4 Results of simulation 2 showing the planned (a) and actual (b) path of the manipulator through T-space, the planned and actual path through C-space (b) and the locations of each joint in time (d).

In the initial simulation for this scenario, the generated path is again capable of avoiding all obstacles. The manipulator's ability to follow the trajectory with a waypoint tolerance of 2° again falls slightly short of the specification, with maximum angular error on both α and η of more than 0.04^c , which is 2.3° . This angular error exceeds the limit of 2.225° , which means that the arm is not capable of following the path without a possible collision when the tolerance between waypoints is set as 2° .

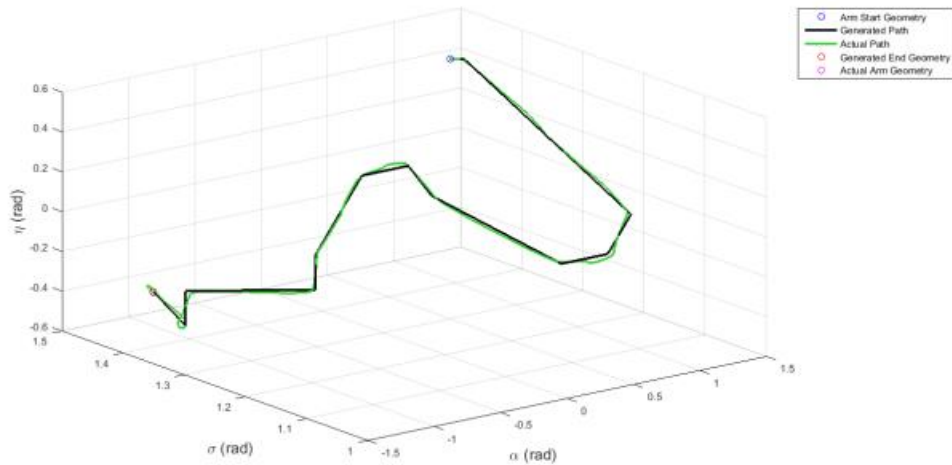


Figure 8-5 Demand and actual C-space paths with a waypoint tolerance of 0.5° in scenario 2.

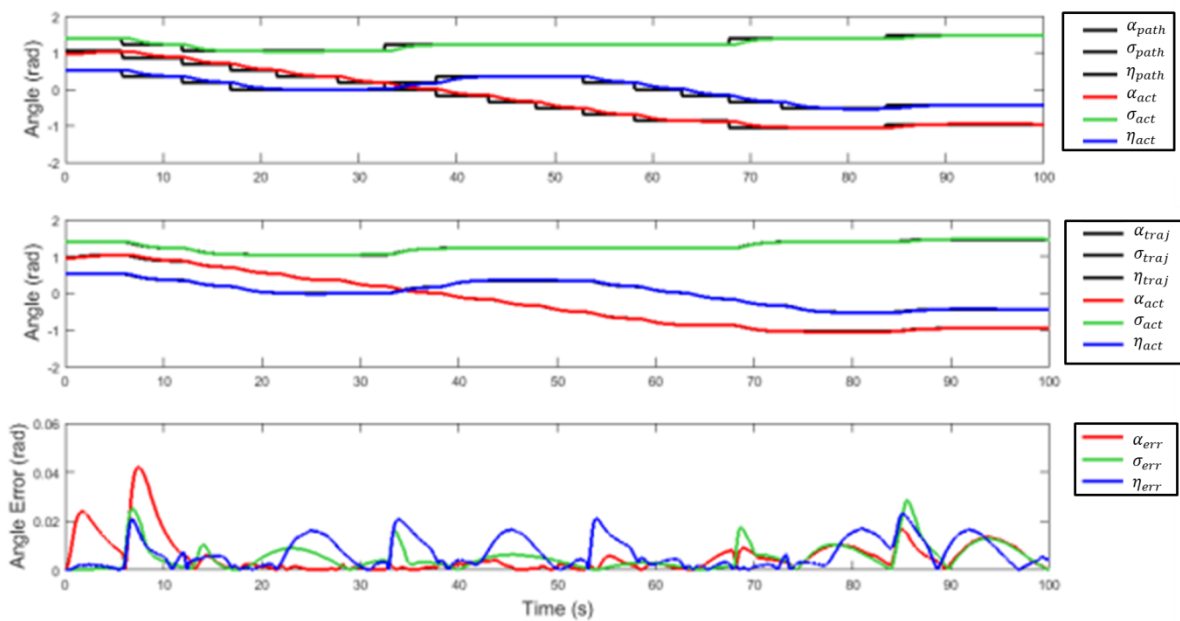


Figure 8-6 Demand path and trajectory, actual path and tracking error over time for a waypoint tolerance of 0.5° in scenario 2.

In this scenario, once the tolerance for waypoint selection is reduced to 0.5° , the arm displays a much greater performance when following the trajectory. The maximum error on each joint is reduced to approximately 0.02° , which is 1° . This would mean

that the arm would safely be able to follow the generated path around the obstacles, however there is one point at which the joint angle error for α still rises above 0.04^c . This means that there is the potential at that point in time that the arm would collide with an obstacle, but in this case, there is no collision. The reason behind the avoidance of a collision is as follows. This error occurs in a transient, where both the path and the trajectory are ahead of the actual position of the joints, so the error does not give a clear indication of the deviation of the arm from the path. To illustrate this statement Figure 8-7 and Figure 8-8 show the time response of a system with similar estimated dynamics to the joints, $G(s) = \frac{1}{s+1}$ to the shaped path/trajectory of a unit step using the filter transfer function $F(s) = \frac{0.9091}{s+0.9091}$, along with the resultant error between the two. Given that the guidance method has produced waypoints which are 10° or 0.175^c apart, the input step will be the same value.

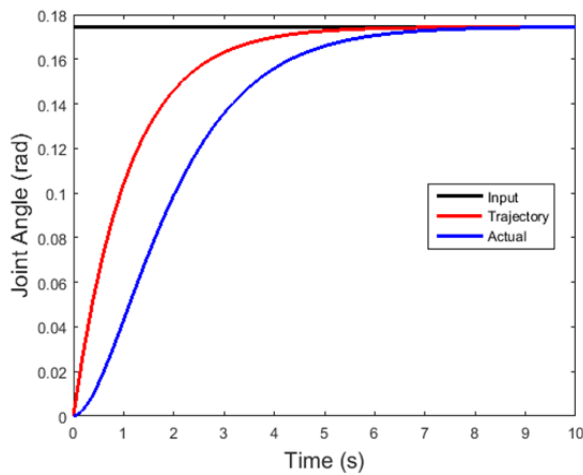


Figure 8-7 Time response of demand path and trajectory and actual manipulator path.

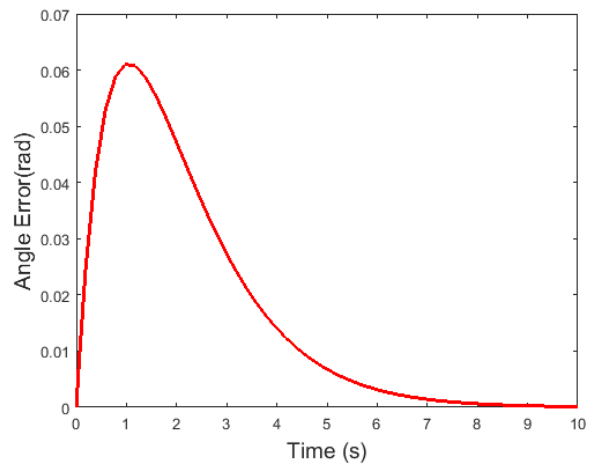


Figure 8-8 Tracking error of manipulator against trajectory.

In the case of a joint with the dynamics $G(s) = \frac{1}{s+1}$, the angular error experienced to with a trajectory which changes the angle by 0.175^c is 0.061^c . Provided that the joints pass through the points in a path together, the arm will not deviate from the path. A method of investigating this is to inspect the error graph for areas where the

three shape of the error for each joint does not line up in the time domain. For example, in the error chart in Figure 8-9 the highlighted locations have a mismatch in the shape of each joint error in time.

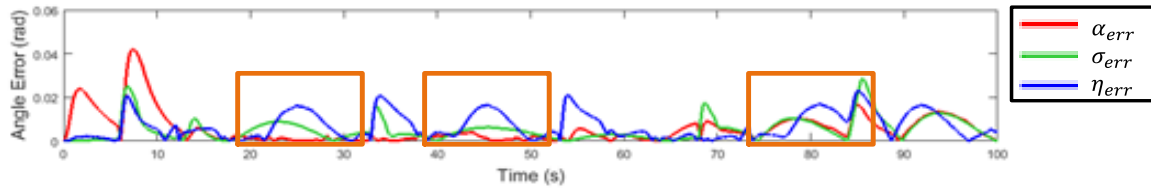


Figure 8-9Tracking error of manipulator joints in scenario 2. The highlighted areas show where the three joint errors do not match up in time, predicting divergence from the demand path.

In the time ranges of 20-30 seconds, 40-50 seconds and 75-85 seconds there is a mismatch in the error over time. This means that the joints are offset in from each other on the trajectory and there will be divergence from the demanded path. This can be seen by inspecting the plot of the path/trajectory and the actual path that the arm takes, but only for the time regions highlighted Figure 8-9.

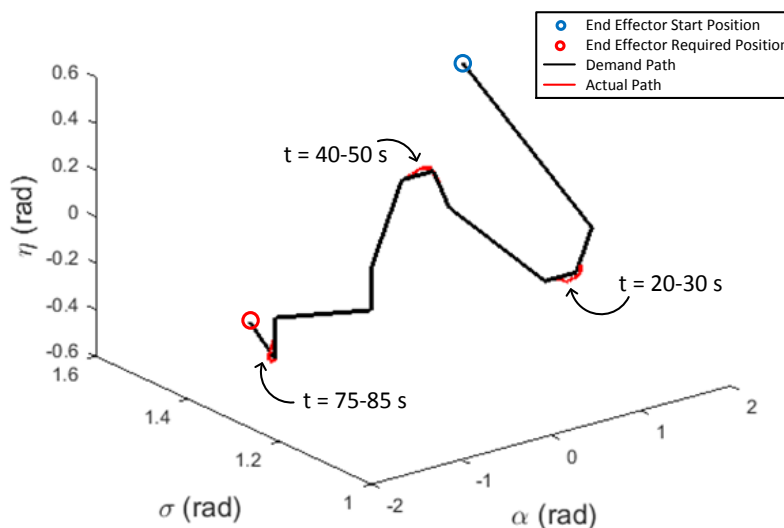


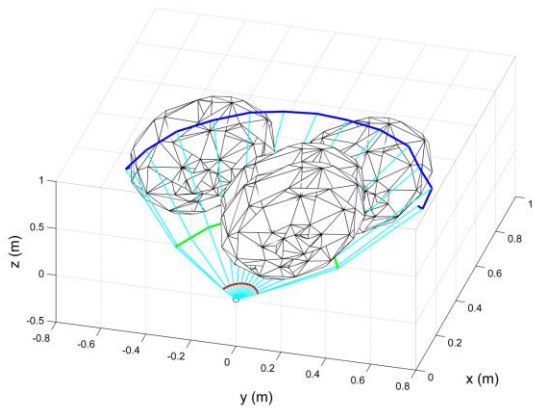
Figure 8-10 C-space demand path for scenario 2 with the actual path plotted only for the time ranges highlighted in Figure 8-9.

Inspection of Figure 8-10 shows that the areas of the actual path which have been plotted are in fact the sections of the path where the arm has deviated in angle. Fortunately, in this case, since the tolerance between demand and actual joint angle has been reduced to 0.5° before the arm is allowed to move onto the next waypoint the angular error for each joint is within the 2.225° limit, therefore the arm does not collide with any obstacle.

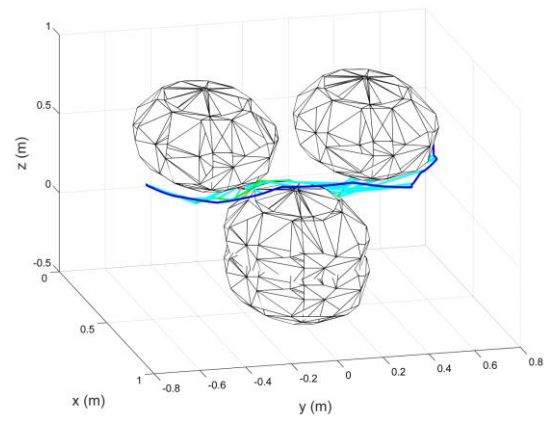
8.2.3 Three Obstacles

This scenario contains three spheres close together as obstacles in the environment. The centre of the spheres are located at $[0.45, 0.36, 0.6]$, $[0.45, -0.36, 0.6]$ and $[0.566, 0, 0]$, again with sphere radii of 0.3 m. In this case the generated path avoids passing in between the obstacles so a further obstacle was implemented below the third to force the path in between the obstacles. The additional sphere was implemented at $[0.566, 0, -0.2]$. The arm is required to move the end effector from one side of one of obstacle 1 to the opposite side of obstacle 2 without a collision. Figure 8-11 contains the same series of results as Figure 8-1, but for scenario 3.

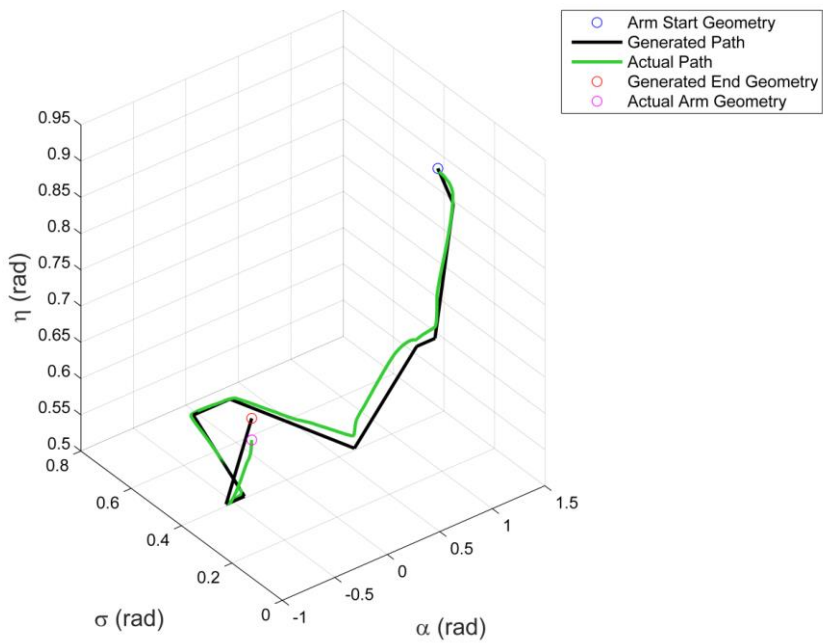
As with the previous two scenarios, the generated path is able to avoid the obstacles, and the arm is able to follow the path, but does show some deviation. Joint angle η shows a maximum angular error of 0.055° or 3.15° . This is larger than the maximum limit by 40%, however where this order of magnitude of error occurs in time, the shape of each of the error plots lines up, which is why the arm does not collide with an obstacle. The arm has deviated from the trajectory mostly in time, and only a small amount in space.



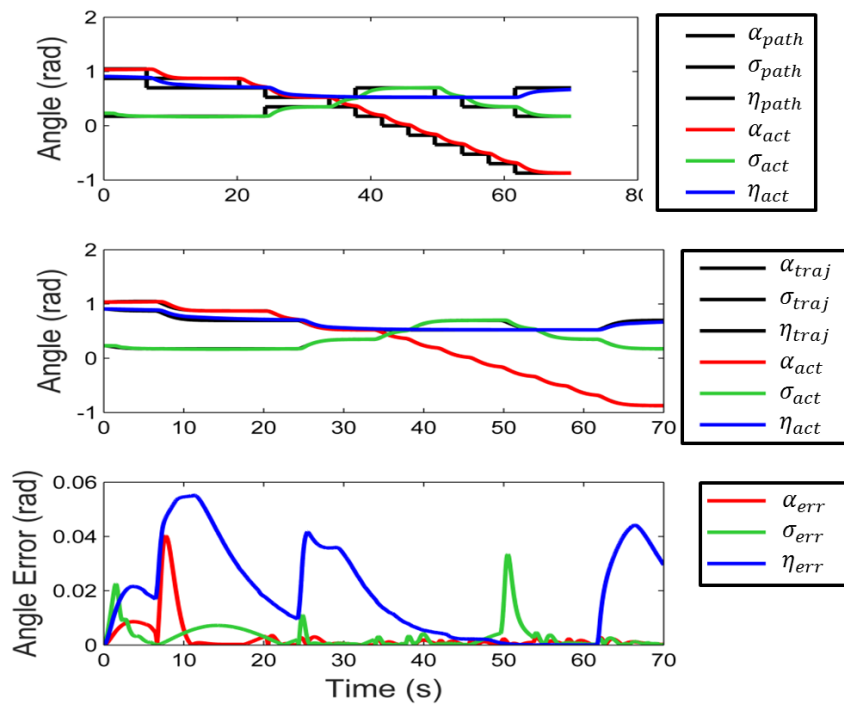
(a) Planned Path in T-space



(b) Actual Path in T-space



(c) Demand and Actual Paths in C-space



(d) Path and Trajectory Demands, Actual Path and Tracking Error.

Figure 8-11 Results of simulation 3 showing the planned (a) and actual (b) path of the manipulator through T-space, the planned and actual path through C-space (b) and the locations of each joint in time (d).

The region of most interest in this figure is that which is highlighted in red. In this area the joint angle η displays a very large error over a sustained period of time, which may correspond to the deviation between the demanded path and the path that the arm actually takes through space. This section of the path is displayed in the following figure.

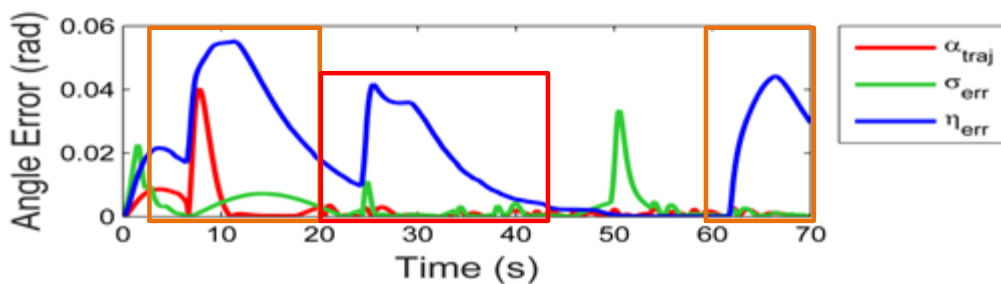


Figure 8-12 Tracking error of manipulator joints in scenario 3. The highlighted areas show where the three joint errors do not match up in time, predicting divergence from the demand path.

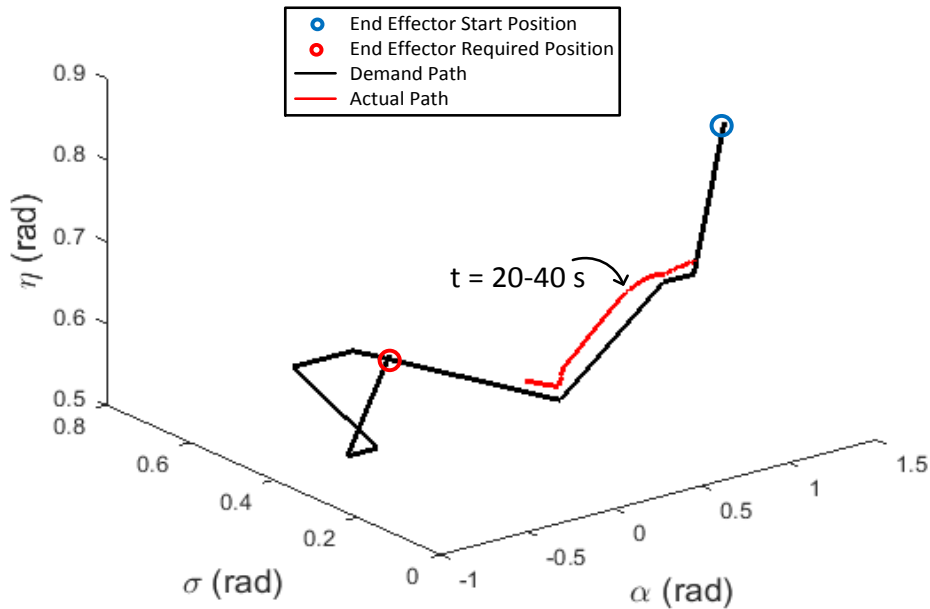


Figure 8-13 C-space demand path for scenario 2 with the actual path plotted only for the time ranges highlighted in Figure 8-12.

Again it is clear from the figure that the regions on the error plot where the shape of each joint error does not line up in time is corresponds to the section of the path where the arm deviated from the demanded trajectory. By removing the time element from the path and lining up the corresponding section of the demanded path the spatial error can be investigated rather than the error over time. Figure 8-14 and Figure 8-15 show the progress of the demanded and actual trajectories the angular error between them in space rather than time. The first figure shows each of the two trajectories with a coloured map as time progresses. The shortest time is blue and the longest time is red.

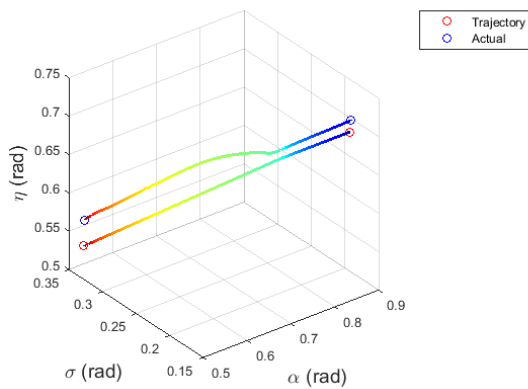


Figure 8-14 Plot of planned C-space trajectory and actual path over time. Blue at the start and red at the end.

