UNIVERSITY OF SOUTHAMPTON

FACULTY OF ENGINEERING AND THE ENVIRONMENT

Experimentally Verified Model Predictive Control of a Hover-Capable AUV

Leo Vincent Steenson

Thesis submitted for the degree of Doctor of Philosophy

June 2013

UNIVERSITY OF SOUTHAMPTON
ABSTRACT
FACULTY OF ENGINEERING AND THE ENVIRONMENT
Fluid Structure Interactions Research Group
Doctor of Philosophy
'Experimentally Verified Model Predictive Control of a Hover-Capable AUV' by Leo Vincent Steenson

This work presents the development of control systems that enable a hover-capable AUV to operate throughout a wide speed range. The Delphin2 AUV was built as part of this project and is used to experimentally verify the prototype control systems. This vehicle is over-actuated with; four through-body tunnel thrusters, four independently-actuated control surfaces and a rear propeller. The large actuator set allows the Delphin2 to operate at low speeds, using the through-body tunnel thrusters, and at high speeds, using the rear propeller and control surfaces.

There lies a region between slow and high speed where neither the control surfaces nor tunnel thrusters are operating optimally. To maintain depth stability, both actuator sets are required to operate simultaneously. The model predictive control (MPC) algorithm is used to control the vehicle given its ability to handle multiple inputs and outputs along with system uncertainties.

The basis of MPC is a mathematical model of the system to be controlled. Several experiments were conducted with the Delphin2 AUV to acquire the data necessary to develop this model. Bollard pull tests were used to measure thruster performance whilst wind-tunnel and open water experiments provided a measure of the control surfaces, hull and propeller performance.

Depth control is the primary focus of this Thesis, however, pitch and surge control are also addressed. Three controllers are developed in this work, of increasing complexity; a depth and pitch controller for low speed operations, a depth and surge velocity controller for medium to high speed operation, and finally, a depth and surge velocity controller for operation from low to high speed operations. All three controllers are multi-input multi-output (MIMO) and use the MPC algorithm. Input constraints are imposed on both the absolute limits and the rate of change limits. Simulations are performed to aid in the design of each controller before it is implemented on the Delphin2 AUV and experimentally verified.

The depth and pitch controller, developed for low speed operation, uses the front and rear vertical thrusters as the system inputs. This case demonstrates the implementation of the MPC algorithm and studies the effects of the various tuning parameters. A model sensitivity study is performed, showing that the controller can handle modelling errors of up to $\pm 30\%$. The controller is experimentally tested and shows excellent performance with zero steady-state errors although there is an undesirably large overshoot of the depth demand. The simulation and experimental results match closely.

The depth and surge controller uses the control surfaces and rear propeller as system inputs. Many of the forces and moments within this system are non-linear functions of the vehicles surge velocity. Therefore the standard MPC algorithm, that utilizes just one linearised model, would not be sufficient to capture the system dynamics of the vehicle throughout the full operational envelope. A time-variant MPC (TV-MPC) algorithm is developed and shown in simulation to have excellent performance. The controller did not perform as well when tested experimentally, however, depth regulation of ± 0.3 m was achieved. This degradation in performance is due to inaccuracies in the estimation of the vehicles surge velocity.

The final controller is also a depth and surge velocity controller, however, it is tasked with maintaining stability through-out the full speed range of the vehicle. All of the system inputs used for depth control are utilised by this controller; the two vertical through-body tunnel thrusters, horizontal control surfaces and the rear propeller. The design of the controller makes use of the TV-MPC algorithm. To improve system performance a modification to the controllers cost function, used within the optimisation process, was made to penalise the use of the thrusters at high speeds. This enables the controller to use the thrusters at low speeds, when performing close-range inspections, but then as surge velocity increases and the thrusters are no longer required, they are switched off. Both simulation and experimental results show excellent performance, although when the thrusters switch off, the depth control is similar to that of the previous controller due to poor surge velocity estimation.

Declaration

I, Leo Vincent Steenson, declare that the thesis entitled

EXPERIMENTALLY VERIFIED MODEL PREDICTIVE CONTROL
OF A HOVER-CAPABLE AUV

and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- none of this work has been published before submission.

Signed:

Date:

Publications

‘Observer based MPC for Depth Control of AUVs with Experimental Verification,’ Leo V. Steenson, Liuping Wang, Alex B. Phillips, Stephen R. Turnock, Maarten E. Furlong, Eric Rogers. 2013 IEEE Conference on Decision and Control (CDC), submitted paper for which a decision will be issued in July 2013.

‘Effect of Measurement Noise on the Performance of a Depth and Pitch Controller using the Model Predictive Control Method,’ Leo V. Steenson, Alexander B. Phillips, Eric Rogers, Maaten E. Furlong, Stephen R. Turnock. Autonomous Underwater Vehicles (AUV) 2012.

‘Experimental Verification of a Depth Controller using Model Predictive Control with Constraints onboard a Thruster Actuated AUV,’ Leo V. Steenson, Alexander B. Phillips, Eric Rogers, Maaten E. Furlong, Stephen R. Turnock. IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles (NGCUV) 2012.

‘Control of an AUV from Thruster Actuated Hover to Control Surface Actuated Flight,’ Leo V. Steenson, Alexander B. Phillips, Eric Rogers, Maaten E. Furlong, Stephen R. Turnock. RTO-MP-AVT-189 Assessment of Stability and Control Prediction Methods for NATO Air and Sea Vehicles 2011.

‘Maneuvering of an over-actuated autonomous underwater vehicle using both through-body tunnel thrusters and control surfaces,’ Leo V. Steenson, Alexander B. Phillips, Eric Rogers, Maaten E. Furlong, Stephen R. Turnock. Unmanned Untethered Submersible Technology (UUST) 2011.

‘Preliminary results of a hover capable AUV attempting transitional flight,’ Leo V. Steenson, Alexander B. Phillips, Eric Rogers, Maaten E. Furlong, Stephen R. Turnock. Unmanned Untethered Submersible Technology (UUST) 2011.

‘The performance of vertical tunnel thrusters on an autonomous underwater vehicle operating near the free surface in waves.’ Steenson, L., Phillips, A., Rogers, E., Furlong, M. and Turnock, S. R., Second International Symposium on Marine Propulsors (SMP 2011).

Acknowledgements

Working on this PhD, whether it was coding into the small hours of the morning, using a lathe on a rolling ship or just simply playing with submarines, has been a fantastically enjoyable few years - thanks to some very important people.

Of course there would not have been a PhD project without my supervisors; Professor Stephen Turnock, Professor Eric Rogers and Dr Maaten Furlong. Their combined wealth of knowledge, encouragement, guidance and occasional eccentricity(!) was an invaluable help to me throughout my studies.

I am also greatly indepted to Dr Alexander Phillips who acted very much as a fourth supervisor but also as a good friend. I must thank him for his help developing Delphin2, proof reading my publications, sitting in the cold and wet whilst I tested the AUV and of course... steering me towards the pub every Friday afternoon at 4pm!

To my past and present colleagues at Schlumberger Gould Research who first inspired me to pursue a career in research I am extremely grateful. In particular I would like to thank Gary Oddie for his continuous enthusiasm and energy for all things science.

Most importantly, I want to thank my friends and family for their support over the past few years. To my lovely girlfriend Amy who has had to put up with me during the (less enjoyable) write-up and for reminding me that there is more to the world than submarines! I am also thankful to my sisters, Oonagh and Brona, who have always been there for me, I will always value your opinion and friendship.

Finally I must thank my parents, Bill and Chris, first for having me(!) but mostly for giving me a fantastic childhood, supporting me throughout all my studies, and for giving me the occasional (but necessary) kick up the... Thank you.

Contents

1 Introduction **1**

1.1 Introduction . . . 2

1.2 Research Objectives . . . 3

1.3 Project Background . . . 4

1.4 Research Aims . . . 5

1.5 Structure of Thesis . . . 6

2 Control of Unmanned Underwater Vehicles **9**

2.1 Introduction . . . 10

2.2 Unmanned Underwater Vehicles (UUV) . . . 10

2.2.1 Remotely Operated Vehicles (ROVs) . . . 10

2.2.2 Autonomous Underwater Vehicles (AUVs) . . . 13

2.2.3 Intervention Autonomous Underwater Vehicles (IAUVs) . . . 14

2.2.4 Discussion . . . 18

2.3 UUV Low-Level Control Systems . . . 18

2.4 Conclusions . . . 24

3 Model Predictive Control (MPC) **25**

3.1 Introduction . . . 26

3.2 MPC Algorithm . . . 27

3.2.1 Augmented Model . . . 28

3.2.2 Prediction of the System Trajectory . . . 29

3.2.3 Optimization . . . 31

3.2.4 Constraints . . . 32

3.2.5 Quadratic Programming . . . 34

3.3 Feedback . . . 40

3.3.1 Quantifying Random Error . . . 40

3.3.2 Signal Noise Ratio . . . 43

3.3.3 Filtering . . . 44

3.3.4 State Observer . . . 45
3.4 Implementation . . . 47
3.4.1 Python Toolboxes . . . 48
3.4.2 Conclusions . . . 49

4 Delphin2 AUV **51**
4.1 Introduction . . . 52
4.1.1 Overview . . . 53
4.2 Mechanical Design . . . 55
4.2.1 Thruster Units . . . 56
4.2.2 Tail Unit . . . 56
4.2.3 Fairings . . . 58
4.3 Actuators . . . 58
4.3.1 Through-body Tunnel Thrusters . . . 58
4.3.2 Control Surfaces . . . 58
4.4 Sensors and Electronics . . . 59
4.4.1 Central Computer . . . 59
4.4.2 Battery and Power Management . . . 59
4.4.3 Communications . . . 59
4.4.4 Thruster and Tail Unit Control Electronics . . . 60
4.4.5 Sensors . . . 61
4.5 Software . . . 62
4.5.1 Software Architecture . . . 63
4.6 Conclusions . . . 65

5 Physics Model of the Delphin2 AUV **67**
5.1 Introduction . . . 68
5.1.1 Axis System . . . 68
5.1.2 AUV Model . . . 68
5.2 Hull and Control Surfaces . . . 70
5.2.1 Wind Tunnel Tests . . . 71
5.3 Through-body Tunnel Thrusters . . . 75
5.3.1 Experimental Set-up . . . 79
5.3.2 Quasi-Steady Model . . . 80
5.3.3 Motor Model . . . 81
5.3.4 Transient Thruster Model . . . 81
5.3.5 Thruster Performance Near the Free Surface . . . 83

5.3.6 Thruster Performance with Positive Surge Velocities 88
5.4 Open Water Tests 89
5.5 Rear Propeller 92
5.6 AUV Model 93
5.7 Conclusions 95

6 Depth and Pitch Control using MPC 97
6.1 Introduction 98
6.2 System Model 98
6.2.1 Linearised State-Space Model 100
6.3 Constraints 102
6.3.1 Implementation 103
6.4 Tuning 104
6.4.1 Estimate of the Minimum Length of the Prediction Horizon 106
6.4.2 Overshoot 107
6.4.3 Depth Settling Time 107
6.4.4 Pitch Settling Time 108
6.4.5 Pitch Disturbance 109
6.4.6 Summary 109
6.4.7 Conclusions 113
6.5 Model Sensitivity 114
6.5.1 Mass and Inertia Terms 115
6.5.2 Damping Terms 118
6.5.3 Restoring Moment 121
6.5.4 Summary 122
6.6 Feedback Quality 123
6.6.1 Polynomial-Type Filter 124
6.6.2 Observed Feedback 127
6.6.3 Summary 129
6.7 Experimental Verification 130
6.7.1 Effect of Tuning Parameters 131
6.7.2 Model Verification 135
6.8 Conclusions 135

7 Flight Style Depth Control using MPC 139
7.1 Introduction 140
7.2 Flight-Style System 140

7.2.1 Surge . . . 141
7.2.2 Depth . . . 142
7.2.3 Pitch . . . 143
7.2.4 Discussion . . . 143
7.3 Controller Design . . . 144
7.3.1 Time-Variant Model Predictive Control . . . 144
7.3.2 Tuning . . . 146
7.3.3 Constraints . . . 147
7.4 Time-Variant State-Space Model . . . 147
7.4.1 Input, Output and State-Variable Vectors . . . 147
7.4.2 Linearisation Methods . . . 148
7.4.3 Surge . . . 149
7.4.4 Depth . . . 150
7.4.5 Pitch . . . 152
7.4.6 Flight-Style State-Space Model . . . 153
7.5 Observer Design . . . 153
7.6 Controller Performance in Simulation . . . 156
7.6.1 Basic Performance . . . 156
7.6.2 Controller Tuning by Model Adaptation . . . 158
7.6.3 TV-MPC Limitations . . . 159
7.7 Experimental Verification . . . 162
7.7.1 Effect of Inner Gains . . . 162
7.7.2 Depth Step-Changes . . . 164
7.8 Discussion . . . 166
7.9 Conclusions . . . 169

8 Depth Control within the Transitional Zone 171
8.1 Introduction . . . 172
8.2 Over-Actuated AUV System . . . 173
8.2.1 Surge . . . 173
8.2.2 Depth . . . 174
8.2.3 Pitch . . . 174
8.3 Linearisation . . . 175
8.3.1 Thruster components . . . 175
8.3.2 State-Space Model . . . 175
8.4 Controller Design . . . 176

8.4.1 Tuning . . . 176
8.4.2 Constraints . . . 177
8.5 Controller Tuning by Model Adaptation . . . 177
8.5.1 Low Speed Operation . . . 178
8.5.2 High Speed Operation . . . 179
8.5.3 Effect of Inner Gains . . . 180
8.6 Actuator Transition . . . 183
8.7 Experimental Verification . . . 185
8.8 Conclusions . . . 188

9 Conclusions 191
9.1 Delphin2 AUV . . . 192
9.1.1 Vehicle Development . . . 193
9.1.2 Future Research . . . 194
9.2 Model Predictive Control . . . 196
9.2.1 Low Speed Hovering Control . . . 196
9.2.2 Flight-Style Control . . . 198
9.2.3 Transitional Control . . . 200
9.2.4 Final Remarks . . . 202
9.3 Future Research . . . 202
9.3.1 Surge Velocity Estimation . . . 202
9.3.2 Six Degrees of Freedom Control . . . 203
9.3.3 Depth Overshoot Reduction . . . 204
9.3.4 Fault Detection and Handling . . . 204
9.3.5 Universal AUV Controller . . . 204
9.3.6 Embedded Low-Level Control . . . 205

A MPC Code 217
A.1 MATLAB . . . 218
A.1.1 Over-Actuated MPC Controller . . . 218
A.2 Python . . . 228

B Video survey of Lough Erne using the Delphin2 AUV 241
B.1 Introduction . . . 242
B.2 Video Survey of Carrickreagh Bay . . . 245
B.2.1 Vehicle Performance . . . 246
B.2.2 Video Data . . . 248

B.3 Delphin2 ROV Operations 248
B.4 Conclusions 249

List of Tables

3.1 Acceptable magnitude of errors for the terms within the controller model with the stable and balanced controller parameter sets. 42

4.1 Overview of Delphin2 54

5.1 Notation used when modelling marine vehicles for all six degrees of freedom [1] 69
5.2 Specification for the control surfaces on the Delphin2 AUV. 71
5.3 Hydrodynamic lift, drag and pitching moment coefficients for Delphin2. 77
5.4 Distances between centre of gravity and thrusters along x axis 78
5.5 Damping coefficients and mass terms along and about z and y axes. 92
5.6 Wave making resistance at various surge velocities for the Delphin2 AUV at zero depth and pitch. 93
5.7 Typical masses, inertia, weight and buoyancy terms for Delphin2. 94

6.1 Depth and pitch controller overview 98
6.2 Linear state-space model parameters 102
6.3 Controller parameters 103
6.4 Controller parameters varied in simulation to find optimum controller parameters . . . 105
6.5 Three sets of controller parameters that will be used later to evaluate controller robustness. 114
6.6 Acceptable magnitude of errors for the parameters within the controller model with the stable and balanced controller parameter sets. 123

7.1 Tuning parameters for flight-style controller 146
7.2 Input constraints for flight-style controller 147
7.3 Weighting scalars used to tune flight-style observer 156

8.1 Tuning parameters for over-actuated controller 186

List of Figures

2.1 Launching of ROV Isis from RRS James Cook. Note the two manipulator arms in the folded position and the various lights, cameras and thrusters. 11
2.2 Holland I Deepwater ROV beside the RV Celtic Explorer in Galway harbour. Note the two large thrusters, that are used to manoeuvre the vehicle, and the extended manipulator arm. 11
2.3 Autosub6000 being recovered onto the RRS James Cook. Note the location of four control surfaces and a propeller at the rear of the vehicle. 14
2.4 Odyssey IV AUV. This vehicle has the ability to hover and transit at up to 2.0 m/s. Picture courtesy of MIT. 15
2.5 The MUST AUV arriving, in sections, at the M-Subs factory in Plymouth, UK, 2012. 16

3.1 Example of the prediction and control horizons calculated by the MPC algorithm. . . 26
3.2 Flow diagram of information into and out of the model predictive controller in the Delphin2 AUV. 28
3.3 Effect of window size used by the Savitzky-Golay filter on the resulting mean and standard deviation of the random error for the depth signal. 41
3.4 An example of a depth signal before and after filtering and at the bottom, the random error on the depth signal. 42
3.5 Histogram distributions for the random noise from the depth and pitch signals. 43
3.6 Standard deviation of the errors between the PT and SG filters for both the depth and heave signals, against the length of the PT array. Note that the minimum for both signals occurs at an array length of 32. 45
3.7 Standard deviation of the errors between the PT and SG filters for both the pitch and pitch rate signals, against the length of the PT array. Note that the minimum for both signals occurs at an array length of 18. 46
3.8 Flow diagram of information into and out of the model predictive controller in the Delphin2 AUV. 48

4.1 Delphin2 AUV at Eastleigh Lakes. 52

4.2 Computer rendered image identifying the location of the Delphin2 AUV actuators. . . 53
4.3 Three major components of the Delphin2 AUV. 54
4.4 Exploded view of one end of the pressure vessel; the end flange inserts into the outer cylinder and the end plate bolts to the end flange. 55
4.5 Side view of the pressure vessel with the outer cylinder set as translucent. Note the position of the battery and electronics shelf. 56
4.6 Thruster units on Delphin2. Both the front and rear thruster units are of identical design, with one vertical and one horizontal tunnel thruster each. 57
4.7 Tail unit with pressure vessel and rear prop-box set to be translucent. Note the location of the linear actuators and associated mechanisms. The propeller motor is located between the linear actuator shelves. 57
4.8 70 mm rim-driven thruster used on Delphin2. Note the location of the thruster motor and propeller. 58
4.9 Architecture of tail unit control electronics. 60
4.10 Locations of the sensors on Delphin2. 61
4.11 Example of a node publishing information to a topic and a second node subscribing to the same topic. 63
4.12 Flow-diagram illustrating the architecture of the Delphin2 AUV software. 64

5.1 Motion variables of a marine vehicle [1]. 68
5.2 Computer rendered image of the rear of the Delphin2 AUV. Note the location of the control surfaces and the magnetic couplings used to transmit the rotational torque between the control surface and actuator inside the tail unit pressure vessel. 70
5.3 Dimensions of the control surfaces on Delphin2. 71
5.4 Delphin2 AUV mounted in the wind-tunnel at the University of Southampton. 72
5.5 Lift coefficient for the Delphin2 hull with the control surfaces at zero degrees, against hull angle of attack. 73
5.6 Drag coefficient for the Delphin2 hull with the control surfaces at zero degrees, against hull angle of attack. 74
5.7 Pitching moment coefficient for the Delphin2 hull with the control surfaces at zero degrees, against hull angle of attack. 75
5.8 Lift coefficient for the Delphin2 control surfaces, against control surface angle of incidence. 76
5.9 Drag coefficient for the Delphin2 control surfaces, against control surface angle of incidence. 76
5.10 Pitching moment coefficient for the Delphin2 control surfaces, against angle of incidence. 77

5.11 Illustration indicating the location and direction of force vectors for the front and rear vertical thrusters. 77

5.12 70 mm diameter thruster used on the Delphin2 AUV. Note the black rims are removed when fitted to the Delphin2 AUV. 78

5.13 Experimental set-up with the vertical tunnel thruster in the Lamont towing tank at the University of Southampton. 79

5.14 Experimental data of measured thrust against $\rho n^2 D^4$. 80

5.15 Thrust coefficient K_T against advance ratio J for the 250 mm thruster, data from [2]. 81

5.16 Experimental and simulated data for a step response in speed demand for the thruster motors. 82

5.17 Simulation using the transient and quasi-steady thruster model along with measured experimental data. Note that the quasi-steady model fails to adequately approximate the thruster dynamics while the transient model does. 84

5.18 Measured thrust against depth for four thruster speeds. 84

5.19 Thrust values, calculated using the differential pressure data, against $\rho n^2 D^4$. 86

5.20 Calculated thrust against depth for four thruster speeds. 86

5.21 Differential pressure across outer tube against depth and thruster speed. 87

5.22 Illustration demonstrating the effective blockage on the thruster jet by the free surface (when operating at depths less than 200mm) and the subsequent high velocity streamlines across the top of the hull causing an area of low pressure. 87

5.23 Normalised thrust against the ratio of surge velocity over thruster jet velocity. Data from [3]. 88

5.24 Flow diagram illustrating the experimental method for achieving terminal velocities in the z axis. 90

5.25 Results from a terminal velocity test along the z axis. Note the AUV first stabilises at 0.8 m depth then the depth controller is switched off and and addition 2N is added to both thrusters. The AUV then dives and eventually reaches a steady-state heave velocity of 0.114 m/s at approximately 80 seconds. 91

5.26 Drag coefficients for the z axis against heave velocity. Note slight variation, with a decreasing trend within increasing surge velocity. This is likely due to changes in flow due to varying Reynolds number. 91

5.27 Surge velocity of the Delphin2 AUV against motor set-point. Note that this data is the mean average of the steady-state velocities measured using the GPS sensor when the AUV is operated on the surface. 93

6.1 Illustration demonstrating the location of the thruster force vectors, moment arms, centre of gravity and buoyancy and the axis system. Note that the AUV in the illustration is at a pitch angle of $-10°$. 99

6.2 Comparison between the linear and non-linear damping forces and moments for pitch and heave. 101

6.3 Annotated simulation results demonstrating how overshoot, pitch disturbance and the depth and pitch settling times are calculated. 106

6.4 Overshoot (m) for different r_w, prediction and control horizons (N_p and N_c respectively). Note the white areas represent low overshoot values. 108

6.5 Depth settling time (s) for different r_w, prediction and control horizons (N_p and N_c respectively). Note the lighter shade of grey represents faster settling times while black indicates that the vehicle has failed to converge within the tolerances after 50 seconds and is most likely unstable. 109

6.6 Pitch settling time (s) for different r_w, prediction and control horizons (N_p and N_c respectively). Note the lighter shade of grey represents faster settling times while black indicates that the vehicle has failed to converge within the tolerances after 50 seconds and is most likely unstable. 110

6.7 Maximum absolute pitch error (deg) for different r_w, prediction and control horizons (N_p and N_c respectively). Note that lighter patches indicate low pitch disturbance angles. 111

6.8 Top sub-plot: overshoot and pitch disturbance against the length of the prediction horizon. Bottom sub-plot: depth and pitch settling times against the length of the prediction horizon. 112

6.9 The optimum length of control horizons that produced the minimum overshoot, pitch disturbance, depth and pitch settling times, all against the length of the prediction horizon. 112

6.10 The optimum weighting scalar, r_w, that produced the minimum overshoot, pitch disturbance, depth and pitch settling times, all against the length of the prediction horizon. 113

6.11 Change in overshoot against change in mass and inertia values. 115

6.12 Change in pitch disturbance against change in mass and inertia values. 116

6.13 Change in depth settling time against change in mass and inertia values. 117

6.14 Change in pitch settling time against change in mass and inertia values. 118

6.15 Change in overshoot against change in depth and pitch damping values. 119

6.16 Change in pitch disturbance against change in depth and pitch damping values. 120

6.17 Change in depth settling time against change in depth and pitch damping values. . . . 120

6.18 Change in pitch settling time against change in depth and pitch damping values. . . . 121

6.19 Change in performance against change in mass and inertia, depth and pitch damping values and restoring moment. 122

6.20 Simulation results of the depth and pitch MPC controller using the polynomial type filter to provide state feedback. Note that noise has not been added to either the depth or pitch signals yet system performance is very poor. 124

6.21 Comparison between the estimated heave velocity, from the polynomial type filter, and the actual heave velocity provided by the non-linear model. Note that the bottom plot is the of the error between the two signals normalised by the RMS heave velocity. . . . 126

6.22 Comparison between the estimated pitch rate, from the polynomial type filter, and the actual pitch rate provided by the non-linear model. Note that the bottom plot is the of the error between the two signals normalised by the RMS pitch rate. 126

6.23 Simulation results of the depth and pitch filter using the polynomial type filter, with the addition of measurement noise on the depth and pitch signals. Note that performance is poor with noise dominating the controller output. 127

6.24 Effect of the observer gain, r_o, on the overshoot and depth settling time. 129

6.25 Simulation results of the MPC controller using the observer with an r_o value of 10. Note that the noise still effects the controller performance but within acceptable limits. 130

6.26 Experimental results of the MPC controller with depth set-points of 1 m, 3 m and 2 m. The pitch set-point remains fixed at 0.0°. Controller parameters: N_p = 60, N_c = 8, r_w = 0.5, r_o = 50. 132

6.27 Experimental results of the MPC controller with depth set-points of 1 m, 3 m and 2 m. The pitch set-point remains fixed at 0.0°. Controller parameters: N_p = 80, N_c = 8, r_w = 4.0, r_o = 50. 133

6.28 Experimental results of the MPC controller with depth set-points of 1 m, 3 m and 2 m. The pitch set-point remains fixed at 0.0°. Controller parameters: N_p = 100, N_c = 8, r_w = 12.0, r_o = 50. 134

6.29 Comparison between simulated and experimental results of the MPC controller with depth set-points of 3 m and 2 m, whilst maintaining 0.0° pitch. Note both simulated and experimental depth control results are very similar with a slight divergence when the heave velocity is negative. The pitch does not agree as closely. 136

7.1 Illustration demonstrating the major forces and moments acting on the Delphin2 AUV when operating in flight style mode. 141

7.2 Flow diagram illustrating the TV-MPC algorithm. The difference between the normal MPC and TV-MPC algorithms is that the latter builds a new linearised model at the beginning of each control loop (in comparison to using just one fixed linear model). . . 145

7.3 Comparison of drag forces calculated using the non-linear function and the linearised function (linearised at 0.5 m/s). 149
7.4 Comparison between observed and real state signals. Note that the observer has feedback of depth, resulting in the observed depth and surge velocity signals tracking the real signals accurately. However, as there is no feedback of the pitch state, both pitch and pitch rate diverge quickly from those of the real system causing the controller to fail. 154
7.5 Comparison between observed and real state signals. In this observer, feedback of all three measured values is fed back into the observer resulting in the observer accurately tracking all five states. 155
7.6 Simulation of the flight-style depth and speed controller. Note both outputs converge to the set-points however the depth signal does have an unacceptably large overshoot. 157
7.7 Simulation of the flight-style depth and speed controller with gains applied to the heave and pitch damping terms. Note the significantly faster convergence to the depth set-point with no overshoot. 159
7.8 Simulation of the flight-style depth and speed controller with step changes in surge velocity set-points. Note that disturbances occur to the depth tracking when the surge velocity changes. 160
7.9 Simulation of the flight-style depth and speed controller with step changes in surge velocity set-points. A very large $r_{w(prop)}$ of 250 has been used to reduce the acceleration of the vehicle in the x axis. 161
7.10 Experimental results of the TV-MPC flight-style controller with surge velocity and depth set-points of 1.0 m/s and 1.0 m respectively. The inner gain has been applied to both the depth and pitch damping terms within the state-space model. 163
7.11 Experimental results of flight-style controller with surge velocity and depth set-points of 1.0 m/s and 1.0 m respectively. No inner gain has been applied to the pitch damping within the state-space model. 164
7.12 Experimental results of flight-style controller with surge velocity and depth set-points of 1.0 m/s and 1.0 m respectively. Pitch damping within the state-space model is set to zero. 165
7.13 Experimental results of flight-style controller with a fixed surge velocity set-point of 0.75 m/s, and depth set-points of 1.0 m, then 2.0 m and finally back to 1.0 m. 166
7.14 Closer look at the results shown in Figure 7.13. 167

8.1 Simulation results of the controller, with only the thrusters active, with a step change in depth set-point from 0.0 to 1.0 m while maintaining a surge velocity of zero. Note a slight disturbance to surge velocity whilst diving phase but converges back to zero. . . 178

8.2 Simulation results using the standard over-actuated controller, with thrusters disabled. Note quick convergence to the surge velocity set-point with no overshoot. 179

8.3 Simulation results using the over-actuated controller, with thrusters disabled, and gains applied to the heave and pitch damping terms (both equal to 40). 180

8.4 Simulation results using the over-actuated controller with gains applied to the heave and pitch damping terms (both equal to 40). A depth set-point of 1.0 m is applied whilst maintaining zero surge velocity. 181

8.5 Simulation results demonstrating the effectiveness of using the surge velocity to schedule the inner gains applied to the system model. 182

8.6 Simulation of over-actuated controller achieving surge velocity and depth set-points of 1.0m and 1.0m/s respectively. The controller has no designated control authority, and so the thruster demands are near their maximum constraints during high speed operating when the control surfaces control depth independently. 184

8.7 Simulation of over-actuated controller achieving surge velocity and depth set-points of 1.0m and 1.0m/s respectively. The control authority is designated to the control surfaces by penalising the use of the thrusters using equations (8.15) and (8.16). . . . 185

8.8 Experimental results of the over-actuated surge velocity and depth controller. One depth set-point of 1.0 m is used with step changes in surge velocity set-points from 0.0 to 0.5 to 0.75 and then back to 0.0 m/s. Initial surge velocity and depth are both approximately zero. 188

8.9 Experimental results of the over-actuated surge velocity and depth controller. One depth set-point of 1.0 m is used with an initial surge velocity set-points of 0.0 m/s, then at 70 seconds the surge velocity set-point is changed to 1.0 m/s. 189

B.1 Map of Northern Ireland indicating the location of Lower Lough Erne that hosted Delphin2 AUVs first science mission. Image from Google Maps. 242

B.2 Map of Lower Lough Erne, indicating were the team was based and the two main areas of study; Carrickreagh Bay and Devenish Island. Image from Google Maps. 243

B.3 The Delphin2 AUV in the back of the AFBI operated boat before sailing to the test location. 243

B.4 Delphin2 AUV being launched off the side of the AFBI boat at Carrickreagh Bay. . . 244

B.5 Delphin2 AUV returning to the jetty in Carrickreagh Bay after a successful video survey of the lough bottom. 244

B.6 Satelite image of Carrickreagh Bay. The three transects that were performed are shown in the Figure. Note the location of the jetty that served as the start and end location of each mission. Image from Google Maps. 245

B.7 Depth, altitude and temperature data during transect 1. 246
B.8 Depth, altitude and temperature data during transect 2. 247
B.9 Depth, altitude and temperature data during transect 3. 247
B.10 Video frames, from the downwards looking camera, from the three transects. Frame 656: Small beds of Zebra mussels and a type of fresh-water sponge. Frame 731: Beds of Zebra mussels feeding using their funnels. Frame 4426: Bed of Zebra mussels. Note the white shells are the remains of dead mussels. Frame 9902: Large bed of Zebra mussels and an unidentified object. 249
B.11 Video frames, from the forwards looking camera, from the three transects. Frame 1219: White object at the bottom of the image is a fresh-water sponge. Frames 3076, 4609, and 7430: Small scattered beds of Zebra mussels. Note the forwards facing camera gives better 3D perspective than the downwards camera. 250
B.12 Video frames, from the forwards looking camera, whilst operating the Delphin2 AUV in ROV mode. Frames 9397 and 9955: Carrickreagh jetty stanchions. Note the vast coverage of Zebra mussels, in particular at the bottom of the stanchion. Frames 11553 and 12222: Fyke net, located to the north of Devenish Island, with a captured eel. . . 251

List of Symbols and Acronyms

α	Angle of incidence ($\circ$)
$\bar{R}$	Weighting matrix used to penalise the magnitude of values within the ΔU vector
ΔC_L	Change in lift coefficient per degree of angle of incidence
ΔU	Vector of future change in control inputs
Δu	Change of system input vector
ΔU^{max}	Vector containing maximum constraints on ΔU
ΔU^{min}	Vector containing minimum constraints on ΔU
$\dot{m}_{exit}$	Mass flow rate out of thruster (m^3/s)
$\dot{m}_{in}$	Mass flow rate into thruster (m^3/s)
$\dot{q_v}$	Pitch acceleration (m/s^2)
$\dot{u_v}$	Surge acceleration (m/s^2)
$\dot{w_v}$	Heave acceleration (m/s^2)
γ	Vector of constraints
$\hat{x}$	Estimate of state-variable vector using an observer
$\hat{y}$	Estimate of output vector using an observer
λ	Lagrange correction vector
∇	The volume of fluid displaced by the AUV (m^3)
ν_{air}	Kinematic viscosity of air (m^2/s)
ν_{water}	Kinematic viscosity of water (m^2/s)
ϕ	Block matrix used in the prediction of the future system output trajectory Y, see subsection 3.2.2.

ψ	Euler angle about the z-axis (yaw) ($\circ$)
ρ	Fluid density (kg/m^3)
θ	Euler angle about the y-axis (pitch) ($\circ$)
^{T}	Transpose of matrix or vector
A	Dynamics matrix of augmented model
A_m	Dynamics matrix for continuous state-space model
A_z	Projected area of AUV along z axis (m^2)
A_{exit}	Cross-sectional area at the end of thruster tunnel (m^2)
B	Buoyancy of AUV (N)
B	Input matrix of augmented model
B_m	Input matrix for continuous state-space model
C	Output matrix of augmented model
C_1	Block matrix used when defining the constraints on a system, see subsection 3.2.4
C_2	Block matrix used when defining the constraints on a system, see subsection 3.2.4
C_D	Drag coefficient
C_L	Lift coefficient
C_M	Moment coefficient
C_m	Output matrix for continuous state-space model
C_{Dcs}	Control surface drag coefficient
C_{Dh}	Hull drag coefficient
C_{Lcs}	Control surface lift coefficient
C_{Lh}	Hull lift coefficient
C_{Mcs}	Control surface moment coefficient (pitch)
C_{Mh}	Hull moment coefficient (pitch)
D	Thruster diameter (m)

e_r	Random error
e_s	Systematic error
E_{qp}	Matrix used in examples for solving quadratic programming problems
F	Block matrix used in the prediction of the future system output trajectory Y, see subsection 3.2.2.
F_{qp}	Matrix used in examples for solving quadratic programming problems
G_q	Gain applied to pitch damping term to move poles away from real axis
G_z	Gain applied to heave damping term to move poles away from real axis
I	Identity matrix
I_y	Inertia term (including added inertia) for the y axis (kgm^2)
J	Advance ratio
J	Cost function
J_0	Advance ratio at which K_T equals zero
K	Moment about the x-axis (roll) (Nm)
k	Sample number
k_i	Current sample number
k_q	Linear drag coefficient for about the y axis
K_T	Thrust coefficient
k_z	Linear drag coefficient for along the z axis
k_{Dcs}	Linearised control surface damping term for flight-style operation
k_{Dh}	Linearised hull damping term for flight-style operation
k_{Lcs}	Linearised control surface lift term for flight-style operation
k_{Lcs}	Linearised hull lift term for flight-style operation
k_{Tu_w}	Coefficient used to normalise the force produced by a thruster as a function of surge velocity
L	Luenberger observer matrix
L	Time constant for the ideal delay of the thruster motor (s)

M	Matrix identifying variables being constrained using constraints in γ
M	Moment about the y-axis (pitch) (Nm)
m	Number of system inputs
M_1	Block matrix used when defining the constraints on a system, see subsection 3.2.4
M_2	Block matrix used when defining the constraints on a system, see subsection 3.2.4
m_x	Mass term (including added mass) for the x axis (kg)
m_z	Mass term (including added mass) for the z axis (kg)
M_{cs}	Moment generated by the control surfaces acting about the y axis (Nm)
M_D	Moment generated by hydrodynamic damping acting about the y axis (Nm)
M_{hull}	Moment generated by the hull acting about the y axis (Nm)
M_{Tf}	Moment generated by the front thruster acting about the y axis (Nm)
M_{Tr}	Moment generated by the rear thruster acting about the y axis (Nm)
N	Moment about the z-axis (yaw) (Nm)
n	Thruster speed (rps)
N_1	Block matrix used when defining the constraints on a system, see subsection 3.2.4
n_1	Number of system states
N_2	Block matrix used when defining the constraints on a system, see subsection 3.2.4
N_c	Length of control horizon (number of samples)
N_p	Length of prediction horizon (number of samples)
P	Propeller pitch ratio
p_v	Angular velocity about the x-axis (roll) ($\circ/s$)
p_{exit}	Pressure immediately downstream of the thruster propeller (Pa)
p_{in}	Pressure immediately upstream of the thruster propeller (Pa)
Q	Weighting matrix for calculating observer matrix L
q	Number of system outputs

q_v	Angular velocity about the y-axis (pitch) ($\circ/s$)
R	Weighting matrix for calculating observer matrix L
$r(k_i)$	Set-point at sample k_i
r_o	Scalar using to adjust weighting matrix R
R_s	Set-point vector
r_v	Angular velocity about the z-axis (yaw) ($\circ/s$)
r_w	Scalar value used within $\bar{R}$. Higher values reduces the magnitude of the values within the ΔU vector
S	Array of samples
T	Array of sample time-stamps
T	Thrust force (N)
t	Time (s)
T_m	Time constant for the ideal delay of the thruster motor (s)
T_w	Difference between thruster motor and thrust time constants (s)
T_{add}	Additional thrust value added to front and rear thrusters used to achieve terminal velocities along the z axis (N)
u	System input vector
u_v	Velocity in the x-direction (surge) (m/s)
V_a	Velocity of advance (m/s)
v_v	Velocity in the y-direction (sway) (m/s)
V_{exit}	Fluid velocity exiting the thruster (m/s)
V_f	Flow velocity the the tunnel-thruster propeller (m/s)
V_{in}	Fluid velocity entering the thruster (m/s)
V_{ss}	Steady-state flow velocity through the tunnel-thrusters (m/s)
W	Weight of AUV (N)
w_v	Velocity in the z-direction (heave) (m/s)

X	Force in the x-direction (surge) (N)
x	Position in the x-direction (heave) (m/s)
x_m	State variable vector for continuous state-space model
X_D	Force generated by hydrodynamic damping acting along the x axis (N)
X_{prop}	Force generated by the rear propeller acting along the x axis (N)
X_{Tf}	Force generated by the front thruster acting along the x axis (N)
x_{Thf}	Distance along x axis between centre of gravity and front horizontal thruster (m)
x_{Thr}	Distance along x axis between centre of gravity and rear horizontal thruster (m)
X_{Tr}	Force generated by the rear thruster acting along the x axis (N)
x_{Tvf}	Distance along x axis between centre of gravity and front vertical thruster (m)
x_{Tvr}	Distance along x axis between centre of gravity and rear vertical thruster (m)
Y	Force in the y-direction (sway) (N)
y	Position in the y-direction (sway) (m/s)
y	System output vector
Z	Force in the z-direction (heave) (N)
z	Position in the z-direction (heave) (m/s)
Z_{cs}	Lift force generated across the control surfaces acting along the z axis (N)
Z_D	Force generated by hydrodynamic damping acting along the z axis (N)
Z_{hull}	Lift force generated across the hull acting along the z axis (N)
Z_{prop}	Force generated by the rear propeller acting along the z axis (N)
Z_{Tf}	Force generated by the front thruster acting along the z axis (N)
Z_{Tr}	Force generated by the rear thruster acting along the z axis (N)
ADC	Analogue to digital converter
AFBI	Agri-food biosciences institute
AI	Artificial intelligence

API	Application programming interface
AUV	Autonomous underwater vehicle
BOP	Blow-out preventer
CAD	Computer aided design
CCD	Charge-coupled device
CFD	Computational fluid dynamics
CG	Centre of gravity
COTS	Commercial of the shelf
CSV	Comma-separated values
CTD	Conductivity, temperature and depth
DAC	Digital to analogue converter
DAQ	Data acquisition system
DVL	Doppler velocity log
GPS	Global positioning system
GRP	Glass re-enforced plastic
GSM	Global system for mobile communications
IAUV	Intervention Autonomous Underwater Vehicle
LTS	Long term support
MPC	Model predictive control
NiMH	Nickelmetal hydride
NMPC	Non-linear model predictive control
OTSC	Off-the-shelf components
PID	Proportional-integral-derivative controller
PMM	Planar motion mechanism
PT	Polynomial type

PWM	Pulse-width-modulation
RA	Rotating Arm
ROS	Robotic operating system
ROV	Remotely operated vehicle
rpm	Revolutions per minute
rps	Revolutions per Second
SG	Savitzky-Golay filter
SSH	Secure shell
TV-MPC	Time-variant model predictive control
USB	Universal serial bus protocol
UUV	Unmanned underwater vehicle
V4L2	Video for Linux 2

Chapter 1

Introduction

1.1 Introduction

Unmanned underwater vehicles (UUVs) have, over the past thirty to forty years, become an essential tool for working in the marine environment. The scientists and engineers that operate UUVs are constantly demanding these vehicles to perform increasingly complex tasks in more hostile and uncertain environments. These demands are being met by advances in sensors, electronics, artificial intelligence (AI) and control system technology. By choosing to operate UUVs, instead of divers, the operator is able to work at greater depths and over larger areas, generally with substantially lower risk.

The variety of tasks that are asked of UUVs has led to a multitude of designs and configurations for both the physical vehicles and their software. There are however two distinct categories of UUVs; remotely operated vehicles (ROVs), and autonomous underwater vehicles (AUVs). The former, ROVs, are generally used for close range, detailed inspection and interaction with subsea structures. These vehicles are piloted by operators on the surface via an umbilical cable. This umbilical provides electrical power and high-bandwidth communications, therefore enabling them to conduct highly complex tasks such as acquiring close range sonar, video, chemical or magnetic measurements from hydrothermal vent sites, [4] [5] [6], or the inspection and interaction of subsea structures such as the damaged blow-out preventer (BOP) that contributed to the Deep-Water Horizon disaster [7]. The supply of electricity from the umbilical also enables ROVs to operate for long durations and use high-power tools and sensors that battery powered vehicles could not.

One of the major disadvantages of using an ROV is the high financial cost. To operate offshore requires a surface vessel, typically a ship, which can cost upwards of £30k per day, along with a highly trained operating team. Moreover, the dynamics of the umbilical often affect the ROV dynamics [8] [9] [10], thus making the vehicle difficult to control. The length of the cable is constrained so as to minimise this coupling effect and the hydrodynamic forces acting on the cable, thus, ROVs have a finite range between the vehicle and operator.

AUVs are typically of torpedo shape and used for long range surveys, [11] [12] [13]. These vehicles operate autonomously and so there is little, if any, communication between the operator and vehicle throughout its mission. The majority of operational AUVs perform relatively simple tasks where the AUV transits to a predefined location and then surveys a specific area. The most common type of survey performed by AUVs are called 'lawn-mower' surveys. These involve the vehicle traversing a large area back and forth whilst collecting data.

An excellent example of an AUV performing a lawn-mower survey can be found in [14]. In this work the Autosub6000 [11] was used to study specific areas within the Mid-Cayman Rise (MCR) in order to help identify the location of hydro-thermal vents. The vehicle collected bathymetry data, using a multi-beam sonar system, and chemical data. Using both data-sets, marine geologists were able to locate possible hydro-thermal vents by correlating narrow, high altitude mounds with areas that exhibited high mineral content, typical of hydro-thermal vent sites. In this case, the AUV was tasked with surveying a predefined area and the data was post-processed after the mission.

As the autonomy of AUVs continues to increase, these vehicles are beginning to perform less predefined missions such as in [15]. In this project the AUV was given a more general objective for the mission; to track a moving thermal front in the Monterey Bay. The ability of the vehicle to detect the front, and its various sections, enabled it to autonomously operate in a closed loop manner causing it to continuously alter its mission as the environment changed. Over the coming years AI will enable AUVs to perform multiple generalised objectives during one mission whereby the vehicle plans and then executes the finer details of the tasks.