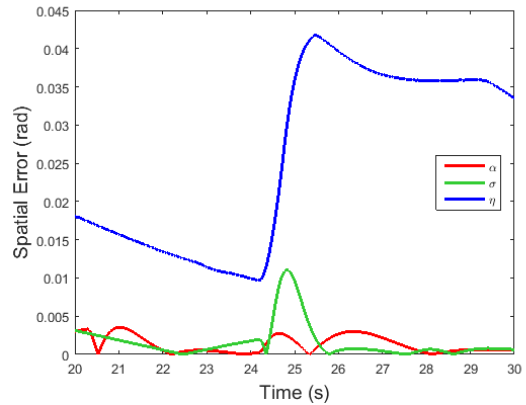


Figure 8-15 Tracking error over the time range shown in Figure 8-14

As can be observed from inspection of the demanded and actual trajectories, the two match up in time. Given this is the case, the maximum deviation of joint angle η from the path is 0.0425^c or 2.44° . This is still outside the error limit for the arm. In this case the error between the arm and the demanded trajectory when each is at the same point along the path. To reduce this error further, the waypoint angle tolerance is reduced from 2.225° to 0.5° . Figure 8-16 and Figure 8-17 display the results.

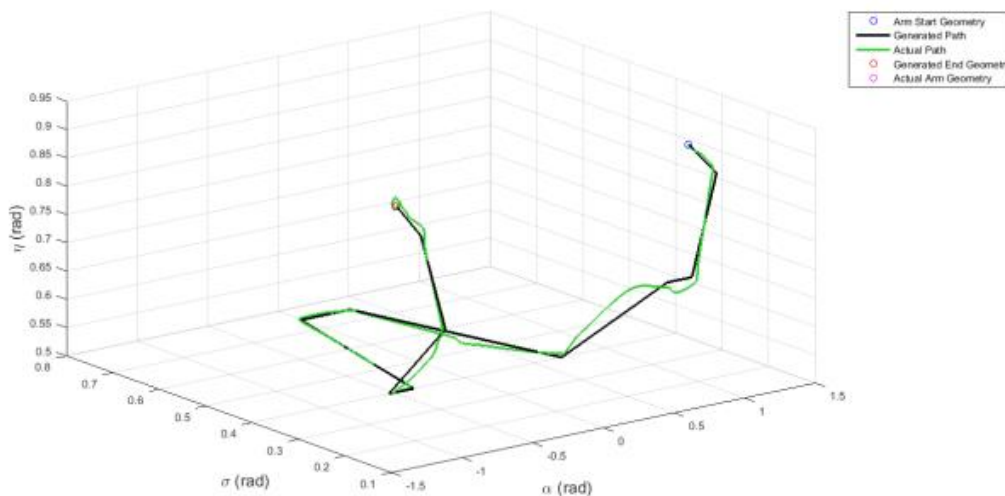


Figure 8-16 Demand and actual C-space paths with a waypoint tolerance of 0.5° in scenario 3.

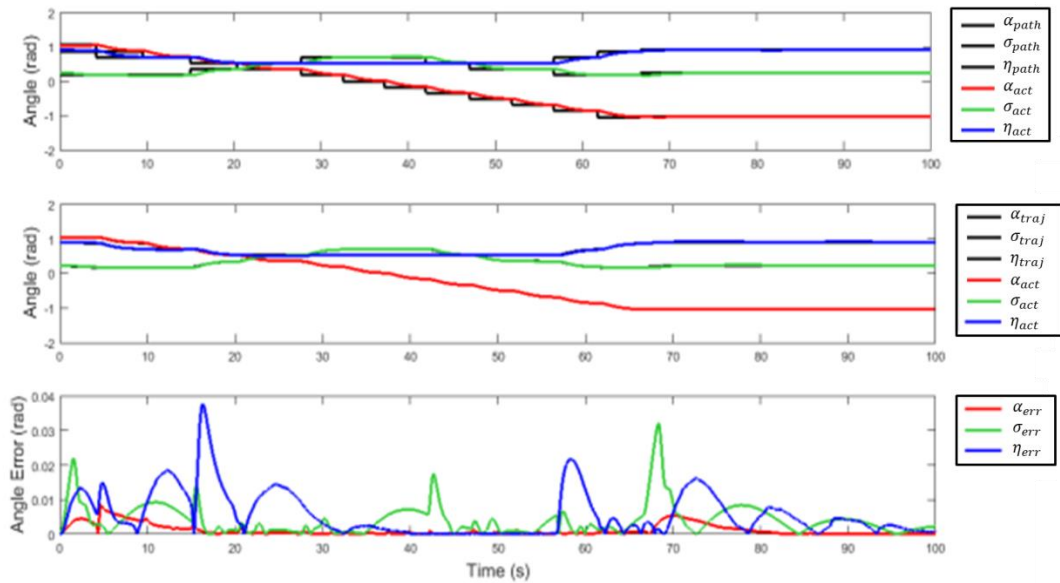


Figure 8-17 Demand path and trajectory, actual path and tracking error over time for a waypoint tolerance of 0.5° in scenario 3.

In the case of the simulation results presented in Figure 8-16 and Figure 8-17 above two figures, the maximum angular error occurs in η , and is 0.0377° , or 2.16° , which is inside the tolerance, therefore the manipulator will not collide with an obstacle when following the path. The areas where the manipulator has diverged from the path can be predicted by inspecting the areas of the largest error where the error on each joint does not line up in time. There are four time ranges where this occurs for this scenario, 7 seconds to 21 seconds, 23 seconds to 28 seconds, 56 seconds to 61 seconds and 66 seconds to 80 seconds.

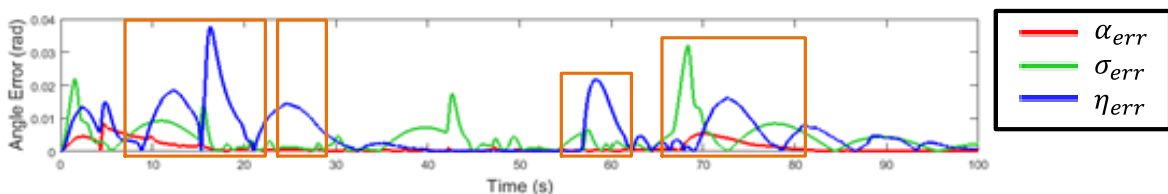


Figure 8-18 Tracking error of manipulator joints in scenario 3. The highlighted areas show where the three joint errors do not match up in time, predicting divergence from the demand path.

Plotting the actual path of the arm for these time ranges yields the result shown in Figure 8-19.

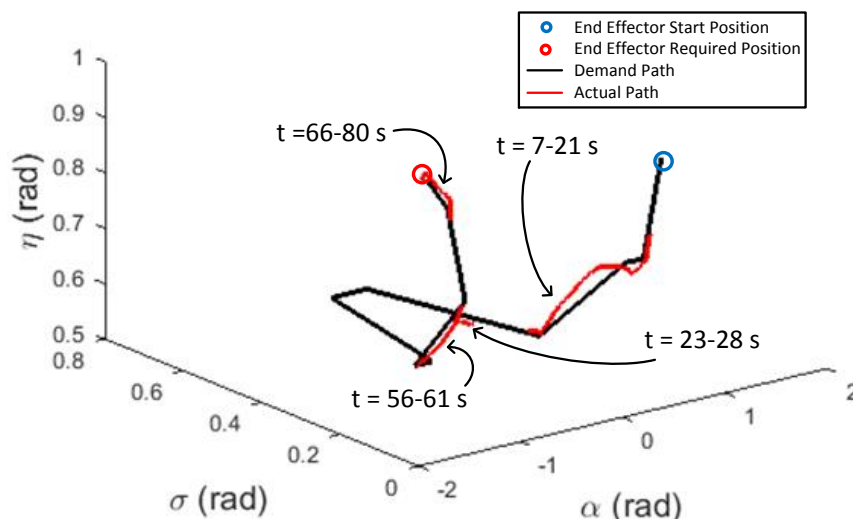


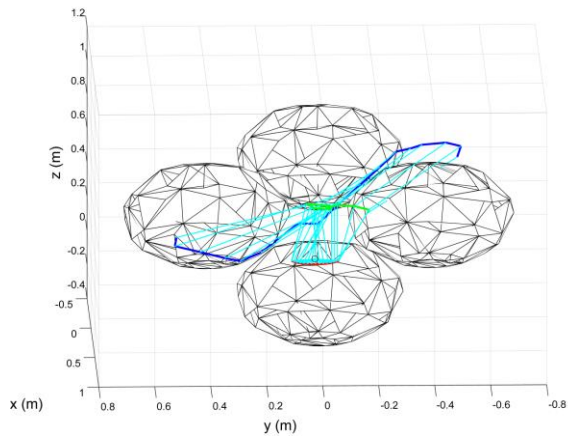
Figure 8-19 C-space demand path for scenario 2 with the actual path plotted only for the time ranges highlighted in Figure 8-18.

In this case the arm has diverged by the error amount shown, but this is less than the 2.225° limit of the guidance method therefore the arm will not collide with its environment.

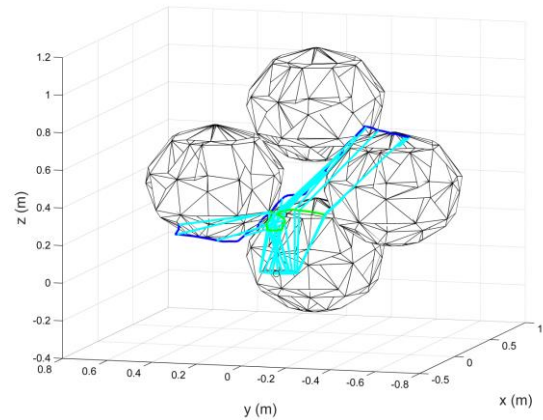
8.2.4 Four Obstacles

This scenario contains three spheres close together as obstacles in the environment. The centre of the spheres are located at $[0.4, 0.4, 0.6]$, $[0.4, -0.4, 0.6]$, $[0.566, 0, 0.8]$ and $[0.566, 0, 0]$, again with sphere radii of 0.3 m. The arm is required to move the end effector from one side of one of obstacle 1 to the opposite side of obstacle 2 without a collision. As with the other scenarios, the guidance method is capable of planning a path which avoids all obstacles through the environment. In this case, there is a large number of situations where the joint angle error on all three joints exceeds the 0.0388^2 or 2.225° limit. For the same reason as explained in scenario 2, this does not appear to have an impact on the ability of the arm to follow the path,

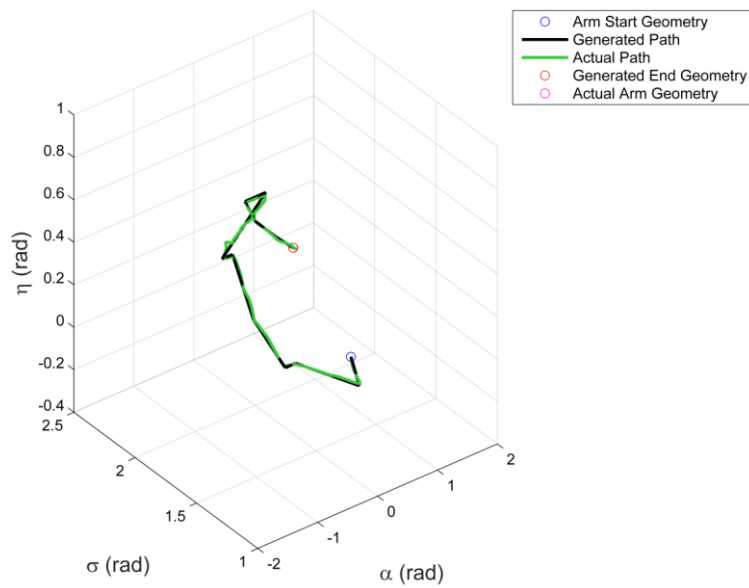
but the maximum angular error experienced by the arm can be reduced using the same method as before. Figure 8-20 contains the same series of results as Figure 8-1, but for scenario 4.



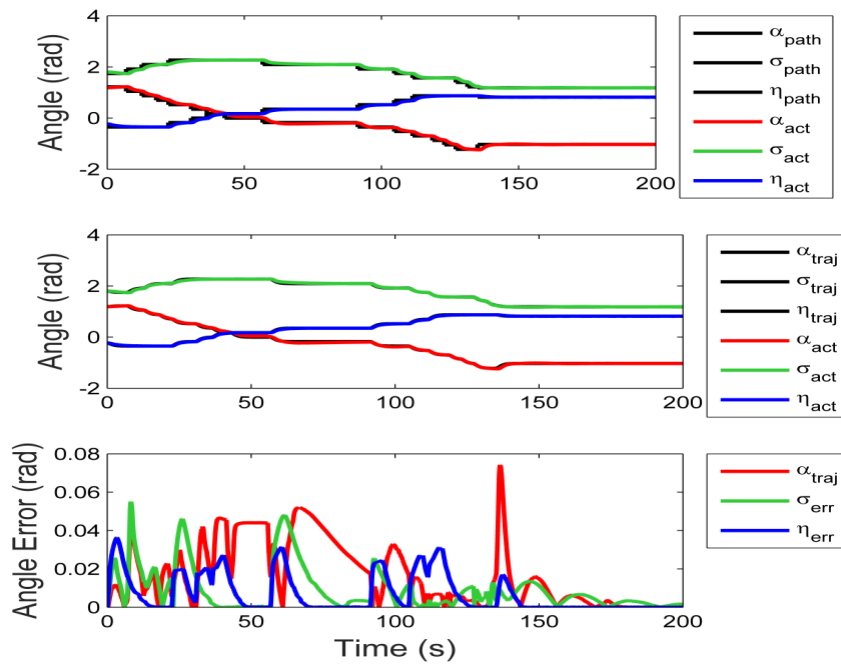
(a) Planned Path in T-space



(b) Actual Path in T-space



(c) Demand and Actual Paths in C-space



(d) Path and Trajectory Demands, Actual Path and Tracking Error.

Figure 8-20 Results of simulation 4 showing the planned (a) and actual (b) path of the manipulator through T-space, the planned and actual path through C-space (b) and the locations of each joint in time (d).

The maximum error in α coincides with peaks in σ and η as well, indicating that it is a transient and all three joints are following the path. An area of more interest is the error in α between 50 seconds and 85 seconds. This error is offset from the corresponding σ and η errors and has a magnitude of 0.05^c , or 2.9° . This suggests a divergence from the demanded path.

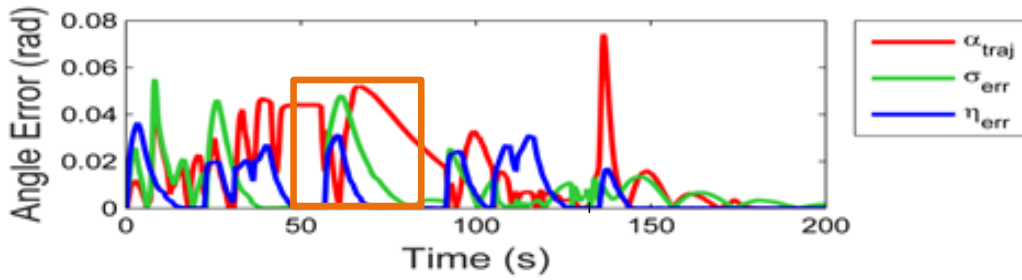


Figure 8-21 Tracking error of manipulator joints in scenario 4. The highlighted areas show where the three joint errors do not match up in time, predicting divergence from the demand path.

An inspection of the actual path over this time range does indeed show a divergence between the demanded path and the path that the manipulator actually takes.

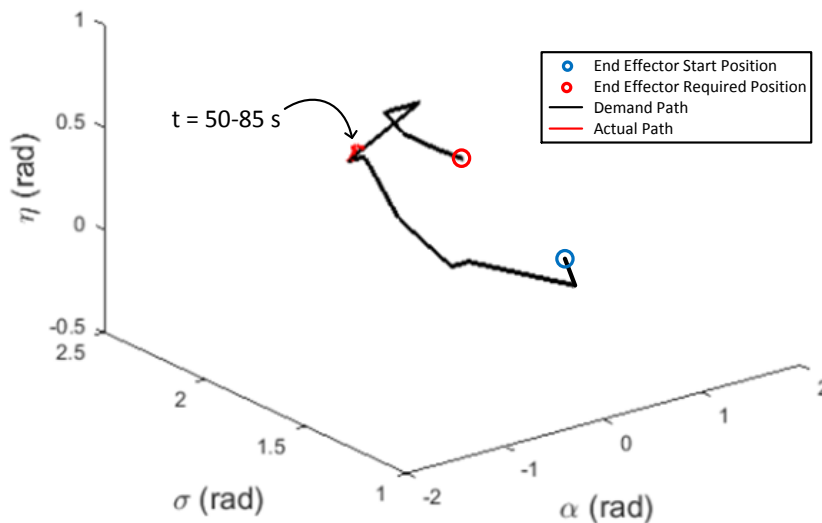


Figure 8-22 C-space demand path for scenario 2 with the actual path plotted only for the time ranges highlighted in Figure 8-21.

In the case of the scenario presented in Figure 8-21 and Figure 8-22, reducing the tolerance for switching to the next waypoint to 0.5° is able to reduce the error experienced by each joint to that of less than 0.015° , or 0.86° . This means that the tuned dynamic model is capable of safely following the path without any collisions.

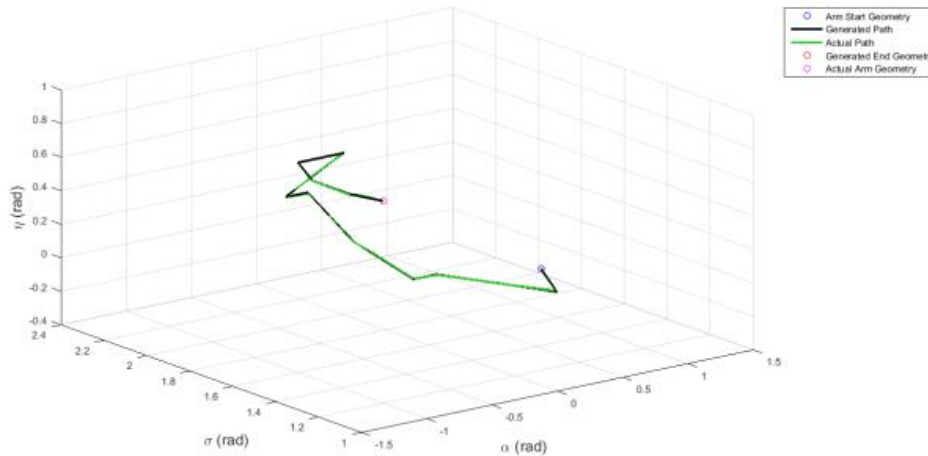


Figure 8-23 Demand and actual C-space paths with a waypoint tolerance of 0.5° in scenario 2.

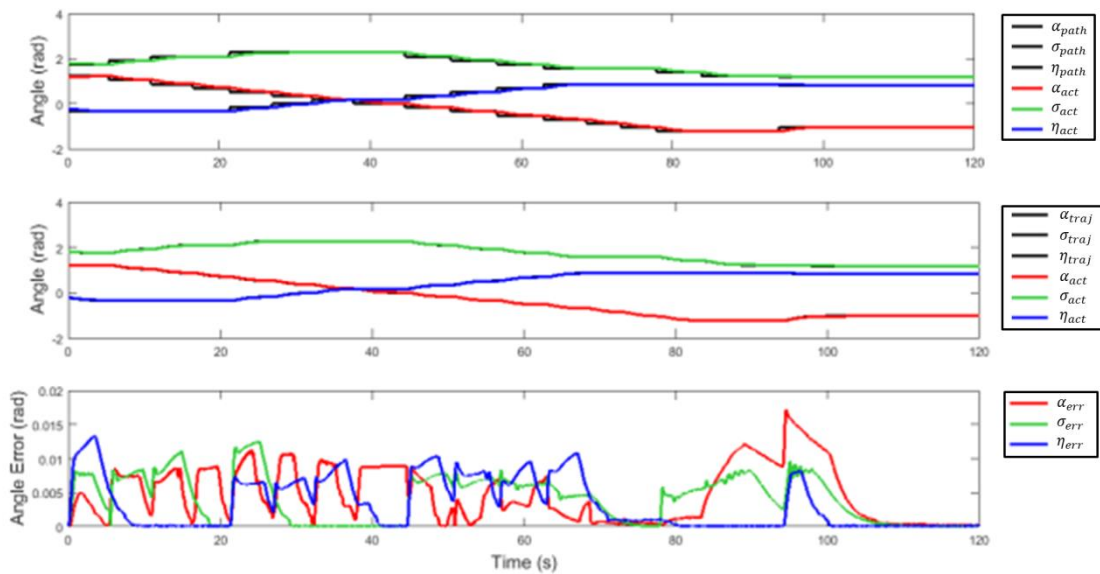
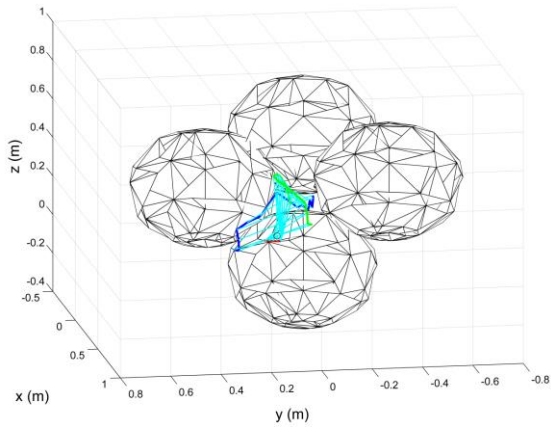


Figure 8-24 Demand path and trajectory, actual path and tracking error over time for a waypoint tolerance of 0.5° in scenario 2.

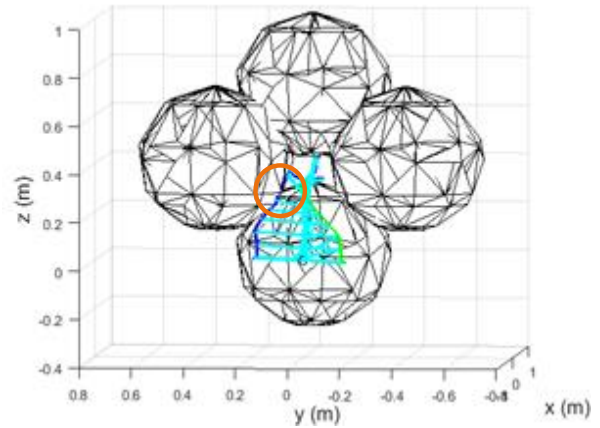
8.2.5 Single Obstacle with Hole

In this scenario 4 spheres are used to construct a single obstacle with a hole through which the arm is required to pass. The centre of each obstacle is placed at $[0.43, 0.38, 0.4]$, $[0.43, -0.38, 0.4]$, $[0.566, 0, 0]$ and $[0.566, 0, 0.7]$ and the radius of each

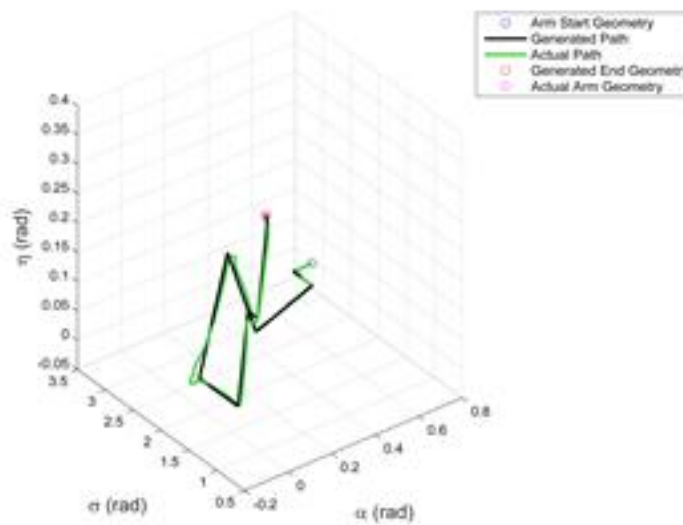
sphere is 0.3 m. This leaves a hole of approximate dimensions of 0.1 m by 0.1 m square through which the arm must travel. Figure 8-25 contains the same series of results as Figure 8-1, but for scenario 5.



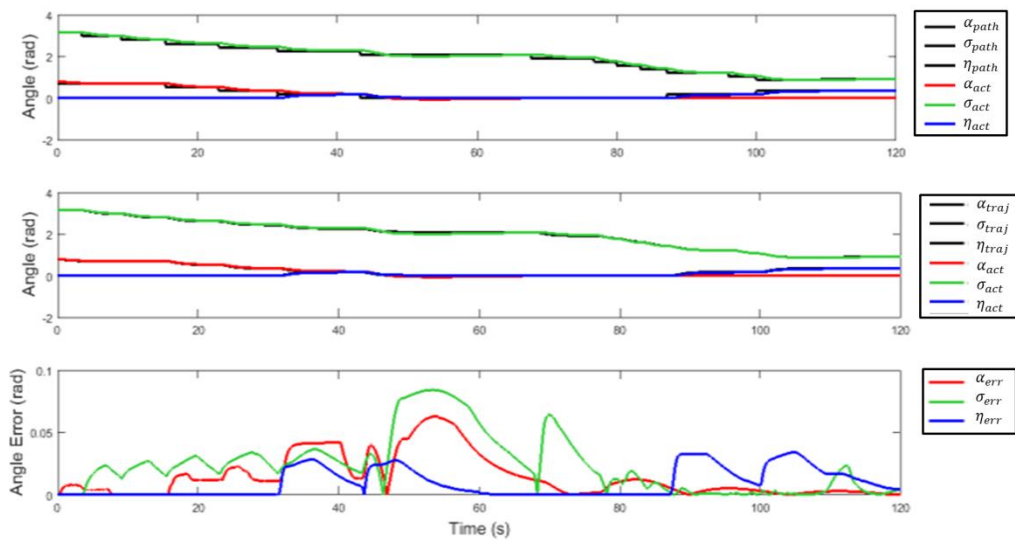
(a) Planned Path in T-space



(b) Actual Path in T-space



(c) Demand and Actual Paths in C-space



(d) Path and Trajectory Demands, Actual Path and Tracking Error.

Figure 8-25 Results of simulation 5 showing the planned (a) and actual (b) path of the manipulator through T-space, the planned and actual path through C-space (b) and the locations of each joint in time (d).

In this case the manipulator arm clearly clips the obstacle. This can be seen highlighted by the orange circle graph c in the above figure. In this case the angle η is offset from the other two over this time range, and the angle error of joint σ reaches 0.084° or 4.8° , which combined causes the divergence.

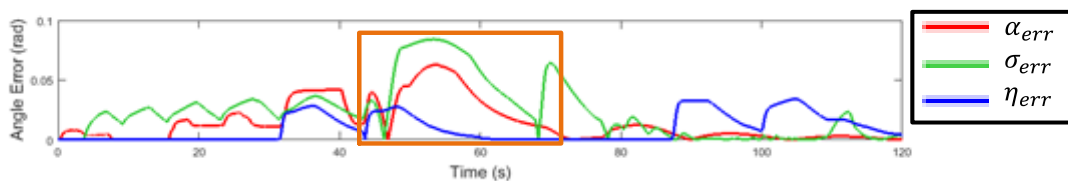


Figure 8-26 Tracking error of manipulator joints in scenario 5. The highlighted areas show where the three joint errors do not match up in time, predicting divergence from the demand path.

The actual path of the manipulator plotted over this range highlights that this is the case.

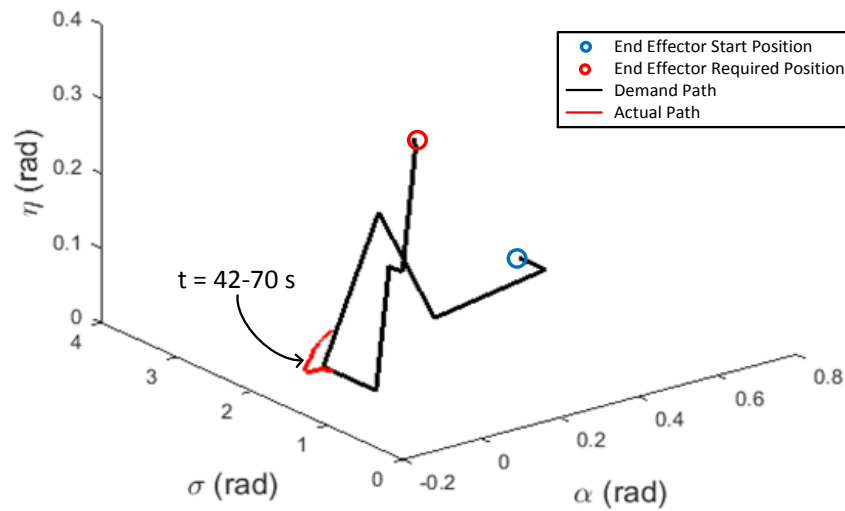


Figure 8-27 C-space demand path for scenario 2 with the actual path plotted only for the time ranges highlighted in Figure 8-26.

Reducing the tolerance on the waypoint selection from 2.225° to 0.5° again solves the problem. The figures below show that with the tighter tolerances on the angular error maximum before waypoints are selected allows the arm to follow the path, even with a small amount of divergence across all joints over the entire path, without colliding with the obstacle.

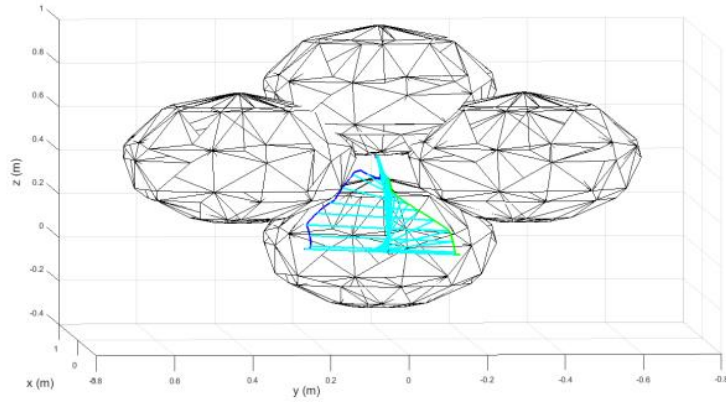


Figure 8-28 Actual path of manipulator arm in scenario 5 with a waypoint tolerance of 0.5° .

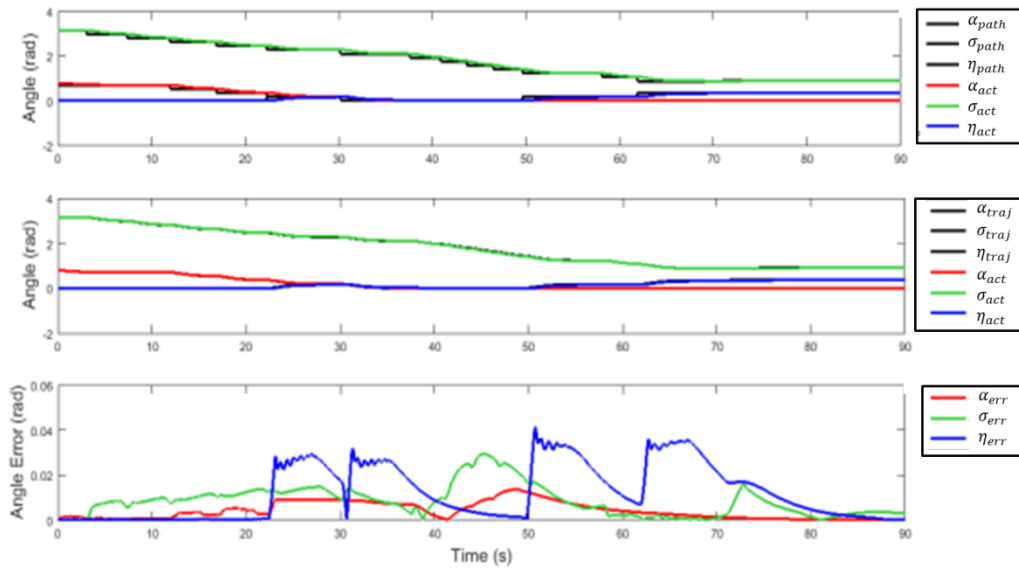


Figure 8-29 Time based plots for planned path and trajectory, actual path and tracking errors in scenario 5 with a waypoint tolerance of 0.5° .

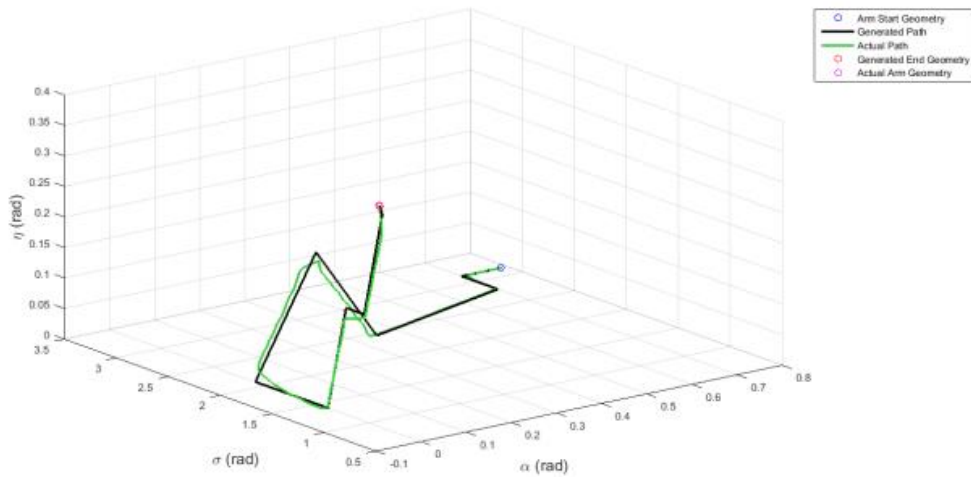
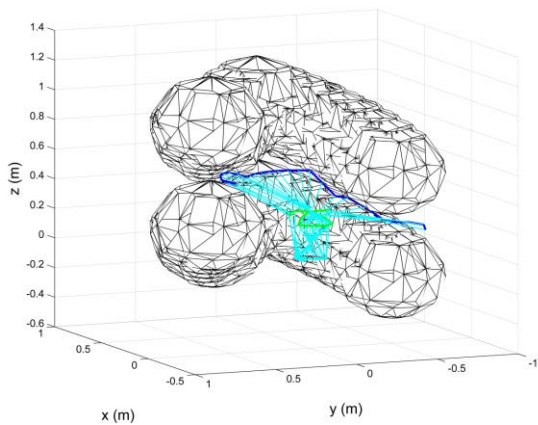


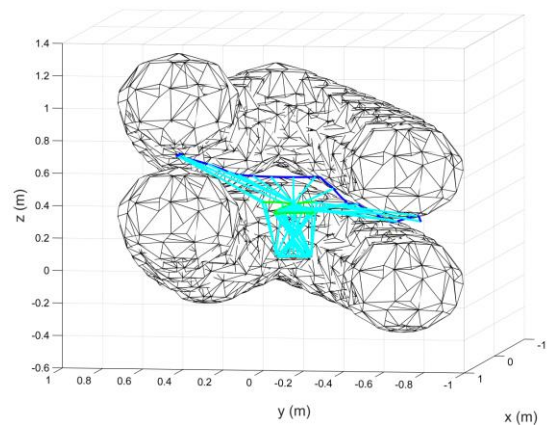
Figure 8-30 Planned and actual C-space paths for the manipulator arm in scenario 5 with a waypoint tolerance of 0.5° .

8.2.6 Narrow Passage Between Two Long Obstacles

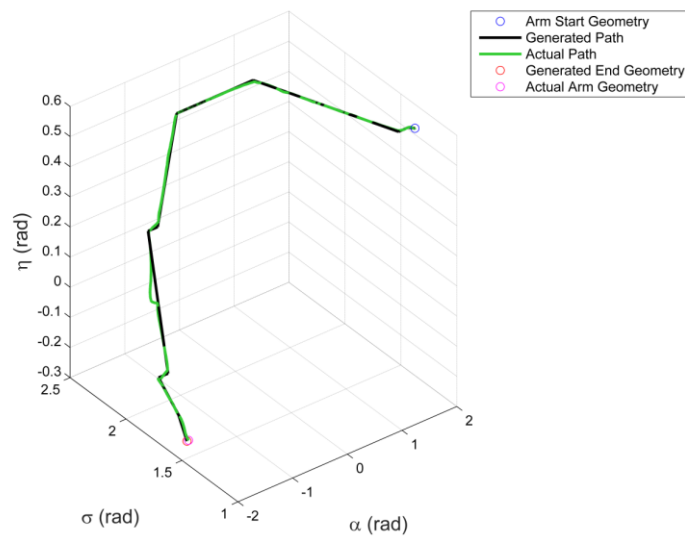
In the final scenario the obstacles chosen are two long, snakelike obstacles situated one above the other, with a narrow gap between them. The arm is required to start at one end of the passage and navigate its way to the other end of the passage. Figure 8-31 contains the same series of results as Figure 8-1, but for scenario 6.



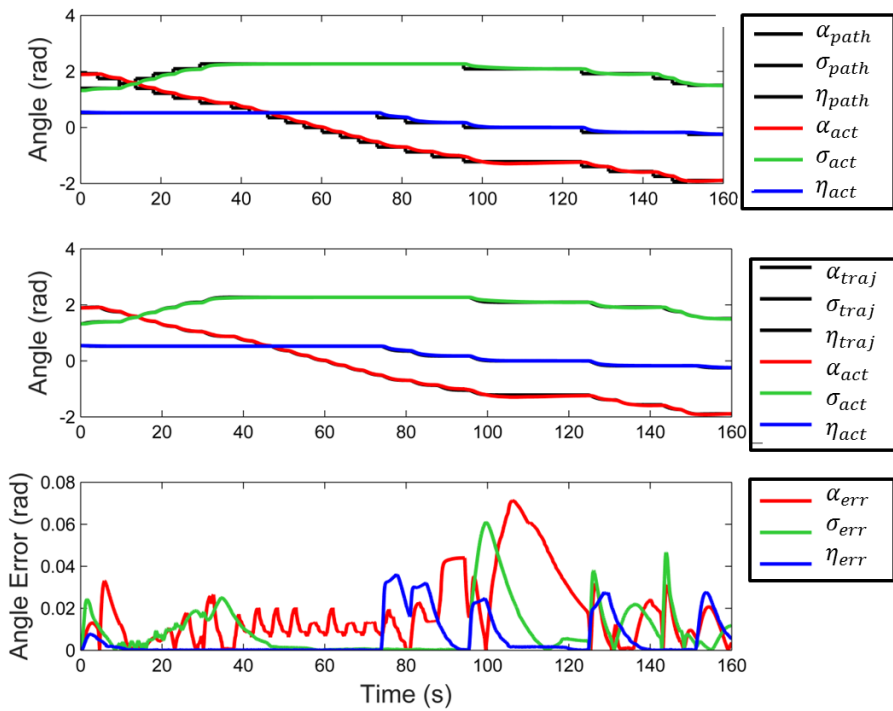
(a) Planned Path in T-space



(b) Actual Path in T-space



(c) Demand and Actual Paths in C-space



(d) Path and Trajectory Demands, Actual Path and Tracking Error.

Figure 8-31 Results of simulation 6 showing the planned (a) and actual (b) path of the manipulator through T-space, the planned and actual path through C-space (b) and the locations of each joint in time (d).

In this case the guidance method was able to successfully plan a path through the environment which achieved the goal of navigating from one end of the trench to the other. The arm was also able to follow the generated trajectory with one appreciable divergence, but this did not have an impact on its ability to navigate through the space without any collisions. Further investigation of this divergence is carried out by again inspecting where the error on each joint are misaligned.

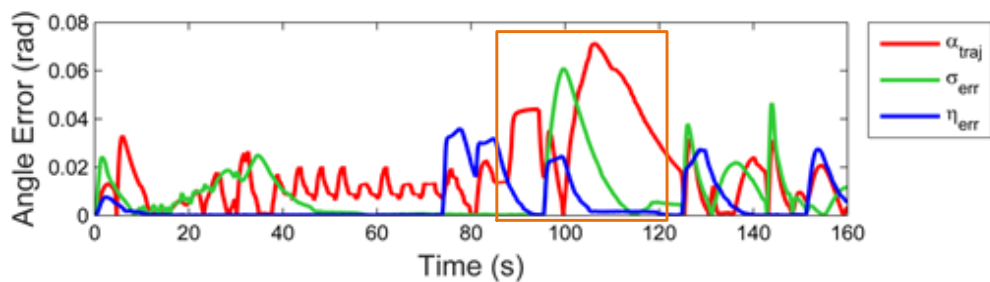


Figure 8-32Tracking error of manipulator joints in scenario 6. The highlighted areas show where the three joint errors do not match up in time, predicting divergence from the demand path.

The divergence of the manipulator from the required path appears to occur between 90 seconds and 120 seconds based upon the misalignment of the error of each joint. The plot of the actual path of the manipulator in C-Space over this time reflects this to be the case.

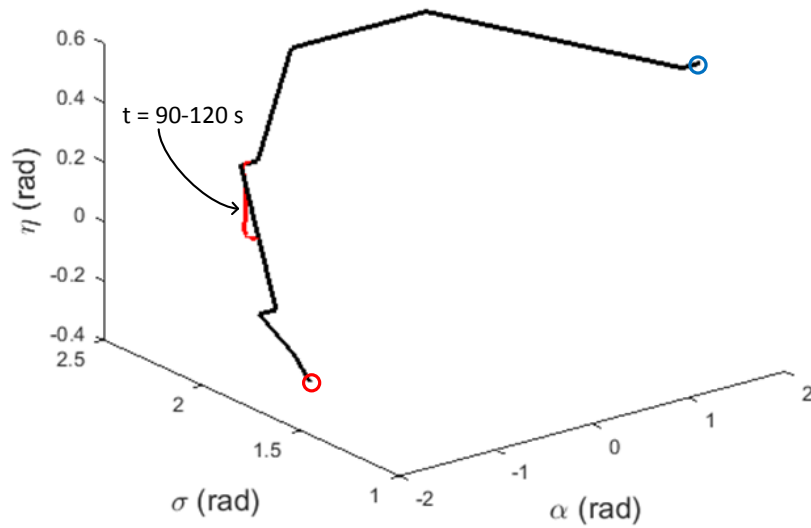


Figure 8-33 C-space demand path for scenario 2 with the actual path plotted only for the time ranges highlighted in Figure 8-32.

A reduction in the tolerance for the waypoint selection once again removes this divergence from the path, and the manipulator arm is again capable of following the generated path without any collision with obstacles.

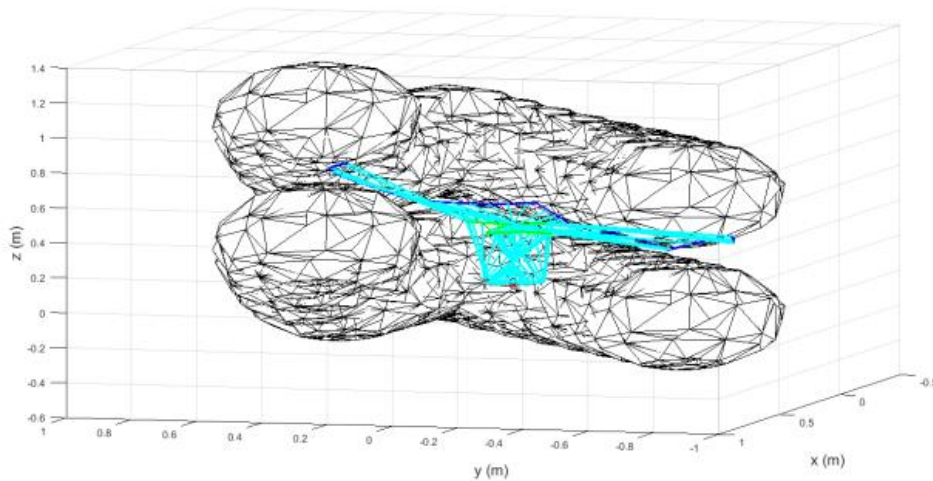


Figure 8-34 Actual path of manipulator arm in scenario 5 with a waypoint tolerance of 0.5° .