With the present level of AI, operators are beginning to reach the limits of AUV capabilities due to the fundamental design of these vehicles. The majority of torpedo shaped AUVs utilise the hydrodynamic forces and moments, produced by the flow of water over the vehicle, in order to manoeuvre. This method of operation is called flight-style. A fundamental difficulty with the flight-style method is that it requires the vehicle to operate above a critical forward speed, often greater than 0.6 m/s, below which the vehicle becomes unstable [16]. This form of AUV is therefore constrained to operate above a minimum surge velocity and as a consequence the variety of missions it can perform is reduced.

It would be highly advantageous if these operational constraints could be relaxed or eliminated. The desire to operate slowly can be attributed to two significant benefits; transiting at slow speeds improves the resolution of sonar or optical data, and the safety and capability to operate close to the sea floor in complex terrains is greatly improved. To develop an AUV that is both controllable and efficient at low and high speeds is possible, see [17] [18] [19] [20], however control of such a vehicle has proven difficult. Early attempts to develop a controller based around a gain-scheduled PID controller did not give satisfactory performance, [21], with large disturbances in depth tracking when accelerating in surge and very slow convergence to depth set-points.

1.2 Research Objectives

Future AUVs will be required to operate at both low and high speeds in order to perform a wide range of missions. Currently most torpedo shaped AUVs have a minimum forward speed below which they become unstable. Thruster actuated ROVs and AUVs maintain stability at low speeds however they are inefficient when operating at high speeds. Thus an AUV that can operate both efficiently and effectively at various different speeds is required. To achieve this requires an AUV to utilize a larger set of actuators. This in turn leads to the primary problem that is addressed in this work, how to control an over-actuated AUV throughout a wide range of speeds using a multitude of different actuators?

The actuators on-board a typical AUV are; a rear propeller and four rear control surfaces. The control surfaces are used to produce forces and moments so as to control the yaw or pitch angle of the vehicle. These forces are proportional to the square of the vehicle's speed relative to the fluid. Therefore as the vehicle's speed tends towards zero, so too do the forces that can be generated. Below a critical speed, the forces that the control surfaces produce are insufficient to control the vehicle, this is known as the Chinese effect [16]. Due to the occurrence of stall, [22], the value of this critical speed is uncertain, further increasing the difficulty in controlling such an AUV. Thus, at low speeds, the forces and moments generated by the control surfaces are both low in magnitude, uncertain and non-linear.

To maintain stability below the critical speed, at which the Chinese effect would normally occur, an AUV can utilize additional actuators such as thrusters. At very low speeds the performance of thrusters is relatively predictable and controllable, however, as the vehicle increases in speed the effective force produced by the thrusters decreases rapidly. The forces produced by the thrusters are non-linear functions of the thruster speeds and the vehicle speed and orientation.

There lies a region of speeds where the force generated by the vehicle's control surfaces is low and uncertain, and where the thrusters forces are also low and uncertain. To control the vehicle in this region would require the controller to utilize all of the available actuators in order to maintain stability. The research objective is therefore how to control an uncertain, non-linear multi-input-multi-output (MIMO) system.

1.3 Project Background

This Thesis is one of many PhD and Masters level research projects, conducted at the University of Southampton (UoS), that have focused on the development of AUV related technologies. The Ship Science department at the UoS is world renowned for its experimental, numerical and theoretical research into the design and performance of marine vehicles, however this is primarily for surface craft such as ships and yachts. Since the mid 1990s, the National Oceanography Centre Southampton (NOCS) has been developing the Autosub series of AUVs. Five Autosub AUVs have been built in total, with two performing regular ocean going operations and another is in the final stages of development [23].

Given their geographical closeness and shared interests, many collaborative projects have been conducted between NOCS and the UoS, for example research on tunnel thruster performance on AUVs, [24], and extracting hydrodynamic derivatives and anomalies from experimental, [25], or computational fluid dynamics (CFD) data, [26]. Since 2005, three AUVs have been developed by a series of PhD and Masters students at the UoS with support from engineers at NOCS. The purpose of developing these vehicles has been to improve the students' 'real world' understanding of how AUVs work and operate whilst providing prototype platforms to experimentally verify novel technologies. The majority of effort for the first two AUVs, sotonAUV and Delphin1, was focused on competing in the student autonomous underwater challenge Europe (SAUC-E) competition [27] however, the latest AUV to be developed, Delphin2, is primarily utilised for academic research purposes and has been tested extensively throughout this work.

The majority of the collaborative projects between the two institutions have been instigated by NOCS in order to facilitate academic research that could generate information aiding the development of future Autosub vehicles. Indeed this Thesis project originated from a NERC Oceans 2025 Work Package [28] that stipulated developments of the Autosub6000 AUV such that:

> *"For detailed 'close in' surveys ... we plan to enhance the slow speed manoeuvring (of the AUV)"*

and

> *"The AUVs control and manipulation capabilities will be developed to permit hovering and landing on the seabed"*

The Delphin 1 and 2 AUVs, the latter of which was developed as part of this thesis, were designed in direct response to these two quoted sentences. Although these vehicles are capable of hovering, slow and high speed manoeuvring, their control has proven difficult [21]. The (deliberate) landing on the seabed has yet to be investigated.

1.4 Research Aims

The aim of this thesis is to develop a suitable control algorithm that can provide stable and agile control of a hover-capable torpedo shaped AUV from zero to high speed operation. This can be broken down into four main objectives:

- Development of a reliable hover-capable AUV whose actuators are suitable for operation throughout a wide range of surge velocities. Along with the ability to test different control algorithms, specific to this work, the vehicle must provide the flexibility for future projects investigating AUV related artificial intelligence, hydrodynamics and electronics.

- The investigation into how to develop the necessary controllers for this AUV can be split into three further objectives:

  - Design and evaluation of a depth and pitch controller for thruster actuated hover.
  - Design and evaluation of a depth and surge velocity controller for the operation of an AUV operating in flight-style mode.

  - Design and evaluation of a depth and surge velocity controller for the operation of an AUV, using all available actuators, to provide stable control from hover to the vehicle maximum surge velocity.

This work makes contributions to the field of AUV design and control, specifically:

- Details on the design of a hover-capable AUV addressing:
  - Its mechanical, electrical and software architecture.
  - Derivation of a mathematical model, from experimental data, that captures the key dynamics of the vehicle.
  - The identification of the non-linear and uncertain performance of tunnel thrusters on an AUV operating near the free surface.
- Developing linearisation methods to produce linear state-space models that accurately approximate the AUV dynamics and is suitable for use with the model predictive control algorithm. Reasons for choosing these linearisation methods and their limitations are given.
- The choice of input constraints that are specific to a hover-capable AUV are discussed and implemented. These constraints address physical and dynamic restrictions of the actuators but also aim to help avoid actuator non-linearities.
- All of the control algorithms developed in this work are experimentally verified. By doing this, many 'real-world' effects on the controller performance are identified.

1.5 Structure of Thesis

In Chapter 2 a review of the latest AUVs is undertaken to understand what vehicle configurations are suitable for different tasks. Further to this, a review of the most common low-level control algorithms used onboard different AUVs is performed and their performance discussed. The model predictive control algorithm is also investigated and its suitability for use onboard an AUV discussed.

In this Thesis the suitability of using a model predictive control (MPC) algorithm to control an AUV is evaluated. In Chapter 3 the MPC algorithm is derived along with the various tools required for its implementation. In order to evaluate the controllers' performance experimentally an AUV is required. This is introduced in Chapter 4 with its various mechanical, electronic and software systems discussed.

At the heart of the model predictive control algorithm is a mathematical model of the system to be controlled. Before producing the models used within the controllers, a non-linear model that captures the key dynamics of the vehicle and its actuators is derived in Chapter 5. The data used to create this mathematical model is found from new and existing experimental data.

To evaluate the general performance of the MPC algorithm, a relatively simple control problem is posed in Chapter 6. Here a depth and pitch controller is developed using the MPC algorithm and tested in simulation. By performing a large number of simulations the effects of the different tuning parameters are evaluated and recommendations given. A second series of simulations is also conducted to evaluate the degradation of controller performance when errors are deliberately introduced within the model used by the controller. Before implementing and testing the controller experimentally, the effect of measurement noise was evaluated in simulation. An observer is then designed and implemented to provide full state-feedback before experimentally testing the controller.

In Chapter 7 a more complex control problem is addressed; the depth and surge control of the Delphin2 AUV operating in flight-style mode. To handle the dominant non-linearities when operating in flight-style mode, a time-variant MPC (TV-MPC) algorithm is proposed. In this chapter, the TV-MPC algorithm is introduced and a suitable time-variant state-space model derived. Simulations were performed to tune the controller and implement further modifications to the system model before experimentally testing the controller using the Delphin2 AUV.

The final control problem is posed in Chapter 8. This involves the design of a depth and surge velocity controller that can operate, using the full actuator set, from hover to high speed. Along with the strong system non-linearities, challenges with control authority are discussed and a solution provided. The controller was tuned iteratively within simulation before being tested experimentally.

Finally, in Chapter 9, the conclusions and contribution from this work are summarised. This is broken into two sections, first addressing the performance and current state of the Delphin2 AUV, before going on the discuss the various contributions made to the field of AUV control.

Chapter 2

Control of Unmanned Underwater Vehicles

2.1 Introduction

As discussed in the introductory chapter, this work is focused on the development of a suitable control system for an AUV for operations from hover to high speed. In this chapter a literature review will be performed. First a thorough introduction to ROVs and AUVs will be given before discussing various intervention AUVs (IAUVs) that have been built over the last twenty-five years. Emphasis on the review of the IAUVs is given to the actuators fitted to these vehicles and their performance. The second half of the chapter is focused on control systems literature and draws conclusions for the various algorithms before identifying gaps that should be investigated.

2.2 Unmanned Underwater Vehicles (UUV)

Unmanned underwater vehicles is a generic term encompassing all vehicles, autonomous or remotely controlled, that operate under the sea surface and without humans physically present with the vehicles. A brief summary of the applications, advantages and disadvantages of ROVs and AUVs is undertaken in this section, before introducing IAUVs and critiquing several different vehicle designs.

2.2.1 Remotely Operated Vehicles (ROVs)

Remotely operated vehicles (ROVs), e.g. Figure 2.1, are the most common type of underwater vehicle used for complex interactions with the subsea environment. Such ROV tasks, along with many others, include:

- riser inspection and repair,
- pipeline or submarine cable installation or repair,
- close range scientific inspection and sampling,
- inspection within confined or hazardous areas such as within nuclear facilities,
- dam wall inspections.

ROVs are used for missions that require the ability to get close to, and often interact with, a subsea object or area [4] [5] [6] [7]. What makes ROVs suitable for these tasks can be attributed to one important design feature common to all ROVs, they are directly controlled by a human operator. The ability to control these vehicles, whilst receiving real-time feedback, is provided by a physical umbilical cable that comprises a data link (typically fibre-optic) and often an electrical power supply. There are some examples of wireless ROVs, see [29] [30], however they are not as common nor as capable as those that use a physical umbilical.

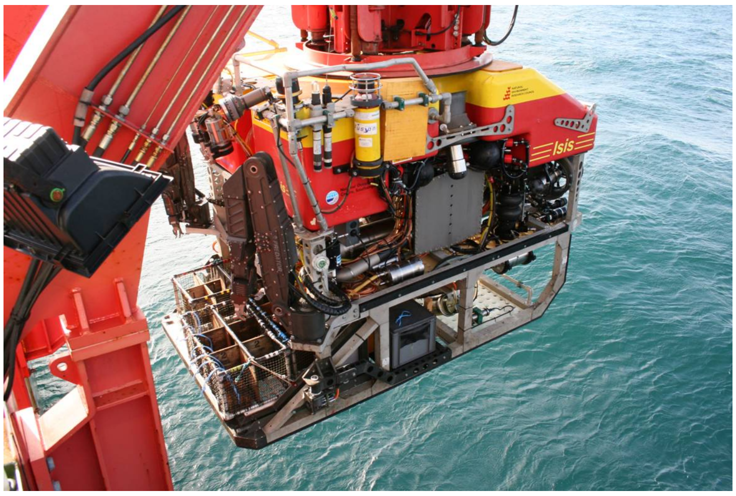


Figure 2.1: Launching of ROV Isis from RRS James Cook. Note the two manipulator arms in the folded position and the various lights, cameras and thrusters.

This high-bandwidth data link provides feedback to the ROV pilot who is normally operating the ROV from a surface vessel. The type of feedback available to the pilot depends on the specific ROV but video, sonar and telemetry data are the most common [31]. Not only can the pilot control the position and speed of the ROV but, if equipped, they can use actuator arms and tools to interact with the subsea environment. The autonomy of ROVs is generally low, with the pilots providing any necessary decision making. There are however some algorithms on the larger, more sophisticated, ROVs that automate many of the actuators and vehicle movements therefore reducing the complexity of controlling the vehicle [32]. This lack of automation increases the skill-set required of an ROV pilot and in doing so increases the likelihood of a human-induced error [33].

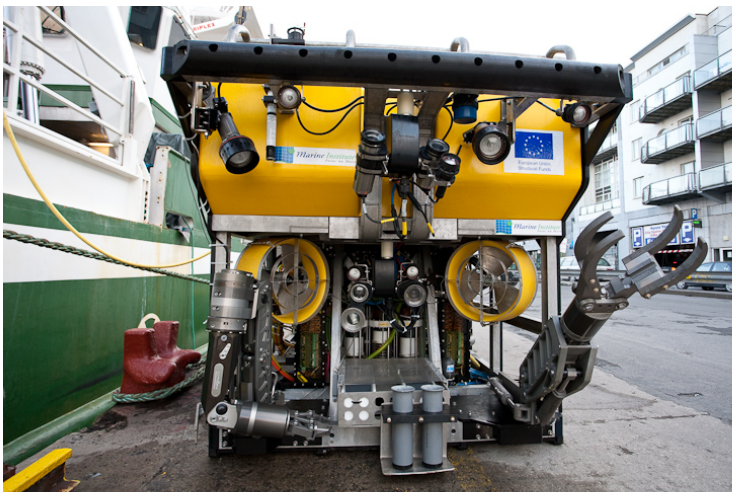


Figure 2.2: Holland I Deepwater ROV beside the RV Celtic Explorer in Galway harbour. Note the two large thrusters, that are used to manoeuvre the vehicle, and the extended manipulator arm.

The same design feature that provides ROVs with its ability to perform complex tasks, the umbilical cable, also severely constrains ROV operations. Physically, the length of the umbilical cord is limited and so the distance between the ROV and operator is constrained to the length of the cable. The dynamics of the cable, along with its interaction with the surface vessel and surrounding environment, also limit the length of the umbilical [8] [9] [10]. For example, if there is a strong tidal current then the cable will experience a force due to hydrodynamic drag which in turn will disturb the ROV. Due in part to an ROVs limited range, and the additional benefit of receiving electrical power through the umbilical, the hydrodynamic efficiency of an ROV hull-form is not a priority for their designers and so these vehicles are typically of a cuboid shape.

The majority of ROVs use a number of thrusters in order to manoeuvre the vehicle, Figure 2.2. The thrusters are fitted at various locations and angles around the vehicle in order to provide controllability of several degrees of freedom. The appropriate sizing of the thrusters for each ROV design is important. If the thrusters are too small then they cannot produce adequate thrust in order to manoeuvre the ROV quickly or in a strong current. Care must also be made to ensure that the thrusters are not too large as the thruster dynamics can then dominate the system dynamics making the ROV difficult to control [34]. This effect from the thruster dynamics can cause limit cycling due to the dead-band associated with the thruster motors.

ROVs are normally configured to be near neutral or positively buoyant. Setting the vehicle up as neutrally buoyant reduces power consumption from the thrusters. Choosing to configure the vehicle to have a positive buoyancy has two significant advantages for ROVs; the vertical thrusters must be continuously run, in order to counter the buoyancy force, and so the thruster dead-band is avoided, and the direction of flow out of the thrusters is always towards the surface therefore reducing disturbance of the sea floor.

Other than the physical limitations, there are significant financial constraints when using an ROV. When operating an ROV offshore a suitable surface vessel, typically a ship, is required that has the ability to both launch and recover the ROV as well as perform reasonable station keeping. The size of such a vessel is dependant on the size of the ROV and distance offshore, but can typically cost more than £30k per day to operate (including crew). To add to this, if the operators want to minimise ship costs, by maximising the ROV time in the water, then more than one pilot and engineering team would be required in order to to run 24/7. The ships' running cost means that operating an ROV offshore can prove extremely expensive.

2.2.2 Autonomous Underwater Vehicles (AUVs)

The second category of UUVs is autonomous underwater vehicles (AUVs). Unlike ROVs, described in the previous section, AUVs have little, if any, direct human interaction once the mission has commenced. Therefore, AUVs rely upon on-board software to utilise the available electronics, sensors and actuators to navigate the vehicle and perform the predefined mission. Such missions include:

- Oceanographic surveys such as collecting water samples [35], chemical or pollution monitoring, and collecting bathymetry data [14].
- Military operations such as mine counter-measures (MCM) [36] [37], submarine monitoring [38] or port surveillance [39].
- Commercial tasks, typically oilfield related, such as pipeline [40] [41] or platform inspections [42].

AUVs are normally of a classical torpedo shape with a rear propeller used to provide forward thrust and three or four control surfaces (fins) at the rear of the vehicle to manoeuvre it, see Figure 2.3. If suitably designed, this configuration of vehicle should result in a low hydrodynamic drag resistance thus enabling it to transit long distances at relatively high speeds. One significant disadvantage of this configuration is that the AUV is required to operate above a critical speed, below which the vehicle becomes unstable, this is known as the Chinese effect [16]. The reason this effect occurs is that the forces and moments generated by the control surfaces, used to manoeuvre the vehicle, are a function of the vehicles' forward velocity relative to the fluid. Further non-linearities, such as stall, increase the complexity required to predict the speed at which the Chinese effect is going to occur. The velocity at which the Chinese effect occurs can be reduced by utilising additional control surfaces mounted at the front of the AUV [43] or by configuring the vehicle to be neutrally buoyant. Despite these efforts the AUV will still become unstable when the vehicle has a zero forward speed relative to the fluid. This inability to perform very low speed operations (< 0.5 m/s) reduces the ability of a typical AUV to perform close range detailed inspections, and completely rules out performing direct interactions with subsea objects.

The lack of direct human interaction and decision making reduces the complexity of the tasks undertaken by AUVs. Typically, AUVs effectively act as underwater buses travelling to various locations carrying a suite of sensors that can be switched on or off to log information to an on-board hard-drive. As the artificial intelligence of AUVs continues to increase, through better data processing and on-board analysis, the complexity of missions being performed also increases. Despite this, AUVs are unlikely to be tasked with the same high complexity missions that ROVs are currently tasked with performing.

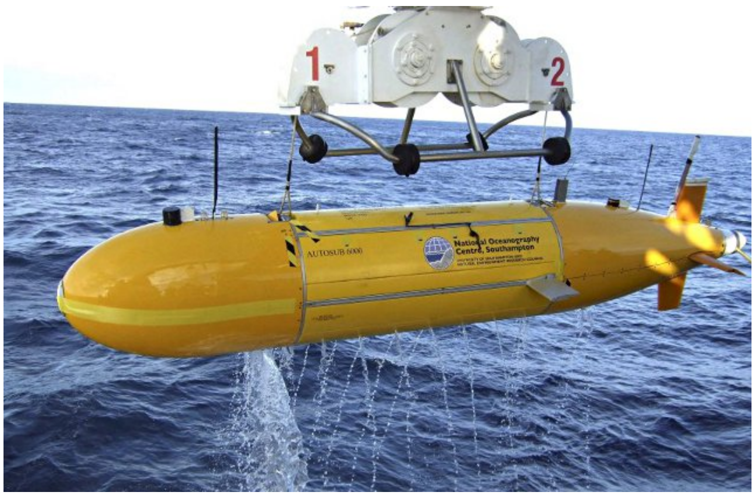


Figure 2.3: Autosub6000 being recovered onto the RRS James Cook. Note the location of four control surfaces and a propeller at the rear of the vehicle.

2.2.3 Intervention Autonomous Underwater Vehicles (IAUVs)

As with all technology, there is a push to develop more adaptable systems that are capable of performing multiple tasks on the one platform, for example, the evolution of mobile telephones with camera and email functionality. This is quickly becoming the case for autonomous vehicles, most notably unmanned aerial vehicles (UAVs) that can typically perform a variety of tasks such as surveillance, search and destroy, and logistics, using the one vehicle. In this section a subsection of AUVs, described here as intervention AUVs (IAUVs), are described. These are tasked with performing missions both common to standard AUVs, as described earlier, and also some of the less complex tasks performed by ROVs.

As alluded to in the previous section, typical AUVs are limited by their inability to accurately process sensor data so as to 'see' and 'interact' with the subsea environment. In order to develop such capabilities, further research into improving sensor quality, integration, and processing is required along with adequate artificial intelligence to make use of the available information. This vast field of study is far beyond the scope of this work, however, if a bold assumption is made that this technology will someday become available then there still remains the challenge of developing a suitable vehicle that can physically perform the range of tasks that would then be achievable. Specifically, IAUVs have been designed to operate throughout a wide range of speeds, often from 0 m/s up to 2.0 m/s or higher.

A range of IAUVs have been produced since the late 1980s with various configurations and degrees of success. Here a few of these vehicles will be examined in order to aid in the design process of a hover-capable AUV that will be developed later in this work. The Odyssey IV AUV, see Figure 2.4, developed by the AUV Lab at MIT Sea Grant [44], uses four thrusters to manoeuvre the vehicle. One

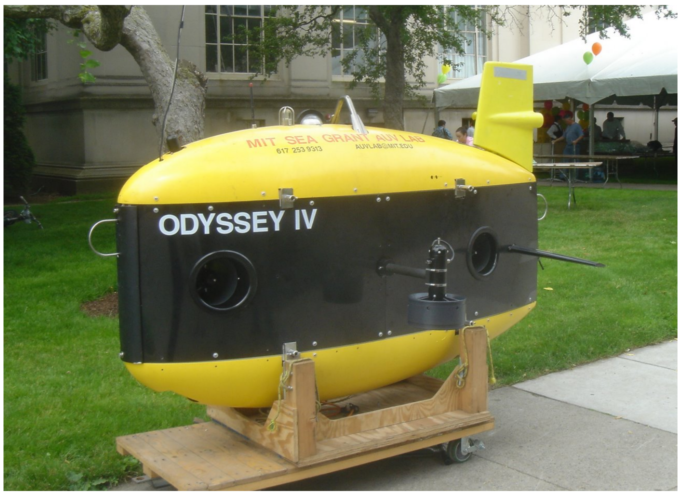


Figure 2.4: Odyssey IV AUV. This vehicle has the ability to hover and transit at up to 2.0 m/s. Picture courtesy of MIT.

through-body tunnel thruster is fixed at the bow and one at the stern of the vehicle; these are used for yaw and sway control. The other two thrusters are fixed to rotatable mounts at either side of the vehicle; these are used for surge and heave control and during survey speeds (>1.0 m/s) can also be used for yaw control. The original design did not incorporate any fins, however after field tests it was discovered that the vehicle was unstable in both pitch and yaw axes and so the large fixed angle stabilising fins were added. Two problems were noted with the use of the horizontal through-body tunnel thrusters used for yaw control. Firstly, the dead-band of the thruster motors resulted in difficulties accurately controlling the heading. This was due to sudden and significant bursts of thrust as the thruster start to spin thus causing the heading to overshoot its set-point. A second problem was noted as the thruster effectiveness was noted to drop significantly as the surge velocity of the vehicle increased.

The C-Scout AUV [20] was developed as a collaborative project between the Institute for Marine Dynamics (IMD), of the National Research Council of Canada (NRC), and the Ocean Engineering Research Centre (OERC) at the Memorial University of Newfoundland (MUN). This vehicle in its fully actuated configuration has four control surfaces at the bow and four control surfaces at the stern used for controlling five degrees of freedom (all but surge) whilst at speed, two vertical through-body tunnel thrusters, one at the bow and another at the stern, used for heave and pitch control, and two horizontal through-body tunnel thrusters, one at the bow and another at the stern, used for controlling yaw and sway, and finally a rear propeller enables surge control. This extensive array of actuators enables the vehicle to manoeuvre in all six degrees of freedom (DOF) throughout a speed range of less than 0 m/s up to 3 m/s. There is not a lot of published information on open-water testing of the C-Scout AUV therefore it is difficult to deduce how effective the actuator set performed with the low-level control algorithms. There was however extensive hydrodynamic testing of this AUV in a towing tank. From these tests it was noted that the effectiveness of the through-body tunnel thrusters was a complex function of the speed and angle of the vehicle's motion relative to the water with a general trend of reduced thruster efficiency at higher surge velocities. An undesirable trait of the C-Scout design is the number of actuators, this will not only add complexity to the low-level controller design but also significantly increase the manufacturing costs of this vehicle.

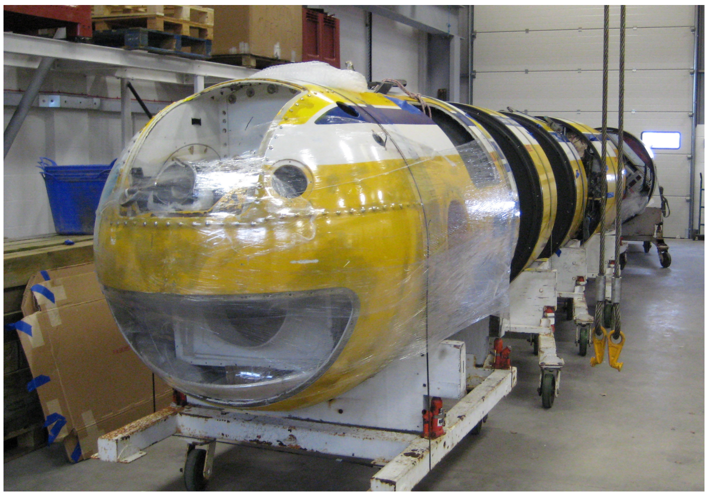


Figure 2.5: The MUST AUV arriving, in sections, at the M-Subs factory in Plymouth, UK, 2012.

The MUST AUV, Figure 2.5, developed by Martin Marietta Aero (now part of Lockheed Martin) and Naval Systems, is an early attempt at designing a vehicle to operate through a wide range of speeds. This AUV is considerably larger than its modern equivalents at over 9 metres long and 1.4 metres in diameter. To manoeuvre, it has four through-body tunnel thrusters, with one vertical and one horizontal at both the bow and the stern of the vehicle, and four control surfaces and a propeller at the stern of the vehicle. This actuator configuration enables control of five DOF throughout its entire speed range of 0 to 4 m/s. The first publication on this AUV describes the physical design of the vehicle and its subsystems, [18]. Along with the huge size of the vehicle, it is also extremely heavy (stated as 1100 kg in [18] although this is thought to be a mistake as similarly sized UUVs are much heavier), therefore to accelerate the vehicle the rear propeller and tunnel thrusters are driven using a 7.5 kW and four 3 kW electric motors. This is a high amount of electrical power, and coupled with the inferior battery technology of its time, the estimated range of the MUST AUV was only 80 km. Later work with the MUST AUV describes the low-level control systems used to manoeuvre the vehicle, [45]. This publication is the first known to describe the problems associated with controlling an AUV from low speed hover, using the tunnel thrusters, to high speed flight using the rear control surfaces. In a later paper, [46], sea trials of the MUST AUV are presented, however, due to publication restrictions many of the figure axes do not have scales so it is difficult to draw conclusions about the effectiveness of this AUV design.

One of the most well known and iconic hover capable AUVs is the Autonomous Benthic Explorer

(ABE) developed by the Woods Hole Oceanographic Institution (WHOI) [47]. This vehicle differs in design from the others described here as it has three main 'bodies,' two large flotation pods at the top and one instrument and system body at the bottom. This configuration enables the vehicle to carry large payloads with little adjustment to its physical appearance. Also, maintaining the centre of gravity near to the bottom of the vehicle, with the centre of buoyancy towards the top, results in a strong restoring moment that keeps the vehicle stable in pitch.

The vehicle utilises two vertical and two horizontal thrusters when operating in hover mode, it is believed that this thruster configuration should provide good low speed control but that the thruster effectiveness is likely to degrade substantially. Three thrusters mounted at the rear of the vehicle provide forward propulsion. A compromise was made in the design as the centre of thrust, from the rear thrusters, is not aligned with the centre of gravity and so an undesirable pitching moment is generated when transiting forward. This can cause the vehicle to deviate from level pitch and so generate hydrodynamic lift from the hull that can disturb the control of depth. To compensate for this the vertical thrusters can be user to restore level pitch, but this is very inefficient at higher speeds. This problem is not significant on torpedo shaped AUVs.

The Sentry AUV, also developed by WHOI, is a single hull hover capable AUV that utilises four thrusters mounted at the fore and aft of the vehicles sides. These thrusters are mounted on small foils that can adjust their angle relative to the body, similar to the Odyssey IV AUV. By changing the angle and speed of the thrusters the depth, pitch and heading of the vehicle can be accurately controlled. This design should provide highly efficient operation at both low and high speeds although the dynamics do become more nonlinear that may present problems designing a suitable control algorithm. One downside to this design is the inability to control sway. For the Sentry AUV this is not a significant problem but for an inspection type AUV the ability to sway is advantageous.

In more recent times there has been a sudden rise in the effort, significantly from commercial companies, to develop hover-capable AUVs. Such examples include adaptations of the Remus 100 and 600, [48] and [49], as well as bespoke designs such as the Seacat from Atlas Elektronik, [19]. Papers have been published on the hover capable Remus vehicles, but no interesting information was divulged as it is rare for a commercial company to openly describe their products in great detail nor divulge product weaknesses. All three of these vehicles share the same actuator configuration as the MUST vehicle, with four through-body tunnel thrusters, four control surfaces and a propeller at the rear. It is expected that the popularity of this design is due to two factors; keeping the common torpedo shape of a typical AUV with the thrusters mounted within the main hull maintains the vehicle's low drag resistance, and keeping the thrusters internal reduces the number of components that could be damaged in the event of a collision.

2.2.4 Discussion

This work is focused on the development of an AUV, and its appropriate control systems. It aims to combine the capabilities of ROVs and AUVs, like the IAUVs described in the previous section.

In order for these vehicles to operate at low surge velocities, additional actuators to those normally found on AUVs are required. These can be generalised within two common configurations; those with externally fitted thrusters whose angles can be adjusted so as to vary the their respective force vectors, and those with thrusters fitted within the main AUV hull. Both designs obviously have various advantages and disadvantages.

The former design may result in a vehicle that is more manoeuvrable at low surge velocities however, the addition of externally mounted actuators has two detrimental effects; the hydrodynamic drag resistance of the vehicle will likely be relatively high, and the possibility of actuator damage or entanglement is increased. The thrusters that have the ability to change their angle, with respect to the AUV, require additional actuation thus increasing the complexity of each actuator and their associated electronics resulting in increased manufacturing costs and risk of failure.

The latter actuator configuration should provide a low drag resistance, similar to that of a typical torpedo shaped AUV (though slightly higher due to the addition of the thruster tunnels). This therefore should help these vehicles maintain the long range endurance that should be a priority for these future vehicle designs. One of the noticeable trends from the literature has been the non-linear and uncertain performance of the through-body tunnel thrusters as a function of the vehicle's surge velocity. Therefore the full actuator set, including any control surfaces and propellers, needs to be capable of controlling the vehicle when the thrusters become ineffective.

2.3 UUV Low-Level Control Systems

All operational AUVs, and most ROVs, use closed-loop low-level control systems to manoeuvre the vehicle within its environment. These controllers are used to calculate suitable actuator settings so as to reduce the error between the current and desired state of the vehicle. The term closed-loop refers to the use of measured system outputs being fed back into a control algorithm as feedback so as to ensure that any control errors are minimised.

There has been a substantial amount of research on the topic of low-level control systems for UUVs since the 1980s. In this section a review of control algorithms is given. These algorithms have been chosen for review due to their historical successes at controlling UUVs or their possible suitability for the control of a hover-capable AUV. Note that many other algorithms have been shown to successfully control UUVs, such as in [50] [51] [52], however only classical control, sliding mode control and model predictive control will be examined here.

It is common for UUVs to be controlled using three independent controllers for depth, heading and speed control. Despite the strong coupling between system states of UUVs, the decoupling of the three different controllers has proven successful for different vehicles and control algorithms. This is partly due to each system being fully actuated and that disturbances to the other states during transience is acceptable. Note that the control system to be developed in this work must be capable of controlling an AUV throughout a wide range of velocities with a large set of actuators, therefore the controller must be able to handle system non-linearities and uncertainties, along with multiple inputs and multiple outputs (MIMO).

Classical Control

The 'first port of call' for most control engineers is to use classical control methods, more specifically the proportional-integral-derivative (PID) algorithm. The reasoning for this is that the algorithm is very easy to implement and there is a vast number of linear techniques that can be used for designing these algorithms. The simplicity of the algorithm means that it is also computationally inexpensive and so can be implemented on low-power electronics typical of AUVs.

The PID algorithm comprises of three main parts; the proportional (P), integral (I) and derivative (D) components. The simplest implementation of this algorithm is just the P term, where the controller output is proportional to the control error and a predefined scalar gain. The P controller can provide adequate levels of control for simple systems however one undesirable trait of these controllers is the steady-state error. This means that, for many systems but not all, a small control error will always exist when using just the P component.

The addition of the integral component helps eliminate the steady-state error by summing the product of a scalar gain and integrated control error (with respect to time) with the P component. When designed correctly a PI controller can work very well though can result in overshooting the desired set-points due to integral wind-up. The third component, the D term, is the time derivative of the error multiplied by a scalar gain. This is then summed with the P or PI terms and provides a means to increase the overall system gain, leading to faster convergence to zero control error, whilst reducing overshoot. The D term can prove difficult to implement experimentally as calculating the derivative of a measured signal can result in a low signal to noise ratio and consequently cause the system to jitter.

For flight-style (torpedo shaped) AUVs, it is common to use two cascaded control loops in order to control the vehicle's depth. The first loop is a function of the depth error and outputs a desired pitch angle, and the second loop then attempts to reduce the pitch error. One fundamental requirement of this design is that the depth controller is slower than the pitch controller otherwise performance is likely to degrade.

Simulated and experimental results using a PID controller can be found in [53]. Here the depth controller uses the PI algorithm while the pitch controller uses the full PID algorithm. The reason for not implementing the D term for depth control is that it is not directly measured therefore direct differentiation of the depth signal would be required and this was deemed not to be suitable. The D term within the pitch controller was directly measured using an on-board rate gyroscope sensor. The experimental results given for the depth controller show very poor performance with depth failing to converge to its set-point. Instead, the vehicle oscillates in depth set-point with an amplitude of $\pm$ 3.0 m. This oscillation is due to the inclusion of the I term within the outer depth control loop and could be reduced to within acceptable limits by reducing the I gain scalar value. It is thought that with further effort to tune the PID gains then the system performance could be improved.

In [54], experimental results of the Autosub 1 AUV using the cascaded controller are presented. This controller differs from that previously discussed as the outer depth control loop comprises of just a P term, and the inner pitch control loop comprises of the PD terms. There are no integral terms within this controller and as a result, the vehicle shows much more stable performance when at depth than in [53] however the vehicle fails to accurately converge to the depth set-point.

Results from two dives from the surface to 10 metres are presented. On the first dive the vehicle overshoots the set-point by approximately 3 metres before reaching a steady-state error of approximately 1.1 metres. On the second dive, the performance is better with a much reduced overshoot and a smaller steady-state error. It is not stated in the paper what changed between the two dives but upon asking the author he disclosed that it was likely that the tuning parameters were changed but not recorded. Another design feature of this specific controller is that the pitch angle demand, from the outer depth control loop, is limited to within safe thresholds. This should ensure that the controller performs with similarly good performance irrespective of the size of the step-change in depth set-point.

Later work with the Autosub6000 uses a PD-PD cascaded controller [55]. In this work the performance of altering the tuning parameters are evaluated by challenging the AUV to perform step-changes in depth demand with three different propeller power set-points. The highest power set-point gave the best controller performance with steady-state errors of less than 1 metre, however the lower powered cases converged much slower and had larger steady-state errors. This work identifies one of the key weaknesses of PID algorithms for controlling AUVs; they can be successfully tuned to work for one specific speed however their flexibility to operate over a wide range of speeds is limited. To enable the controller gains to vary with the system state, called gain-scheduling, requires additional processing and further tuning effort therefore reducing the original desirable qualities of the algorithm.

Experienced control engineers often tune PID controllers using a combination of linear control theory and trial and error, however, there has been an emerging number of algorithms in recent years that provide automatic tuning, for an AUV related example see [56]. Although good simulation results have been produced it is believed by the author that the UUV community may be reluctant to adopt adaptive controllers on their vehicles as the controller may fail to converge to stable controller gains and the vehicle could be lost.

As an overview, for simple vehicles where the precision in tracking certain set-points is not deemed a priority, and for cases where computational resources are very low, then the PID algorithm has been proven to be a suitable control algorithm. However, when operations require flexibility and precision then the suitability of this algorithm becomes less valid.

Sliding Mode Control

Early in the emergence of AUVs the sliding mode control (SMC) algorithm became extremely popular and continues to be of interest to the AUV community. The key advantage of using SMC is that it can be used to force the vehicle along a particular trajectory using an effectively high gain switching control law. This control law helps remove the controller dependence on system non-linearities and uncertainties. One disadvantage with SMC is that it can cause actuator 'chatter,' [57]. This is a result of the switching process and high controller gain and can lead to excessive actuator wear and increased energy consumption. Attempts to reduce this chatter with the inclusion of the tanh function within the control law have been shown to be successful [58].

Thruster actuated AUVs that have been controlled using SMC include the NPS PHOENIX AUV [59] and the MUST AUV [45]. In [59], two controllers are developed using SMC to control depth and heading of the NPS PHOENIX AUV and experimental results are given. Both of these controllers calculated the desired future trajectory of the vehicle using fifth order zero jerk profiles to provide smooth transition to the depth or heading set-point. The experimental results show excellent performance in both depth and heading control however there is a slight undesirable depth overshoot. The time taken to achieve the depth set-point is also very slow at 60 seconds for a 1.0 m step change. It would be expected that a hover capable AUV should be able to manoeuvre more quickly however this may be a result of the choice of the vehicle trajectory rather than the actual SMC controller. No results of the system inputs are presented therefore concerns regarding 'chatter' cannot be evaluated.

Simulation results of controllers developed for a flight-style AUV are given in [58]. Again the controllers are designed for the separate system outputs, yet in this work the system is significantly more non-linear than in [59]. Despite this, all of the controllers appear to work very well, with very fast yet smooth convergence to their appropriate set-points. The system inputs appear smooth and without chatter. This is most likely due to the inclusion of the tanh function but may also be due to the smooth (noise free) feedback signals.

The MUST vehicle, introduced earlier in this Chapter, has an actuator configuration that enables operating at both zero and surge velocity, as required by the vehicle developed later in this work. In [45] the design of an SMC controller for controlling depth, heading and surge velocity (seperately), for both low and high speed operation is given. In this work the transition from thruster actuated to control surface actuated depth control is discussed. Due to the decreased efficiency in thruster performance as surge velocity increases, the author describes forcing the controller to utilise the contol surfaces as soon as possible and wind-back the use of the thrusters. This has been done by estimating the forces that the control surfaces are capable of and then if control error is low start to reduce the thruster values. Experimental results using these controllers are provided in [46] and generally show excellent performance at controlling all three outputs; depth, heading and surge velocity. In this paper, a Figure presents depth against time whilst accelerating in surge velocity from zero. There is a noticeable disturbance to the vehicle's depth but this is relatively low at 0.3 m. It is difficult to draw conclusions from the data presented as neither the corresponding surge velocity nor system inputs are presented.

In general, SMC is one of the most common control algorithms used for low-level control of AUVs. Assuming care is made to remove or reduce actuator chatter the SMC algorithm can be used to design robust, high performance controllers. Although research into the application of the SMC algorithm for AUV control is still ongoing, controller performance has not significantly improved since the mid-nineties.

Model Predictive Control

In comparison to SMC, neither model predictive control (MPC) or optimal control have been extensively studied for use in UUV applications. In this section a review of what has been done using MPC to control UUVs as well as other similar systems will be given.

The MPC algorithm can be categorised as a type of optimal control and was first developed in the 1960s and later applied in the 1980s for use in the process industries such as oil refining and large chemical plants. The MPC algorithm is now the de facto control algorithm for the process industries due to its ability to handle high dimensional MIMO systems whilst imposing input, output and state constraints within the optimisation solution.

The fundamental idea of MPC is to utilize a mathematical model of the system dynamics (that is to be controlled) so as to predict the future trajectory of the system for future control inputs. Using this ability, the optimal future system inputs can be found by solving a quadratic programming problem with inequality constraints. One of the key limitations of MPC, especially in the 1960s to 1980s, is the computational effort required to find the solution. In those times the computational capabilities were clearly much inferior to today's, however, this same problem has hindered the acceptance of MPC within the autonomous vehicle industry due to the preference to use low power computers. As computing power becomes more and more efficient this is less of an issue and implementations on unmanned aerial vehicles (UAVs), with modest computational resources, have been experimentally verified running at a relatively high sampling rate of 40 Hz [60]. Piezo actuator systems have been controlled using MPC at up to 10 kHz sampling rate, [61], although the computer was more powerful than that typically found in most AUVs.

In [62] a heading controller for the Hammerhead AUV was designed using the MPC algorithm, with constraints applied to the rate of change and absolute rudder angle. The simulated results show excellent system performance, with only a slight overshoot of the heading demand followed by smooth tracking. However, an issue of concern is that the same simulation results show the input signal to be constantly oscillating. This ‘chatter’ could both increase energy consumption and mechanical wear of the rudder actuators. It was noted that if one of the weighting scalars, from within the cost function, was increased then this oscillation could be reduced at the expense of slower convergence to the set-point.

The experimental results were not as good as the simulation results. This is not surprising as the simulations were performed using the same linear model as was implemented within the controller design. Despite this, the heading control did work and track the heading demand with an error of less than $\pm 20^o$. For operational AUVs this would be deemed unacceptable however the algorithm does show some promise as it did manage to compensate for the offset errors of the rudder as it was not properly zeroed. It should also be noted that the Hammerhead AUV was originally built in the 1960s and used the original electronics and actuators which lead to implementation problems due to poor vehicle condition. If this controller had been implemented on a more modern UUV it may have performed better.

One aspect of the controller design that was not fully justified was the use of a genetic algorithm (GA) to find the optimal solution of the quadratic cost function rather than the many different solvers specifically designed for solving quadratic programming problems with constraints. As a result of using a GA the controller operated at a very slow sampling rate which most likely hindered the overall controller performance.

In [63] simulation results were presented for the pitch control of the Squid AUV. This controller differed slightly from that of [62] as both input and state constraints were imposed. There are several factors of this paper that are unclear including, how the state constraints were imposed and why the values for the different constraints were chosen. The results are good, although again it is unclear why the operator would want to directly control the vehicle’s pitch rather than more useful states such as depth or surge velocity. The ability to handle state-constraints within the optimization process is highly advantageous but this work fails to fully explain how this is implemented.

These examples from the UUV community do not present the MPC algorithm as a viable algorithm for this work. However, work within other similar industries does present some interesting results. In [64] the MPC algorithm is used to control the attitude and altitude of a quad-rotor helicopter whilst imposing constraints on the absolute and rate of change values of the system inputs along with constraints on the system outputs. Both the simulated and experimental results show excellent performance despite the mathematical model being relatively simplistic. The performance of the altitude control was especially good which could be compared to to depth control of an AUV.

A similar study showed that the MPC algorithm could be used to optimally drive an electric car that was designed to compete in an 'Eco-marathon,' [65]. Again, input and state constraints were imposed whilst the controller was tasked with controlling the speed of the vehicle. In this work the controller successfully tracked the desired velocity trajectory for each part of the track.

In [60] a learning-based MPC algorithm (LBMPC) is proposed for controlling the position and attitude of a quadcopter UAV. Both a standard MPC and LBMPC controller showed excellent performance when tested experimentally, using similar computational resources to what would be available on-board a modern UUV. What is especially impressive is that the LBMPC controller is capable of improving the accuracy of the inner controller model and in turn the controller performance. There are some weaknesses with the LBMPC algorithm as it has been shown to occasionally do 'improper' learning, causing the vehicle to momentarily lose control. It is unlikely that LBMPC would be adopted by the UUV community for this reason however it may prove useful within UUV research.

The two papers using the MPC algorithm to control UUVs show some promise but generally have not thoroughly examined the algorithm to its full potential whilst the other two papers, controlling an autonomous aircraft [64] and car [65], have shown that the MPC algorithm provides excellent control characteristics if designed correctly.