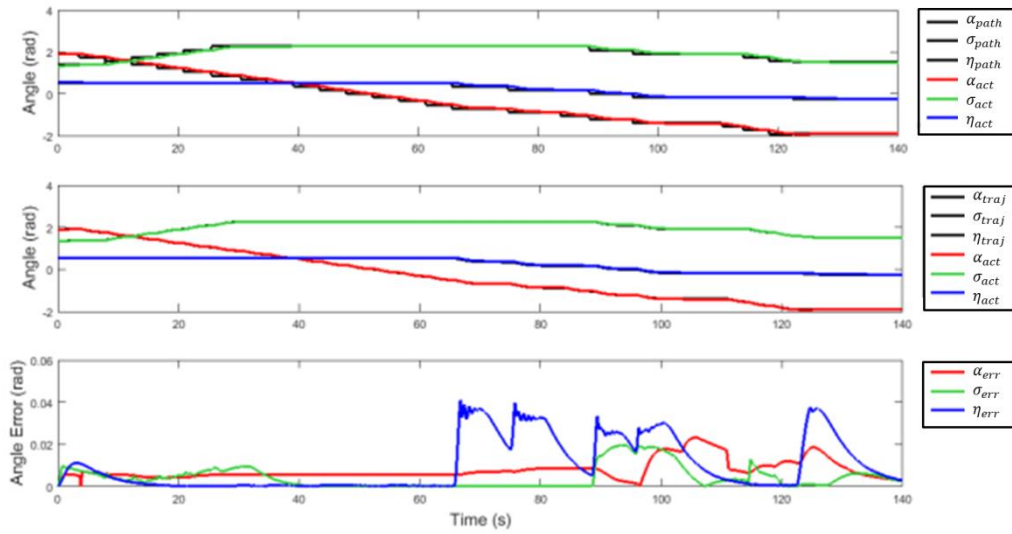


Figure 8-35 Time based plots for planned path and trajectory, actual path and tracking errors in scenario 5 with a waypoint tolerance of 0.5° .

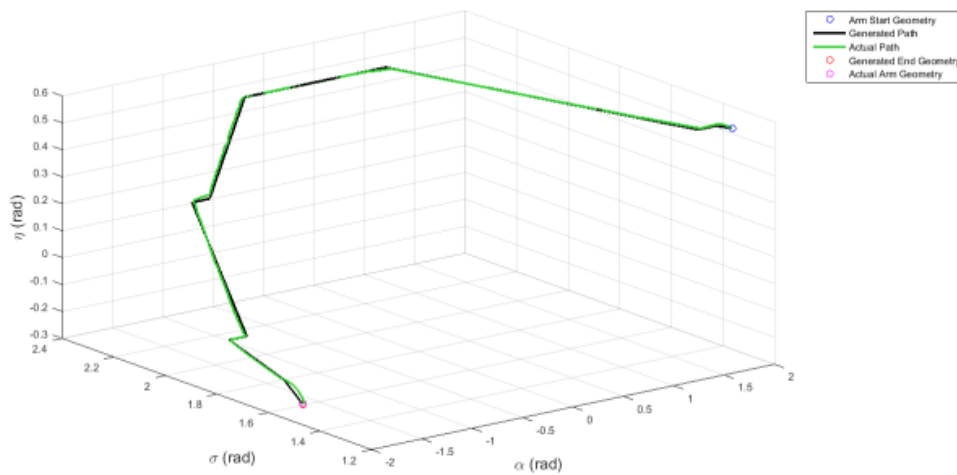


Figure 8-36 Planned and actual C-space paths for the manipulator arm in scenario 5 with a waypoint tolerance of 0.5° .

8.3 Summary

In this chapter a series of simulations was devised to test the ability of the developed method of control and guidance of a 3-DoF robotic manipulator to plan and follow a path through space that would avoid all obstacles in the environment.

The results show that in some cases the manipulator is able to follow the generated paths without any problems, and as such the arm is able to navigate through the environment without any collisions. In these cases one of two situations occurred. Either the arm was able to navigate with a maximum angular error for each joint which was less than the limit specified when generating the path, or the angular error for one or more joints exceeded the limit, but this error the difference between the demand and actual trajectories and so measures not only angular divergence from the path, but also the lag between the time that the demand trajectory reaches a given angle combination and the time the actual trajectory reaches the same angle combination. In essence, the manipulator is following the path, but is behind the demanded trajectory, so the error between them is not representative of a divergence from the path.

In other cases, the manipulator was initially not able to follow the path with a small enough error to avoid a collision. Reducing the tolerance of error that was allowed before the path tracking would allow the next waypoint to be used provided time for each of the joints to slow down as they converged on the waypoint. This prevented overshoots caused by the inertia of the arm when required to change direction and velocity instantaneously. This concept was described in more detail in section 5.5 of the Manipulator Control chapter. A graph illustrating the phenomenon was displayed in Figure 5-34.

In the final chapter of this thesis, the work carried out will be summarised and the overall findings and contributions presented.

9 CONCLUSIONS

This chapter summarises the work carried out in this thesis in the context of the conclusions drawn and the contributions made. The work presented in this thesis was carried out to achieve the aim laid out in Chapter 1: to create novel path-generation algorithm for a three-degree-of-freedom robotic manipulator arm to provide the ability for safe navigation around obstacles in dangerous environments such as those encountered during a mission of IED detection and disposal.

The first objective specified in Chapter 1 was the derivation of a dynamic model of a 3-DoF robotic manipulator arm, which was implemented in Chapter 4. This was carried out using Newton-Euler mechanics with d'Alembert's principle rather than Lagrangian mechanics since the latter required significantly more computations (on the order of 3 times more for the specific problem). The resulting model was qualitatively validated with a series of input torques intended to produce a predicted set of results. The resulting outputs of the system were compared to the predicted result in order to validate whether the system is behaving in the expected manner and the model was found to behave as expected.

The second objective specified in Chapter 1 was the implementation of a suitable control schema for the developed dynamic model, which was implemented in Chapter 5. The selected control method was a gain-scheduled PID controller and the gains were selected by optimisation. Several optimisation methods including Genetic Algorithms, Least Squares Minimisation and the Nelder-Mead method were investigated. The least squares minimisation method followed by the Nelder-Mead method were found to require the shortest computation time per optimisation but the Genetic Algorithm displayed the capability of a fitness value hundreds of times better for the manipulator arm tuning problem, therefore it was implemented in order to carry out the optimisation and select a set of gains for the controller.

The third objective specified in Chapter 1 was to develop a guidance method for safe navigation of the controlled dynamic model through close-proximity environments. This implementation was presented in Chapters 6 and 7 of the thesis. This method

requires two parts: The first is to convert sensor data about obstacles from the Euclidean domain into the joint-angle domain before carrying out the path generation on a point mass in the joint angle domain; The second is to carry out path generation on the C-space map created by the conversion of obstacle data into the joint angle domain. To carry out the conversion to C-space, the inverse kinematics of the manipulator arm were required to be solved. Several different solutions were investigated and the method which was able to carry out the conversion of multiple obstacles in the shortest time is presented in section 6.1.2 of this thesis starting on page 219. This method required trigonometry only and was able to convert an obstacle set containing on the order of 10000 points in less than 0.1 seconds.

Chapter 7 continues the development of the guidance technique by investigating methods of path planning using a node graph. The results of the comparison between Dijkstra's Algorithm and the Bellman-Ford Algorithm confirm the comparison of computational complexity included in the literature review. Three methods were investigated and Dijkstra's Algorithm was implemented in the guidance method since it provided the shortest path with no collisions with obstacles in the shortest computation time.

The final objective specified in Chapter 1 was the validation of the guidance method to assess its strengths and limitations. This was presented in Chapter 8 using a series of scenarios with increasing numbers of obstacles.

9.1 Further Conclusions

Further conclusions drawn as a result of the work carried out in this thesis are described in the following paragraphs.

The investigation carried out in this thesis in order to develop a guidance method suitable for path planning in close-proximity environments was able to gain significant insights into the process of tuning PID controllers for the type of non-linear system that is involved when dealing with robotic manipulator arms. The resultant optimisation of PID control gains for the non-linear dynamic model derived in Chapter 4 showed that for non-linear but continuous operation spaces for manipulator arms, the gain profile produced would also be non-linear but continuous.

This also assisted with a second finding during the optimisation of the control for the derived non-linear dynamic model. After having optimised the controller for the first range of motion of the arm using a random initial estimate for the gains, the use of the resultant solution as the initial estimate for the next range of motion (which is a discrete step of the continuous range of operation of the manipulator) reduced the number of required generations of the optimiser to converge on the specified fitness value to half of the original optimisation; i.e. use of the solution set K_1^2 from $\begin{bmatrix} \alpha \\ \sigma \\ \eta \end{bmatrix}_1 \rightarrow$

$\begin{bmatrix} \alpha \\ \sigma \\ \eta \end{bmatrix}_2$ as the initial predicted set of gains for the range of motion $\begin{bmatrix} \alpha \\ \sigma \\ \eta \end{bmatrix}_2 \rightarrow \begin{bmatrix} \alpha \\ \sigma \\ \eta \end{bmatrix}_3$ reduced the number of required generations for that optimisation by approximately half of that of the original optimisation.

In Chapter 7, when converting a generated path to a trajectory, using a time constant that is slower than that of the slowest joint in the manipulator arm reduces the absolute error (or divergence) from the demanded path by the arm, but has very little impact on the tracking error of the manipulator actual location from the time-based trajectory demand. This is the case because the addition of time-based information to the path means that the slowest joint follows the angle demand as fast as it can, but the two other joints, which are faster than the slowest joint, are limited to the speed at which the trajectory demand changes. This means that the three joints arrive at the waypoint almost simultaneously. The reason for the tracking error between the manipulator actual position and the demanded trajectory is the dynamics of each joint, which provide a lag in time between the trajectory arriving at a location and the dynamic model arriving at the same location.

It was found during the validation of the guidance technique in Chapter 8 that the angle tolerance that the manipulator joints would have to be within from a waypoint before moving to the next waypoint had a large effect on the ability of the arm to follow the demanded path. A smaller tolerance (i.e. error between demanded joint angles and actual joint angular positions) before switching to the next waypoint allows $\dot{\theta}$ to be reduced further, leading to less overshoot immediately after switching to the next waypoint caused by the inertia of the manipulator arm, therefore less divergence from the path.

The divergence from the demanded path can be predicted by inspecting error between the demanded path and actual response of each joint. Where the error on each joint does not occur at the same time, divergence from the path occurs. This is useful when assessing whether the manipulator has followed the demanded path since error between the actual manipulator and demand trajectory will always display a tracking error due to the dynamics of the manipulator arm.

9.2 Research Contributions

The research carried out in this thesis is novel because it implements a path-generation and following algorithm for a manipulator arm that is designed for use on a skid-steer vehicle with the main purpose being IED disposal. This means that there is no a priori knowledge of the environment at the start of every new mission, and the algorithm has to generate a map in real-time of the environment in C-space and generate a safe path around obstacles in order to reach the target end-effector position.

While there is work presented in the literature on manipulator guidance that seeks to solve the problem of path planning for the entire manipulator or attempts to solve the problem of path planning for an end effector in real-time, no work exists in the literature which combines all these factors, especially in a completely unknown environment, and in such close proximity to obstacles.

This work has the potential to be expanded to higher degrees-of-freedom than the 3-DoF manipulator used in this research, and initial investigation carried out in 11Appendix B has shown promising results, which show that this method is applicable in real-time for higher degrees-of-freedom, currently up to 9-DoF. Work using graph theory in C-space has not been shown out in the literature, and that it has been implemented here shows that it can be used in this way, and highlights its power in such a control domain since it has the capability to handle path planning in an unlimited number dimensions providing that an adjacency matrix can be calculated for n-dimensional points in a space.

In the literature many methods of path planning are investigated. These were further explored in Chapter 2. While there are merits to each of the methods investigated,

none of them had the potential to be singularly applied to the application of IED disposal. In this application the path-planning problem requires to be solved in real-time, which some methods are able to do. However, this application also requires path planning for a 3-DoF manipulator in a 3-D environment, which makes path planning methods for planar manipulators inappropriate for this application. Some of the methods for path planning discussed in the review of literature can satisfy these two requirements but do not consider the entire manipulator, only the end effector. Most of the methods presented in the literature are concerned with obstacle avoidance, only approaching obstacles should there be no other option. However, in this application it is reasonable to assume that IEDs are likely to be hidden amongst or inside obstacles, so it is a requirement to get close to objects that may be hiding the IED and follow their edges without collision since this could cause detonation. In this application all of these requirements must be considered.

The major contribution of the research presented in this thesis is that existing methods in the areas of robotic manipulator guidance and control, environment mapping and path planning have been drawn together and combined in such a way as to develop a guidance technique that is capable of satisfying all of the following attributes.

- The developed technique is capable of path planning in high complexity environments in real-time.
- It has the potential to be applicable to n-DoF manipulator arms for 3-D environments.
- This method is capable of dealing with unknown environments as the manipulator arm is installed on a mobile vehicle therefore the environment is not a permanent reachable space that can be mapped a priori.
- Joints are not decoupled for path generation and so there is only one trajectory, therefore trajectories do not need to be resynchronised.
- This method is able to track around obstacles in the control domain which translates into edge following in Euclidean space.

9.3 Advantages and Limitations of the Technique

The advantages of this algorithm are that it very quickly generates a trajectory that allows for the entire arm to avoid obstacles in the environment while still achieving the objective. The algorithm is very powerful in that it converts the three-dimensional world into a map that includes the entire control domain data required to avoid obstacles within it. This map will have the same dimensionality as the number of degrees-of-freedom that are being controlled, but even at this state $n \times 3$ -D paths (one for each link in an arm) are converted into $1 \times n$ -D problem. A further advantage of the algorithm is that converting the path-generation problem it into a graph theory problem, which uses a 2-D connectivity matrix between nodes regardless of the number of coordinate dimensions the nodes have, the problem is always reduced to one of two dimensions. This means that even though the output of the algorithm is an n -dimensional trajectory, the solution that is calculated is in fact only 2-D, so for any n , an $n \times 3$ -D problem becomes a 2-D one. This is very powerful since it reduces the computational complexity of the path-planning problem significantly, especially at higher degrees-of-freedom.

An advantage of using C-space as the domain of choice for the guidance method is that both environment data and predicted measurement errors in sensors and errors caused by the resolution of sensors can be take into account when generating the safe boundaries around obstacles in C-space.

This method also has the advantage of using a pre-existing node graph representing the accessible space of the manipulator arm with inaccessible nodes being temporarily removed when an obstacle is detected meaning that the node graph does not need to be repeatedly built, reducing computational overhead significantly. A drawback of this method however, is that the computational memory available for use in the system will limit the resolution of the node graph since smaller spacing between nodes means more nodes in the graph.

A limitation to the developed technique is that inverse kinematic solutions become more complex with increasing degrees-of-freedom and for an arm with more degrees-of-freedom than the system used in this thesis the solution is computationally intensive and takes significant time. If this problem can be solved then the technique becomes very transferrable, especially since manufacturers of

automated manipulator arms will often derive the forward and inverse kinematics during the design phase, so this information will be known prior to implementation of the path-following algorithm. Also, once the inverse kinematic solution is known for a specific arm, it does not need to be calculated again, and it does not need to be calculated on-line, in real time.

10 FUTURE RESEARCH WORK

This chapter outlines the work which could be carried out in the future based on the technique and investigation presented in this thesis. This future work could build upon the contributions to the field of robotic manipulator guidance that are presented here.

Further work that could be carried out in this area involves the mapping and path planning when obstacles in the environment are no longer static. Since the developed method is able to carry out these functions in real time, both the map and generated path can be renewed repeatedly, there is no immediate issue from the path-planning point of view. However, since the obstacles in the environment would now be in motion, the path that has been generated may be obstructed at some time in the future. The brief discussion on decision making which was carried out in Section 2.4 regarding how to weight node graphs with probability of a vertex being broken could be investigated here. Using historical data about the previous and current states of obstacles in motion, predictions could be made about the future motion of the obstacles, informing the probability weighting of the node graph, and driving the manipulator arm through paths which are less likely to be obstructed.

The concept of using the joint-angle domain to map the environment and generate a trajectory to guide the arm through it has the potential to be extended further than only the three degree-of-freedom robotic manipulator arm. Given that a node graph can be represented by a two-dimensional adjacency matrix which indicates which nodes in the graph are connected directly to one another (adjacent), the coordinates in the space that the nodes occupy do not have to be two or three-dimensional, they can have coordinates in more than three dimensions. This would allow for the conversion of a Euclidean map into a map in the control domain, where the number of dimensions in the map represent the degrees-of-freedom of a four-, five-, six-, etc degree-of-freedom manipulator, hence the map would have four, five or six dimensions, respectively. This technique has the potential for providing a single trajectory for an entire controllable vehicle with any number of control inputs.

This method of trajectory generation requires the knowledge of the inverse kinematics of the robotic manipulator arm. The inverse kinematics often provide multiple solutions to the problem of joint angle calculation based on the end effector location. The number of joint angle solutions becomes larger for higher degrees of freedom and this causes the technique to become very calculation intensive, given that each solution must be calculated for points along each joint in the arm. Therefore future work could concentrate on solving the inverse kinematics problem of calculating multiple solutions to the joint angle combinations for a specific end effector position.

During the development of the C-space mapping method, the issues of LIDAR sensor and servo encoder error were briefly discussed as potential problems that could affect the success of the manipulator arm to follow any planned path safely, without collisions with obstacles; the given solution was to expand the safe boundary around obstacles in C-space in order to compensate for this. Future work could investigate the effects of these errors in more detail and propose solutions to these effects. A further investigation in this area is to determine how the safe boundary around obstacles in C-space must be expanded to take into account the thickness of the arm.

A final area for future work would be to investigate the optimisation of the path generation to generate minimum angular distance, minimum Euclidean distance, or even minimum energy paths. This would involve the development of a solution for time or energy optimal solutions to the path-planning problem by this method.

11 CLOSING SUMMARY

Based on the aim and objectives laid out in Chapter 1 of this thesis a path planning technique for mobile 3-DoF manipulators was developed. This technique can, in real-time, plan a safe path through an environment towards and around obstacles, not just for the end effector but with consideration to the manipulator arm in its entirety. The resultant technique operates in a problem space of path planning for mobile robotic manipulators in a solution space which uses node graphs in C-space. It is designed to overcome the problems inherent to close-proximity environments, such as tight clearance between obstacles and high chances of collisions. Simulation-based validation of the method was carried out in order to answer a research question that was specified in Chapter 1 prior to the development of the technique:

“Is it possible and feasible to implement a path-generation algorithm that is capable of guiding a robotic manipulator arm through a close-proximity environment with the aim of carrying out Improvised Explosive Device disposal missions?”

The work reported in this thesis demonstrates that it **is** possible and feasible to implement a path-planning algorithm that can satisfy these requirements, and it is achievable in real-time.

REFERENCES

- Ab-Rahman, A. A. H. et al., 2005. *Vestro: Velocity Estimation Using Stereoscopic Vision*. pp. 120-124.
- Ahmadi, M., Polotski, V. & Hurteau, R., 2000. *Path Tracking Control of Tracked Vehicles*. San Francisco, California. pp. 2938-2943.
- Akai, D. et al., 2006. Pyroelectric infrared sensors with fast response time and high sensitivity using epitaxial Pb(Zr, Ti)O₃ films on epitaxial γ -Al₂O₃/Si substrates. *Sensors and Actuators A: Physical*, 14 August. pp. 111-115.
- Alexander, Q., 2003. *A Survey of Robotic Coverage*, Alabama:
- Anon., 2001. Depth-first search. In: *Introduction to Algorithms*. s.l.:MIT Press and McGraw-Hill, pp. 540-549.
- Anon., 2001. Dijkstra's Algorithm. In: *Introduction to Algorithms*. s.l.:MIT Press and McGraw-Hill, pp. 595-601.
- Apkarian, P. & Adams, R. J., 1998. Advanced gain-scheduling techniques for uncertain systems. *IEEE Transactions on Control System Technology*, January. pp. 21-32.
- Audenaert, K., Peremans, H., Kawahara, Y. & Van Campenhout, J., 1992. *Accurate Ranging of Multiple Objects Using Ultrasonic Sensors*. pp. 1733-1738.
- Back, T., 1995. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford: Oxford University Press.
- Barshan, B. & Durrant-Whyte, H. F., 1995. Inertial Navigation Systems for Mobile Robots. *IEEE Transactions on Robotics and Automation*, 11(3), pp. 328-342.
- Bellman, R., 1958. On a routing problem. *Quarterly of Applied Mathematics*, 87-90. Volume 16.
- Beltran-Gonzalez, C. et al., 2007. *Methods and techniques for intelligent navigation and manipulation for bomb disposal and rescue operations*. Rome, IEEE.

- Benet, G., Blanes, F., Simo, J. E. & Perez, P., 2002. Using Infrared Sensors for Distance Measurement in Mobile Robots. *Robotics and Autonomous Systems*, 40(4), pp. 255-266.
- Biggs, N. E., Lloyd, K. & Wilson, R. J., 1976. In: *Graph Theory, 1736-1936*. :Oxford University Press.
- Binh, T., 1999. *A multiobjective evolutionary algorithm. The study cases.*, Barleben: Institute for Automation and Communication.
- Binh, T. & Korn, U., 1997. MOBES: A Multiobjective Evolution Strategy for Constrained Optimisation Problems. *Proc. International Conference on Genetic Algorithms*, pp. 176-182.
- Bjork, A., 1996. *Numerical methods for least squares problems*. :Siam.
- Blanchett, T. P., Kember, G. C. & Dubay, R., 2000. PID Gain Scheduling Using Fuzzy Logic. *ISA Transactions*, July, 39(3), pp. 317-325.
- Buchberger, M., Jorg, K.-W. & von Puttkamer, E., 1993. *Laserradar and Sonar Based World Modeling and Motion Control for Fast Obstacle Avoidance of the Autonomous Mobile Robot MOBOT-IV*. pp. 534-540.
- Burguera, A., González, Y. & Oliver, G., 2010. *Underwater scan matching using a mechanical scanned imaging sonar*.
- Burguera, A., Oliver, G. & González, Y., 2010. *A trajectory based Framework to Perform Underwater SLAM using Imaging Sonar Scans*.pp. 1-6.
- Butenko, S., Murphey, R. & Pardalos, P. eds., 2002. Path Planning for Unmanned Aerial Vehicles in Uncertain and Adversarial Environments. In: *Cooperative Control: Models, Applications and Algorithms**. :Kluwer.
- Caracciolo, L., De Luca, A. & Iannitti, S., 1999. *Trajectory Tracking of a Four-Wheel Differentially Driven Mobile Robot*. Detroit, Michigan, US, pp. 2632-2638.
- Caravita, L. et al., 2007. Control Strategies Applied to Waypoint Navigation and Obstacle Avoidance Navigation. *Advances in Control and Optimization of Dynamical Systems*, pp. 203-210.

Cassandra, A. R., Kalebling, L. P. & Kurien, J. A., 1996. *Acting Under Uncertainty: discrete Bayesian models for mobile-robot navigation*. pp. 963-972.

CBS, N. & McCarthy, C., 2010. *Facebook: Once Social Graph to Rule Them All?*. [Online]

Available at: <http://www.cbsnews.com/news/facebook-one-social-graph-to-rule-them-all/>

[Accessed 03 January 2016].

Chang, H. J., Lee, C. S. G., Lu, H.-S. & Hu, Y. C., 2007. P-SLAM: Simultaneous Localization and Mapping With Environmental-Structure Prediction. *IEEE Transactions on Robotics*, April, 23(2), pp. 281-293.

Chen, Y., Wang, Y. & Yu, X., 2012. *Obstacle Avoidance Path Planning Strategy for Robot Arm Based on Fuzzy Logic*. Guangzhou, China, IEEE, pp. 1648-1653.

Chetty, R. M. K. & Ponnambalam, S. G., 2012. *A Heuristic Approach Towards Path Planning and Obstacle Avoidance Control of Planar Manipulator*. Kuala Lumpur, Malaysia, Springer Berlin Heidelberg, pp. 1-11.

Cloutier, J. R., 1997. *State-Dependent Riccati Equation Techniques: An Overview*. Albuquerque, New Mexico, pp. 932-936.

Colyer, R. E. & Economou, J. T., 1997. *Modelling and Simulation of 4x4-wheel skid-steer vehicles*. Honolulu, U.S.A., pp. 301-304.

Colyer, R. E. & Economou, J. T., 1998. *Comparison of Steering Geometries for Multi-Wheeled Vehicles by Modelling and Simulation*. Tampa, Florida, U.S.A., pp. 3131-3133.

Colyer, R. E. & Economou, J. T., 1999. Soft Modelling and fuzzy logic control of wheeled skid-steer electric vehicles with steering prioritisation. *International Journal of Approximate Reasoning*, Volume 22, pp. 31-52.

Coulter, R. C., 1992. *Implementation of the Pure Pursuit Tracking Algorithm*, Pittsborough: s.n.

Csorba, M., 1997. *Simultaneous Localisation and Mapping*, Oxford:

- Deb, K., 2002. *Multiobjective optimization using evolutionary algorithms*. Repr. ed. Chichester: J Wiley & Sons.
- Denavit, J. & Hartenberg, R. S., 1955. A Kinematic Notation for Lower-Pair Mechanics Based on Matrices. *ASME Journal Of Applied Mechanics*, Volume 23, pp. 215-221.
- Dijkstra, E. W., 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, Volume 1, pp. 269-271.
- Ding, H., Zhou, M. & Stursberg, O., 2009. *Optimal Motion Planning for Robotic Manipulators with Dynamic Obstacles using Mixed-Integer Linear Programming*. Thessaloniki, Greece, IEEE, pp. 934-939.
- Dissanayake, M. W. M. G. et al., 2001. *A Solution to the Simultaneous Localization and Map Building (SLAM) Problem*. IEEE, pp. 229-241.
- Doitsidis, L., Valavanis, K. & Tsourveloudis, N., 2002. *Fuzzy Logic Based Autonomous Skid Steering Vehicle Navigation*. Washington, DC., pp. 2171-2177.
- Dorf, R. C. & Bishop, R. H., 2006. In: *Modern Control Systems*. s.l.:Pearson, p. 65.
- dos Santos, R. R., Steffen, V. J. & de F. P. Saramago, S., 2008. Robot Path Planning in Constrained Workspace by Using Optimal Control Techniques. *Multibody System Dynamics*, 19(1-2), pp. 159-177.
- Doucet, A., de Freitas, N., Murphy, K. & Russell, S., 2000. *Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks..*, pp. 176-183.
- Drumbeller, M., 2009. Mobile Robot Localization using Sonar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Issue 2, pp. 325-332.
- D'Silva, T. & Miikkulainen, R., 2009. Learning Dynamic Obstacle Avoidance for a Robot Arm Using Neuroevolution. *Neural Processing Letters*, 30(1), pp. 59-69.
- Dubins, L. E., 1957. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics*, 79(3), pp. 497-516.

- Dubowsky, S. & Vance, E. E., 1989. *Planning Mobile Manipulator Motions Considering Vehicle Dynamic Stability Constraints*. Scottsdale, IEEE, pp. 1271-1276 vol.3.
- Duda, R. O. & Hart, P. E., 1972. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1), pp. 11-15.
- Economou, J. T., 2002. *Fuzzy Logic Force Modelling*. Anchorage, Alaska, s.n., pp. 525-530.
- Economou, J. T. & Colyer, R. E., 2000. *Modelling of Skid Steering and Fuzzy Logic Vehicle Ground Interaction*. Chicago, Illinois, , pp. 100-104.
- Economou, J. T., Colyer, R. E., Tsourdis, A. & White, B. A., 2002. *Fuzzy Logic Approaches for Wheeled Skid-Steer Vehicles*. pp. 990-994.
- Elfes, A., 1989. Using occupancy grids for mobile robot perception and navigation. 22(6), pp. 46-57.
- Elfes, A. & Matthies, L., 2007. *Sensor Integration for Robot Navigation: Combining Sonar and Stereo Range Data in a Grid-Based Representation*. pp. 1802-1807.
- Endo, D., Okada, Y., Nagatani, K. & Yoshida, K., 2007. *Path Following Control for Tracked Vehicles Based on Slip-Compensating Odometry*. pp. 2871-2876.
- Euler, L., 1766. In: *Commentationes Arithmeticae Collectae*. St. Petersburg: 66-70.
- Everett, L. J. & Suryohadiprojo, A. H., 1988. *A Study of Kinematic Models for Forward Calibration of Manipulators*. pp. 798-800.
- Fang, Q. & Xie, C., 2004. *A Study on Intelligent Path Following and Control for Vision-Based Automated Guided Vehicles*. pp. 4811-4815.
- Ford, L. R. & Fulkerson, D. R., 1956. Maximal flow through a network. *Canadian Journal of Mathematics*, Volume 8, pp. 399-404.
- Frezza, R., 1999. *Path Following For Air Vehicles in Coordinated Flight*. pp. 884-889.
- Gasparetto, A. & Zanutto, V., 2007. A new method for smooth trajectory planning of robot manipulators. *Mechanism and Machine Theory*, April, 42(4), pp. 455-471.

- Gasparetto, A. & Zanotto, V., 2008. A Technique for Time-Jerk Optimal Planning of Robot Trajectories. *Robotics and Computer-Integrated Manufacturing*, Volume 24, pp. 415-426.
- Gelbart, A. et al., 2003. *FLASH Lidar Data Collections in Terrestrial and Ocean Environments*. pp. 27-38.
- Goldberg, D., 1989. *Genetic Algorithms in Search, Optimisation and Machine Learning*, Reading, MA: Addison-Wesley Profesional.
- Goldenberg, A. et al., 2000. USA, Patent No. US6113343 A.
- Gutin, G., Yeo, A. & Zverovich, A., 2002. Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP.. *Discrete Applied Mathematics*, Volume 117, pp. 81-86.
- Haddad, W. M., Bernstein, D. S. & Mustafa, D., 1991. Mixed-norm H_2/H_∞ regulation and estimation: The discrete-time case. *Systems and Control Letters*, Volume 16, pp. 235-247.
- Hanebeck, U. D. & Schmidt, G., 1996. *Set theoretic localization of fast mobile robots using angle measurement techniques*. Minneapolis, Minnesota, pp. 1387-1394.
- Hanna, B., Chai, B.-B. & Hsu, S., 2005. *Wide-Area Terrain Mapping By Registration of Flash LIDAR Imager*. pp. 193-207.
- Hartenberg, R. S. & Denavit, J., 1964. *Kinematic Synthesis of Linkages*. New York: McGraw-Hill.
- Haschke, R., Weitnauer, E. & Ritter, H., 2008. *On-Line Planning of Time-Optimal, Jerk-Limited Trajectories*. Nice, pp. 3248-3253.
- Haupt, R. L. & Ellen, S., 2004. *Practical genetic algorithms with DC-Rom*. 2nd ed. New York: J. Wiley & Sons.
- Hellström, T. & Ringdahl, O., 2005. *Autonomous Path Tracking Using Recorded Orientation Steering Commands*.

Hellström, T. & Ringdahl, O., 2006. Follow the Past - A Path Tracking Algorithm for Autonomous Vehicles. *International Journal of Vehicle Autonomous Systems*, 4(2-4), pp. 216-224.

He, P.-J., Gao, S.-S., Jiao, Y.-L. & Zheng, P., 2010. A Robust Adaptively Filtering Algorithm in GPS/DR Integrated Navigation. pp. 3742-3745.

Hokuyo Automatic Co., Ltd., 2016. *Hokuyo UTM-30LX*. [Online] Available at: https://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html [Accessed 02 Jan 2016].

Hota, S. & Ghose, D., 2010. Rectilinear Path Following in 3D Space. *Trends in Intelligent Robotics*, 103(4), pp. 210-217.

Huang, S. & Dissanayake, G., 2006. *Convergence Analysis for Extended Kalman Filter based SLAM*. Orlando, pp. 412-417.

Ibraeem, M., 2010. *Gyroscope-enhanced Dead Reckoning Localization System for an Intelligent Walker*.

Indiveri, G., Nuchter, A. & Lingemann, K., 2007. *High Speed Differential Drive Mobile Robot Path Following Control With Bounded Wheel Speed Commands*. Rome, Italy, pp. 2202-2207.

Julier, S. J. & Uhlmann, J. K., 2004. *Unscented Filtering and Nonlinear Estimation*. pp. 401-422.

Jung, S. & Hsia, T. C., 1995. *New Neural Network Control Technique for Non-model Based Robot Manipulator Control*. pp. 2928-2933.

Kawato, M., Furukawa, K. & Suzuki, R., 1987. A Hierarchical Neural-Network Model For Control And Learning of Voluntary Movement. *Biological Cybernetics*, 57(3), pp. 169-185.

Kim, S. K., Silson, P., Tsourdos, A. & Shanmugavel, M., 2010. Dubins Path Planning of Multiple Unmanned Airborne Vehicles for Communication Relay. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 225(1), pp. 12-25.

- Kladis, G., Economou, J., Knowles, K. T. A. & White, B. A., 2008. *Aerospace Energy Conservation Utilising Optimum Methods*. Harbin, China, IEEE, pp. 1-6.
- Klanke, S. et al., 2006. *Dynamic Path Planning for a 7-DOF Robot Arm*. Beijing, China, IEEE, pp. 3879-3884.
- Koivo, A. J. & Guo, T. H., 1981. *Control of Robotic Manipulator With Adaptive Controller*. pp. 271-276.
- Korayem, M. H., Haghpanahi, M., Rahimi, H. N. & Nikoobin, A., 2009. *Finite Element Method and Optimal Control Theory for Path Planning of Elastic Manipulators*. s.l., Springer-Verlag Berlin Heidelberg, pp. 118-126.
- Koren, Y. & Borenstein, J., 2002. *Potential Field Methods and their Inherent Limitations for Mobile Robot Navigation*. pp. 1398-1404.
- Kostic, D., de Jager, B., Steinbuch, M. & Hensen, R., 2004. Modeling and Identification for High-Performance Robot Control: An RRR-Robotic Arm Case Study. *IEEE Transactions on Control System Technology*, November, 12(6), pp. 904-919.
- Kozlowski, K. & Pazderski, D., 2004. Modelling and Control of a 4-Wheel Skid-Steering Mobile Robot. *International Journal of Applied Mathematics and Computer Science*, 14(4), pp. 477-496.
- Kozlowski, K. & Pazderski, D., 2006. *Practical Stabilization of a Skid-Steering Mobile Robot - A Kinematic-Based Approach*. pp. 519-524.
- Kruskal, J. B., 1956. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society*, Feb, 7(1), pp. 48-50.
- Kubota, N., Arakawa, T., Fukuda, T. & Shimojima, K., 1997. *Trajectory generation for redundant manipulator using virus evolutionary genetic algorithm*. Albuquerque, NM, US, IEEE, pp. 205-210.
- Kuipers, B. & Byun, Y.-T., 1991. A robot exploration and mapping strategy based on semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8(1-2).

- Kunz, T., Reiser, U., Stilman, M. & Verl, A., 2010. *Real-Time Path Planning for a Robot Arm in Changing Environments*. Taipei, Taiwan, IEEE, pp. 5906-5911.
- Lahouar, S., Zeghloul, S. & Romdhane, L., 2005. *Path Planning for Manipulator Robots in Cluttered Environments*. American Society of Mechanical Engineers, s.n., pp. 633-639.
- Langville, A. N. & Meyer, C. D., 2006. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. :Princeton University Press.
- Lea, R. N., Hoblit, J. & Jani, Y., 1993. *Fuzzy Logic Based Robotic Arm Control*. pp. 128-133.
- Le, A. T., Rye, D. & Durrant-Whyte, H., 1997. *Estimation of Track-Soil Interactions for Autonomous Tracked Vehicles*. Albuquerque, New Mexico, pp. 1388-1393.
- Lee, C. S. G., 1982. Robot Arm Kinematics, Dynamics, and Control. *Computer*, December, 15(12), pp. 62-80.
- Lee, C. S. G. & Chung, M. J., 1982. *An Adaptive Control Strategy For Computer-Based Manipulators*. pp. 95-100.
- Leven, P. & Hutchinson, S., 2002. A Framework for Real-Time Path Planning in Changing Environments. *International Journal of Robotics Research*, Volume 21, pp. 999-1030.
- Levitt, T. S. & Lawton, D. T., 1990. Qualitative Navigation for Mobile Robots. *Artificial Intelligence*, 44(3).
- Lin, C.-C., Kuo, L.-W. & Chuang, J.-H., 2005. Potential-Based Path Planning for Robot Manipulators. *Journal of Robotic Systems*, 22(6), pp. 313-322.
- Lucet, E., Grand, C., Salle, D. & Bidaud, P., 2009. *Dynamic Yaw and Velocity Control of the 6WD Skid-Steering Mobile Robot RobuROC6 Using Sliding Mode Technique*. St. Louis, U.S.A., pp. 4220-4225.
- Lu, F. & Milios, E., 1997. Robot pose estimation in unknown environments by matching 2d range scans. 18(3), pp. 249-275.

Luh, J. Y. S., Walker, M. W. & Paul, R. P. C., 1980. Resolved-Acceleration Control of Mechanical Manipulators. *IEEE Transactions on Automatic Control*, June, AC-25(3), pp. 468-474.

Lundgren, J., 2010. *Matlab File Exchange*. [Online] Available at: <http://www.mathworks.co.uk/matlabcentral/fileexchange/28851-alpha-shapes/content/alphavol.m>

[Accessed 2012].

Lu, S. & Chung, J. H., 2005. Weighted Path Planning Based on Collision Detection. *Industrial Robot: An International Journal*, pp. 477-484.

Lyapunov, A. & Walker, J., 1994. The General Problem of the Stability of Motion (A.T. Fuller trans.). *Journal Of Applied Mathematics*, Volume 61, pp. 226-.

Macfarlane, S. & Croft, E. A., 2003. Jerk-bounded manipulator trajectory planning: design for real-time applications. *IEEE Transactions on Robotics and Automation*, February, 19(1), pp. 42-52.

Maclaurin, B., 2007. A Skid Steering Model With Track Pad Flexibility. *Journal of Terramechanics*, 44(1), pp. 95-110.

Maclaurin, B., 2008. Comparing the steering performances of skid-and Ackermann-steered vehicles. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 222(5), pp. 739-756.

Madow, A. et al., 2007. *Experimental Kinematics for Wheeled Skid-Steer Mobile Robots*. San Diego, California, pp. 1222-1227.

Martinez-Cantin, R. & Castellanos, J., 2005. *Unscented SLAM for Large-Scale Outdoor Environments*. pp. 3427-3432.

Martinez, J. et al., 2004. *Kinematic Modelling of Tracked Vehicles by Experimental Identification*. Sendai, Japan, pp. 1487-1492.

Martinez, J. et al., 2005. Approximating Kinematics for Tracked Mobile Robots. *International Journal of Robotics Research*, 24(10), pp. 867-868.

McKinnon, K. I., 1998. Convergence of the Nelder-Mead Simplex Method to a Nonstationary Point. *SIAM Journal on Optimisation*, 9(1), pp. 148-158.

- McLennan Servo Supplies, Ltd, 2014. *M452E Servo Motor Datasheet*.
- Menegatti, E., Maeda, T. & Ishiguro, H., 2004. Image-based Memory for Robot Navigation Using Properties of Omnidirectional Images. *Robotics and Autonomous Systems*, 47(4), pp. 251-267.
- Meriam, J. L. & Kraige, L. G., 2012. *Engineering Mechanics: dynamics (Vol. 2)*. 7th ed. s.l.:John Wiley & Sons.
- Merritt, H., 1946. The Evolution of a Tank Transmission. *ARCHIVE: Proceedings of the Institution of Mechanical Engineers 1847-1982 (vols 1-196)*, 154(1946), pp. 412-428.
- Mobus, R. & Kolbe, U., 2004. *Multi-Target Multi-Object Tracking, Sensor Fusion of Radar and Infrared*. pp. 732-737.
- Mohammad, T., 2009. Using Ultrasonic and Infrared Sensors for Distance Measurement. *World Academy of Science, Engineering and Technology*, Volume 51.
- Montemerlo, M. & Thrun, S., 2003. *Simultaneous Localization and Mapping with Unknown Data Association Using FastSLAM*. pp. 1985-1991.
- Montemerlo, M., Thrun, S., Koller, D. & Wegbreit, B., 2002. *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem*. pp. 593-598.
- Montemerlo, M., Thrun, S., Koller, D. & Wegbreit, B., 2003. *FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges*. pp. 1151-1156.
- Mooring, B. W. & Padavala, S. S., 1989. *The Effect of Kinematic Model Complexity on Manipulator Accuracy*. pp. 593-598.
- Moravec, H. & Elfes, A., 1985. *High Resolution Maps from Wide Angle Sonar*. pp. 116-121.
- Murphy, K. P., 2000. Bayesian Map Learning in Dynamic Environments. *Advances in Neural Information Processing Systems*, Volume 12, pp. 1015-1021.
- Murray, D. & Little, J. J., 2000. Using Real-Time Stereo Vision for Mobile Robot Navigation. *Autonomous Robots*, 8(2), pp. 161-171.

- Nahapetian, N., Jahed Motlagh, M. R. & Analoui, M., 2008. *PID Gain Tuning using Genetic Algorithms and Fuzzy Logic For Robot Manipulator Control*. pp. 346-350.
- Nakamura, T., 2013. *Real-Time 3-D Path Generation Method for a Robot Arm by a 2-D Dipole Field*. Wollongong, Australia, IEEE, pp. 745-749.
- Nebot, E. M., Durrant-Whyte, H. & Sheding, S., 1998. Frequency Domain Modelinh of Aided GPS for Vehicle Navigation Systems. *Robotics and Autonomous Systems*, 25(12), pp. 72-83.
- Nguyen, H. G. & Bott, J. P., 2000. *Robotics for law enforcement: Applications beyond explosive ordnance disposal*. Boston, SPIE.
- Niem, W. & Wingbermuehle, J., 1997. *Automatic Reconstruction of 3D Objects Using a Mobile Monoscopic Camera*. pp. 173-180.
- Nise, N. S., 1998. Time Response. In: *Control Systems Engineering*. s.l.:Wiley, pp. 178-179.
- Novotny, P. M. & Ferrier, N. J., 1999. *Using Infrared Sensors and the Phong Illumination Model to Measure Distances*. pp. 1644-1649.
- Ojeda, L. & Borenstein, J., 2006. *Non-GPS NAVigation for Emergency Responders*. pp. 12-15.
- Padula, F. & Perdereau, V., 2011. *A New Pseudoinverse for Manipulator Collision Avoidance*. Milan, Italy, pp. 14687-14692.
- Pakki, K., Chandra, B., Gu, D.-W. & Postlethwaite, I., 2010. *SLAM Using EKF, EH^∞ and Mixed EH_2/H^∞ Filter*. Yokohoma, Japan, IEEE, pp. 818-823.
- Panzieri, S., Pascucci, F. & Ulivi, G., 2002. An Outdoor Navigation System Using GPS and Inertial Platform. *IEEE/ASME Transactions on Mechatronics*, 7(2), pp. 134-142.
- Pazderski, D. & Kozlowski, K., 2008. *Trajectory Tracking of Underactuated Skid-Steering Robot*. Seattle, Washington, pp. 3506-3511.
- Petridis, V. & Zikos, N., 2010. *L-SLAM: reduced dimensionality FastSLAM algorithms*. pp. 1-7.