MPC has two key advantages when compared to many other algorithms; the ability to 'look-ahead' should lead to smooth and optimal control, and the ability to find an optimal solution within constraints both increases the validity of the solution and also maintains the system within safe limits. The latter point is particularly suitable for UUVs. Firstly, all mechanical actuators have limitations that can be handled using constraints. Secondly, due to the wide range of operating conditions of this AUV, the performance of the various actuators is likely to vary significantly. Without the use of effective constraints then the controller could 'wind-up' to unrealistic actuator set-points leading to control degradation.

2.4 Conclusions

From the literature discussed in this Chapter it may be concluded that SMC has proven to show excellent performance for a wide variety of vehicles. It has also been explicitly used to control a hover-capable AUV, unlike MPC or classical methods, and so would appear the logical solution for controlling any hover capable AUV developed in this work.

However, the use of MPC for UUVs has yet to be thoroughly examined as the previous work had several limitations. The ability to design a MIMO controller with both input and state constraints is highly desirable. The implementation challenge of achieving a suitably high sampling rate with modest computing power is no longer a hard constraint for UUV engineers.

Chapter 3

Model Predictive Control (MPC)

3.1 Introduction

First variations of model predictive control (MPC) were developed in the 1980s for the process industry. Since then many different MPC algorithms and related tools have been produced, however MPC is still a major topic of research for control systems theory and application. For a comprehensive overview of existing MPC algorithms please refer to [66] and the references within. This research focuses on the specific details necessary to implement an MPC algorithm in order to control an AUV that is required to operate within constraints. Attention is therefore restricted to one MPC algorithm class and the contribution is in detailed design supported by an experimental programme.

In this chapter the MPC algorithm, that is used throughout this thesis, will be introduced along with the necessary tools required for its implementation. The idea of MPC is to utilize a mathematical model of the system that is to be controlled, so that the controller can find the optimal future change of system inputs (control horizon) for a future prediction horizon, see Figure 3.1. The system input, that is calculated by the controller, is the previous system input plus the first control set of the optimal solution, therefore the controller output is the integral of the optimal solutions. This integral action is a key attribute of the MPC algorithm and helps compensate for model inaccuracies and unmodelled disturbances.

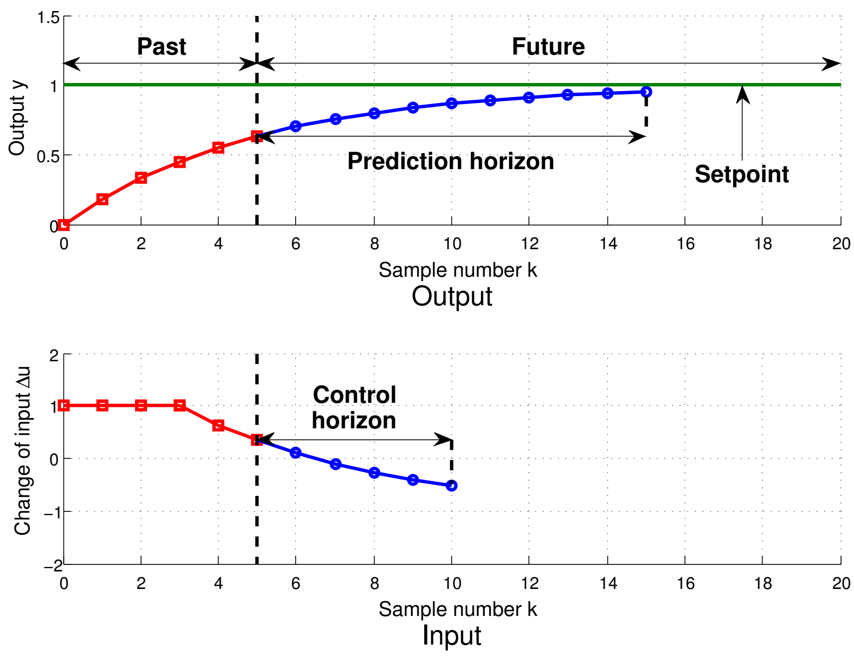


Figure 3.1: Example of the prediction and control horizons calculated by the MPC algorithm.

3.2 MPC Algorithm

The model predictive control algorithm can be described as two main parts; initialisation and the main control loop. An overview of the processes within the MPC algorithm are illustrated in Figure 3.2.

Initialisation

First a continuous state-space model of the system is developed and then discretized using the sampling rate that the controller will operate at. This is then transformed into the augmented model that is used to predict the future trajectory of the system outputs, Y, by using the current state-variable vector and the future change of control values, ΔU. A quadratic cost function is defined using the augmented model and is a function of the error between the controller set-point and future system trajectory, and the magnitude of the future change of control inputs ΔU. The final initialisation process includes defining any constraints that are to be adhered to and any controller tuning parameters. Both the constraints and tuning parameters will be discussed later in the chapter.

Main Control Loop

The purpose of the main control loop is to find the optimal change of N_c future control inputs, ΔU, for a given prediction horizon of N_p samples. The first value of the ΔU vector is added to the previous control input value, u, then applied to the system.

The optimal solution is found by minimizing the cost function defined in the initialisation stage using quadratic programming methods. If this optimal solution violates any of the constraints then the Hildreth programming procedure (HPP) [67] is used to find a correction term that enables the calculation of an optimal solution within the constraints.

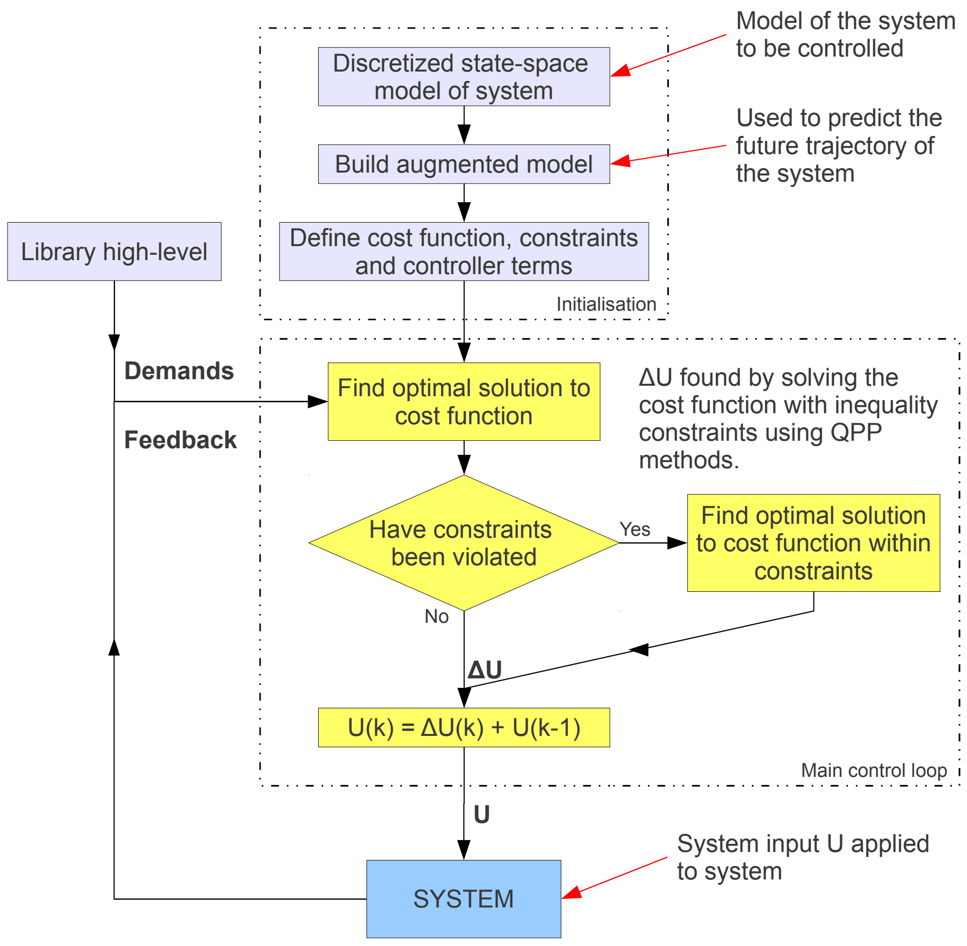

Model of the system
to be controlled
Discretized state-space model of system
Build augmented model
Used to predict the future trajectory of the system
Library high-level
Define cost function, constraints and controller terms
Initialisation
Demands
Feedback
Find optimal solution to cost function
ΔU found by solving the cost function with inequality constraints using QPP methods.
Have constraints been violated
Yes
No
Find optimal solution to cost function within constraints
ΔU
U(k) = ΔU(k) + U(k-1)
Main control loop
U
SYSTEM
System input U applied to system

Figure 3.2: Flow diagram of information into and out of the model predictive controller in the Delphin2 AUV.

This work is focused on the implementation of the model predictive control algorithm on the Delphin2 AUV. Although this does involve some adaptation of the algorithm, the full derivation of the algorithm presented here can be found in [68]. In this section the derivation of the algorithm and the tools used to solve the optimization problem will be presented.

3.2.1 Augmented Model

A linear system can be modelled using a linear model with m inputs, q outputs and n_1 states:

$$\dot{x_m}(t) = A_m x_m(t) + B_m u(t), \tag{3.1}$$

$$y(t) = C_m x_m(t), \tag{3.2}$$

where u and y are the system input and output respectively and x_m is the state variable vector. It is assumed that $q \leq m$, otherwise it is unlikely that each output can be controlled without steady-state errors.

In order for the MPC algorithm to predict the future trajectory of the system, the system output can only be a function of the current state variable vector and not the input vector. This is the case with all inertia type mechanical systems; the input cannot have an instantaneous effect on the system output and therefore the D matrix, commonly found within state-space literature, is taken to equal zero.

The MPC algorithm used here embeds an integrator into the model, enabling the controller to deal with model inaccuracies. This is done by changing the system input from u to Δu by taking the difference between the current and previous state and input vectors:

$$\Delta x_m(k+1) = x_m(k+1) - x_m(k);\ \Delta x_m(k) = x_m(k) - x_m(k-1), \tag{3.3}$$

$$\Delta u(k) = u(k) - u(k-1), \tag{3.4}$$

using equations (3.3) and (3.4) the state-space model becomes:

$$\Delta x_m(k+1) = A_m \Delta x_m(k) + B_m \Delta u(k), \tag{3.5}$$

By connecting $\Delta x_m(k)$ to the output $y(k)$, a new state variable vector is created:

$$x(k) = \left[\Delta x_m(k)^T y(k)\right]^T, \tag{3.6}$$

Note that:

$$\begin{aligned} y(k+1) - y(k) &= C_m(x_m(k+1) - x_m(k)) = C_m \Delta x_m(k+1) && (3.7)\\ &= C_m A_m \Delta x_m(k) + C_m B_m \Delta u(k). && (3.8) \end{aligned}$$

Combining equations (3.5), (3.6) and (3.7) results in the state-space model used within the MPC controller:

$$\begin{aligned} \begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix} &= \overbrace{\begin{bmatrix} A_m & o_m^T \\ C_m A_m & I_{q\times q} \end{bmatrix}}^{\text{A}} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} + \overbrace{\begin{bmatrix} B_m \\ C_m B_m \end{bmatrix}}^{\text{B}} \Delta u(k) && (3.9)\\ y(k) &= \overbrace{\begin{bmatrix} 0 & I \end{bmatrix}}^{\text{C}} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}, && (3.10) \end{aligned}$$

where in this thesis 0 and I denote the zero and identity matrices, respectively, with compatible dimensions.

3.2.2 Prediction of the System Trajectory

In order to create a controller that utilises the system dynamics, first the ability to predict the future state for a given system input is created. The future state is calculated for a prediction horizon of

N_p samples, for a future control trajectory of N_c samples. N_c must be less than or equal to N_p. The prediction will be denoted here as starting from sample number k_i where $k_i > 0$.

The future control trajectory N_c can be denoted as:

$$\Delta u(k_i), \Delta u(k_i + 1), ..., \Delta u(k_i + N_c - 1), \tag{3.11}$$

The prediction horizon N_p is the future state variables for the system and can be denoted as:

$$x(k_i + 1|k_i), x(k_i + 2|k_i), ..., x(k_i + m|k_i), ..., x(k_i + N_p|k_i), \tag{3.12}$$

where x represents the state variable vector at each sampling instant starting with the current plant information (measured or observed) at sample k_i. Using the augmented state space model from equation (3.9) the future state vectors can be calculated:

$$\begin{aligned}
x(k_i + 1|k_i) &= Ax(k_i) + B\Delta u(k_i) \\
x(k_i + 2|k_i) &= Ax(k_i + 1|k_i) + B\Delta u(k_i + 1) \\
&= A^2 x(k_i) + AB\Delta u(k_i) + B\Delta u(k_i + 1) \\
&\vdots \\
x(k_i + N_p|k_i) &= A^{N_p} x(k_i) + A^{N_p-1} B\Delta u(k_i) + A^{N_p-2} B\Delta u(k_i + 1) + ... \\
&\quad + A^{N_p-N_c} B\Delta u(k_i + N_c - 1).
\end{aligned}$$

And therefore the future system outputs can be predicted as:

$$\begin{aligned}
y(k_i + 1|k_i) &= CAx(k_i) + CB\Delta u(k_i) \qquad (3.13) \\
y(k_i + 2|k_i) &= CA^2 x(k_i) + CAB\Delta u(k_i) + CB\Delta u(k_i + 1) \\
y(k_i + 3|k_i) &= CA^3 x(k_i) + CA^2 B\Delta u(k_i) + CAB\Delta u(k_i + 1) + CB\Delta u(k_i + 2) \\
&\vdots \\
y(k_i + N_p|k_i) &= CA^{N_p} x(k_i) + CA^{N_p-1} B\Delta u(k_i) + CA^{N_p-2} B\Delta u(k_i + 1) + ... \\
&\quad + CA^{N_p-N_c} B\Delta u(k_i + N_c - 1). \qquad (3.14)
\end{aligned}$$

The future output vector Y, of length N_p, can be represented as:

$$Y = [y(k_i + 1|k_i), y(k_i + 2|k_i), y(k_i + 3|k_i), ..., y(k_i + N_p|k_i)]^T \tag{3.15}$$

Note that the output vector and forces in the sway direction share the nomenclature Y. As this work does not focus on operation in the sway direction any term of Y is for the output vector. And the future control horizon ΔU, of length N_c, can be represented as:

$$\Delta U = [\Delta u(k_i), \Delta u(k_i + 1), \Delta u(k_i + 2), ...\Delta u(k_i + Nc - 1)]^T \tag{3.16}$$

Combining equations (3.13), (3.14), (3.15) and (3.16) into a compact matrix form:

$$Y = Fx(k_i) + \phi\Delta U, \tag{3.17}$$

where

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{N_p} \end{bmatrix}; \phi = \begin{bmatrix} CB & 0 & 0 & ... & 0 \\ CAB & CB & 0 & ... & 0 \\ CA^2B & CAB & CB & ... & 0 \\ \vdots & & & & \\ CA^{N_p-1}B & CA^{N_p-2}B & CA^{N_p-3}B & ... & CA^{N_p-N_c}B \end{bmatrix} \tag{3.18}$$

3.2.3 Optimization

In order to find an optimal control solution, first the set-point vector must be defined and a cost function created. The set-point vector, of length N_p, is defined as:

$$R_s^T = [1\ 1\ ...\ 1]\, r(k_i), \tag{3.19}$$

where $r(k_i)$ is the set-point at instant k_i. The cost function is then defined as a quadratic function of the error over the future trajectory (between the set-point and system output) and the magnitude in the future control vector:

$$J = (R_s - Y)^T(R_s - Y) + \Delta U^T \bar{R} \Delta U, \tag{3.20}$$

where the first quadratic term defines the objective of minimizing the error between the future system prediction vector, Y, and the set-point vector R_s.

The second quadratic term is a function of ΔU and $\bar{R}$, where $\bar{R}$ is a diagonal matrix of the form:

$$\bar{R} = r_w I_{mN_c \times mN_c}, \tag{3.21}$$

and with $r_w > 0$ penalises the magnitude of ΔU. Therefore if it is desirable to minimise the error as quickly as possible then an r_w value of zero should be taken, resulting in large values within the ΔU vector. These large values may not be feasible in reality and will need to be handled using constraints. This will be discussed later in Section 3.2.4.

To find the optimal ΔU vector, J must be minimized. Substituting equation (3.17) into equation (3.20), the cost function becomes:

$$J = (R_s - Fx(k_i))^T(R_s - Fx(k_i)) - 2\Delta U^T \phi^T (R_s - Fx(k_i)) + \Delta U^T(\phi^T\phi + \bar{R})\Delta U. \tag{3.22}$$

To minimize J, the first derivative with respect to ΔU is found:

$$\frac{\delta J}{\delta \Delta U} = -2\phi^T(R_s - Fx(k_i)) + 2(\phi^T\phi + \bar{R})\Delta U, \tag{3.23}$$

and taking $\frac{\delta J}{\delta \Delta U}$ to equal 0, the optimal ΔU is found using:

$$\Delta U = (\phi^T \phi + \bar{R})^{-1}(\phi^T R_s - \phi^T F x(k_i)), \tag{3.24}$$

where matrix $\phi^T \phi$ has dimensions of $mN_c \times mN_c$, $\phi^T F$ has dimensions of $mN_c \times n$, and $\phi^T \bar{R}_s$ is the last q columns of $\phi^T F$.

3.2.4 Constraints

If the controller designer sets r_w to zero or near zero, then the resulting optimal solution for ΔU could potentially include some extremely large values. In reality the system inputs are generally limited by both rate of change and absolute saturation limits. If the controller output vector ΔU is not constrained within these limits then system performance will likely degrade. In this work, both limits on Δu and u are set for each system input, e.g.

$$\Delta u^{min} \leq \Delta u(k_i) \leq \Delta u^{max},$$

and

$$u^{min} \leq u(k_i) \leq u^{max}$$

Constraints on Δu are due to the rate of change for the system input. For example, a control surface may only change its angle at up to $\pm 20^o s^{-1}$. Constraints on u are more common and often due to the physical limitations of the actuators, for example a control surface only able to move to a delta angle of $\pm 30^o$. When constraining a multi-input system, constraints need to be individually assigned to each input:

$$\begin{aligned}
\Delta u_1^{min} &\leq \Delta u_1(k_i) \leq & \Delta u_1^{max}, \\
\Delta u_2^{min} &\leq \Delta u_2(k_i) \leq & \Delta u_2^{max}, \\
&\vdots& \\
\Delta u_m^{min} &\leq \Delta u_m(k_i) \leq & \Delta u_m^{max},
\end{aligned} \tag{3.25}$$

and

$$\begin{aligned}
u_1^{min} &\leq u_1(k_i) \leq & u_1^{max}, \\
u_2^{min} &\leq u_2(k_i) \leq & u_2^{max}, \\
&\vdots \\
u_m^{min} &\leq u_m(k_i) \leq & u_m^{max},
\end{aligned} \tag{3.26}$$

For this MPC algorithm, the constraints are applied to the future ΔU vector. All the constraints are broken down into individual constraints, e.g.

$$\Delta U^{min} \leq \Delta U \leq \Delta U^{max}, \tag{3.27}$$

becomes

$$\begin{aligned}
-\Delta U &\leq -\Delta U^{min} \\
\Delta U &\leq \Delta U^{max}.
\end{aligned} \tag{3.28}$$

In matrix form this can be represented as:

$$\begin{bmatrix} -I \\ I \end{bmatrix} \Delta U \leq \begin{bmatrix} -\Delta U^{min} \\ \Delta U^{max} \end{bmatrix}, \tag{3.29}$$

where I is an identity matrix of dimension $m \times m$.

To take into account the absolute limits both the current value of u and future values of Δu must be considered. Therefore the future values of u are defined as:

$$\begin{bmatrix} u(k_i) \\ u(k_i+1) \\ u(k_i+2) \\ \vdots \\ u(k_i+N_c-1) \end{bmatrix} = \overbrace{\begin{bmatrix} I \\ I \\ I \\ \vdots \\ I \end{bmatrix}}^{C_1} u(k_i-1) + \overbrace{\begin{bmatrix} I & 0 & 0 & \dots & 0 \\ I & I & 0 & \dots & 0 \\ I & I & I & \dots & 0 \\ \vdots & & & & \\ I & I & I & \dots & I \end{bmatrix}}^{C_2} \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i+1) \\ \Delta u(k_i+2) \\ \vdots \\ \Delta u(k_i+N_c-1) \end{bmatrix}, \tag{3.30}$$

The absolute constraints on u can therefore be represented as:

$$-(C_1 u(k_i-1) + C_2 \Delta U) \leq -U^{min} \tag{3.31}$$

$$(C_1 u(k_i-1) + C_2 \Delta U) \leq U^{max} \tag{3.32}$$

Combining all the constraints in compact matrix form produces:

$$\overbrace{\begin{bmatrix} M_1 \\ M_2 \end{bmatrix}}^{M} \Delta U \leq \overbrace{\begin{bmatrix} N_1 \\ N_2 \end{bmatrix}}^{\gamma}, \tag{3.33}$$

where

$$M_1 = \begin{bmatrix} -C_2 \\ C_2 \end{bmatrix};\ N_1 = \begin{bmatrix} -U^{min} + C_1 u(k_i - 1) \\ U^{max} - C_1 u(k_i - 1) \end{bmatrix};\ M_2 = \begin{bmatrix} -I \\ I \end{bmatrix};\ N_2 = \begin{bmatrix} -\Delta U^{min} \\ \Delta U^{max} \end{bmatrix}. \tag{3.34}$$

As the cost function from equation (3.22) is quadratic, and the constraints are linear inequalities, the optimal solution of ΔU is a standard quadratic programming problem. The solution of this can be found using various methods described in the next section.

3.2.5 Quadratic Programming

Several methods exist for solving quadratic programming problems (QPP) such as finding the optimal solution for ΔU, [69]. In this subsection, the active set solver methodology will be discussed and the Hildreth programming procedure introduced, [67].

First the method of quadratic programming for equality constraints will be introduced and then quadratic programming with inequality constraints discussed. Finally the active set method is introduced, this will be used when solving quadratic programming problems with inequality constraints.

From standard literature we define the objective function and constraints as:

$$J = \frac{1}{2} x^T E_{qp} x + x^T F_{qp} \tag{3.35}$$

$$Mx \leq \gamma \tag{3.36}$$

where for the MPC algorithm:

$$x = \Delta U; \tag{3.37}$$

$$E_{qp} = -(\phi^T \phi + \bar{R}) \tag{3.38}$$

$$F_{qp} = \phi^T (R_s - Fx(k_i)) \tag{3.39}$$

Note the subscript on E_{qp} and F_{qp}. In common literature on quadratic programming these subscripts do not exist but are used here in order to distinguish F, from equation (3.18), and F_{qp}.

Quadratic Programming for Equality Constraints

An example of solving a QPP for an equality constraint, minimize:

$$J = (x_1 - 2)^2 + (x_2 - 2)^2, \tag{3.40}$$

subject to

$$x_1 + x_2 = 1 \tag{3.41}$$

To satisfy constraint (3.41) we substitute it into the objective function:

$$J = (x_1 - 2)^2 + (1 - x_1 - 2)^2, \tag{3.42}$$

Then differentiate with respect to x_1 to solve for the minimum:

$$\begin{aligned} J &= 2x_1^2 - 2x_1 + 5 && (3.43) \\ \frac{\delta J}{\delta x_1} &= 4x_1 - 2 && (3.44) \end{aligned}$$

Setting (3.44) to equal zero then the solution for x_1 is 0.5. Substituting this into constraint (3.41) we find x_2 to equal 0.5. This is a relatively simple example with one equality constraint. For multiple constraints it is more convenient to use Lagrange multipliers.

Lagrange Multipliers

To minimise the objective function subject to equality constraints a Lagrange function can be used:

$$J = \frac{1}{2}x^T E_{qp} x + x^T F_{qp} + \lambda^T (Mx - \gamma). \tag{3.45}$$

where λ is the Lagrange multiplier vector, [70].

In order to minimise equation (3.45) the first partial derivatives with respect to x and λ are taken:

$$\begin{aligned} \frac{\delta J}{\delta x} &= E_{qp}x + F_{qp} + M^T\lambda = 0 && (3.46) \\ \frac{\delta J}{\delta \lambda} &= Mx - \gamma = 0 && (3.47) \end{aligned}$$

Equation (3.46) can then be rearrange such that:

$$x = -E_{qp}^{-1}F_{qp} - E_{qp}^{-1}M^T\lambda \tag{3.48}$$

Multiplying through Equation (3.48) by M and inserting into equation (3.47) gives:

$$\gamma = -E_{qp}^{-1}F_{qp} - E_{qp}^{-1}M^T\lambda \tag{3.49}$$

Assuming $ME_{qp}^{-1}M^T$ is invertible, then by rearranging equations (3.48) and (3.49) the optimal λ and x can be found:

$$\begin{aligned} \lambda &= -(ME_{qp}^{-1}M^T)^{-1}(\gamma + ME_{qp}^{-1}F_{qp}) && (3.50) \\ x &= -E_{qp}^{-1}(M^T\lambda + F_{qp}). && (3.51) \end{aligned}$$

Given equality constraints that are independent of each other, equations (3.50) and (3.51) can be used to find the optimal solution for the quadratic programming problem. If the equality constraints are dependant on each other then no solution exists.

For example, minimise:

$$J = \frac{1}{2}x^T E_{qp} x + x^T F_{qp}, \tag{3.52}$$

subject to the independent equality constraints:

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 & (3.53)\\ 3x_1 - 2x_2 - 3x_3 &= 1, & (3.54) \end{aligned}$$

where

$$E_{qp} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} ; \; F_{qp} = \begin{bmatrix} -2 \\ -3 \\ -1 \end{bmatrix} . \tag{3.55}$$

From equations (3.53) and (3.54) the M and γ matrices are formed:

$$M = \begin{bmatrix} 1 & 1 & 1 \\ 3 & -2 & -3 \end{bmatrix} ; \; \gamma = \begin{bmatrix} 1 \\ 1 \end{bmatrix} . \tag{3.56}$$

Using equations (3.50) and (3.51) we can find the optimal λ and x:

$$\lambda = -(ME_{qp}^{-1}M^T)^{-1}(\gamma + ME_{qp}^{-1}F_{qp}) = \begin{bmatrix} 1.5452 \\ -0.0323 \end{bmatrix} . \tag{3.57}$$

Therefore:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = -E_{qp}^{-1}F_{qp} - E_{qp}^{-1}M^T\lambda = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 & 3 \\ 1 & -2 \\ 1 & -3 \end{bmatrix} \begin{bmatrix} 1.5452 \\ -0.0323 \end{bmatrix} = \begin{bmatrix} 0.4516 \\ 1.2903 \\ -0.7419 \end{bmatrix} \tag{3.58}$$

Quadratic Programming for Inequality Constraints

In the case that a constraint states that the variable must be less than or equal to a certain value, then this constraint can be defined as active or inactive. For an individual constraint, if $M_i x = \gamma$ then the constraint is said to be active, however, if $M_i x < \gamma$ then the constraint is inactive. In terms of the Lagrange multipliers, we use the Kuhn-Tucker conditions to define whether a constraint is active or inactive:

$$\begin{aligned} E_{qp}x + F_{qp} + M^T\lambda &= 0 \\ Mx - \gamma &\leq 0 \\ \lambda^T(Mx-\gamma) &= 0 \\ \lambda &\geq 0, \end{aligned} \tag{3.59}$$

which can be written in terms of the active set of constraints, where S_{act} denotes the index set of active constraints:

$$E_{qp}x + F_{qp} + \sum_{i \in S_{act}} \lambda_i M_i^T = 0$$

$$M_i x - \gamma_i = 0 \quad i \in S_{act} \tag{3.60}$$

$$M_i x - \gamma_i < 0 \quad i \notin S_{act} \tag{3.61}$$

$$\lambda_i \geq 0 \quad i \in S_{act} \tag{3.62}$$

$$\lambda_i = 0 \quad i \notin S_{act} \tag{3.63}$$

The purpose of the Kuhn-Tucker conditions is to define which constraints are active. Those active constraints can then be treated as equality constraints and the inactive constraints ignored. With the set of active constraints we can solve the quadratic programming problem as in the previous subsection. The problem now lies with identifying which constraints are active and inactive, this is addressed by using what is called active set methods.

The idea of the active set methods is to define what is called the working set that is a subset of the constraints that are active. On each iteration an equality constraint is solved resulting in a set of Lagrange multipliers. If one of the Lagrange multipliers is less than zero then that constraint is relaxed by deleting it from the working set. The process is continued until the solution for λ provides a set of values greater than or equal to zero.

For example, optimise the objective function where:

$$E_{qp} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \; F_{qp} = \begin{bmatrix} -2 \\ -3 \\ -1 \end{bmatrix}, \tag{3.64}$$

subject to the inequality constraints:

$$x_1 + x_2 + x_3 \le 1 \tag{3.65}$$

$$3x_1 - 2x_2 - 3x_3 \le 1 \tag{3.66}$$

$$x_1 - 3x_2 + 2x_3 \le 1 \tag{3.67}$$

Starting with the all the constraints as the working set of constraints, $M_w x = \gamma_w$, the Lagrange multipliers are calculated using equation (3.50):

$$\lambda = -(M_w E_{qp}^{-1} M_w^T)^{-1}(\gamma_w + M_w E_{qp}^{-1} F_{qp}) = \begin{bmatrix} 1.6873 \\ 0.0309 \\ -0.4352 \end{bmatrix}. \tag{3.68}$$

As can be seen, the third Lagrange multiplier is negative so will be deleted from the working set of constraints. Again the Lagrange multipliers are calculated using the working set:

$$\lambda = \begin{bmatrix} 1.6452 \\ -0.0323 \end{bmatrix}. \tag{3.69}$$

Again the constraint corresponding to the negative Lagrange multiplier is deleted from the working set and the Lagrange multipliers found. Finally the set of Lagrange multipliers is greater than zero and so the remaining constraints are defined as the active set and the optimal solution for x can be found using equation (3.51). As the example shows, the solution is found after an iterative process to find the active set. If the active set of constraints can be identified beforehand then the iterative process could be shortened.

Primal-Dual Method

The Primal method, used in the previous subsection, can be computationally expensive if there are a large number of constraints. To reduce the computational cost the primal method can be used with the dual method that is used to identify the constraints that are not active.

Assuming the problem is feasible and there exists an x such that $Mx < \gamma$, the primal problem is equivalent to:

$$\max_{\lambda \ge 0} \min_{x} \left[\frac{1}{2} x^T E_{qp} x + x^T F_{qp} + \lambda^T (Mx - \gamma)\right]. \tag{3.70}$$

The minimization over x is unconstrained and is found using:

$$x = -E_{qp}^{-1}(F_{qp} + M^T \lambda). \tag{3.71}$$

Substituting equation (3.71) into equation (3.70) gives the dual problem:

$$\max_{\lambda \ge 0}(-\frac{1}{2}\lambda^T H \lambda - \lambda^T K - \frac{1}{2} F_{qp}^T E_{qp}^{-1} F_{qp}), \tag{3.72}$$

where the matrices H and K are given by:

$$H = ME_{qp}^{-1}M^T \tag{3.73}$$

$$K = \gamma + ME_{qp}^{-1}F_{qp}. \tag{3.74}$$

Equation (3.72) is a quadratic programming problem with λ as the decision variable and can be written in terms of finding the minimum:

$$\min_{\lambda \geq 0}(\frac{1}{2}\lambda^T H\lambda + \lambda^T K + \frac{1}{2}\gamma^T E_{qp}^{-1}\gamma), \tag{3.75}$$

The set of Lagrange multipliers that minimize the dual objective function:

$$J = \frac{1}{2}\lambda^T \lambda + \lambda^T K + \frac{1}{2}\gamma^T E_{qp}^{-1}\gamma, \tag{3.76}$$

subject to $\lambda \geq 0$, are denoted as λ_{act}, and the corresponding constraints are described by M_{act} and γ_{act}. With the values of λ_{act} and M_{act}, and the active constraints being solved as equality constraints, the primal variable vector x can be solved using:

$$x = -E_{qp}^{-1}F_{qp} - E_{qp}^{-1}M_{act}^T\lambda_{act}, \tag{3.77}$$

Hildreth Programming Procedure

The Hildreth programming procedure is used for solving the dual problem. Each Lagrange multiplier, λ_i, is individually solved in an iterative process so as to minimise the objective function. If the solution results in $\lambda_i < 0$ then $\lambda_i = 0$. One iteration is considered as solving the full set of λ_i values. The program continues to iterate until λ converges to a fixed set of values.

The method can be described using:

$$\lambda_i^{m+1} = \max(0, w_i^{m+1}), \tag{3.78}$$

with

$$w_i^{m+1} = -\frac{1}{h_{ii}}\left[k_i + \sum_{j=1}^{i-1} h_{ij}\lambda_j^{m+1} + \sum_{j=i+1}^{n} h_{ij}\lambda_j^m\right], \tag{3.79}$$

where the scalar h_{ij} is the ij^{th} element in the matrix $H = ME_{qp}^{-1}M^T$, and k_i is the i^{th} element in the vector $K = \gamma + ME_{qp}^{-1}F_{qp}$.

Once λ has converged to within a predefined tolerance the result is defined as λ^* and the optimal solution within the constraints can be found using:

$$x = -E_{qp}^{-1}(F_{qp} + M^T\lambda^*). \tag{3.80}$$

It is possible that this algorithm can finish before it has successfully converged to a suitable λ^* set. Reasons for poor convergence include; the algorithm reaching a predefined maximum number of iterations or it has been interrupted in order to maintain real-time operation. To better ensure that the optimal solution is within the constraints, instead of using λ^* to find the solution, it is used to indicate the set of active constraints. Therefore taking the constraints corresponding with λ^* values > 0, M_{act} and γ_{act}, the optimal solution is found using:

$$\lambda^*_{act} = -(M_{act}E_{qp}^{-1}M_{act}^T)^{-1}(\gamma_{act} + M_{act}E_{qp}^{-1}F_{qp}). \tag{3.81}$$

3.3 Feedback

By definition closed loop control systems require some form of feedback acquired by measurement using sensors, estimation using observers, or a combination of both. In reality, a feedback signal can be represented as:

$$signal = x + e_s + e_r, \tag{3.82}$$

where x is the true value of the variable that is being measured, e_s is systematic error and e_r is random error.

The systematic error is a variable offset from the x value due to inherent inaccuracies with the sensor or observer. Such errors can be caused by poor sensor calibration or model inaccuracies within an observer, these will always exist with varying magnitude as no measurement will ever be exact. Random error, also known as noise, can be due to many factors, but for sensors it is often caused by electromagnetic interference. As time tends towards infinity the mean average of the random error will tend towards zero. Random errors on feedback signals can cause significant problems for control systems, therefore in this section methods for reducing the magnitude of random error and estimating system states are presented.

3.3.1 Quantifying Random Error

Before developing methods to handle errors, first the magnitude of the random error on the depth and pitch signals is quantified. To do this, data from a previous experimental test with the Delphin2 AUV is used. The compass sensor, coupled with a pressure transducer, provides the depth and pitch measurement. To study the random error, the system dynamics are subtracted from the measurement signal:

$$e_r = signal - (x + e_s), \tag{3.83}$$

First the random error is eliminated from the measurement signal using a Savitzky-Golay (SG) filter. This filter works by fitting a polynomial, of order k, to a window size of f samples. The SG filter is particularly suitable for eliminating random error as the resultant polynomial is found using the least squares method. The order of the polynomial is chosen to be 2 as higher orders are more

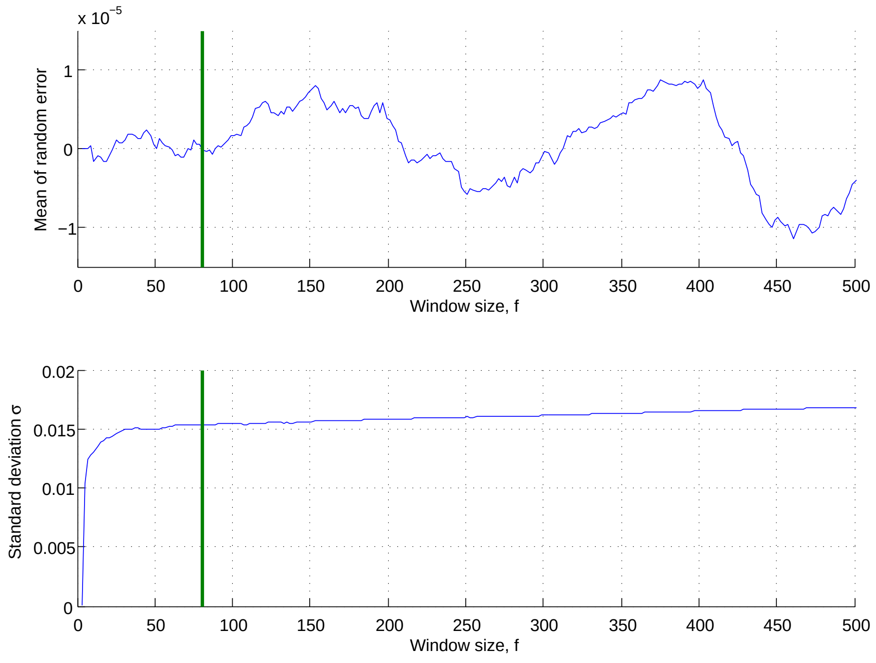


Figure 3.3: Effect of window size used by the Savitzky-Golay filter on the resulting mean and standard deviation of the random error for the depth signal.

likely to capture the higher frequencies of the measurement noise. Subtracting the filtered signal from the unfiltered filter should give in the random error, assuming the filter is performing correctly.

In order to find a suitable length of sample window for the SG depth and pitch filters, both the mean and standard deviation of the random errors are evaluated as a function of f, see Figure 3.3 for the results with the depth signal.

The mean average of the random error, for the depth signal, has a lower absolute value with window sizes less than 91, and above this it starts to deviate as the filter begins to filter out the lower frequency dynamics of the signal. The plot of standard deviation against window size demonstrates that if the window size is too small then the filter will not adequately attenuate the random error. A suitable window size is chosen, for the depth signal, to be 81 samples. The pitch signal is filtered using the same method with a window size of 49 samples. The standard deviation and mean random error for the depth and pitch signals are presented in Table 3.1. The unfiltered, filtered and random error for the depth signal is presented in Figure 3.4.

Table 3.1: Acceptable magnitude of errors for the terms within the controller model with the stable and balanced controller parameter sets.

Parameter	**Value**
Total number of samples	14001
k	2
Depth f	81
Pitch f	49
Depth mean random error	2.4039×10^{-5} m
Pitch mean random error	$-3.3167 \times 10^{-5}\ ^{\circ}$
Depth random error standard deviation	0.0144m
Pitch random error standard deviation	0.0715°

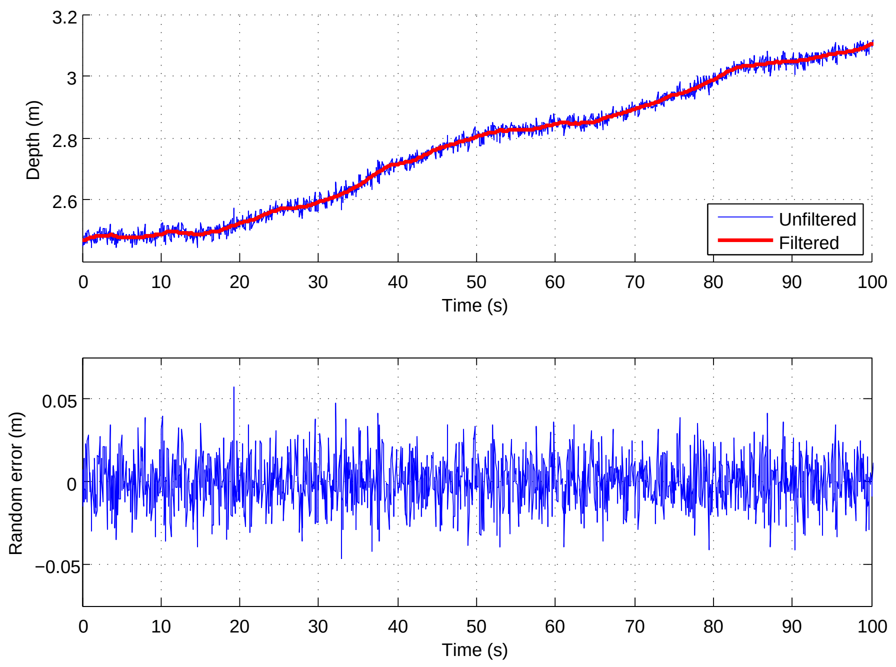


Figure 3.4: An example of a depth signal before and after filtering and at the bottom, the random error on the depth signal.

Another feature of the random error that needs to be investigated is the cause, or type, of random error. In Figure 3.5 two histograms are presented for the distribution of the random errors for both the pitch and depth signals. Both histograms show a Gaussian distribution centred about zero. For the depth signal the random error is most likely caused by electromagnetic interference on the analogue output from the pressure transducer. Shielding of the pressure transducer electrical wiring and filtering the supply voltage may reduce the magnitude of the random error on the depth signal.

The cause of the random noise on the pitch signal is not solely a result of interference but also the quantization of the analogue pitch value into a digital value. The relatively coarse resolution (0.1^o) causes sudden jumps in the value of the signal that, when compared to the filtered signal, results in similar issues as random errors. With regards to control systems, this low resolution can have an equally detrimental effect on system performance as normal measurement noise. Increasing the resolution of the analogue to digital converter within the compass module would decrease this effect but this option is not possible with the available compass on the Delphin2 AUV. It is not known what the quantization method is used within the ADC but for rounding and truncated methods the quantisation errors would produce a standard deviation of 0.025° and 0.051° respectively. Both are smaller than what is measured and so the rest of the noise can be attributed to electromagnetic interference.

The depth and deduced noise signals were analysed using a FFT function within MATLAB. It was found that there was no distinct dominant frequency and the noise was approximately white. It should also be noted that there was no noticeable influence from other electrical systems on the vehicle, for example the thrusters, with the exception when one of the thrusters failed due to electrical problems.

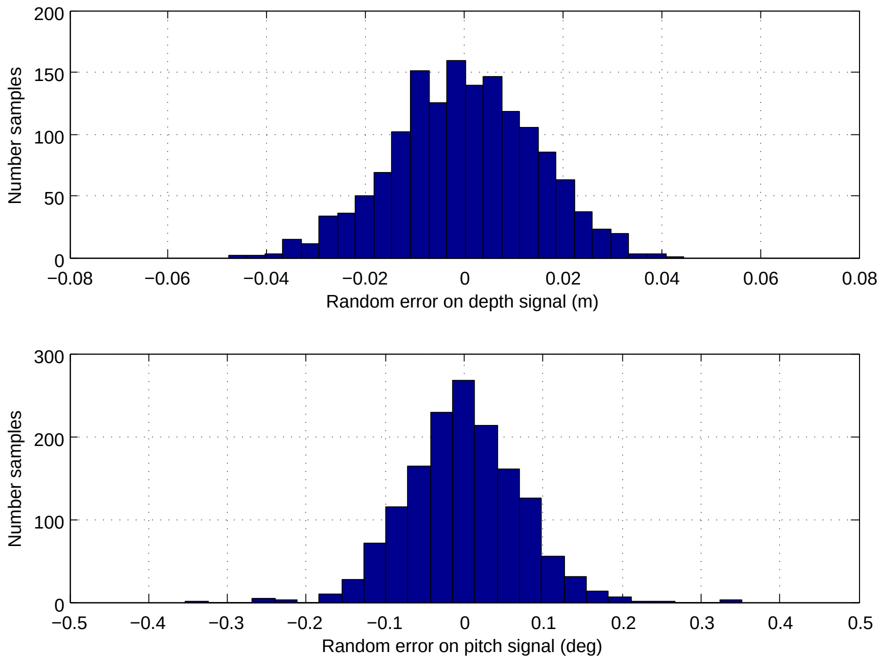


Figure 3.5: Histogram distributions for the random noise from the depth and pitch signals.

3.3.2 Signal Noise Ratio

There are various methods of quantifying signal-to-noise ratio (SNR), the suitability of each depends on the type of signal being evaluated. For this work the SNR will be evaluated as:

$$SNR = \frac{\mu}{\sigma}, \tag{3.84}$$

where μ is the signal mean or expected value and σ is the standard deviation of the signal's random error.

Inherent within the MPC algorithm, used in this work, is full state feedback. The states of the model within the controller are not the same as the outputs (depth and pitch), recalling equation (3.3) the states within the controller are the difference between the present and previous measured sample. Effectively the signal being used as feedback is the time derivative of the measured or observed signals. Therefore as a signal (e.g. depth) tends towards a steady-state value, and the standard deviation of the random error remains constant, the SNR will tend towards zero. If these assumptions hold true then dynamics of the controller, when the system converges towards zero error and is stable, will be dominated by the random error. As the AUV is a heavily damped system, these fluctuations due to measurement noise could be damped out but this depends on the magnitude of the noise, the damping of the system, and the controller tuning parameters.

3.3.3 Filtering

In order to reduce the random error on the depth and pitch signals, and calculate the time derivative of the signals, a polynomial type filter will be used. Other methods of filtering, including various low-level filters, may provide sufficient levels of random error attenuation however these filters are also renowned for inducing a time delay that can cause stability issues within a control system. The polynomial type (PT) filter can be designed with a very low time delay (assuming the second derivative of the signal is low), and also provides a means of calculating the time derivative. An example of this filter with a MPC controller is given later in this work.

The basic idea behind the PT filter is similar to that of the Savitzky-Golay (SG) filter as it fits a polynomial to an array of data. The primary difference between the SG and PT filters is that the PT filter fits a polynomial to the latest f samples while the SG filter fits a polynomial to data points before (past) and after (future) the current data point. The SG filter will work better at attenuating random error than the PT filter, however, it is only feasible for post-processing data and not for real-time applications.

Two arrays are used; one with the latest f samples, and another of the sample time-stamps. The array of samples is filled with the latest sample in the first array index:

$$S = [s(k), s(k-1), s(k-2), ..., s(k-f+1)] \tag{3.85}$$

while the time array takes the form:

$$T = -\left[0, dt, dt \times 2, ..., dt \times (f-1)\right] \tag{3.86}$$

Using the two arrays a second order polynomial can be fitted to the data using least squares regression. This is done using optimised functions within MATLAB and Python, both called polyfit.

The advantage of this filter is that the fitted polynomial will track the general trajectory of the data array, thus smoothing the results. The latest filtered sample is calculating by evaluating the polynomial at time equal to zero. Likewise the derivative of the signal is found by calculating the derivative of the fitted polynomial and then evaluating it at time equal to zero.

Like the SG filter, the length of the sample array is a key parameter with regards to filter performance. To help choose the array lengths, for the depth and pitch signals, the standard deviation of the error between the PT and SG filter is calculated for both the normal signal and the derivative of the signal. Figures 3.6 and 3.7 present these results. The length of the arrays are taken at the minimum of each curve, thus the optimal length for the depth and pitch arrays are taken as 32 and 18 samples respectively. These array lengths are chosen for sampling rates of 10 Hz.

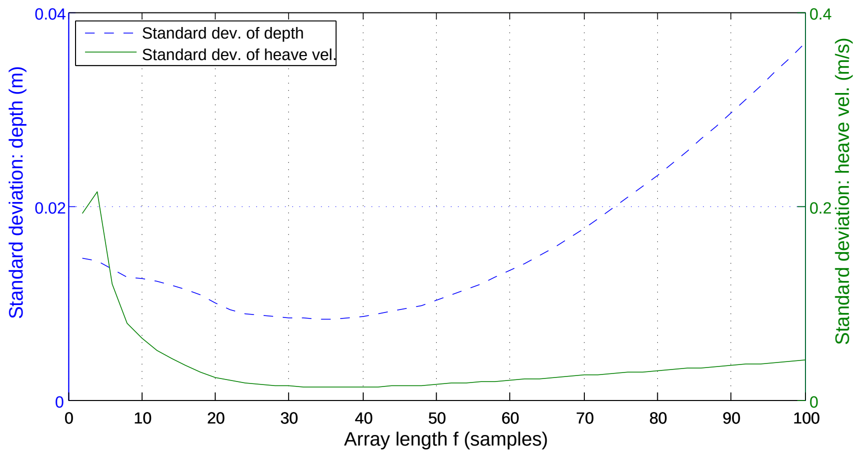


Figure 3.6: Standard deviation of the errors between the PT and SG filters for both the depth and heave signals, against the length of the PT array. Note that the minimum for both signals occurs at an array length of 32.

This filter can be used to provide the state feedback to the MPC algorithm. The state variable vector comprises of the controller outputs, $y(t)$, and the derivatives of the system states, Δx_m, see equation (3.6). The outputs are taken as the filtered samples while the values for the state derivatives are taken as the difference between the respective state polynomials at time zero and minus one sample time-stamp.

3.3.4 State Observer

The MPC algorithm derived earlier in this Chapter requires full state feedback. The filter that was introduced in the previous subsection provides a means to filter and calculate the derivative of a signal however the signal could still be too noisy for the MPC algorithm. Also, for many systems it is too difficult or expensive to measure all the states, if they can be measured at all. For systems were measurement of all the states is not feasible then a mathematical model of the system can be used to

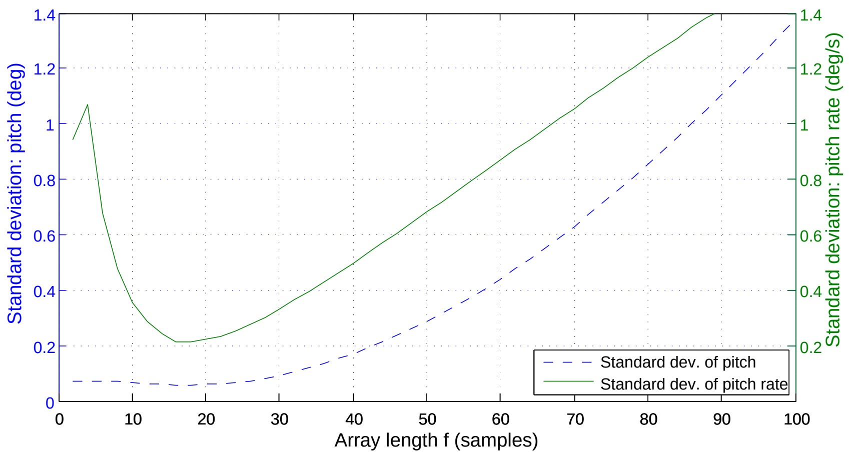


Figure 3.7: Standard deviation of the errors between the PT and SG filters for both the pitch and pitch rate signals, against the length of the PT array. Note that the minimum for both signals occurs at an array length of 18.

estimate, or observe, the system states using the system outputs.

If a mathematical model of a system is perfectly accurate and the exact initial conditions are known, then the system states and outputs could be simply calculated using:

$$x(k+1) = Ax(k) + Bu(k) \tag{3.87}$$

$$y(k) = Cx(k) \tag{3.88}$$

Unfortunately a perfectly accurate mathematical model of a system is, in reality, impossible. Therefore as time approaches infinity the error between the real system states and those calculated using the model could become unacceptably large. The measured outputs of the system can be used to reduce these errors, producing estimates of the states by using a Luenberger observer:

$$\hat{x}(k+1) = A\hat{x} + L\left[y(k) - \hat{y}(k)\right] + Bu(k) \tag{3.89}$$

$$\hat{y}(k) = C\hat{x} \tag{3.90}$$

where $\hat{x}$ and $\hat{y}$ denote the estimates of the state-variable and output vectors respectively.

Before continuing with the design of an observer the system must be checked to see if it is indeed observable. This is checked using the observability matrix:

$$Ob = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{3.91}$$

then comparing its rank against the dimensions of A:

$$\text{Number of unobservable states} = \text{length}(A) - \text{rank}(Ob). \tag{3.92}$$

If the system is observable (number of unobservable states equals zero) then an observer can be designed by finding a suitable L matrix. First the error between the true state values and their estimates is defined as:

$$e(k+1) = \hat{e}(k) - e(k) \tag{3.93}$$

It is intuitive that the Luenberger observer can be said to be asymptotically stable if this error approaches zero as the sample number, k, approaches infinity. Combining equations (3.89) and (3.93) gives:

$$e(k+1) = (A - LC)e(k) \tag{3.94}$$

Thus for the observer to be stable the eigenvalues of $(A - LC)$ must be within the unit circle (for a discrete system). Within the control literature it is common for the observer to be designed by choosing suitable eigenvalues and then finding the L matrix, [71]. For certain real-time systems this may not be the best solution and a computer algorithm can be employed to find suitable values for L so that the observer is stable.

Several algorithms exist for this but for this work a linear-quadratic state-feedback regulator for discrete-time state-space systems (MATLAB function: dlqr) is used. This calculates the values for the L matrix to provide a stable observer using two weighting matrices Q and R that are chosen using trial and error to provide good overall system performance.

3.4 Implementation

Here the basic implementation will be described with more detail for each controller described in the relevant sections. There are three main stages of development for each controller;

1. MPC controller developed within MATLAB and simulated using the full non-linear model.
2. The controller is then translated into the Python programming language and simulated using the same non-linear model as a means of error checking the code.
3. Finally the controller is modified to receive feedback from the AUV sensors and observers and then experimentally tested using the Delphin2 AUV.

The initial development of the controller is conducted using MATLAB with the non-linear model that will be produced later in Chapter 5. The reasons for using MATLAB at the start of development are; ease of programming, simple matrix operations and flexible plotting functions to evaluate controller performance. The majority of functions are available using the standard MATLAB package, however some of the functions are from the Control Toolbox.

Finally, assuming that the controller is working in simulation and that suitable controller tuning parameters have been found, the controller is ready to be experimentally tested with the Delphin2 AUV.

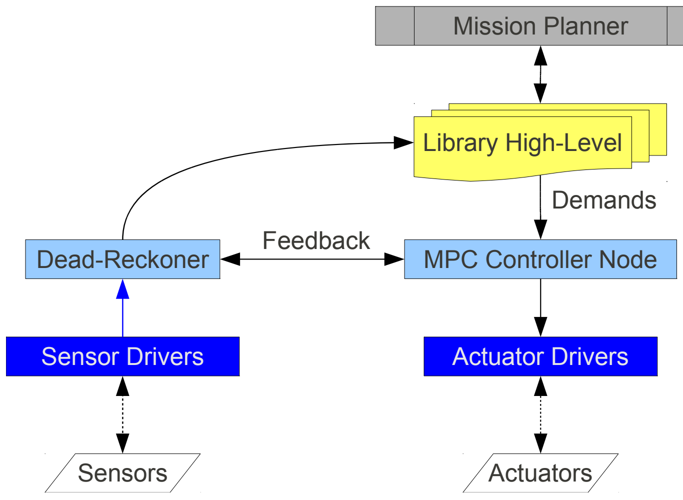


Figure 3.8: Flow diagram of information into and out of the model predictive controller in the Delphin2 AUV.

Once the controller is working within the MATLAB environment, the controller is translated into the Python programming language as a separate node within the Delphin2 control software. To ensure that no mistakes have been made in translating the code, the controller node is coupled with a simulation node. This simulation node is identical to the dead-reckoner node in Section 4.5, but without sensor feedback. The Python simulation node and MATLAB simulator are also identical, therefore the controller is assumed to be working and correct when the simulation results from the Python code matches the results from the previous MATLAB simulations. No results from the Python simulations will be presented in this work as this stage is only used for checking for coding errors.

Figure 3.8 demonstrates the basic flow of information into and out of the MPC controller. Controller demands, for example, depth demand, are published from the mission planner. The MPC controller then subscribes to the different demands for which it is designed to control. Feedback is published by the dead-reckoner node and the controller subscribes to this topic. Publishers are then set up within each controller to enable the controller to publish the necessary demands to the actuators. For a fuller explanation on the software architecture see Section 4.5.

3.4.1 Python Toolboxes

As previously stated, the MATLAB functions that have been used in this work are available from the standard MATLAB library and control toolbox. In order to effectively translate the MATLAB code into efficient Python code several different packages were required. In this section a brief summary of the recommended packages used in this work is given.

The first packages to be described are the Numpy and Scipy packages [72]. These are very popular packages and provide a vast array of mathematical functions, including linear algebra functions and solvers. For quick and relatively efficient mathematical computations, these toolboxes are very useful. The wide range of available functions also helps reduce the number of packages' that a particular node needs. Also, due to these packages popularity and as they are often installed as standard with the Python language, the chance of a node working on various different computers without having to install additional software is highly advantageous.

If a node contains a significant amount of linear algebra calculations then it may prove beneficial to use the CVXOPT package [73]. This package typically solves linear algebra problems significantly faster than the Numpy package. It also contains quadratic programming solvers that are quicker at finding a solution compared to the Hildreth programming procedure that has been written using standard Python syntax. The major disadvantage of these solvers is that they do not provide the option to operate in real-time. It is therefore recommended that the controller designer runs the MPC controller with the Hildreth programming procedure and then, once the controller is working reliably, switch to using the solvers from CVXOPT.

The final package that is worth including is Yottalab [74]. This contains two useful functions; a continuous to discrete function for discretizing state-space models, and a discrete linear quadratic regulator that can be used to find a stable gain matrix for the Luenberger observer. Note that the Yottalab package also requires the Slycot toolbox to be installed [75].

3.4.2 Conclusions

In the later Chapters of this Thesis several controllers will be developed, first simulated using MATLAB and then experimentally verified using the Python programming language. Examples of both the MATLAB and Python code can be found in Appendix A.

Chapter 4

Delphin2 AUV

4.1 Introduction

The majority of literature on control systems for autonomous underwater vehicles (AUVs) has been produced by performing simulations that use mathematical models of existing vehicles. To fully evaluate the performance of novel AUV control systems requires the experimental application of them on-board an actual AUV. It was therefore decided early in this project that an AUV would be designed and built at the University of Southampton, to a specification that would enable research into the next generation of AUV design and control.

The fundamental concept for this AUV is that it is to be capable of both high-speed transits and low-speed inspection type manoeuvres. The specifications for this AUV are:

- The capability to operate in flight-style and low-speed hover modes, with full six degrees-of-freedom (DOF) controllability whilst operating with forward speed.
- Length and dry weight below 2 metres and 60 kg respectively, and a depth rating of 50 metres.
- Design focused on ergonomics and reliability, and the use of open-source software.
- Modular design to reduce costs of future modifications and ease transportation of the AUV, and the use of components that do not require special treatment during transportation or shipping (e.g. lithium batteries).

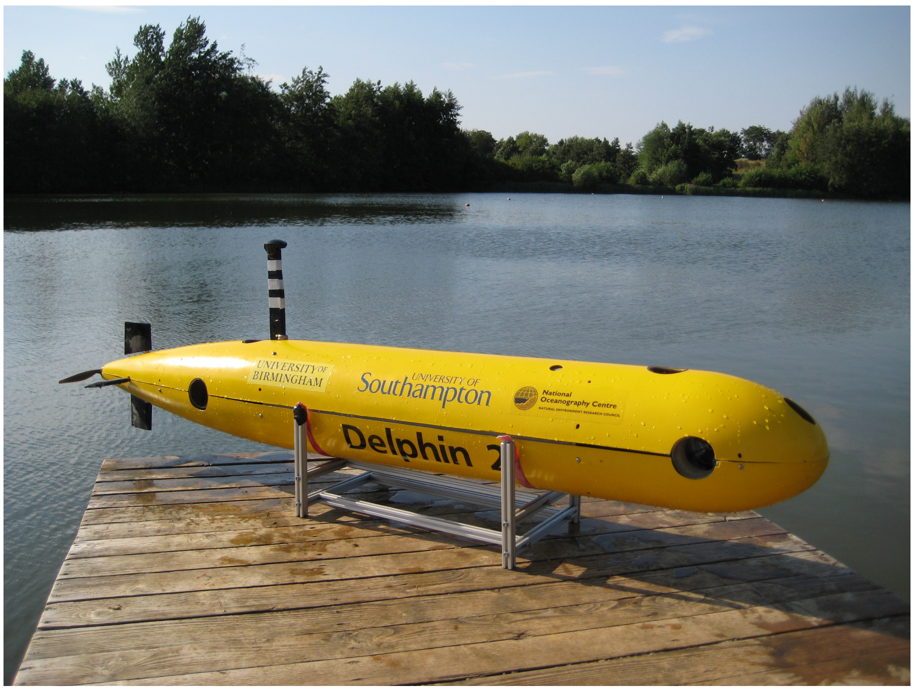


Figure 4.1: Delphin2 AUV at Eastleigh Lakes.

The resulting vehicle is the Delphin2 AUV, Figure 4.1. It is the third AUV developed by PhD and Masters level students at the University of Southampton under the supervision of research staff at the University and the National Oceanography Centre Southampton. The concept and configuration of Delphin2 is similar to that of Delphin1 [17], but with a greater focus on system reliability and ease of use. The mechanical design of this vehicle was produced using a commercial computer aided design (CAD) package and is primarily that of the author. In this chapter the mechanical design of the Delphin2 AUV, the on-board sensors and actuators, and the control software, will be presented and discussed.

4.1.1 Overview

The Delphin2 AUV is a hover-capable torpedo shaped AUV. It has a rear propeller, four independently actuated control surfaces at the rear, and two vertical and two horizontal through-body tunnel thrusters, see Figure 4.2. The rear propeller produces the thrust necessary for forward propulsion and the control surfaces are used to adjust the vehicle's yaw, pitch, and if desired, roll, at medium to high surge velocities. At low (and zero) surge velocities the two vertical tunnel thrusters are used to control the AUV's depth and pitch, whilst the two horizontal tunnel thrusters are used to control the AUV's yaw and sway velocity. When operating at low speeds the vehicle has the ability to control five degrees-of-freedom (DOF), but when the surge velocity of the vehicle is sufficiently high the AUV has the ability to control all six DOF. When operating with a surge velocity the vehicle is over-actuated, with more active actuators than DOF.

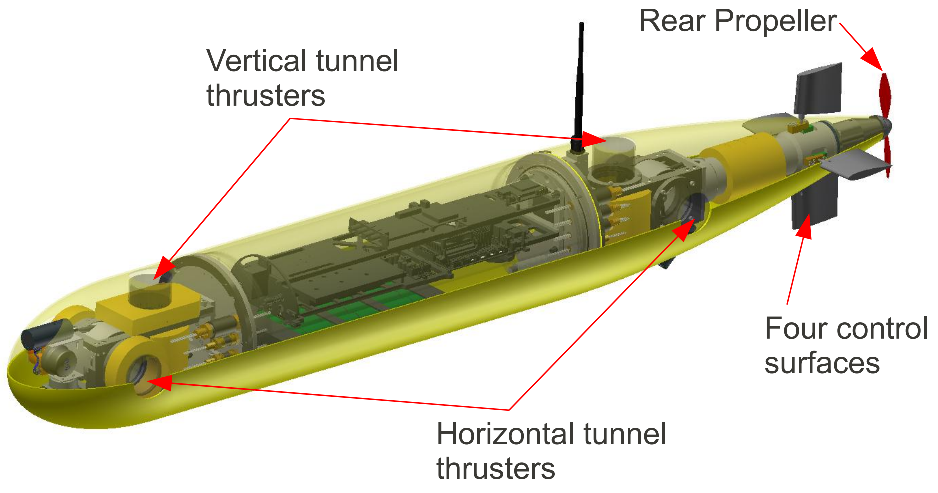


Figure 4.2: Computer rendered image identifying the location of the Delphin2 AUV actuators.

The vehicle can be quickly split into three main sections plus the fairings, Figure 4.3, and then transported in a medium sized car with its support equipment. The vehicle has one main pressure vessel in the middle of the vehicle, which contains the majority of the system electronics and a rechargeable nickelmetal hydride (NiMH) battery that is used as the sole power source for the vehicle. The front section contains two thrusters (one vertical and one horizontal), the sonar, altimeter and two colour cameras. The rear section contains two thrusters (one vertical and one horizontal) and the tail unit. The majority of the AUV is depth rated to 100 metres but the cameras and pressure transducer are rated at 50 metres so the Delphin2 AUV is currently rated to 50 metres.

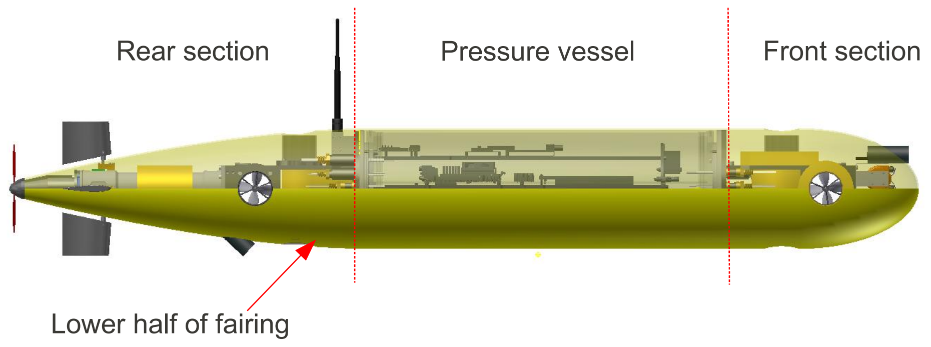


Figure 4.3: Three major components of the Delphin2 AUV.

The hull for Delphin2 is a scaled version of the hull used on the Autosub6000 AUV, built at the National Oceanography Centre Southampton [11]. This hull design was chosen for two reasons; it has good hydrodynamic performance and the capabilities of the Delphin2 AUV can be compared to an operational oceangoing AUV.

The software on the Delphin2 AUV is open-source, using Robotic Operating System (ROS) [76] as the platform for software development and Ubuntu 10.04 long term support (LTS) for the operating system [77]. Communication between the operator and AUV is achieved via Wi-Fi whilst the vehicle is on the surface and within 250 metres of the operator (with a laptop and outdoor Wi-Fi antenna). MATLAB is used on the operators laptop to handle communications between the AUV and operator (using SSH protocol) enabling the operator to download and view operational data. Software code is edited on the operator's laptop then sent to the AUV, the SSH protocol handles error checking.

Table 4.1: Overview of Delphin2

Number of actuators	9
Length	1.96 m
Diameter	0.26 m
Depth rating	50 m
Endurance	8 hrs
Maximum speed	+1.2 m/s
Minimum speed	-0.2 m/s

4.2 Mechanical Design

The major mechanical components of Delphin2 will be discussed in this section; the pressure vessel, the front and rear sections and the hull.

Pressure Vessel

The pressure vessel houses the majority of the vehicle's electronics and the battery (see Section 4.4), and has a diameter and length of 0.254 and 0.7 metres respectively. The pressure vessel includes five main mechanical components; the outer cylinder, two end flanges and two end plates. The outer cylindrical wall is 6 mm thick and made from 6082 T6 aluminium. The end flanges are located in the cylinder at both ends using interference fits to secure their position and two o-rings ensure each joint is water-tight. Each end plate is bolted onto each end flange using four M6 bolts, and an o-ring face seal ensures the joint is watertight, see Figure 4.4. To reduce corrosion of the pressure vessel the components that are in contact with water are anodized using chromic acid and two sacrificial zinc anodes are attached to the pressure vessel end plates and are electrically connected to the entire pressure vessel. Rubber molded connectors are used to pass signals and electrical power through the pressure vessel end plates [78].

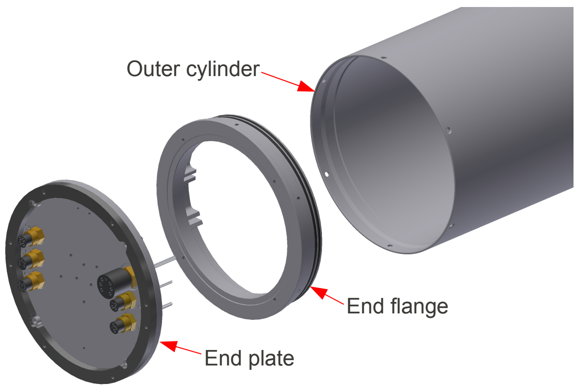


Figure 4.4: Exploded view of one end of the pressure vessel; the end flange inserts into the outer cylinder and the end plate bolts to the end flange.

Two pairs of rails run the length of the pressure vessel at two different levels. These rails are used to guide and support the electronics and battery shelves, Figure 4.5. The battery is located on the bottom shelf and is as low as possible within the pressure vessel so as to lower the centre of gravity of the vehicle, thus improving the static stability of the vehicle. In order to remove the battery first the front end plate is removed and the two bolts holding the battery are unscrewed. The battery is then disconnected from the electronics shelf and slides out the front of the pressure vessel. Charging of the battery can be done whilst it is in situ without opening the pressure vessel. The position of the battery can be adjusted fore and aft within the pressure vessel so to adjust the trim of the vehicle.

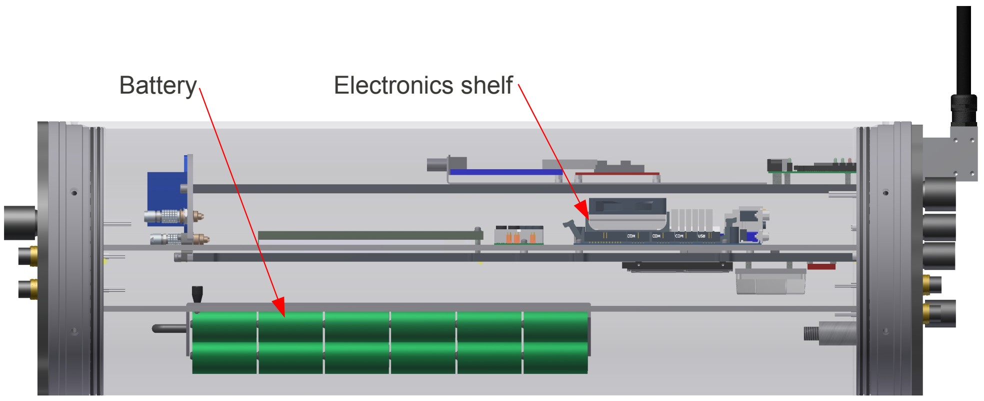


Figure 4.5: Side view of the pressure vessel with the outer cylinder set as translucent. Note the position of the battery and electronics shelf.

The electronics are mounted on the top shelf, which is bolted to the rear end plate. To remove the electronics shelf first the front end plate is removed and the connections with the end plate and battery disconnected. The bolts securing the rear end plate are then removed and the electronics shelf (still connected to the rear end plate) slides out.

One disadvantage with this design is that the inner diameter of the end flanges restrict the dimensions of the electronics shelf and battery, therefore not making use of all the internal space of the pressure vessel. A different flange design may help reduce this problem or the empty space could be filled with stiffener rings to improve the depth rating of the pressure vessel.

4.2.1 Thruster Units

There are two thruster units on the Delphin2 AUV; one making up the front section and the other as part of the rear section. The units bolt onto the front and rear of the pressure vessel using four M8 bolts each. Each thruster unit comprises of a plastic frame, machined from sheet Acetal, onto which a vertical and horizontal thruster is bolted. Both thruster units are identical in design, therefore reducing manufacturing costs and the number of spare components. Several threaded holes are located around the unit frame which can be used to mount different sensors and devices.

4.2.2 Tail Unit

The tail unit is located at the aft of the AUV and is bolted to the rear thruster unit using six M5 bolts. It comprises of one pressure vessel, four control surfaces and the rear propeller. The pressure vessel contains four linear actuators, a brushless DC motor and gearbox, magnetic couplings, and the control electronics. The pressure vessel is made from 6082 T6 aluminium that has been hard anodized, Figure 4.7, the other components are made from a mix of 316 stainless steel and Acetal plastic. A zinc anode is fitted to the tail unit to reduce corrosion.

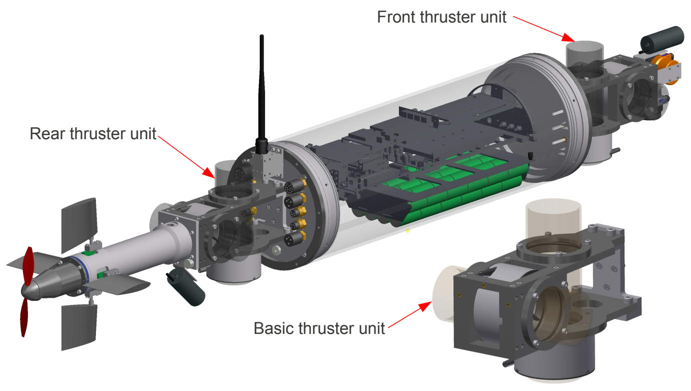


Figure 4.6: Thruster units on Delphin2. Both the front and rear thruster units are of identical design, with one vertical and one horizontal tunnel thruster each.

Each linear actuator is mounted on separate shelves made from 6082 T6 aluminium. The linear motion of the linear actuators is converted into rotational motion using a slider-crank mechanism. The rotational torque from the slider-crank mechanism is applied, through the pressure vessel wall, to the control surface shaft using a magnetic coupling.

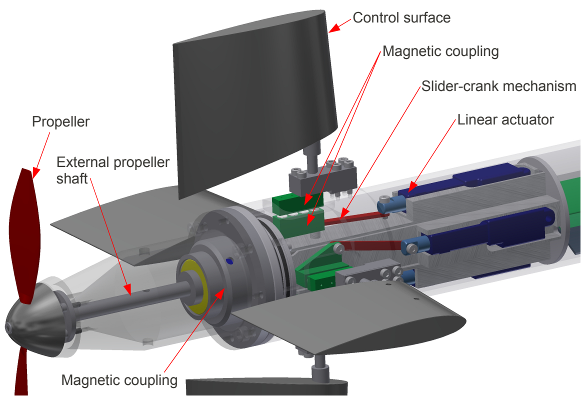


Figure 4.7: Tail unit with pressure vessel and rear prop-box set to be translucent. Note the location of the linear actuators and associated mechanisms. The propeller motor is located between the linear actuator shelves.

The brushless DC motor is coupled with a 50:1 gearbox and provides rotational torque to the rear propeller using a magnetic coupling. It is mounted in the centre of the pressure vessel between the linear actuator shelves. The use of the magnetic couplings, for both the propeller and control surfaces, improves the reliability of the tail unit substantially (compared to an oil-filled design with dynamic seals).

4.2.3 Fairings

The fairings used on Delphin2 are made up of two halves; a top and bottom half. Each half is located in position using four M5 bolts. The fairings have been manufactured using glass re-enforced plastic (GRP).

4.3 Actuators

4.3.1 Through-body Tunnel Thrusters

The four thrusters used on the Delphin2 AUV are all of identical design, and are a scaled version of those presented in [2]. Each thruster has two main components; the thruster motor and the propeller, Figure 4.8.

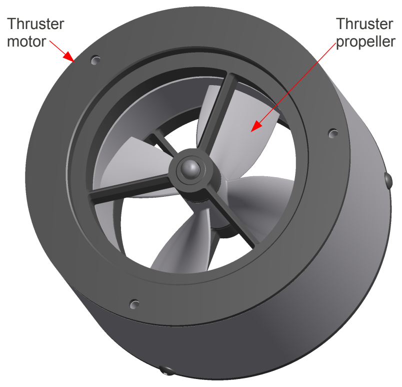


Figure 4.8: 70 mm rim-driven thruster used on Delphin2. Note the location of the thruster motor and propeller.

The motor is on the outside of the thrusters, with the motor coils potted in epoxy. The propeller is 70mm in diameter with a pitch ratio of 1.4 at 70% of the propeller radius. On the outside of the propeller are magnets that are driven by the external coils in the motor. This design provides a compact and reliable thruster that is capable of operation in depths far beyond the Delphin2 AUVs capability. The thrusters can operate at speeds of $\pm$3000 rpm, producing $\pm$28 N of thrust.

4.3.2 Control Surfaces

There are four identical control surfaces at the rear of Delphin2. These use the NACA 0014 foil profile and can be independently operated to angles of $\pm 30°$ relative to the vehicle. The forces generated by the control surfaces are proportional to the square of the inflow velocity of the fluid, therefore at low speeds the forces from the control surfaces are insufficient to control the vehicle.

4.4 Sensors and Electronics

4.4.1 Central Computer

The central computer is used to run the control software on the vehicle. Most of the sensors and actuator electronics are directly connected to the computer using USB or RS-232 connections. The computer has a mini-ITX form factor, a compact industry-standard design and therefore enabling the computer to be easily upgraded without substantial modifications to the electronics shelf. The computer has an Intel® Atom D525 dual-core processor 1.8 GHz, with 3 GB of RAM and a 500 GB hard drive. Peak power consumption of the computer is approximately 25 W.

4.4.2 Battery and Power Management

A 30 Ah, 21.6 V (nominal) nickel-metal hydride (NiMH) battery is used as the sole power source for the Delphin2 AUV. The battery is located at the bottom of the main pressure vessel and is connected to the electronics shelf. There are two power connectors on the rear pressure vessel end plate; one charging line and one shore power line. The charging line is for charging the battery, and the shore power line provides direct power to the vehicle. The direction of current flow on the different power lines, and from the battery, is managed using a diodes on a custom electronics board. Two reed-switches are connected to this board and are used, along with transistors, to switch power to the electronics and actuators on and off. The reed switches are enabled using magnets on the outside of the pressure vessel. Total hotel load of the Delphin2 AUV is approximately 30 W.

4.4.3 Communications

Communication to the AUV is performed using a Wi-Fi or Ethernet connection. For autonomous operation Wi-Fi is used, whilst the vehicle is on the surface, to transmit code and download data from the vehicle. A wireless access point, with an external antenna, is connected to the central computer using an Ethernet port. Wi-Fi range between the AUV and operator laptop (with a 1 W amplified antenna) is weather dependant, but on a clear day is up to 250 metres.

The Delphin2 AUV can be operated in ROV mode using an Ethernet cable to provide communications between the vehicle and operator laptop. An SMS modem is currently being integrated into the vehicle and will provide long-range low-bandwidth communication with the vehicle when it is on the surface and has GSM network reception.

4.4.4 Thruster and Tail Unit Control Electronics

Thrusters

The thrusters are controlled using a six channel motor controller (only four channels are used), designed and supplied by the same manufacturer as the thrusters [79]. The controller uses the sensorless control method to measure and control thruster speed, this involves measuring the back-emf generated by the thrusters when they are are spinning. One disadvantage of this method is that the back-emf generated at slow speeds is too low, resulting in a minimum thruster speed of ±450 rpm.

Tail Unit

The control systems with the tail unit comprise of a micro-controller, a motor controller and two sensors. The basic architecture of the electronics is shown in Figure 4.9. The micro-controller is used to process demands from the central computer and implement those demands to the actuators, and then return actuator feedback to the computer. The position's of the linear actuators are set by a voltage (0-5 V) from the micro-controller, and feedback is returned to the micro-controller also as a voltage (0-3.3 V) and read using its analogue-to-digital converters (ADC).

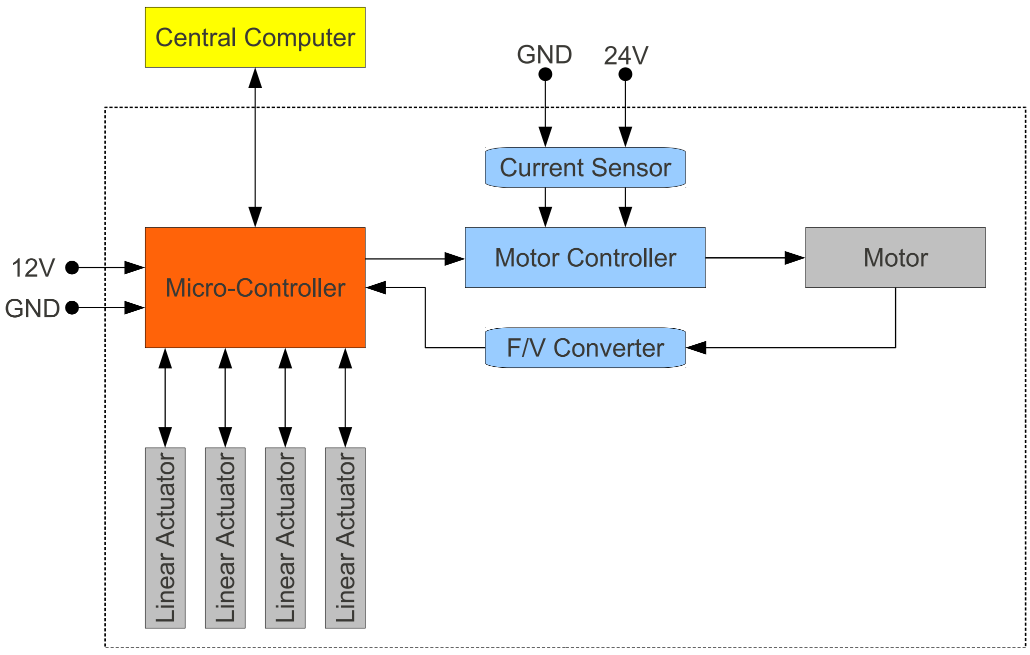


Figure 4.9: Architecture of tail unit control electronics.

The motor is controlled by a motor controller that receives a pulse-width-modulation (PWM) input signal from the micro-controller. A frequency-to-voltage converter chip is used to convert the digital pulses from the motor's Hall sensors to an analogue voltage proportional to motor speed. A current sensor is used to measure the electrical current flowing through the motor controller, and outputs a voltage signal. Both the motor speed and current voltages are measured using the micro-controller's ADC channels. Although these sensors do provide information of motor speed and current draw, both signals are extremely noisy and not of good enough quality to perform closed loop speed control of the motor. This measurement noise may be reduced by the addition of a suitably designed low-pass filter (e.g. RC seriers circuit) to help filter the high frequency components on the electrical signal. Further improvements in the tail section electronics are described in the conclusions chapter of this Thesis.

4.4.5 Sensors

Without sensors an autonomous vehicle would be unable to navigate or collect scientific data, thus rendering the vehicle ineffective. The sensors that are permanently fitted to the AUV are used by the control systems for navigation and manoeuvring. There are no sensors permanently fitted on-board the Delphin2 AUV whose sole purpose is for scientific data collection (e.g. conductivity, temperature and depth (CTD) sensor). The location of the sensors on-board Delphin2 are shown in Figure 4.10.

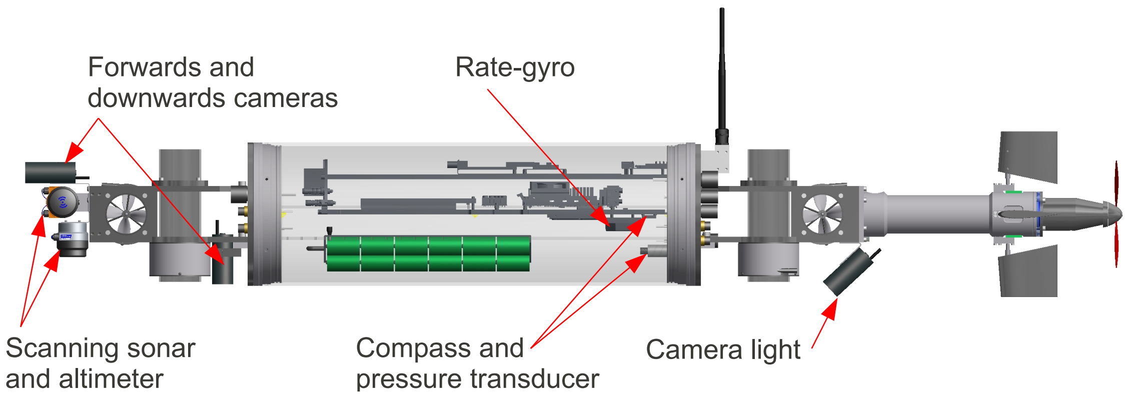

Forwards and downwards cameras
Rate-gyro
Scanning sonar and altimeter
Compass and pressure transducer
Camera light

Figure 4.10: Locations of the sensors on Delphin2.

Compass and Pressure Transducer

The compass on Delphin2 is a three-axis tilt-compensated digital compass with an ADC channel. It has three magnetometers and three accelerometers, that together provide vehicle bearing relative to magnetic north, and its pitch and roll angles. The ADC channel is used to measure the voltage output from the pressure transducer (rated: 0-5 bar), and in turn, this information is used to determine the vehicle's depth below the free surface.

Rate-Gyro Sensor

A rate-gyro sensor is fitted horizontally within the main pressure vessel, near to the centre of gravity of the vehicle, and provides a measurement of the yaw velocity. Future development of this sensor will include combining the heading signal from the compass with the integrated output of the rate-gyro using a Kalman filter. This should reduce the magnitude of the errors between magnetic and true north caused by fluctuations in the magnetic field due to the presence of magnetic objects near to the vehicle.

Scanning-Sonar and Altimeter

Both the scanning-sonar and altimeter are used to track the distance between the vehicle and sea-bed using acoustic back-scatter methods. This information can be used by the control system to maintain a defined altitude between the vehicle and seabed. The altimeter performs badly when operating at distances less than 0.5 metres, often outputting a much larger distance. This must be taken into account when designing bottom tracking algorithms.

GPS sensor

A global positional system (GPS) sensor is used to provide the location (latitude and longitude) of the AUV whilst it is on the surface. When submerged, the AUV estimates its location using dead-reckoning.

Cameras

There are two analogue colour CCD (charge-coupled device) cameras on Delphin2; one pointing forwards and one downwards. These can be used, along with image processing software, to locate objects of interest near to the AUV (e.g. an underwater pipe), or collect topography data of the seabed. The analogue video signals from the cameras are digitized using a four channel frame-grabber [80] that connects to the central computer using one USB connection. The cameras are operated using the standard V4L2 application programming interface (API). The cameras can be operated with resolutions of 704x576 or 352x288, with interlaced or de-interlaced frames respectively.

A light is fitted to the rear thruster unit and is used to help illuminate the seabed and improve image quality for the downwards pointing camera. The light contains six 1 W high intensity light emitting diodes (LEDs) that can be collectively switched on and off from the central computer.

4.5 Software

The software on the Delphin2 AUV is used for the autonomous operation of the vehicle including; artificial intelligence (AI), navigation, manoeuvring, sensor and actuator interfacing, and fault detection.

Robotics Operating System (ROS), an open-source robotics platform [76], provides the underlying functionality of the software on Delphin2. The software comprises several nodes, which are effectively standalone programs that communicate with each other and together make up, along with one central library, the Delphin2 software. Each node has a primary function, for example the compass sensor node reads data from the compass USB serial port and processes the data into usable information that can be read by other nodes.

The ROS platform provides the communication functionality that enables the nodes to publish information to what are called topics. Nodes that require information, from another node, can then subscribe to these topics. Figure 4.11 provides an example of the communication method; Node A publishes information to a topic and then Node B subscribes to this topic and receives the information from Node A.

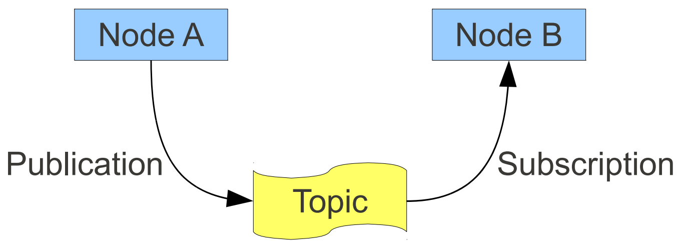


Figure 4.11: Example of a node publishing information to a topic and a second node subscribing to the same topic.

The nodes can be written in either Python or C++ programming languages (other languages are currently being developed). On Delphin2, all the code is written using the high-level programming language Python. The reason for this is that Python enables rapid code development due to its relatively simple syntax, availability of open-source libraries and that the code does not require compiling. One disadvantage of using Python over C++ is that its computational efficiency is generally lower. Therefore to achieve equivalent performance using Python compared to C++ requires a more powerful computer that will, in general, use more electrical power. As Delphin2 is designed as a research platform, electrical power consumption is not of primary concern and instead the rate of software development takes precedence. If Delphin2 was developed as a commercial vehicle then it would be advantageous to use C++ and a less powerful computer.

4.5.1 Software Architecture

The architecture of the software on Delphin2 can be described as hierarchical with a mission planner at the highest layer and the actuator and sensor drivers at the lowest layer, Figure 4.12. The main components of the software structure will be described upwards from the lowest layer.