- Prim, R. C., 1957. Shortest connection networks and some generalizations. *Bell System Technical Journal*, Volume 36, pp. 1389-1491.
- Raimondi, F. M. & Ciancimino, L. S., 2008. *Intelligent Neruo-Fuzzy Dynamic Path Following for Car-Like Vehicle*. pp. 744-750.
- Ramos, F., Gajamohan, M., Heubel, N. & Ritter, H., 2013. *Time-Optimal Online Trajectory Generation for Robotic Manipulators*, Zürich: Institute for Dynamic Systems and Control.
- Rencken, W., 1993. *Concurrent Localisation and Map Building for Mobile Robots Using Ultrasonic Sensors*. pp. 2192-2197.
- Riaz, Z., Pervez, A., Ahmer, M. & Iqbal, J., 2010. *A Fully Autonomous Indoor Mobile Robot using SLAM*. pp. 1-6.
- Rosales, E. M. & Gan, Q., 2002. *Forward and Inverse Kinematics Models for a 5-dof Pioneer 2 Robot Arm*, Essex:
- Rosenfield, A. & Thurston, M., 2006. Edge and Curve Detection for Visual Scene Analysis. *IEEE Transactions on Computers*, 100(5), pp. 562-569.
- Rugh, W. J. & Shamma, J. S., 2000. Research on gain scheduling. *Automatica*, October, 36(10).
- Ryu, S. U., Kim, C. J. & Choi, K. H., 2007. *Multi-Arm Path Generation Method for Humanoid Robots*. Jeju, Korea, IEEE, pp. 224-227.
- Saridis, G. N. & Lee, C. G., 1979. An Approximation Theory of Optimal Control. *IEEE Transactions on Systems, Man and Cybernetics*, March, SMC-9(3), pp. 152-159.
- Schröter, C., Böhme, H.-J. & Gross, H.-M., 2007. *Memory-Efficient Gridmaps in Rao-Blackwellized Particle Filters for SLAM using Sonar Range Sensors*. pp. 138-143.
- Schröter, C. & Gross, H.-M., 2008. *A sensor-independent approach to RBPF SLAM - Map Match SLAM applied to Visual Mapping*. Nice, pp. 2078-2083.
- Shanmugavel, M., Tsourdos, A., White, B. A. & Zbikowski, R., 2007. Differential Geometric Path Planning of Multiple UAVs. *Journal of Dynamic Systems, Measurement and Control*, September, Volume 129, pp. 620-632.

- Shanmugavel, M., Tsourdos, A., White, B. A. & Zbikowski, R., 2009. Co-operative Path Planning of Multiple UAVs Using Dubins Paths with Clothoid Arcs. *Control Engineering Practice*, 18(9), pp. 1084-1092.
- Shen, X. & Deng, L., 1997. Game Theory Approach to Discrete H^∞ Filter Design. 49(4), pp. 1092-1095.
- Shiller, Z., Serate, W. & Hua, M., 1993. *Trajectory Planning of Tracked Vehicles*. pp. 796-801.
- Shimizu, M. & Okutomi, M., 2007. *Monocular Range Estimation through a Double-Sided Half-Mirror Plate*. pp. 347-354.
- Shuang, G. et al., 2007. *Skid Steering in 4-Wheel-Drive Electric Vehicle*. pp. 1548-1553.
- Simionescu, P. A., 2014. *Computer Aided Graphing and Simulation Tools*. 1st ed. Boca Raton: CRC Press.
- Singh, S. & Leu, M. C., 1987. Optimal Trajectory Generation for Robotic Manipulators Using Dynamic Programming. *Journal of Dynamic Systems, Measurement, and Control*, 1 June, 1-09(2), pp. 88-96.
- Soetanto, D., Lapiere, L. & Pascoal, A., 2003. *Adaptive, Non-Singular Path-Following Control of Dynamic Wheeled Robots*. Rome, Italy, pp. 1765-1770.
- Spero, D. J., 2004. *A Review of Outdoor Robotics Research*, Victoria, Australia: Monash University.
- Stasse, O., Dupitier, S. & Yokoi, K., 2006. *3D Object Recognition Using Spin-Images for a Humanoid Stereoscopic Vision System*. pp. 2955-2960.
- Steeds, W., 1950. Tracked vehicles - an analysis of the factors involved in steering. *Proceedings of the Institute of Automobile Engineers*, Volume 42, pp. 143-148.
- Tan, C.-W. & Park, S., 2005. Design of Accelerometer-Based Inertial Navigation Systems. *IEEE Transactions on Instrumentation and Measurement*, 54(6), pp. 2520-2530.

Tang, Z., Yang, M. & Pei, Z., 2010. *Self-Adaptive PID Control Strategy Based on RBF Neural Network for Robot Manipulator*. s.l., s.n., pp. 932-935.

Technical Avenue Sdn Bhd, 2016. *Technical Avenue UXM-30LN-PW Datasheet*. [Online]

Available at: <http://www.technical-avenue.com/products-search/type/viewtype/typeid/31/currentpage/2/lidar-laser-range-scanner.aspx>

[Accessed 02 Jan 2016].

Thrun, S., Burgard, W. & Fox, D., 1998. A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots. *Autonomous Robots*, Volume 5, pp. 253-271.

Tomono, M., 2010. *3D Localization Based on Visual Odometry and Landmark Recognition Using Image Edge Points*. pp. 5953-5959.

Turney, J. L., Mudge, T. N. & Lee, C. S. G., 1980. *Equivalence of Two Formulations For Robot Arm Dynamics*, Ann Arbor:

Velodyne LIDAR, 2016. *High Definition LIDAR HDL-64E S2*. [Online]

Available at: http://velodynelidar.com/lidar/products/brochure/HDL-64E%20S2%20datasheet_2010_lowres.pdf

[Accessed 01 Jan 2016].

Walther, H., 2012. *Ten applications of graph theory (Vol. 7)*. s.l.:Springer Science & Business Media.

Wang, Y., 2005. *Matlab File Exchange*. [Online]

Available at: <http://www.mathworks.co.uk/matlabcentral/fileexchange/7869-dijkstra-algorithm-consistent-with-cyclic-paths/content/matlab-dijkstra-cycle/dijkstra.m>

[Accessed 2011].

Wang, Z., Huang, S. & Dissanayake, G., 2007. D-SLAM: A Decoupled Solution to Simultaneous Localization and Mapping. *International Journal of Robotic Research*, Volume 26, pp. 187-204.

Wang, Z., Huang, S. & Dissanayale, G., 2007. *Multi-robot simultaneous localization and mapping using D-SLAM framework*. pp. 317-322.

- Wanh, J.-H. & Gao, Y., 2010. Land Vehicle Dynamics-Aided Inertial Navigation. *IEEE Transactions on Aerospace and Electronic Systems*, 46(4), pp. 1638-1653.
- Wei, W. & Shimin, W., 2010. *3-D Path Planning using Neural Networks for a Robot Manipulator*. Changsha, IEEE, pp. 3-6.
- White, B. A., Zbikowski, R. & Tsourdos, A., 2007. Direct Intercept Guidance using Differential Geometry Concepts. *IEEE Transactions on Aerospace and Electronic Systems*, July, 43(3), pp. 899-919.
- Whitney, D. E., 1969. Resolved Motion Rate Control of Manipulators and Human Prostheses. *IEEE Transactions on Man-Machine Systems*, Jun, MMS-10(2), pp. 47-53.
- Xin, M., Balakrishnan, S. N. & Huang, Z., 2001. *Robust State Dependant Riccati Equation Based Robot Manipulator Control*. Mexico City, Mexico, pp. 369-374.
- Yamada, K., Komada, S., Ishida, M. & Hori, T., 1998. *Robust Control of Robot Manipulators by MIMO Disturbance Observer*. pp. 1451-1456.
- Yamauchi, B., 1997. *A frontier-based approach for autonomous exploration*. pp. 146-151.
- Yamauchi, B., Schultz, A. & Adams, W., 1998. *Mobile robot exploration and map-building with continuous localization*. pp. 3715-3720.
- Yao, Z. & Gupta, K., 2007. Path Planning with General End-Effector Constraints. *Robotics and Autonomous Systems*, Volume 55, pp. 316-327.
- Zhang, Y., Zhang, Z. & Zhang, J., 2005. 3D Building Modelling with Digital Map, Lidar Data and Video Image Sequences. *The Photogrammetric Record*, 20(111), pp. 285-302.
- Zhang, Z., 1994. Iterative Point Matching for Registration of Free-Form Curves and Surfaces. 13(2), pp. 119-152.
- Zhao, Z.-Y., Tomizuka, M. & Isaka, S., 1992. *Fuzzy gain scheduling of PID controllers*. Dayton, Oh, US., IEEE, pp. 698-703 vol. 2.

Zha, X. F., 2002. Optimal Pose Trajectory Planning for Robot Manipulators. *Mechanism and Machine Theory*, Volume 37, pp. 1063-1086.

Zhuang, H., Roth, Z. S. & Hamano, F., 1992. A Complete and Parametrically Continuous Kinematic Model for Robot Manipulators. *IEEE Transactions on Robotics and Automation*, August, 8(4), pp. 451-463.

Zweiri, Y., Seneviratne, L. & Althoefer, K., 2003. Modelling and Control of an Unmanned Excavator Vehicle. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 217(4), pp. 259-274.

APPENDICES

Appendix A Genetic Algorithm Supplementary Information and Results

In this appendix the full list of optimisation problems used in Chapter 5 is presented in Table A-1 and Table A-2. Figure A-1 and Figure A-2 present the results of the optimisation technique shown in Chapter 5 in an alternate way, with lines connecting each result in order to assess trends. Figure A-3 to Figure A-25 display further results from the random waypoint path testing used to validate the GA optimised PID controller which was also carried out in Chapter 5.

Table A-1 Full list of optimisation problems used to validate the developed GA.

Function Name	Function	Search Domain
Ackley's function	$f(x, y) = -20e^{-0.2\sqrt{0.5(x^2+y^2)}} - e^{\cos(2\pi x)+\cos(2\pi y)} + 20 + e$	$-5 \leq x, y \leq 5$
Sphere function	$f(x) = \sum_{i=1}^n x_i^2$	$-\infty \leq x_i \leq \infty,$ $1 \leq i \leq n$
2nd Order Polynomial	$f(x) = 3x^2 + 2x + 1$	$-\infty \leq x \leq \infty$
3 variable quadratic	$f(x, y, z) = \sqrt{x^2 + y^2 + z^2 + xy + yz + xz}$	$-\infty \leq x, y, z \leq \infty$
Rosenbrock function	$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$-\infty \leq x_i \leq \infty,$ $1 \leq i \leq n$

Beale's function	$f(x, y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$	$-4.5 \leq x, y \leq 4.5$
Goldstein-Price function	$f(x, y) = (1 + (x + y + 1)^2(19 - 14x + 3x^2 - 14y + 6xy + 3y^2))(30 + (2x - 3y)^2(18 - 32x + 12x^2 + 48y - 36xy + 27y^2))$	$-2 \leq x, y \leq 2$
Booth's Function	$f(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2$	$-10 \leq x, y \leq 10$
Bukin Function	$f(x, y) = 100\sqrt{ y - 0.01x^2 } + 0.01 x + 10 $	$-15 \leq x \leq -5,$ $-3 \leq y \leq 3$
Matyas Function	$f(x, y) = 0.26(x^2 + y^2) - 0.48xy$	$-10 \leq x, y \leq 10$
Levi Function No. 13	$f(x, y) = \sin^2(3\pi x) + (x - 1)^2(1 + \sin^2(3\pi y)) + (y - 1)^2(1 + \sin^2(2\pi y))$	$-10 \leq x, y \leq 10$
Three-hump Camel Function	$f(x, y) = 2x^2 - 1.05x^4 + \frac{x^6}{6} + xy + y^2$	$-5 \leq x, y \leq 5$
Easom Function	$f(x, y) = -\cos(x) \cos(y) e^{-((x-\pi)^2 + (y-\pi)^2)}$	$-100 \leq x, y \leq 100$
Cross-in-tray Function	$f(x, y) = -0.0001 \left(\sin(x) \sin(y) e^{\left 100 - \frac{\sqrt{x^2 + y^2}}{\pi} \right } + 1 \right)^{0.1}$	$-10 \leq x, y \leq 10$

Eggholder Function	$f(x, y) = -(y + 47) \sin\left(\sqrt{\left y + \frac{x}{2} + 47\right }\right) - x \sin\left(\sqrt{ x - (y + 47) }\right)$	$-512 \leq x, y \leq 512$
Hölder table Function	$f(x, y) = -\left \sin(x) \cos(y) e^{\left 1 - \frac{\sqrt{x^2 + y^2}}{\pi}\right }\right $	$-10 \leq x, y \leq 10$
McCormick Function	$f(x, y) = \sin(x + y) + (x - y)^2 - 1.5x + 2.5y + 1$	$-1.5 \leq x \leq 4, -3 \leq y \leq 4$
Schaffer No.2 Function	$f(x, y) = 0.5 + \frac{\sin^2(x^2 - y^2) - 0.5}{(1 + 0.001(x^2 + y^2))^2}$	$-100 \leq x, y \leq 100$
Schaffer No.4 Function	$f(x, y) = 0.5 + \frac{\cos^2(\sin x^2 - y^2) - 0.5}{(1 + 0.001(x^2 + y^2))^2}$	$-100 \leq x, y \leq 100$
Styblinski- Tang Function	$f(x) = \frac{\sum_{i=1}^n x_i^4 - 16x_i^2 + 5x_i}{2}$	$-5 \leq x_i \leq 5, 1 \leq i \leq n$
Simionescu Function	$f(x, y) = 0.1xy$	$-1.25 \leq x, y \leq 1.25$
	$\text{Subject to: } x^2 + y^2 \leq \left(1 + 0.2 \left(\cos 8 \arctan \frac{x}{y}\right)\right)^2$	
Binh and Korn Function	$\min \begin{cases} f_1(x, y) = 4x^2 + 4y^2 \\ f_2(x, y) = (x - 5)^2 + (y - 5)^2 \end{cases}$	$0 \leq x \leq 5, 0 \leq y \leq 3$
Chakong and Haimes Function	$\min \begin{cases} f_1(x, y) = 2 + (x - 2)^2 + (y - 1)^2 \\ f_2(x, y) = 9x - (y - 1)^2 \end{cases}$	$-20 \leq x, y \leq 20$

Fonseca
and
Fleming
Function

$$\min \begin{cases} f_1(x) = 1 - e^{-\sum_{i=1}^n (x_i - \frac{1}{\sqrt{n}})^2} \\ f_2(x) = 1 - e^{-\sum_{i=1}^n (x_i + \frac{1}{\sqrt{n}})^2} \end{cases} \quad \begin{array}{l} -4 \leq x_i \leq 4, \\ 1 \leq i \leq n \end{array}$$

Test
Function 4

$$\min \begin{cases} f_1(x, y) = x^2 - y \\ f_2(x, y) = -0.5x - y - 1 \end{cases} \quad -7 \leq x, y \leq 4$$

Kursawe
Function

$$\min \begin{cases} f_1(x) = \sum_{i=1}^2 -10e^{-0.2\sqrt{x_i^2 + x_{i+1}^2}} \\ f_2(x) = \sum_{i=1}^3 |x_i|^{0.8} + 5 \sin(x_i^3) \end{cases} \quad \begin{array}{l} -5 \leq x_i \leq 5, \\ 1 \leq i \leq 3 \end{array}$$

Schaffer
Function
No. 1

$$\min = \begin{cases} f_1(x, y) = x^2 \\ f_2(x, y) = (x - 2)^2 \end{cases} \quad -A \leq x \leq A$$

Values of A from 10 to 10^5 have been used successfully. Higher values of A increase the difficulty of the problem.

Schaffer
Function
No. 2

$$\min \begin{cases} f_1(x) = \begin{cases} -x, & \text{if } x \leq 1 \\ x - 2, & \text{if } 1 < x \leq 3 \\ 4 - x, & \text{if } 3 < x \leq 4 \\ x - 4, & \text{if } x > 4 \end{cases} \\ f_2(x) = (x - 5)^2 \end{cases} \quad -5 \leq x \leq 10$$

Poloni's
Two
Objective
Function

$$\min \begin{cases} f_1(x, y) = 1 + (A_1 - B_1(x, y))^2 + (A_2 - B_2(x, y))^2 \\ f_2(x, y) = (x + 3)^2 + (y + 1)^2 \end{cases} \quad -\pi \leq x, y \leq \pi$$

where,

$$\begin{cases} A_1 = 0.5 \sin(1) - 2 \cos(1) + \sin(2) - 1.5 \cos(2) \\ A_2 = 1.5 \sin(1) - \cos(1) + 2 \sin(2) - 0.5 \cos(2) \\ B_1(x, y) = 0.5 \sin x - 2 \cos x + \sin y - 1.5 \cos y \\ B_2(x, y) = 1.5 \sin x - \cos x + 2 \sin y - 0.5 \cos y \end{cases}$$

Zitzler-Deb-
Thiele's
Function
No.1

$$\min \begin{cases} f_1(x) = x_1 & 0 \leq x_i \leq 1, \\ f_2(x) = g(x)h(f_1(x), g(x)) & \\ g(x) = 1 + \frac{9}{29} \sum_{i=2}^{30} x_i & 1 \leq i \leq 30 \\ h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} \end{cases}$$

Zitzler-Deb-
Thiele's
Function
No.2

$$\min \begin{cases} f_1(x) = x_1 & 0 \leq x_i \leq 1, \\ f_2(x) = g(x)h(f_1(x), g(x)) & \\ g(x) = 1 + \frac{9}{29} \sum_{i=2}^{30} x_i & 1 \leq i \leq 30 \\ h(f_1(x), g(x)) = 1 - \left(\frac{f_1(x)}{g(x)}\right)^2 \end{cases}$$

Zitzler-Deb-
Thiele's
Function
No.3

$$\min \begin{cases} f_1(x) = x_1 & 0 \leq x_i \leq 1, \\ f_2(x) = g(x)h(f_1(x), g(x)) & \\ g(x) = 1 + \frac{9}{29} \sum_{i=2}^{30} x_i & 1 \leq i \leq 30 \\ h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} - \left(\frac{f_1(x)}{g(x)}\right) \sin(10\pi) \end{cases}$$

Zitzler-Deb-
Thiele's
Function
No.4

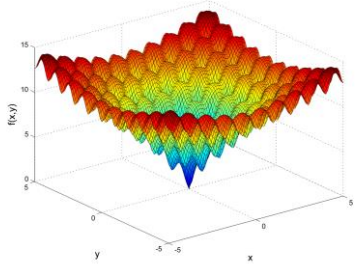
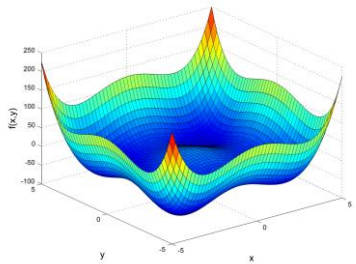
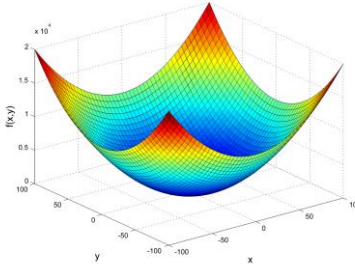
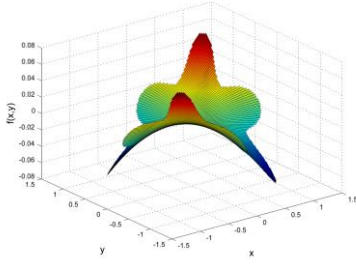
$$\min \begin{cases} f_1(x) = x_1 & 0 \leq x_1 \leq 1, \\ f_2(x) = g(x)h(f_1(x), g(x)) & \\ g(x) = 91 + \sum_{i=2}^{10} (x_i^2 - 10 \cos 4\pi x_i) & -5 \leq x_i \leq 5, \\ h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} & 2 \leq i \leq 10 \end{cases}$$

Zitzler-Deb-
Thiele's
Function
No.6

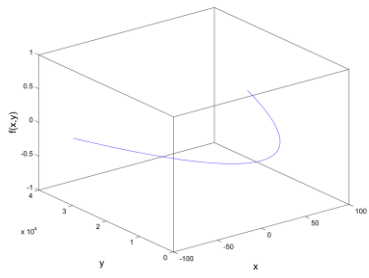
$$\min \begin{cases} f_1(x) = 1 - e^{-4x_1} \sin^6(6\pi x_1) & 0 \leq x_i \leq 1, \\ f_2(x) = g(x)h(f_1(x), g(x)) & \\ g(x) = 1 + 9 \left(\frac{\sum_{i=2}^{10} x_i}{9}\right)^{0.25} & 1 \leq i \leq 10 \\ h(f_1(x), g(x)) = 1 - \left(\frac{f_1(x)}{g(x)}\right)^2 \end{cases}$$

Viennet Function	$\min \begin{cases} f_1(x, y) = 0,5(x^2 + y^2) + \sin(x^2 + y^2) \\ f_2(x, y) = \frac{(3x + 2y + 4)^2}{8} + \frac{(x - y + 1)^2}{27} + 15 \\ f_3(x, y) = \frac{1}{x^2 + y^2 + 1} - 1.1e^{-(x^2 + y^2)} \end{cases}$	$-3 \leq x, y \leq 3$
Osyczka and Kundu Function	$\min \begin{cases} f_1(x) = -25(x_1 - 2)^2 - (x_2 - 2)^2 - (x_3 - 1)^2 \\ \quad - (x_4 - 4)^2 - (x_5 - 1)^2 \\ f_2(x) = \sum_{i=1}^6 x_i^2 \end{cases}$	$0 \leq x_1, x_2, x_6 \leq 10,$ $1 \leq x_3, x_5 \leq 5,$ $0 \leq x_4 \leq 6$
CTP1 Function	$\min \begin{cases} f_1(x, y) = x \\ f_2(x, y) = (1 + y)e^{-\frac{x}{1+y}} \end{cases}$	$0 \leq x, y \leq 1$
Constr-Ex Function	$\min \begin{cases} f_1(x, y) = x \\ f_2(x, y) = \frac{1 + y}{x} \end{cases}$	$0.1 \leq x \leq 1,$ $0 \leq y \leq 5$

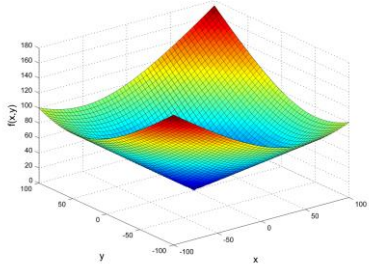
Table A-2 Graphical representation of all tested optimisation problems.

Function Name	Function Surface	Function Name	Function Surface
Ackley's function		Styblinski-Tang Function	
Sphere function		Simionescu Function	

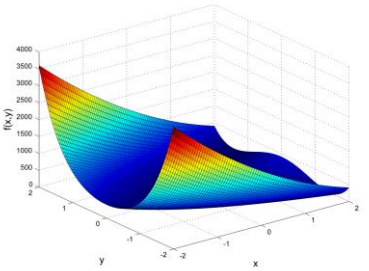
2nd Order Polynomial



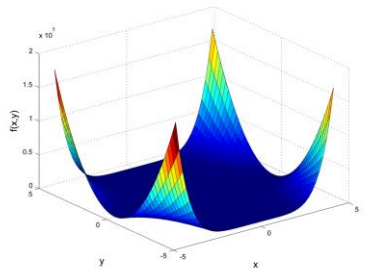
3 variable quadratic



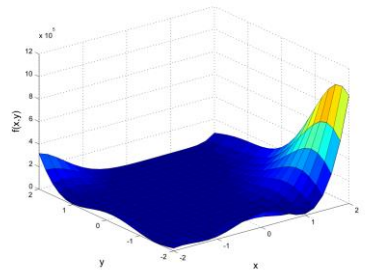
Rosenbrock function



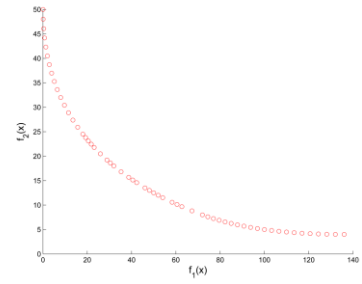
Beale's function



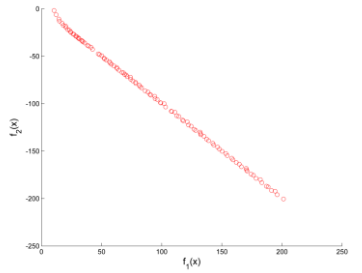
Goldstein n-Price function



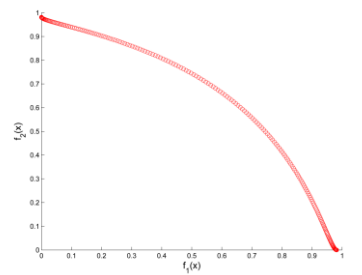
Binh and Korn Function



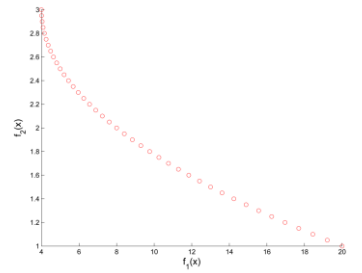
Chakong and Haines Function



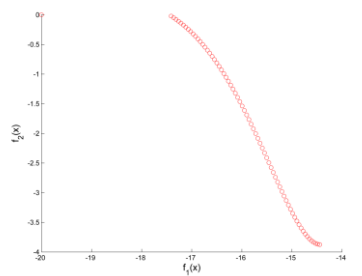
Fonseca and Fleming Function



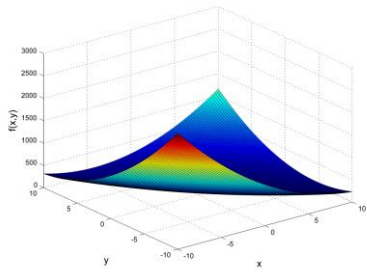
Test Function 4



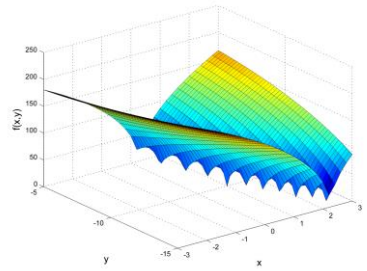
Kursawe Function



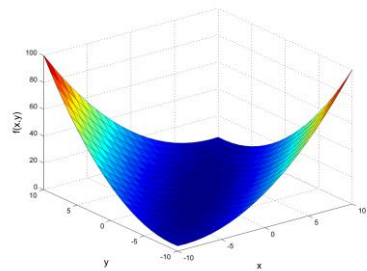
**Booth's
Function**



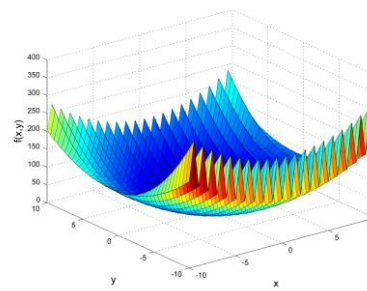
**Bukin
Function**



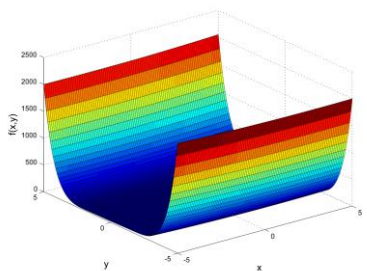
**Matyas
Function**



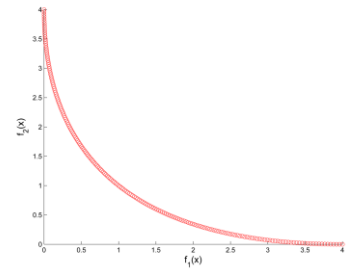
**Levi
Function
No. 13**



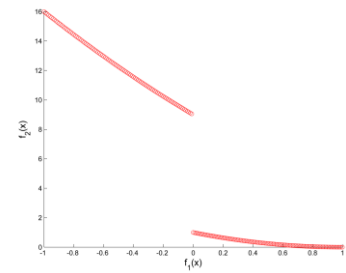
**Three-
hump
Camel
Function**



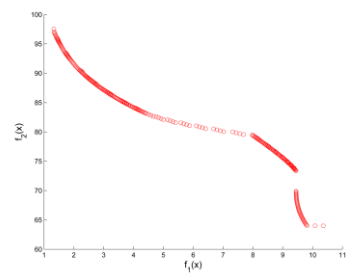
**Schaffer
Function No.
1**



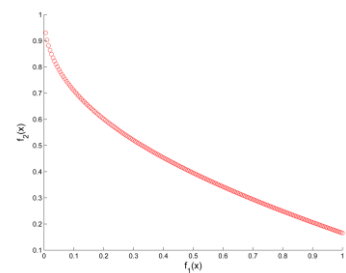
**Schaffer
Function No.
2**



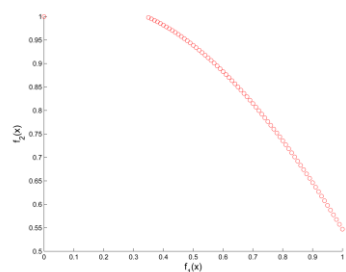
**Poloni's Two
Objective
Function**



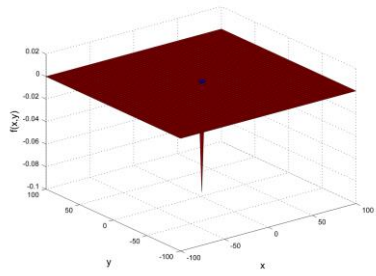
**Zitzler-Deb-
Thiele's
Function
No.1**



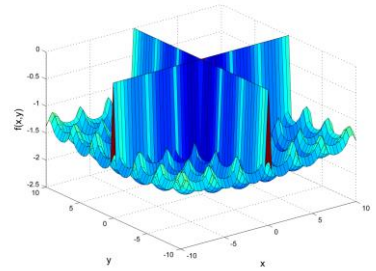
**Zitzler-Deb-
Thiele's
Function
No.2**



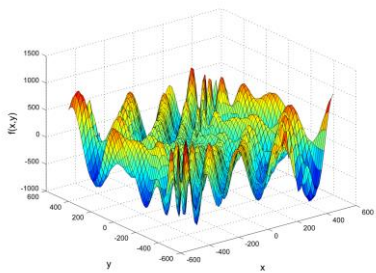
**Easom
Function**



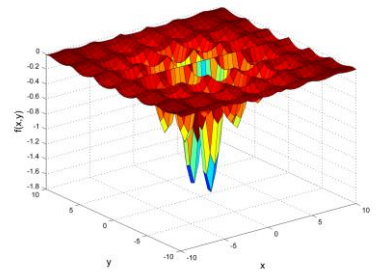
**Cross-in-
tray
Function**



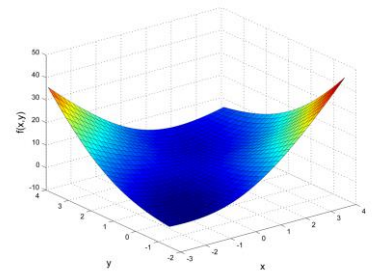
**Eggholde
r
Function**



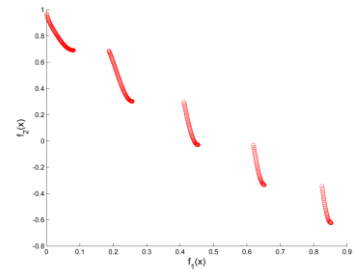
**Hölder
table
Function**



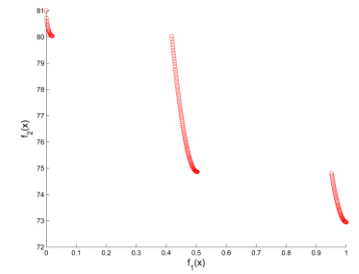
**McCormic
k
Function**



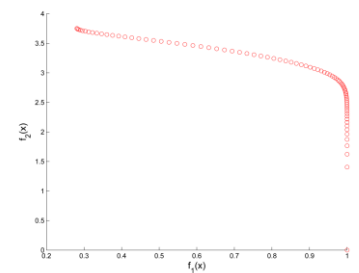
**Zitzler-Deb-
Thiele's
Function
No.3**



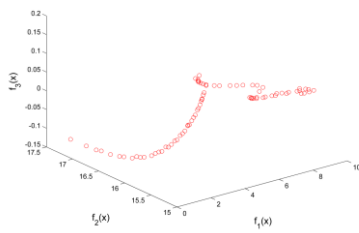
**Zitzler-Deb-
Thiele's
Function
No.4**



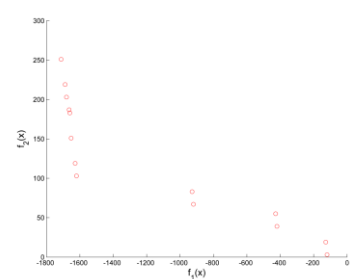
**Zitzler-Deb-
Thiele's
Function
No.6**



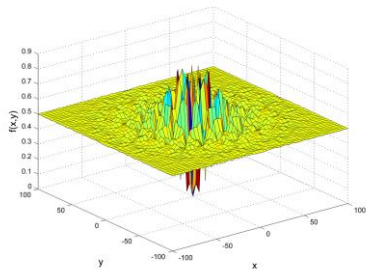
**Viennet
Function**



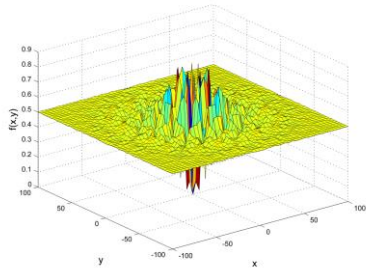
**Osyczka and
Kundu
Function**



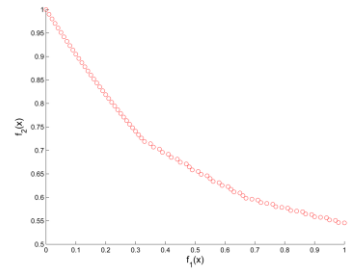
Schaffer
No.2
Function



Schaffer
No.4
Function



CTP1
Function



Constr-Ex
Function

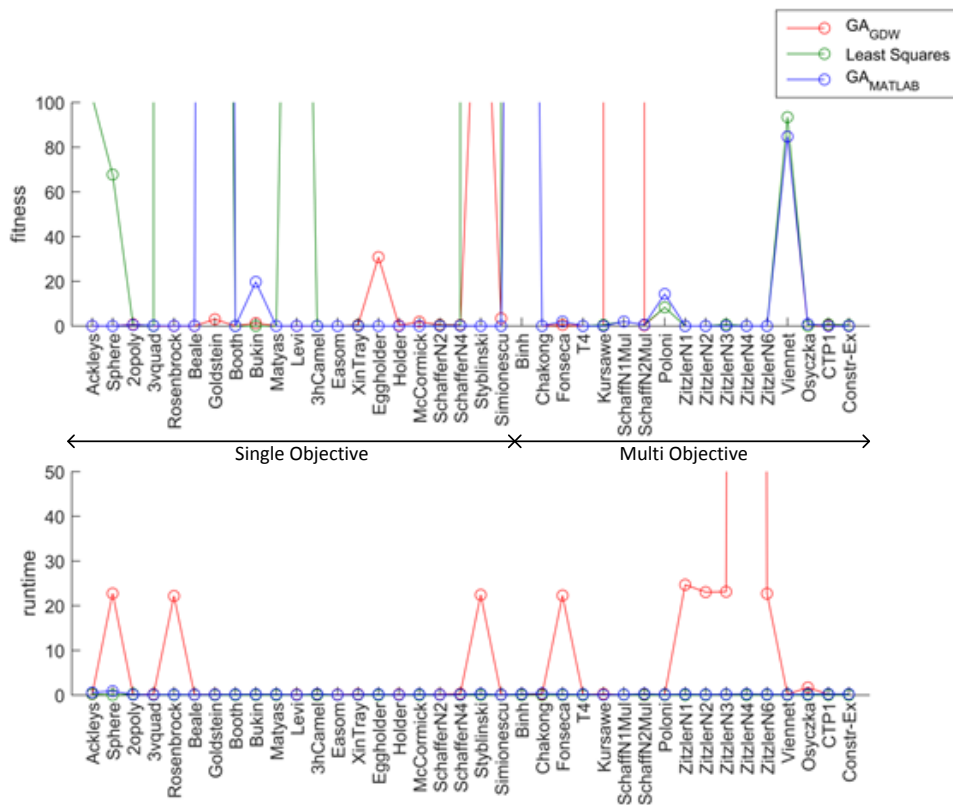
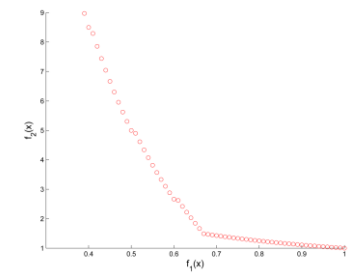


Figure A-1 Alternate representation of optimisation technique comparison (linear).

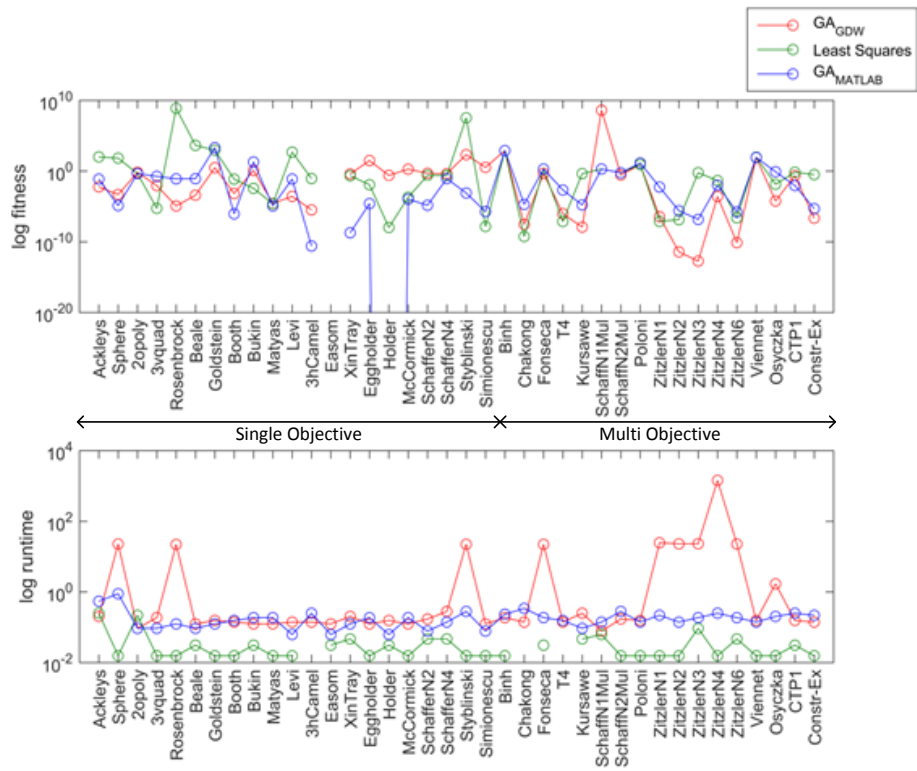


Figure A-2 Alternate representation of optimisation technique comparison (logarithmic).

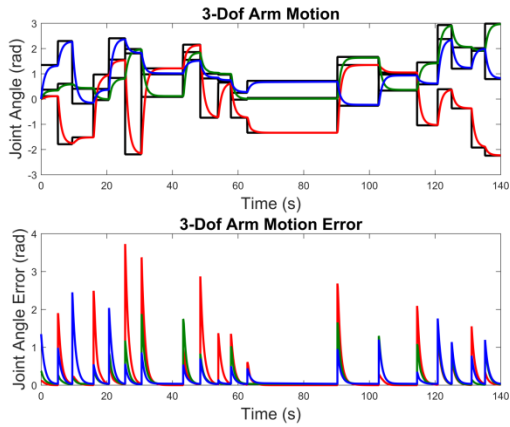


Figure A-3 Further PID controller testing results.

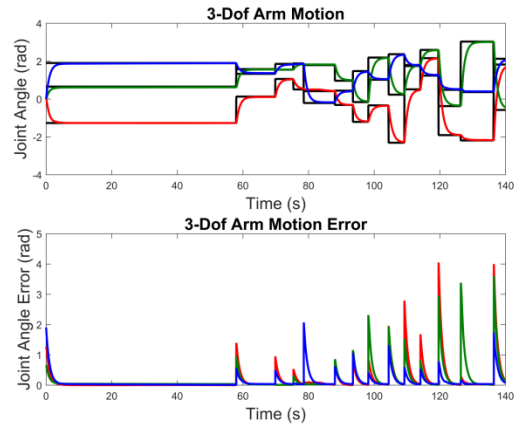


Figure A-4 Further PID controller testing results.

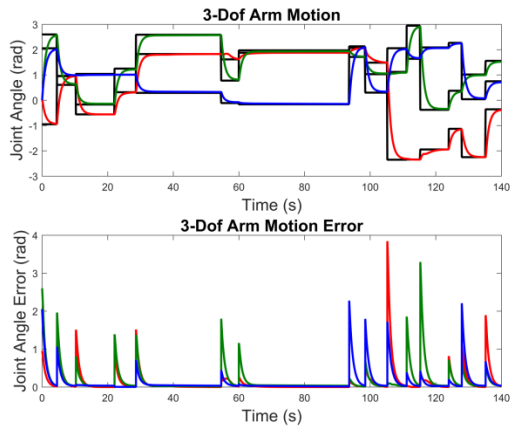


Figure A-5 Further PID controller testing results.

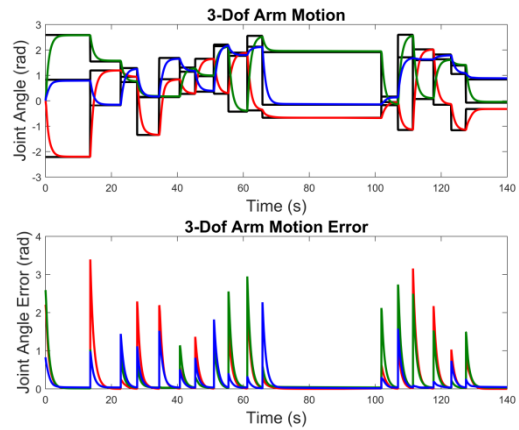


Figure A-6 Further PID controller testing results.

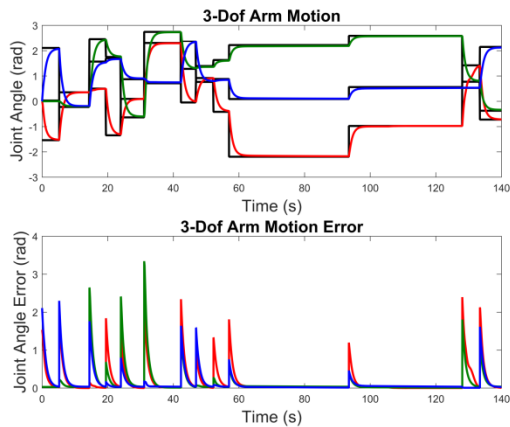


Figure A-7 Further PID controller testing results.