The sensor and actuator drivers are nodes that handle the communication between the hardware and computer. All of the hardware (with the exception of the frame-grabber) on Delphin2 interfaces with the computer using the RS-232 serial protocol, using either a direct serial connection or a USB-

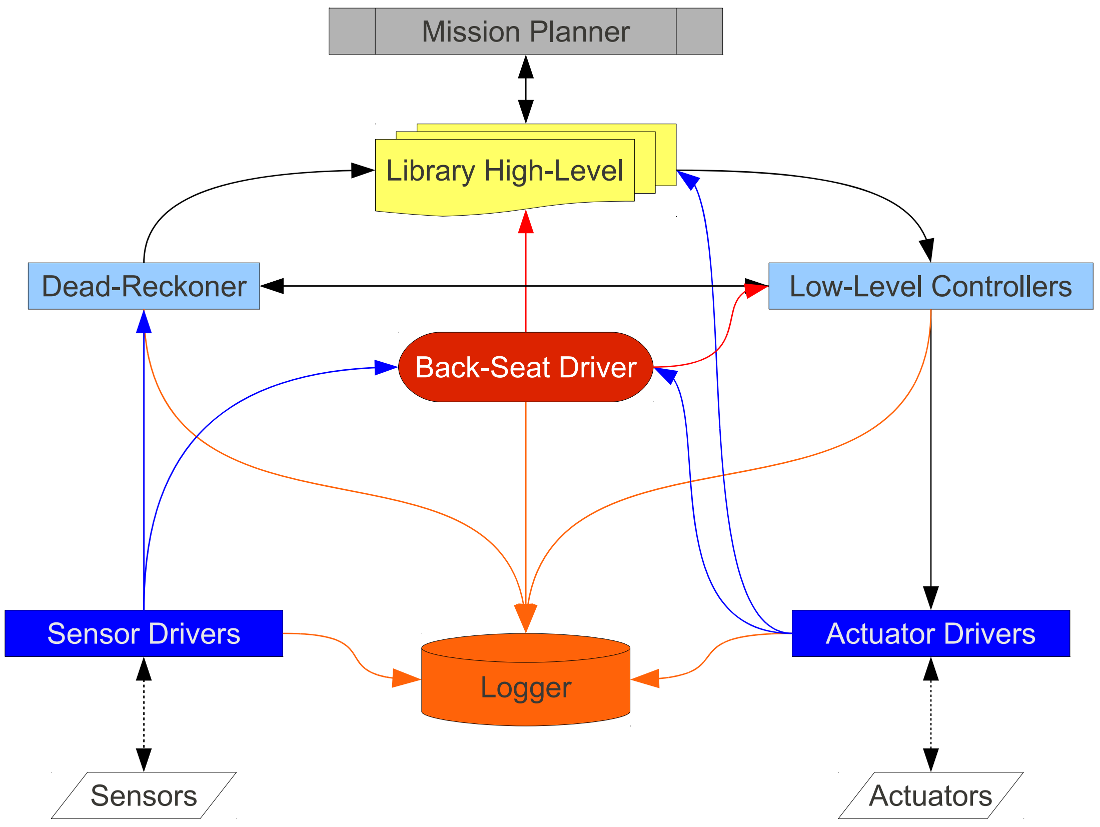


Figure 4.12: Flow-diagram illustrating the architecture of the Delphin2 AUV software.

serial adapter. Most of the sensor nodes only read information from the sensors and do not transmit. The actuator nodes transmit information to hardware, such as thruster demands, and also read information returned by the hardware, such as thruster speed feedback. The processed information read by the actuator and sensor nodes are published to topics specific to each node.

The second layer includes three main components; the dead-reckoner node, the low-level controller nodes, and the back-seat driver node. The dead-reckoner node computes navigational information such as (X,Y,Z) location and surge velocity. It does this by subscribing to information from the sensors and low-level controllers and then combining this information with a five DOF non-linear hydrodynamic model of the Delphin2 AUV. The performance of the dead-reckoner is critical for navigation underwater when (X,Y) location information is not available (GPS signal is available only whilst the AUV is on the surface).

There are typically two low-level controller nodes; a heading controller node, and a depth and speed controller node. As these controllers are the main focus of this research, the number of nodes varies throughout this work. The low-level controllers compute suitable actuator set-points so as to minimize the error between high-level demands (e.g. go to 2 metres depth) and the current vehicle states.

The back-seat driver node continuously monitors sensor, actuator and mission information to ensure all the critical parameters are within predefined limits. If any of these limits are exceeded, such as maximum depth, then the back-seat driver publishes an error flag. The library high-level (described in the next paragraph) and the low-level controllers both subscribe to the back-seat driver topic. In the event of an error flag occurring, the low-level controllers immediately switch off the actuators (thrusters and rear propeller) and the library high-level informs the mission planner of the error so it can determine how to proceed.

The top layer will be described as two sub-layers; on the bottom sub-layer is the library high-level and on the top is the mission planner. The library high-level is a compilation of functions and system information that can be called by the mission planner. The purpose of the library high-level is to simplify and reduce the quantity of code in the mission planner, thus reducing the likelihood of a coding error.

The mission planner can then call functions, for example 'go to 2 metres depth', and the library high-level will publish the depth demand as well as the flags that enable the depth controller and actuator drivers. The mission planner contains states that handle tasks within the mission planner. The states are similar to nodes but are directly connected to the mission planner node. When each task finishes it returns one of three possible outcomes; succeeded, aborted or pre-empted. Succeeded means that the task completed successfully, aborted means it did not complete successfully (often due to a predefined time-out criterion being exceeded), and pre-empted means that the task has stopped due to an error flag from the back-seat driver. The next task is defined by the outcome of the previous task. Future development of the mission planner will include more outcome options so as to enable a more reactive, and less linear, planner.

The logger node is not defined within a specific level. It subscribes to most of the available topics and writes the published information to individual files in the comma-separated values (CSV) format. These files are post-processed to analyse the vehicle performance.

4.6 Conclusions

To conduct this research the Delphin2 AUV has been designed and built. It is a hover-capable torpedo shaped AUV that, when moving with a positive surge velocity, is over-actuated with more actuators than degrees-of-freedom. The vehicle has four through-body tunnel thrusters used for controlling the vehicle at zero and low speeds. Four control surfaces are used to control the heading, pitch and roll angles of the vehicle at medium to high speeds. A rear propeller provides forwards propulsion. The mechanical and software design has been described and the available sensors introduced.

Chapter 5

Physics Model of the Delphin2 AUV

5.1 Introduction

In this section a mathematical model of the Delphin2 dynamics will be derived. This model will be used later in this work to simulate the vehicle performance and in the design of different MPC controllers. This research is not focused on the mathematical modelling of AUVs but instead the control of them, therefore only the key components of the AUV will be investigated.

5.1.1 Axis System

The standard nomenclature for modelling marine vehicles will be used [1], Figure 5.1 and Table 5.1. The focus of research is the development of control methods for the z, y and q axes, therefore only these three degrees of freedom shall be modelled.

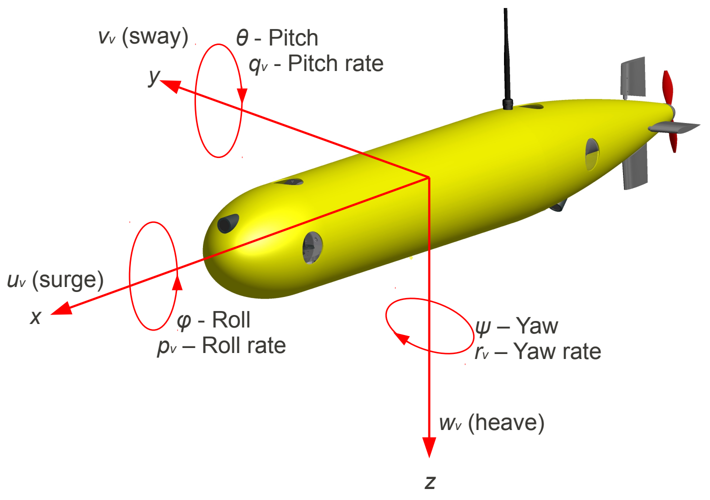


Figure 5.1: Motion variables of a marine vehicle [1].

5.1.2 AUV Model

The equations of motion for submarines can be easily applied to slender-bodied AUVs. Since the late 1960s there have been many variations of these equations however most can be traced back to those developed in [81]. The equations are non-linear and are modelled either algebraically [82] or in matrix form [83]. The axis system is the standard motion variables used by marine vehicles [1], however, the subscript v has been included to distinguish these terms from those used later in this work. The attitude of the vehicle is described using Euler angles relative to Earth while body-fixed terms are used to describe the linear and angular velocities of the vehicle.

As previously stated, only the dominating components of the system will be modelled. Therefore the equations of motion that will be used to model the Delphin2 AUV in this work are:

Table 5.1: Notation used when modelling marine vehicles for all six degrees of freedom [1]

DOF		forces and moments	linear and angular velocities	positions and Euler angles
1	motions in the x-direction (surge)	X	u_v	x
2	motions in the y-direction (sway)	Y	v_v	y
3	motions in the z-direction (heave)	Z	w_v	z
4	rotation about the x-direction (roll)	K	p_v	ϕ
5	rotation about the y-direction (pitch)	M	q_v	θ
6	rotation about the z-direction (yaw)	N	r_v	ψ

$$\dot{u} = \frac{1}{m_u}\left[T_{prop}\cos\theta + T_{vf}\sin\theta + T_{vr}\sin\theta + X_D\right] \tag{5.1}$$

$$\dot{w} = \frac{1}{m_w}\left[W - B + T_{prop}\sin\theta + T_{vf}\cos\theta ... \right.$$
$$\left. + T_{vr}\cos\theta + Z_{hull} + Z_{cs} + Z_D\right] \tag{5.2}$$

$$\dot{q} = \frac{1}{I_q}\left[x_{Tvf}T_{vf} + x_{Tvr}T_{vr} + M_{hull} + M_{cs} + M_R + M_D\right] \tag{5.3}$$

where the $\dot{u_v}$, $\dot{w_v}$ and $\dot{q_v}$ terms are the surge, heave and pitch accelerations. The m_u, m_w and I_q terms are the mass and inertia terms for the x, y, and z axes. The W and B terms are the weight and buoyancy of the AUV. The T_{prop}, T_{vf}, T_{vr} terms are the forces from the propeller, front and rear vertical thrusters. The x_{Tvf} and x_{Tvr} are the moment arms for the front and rear vertical thrusters. The Z_{hull}, Z_{cs}, Z_D are the hydrodynamic forces produced by the hull, control surface and damping along the z axis. The M_{hull}, M_{cs}, M_R and M_D terms are the moments about the y axis generated by the hull, the control surfaces, restoring moment and hydrodynamic damping.

In this chapter the terms used in equations (5.1) to (5.3) will be found using experimental data. First the AUV is tested in a wind-tunnel to determine the drag, lift and moment coefficients for the hull and control surfaces. Secondly, the thrusters will be modelled by performing and evaluating bollard pull tests. Finally, semi-autonomous open water tests are conducted and further performance data extracted from the results.

5.2 Hull and Control Surfaces

In this section the hull and control surfaces from the Delphin2 AUV will be introduced. The electro-mechanical system used for setting the control surface angles will be described and the reasons for choosing the foil profile given. The performance of the control surfaces, along with the hull, will then be evaluated and modelled using existing experimental data from wind-tunnel tests.

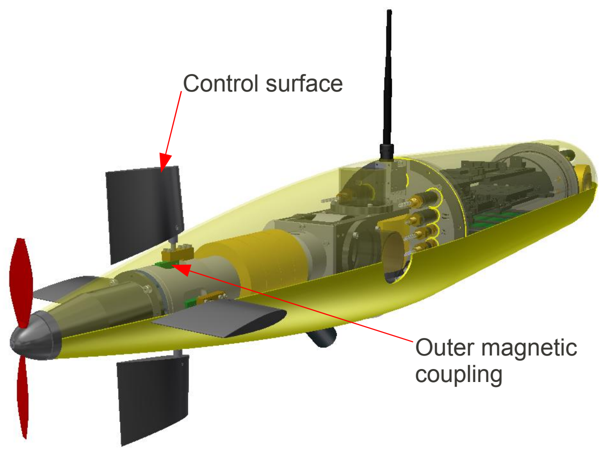


Figure 5.2: Computer rendered image of the rear of the Delphin2 AUV. Note the location of the control surfaces and the magnetic couplings used to transmit the rotational torque between the control surface and actuator inside the tail unit pressure vessel.

There are four identical, independently operated control surfaces fitted to the rear of the Delphin2 AUV, Figure 5.2. The angle of incidence relative to the body frame, δ, of each control surface is adjusted using a linear actuator within the sealed tail section pressure vessel. The force of the linear actuator is translated into rotational torque using a simple slider-crank mechanism and this torque is then applied to the control surfaces via a magnetic coupling.

Several foil profiles were evaluated using XFOIL, a 2D CFD program [84]. Foil profiles specifically designed for operating in low Reynolds number flow gave excellent performance at low angles of attack with high lift-drag ratios, but they also stalled at relatively low angles of attack. As the Delphin2 AUV has been designed for research into control system design, it was decided that predictable foil performance is more important than slight improvements in lift and drag characteristics. The NACA-0014 profile was chosen for its overall performance attributes, in particular its near linear gradient of lift coefficient C_L against angle of attack α, and its relatively high angles of attack before stall occurs.

To ensure the AUV is dynamically stable, the dimensions of the foil were iteratively varied to

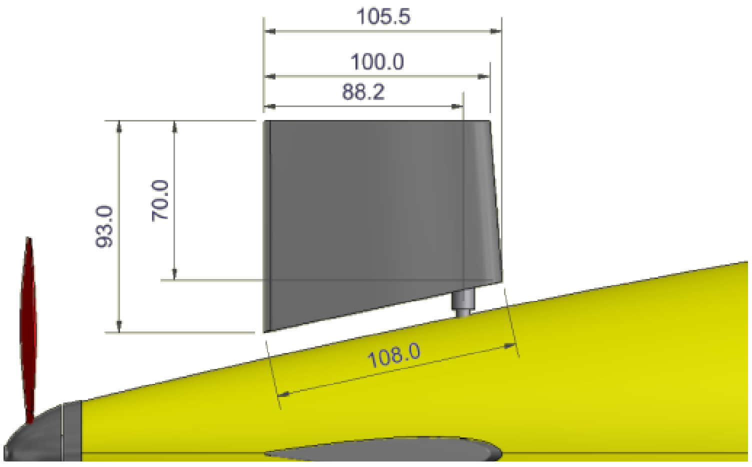


Figure 5.3: Dimensions of the control surfaces on Delphin2.

ensure that the stability criterion, [85], was met:

$$N'_r Y'_v - N'_v \left(Y'_r - m' \right) > 0 \tag{5.4}$$

The hydrodynamic derivatives in equation (5.4) are estimated using equations from [85] and so the calculated value could have a large error. Foil dimensions are therefore taken for a stability value significantly above zero (0.61) and that also gave a suitable form-factor for the vehicle. The dimensions of the foils used on the Delphin2 AUV are shown and given in Figure 5.3 and Table 5.2 respectively.

Table 5.2: Specification for the control surfaces on the Delphin2 AUV.

Foil profile	NACA-0014
Min/max rudder angle	$\pm 30°$
Foil area	8.4×10^{-3} m^2
Mean chord	0.104 m
Mean span	0.0815 m
Mean root gap	0.01 m

5.2.1 Wind Tunnel Tests

To accurately calculate the lift, drag and moment coefficients for both the hull and control surfaces, data from [86] is analysed. In [86] the Delphin2 AUV was tested in the high-speed section of the 7'x5' wind-tunnel at the University of Southampton. The vehicle was mounted with faired struts on the side of the vehicle, and a rod attached to the rear vertical thruster tunnel, to a dynamometer [87] that is used to measure the forces acting on the vehicle along the x (drag) and z (lift) axes and about the y axis (pitching moment), Figure 5.4. The rear propeller was removed from the vehicle for these tests.

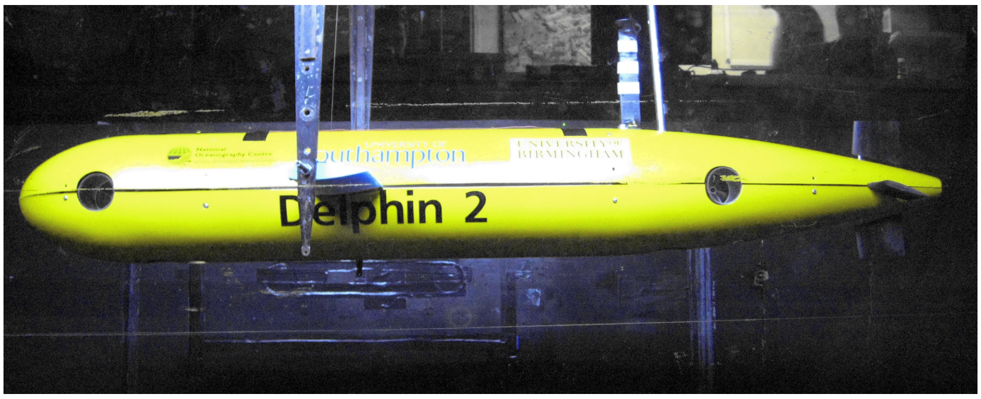


Figure 5.4: Delphin2 AUV mounted in the wind-tunnel at the University of Southampton.

The AUV was switched on during the tests and an automated script was used to cycle through the control surface angles $\pm25°$ at $5°$ intervals. There is a 2 minute delay between each control surface angle so as to ensure that the dynamometer has settled to a steady-state measurement. The AUV and wind-tunnel software were operated separately with communication between the AUV and the operator laptop performed using the standard Wi-Fi system. After 90 seconds at each control surface angle, the AUV provided an instruction (on the operator laptop screen) to record the forces and moments using the dedicated data acquisition system for the wind-tunnel.

The pitch of the vehicle was varied, from $-10°$ to $+2°$, by adjusting the length of the rear mounting rod, taking care to maintain a consistent angle of attack for the faired struts. The angle of attack for these tests is equivalent to the pitch angle of the vehicle and is measured using the on-board digital compass sensor within the main pressure vessel.

The wind speed in the tunnel was controlled using the dedicated wind-tunnel control systems. The appropriate wind speeds were calculated by scaling the speeds that Delphin2 is expected to operate at, in water, using Reynolds number similarity:

$$V_{air} = \left(\frac{\nu_{air}}{\nu_{water}}\right) V_{water} \tag{5.5}$$

where V_{air} is the air speed in the wind tunnel, V_{water} is the water speed, and ν_{air} and ν_{water} are the kinematic viscosities of air and water respectively.

Therefore, with both fluids at $10°C$ and at atmospheric pressure, the air speed needs to be 10.87 times greater than the equivalent water speed to ensure the Reynolds number remains similar. The wind speed was varied from 2.5 to 25 m/s, corresponding to water speeds of 0.23 to 2.3 m/s.

The recorded wind speeds are corrected for the solid blockage effects of the AUV and mounting struts using the approximation [88]:

$$\varepsilon_t = \frac{\Delta V}{V_{air}} = \frac{1}{4}\frac{\text{model frontal area}}{\text{test-section area}}, \tag{5.6}$$

where ΔV is the increase in wind-speed due to blockage and V_{air} is the unaltered wind speed. With a model frontal area of $7.99 \times 10^{-2} m^2$ and test-section area of $3.15 m^2$, the recorded wind-speed data is increased by 0.63%.

To accurately calculate the lift, drag and moment coefficients, the forces and moments generated by the flow past the mounting struts must be subtracted from the total forces and moments measured using the dynamometer. With the AUV removed, the forces and moments acting on the mounting struts are measured using the full range of air speeds. A polynomial is fitted to the data and used to correct the measurement values so as to remove the force and moments due to the mounting struts. Because of the difference in flow around the mounting struts, with and without the AUV in situ, the forces and moments will not be precisely equivalent and so there will be an error produced when subtracting these force and moment values.

Coefficients for lift, drag and pitching moment are used to model the hydrodynamic performance of the vehicle. Note that the coefficients for the control surfaces also use the dimensions of the vehicle hull. The coefficients are defined as:

$$C_L = \frac{Z}{\frac{1}{2}\rho L^2 v_{air}^2} \tag{5.7}$$

$$C_D = \frac{X}{\frac{1}{2}\rho \nabla^{2/3} v_{air}^2} \tag{5.8}$$

$$C_M = \frac{M}{\frac{1}{2}\rho L^3 v_{air}^2} \tag{5.9}$$

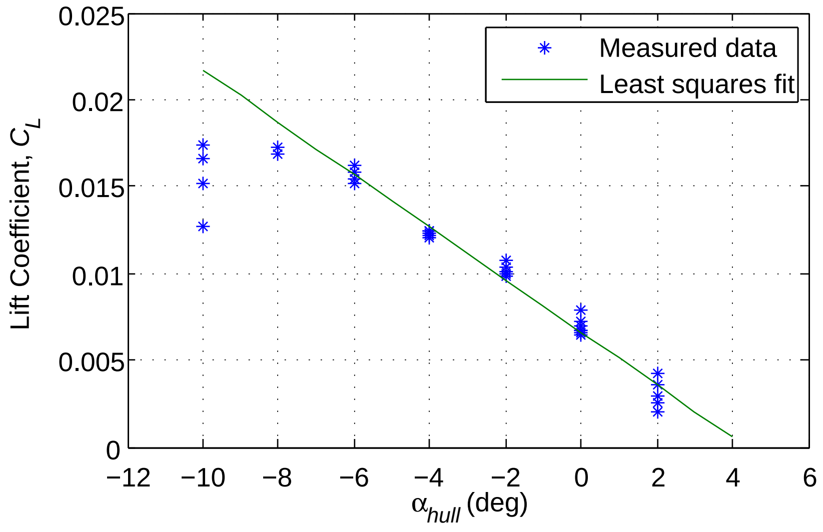


Figure 5.5: Lift coefficient for the Delphin2 hull with the control surfaces at zero degrees, against hull angle of attack.

First the performance of the hull will be evaluated then the control surfaces. Figures 5.5, 5.6 and 5.7 present the lift, drag and pitching moment against angle of attack. The control surfaces are set to 0° for all these data points. The first observation is that the lift and moment coefficients do not intercept zero at zero angle of attack as might be expected. The two main reasons for this are; the AUV is not perfectly symmetrical about the xy plane (due to the Wi-Fi antenna, camera holes, and manufacturing inaccuracies), and there are systematic errors on the pitch and control surface angles due to calibration and fitting inaccuracies.

Hull

A least squares fit has been performed on the lift data for angle of attack values from $-6°$ to $2°$, Figure 5.5. Below $-6°$ the measured lift coefficient values deviate from the linear fit. This is most likely due to the onset of stall. The linear fit, minus the constant (zero offset), is used for modelling purposes.

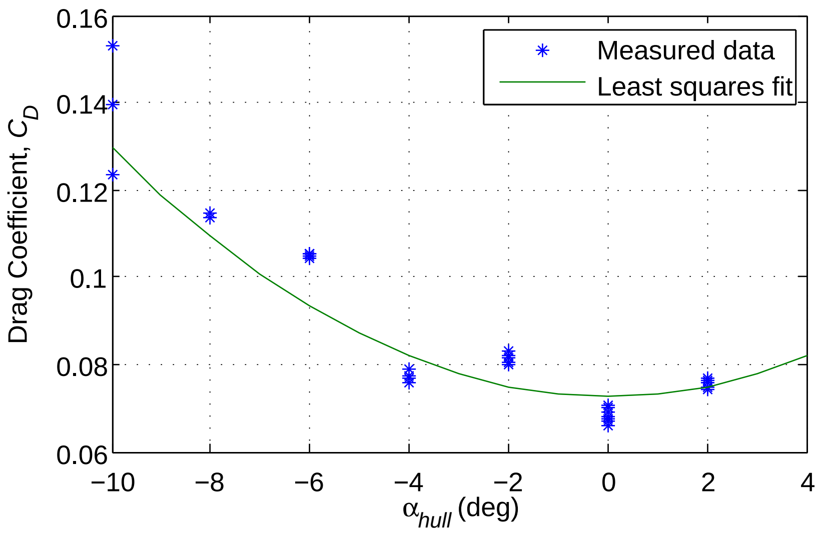


Figure 5.6: Drag coefficient for the Delphin2 hull with the control surfaces at zero degrees, against hull angle of attack.

The hull drag coefficients are fitted, using least squares regression, to an equation with a quadratic component plus a constant. The trend of the data is less consistent than the lift coefficient, resulting in a lower R^2 value of only 0.904. The values should however be accurate enough to develop a model for controller development.

The pitching moment coefficients show a negative trend, meaning that the moment is destabilizing. A linear fit has been calculated using the data points with angle of attack from $-4°$ to $2°$. Below $-4°$ the data points diverge from the linear fit but continue to decrease. This divergence from the linear fit is likely due to the onset of stall. As with the lift coefficient the constant term of the fit will be ignored for modelling purposes.

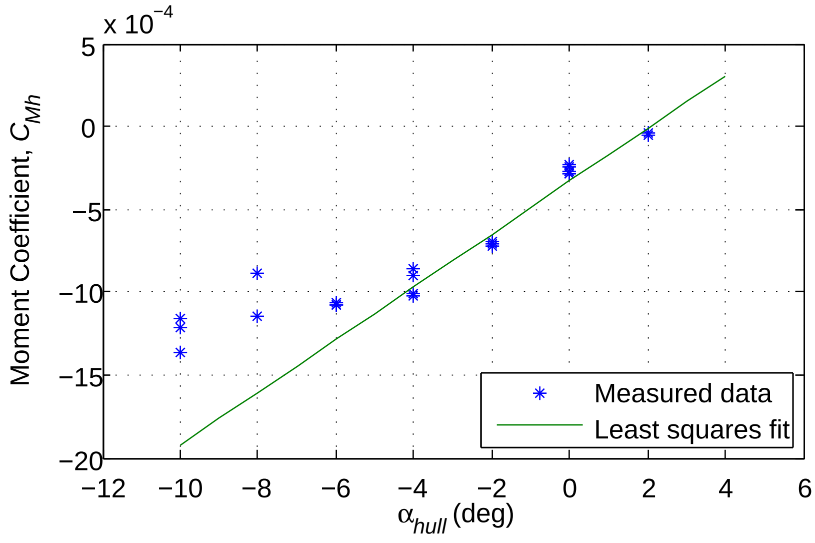


Figure 5.7: Pitching moment coefficient for the Delphin2 hull with the control surfaces at zero degrees, against hull angle of attack.

Control Surfaces

The forces and moments for the control surfaces are modelled separately from those of the hull. Therefore the mean average of the lift and drag forces, and the pitching moment, are subtracted from the data and then the effect of control surface angle, δ_{cs}, is evaluated.

As both the lift and pitching moment coefficients are closely coupled (pitching moment is the product of the lifting force and a moment arm), they both follow similar trends. Both the pitching moment and lift coefficients are fitted with linear equations for modelling purposes. The drag coefficients for the control surfaces are modelled using one quadratic component. It should be noted that the angle of minimum drag for the experimental drag data is not at an angle of attack of zero, hence there is a significant offset between the fitted curve and experimental data. Again for controller development this should be sufficiently accurate.

Table 5.3 presents a summary of the coefficients calculated using the wind tunnel data. It should be noted that the dominating coefficients are the lift and drag for the hull, and the pitching moment for the control surfaces.

5.3 Through-body Tunnel Thrusters

In this section the through-body tunnel thrusters, that are used on the Delphin2 AUV, are introduced. The Delphin2 AUV is fitted with two vertical and two horizontal through-body tunnel thrusters. The vertical thrusters generate forces and moments in the z axis and y axis respectively, therefore the vertical thrusters are used for depth and pitch control, Figure 5.11. The horizontal thrusters generate

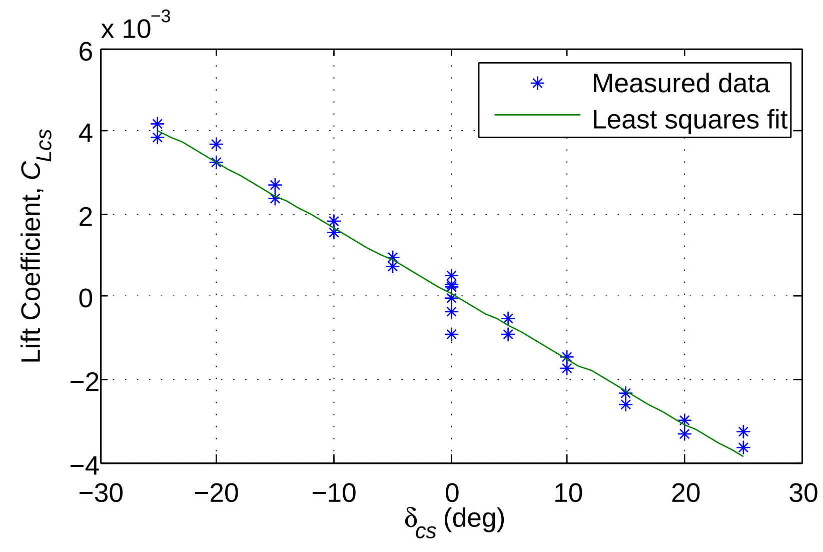


Figure 5.8: Lift coefficient for the Delphin2 control surfaces, against control surface angle of incidence.

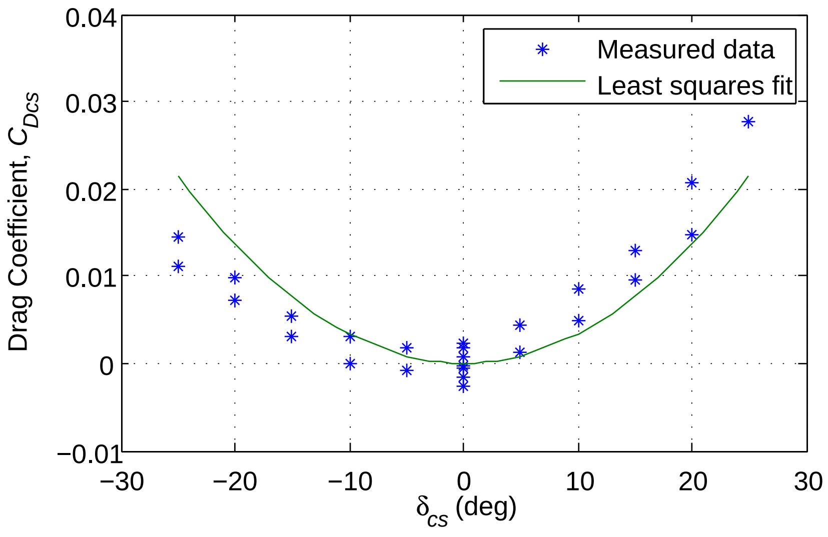


Figure 5.9: Drag coefficient for the Delphin2 control surfaces, against control surface angle of incidence.

forces and moments in the y axis and z axis respectively, and so are used for sway and yaw control. The generated moments are the product of the thrust force and the distance between the thruster and the centre of gravity (CG) of the AUV. The location of the CG is estimated using the CAD model of the vehicle, the subsequent distances used for calculating the moments produced by the thrusters can be found in Table 5.4.

The (thruster) propeller has a diameter of 70 mm and has a pitch ratio of 1.4 at 70% of the propeller radius. The propeller design is a scaled version of a 250 mm diameter propeller [2]. The thruster units, Figure 5.12, are fitted within 70 mm diameter Perspex tunnels and the propeller is

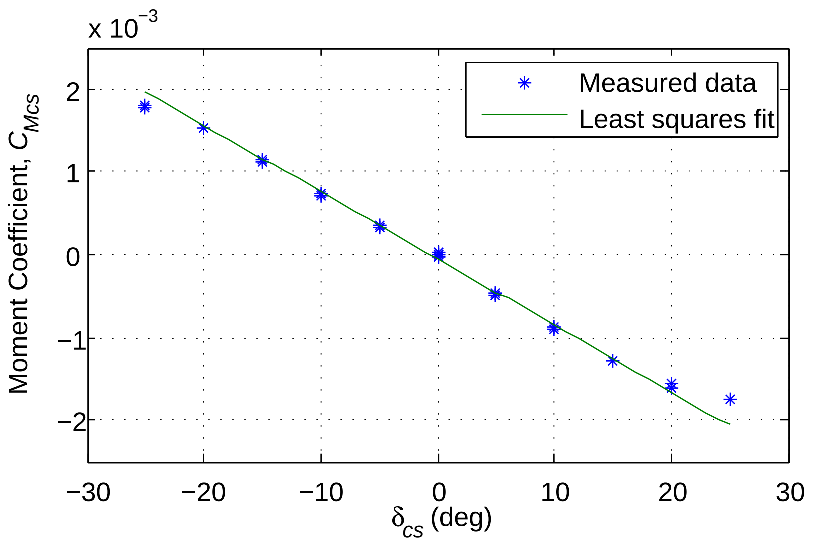


Figure 5.10: Pitching moment coefficient for the Delphin2 control surfaces, against angle of incidence.

Table 5.3: Hydrodynamic lift, drag and pitching moment coefficients for Delphin2.

C_{Lh}	$\left(-1.512 \times 10^{-3}\right) \alpha_{hull}$
C_{Dh}	$\left(5.719 \times 10^{-4}\right) \alpha_{hull}^{2} + 0.07265$
C_{Mh}	$\left(1.596 \times 10^{-4}\right) \alpha_{hull}$
C_{Lcs}	$\left(1.578 \times 10^{-4}\right) \delta_{cs}$
C_{Dcs}	$\left(3.421 \times 10^{-5}\right) \delta_{cs}^{2}$
C_{Mcs}	$\left(-8.057 \times 10^{-5}\right) \delta_{cs}$

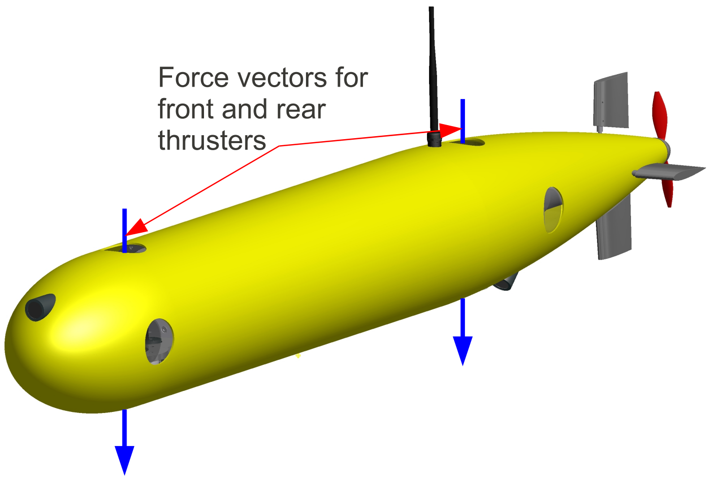


Figure 5.11: Illustration indicating the location and direction of force vectors for the front and rear vertical thrusters.

Table 5.4: Distances between centre of gravity and thrusters along x axis

x_{Tvf}	$0.55m$
x_{Tvr}	$-0.49m$
x_{Thf}	$0.65m$
x_{Thr}	$-0.59m$

driven by an externally fitted 250 W brushless DC motor which drives permanent magnets that are fitted to the propeller unit. This design enables the central hub of the propeller to be small, thus maximizing the area of the foil sections. The electronics used to drive the individual thrusters use sensorless control to control the thruster speed [79]. Thruster speed demands are provided to the electronics from the central AUV computer via RS-232 serial communication.

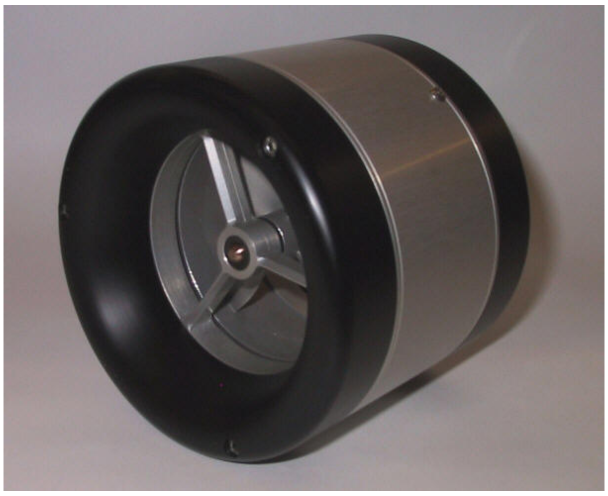
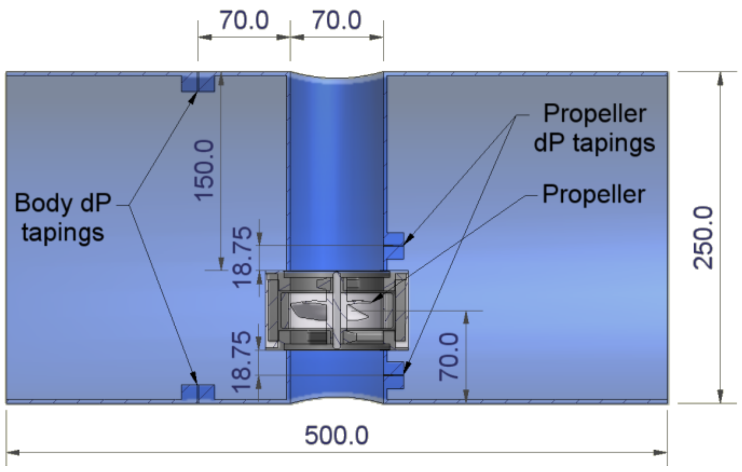


Figure 5.12: 70 mm diameter thruster used on the Delphin2 AUV. Note the black rims are removed when fitted to the Delphin2 AUV.

To successfully design a vehicle and its subsequent control systems, it is essential that an accurate model of the thrusters is known. The information required to build a mathematical model of the thruster dynamics can be found either by experimentation [89] or using computational fluid dynamics (CFD) [90]. As the flow, particularly when operating near the free surface, is unsteady and turbulent it has been decided that the necessary information will be found using experimentation. This decision has been taken as the computational time and the effort to produce an accurate result would be much more expensive using CFD than performing the experimentation.

5.3.1 Experimental Set-up

To measure the quasi-steady and transient performance of the tunnel thrusters a new experimental set-up has been developed. The tunnel thruster is mounted vertically in a frame along with an outer plastic tube of the same diameter as the Delphin2 hull (length: 500mm, diameter: 250mm), Figure 5.13. The experiments were conducted at the Lamont towing tank at the University of Southampton. This tank measures 30 $\times$ 2.4 $\times$ 1.2 metres deep. The water temperature remained constant at 5$^{\circ}C$ throughout all the tests.



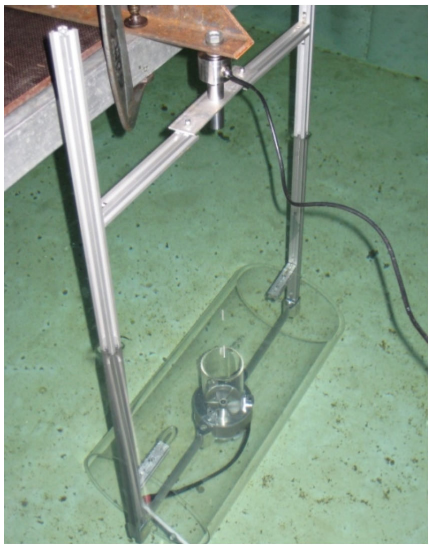


Figure 5.13: Experimental set-up with the vertical tunnel thruster in the Lamont towing tank at the University of Southampton.

The thruster control and data acquisition of the sensors has been conducted using the computer from the Delphin2 AUV. This enables the logging of thruster set-point, thruster speed, differential pressures, and the load cell voltage to be carried out on one system. To measure the output of the sensors that are not normally fitted to the Delphin2 vehicle, a data acquisition system (DAQ) with 16-bit resolution and a sample rate of 500 Hz is used.

A load cell, rated to 50 N, is used to measure the force on the set-up in the z axis. This is connected between the thruster assembly and the walkway across the tank. The output voltage of the load cell is amplified by a gain of 100 using a voltage amplifier so as to improve the effective resolution of the DAQ.

Two differential pressure measurements were taken during these tests; one across the propeller within the tunnel, and one across the top and bottom of the large outer tube, Figure 5.13. A differential pressure transducer (rated: 0 - 2500 Pa) was used to make these measurements. The transducer outputs a 0-5 V signal that is directly measured using the DAQ.

5.3.2 Quasi-Steady Model

The results of the quasi-steady thrust generated by the tunnel thruster unit will now be presented and modelled. In this experiment only the vertical thruster performance is measured, however, the horizontally mounted thrusters should, in most conditions, operate with similar performance.

The quasi-steady performance of the thruster can be modelled using the advance ratio, J, and the thruster coefficient, K_T:

$$J = \frac{V_a}{nD}, \quad K_T = \frac{T}{\rho n^2 D^4} \tag{5.10}$$

where V_a is the velocity of advance (m/s), n is the thruster speed (rps), D is the thruster diameter (m), T is the force produced by the thruster (N), and ρ is the fluid density (kg/m^3).

Figure 5.14 plots experimentally measured values of thrust against $\rho n^2 D^4$. The gradient of the linear fit in Figure 5.14 is the K_T value specific to this thruster design. The K_T value for the Delphin2 tunnel thrusters, at an advance ratio of zero, is 0.464.

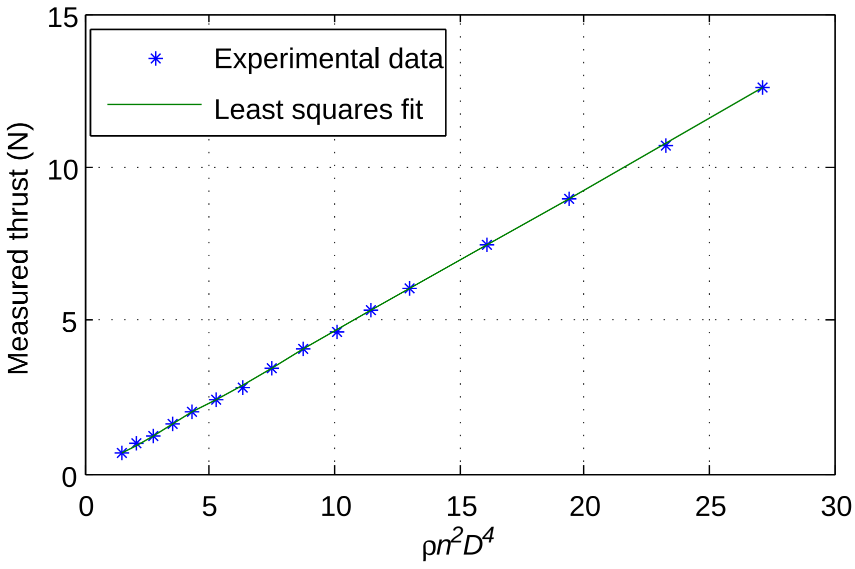


Figure 5.14: Experimental data of measured thrust against $\rho n^2 D^4$.

The effect of the advance ratio, J, on the thruster K_T value has not been measured for this thruster. However, the 70 mm thrusters used in this work are a scaled-down design of a 250 mm thruster for which the advance ratio data exists [2], Figure 5.15. The experimental set-up for the 250 mm thruster is different as it uses a short duct rather than the tunnel used with the 70 mm thruster. This explains some the slight variation in K_T values (0.464 and 0.42 for the 70 mm and 250 mm thrusters respectively). Assuming that the difference in set-up does not vary the advance ratio data substantially, the data for the 250 mm thruster will be taken to be the same as that of the 70 mm thruster. The advance ratio at which K_T equals zero, J_0, is taken from the 250 mm data, Figure 5.15.

Equation (5.11) demonstrates the correlation between K_T and the advance ratio. Here the value for x is found using a least squares fit to the experimental data. The values of J_0 and x are 0.55 and 0.99 respectively.

$$K_T \propto \left(1 - \frac{J}{J_0}\right)^x \tag{5.11}$$

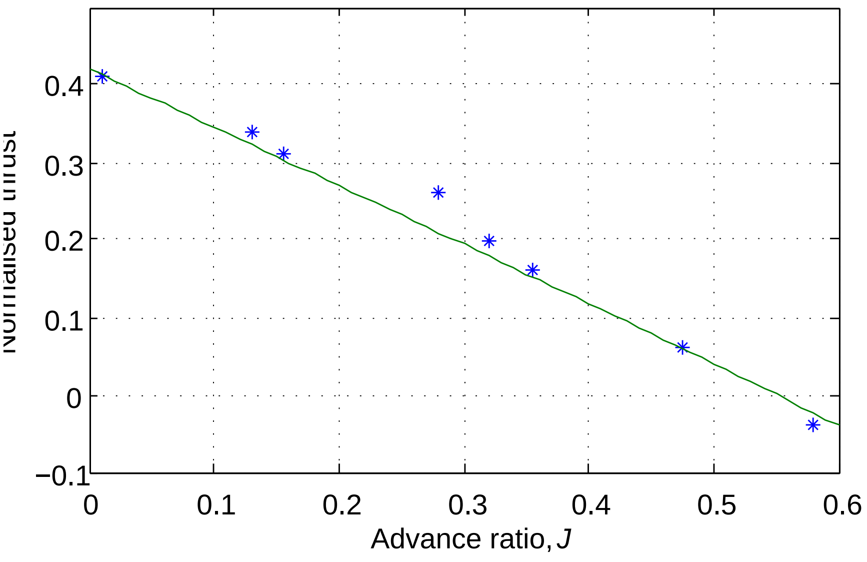


Figure 5.15: Thrust coefficient K_T against advance ratio J for the 250 mm thruster, data from [2].

5.3.3 Motor Model

In order to create a transient model of the thrusters, first the transient performance of the thruster motors must be modelled. Here the motor speed has been simplified as a first order system with an ideal delay:

$$n = \frac{e^{-sL}}{T_m s + 1} u(s) \tag{5.12}$$

where n is the motor speed ($rev.s^{-1}$), L is the time constant for the ideal delay, T_m is the motor time constant and u is the thruster speed demand.

The values for T_m and L are calculated using experimental data of step changes in speed demand, Figure 5.16. For the thrusters on Delphin2 values of 0.32 and 0.18 seconds for T_m and L closely match experimental data.

5.3.4 Transient Thruster Model

One method to calculate the force generated by the thruster when it is accelerating would be to use equations (5.10) and (5.11), thus assuming that the thrust generated is linearly proportional to the

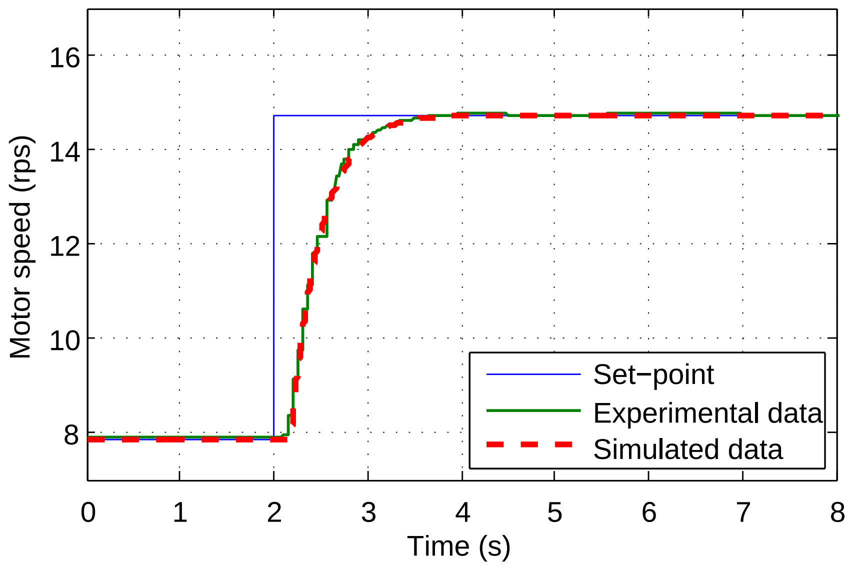


Figure 5.16: Experimental and simulated data for a step response in speed demand for the thruster motors.

steady-state thrust in all conditions. It has been noted in this work and in previous literature, [91] and [92], that the thrust rise time is faster than that of the motor rise time, and in many cases this leads to an overshoot of the steady-state thrust before settling to a steady-state value.

This can be explained by the difference in accelerations between the propeller and the fluid flowing through the thruster. This momentary difference in accelerations causes the angle of attack of the propeller foil to change from its steady-state value. This changes the lift generated by the foil, and thus the thrust, compared to the steady-state value.

Assuming the fluid has zero velocity before interacting with the thruster then the steady-state flow velocity, V_{ss}, at the propeller can be estimated as:

$$V_{ss} = \frac{1}{2}\left[\frac{8K_T}{\pi}n^2D^2\right]^{\frac{1}{2}} \tag{5.13}$$

The effect of the added mass of the fluid interacting with the thruster is modelled using a first order transfer function with a gain of one. The time constant T_w is taken as the difference between the thrust and motor speed time constants. This is calculated from experimental data and for this work is taken to equal 0.2 seconds. The velocity of the flow at the propeller, V_f, is therefore:

$$V_f = \frac{V_{ss}}{T_w s + 1} \tag{5.14}$$

The angle of incidence α can now be calculated using the fixed pitch of the propeller P, the inflow speed V_f, and the propeller tip velocity:

$$\alpha = \tan^{-1}(P) - \tan^{-1}\left(\frac{2V_f}{nD}\right) \tag{5.15}$$

The value of interest for modelling the transient performance of the thruster is the change of α from its steady-state value. Therefore:

$$\Delta\alpha = \tan^{-1}\left(\frac{2V_f}{nD}\right) - \tan^{-1}\left(\frac{2V_{ss}}{nD}\right) \tag{5.16}$$

Equation (5.16) calculates the change of inflow angle to the propeller blade. To translate this value into a change in thrust requires knowledge of the lift coefficient for the foil. This information is not available therefore the change in lift coefficient, ΔC_L, per degree change of angle of incidence, $\Delta\alpha$, is estimated to be 0.1. This is an average value, similar to that of the common NACA 0012 foil.

$$\Delta C_L = 0.1\Delta\alpha \tag{5.17}$$

$$T \propto 1 + \Delta C_L \tag{5.18}$$

Using the motor model (5.12) to calculate the thruster speed n, and equations (5.13) to (5.17), the thrust can be calculated as:

$$T = K_T \rho n^2 D^4 \left(1 - \frac{J}{J_0}\right)^x (1 + \Delta C_L) \tag{5.19}$$

Figure 5.17 presents experimental and simulated thrust. Both the transient model produced here and a quasi-steady model have been simulated. Clearly the quasi-steady model does not accurately capture the thruster dynamics as there is a significant lag between the simulated and experimental thrust. The transient model does appear to capture the thruster dynamics accurately, with the thrust rise time matching that of the experimental data. The experimental measurement in Figure 5.17 is not smooth like the simulations. This additional noise on the experimental data is due to the interaction of surface waves, generated by the thruster jet, and the experimental framework.

An advantage of the thrust rise time being faster than the motor rise time is that the transient dynamics of the thruster have less of an influence on the overall dynamics of the AUV.

5.3.5 Thruster Performance Near the Free Surface

From previous experience operating the Delphin1 AUV, it had been noted that the control system was required to drive the vertical tunnel thrusters at higher speeds, when diving from the free surface, in order to achieve similar accelerations to those when the AUV was submerged. It was therefore suspected that the thruster performance was lower when on, or near the free surface, than at depth. To quantify the magnitude of any reduction in thruster performance as a function of depth, the experimental set-up used in the previous section is used. The depth of the outer tube is measured from the top of the thruster tunnel to the free surface and is varied from 10 to 400 mm. In addition to the load cell, used to measure the thrust in the previous section, a differential pressure transducer is used to measure the pressure across the propeller and across the outer tube, Figure 5.13.

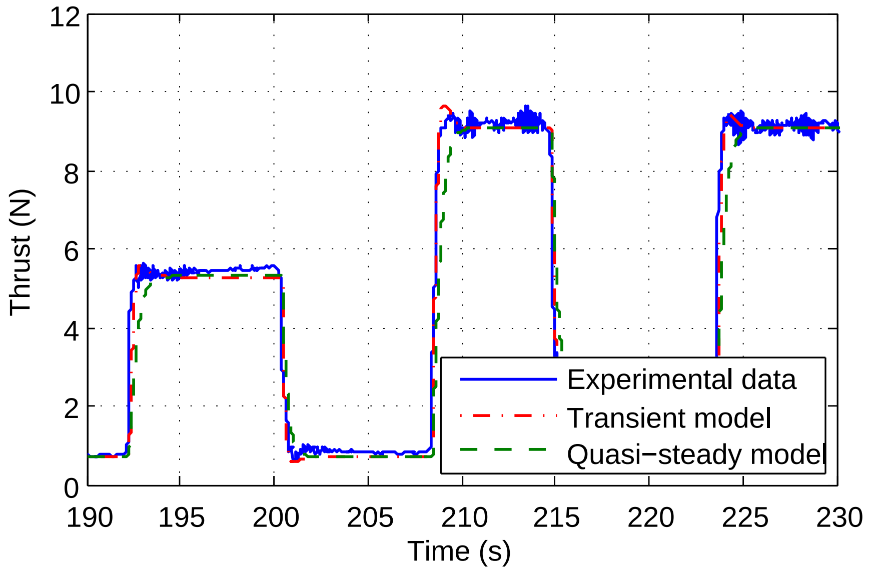


Figure 5.17: Simulation using the transient and quasi-steady thruster model along with measured experimental data. Note that the quasi-steady model fails to adequately approximate the thruster dynamics while the transient model does.

Figure 5.18 presents experimental data of the measured thrust, using the load cell, against depth for four different thruster speeds. The top two plots, for thruster speeds of 550 rpm and 975 rpm, show a significant reduction in thrust below 100 mm depth. For the thruster speed of 1550 rpm the magnitude of this reduction is much less and for the thruster speed of 2200 rpm there is no apparent reduction in thrust near the surface.

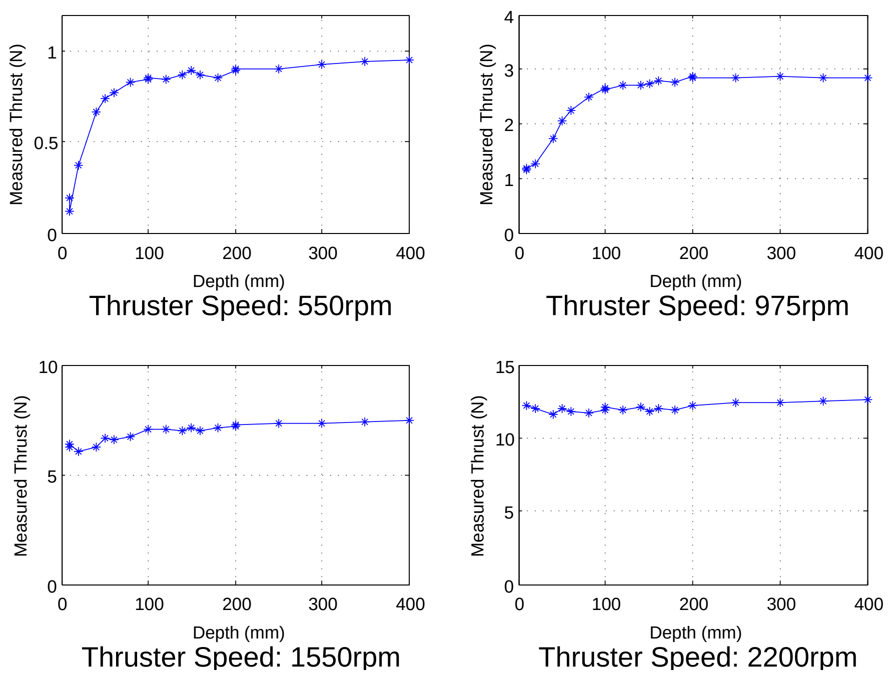


Figure 5.18: Measured thrust against depth for four thruster speeds.

The total thrust acting on the load cell is the sum of the force generated by the thruster propeller and the forces on the outer tube due to the external fluid flow. To evaluate if this reduction in thrust is due to a degradation of propeller performance the differential pressure across the propeller is investigated. First the general thrust equation is examined:

$$T = \dot{m}_{exit} V_{exit} - \dot{m}_{in} V_{in} + (p_{exit} - p_{in}) A_{exit} \tag{5.20}$$

where $\dot{m}_{in}$ and $\dot{m}_{exit}$ are the mass flow rates entering and exiting the thruster respectively, V_{in} and V_{exit} are the fluid velocity entering and exiting the thruster, p_{in} and p_{exit} are the pressures immediately upstream and downstream of the thruster propeller, and A_{exit} is the cross-sectional area at the end of thruster tunnel.

It can be seen that the thrust, T, is a function of the change in mass flow rate, velocity and pressure across the thruster. As the propeller is fitted within a tunnel, the average velocity and mass flow rate just before and after the propeller are equal, assuming the fluid is incompressible. Therefore the thrust equation can be simplified to:

$$T = (p_{exit} - p_{in}) A_{exit} \tag{5.21}$$

As noted in Figure 5.18, the measured thrust does not appear to be a function of depth at depths greater than 200 mm. The forces measured using the load cell at 400 mm depth are assumed to be primarily that of the thrust generated by the thruster propeller, and that the hydrodynamic forces acting on the outer tube are low. With these assumptions, the thrust can be calculated using equation (5.21) and the (measured) differential pressures across the propeller. Figure 5.19 presents the calculated thrust, using the differential pressure measurements, and a linear fit of the measured thrust at 400 mm depth.

The correlation between the calculated and measured thrust in Figure 5.19 is very close, supporting the assumptions that the measured thrust is dominated by the force generated by the propeller above 200 mm depth. The differential pressure across the propeller will now be used to examine whether the reduction in measured thrust is due to a degradation in propeller performance.

In Figure 5.20 the calculated thrust is plotted against depth for four thruster speeds. The differential pressure across the propeller does not appear to be influenced by depth, so it can be concluded that the propeller is generating a consistent magnitude of thrust at all the tested depths. The reduction in thrust near the free surface must therefore be due to forces generated by the local fluid flow around the replica hull.

The differential pressure across the hull is presented, against depth and thruster speed, in Figure 5.21. The data has been interpolated using the MATLAB Biharmonic method [93]. For the majority of the data points the differential pressure is approximately zero. However, when the thruster is operating near the free surface, and at speeds less than 1000 rpm, there is a significant negative

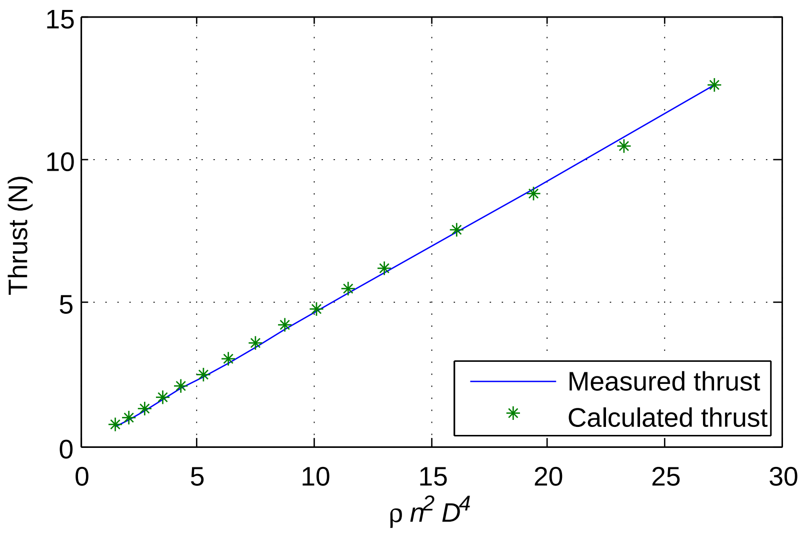


Figure 5.19: Thrust values, calculated using the differential pressure data, against $\rho n^2 D^4$.

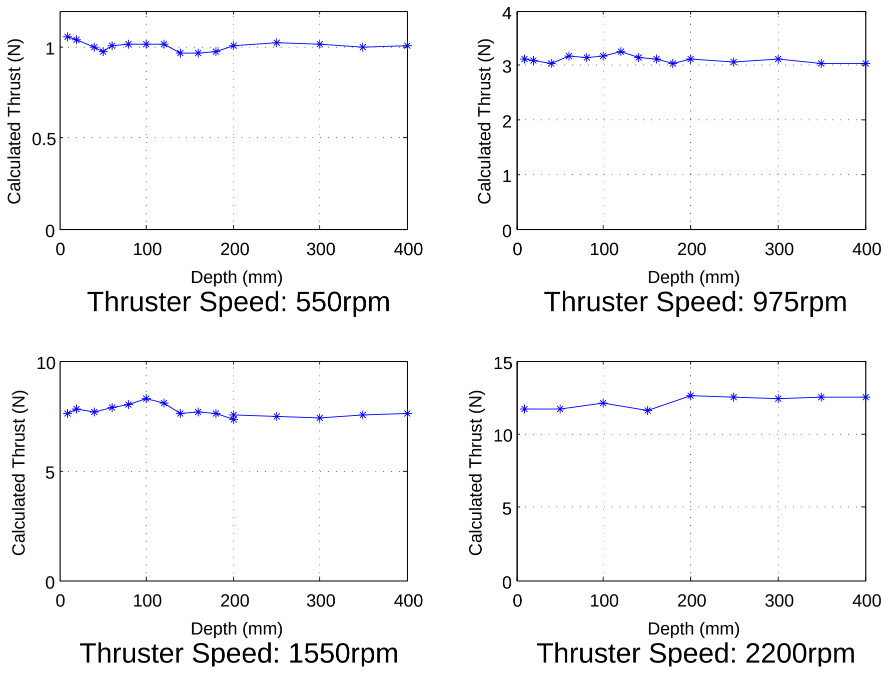


Figure 5.20: Calculated thrust against depth for four thruster speeds.

pressure across the hull. This pressure acts over an area of the hull resulting in a force of the opposite sign to the thrust generated by the thruster propeller, therefore reducing the net force in the Z axis.

The reason for this area of low pressure is thought to be caused by an increase in flow velocity across the top of the outer tube as the AUV approaches the surface. This increase in flow velocity is due to an effective blockage arising from the free surface which forces the flow sideways (perpendicular to tunnel axis) and across the top of the outer tube, Figure 5.22. This hypothesis does fit with the experimental data however it has not yet been proven. A mathematical model of the reduction in

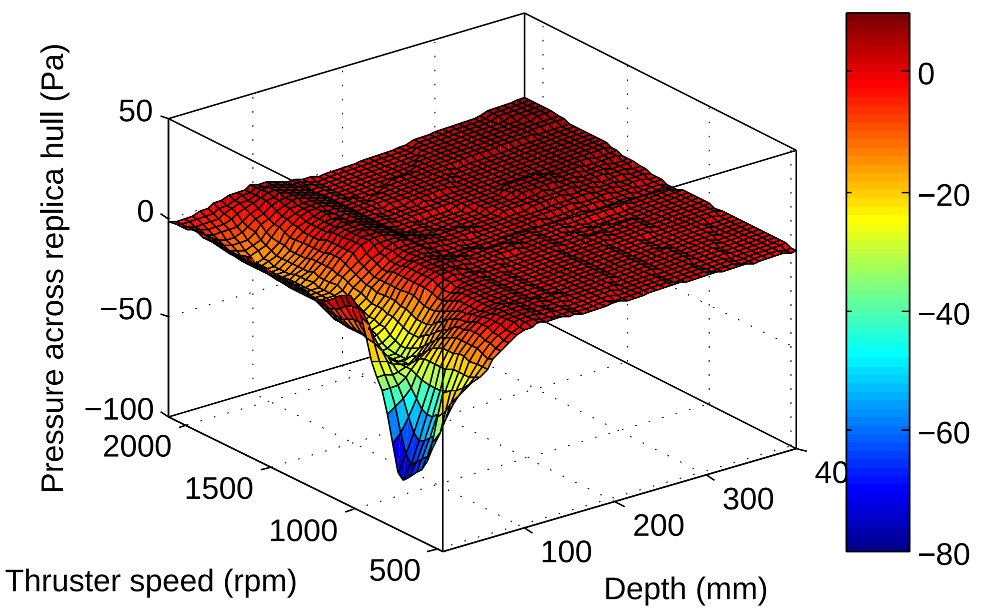
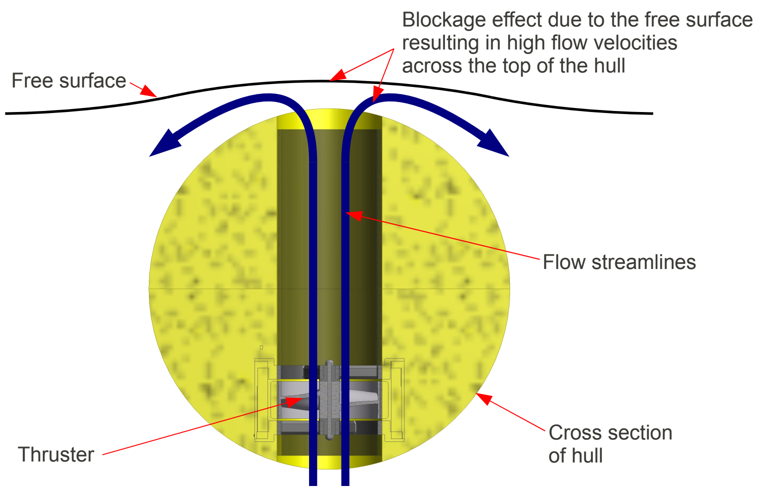


Figure 5.21: Differential pressure across outer tube against depth and thruster speed.

thrust is produced in [94].



Blockage effect due to the free surface resulting in high flow velocities across the top of the hull
Free surface
Flow streamlines
Thruster
Cross section of hull

Figure 5.22: Illustration demonstrating the effective blockage on the thruster jet by the free surface (when operating at depths less than 200mm) and the subsequent high velocity streamlines across the top of the hull causing an area of low pressure.

The reduction in thrust, as a function of depth, only occurs in a very small region of the Delphin2 AUVs operational depth range. This non-linearity could present problems for a depth controller when tasked with diving the vehicle from the surface. To handle this, an additional component should be added to the controller to amplify the controller output when operating on the surface. Increasing the normal controller gains in order to overcome the reduction in thrust on the surface could significantly degrade the performance of the controller when operating at depth. It will be confirmed in Chapter 6 that the integral action of the MPC controller helps to quickly compensate for this nonlinearity.

This non-linearity must also be taken into account when designing vehicles similar to Delphin2. If the thrusters are chosen to be of a suitable size for operating at depth then they may prove to have insufficient thrust to dive the vehicle from the free surface.

5.3.6 Thruster Performance with Positive Surge Velocities

It has been shown that there is a strong interaction between the local flow regime around the AUV hull and the thruster jet, [24], [91] and [94]. When the AUV is travelling forwards with a positive surge velocity, the thruster jets interact with the hull causing areas of low pressure that generate forces of opposite sign to the generated thrust. This effect is not unique to AUVs and also occurs on ships operating with bow thrusters [95].

Figure 5.23 is from [3] and presents normalised thrust against the speed ratio of vehicle surge velocity over thruster jet velocity (u/V_f). As can be seen the reduction in thrust is significant. During normal operation of the Delphin2 AUV the speed ratio is typically between 0 and 0.3, therefore a control system using the though-body tunnel thrusters needs to be able to cope with reductions in effective thrust of up to 50%.

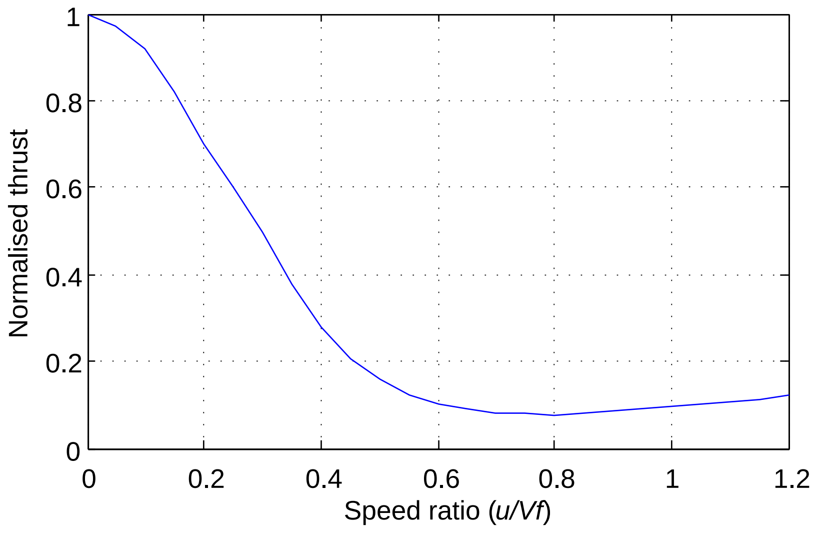


Figure 5.23: Normalised thrust against the ratio of surge velocity over thruster jet velocity. Data from [3].

5.4 Open Water Tests

In this section the linear and angular drag coefficients are calculated for the z and y axes. Open water tests are conducted with the Delphin2 AUV in the AB Wood tank at the University of Southampton. The vehicle is operated semi-autonomously with the objective of measuring the steady-state velocities of the vehicle for known forces and moments generated by the thrusters. Using information from Section 5.3 these forces and moments are calculated and then used, along with the steady-state velocities, to calculate the drag coefficients.

On Delphin2 there is no measurement of sway velocity and achieving stead-state velocities about the pitch (y) axis is not realistically possible due to the restoring moment. The shape of the Delphin2 AUV can be approximated as a body of revolution and so the damping coefficients for the y axes are taken to be identical to those found for the z axis.

The method used in this section involved the AUV diving, using the front and rear thrusters, to a depth of 0.8 m. Once the vehicle has stabilised to within ± 0.1 m of the depth set-point for 30 seconds, the average thrust values for the past 10 seconds are calculated (for the front and rear vertical thrusters). The sum of these average thrust values should then be equivalent to the vehicle's buoyancy but with an opposite sign. The depth controller is then switched off and a predefined thrust value, T_{add}, is added to the front and rear thrust averages and these values passed to the thruster controller. The addition of this extra thrust causes the vehicle to dive and eventually reach a steady-state heave velocity. Once the vehicle reaches 3 m depth the test is stopped and the AUV returns to the surface. The experimental method is illustrated in Figure 5.24. Note that the heave velocity is calculated by post-processing the depth signal to first smooth the signal then differentiate it with respect to time.

Figure 5.25 presents results from one heave test when an additional 2 N of thrust is added to both front and rear thrusters. Note that the AUV is stable at 0.8 m depth then at approximately 68 seconds into the test the depth controller is switched off and the additional thrust is applied. The AUV reaches a steady-state heave velocity at approximately 80 seconds.

Using equation (5.7), the total additional thrust and the terminal velocities are used to calculate the drag coefficient for the AUV along the z axis. Figure 5.26 presents the drag coefficient against Reynolds number. There is some variation in the coefficients values with a higher drag coefficient at lower velocities. This is typical of low Reynolds number operation [85]. For modelling purposes the drag coefficient is taken as the mean average for the range of heave velocities and for this vehicle is taken as 3.33.

A similar set of tests were conducted to measure the terminal yaw angular velocities against known moments generated by the thrusters. For these tests the AUV was set to dive to 1.0 m depth and stabilise for 30 seconds. Then known thrust values were applied to each horizontal thruster with opposite sign. This causes the AUV to turn about the z axis and eventually reach a steady-state yaw

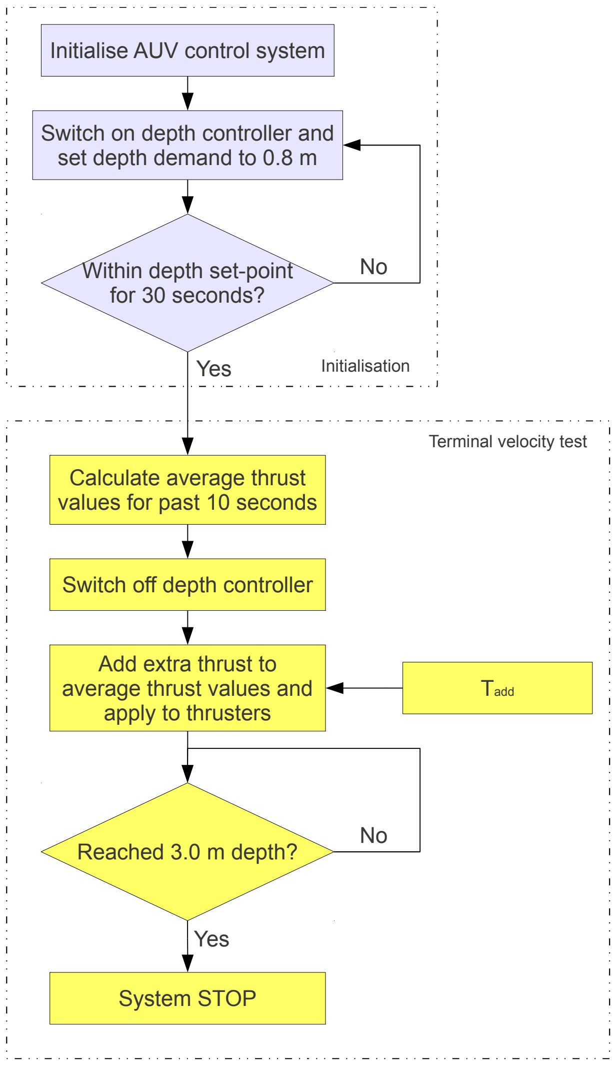

Initialise AUV control system
Switch on depth controller and set depth demand to 0.8 m
Within depth set-point for 30 seconds?
No
Yes
Initialisation
Terminal velocity test
Calculate average thrust values for past 10 seconds
Switch off depth controller
Add extra thrust to average thrust values and apply to thrusters
T add
Reached 3.0 m depth?
No
Yes
System STOP

Figure 5.24: Flow diagram illustrating the experimental method for achieving terminal velocities in the z axis.

angular velocity. The angular velocity is measured using a rate-gyroscope that was specially fitted for these tests. Using these steady-state velocities, and the known moments generated by the thrusters, the drag coefficients are calculated using equation (5.7). The drag coefficient is calculated using the mean average of the results and for this work is taken as 0.027.

Using the newly found drag coefficients a series of $F = ma$ type simulations were conducted to find suitable mass and inertia terms for the AUV accelerating along and about the z axis respectively. The values for m_w and I_q were found to be 167.5 kg and 70.0 kg.m^2.rad^{-2}. The results from this section can be found in Table 5.5.

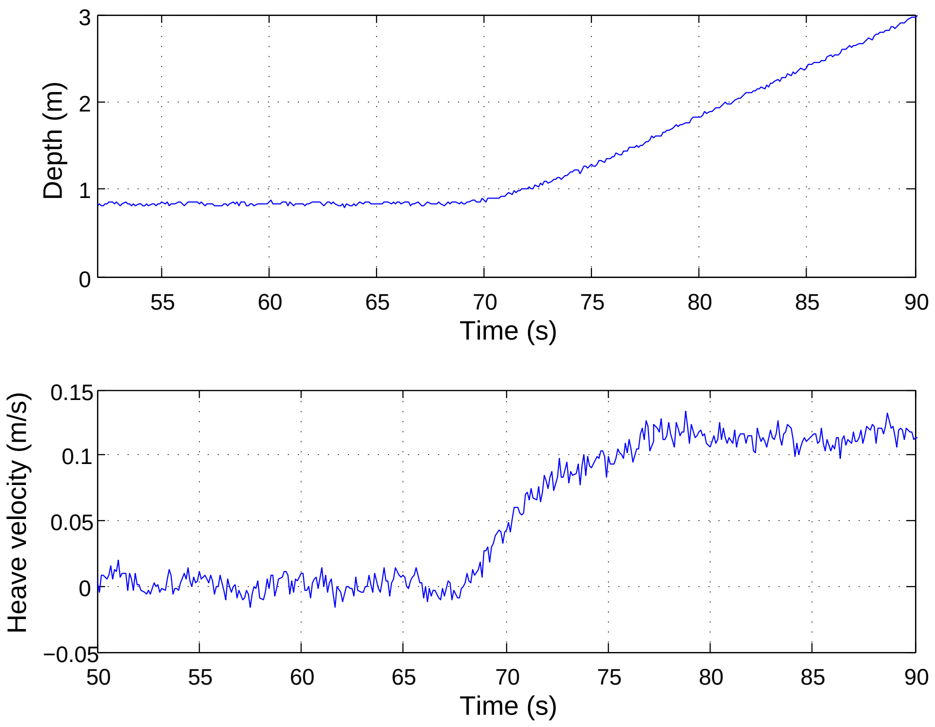


Figure 5.25: Results from a terminal velocity test along the z axis. Note the AUV first stabilises at 0.8 m depth then the depth controller is switched off and and addition 2N is added to both thrusters. The AUV then dives and eventually reaches a steady-state heave velocity of 0.114 m/s at approximately 80 seconds.

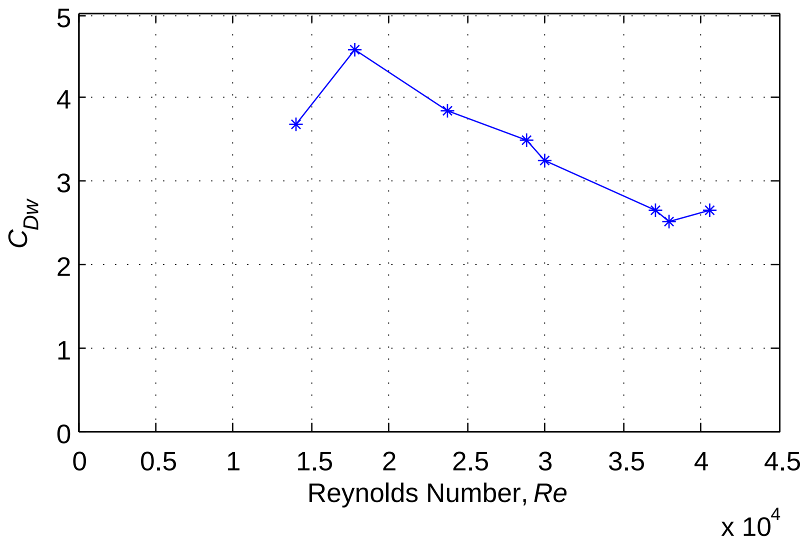


Figure 5.26: Drag coefficients for the z axis against heave velocity. Note slight variation, with a decreasing trend within increasing surge velocity. This is likely due to changes in flow due to varying Reynolds number.

Table 5.5: Damping coefficients and mass terms along and about z and y axes.

C_{Dw}	3.33
C_{Dq}	0.027
I_q	70.0 kgm^{2}rad^{-2}
m_w	167.5 kg

5.5 Rear Propeller

The brushless DC motor that is used to drive the rear propeller is operated by a motor controller designed for remote control hobby boats. The computer transmits set-points to the motor controller via a micro-controller in the tail unit. These set-points are proportional to the power produced by the motor but have no direct physical meaning. There is therefore no speed control, nor speed measurement of the propeller motor.

The direct measurement of the surge velocity is not available whilst the Delphin2 AUV is submerged, however, when operating on the surface the GPS sensor does provide a speed measurement. To calculate the thrust produced by the propeller at the arbitrary motor set-points, the AUV is operated on the surface and the full range of motor set-points are applied. Using the steady-state velocities at each set-point the thrust is taken to be equal and opposite to the hydrodynamic drag force acting against the vehicle. Surge velocity against motor set-point is shown in Figure 5.27. Note that the minimum and maximum set-points for the motor are 10 and 22 respectively, values between 0 and 9 result in the propeller not spinning. These values are also integers therefore fine thrust control is not possible using this motor controller.

The drag forces for the hull and control surfaces can be calculated using the coefficients found in Section 5.2. These forces are accurate when the AUV is submerged but an additional drag component is required whilst operating on or near the free surface due to wave making resistance. Software developed at the University of Southampton is used to provide a numerical estimate of the wave resistance force acting on the Delphin2 AUV at speeds from 0.25 m/s to 1.25 m/s at zero depth (top of AUV hull touching the free surface). The software uses the thin ship methodology developed in [96]. The estimated forces can be found in Table 5.6.

By fitting a polynomial function to the surge velocity against motor set-point it is possible to calculate thrust as a function of set-point such that:

$$T_{prop} = \text{hydrodynamic drag}(u_v) + \text{wave making resistance}(u_v, z) \tag{5.22}$$

Using the current motor controller, the rear propeller can generate thrust from 0.98 to 15.6 N. For control purposes it is desirable to demand a magnitude of thrust in newtons, to convert this thrust

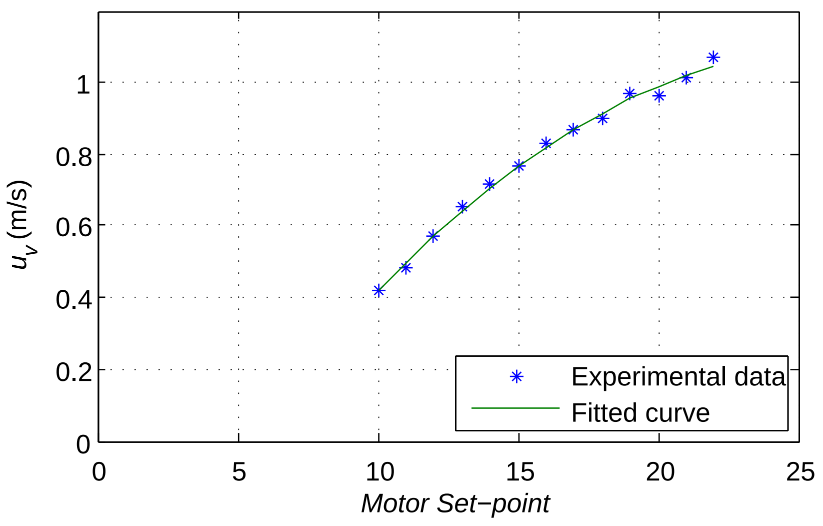


Figure 5.27: Surge velocity of the Delphin2 AUV against motor set-point. Note that this data is the mean average of the steady-state velocities measured using the GPS sensor when the AUV is operated on the surface.

Table 5.6: Wave making resistance at various surge velocities for the Delphin2 AUV at zero depth and pitch.

Surge velocity (u_v)	**Wave making resistance (N)**	**Normal hydrodynamic drag (N)**
0.25 m/s	0.0067	0.3854
0.5 m/s	0.0034	1.5417
0.75 m/s	0.792	3.4688
1.0 m/s	4.97	6.1668
1.25 m/s	15.2	9.6356

value into a motor set-point the fitted polynomial can be used:

$$r_{prop} = round\left[17.22.e^{(0.02008.T_{prop})} - 9.014.e^{(-0.1857.T_{prop})}\right] \tag{5.23}$$

5.6 AUV Model

To help develop low-level control systems for the Delphin2 AUV a non-linear model of the vehicle dynamics is required. This model will be used to simulate the vehicle performance with various controller designs, and will also be used on-board the vehicle to provide dead-reckoning information such as position and surge velocity.

Recalling equations (5.1), (5.2) and (5.3) the surge, heave and pitch rate equations of motion for

Delphin2 can be represented using:

$$
\begin{aligned}
\dot{u} &= \frac{1}{m_u}\left[T_{prop}\cos\theta + T_{vf}\sin\theta + T_{vr}\sin\theta + X_D\right] \\
\dot{w} &= \frac{1}{m_w}\left[W - B + T_{prop}\sin\theta + T_{vf}\cos\theta ...\right. \\
&\qquad\qquad \left. + T_{vr}\cos\theta + Z_{hull} + Z_{cs} + Z_D\right] \\
\dot{q} &= \frac{1}{I_q}\left[x_{Tvf}T_{vf} + x_{Tvr}T_{vr} + M_{hull} + M_{cs} + M_R + M_D\right]
\end{aligned}
$$

The masses, inertia, weight and buoyancy of the vehicle should remain fixed, however due to variations in payload can vary. Typical values for Delphin2 are found in Table 5.7.

Table 5.7: Typical masses, inertia, weight and buoyancy terms for Delphin2.

m_u	85 Kg
m_w	167.5 Kg
I_q	70 kgm^{2}rad^{-2}
W	540 N
B	545 N

The force produced by the rear propeller, T_{prop}, ranges from either 0 or 0.98 to 15.6 N. The dynamics of the rear propeller have been ignored for this work. The forces produced by the vertical tunnel thrusters, T_{vf} and T_{vr}, range from 0.7 N to 10 N. These thrusters can produce more thrust, however for this work will be constrained at 10 N in order to reduce peak power consumption. The transient thruster dynamics are modelled in Section 5.3. For most cases the thruster dynamics can be simplified to a first order system or ignored, however including the full thruster dynamics will provide a more accurate simulation. The moment arms for the front and rear vertical thrusters can be found in Table 5.4.

The forces and moments produced by the hydrodynamics around the vehicle can be calculated using:

$$X_D = \frac{1}{2}\rho V^{2/3} C_{Dh}(\alpha_{hull}^2) u_v^2 \tag{5.24}$$

$$Z_{hull} = \frac{1}{2}\rho L^2 C_{Lh}(\alpha_{hull}) u_v^2 \tag{5.25}$$

$$Z_{cs} = \frac{1}{2}\rho L^2 C_{Lcs}(\delta_{cs}) u_v^2 \tag{5.26}$$

$$Z_D = \frac{1}{2}\rho V^{2/3} C_{Dw} w_v^2 \tag{5.27}$$

$$M_{hull} = \frac{1}{2}\rho L^3 C_{Mh}(\alpha_{hull}) u_v^2 \tag{5.28}$$

$$M_{cs} = \frac{1}{2}\rho L^3 C_{Mcs}(\delta_{cs}) u_v^2 \tag{5.29}$$

$$M_D = \frac{1}{2}\rho L^3 C_{Dq} u_v^2 \tag{5.30}$$

where the coefficients; C_{Dh}, C_{Lh}, C_{Lcs}, C_{Mh}, and C_{Mcs}; can be found in Table 5.3 and the coefficients; C_{Dw} and C_{Dq}; can be found in Table 5.5.

5.7 Conclusions

In this Chapter the dynamics of the Delphin2 AUV have been studied and a mathematical model produced. The vehicle has been modelled by studying each individual component of the system separately then combining the information at the end. There may be some inaccuracies using this method as coupling between the various subsystems will have an influence. For example, the wind tunnel data was performed with the propeller stationary and so its influence upon flow past the hull was not measured. Due to the propellers large diameter and low rotational speed this is not thought to be significant. Another potential error is the effect of the thruster jets on the vehicle drag coefficient. In general however, these effects are thought to be relatively small and, as stated at the beginning of this chapter, the purpose of the model is to capture the key physics and trends rather than be high fidelity.

Both a quasi-steady and transient model of the through-body tunnel thrusters have been developed. For simulations it would be more accurate to include the transient thruster dynamics, however, for controller design it is likely that using the inverse of the quasi-steady model (to handle a non-linear quadratic term) should suffice.

The hull and control surfaces have been modelled using mostly experimental data from wind-tunnel tests. These tests provides the lift, drag and pitching moments generated by both the hull and control surfaces. It was noted that stall occurred during these tests when operating at high angles of attack. Semi-autonomous manoeuvres were conducted to determine the drag coefficients for the hull in the z and y axis.

The accuracy of the AUV model will be determined later in this work when experimental and simulated controller performance are compared.

Chapter 6

Depth and Pitch Control using MPC

6.1 Introduction

To evaluate the suitability of using the MPC algorithm to control the Delphin2 AUV, first the control of a relatively linear system is attempted. The objective of this chapter is to design a depth and pitch controller for the Delphin2 AUV, when operating at zero forward speed, using the two vertical tunnel thrusters. The system is therefore a multi-input multi-output (MIMO) system with two degrees of freedom. Although the controller is designed to control both depth and pitch, the primary objective is depth control whilst maintaining zero pitch.

The depth and pitch model of the vehicle is first introduced and then this model is linearised about zero forward speed. This linear model is then used within the MPC controller. Simulations of the MPC controller, coupled with the non-linear model, are performed within MATLAB and used to examine:

- the effect of the various tuning parameters and to find a stable set that provides good overall performance,
- the controller sensitivity to model inaccuracies,
- the controller sensitivity to low quality feedback.

Table 6.1: Depth and pitch controller overview

Number of inputs	2
Number of outputs	2
Number of tuning parameters	3
Sampling rate	10 Hz

After the controller has been sufficiently tested in simulation, and suitable tuning parameters found, the controller is translated into Python as a node within the Delphin2 software. The controller is then tested experimentally and its performance evaluated and compared to the simulations.

6.2 System Model

The system that is to be controlled can be modelled using two second-order subsystems that are coupled; depth and pitch, see Figure 6.1. The equations of motion are taken to be a subset of those found in Section 5.6, with terms relevant to the x axis emitted, equations (6.1) to (6.4). Several assumptions have been made including; the pitch angles will generally be small ($< 10°$), the surge velocity is negligible, and the buoyancy does not vary with depth.

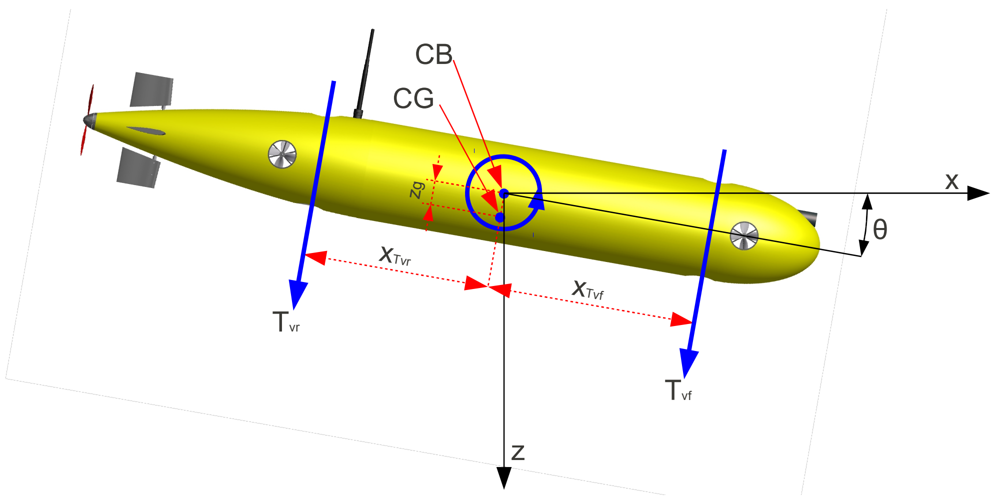


Figure 6.1: Illustration demonstrating the location of the thruster force vectors, moment arms, centre of gravity and buoyancy and the axis system. Note that the AUV in the illustration is at a pitch angle of $-10°$.