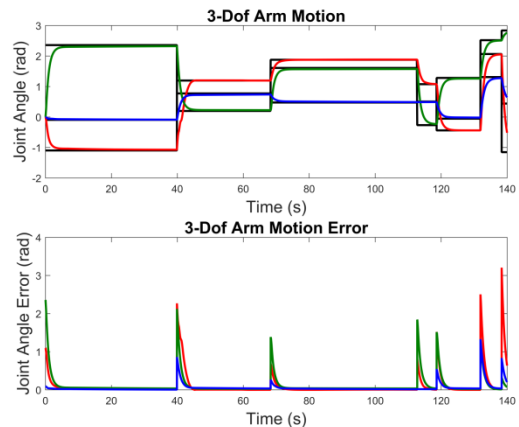


Figure A-8 Further PID controller testing results.

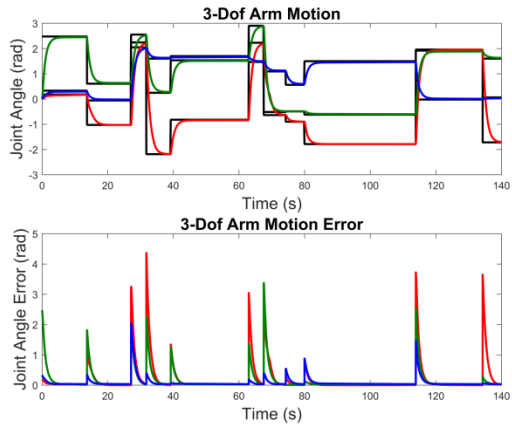


Figure A-9 Further PID controller testing results.

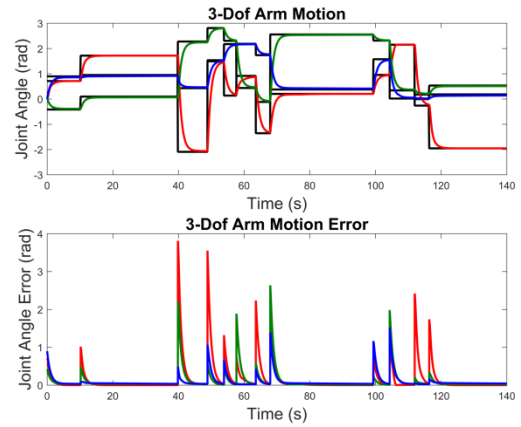


Figure A-10 Further PID controller testing results.

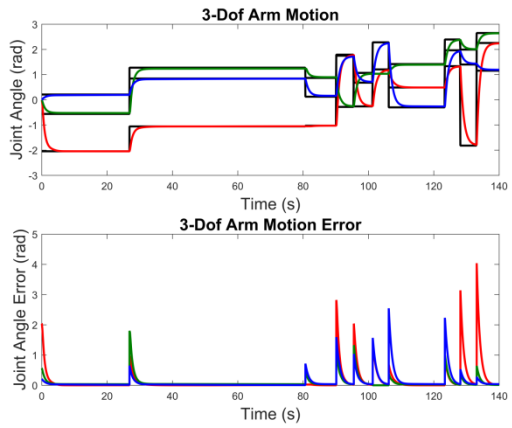


Figure A-11 Further PID controller testing results.

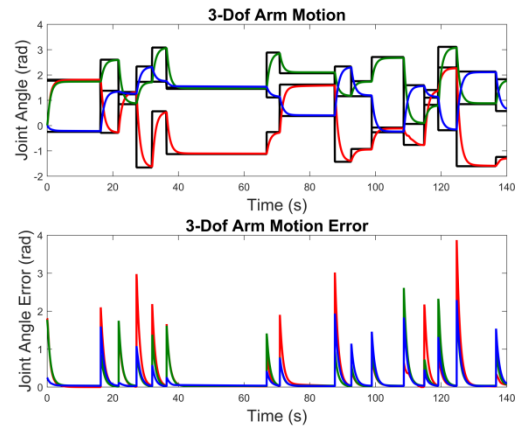


Figure A-12 Further PID controller testing results.

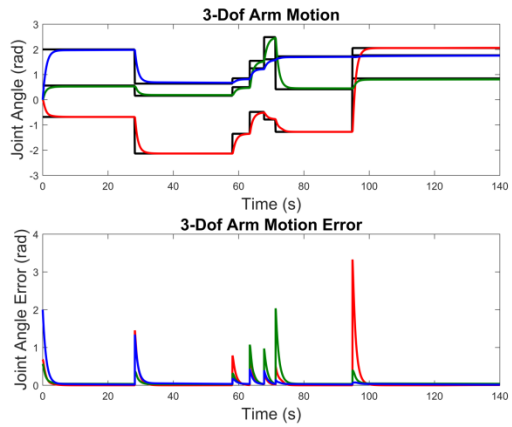


Figure A-13 Further PID controller testing results.

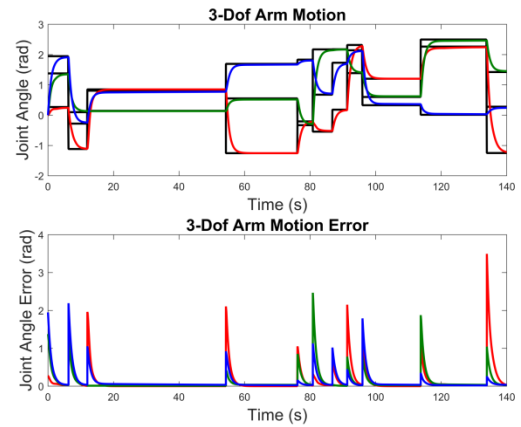


Figure A-14 Further PID controller testing results.

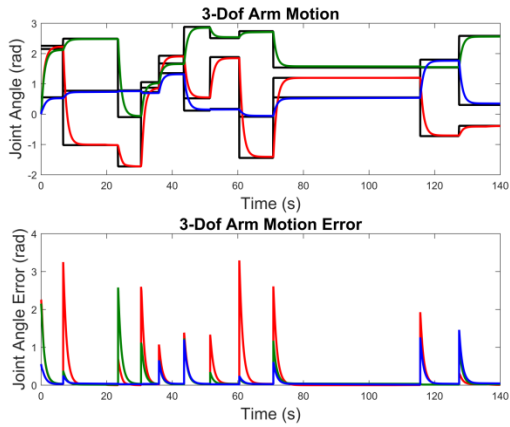


Figure A-15 Further PID controller testing results.

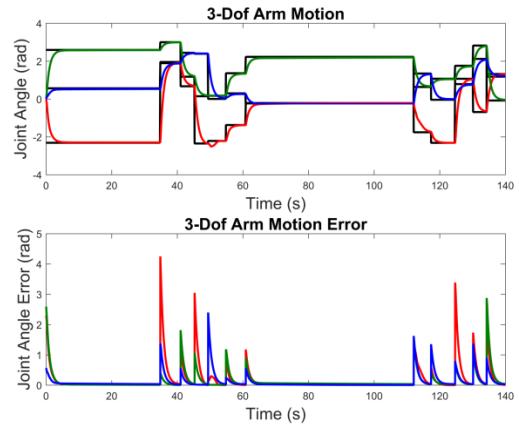


Figure A-16 Further PID controller testing results.

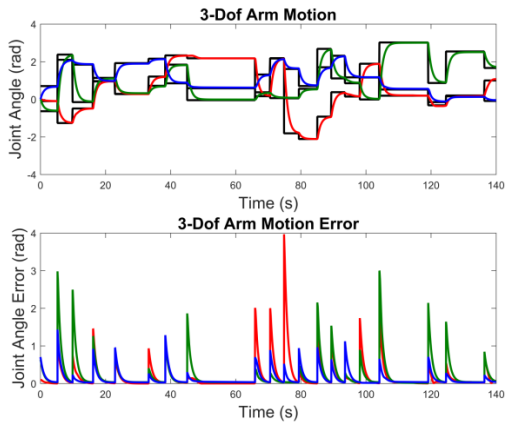


Figure A-17 Further PID controller testing results.

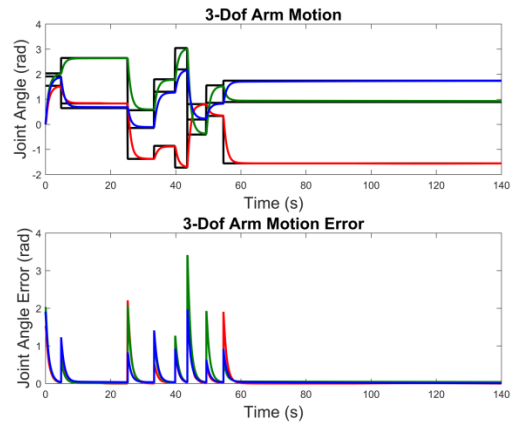


Figure A-18 Further PID controller testing results.

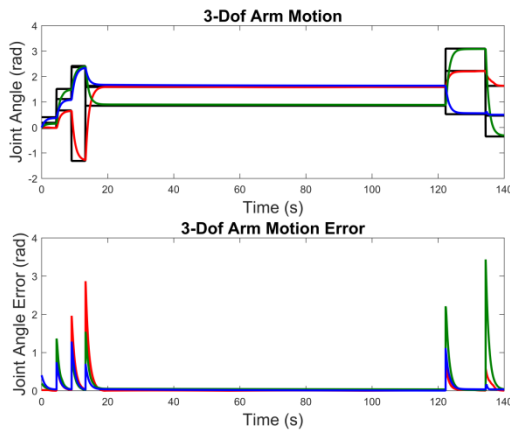


Figure A-19 Further PID controller testing results.

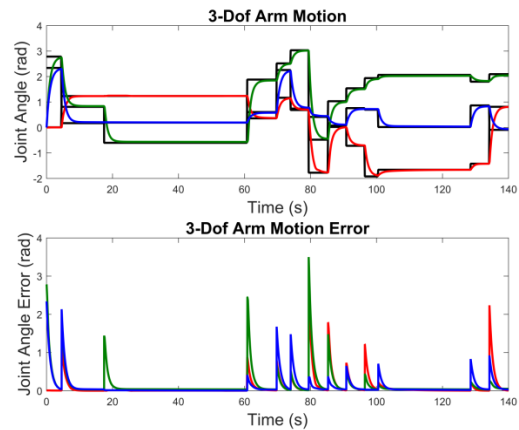


Figure A-20 Further PID controller testing results.

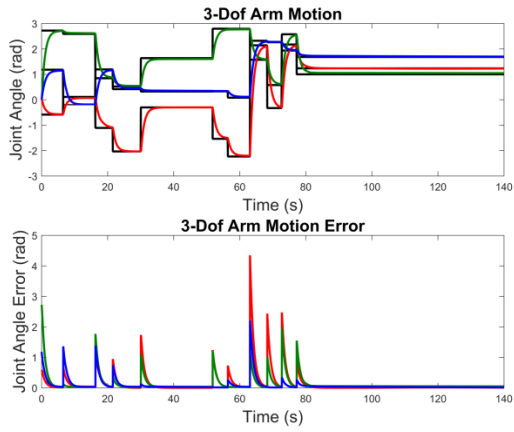


Figure A-21 Further PID controller testing results.

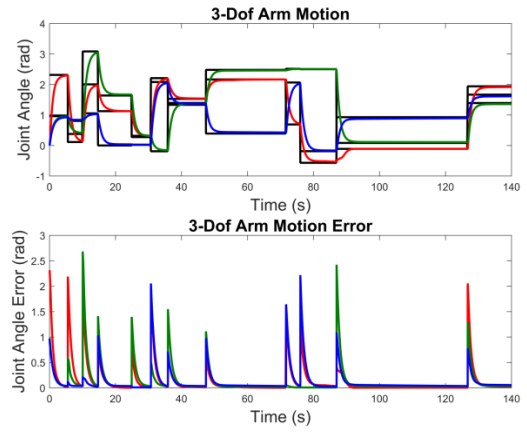


Figure A-22 Further PID controller testing results.

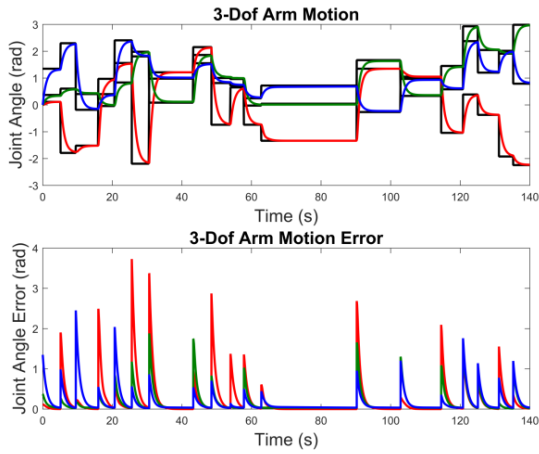


Figure A-23 Further PID controller testing results.

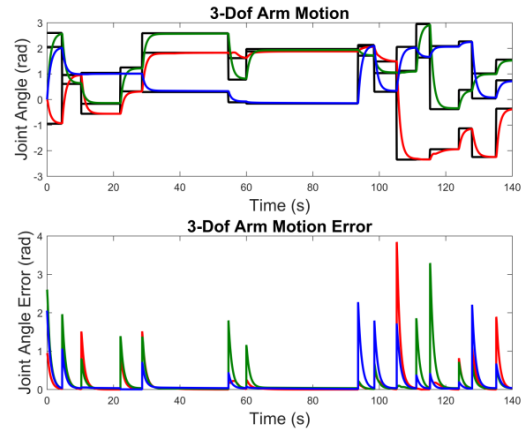


Figure A-24 Further PID controller testing results.

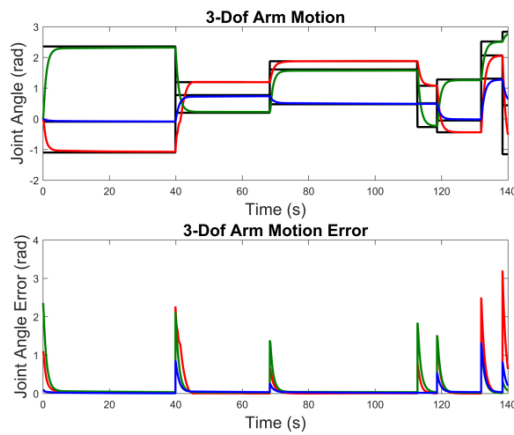


Figure A-25 Further PID controller testing results.

Appendix B Extension of a 3-DoF Path Planning Algorithm to 9-DoF

Having validated the effectiveness of the algorithm for a 3-DoF manipulator arm, initial simulations can be carried out to test the potential of the method to handle path planning for manipulator arms of higher degrees of freedom. Since the arm in use for this project is a 3-DoF arm, no inverse kinematic model has been developed in order to convert environment data into c-space. Instead, simulated c-space data is used by generating clusters of random n-dimensional numbers in an n-dimensional space of varying size. This method of simulation allows for the number of degrees-of-freedom in the c-space, the number of objects used, the number of inspection points in the object, and the Euclidean maximum spread of each object, which gives the ability to measure calculation time for different combinations of these points.

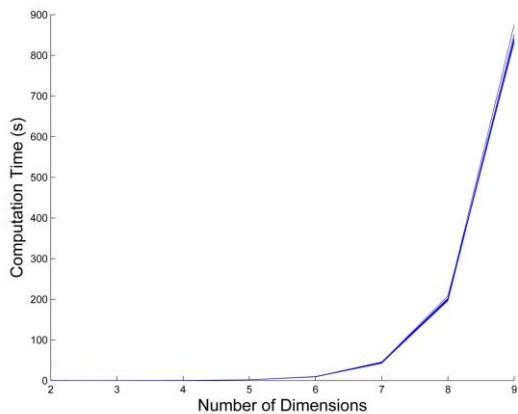
This version of the algorithm utilises, in the majority, previous development carried out in this project. The method of forming a node map from a triangulation of a shape, and the planning of the path through it for the manipulator by the use of Dijkstra's Algorithm is identical to that previously used. An algorithm for generating the alpha hull of a cluster of points in n-dimensions has not been developed, but Matlab's 'convhulln' function, which can calculate the convex hull of a cloud of points in up to 9-dimensions has been used instead. This means that the functionality of the n-dimensional algorithm thus far can only work with its maximum effectiveness when the cluster of points in the objects have no concavity, but future work will investigate the tunnelling process into a convex hull in n-dimensions in order to increase the ability of the algorithm to be able to handle concave objects.

Experimentation in this case will take the form of a series of Matlab simulated runs of the algorithm for a set number of obstacles in a space with a fixed length in each of the n-dimensions. The number of inspection points will be fixed, and the maximum spread, hence the size of each shape, will also be fixed. This experiment will deal with the effect of increasing the number of dimensions on the calculation time required to generate the node map and plan a path in order to determine if the algorithm can be used to generate a path through n-dimensional c-space. For this investigation, the following specifications are used:

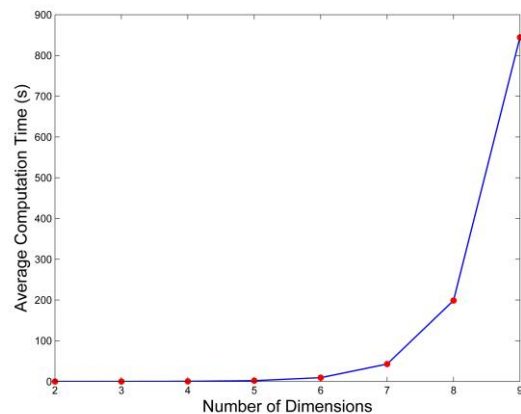
Table B-3 Parameters for the simulated C-space generation for use in the n-DoF path planning experimentation.

Number of objects	100
Number of inspection points per object	100
Spread of points per object per dimension	3 m
Range of each dimension	100 m

This experiment is run ten times and then the mean of the time taken per run is used to investigate the usability of the algorithm for each number of dimensions, i.e. degrees-of-freedom. The results are displayed in the following figures.



(a) Multiple runs of path planner.



(b) Average of multiple path planner runs.

Figure B-26 Results from a number of runs of the path planning algorithm for 2 to 9 degrees-of-freedom.

From these results it can be seen that it is possible to plan a path through a 9-dimensional space, therefore for a 9-DoF manipulator arm. The time it takes to plan a path increases exponentially with the number of degrees-of-freedom, but for 2 to 6

degrees-of-freedom, the time taken to calculate a path is on the order of seconds. This can be seen in Figure B-27.

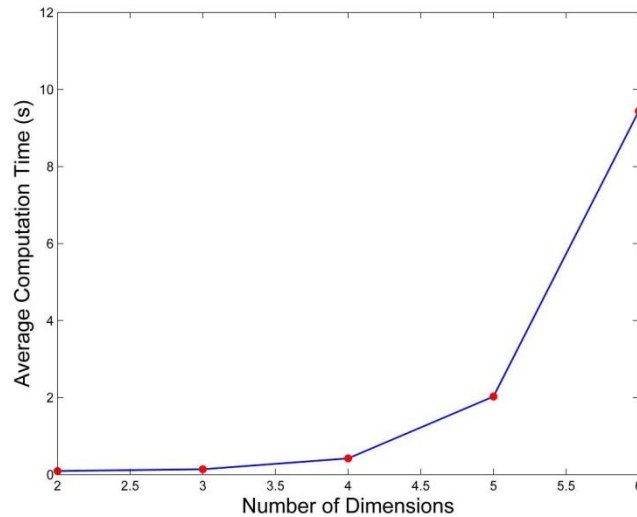


Figure B-27 Enlarged view of Figure B-26 to display only the results for path planning in 2 to 6 degrees-of-freedom.

Based on these results it is feasible to use this algorithm to generate a path in 6D space, so for 6-DoF. There are currently some limitations to this version of the path planning algorithm in that the inverse kinematic solution to the arm in question still has to be developed in order to convert environment data to the C-space. Also, since the method of converting clouds of points in C-space in greater than 3D uses the convex hull rather than an alpha hull with concavity, the method cannot currently work for concave shapes in the C-space.

Appendix C List of Publications

C.1 Published Works

D. Galvão Wall, J. Economou, H. Goyder, K. Knowles, P. Silson, M. Lawrance, (2015) *Mobile Robot Arm Trajectory Generation for Operation in Confined*

Environments, Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering. (vol. 229, part. 3, p215-234 DOI 10.1177/0950651814559760)

C.2 Submitted for Publication

D. Galvão Wall, J. T. Economou, D. Purdy, K. Knowles, *Genetic Algorithm Gain Scheduler Tuning for Quasi-Linear Parameter Variant Robotic Manipulator Dynamic Model*, European Control Conference (ECC 2016), Aalborg, Denmark.

J. T. Economou, D. Galvão Wall, K. Knowles, *Novel System Energy and Performance Analysis for Classical Inverted Pendulum Control Problem with Guaranteed Stability*, European Control Conference (ECC 2016), Aalborg, Denmark.

C.3 Submitted Abstracts

D. Purdy, D. Simner, D. Diskett, J. Economou, D. Wall, *Yaw Stability Controller for Tracked Vehicles*, International Symposium on Advanced Vehicle Control (AVEC 2016), Munich, Germany.

J. Economou, D. Purdy, D. Galvão Wall, D. Diskett, D. Simner, *Intelligent Based Terrain Preview Controller for 3-axle Vehicle*, International Symposium on Advanced Vehicle Control (AVEC 2016), Munich, Germany.

C.4 Under Preparation

D. Galvão Wall, J. Economou, K. Knowles, *GA-based controller tuning for a quasi-linear parameter varying manipulator arm* (2016) IMechE .Journal Article

D. Galvão Wall, J. Economou, K. Knowles, *A graph-theory-based C-space path planner for mobile robotic manipulators in close-proximity environments*. (2016)
IMechE .Journal Article