Pitch Model

$$\dot{q_v} = -\frac{1}{I_y}[x_{Tvf}T_{vf} + x_{Tvr}T_{vr} - z_g W \sin\theta + \frac{1}{2}\rho V^{2/3} C_{Dq}|q_v|q_v] \tag{6.1}$$

$$q_v = \int_0^t \dot{q_v}\ dt,\ \theta = \int_0^t q_v\ dt \tag{6.2}$$

Depth Model

$$\dot{w_v} = \frac{1}{m_z}[T_{vf}\cos\theta + T_{vr}\cos\theta + (W - B) - \frac{1}{2}\rho V^{2/3} C_{Dw}|w_v|w_v] \tag{6.3}$$

$$w_v = \int_0^t \dot{w_v}\ dt,\ z = \int_0^t w_v\ dt \tag{6.4}$$

Equations (6.1) and (6.3) calculate the angular acceleration about the y (pitch) axis and linear acceleration on the z (depth) axis respectively. Both of these equations include non-linear terms that need to be linearised for use with the MPC algorithm, this will be discussed in the next section. For simplicity the damping terms will be described as quadratic terms despite the inclusion of the absolute value (used to maintain the correct sign).

Inverse Thruster Model

The output from the controller will be force demands for each thruster (in Newtons), to translate these force values into thruster speed set-points, that can be sent to the thruster controller, the inverse of the thrust equation is used, equation (6.5). This model is derived from that developed in Section 5.3. Note that the transient dynamics of the thrusters are not modelled, nor the effect of the advance ratio. Early work included first order approximations of the transient thruster dynamics however it had no significant effect on controller performance but did increase computational complexity as the dimensions of the A matrix increased.

$$n = 60 \times \left[\frac{T_{demand}}{\rho K_T D^4}\right]^{0.5} \tag{6.5}$$

6.2.1 Linearised State-Space Model

Both the pitch and depth models, equations (6.1) and (6.3), have non-linear trigonometric and quadratic terms that need to be linearised in order to produce a model that can be utilised by the MPC algorithm. In this section the linearisation process will be described and a linear state-space model will be produced that provides an accurate approximation of the AUV dynamics.

The trigonometric terms within the model include both the sine and cosine functions. The cosine function is used to calculate the magnitude of thrust in the z axis. Here the cosine function is approximated to equal 1, effectively removing the influence of pitch on thrust in the z axis. As the controller is being designed to maintain zero pitch, it is thought that it is unlikely that the AUVs pitch will exceed $\pm 10°$. Therefore the maximum error produced by this approximation, equation (6.6), is within $\pm 1.5\%$.

The sine term is used to model the restoring moment within the pitch subsystem. Using small angle approximation, the sine function is omitted from the equation leaving the angle (in radians). The maximum error produced using this approximation, equation (6.7), within pitch angles of $\pm 10°$ is $\pm 0.5\%$.

$$\cos(\theta) = 1 \tag{6.6}$$

$$\sin(\theta) = \theta \tag{6.7}$$

Linearising the quadratic damping terms is more challenging. When the AUV has successfully arrived at the defined depth and pitch set-points, and is stable, the heave and pitch velocities are zero, thus the damping forces and moments are equal to zero. Linearising about zero heave and pitch velocities results in the damping coefficients also equalling zero, therefore eliminating damping from the model. Removing damping from the model would force the MPC algorithm to reduce system velocities using the system inputs (thrusters) rather than the natural dynamics, causing the controller to perform slowly and inefficiently. It is also intuitive that if the damping terms are estimated to be too high then the controller will overshoot the set-point.

The linear damping terms need to be a compromise between low and high speed operation. The linearised and non-linear damping terms are compared using the square of the normalised error:

$$(\text{Normalised error})^2 = \left[\frac{\text{non-linear term} - \text{linear term}}{\text{non-linear term}}\right]^2 \tag{6.8}$$

The sum of the squared normalised errors, from zero to the maximum estimated velocities, is then minimised to produce the linear damping coefficients for this work. The linear and non-linear damping terms are compared in Figure 6.2. The controller's sensitivity to varying the magnitude of these damping coefficients is investigated later in this Chapter. A scheduling method for improving the accuracy of the damping coefficients is developed in the next chapter, see Section 7.4.

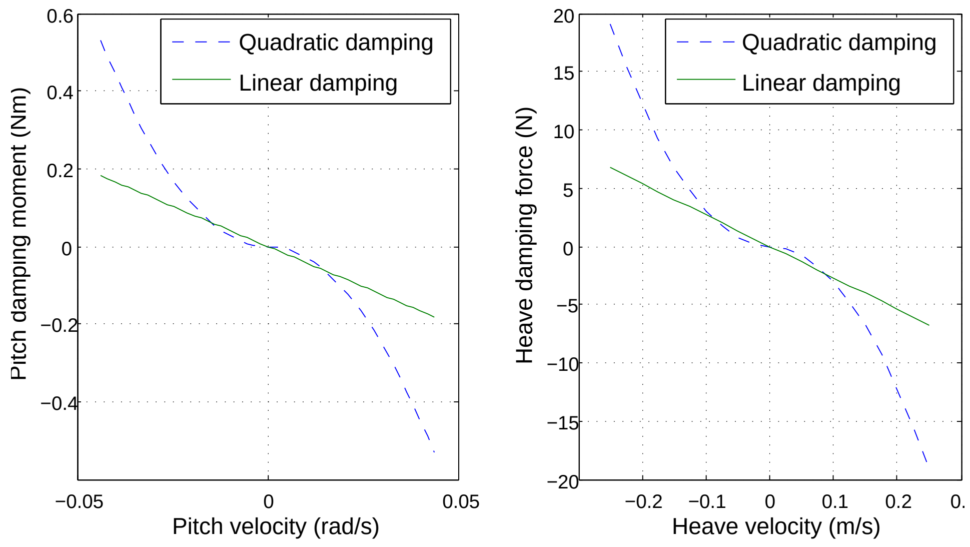


Figure 6.2: Comparison between the linear and non-linear damping forces and moments for pitch and heave.

Combining equations (6.1) to (6.5) the AUV dynamics can be approximated with a time invariant state space model (assuming zero initial conditions):

$$\dot{x}(t) = \overbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{\rho V^{2/3} k_z}{2m_z} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{-z_g W}{I_y} & \frac{\rho V^{2/3} k_q}{2I_y} \end{bmatrix}}^{A_m} \begin{bmatrix} z \\ w_v \\ \theta \\ q_v \end{bmatrix}(t) + \overbrace{\begin{bmatrix} 0 & 0 \\ \frac{1}{m_z} & \frac{1}{m_z} \\ 0 & 0 \\ \frac{-x_{Tvf}}{I_y} & \frac{-x_{Tvr}}{I_y} \end{bmatrix}}^{B_m} \begin{bmatrix} T_{vf} \\ T_{vr} \end{bmatrix}(t) \tag{6.9}$$

$$y(t) = \overbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}^{C_m} \begin{bmatrix} z \\ w_v \\ \theta \\ q_v \end{bmatrix}(t) \tag{6.10}$$

where k_z and k_q are the linear damping coefficients. The values of the parameters used in this state-space model are given in Table 6.2.

The state space model is then discretized using the zero-order hold method [71] using the sampling rate that the controller is chosen to operate at. Next the discretized state space model is then transformed into the augmented model using equation (3.9) and the matrices used for calculating and optimizing the future prediction horizon, F and ϕ, are formed using equation (3.18).

Table 6.2: Linear state-space model parameters

Parameter	**Value**
ρ	1000 kg/m^3
V	0.08 m^3
m_z	167.5 kg
I_y	70 kgm^2
W	540 N
k_z	8.62×10^{-2}
k_q	1.31×10^{-2}
x_{Tvf}	0.55 m
x_{Tvr}	-0.49 m

6.3 Constraints

Constraints are applied to the system inputs, on both the rate of change and absolute limits. The rate of change of thrust demand should be chosen to be lower than the maximum rate that the thruster can change its speed (this is an approximation, as thrust is proportional to thruster speed squared). If the controller demands larger changes of thrust than the thruster is capable of, then the thruster dynamics will effectively act as a low pass filter. This will most likely prove detrimental to controller performance. In this chapter, the rate of change constraints for each thruster are set at ± 2 N per sample. This value is chosen by analysing data from Figure 5.17 such that ± 2 N corresponds to the maximum derivative value per sample time.

The maximum absolute thrust value is set at +10 N, corresponding to a thruster speed of approximately 1800 rpm. The thrusters are capable of driving at higher speeds, and clearly producing more thrust. It is, however, more desirable to reduce the peak thrust demands so as to reduce the peak electrical power consumption.

The minimum absolute thrust value is set at +0.7 N, corresponding to a thruster speed of approximately 500 rpm. This is the minimum thruster speed below which the thruster dynamics become more non-linear due to the performance of the thruster controller (hardware) and the motor dead-band around zero thruster speed. Avoiding these non-linearities is the reason for setting the limit above zero speed.

Note that both of the thruster constraints are positive. By setting these constraints it is assumed that the buoyancy force of the vehicle is greater than twice the minimum thrust constraint, otherwise the vehicle will be unstable in depth.

6.3.1 Implementation

To demonstrate the implementation of the MPC algorithm an example is provided for finding the optimal ΔU set at the start of a simulation. First the global optimal solution is found by solving equation (6.11). Using zero initial conditions, with depth and pitch demands of 1 metre and 0 degrees respectively, and the controller parameters stated in Table 6.3, the first global optimal solution is:

$$\Delta U = (\phi^T\phi + \bar{R})^{-1}(\phi^T R_s - \phi^T F x(k_i)) = \begin{bmatrix} 2.9798 \\ 3.1606 \\ 2.7886 \\ 2.9616 \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \end{matrix}\right\} \Delta u(k_i) \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} \Delta u(k_i+1) \end{matrix} \tag{6.11}$$

Table 6.3: Controller parameters

Parameter	**Variation**
T_{max}	10 N
T_{min}	0.7 N
ΔT_{max}	2 N
ΔT_{min}	-2 N
N_p	50
N_c	2
r_w	0.2

For this controller the length of the control horizon N_c is two and the number of system inputs is also two, therefore the ΔU vector contains the values of two sets of controller inputs, resulting in the four values shown in equation (6.11). The first two values are the first input set, and the next two values are the second set.

Note that all four of the ΔU values have violated the ΔT_{max} constraints and therefore the Hildreth programming procedure, Section 3.2.5, is used to find the active constraints. After five iterations of the Hildreth programming procedure, the solution for λ^* converges to:

$$\lambda^* = \begin{bmatrix} 0.3506 \\ 0 \\ 0 \\ 0 \\ 0.3052 \\ 0 \\ 0 \\ 0 \\ 0.4007 \\ 0 \\ 0 \\ 0 \\ 0.3532 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.12}$$

Discarding the constraints corresponding to the λ^* values of zero, the active constraints are now set as equality constraints and the optimal solution found within the constraints:

$$\Delta U = (\phi^T \phi + \bar{R})^{-1}(\phi^T R_s - \phi^T F x(k_i)) - (\phi^T \phi + \bar{R})^{-1} M_{act}^T \lambda_{act}^* = \begin{bmatrix} 2.0 \\ 2.0 \\ 2.0 \\ 2.0 \end{bmatrix} \tag{6.13}$$

The first m values from ΔU are added to the previous U vector to provide the controller output. The U values are converted into appropriate thruster speed set-points using equation (6.5) and sent to the thruster controller. At the end of the loop the states and outputs are updated with the latest feedback and the loop starts again.

6.4 Tuning

There are three parameters that can be used to tune the MPC controller; N_p, N_c and r_w. Before tuning the controller, first the tuning parameters will be examined and discussed. The prediction horizon, of length N_p, is used to estimate the future system trajectory for a given control horizon, of length N_c. The idea of the prediction horizon is to look forward in time and use this information to find the optimal change of future system inputs.

An analogy that can be used is when driving a car. The prediction horizon is analogous to the drivers ability to look at the road ahead and judge the best trajectory to stay safely on the road. If the prediction horizon is too short, equivalent to only looking at the road immediately in front of the car bonnet, then the driver will not see the approaching corner and will not take action soon enough, resulting in a crash. Similarly, if the prediction horizon is too long, equivalent to looking several miles ahead, then the driver will drive smoothly but may not react to disturbances such as potholes that are immediately in front of the car. Clearly too short of a prediction horizon will result in an unstable or marginally unstable system, but to maintain good system response time the prediction horizon cannot be too long.

The control horizon is the estimate of the future change of system inputs, such as how the driver will adjust the steering wheel to successfully steer the car around the corner. If the control horizon is too short then the driver will try to get around the corner using aggressive changes in the steering wheel angle. A longer control horizon will result in the driver changing the steering wheel angle more smoothly. If the control horizon is the same length as the prediction horizon then there is a risk that the inaccuracies of the prediction will cause the optimal control horizon to be inaccurate. Obviously the control horizon cannot be longer than the prediction horizon as the driver cannot calculate the optimal future change in steering wheel angles for corners that he/she has not yet seen.

The last tuning term is the weighting scalar r_w. This scalar is used to penalise large changes in system input. Therefore the optimal ΔU vector with an r_w value of zero will be the fastest, or most aggressive, system inputs that will reduce the control error. An r_w value greater than zero will reduce the magnitude of the values in the ΔU vector, therefore slowing or smoothing the future control trajectory. The r_w scalar is similar to the inverse of the gain used within a classical controller; a low value will result in a fast reacting system while a higher value will result in a slower and smoother reacting system.

In order to understand the effect of each tuning parameter, a large set of simulations were conducted for different N_p, N_c and r_w values. The results are evaluated by the time taken to converge to the desired set-points, depth overshoot and maximum pitch disturbance. The set-points for depth and pitch are 1 metre and $0°$ respectively and each simulation is 50 seconds in duration. The values of the varied parameters are shown in Table 6.4.

Table 6.4: Controller parameters varied in simulation to find optimum controller parameters

Parameter	**Variation**
N_p	$50 \rightarrow 100$
N_c	$2 \rightarrow 20$
r_w	$0 \rightarrow 12$

Performance of each parameter set will be evaluated using; depth overshoot, pitch disturbance and the time taken by the vehicle to stabilise within a predefined tolerance of the depth and pitch set-points. The calculation of these parameters is demonstrated in Figure 6.3. If the system is stable and converges to the test set-points then the controller is deemed stable, however it should be noted that the controller has not been theoretically proven stable using Lyapunov theory.

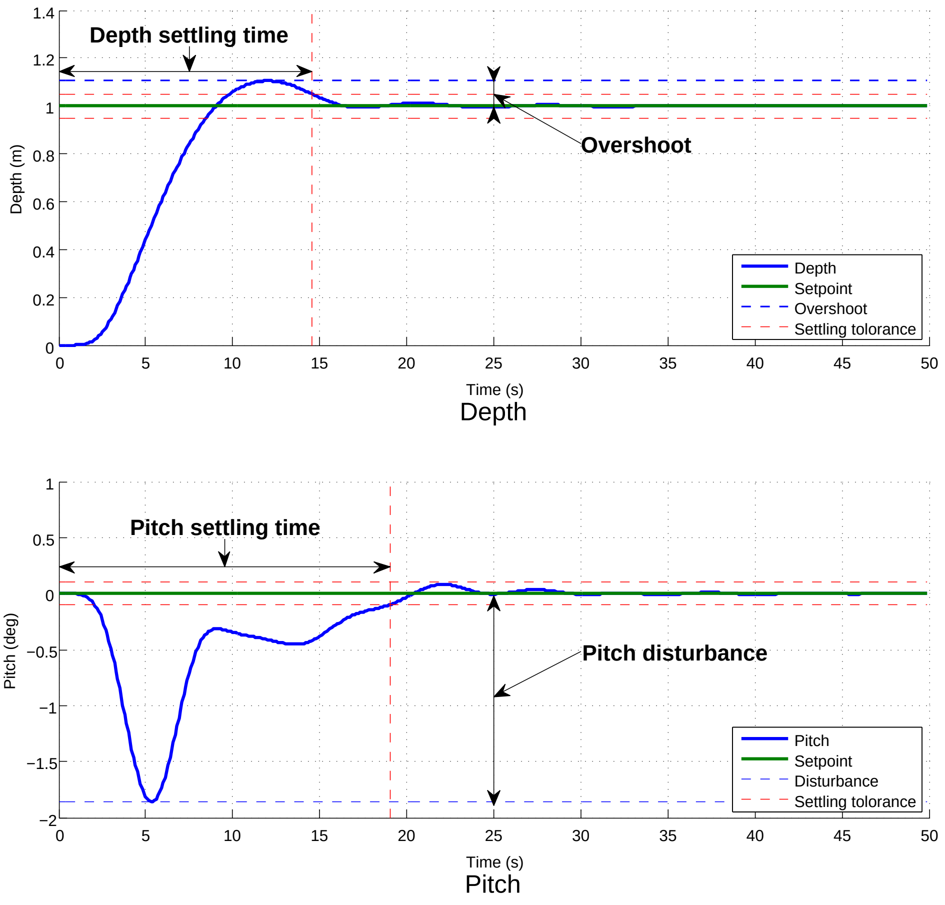


Figure 6.3: Annotated simulation results demonstrating how overshoot, pitch disturbance and the depth and pitch settling times are calculated.

6.4.1 Estimate of the Minimum Length of the Prediction Horizon

An approximate value for the prediction horizon N_p can be found by calculating the time taken to decelerate from the maximum estimated heave velocity, with both thrusters at their maximum absolute constraints, to zero heave velocity, with both thrusters at their minimum absolute constraints. This is an estimate of the minimum length of time, for the prediction horizon, so that the controller can adjust u and reach the depth set-point with minimal overshoot. This calculation does not account for the constraints on ΔU or the effect of pitch.

The maximum heave velocity is:

$$w_v \max = \left[\frac{2 \times T_{max} - B}{\frac{1}{2}\rho V^{2/3} C_{D_w}} \right]^{0.5} \tag{6.14}$$

and minimum heave velocity is:

$$w_v \min = \left[\frac{2 \times T_{min} - B}{\frac{1}{2}\rho V^{2/3} C_{D_w}} \right]^{0.5} . \tag{6.15}$$

The linear model, as used within the controller, is then used to calculate the time taken to reach zero speed from the maximum terminal velocity:

$$t = \ln \left[1 - \frac{w_v \max}{w_v \max - w_v \min} \right] \frac{m}{-k_z} = 4.658s \tag{6.16}$$

Dividing t by the sampling rate, so as to convert the length of time into a number of samples, and rounding up to the next integer, the minimum number of samples for the prediction horizon is calculated to be 47. It should be noted that this value is not a precise calculation but merely a starting point for tuning.

6.4.2 Overshoot

Arguably the most important performance attribute of a depth controller, assuming the controlled system is stable, is its ability to track depth set-points with minimal, if any, overshoot. Significant overshoot could result in the vehicle colliding with the sea bottom, potentially damaging both the vehicle and the environment. A small amount of overshoot, the magnitude of which can be determined by the controller designer, can be acceptable in order to improve controller performance in other aspects. For this work, an overshoot of 0.1 m is deemed acceptable.

Figure 6.4 presents contour plots of overshoot for six different N_p values, as functions of N_c and r_w. The lighter patches represent low overshoot, with darker patches representing large unacceptable overshoot. For an N_p value of 50, only one N_c-r_w combination was found that resulted in an overshoot of less than 0.1 m. As the N_p values increase, both the area and the intensity of the lighter patches increase.

From Figure 6.4 it can be concluded that, for this system, there is a trend that the higher the N_p value the lower the overshoot magnitude. This agrees with theory, as the longer prediction horizon enables the controller to take action sooner so as to slow and converge to the set-point smoothly. The sensitivity to the N_c and r_w values also appears to decrease with increasing length of the prediction horizon.

6.4.3 Depth Settling Time

Figure 6.5 presents the time taken for the depth to converge to within ±0.05 m for six N_p values, as functions of N_c and r_w. The lighter patches represent shorter settling times and the darker patches

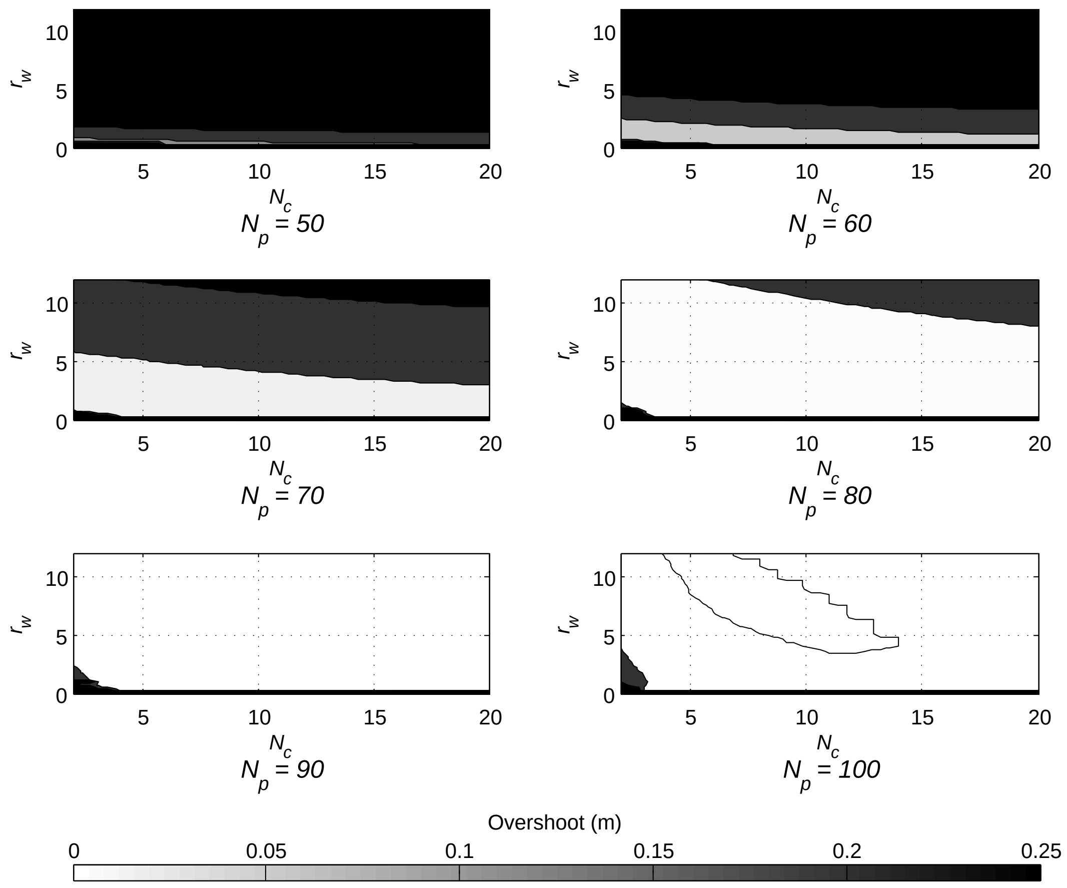


Figure 6.4: Overshoot (m) for different r_w, prediction and control horizons (N_p and N_c respectively). Note the white areas represent low overshoot values.

represent longer settling times. Patches that appear black indicate that the system has not converged and is likely unstable or marginally unstable. Similar to Figure 6.4, the area of the lighter patches increase as N_p increases. However, it differs from Figure 6.4 as the intensity of the lighter patch decreases slightly with increasing N_p. From Figure 6.5 it can be seen that for each N_p value, there is a band of N_c and r_w values within which the system is stable. The area of this band increases with increasing N_p. It should also be noted that the controller stability appears to be more sensitive to r_w at low N_p values and, to a lesser extent, more sensitive to N_c at high N_p values. In summary, the settling time increases slightly with increasing N_p values (given suitable r_w and N_c values) but the sensitivity to the controller parameters, especially r_w, increases greatly with decreasing N_p.

6.4.4 Pitch Settling Time

Figure 6.6 presents the time taken for the pitch to converge to within $\pm 0.1^\circ$ for six N_p values, as functions of N_c and r_w. The pitch settling times follow the same pattern as with the settling times for depth. For low N_p values, there are narrow bands of controller parameters that provide stable performance. As N_p increases the width of this band increases, improving the probability that the

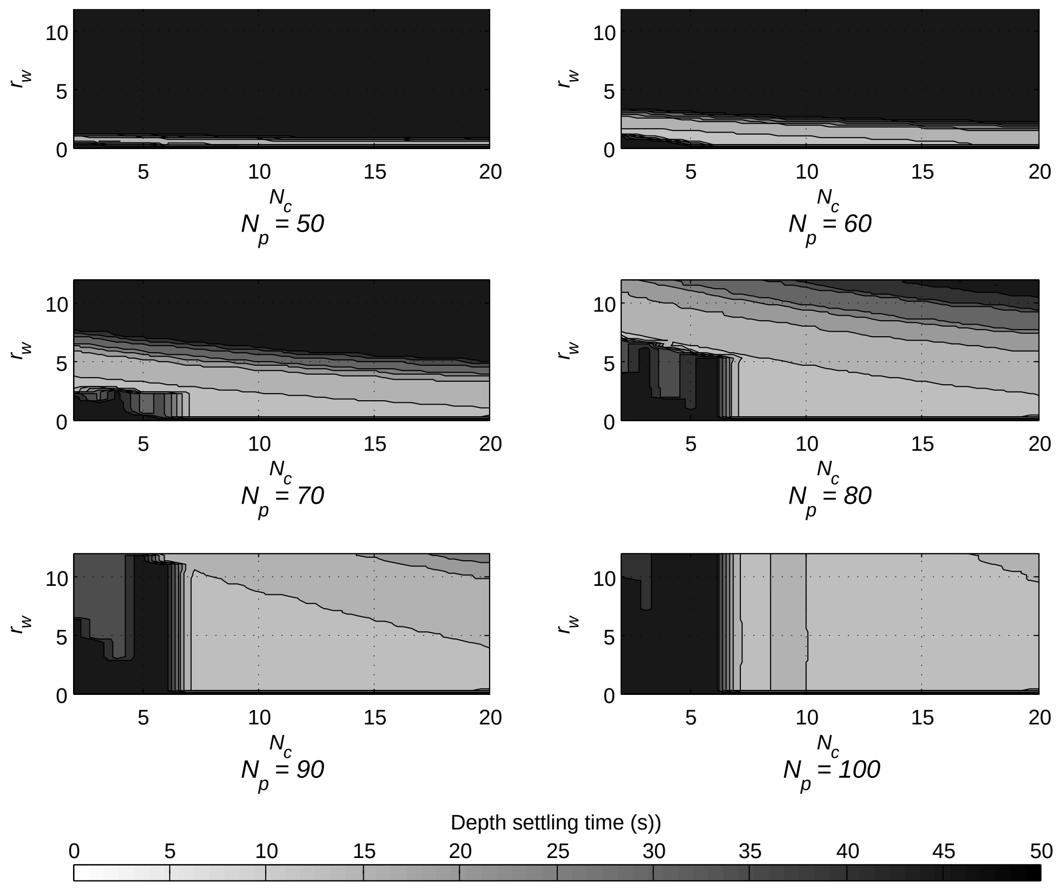


Figure 6.5: Depth settling time (s) for different r_w, prediction and control horizons (N_p and N_c respectively). Note the lighter shade of grey represents faster settling times while black indicates that the vehicle has failed to converge within the tolerances after 50 seconds and is most likely unstable.

real system will also be stable. In comparison to the results for depth settling time, the regions of stable controller values are smaller for pitch settling time indicating that the pitch subsystem is less stable.

6.4.5 Pitch Disturbance

Finally, Figure 6.7 presents the maximum absolute pitch error for six N_p values, as a function of N_c and r_w. It can be argued that this is the least important of the four parameters on which the controller is evaluated. However, for some inspection type manoeuvres, minimizing pitch disturbance may prove important. The results in Figure 6.7 follow a similar trend to the overshoot results in Figure 6.4, with large disturbances at low N_p values and significantly lower disturbances at higher N_p values.

6.4.6 Summary

Figures 6.4 to 6.7 have shown the varying sensitivity to the different tuning parameters. Figures 6.8 to 6.10 have been created to further investigate the different trends in controller performance against

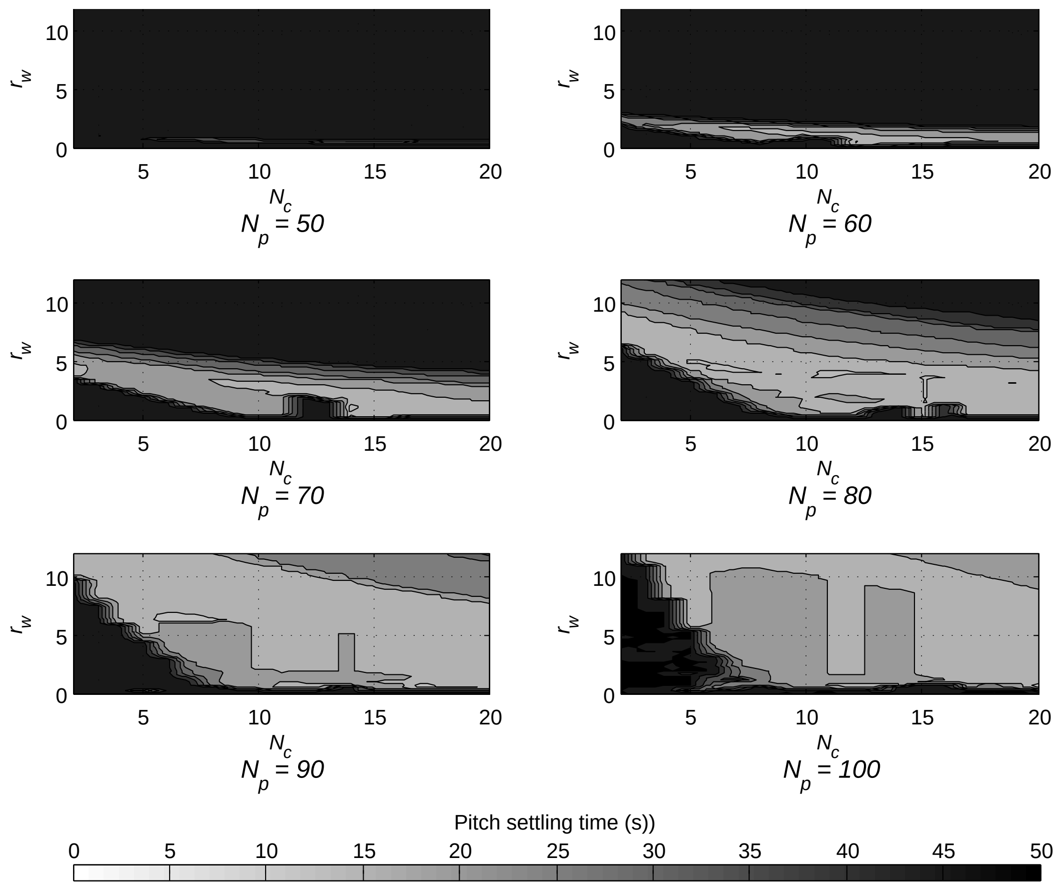


Figure 6.6: Pitch settling time (s) for different r_w, prediction and control horizons (N_p and N_c respectively). Note the lighter shade of grey represents faster settling times while black indicates that the vehicle has failed to converge within the tolerances after 50 seconds and is most likely unstable.

tuning parameters.

In Figure 6.8 the results, using the optimum N_c and r_w combinations, for each prediction horizon N_p are presented. These optimum controller parameters are those that provided the lowest overshoot, pitch disturbance, depth and pitch settling times at each length of prediction horizon.

In the top sub-plot of Figure 6.8, the depth overshoot and the pitch disturbance are plotted against the length of the prediction horizon, N_p. Both curves follow similar trends with decreasing overshoot and pitch disturbance with increasing N_p values, and an overshoot of effectively zero above an N_p value of 80.

The bottom sub-plot of Figure 6.8 presents the depth and pitch settling times against the length of the prediction horizon, N_p. The settling time for depth does not significantly vary as a function of N_p with the shortest settling time at an N_p value of 60. In contrast, the settling time for pitch decreases significantly with increasing values of N_p, such that the settling time at an N_p value of 100 is over 44% shorter that that with an N_p value of 50.

The corresponding values of N_c, for the optimum results presented in Figure 6.8, are presented in Figure 6.9. From this information it is difficult to draw significant conclusions. There is a general

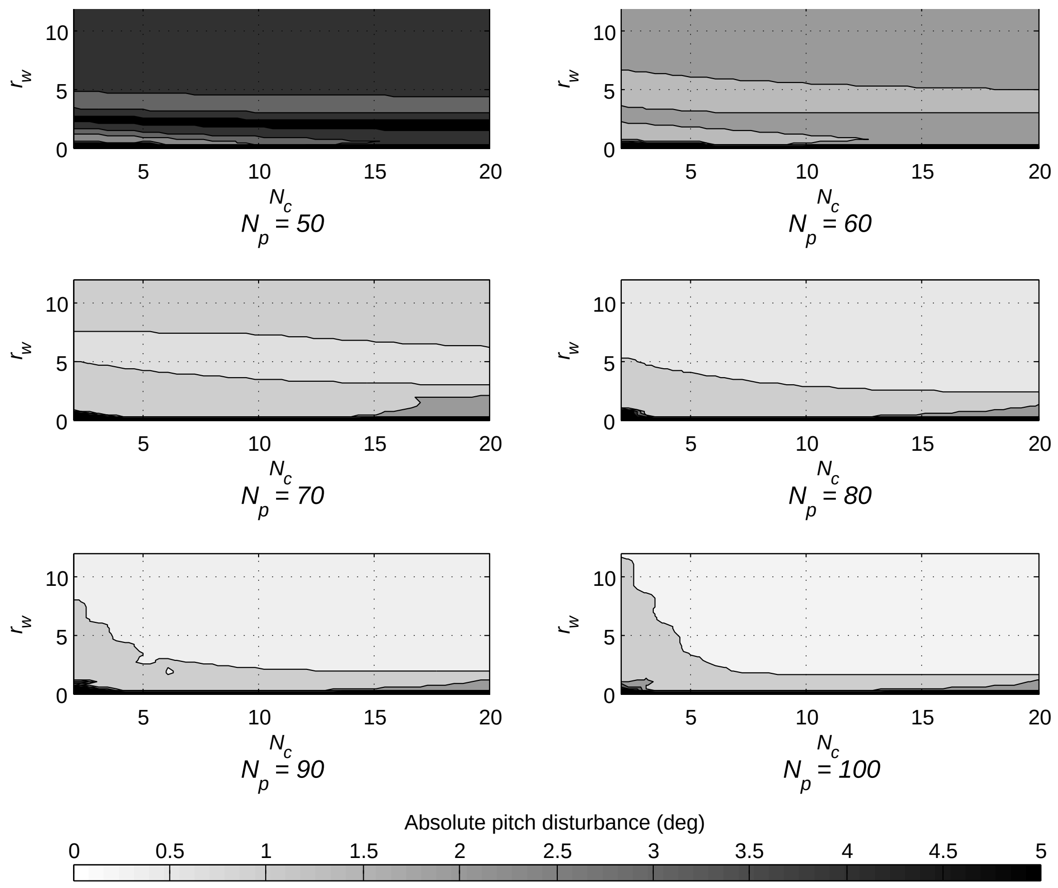


Figure 6.7: Maximum absolute pitch error (deg) for different r_w, prediction and control horizons (N_p and N_c respectively). Note that lighter patches indicate low pitch disturbance angles.

trend of increasing N_c value with increasing N_p values, however, this correlation is statistically weak.

In Figure 6.10 the r_w values, for the optimum results presented in Figure 6.8, are presented. The only significant trend in this data is for the optimum overshoot, with an exponentially increasing value of r_w with increasing N_p value. For both settling time values, the optimum r_w values are low. This corresponds with theory as smaller r_w values increase the priority of minimizing the errors, between the set-points and current vehicle state, and in turn decreases the priority of minimising the values in the ΔU vector. Referring back to Figure 6.6, the low r_w values tend to be near areas of instability. Therefore to optimise primarily for speed of convergence could increase the risk that the real system will become unstable.

In Figures 6.9 and 6.10 the pitch settling time deviates from the general trends with a prediction horizon of 80. These graphs plot the optimum results from the simulations. A combination of the non-linear behaviour of the system and the discrete sets of parameters used by the simulations resulted in the optimum converging to a different local minima than the other sets at different prediction horizons.

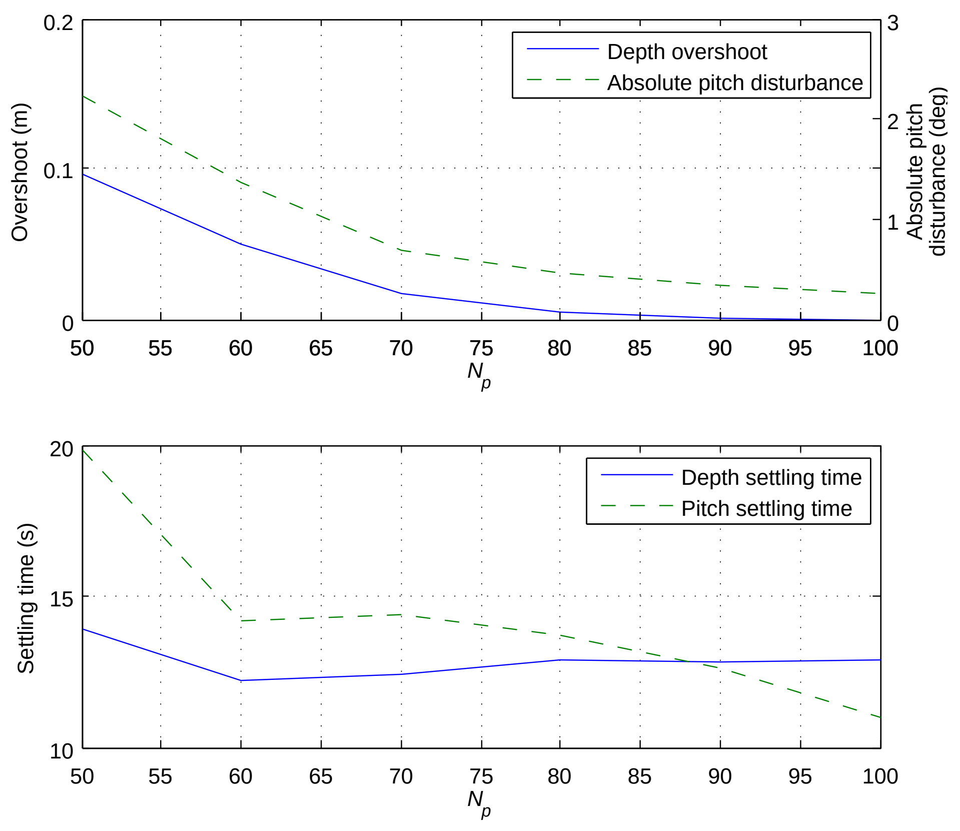


Figure 6.8: Top sub-plot: overshoot and pitch disturbance against the length of the prediction horizon. Bottom sub-plot: depth and pitch settling times against the length of the prediction horizon.

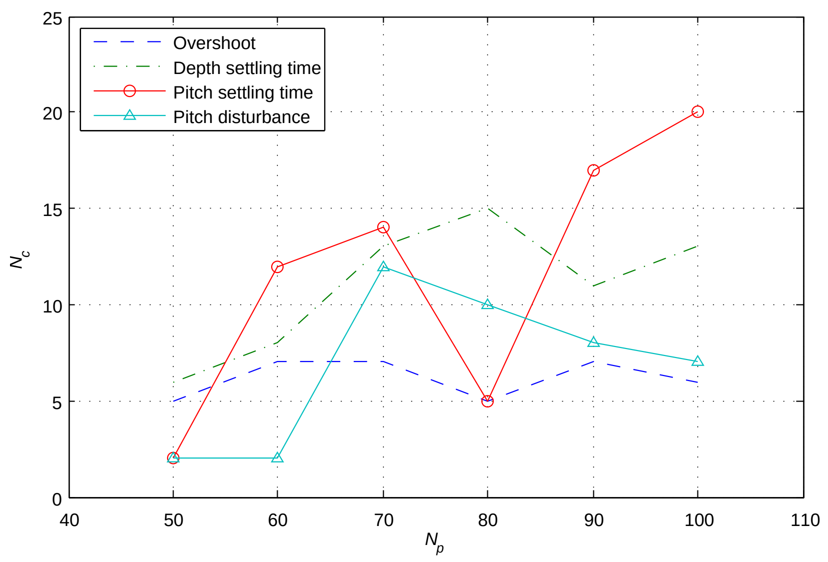


Figure 6.9: The optimum length of control horizons that produced the minimum overshoot, pitch disturbance, depth and pitch settling times, all against the length of the prediction horizon.

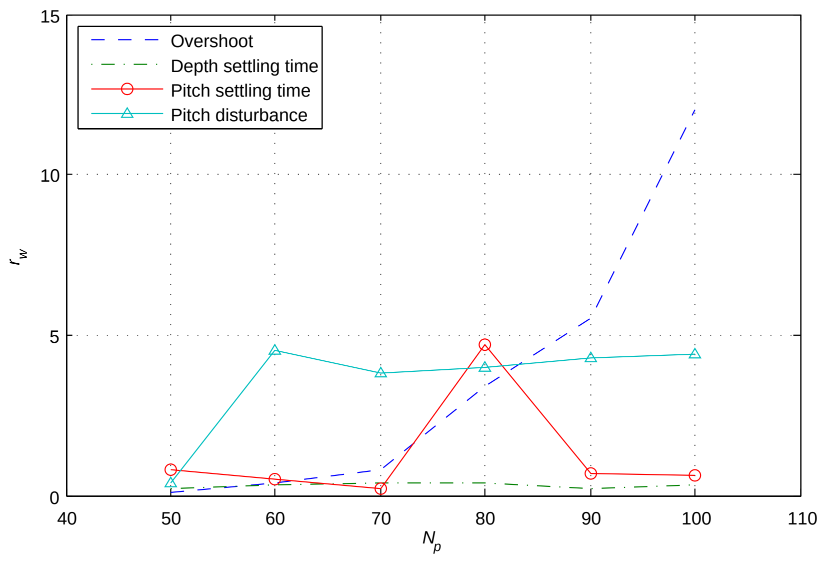


Figure 6.10: The optimum weighting scalar, r_w, that produced the minimum overshoot, pitch disturbance, depth and pitch settling times, all against the length of the prediction horizon.

6.4.7 Conclusions

Three parameters used for tuning the MPC algorithm, N_p, N_c and r_w, have been varied and their effects on performance evaluated. An initial estimate for the minimum value of N_p was calculated to be 47 samples. The minimum value of N_p that was investigated in this section was 50. The overall performance using an N_p length of 50 was deemed poor, with large overshoots and a very small region of r_w and N_c values that provided stable performance.

As N_p increases, overall system performance generally improves and the region of r_w and N_c values that provide stable performance increases, thus making it more probable that the controller parameters will work on the real system. Higher N_p values gave faster settling times for pitch, and the overshoot and pitch disturbances are also lower. However, the time taken for depth to converge is slightly slower with higher N_p values.

The control horizon appears to have the least significant effect on system performance, with stability only compromised with low values of N_c at higher N_p values. The variation in system performance with different r_w values it is highly dependant on the choice of the other two controller parameters. At low values of N_p the range of suitable r_w values is narrow, decreasing the probability that that a suitable r_w will work when experimentally tested on the real system.

To aid in the evaluation of other effects, such as model inaccuracies and low quality feedback, three sets of controller parameters are chosen; a sensitive, a stable and a balanced set. The sensitive set of controller parameters are chosen for good system performance but with parameters chosen near to unstable parameter values. The stable set of parameters are chosen to be within a stable region of controller parameters, far from unstable values. The balanced set is a compromise between the sensitive and stable sets.

Table 6.5: Three sets of controller parameters that will be used later to evaluate controller robustness.

	N_p	N_c	r_w
Sensitive	60	8	0.5
Balance	80	8	4.0
Stability	100	8	12

6.5 Model Sensitivity

Using the three sets of controller parameters from Table 6.5, the effect of varying the different components from within the linear model will be evaluated. The reasoning for doing this is that no mathematical model will ever truly replicate a real system exactly. This is due to an endless number of reasons and is beyond the scope of this work.

This sensitivity study is being carried out to evaluate how closely the linear model must approximate the real system so as to ensure that the controller is stable. It is also used to evaluate which parameters dominate system performance and require the most amount of effort to quantify an accurate value of prior to controller design.

First the effect of the mass and inertia terms are evaluated, followed by the effects of the damping and restoring terms. The effects of varying these parameters will be evaluated by the change in the overshoot, the pitch disturbance, and the depth and pitch settling times, against the change in parameter value. Previous studies [97] similar to this have evaluated the sensitivity of each parameter using the difference from the normal value for both the input and output variable such as:

$$S = \frac{(R - R_{nom})/R_{nom}}{(P - P_{nom})/P_{nom}} \tag{6.17}$$

where S is the sensitivity, R and P are the system output (response) and input (parameter) respectively, and R_{nom} and P_{nom} are the normal values with the parameters unchanged.

The difficulty with using this method to evaluate the results, for this sensitivity study, is encountered when working with values near zero, such as overshoot. Equation (6.17) calculates the change of both the input and output from its normal value. If this normal value is approximately zero then any change presents an extremely large value that may not truly reflect the actual variation in system performance. Instead, this work will present the change of measured values against the percentage change of the input parameter. Controller sensitivity needs to be evaluated carefully to determine what is an acceptable degradation in controller performance and what could potentially result in an unstable controller.

6.5.1 Mass and Inertia Terms

The linear model used within the controller design contains one mass term, m_z, and one inertia term, I_y. Both of these terms include the rigid body and the added masses and inertias. The calculation of the rigid body terms is relatively straight-forward using modern computer aided design (CAD) packages. Added mass terms are often calculated using a panel method with a simplified geometric model of the vehicle, or deduced from experimental data of the vehicle performing step changes in speed. The combination of CAD model inaccuracies, and errors calculating added mass, can result in significant errors, compared to the real system, for the mass and inertia terms. This part of the sensitivity study evaluates the effect of deliberately introducing an error on the mass and inertia terms.

Figure 6.11 presents the effect on depth overshoot by varying the mass and inertia terms $\pm 50\%$. Both the balanced and stable set of controller parameters follow very similar trends with very little variation in overshoot with varying input parameters. The sensitive controller parameters show the largest increase in overshoot with a mass term 50% lower than normal. The magnitude of the change in overshoot for all the mass terms is low, with a maximum increase of less than 0.07 m. Change in overshoot with varying inertia is insignificant with a maximum increase of less than 0.01 m with an increase of inertia of 50%. Although overshoot performance does degrade with the variation of the mass and inertia values, the system performance is still stable and the magnitude of overshoot variation is acceptably low.

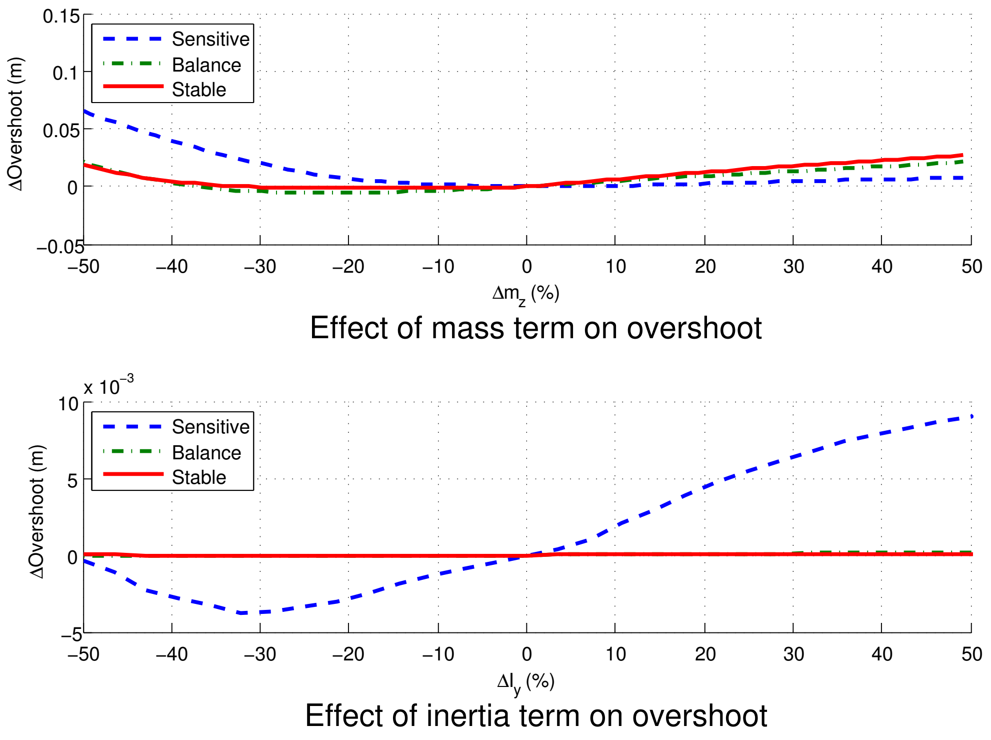


Figure 6.11: Change in overshoot against change in mass and inertia values.

Figure 6.12 presents the effect on pitch disturbance by varying the mass and inertia terms ±50%. The pitch disturbance, with the stable controller parameters, is relatively unaffected by the variation in mass and inertia terms. However, with the balanced and sensitive controller parameters, the pitch disturbance varies significantly. Decreasing mass and inertia terms both increase the magnitude of the pitch disturbance by over 0.5°. If both mass and inertia terms were 50% lower than reality then, given the trends in Figure 6.12, the system performance could degrade beyond acceptable limits or even become unstable. From these results it appears that overestimating the mass and inertia terms is less detrimental than underestimating them.

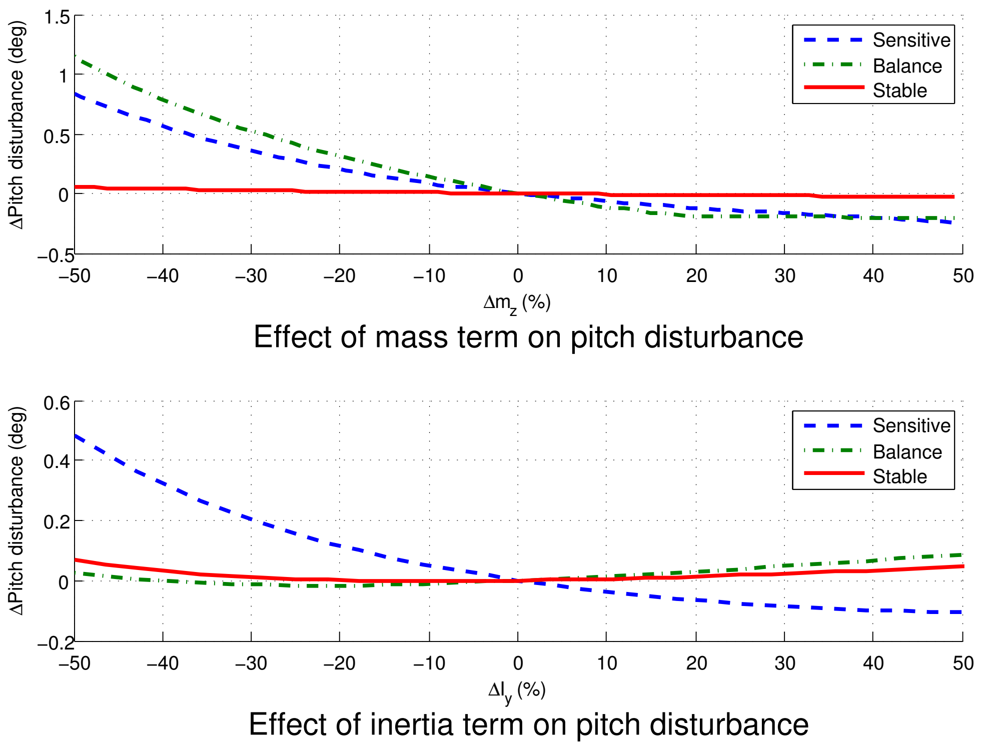


Figure 6.12: Change in pitch disturbance against change in mass and inertia values.

Figure 6.13 presents the effect on depth settling time by varying the mass and inertia terms ±50%. The variation of depth settling time is very low within ±30% of the normal mass value, beyond this the settling time increases more significantly. For the stable and balanced controller parameters with extreme Δm_z values, the change in settling time jumps from approximately zero to six seconds. This is due to the vehicle performing a second oscillation, about the set-point, that is outside of the convergence tolerance, before successfully converging. Therefore the settling time increases by approximately one period of system oscillation before converging. The magnitude of I_y has no apparent effect on depth settling time with the stable and balanced controller parameters, and a very small effect with the sensitive controller parameters.

The last parameter to evaluate is the effect on pitch settling time by varying the mass and inertia

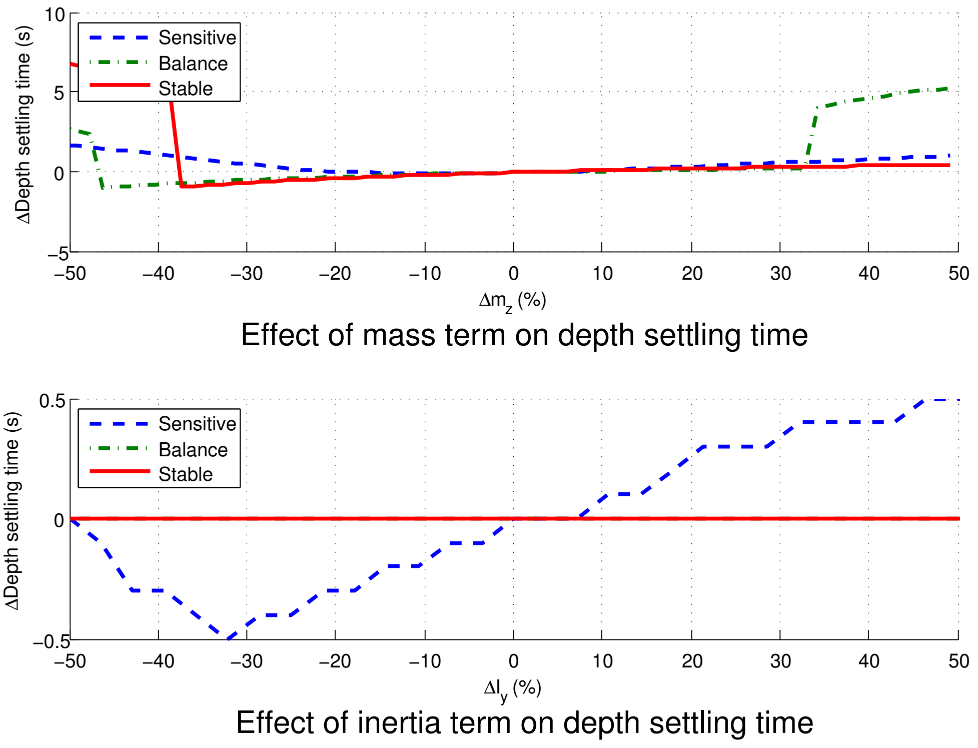


Figure 6.13: Change in depth settling time against change in mass and inertia values.

terms ±50%, Figure 6.14. Simply comparing the range of the Figures y-axes, between Figures 6.14 and 6.13, it is clear that the pitch settling time is much more sensitive to changes in mass and inertia than depth settling time. For both the stable and balanced controller parameters, the pitch settling times for the range of Δm_z and ΔI_y values vary relatively little in comparison to the results with the sensitive controller parameters. With these parameters the system becomes unstable beyond Δm_z values of +25% and ΔI_y values of +5%. From this information it is clear that if stability in the pitch axis is of priority then underestimating the mass and inertia terms is more likely to produce a stable system even if performance is not optimal.

It is clear that errors in the mass and inertia terms can have a significant effect on system performance. Within errors of ±30% of m_z and I_y terms the system performance is within acceptable limits, with the exception of pitch settling time with the sensitive controller parameters. It appears that for short term performance attributes, overshoot and pitch disturbance, then overestimating the mass and inertia terms is better. In contrast, for performance attributes found later in the simulation, depth and pitch settling times, it is more advantageous to underestimate the mass and inertia terms. The stable controller parameters provided the most reliable performance with the varying input errors. In general the sensitive controller parameters performed worse than the other parameter sets. If the controller designer is concerned that the controller parameters may be on the verge of instability then, from Figure 6.14, it would be more sensible to underestimate the mass and inertia values.

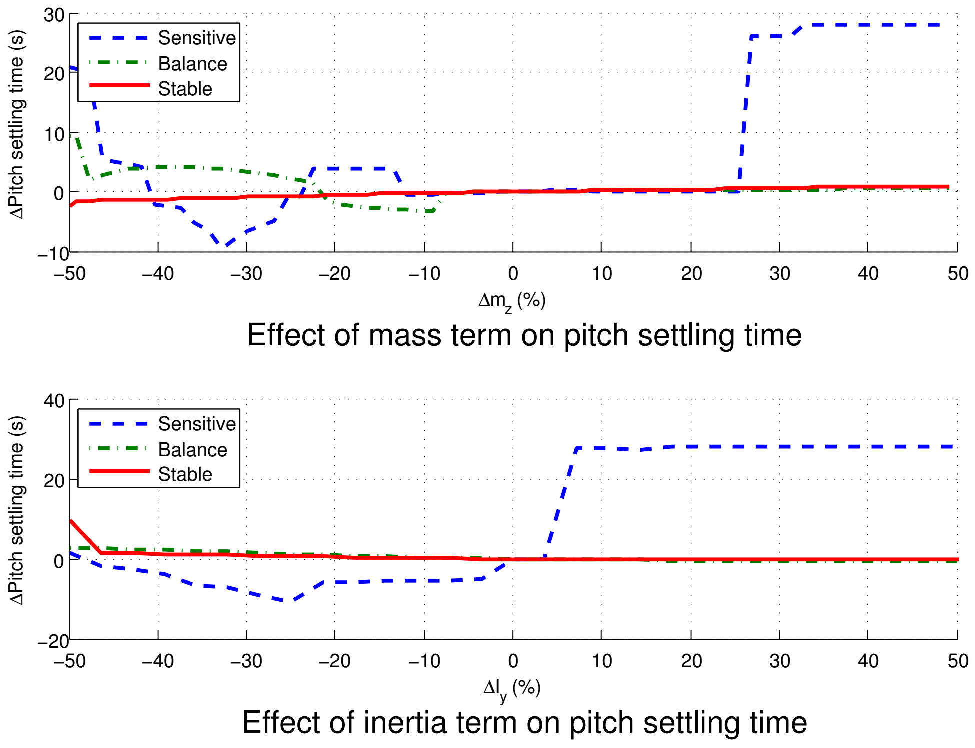


Figure 6.14: Change in pitch settling time against change in mass and inertia values.

6.5.2 Damping Terms

The linear damping terms, used in the model that is embedded within the controller, will inherently have errors when compared to reality. This is partly due to errors created when estimating the drag terms but primarily from the linearisation process. In this section, the linear damping term in the z axis is varied from -100% (zero damping) to $+150\%$ of its normal approximation, and the linear damping term in the y axis is varied from -100% (zero damping) to $+900\%$ of its normal approximation. The reason for evaluating a greater range of damping values for the y axis, compared to the z axis, is due to a greater uncertainty for the value of this damping term.

Figure 6.15 presents the effect on overshoot by varying the damping terms in the z and y axis. Variation of the k_z parameter has a notable effect on overshoot, in particularly for the stable and balanced controller parameters. The variation of overshoot for the results with the sensitive controller parameters are lower than the other two sets. Using the sensitive controller parameters, performance is actually improved (Δovershoot < 0) when Δk_q is higher than 0% . Overall, the variation in overshoot is relatively low and does not present any major cause for concern. Variation of the pitch damping term, k_q, has no notable effect on overshoot with the largest difference of just over 1 mm.

Figure 6.16 presents the effect on pitch disturbance by varying the damping terms in the z and y axis. The pitch disturbance increases with Δk_z values greater than zero, and decreases with values less than zero. The same trend applies to the change in pitch disturbance as a function of Δk_q. The

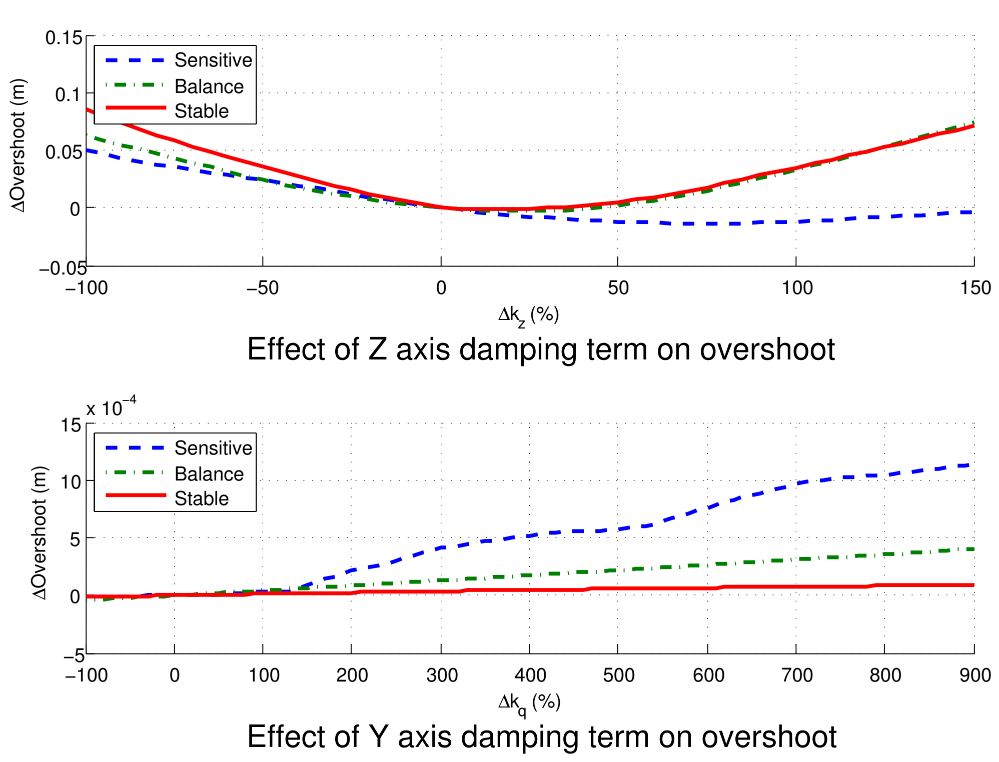


Figure 6.15: Change in overshoot against change in depth and pitch damping values.

magnitude of the pitch disturbances, for variations in both k_z and k_q, are relatively small and show no signs of instability.

Figure 6.17 presents the effect on depth settling time by varying the damping terms in the z and y axis. The first point to note from this Figure, for the effect due to the change in k_z values, is that the sensitive controller parameters perform significantly better than the other two parameter sets, with improved performance over normal conditions with k_z values greater than zero. With the stable and balanced controller parameter sets, performance degrades with Δk_z values below -30% and above $+55\%$. The system appears to become unstable, or marginally unstable, above a Δk_z value greater than 135%. Variation in the pitch damping term has no significant effect on the depth settling time.

Figure 6.18 presents the effect on pitch settling time by varying the damping terms in the z and y axis. Similar to Figure 6.17, the pitch settling time can be improved from the normal by increasing the k_z value when using the sensitive controller parameters. Unlike the depth settling time, although the performance does degrade, the pitch settling time appears to remain stable with all Δk_z values and controller parameters, with the exception of the sensitive controller parameters with a k_z near to or at zero. Surprisingly, k_q has little effect on the pitch settling time, with only extremely high values producing unstable performance when using the sensitive controller parameters.

The k_z damping term appears to have a much more significant influence on overall system performance than the k_q value. When using the stable and balanced controller parameters, choosing low

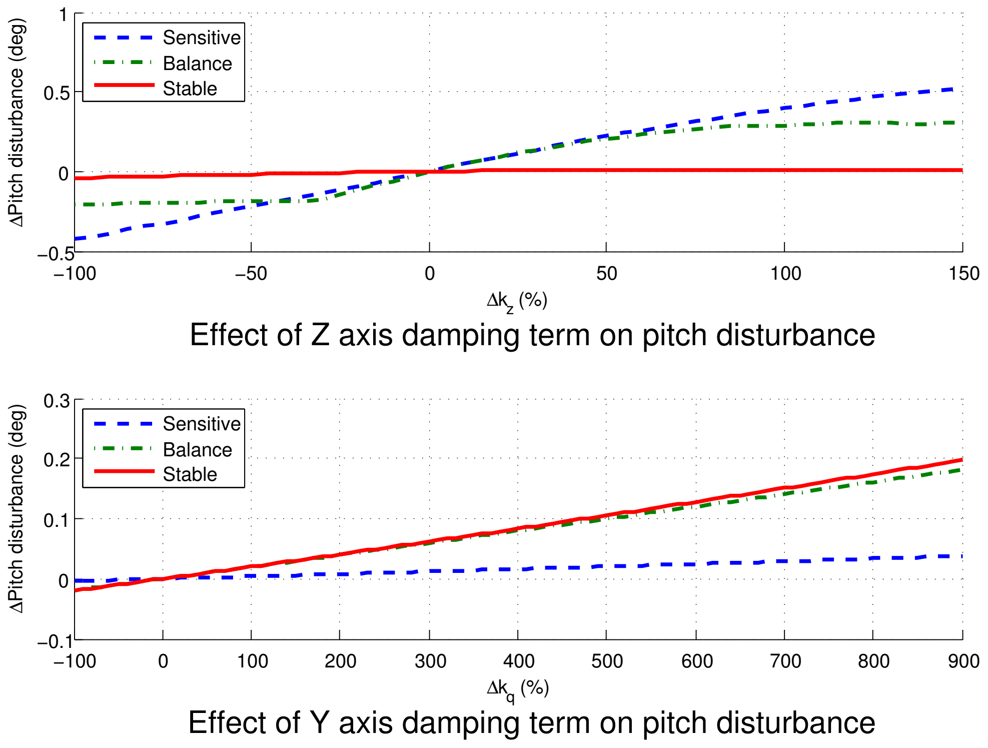


Figure 6.16: Change in pitch disturbance against change in depth and pitch damping values.

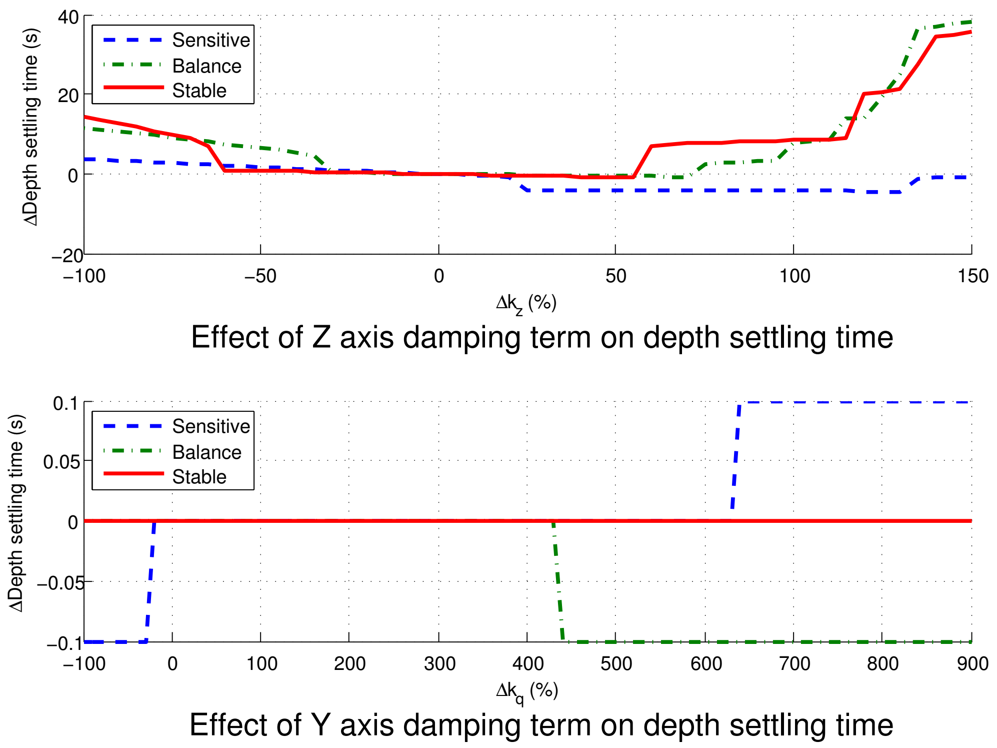


Figure 6.17: Change in depth settling time against change in depth and pitch damping values.

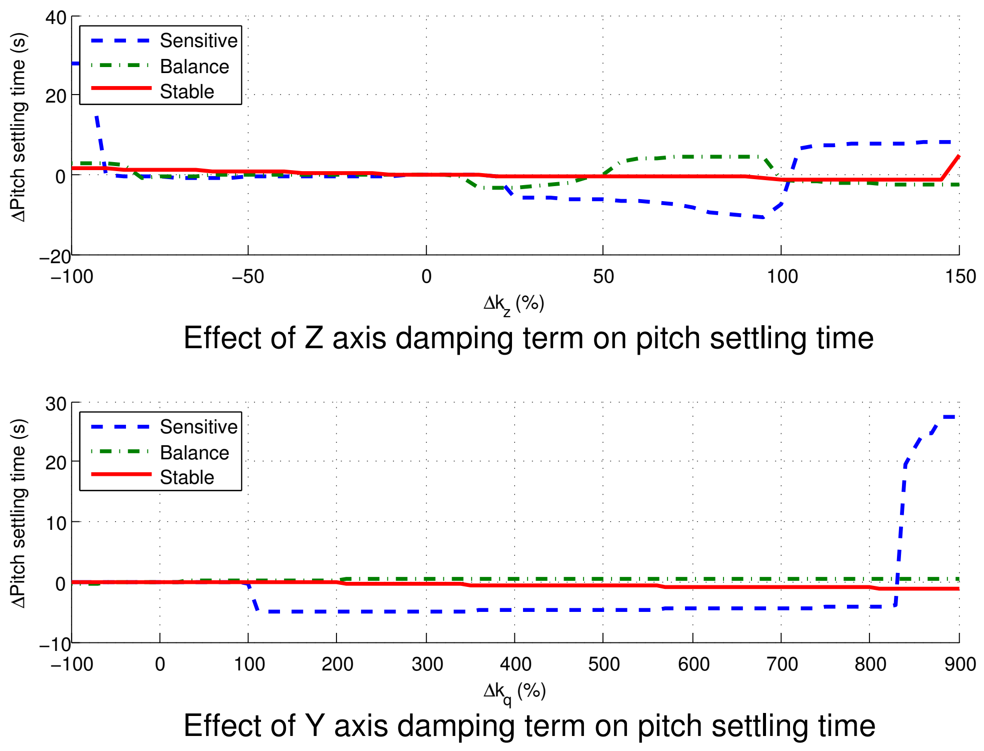


Figure 6.18: Change in pitch settling time against change in depth and pitch damping values.

values for the damping terms will improve the probability of system stability, with the compromise of a slightly larger overshoot. Improved performance over the normal settings can be achieved with the sensitive controller parameters by using higher k_z values. However, it should be noted that this is only for this controller and should not be taken as a general rule for this system.

6.5.3 Restoring Moment

The final component to be evaluated is the effect on system performance due to varying the magnitude of the restoring moment. The z_g parameter is commonly found using CAD software to accurately calculate the location of the centre of gravity and centre of volume. Differences between the CAD model and reality, such as not including cabling or manufacturing inaccuracies, will alter the magnitude of the restoring moment parameter. This section evaluates the effect of varying the magnitude of the restoring moment parameter ($z_g W$) from -100% (zero) to $+900\%$ of its normal value.

From Figure 6.19 it can be seen that altering the restoring moment parameter has no significant effect on the depth control, with little variation in overshoot or depth settling time. However there is significant variation on the pitch disturbance angles and pitch settling times. The pitch settling time is a good indicator as to the stability of the system in the pitch axis. For all controller sets, large values of the restoring moment parameter resulted in pitch instability, with the stable controller parameter set the first to become unstable at a $\Delta z_g W$ value of $+150\%$. The instability indicated

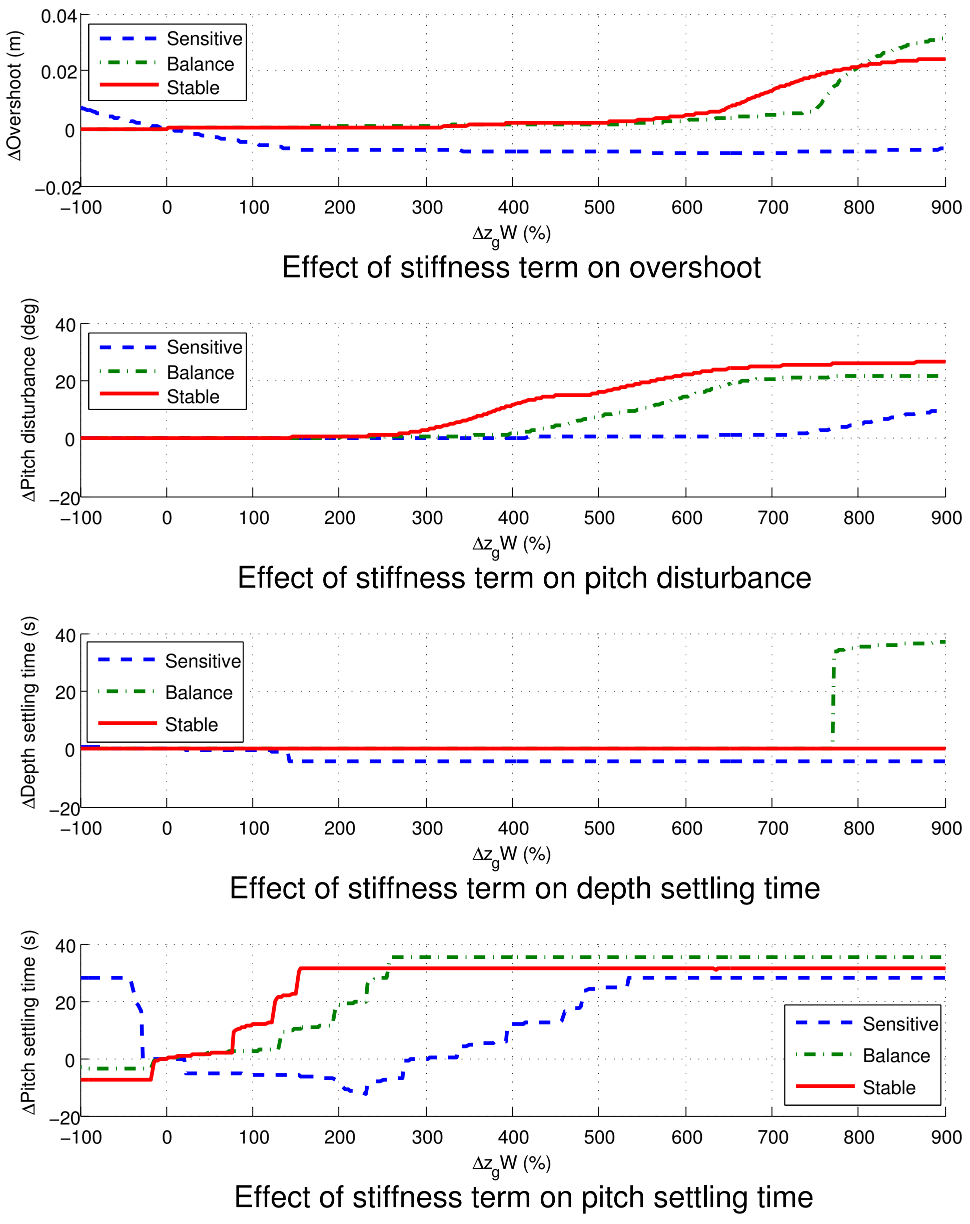


Figure 6.19: Change in performance against change in mass and inertia, depth and pitch damping values and restoring moment.

by the pitch settling time means that the pitch has not converged within $\pm 0.1°$. An oscillation of near this tolerance would not be deemed detrimental to vehicle performance. The magnitude of this oscillation can be approximated using the magnitude of the pitch disturbance. From this it is clear that the vehicle becomes unstable above a $\Delta z_g W$ value of +250%.

6.5.4 Summary

With the stable or balanced set of controller parameters, the system's sensitivity to the majority of the variables is very low. With the sensitive controller parameters the system performance is unpredictable, yet with these parameters, system performance can actually be improved by varying the model terms. It is believed varying the controller model so as to improve system performance could prove to be a useful tuning tool. However, it may also be potentially dangerous and should only be performed once a thorough effort to tune the system using the controller parameters, as discussed in Section 6.4, has been exhausted.

Surprisingly what appears to be one of the most sensitive parameters is the naturally stabilising z_gW term. Although with the stable and balanced controller parameters it is possible to omit the z_gW term completely without hindering system performance. A list of acceptable errors, from this study, for each term is given in Table 6.6.

Table 6.6: Acceptable magnitude of errors for the parameters within the controller model with the stable and balanced controller parameter sets.

Term	**Error**
M	$\pm35\%$
I	$\pm50\%$
k_d	$-100 \rightarrow 150\%$
k_p	$-100 \rightarrow 800\%$
BG	$-100 \rightarrow 80\%$

6.6 Feedback Quality

Until now, the non-linear model that is used as the system when performing simulations, has provided full state feedback to the controller. The reason for this is to simplify the evaluation process by reducing the number of variables influencing the controller performance. In reality, full state feedback is rarely available and, even if it is, the quality of the feedback may be poor due to measurement noise or random errors.

On the Delphin2 AUV, the depth and pitch are measured using a pressure transducer and a three-axis accelerometer respectively. As was noted in Section 3.3, neither of these signals are perfectly accurate nor noise free. Therefore to improve the signal to noise ratio of the feedback, and provide an estimate for the unmeasured states, two different methods are proposed; a polynomial based filter and an observer. In this section both methods are implemented in simulation, with realistic levels of measurement noise added to both the depth and pitch signals, and their performance evaluated.

6.6.1 Polynomial-Type Filter

Before evaluating the performance of the polynomial type (PT) filter with noisy signals it will be tested with the controller without the addition of measurement noise. This is an important first step enabling the operator to identify any problems that the filter may induce into the system that may otherwise have been attributed to the measurement noise.

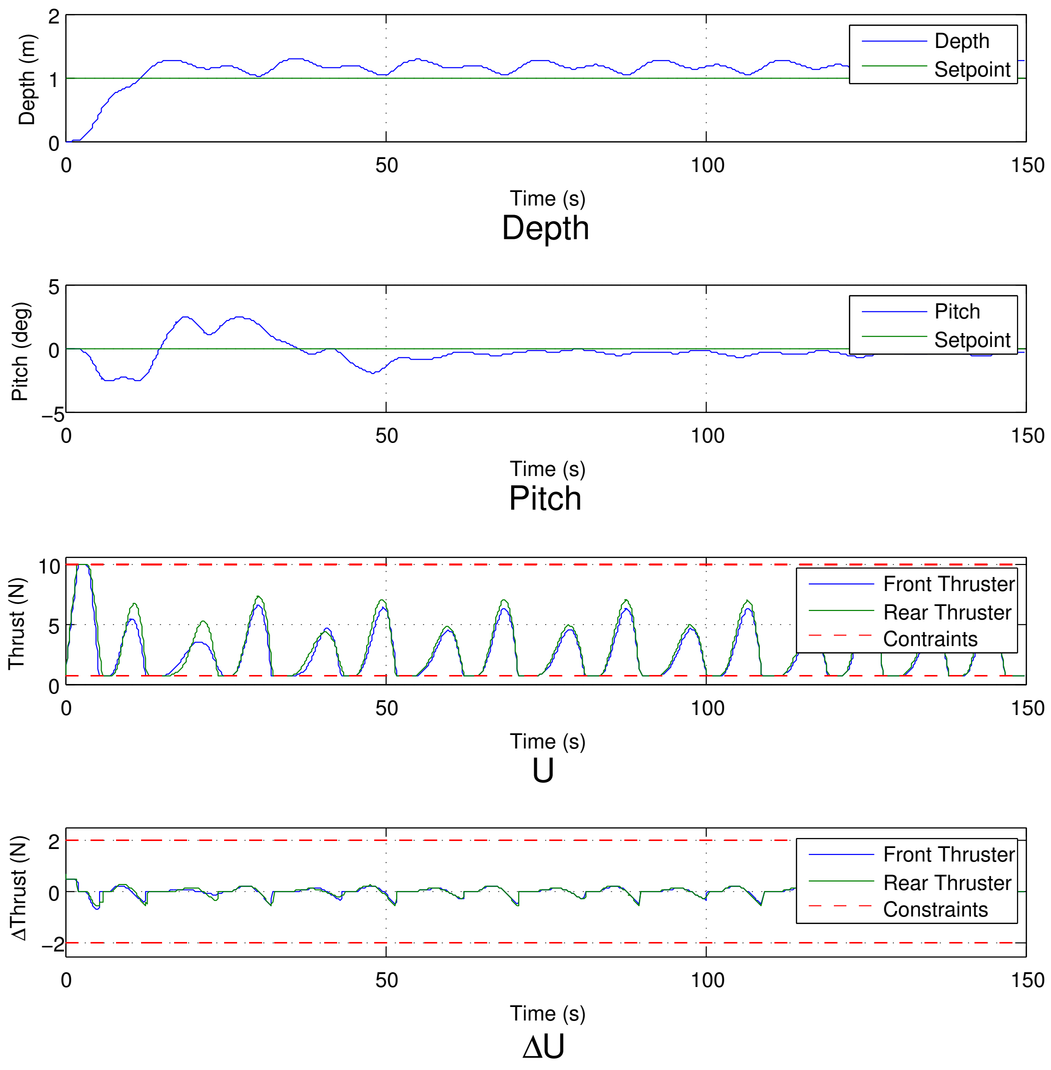


Figure 6.20: Simulation results of the depth and pitch MPC controller using the polynomial type filter to provide state feedback. Note that noise has not been added to either the depth or pitch signals yet system performance is very poor.

The filter has been designed using the methodology stated in Section 3.3.3. The filter works by fitting second-order polynomial functions to arrays of the last f depth and pitch samples. The fitted polynomial functions for depth and pitch are defined as P_Z and P_θ, with their derivative functions defined as $\dot{P}_Z$ and $\dot{P}_\theta$ respectively. The state-variable vector is then estimated by evaluating the respective polynomials:

$$x(k) = \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} = \begin{bmatrix} P_Z(0) - P_Z(-dt) \\ \dot{P}_Z(0) - \dot{P}_Z(-dt) \\ P_\theta(0) - P_\theta(-dt) \\ \dot{P}_\theta(0) - \dot{P}_\theta(-dt) \\ P_Z(0) \\ P_\theta(0) \end{bmatrix} \tag{6.18}$$

where $P_Z(0)$ and $P_Z(-dt)$ are the values of the depth polynomial evaluated at time equal to zero and minus one time step (inverse of controller sampling rate).

A simulation is performed using the PT filter with array lengths of 32 and 18 for the depth and pitch signals. These lengths were chosen earlier in Section 3.3.3. The simulation has zero initial conditions and the controller is tasked with achieving depth and pitch set-points of 1.0 m and 0.0°, respectively, using the balanced set of controller parameters. Figure 6.20 presents the results of this simulation. Clearly the performance has deteriorated beyond acceptable levels. Neither depth or pitch converges to the set-points and the thrust is oscillating wildly.

To investigate what is inducing this instability the filtered values are compared with the unfiltered signal. Given that there is no additional noise, the filtered signal should track the unfiltered signal almost exactly. Figures 6.21 and 6.22 present the heave velocities and pitch rates and the errors normalised by the standard deviations of the heave velocities and pitch rates, respectively.

Out of the two signals the heave velocity produces the largest errors, often greater than 200%. It is believed that these large errors are what is inducing a limit cycling behaviour within the controller. Looking at both figures, the errors are largest at the peaks and troughs of the oscillations. This is due to the polynomial being too long and therefore 'slow' to react to changes in signal trajectory. In order to fix this problem requires that the length of the arrays are shortened. This in itself will potentially cause problems as shortening the array lengths will reduce the magnitude of attenuation of any random errors on the measured signal. In order to retain stable performance, with the filter in the loop, the depth and pitch arrays have been shorted significantly to 8 and 6 samples respectively.

With the addition of realistic measurement noise on both the depth and pitch signals, Figure 6.23 demonstrates that the filter performance is far from adequate. The overall system performance is far beyond acceptable with neither output converging accurately to the desired set-points. Visually examining the controller outputs, ΔU and U, shows that after the initial dive phase the signals appear to be dominated by the measurement noise on the feedback signal.

Another possible method of improving the performance of this controller would be to use the original array lengths but increase the sampling rate of the depth and pitch signals. This should improve the frequency response of the filter whilst maintaining the good noise attenuation capabilities. This assumes that the magnitude of the measurement noise is not influenced by the sampling rate.

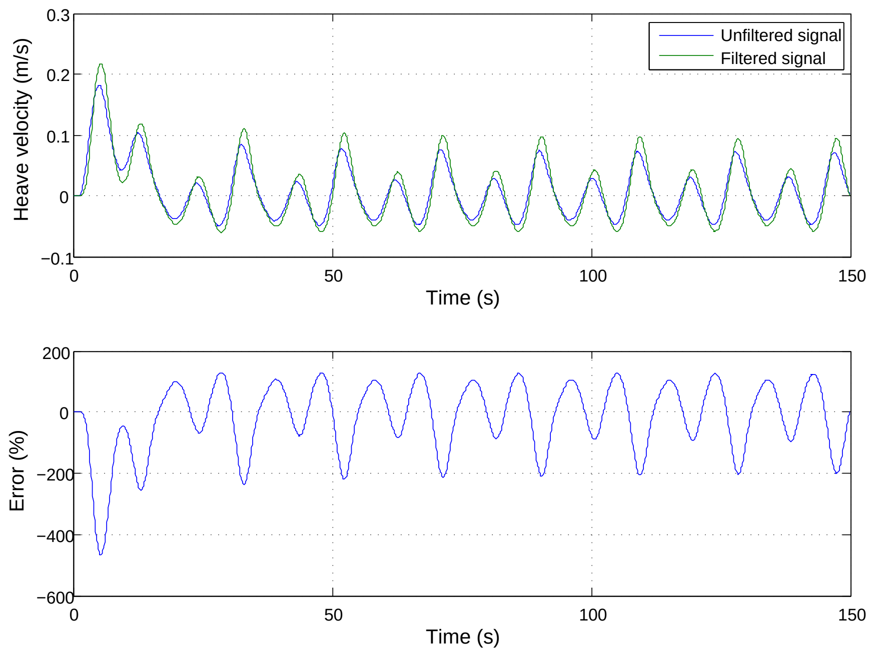


Figure 6.21: Comparison between the estimated heave velocity, from the polynomial type filter, and the actual heave velocity provided by the non-linear model. Note that the bottom plot is the of the error between the two signals normalised by the RMS heave velocity.

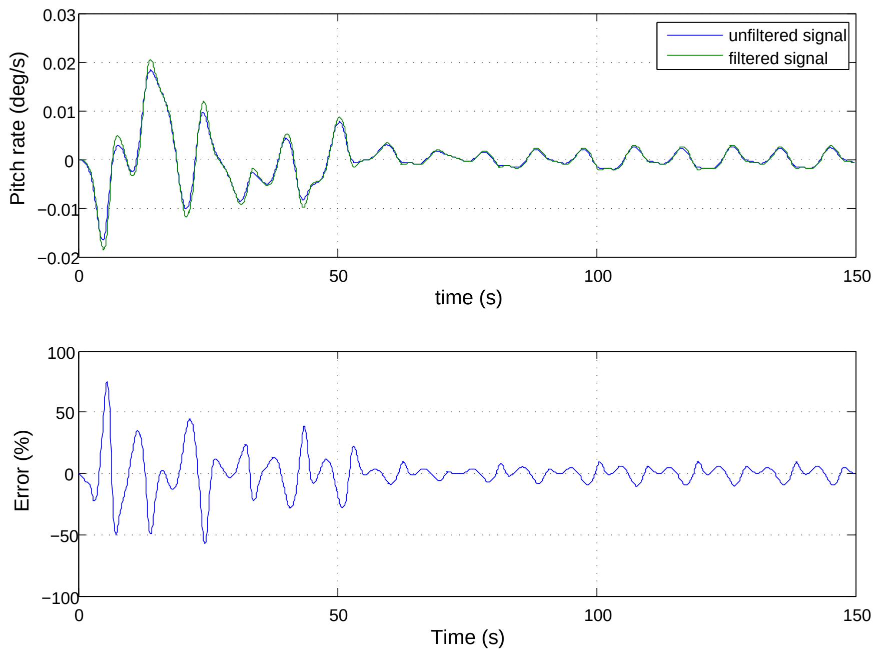


Figure 6.22: Comparison between the estimated pitch rate, from the polynomial type filter, and the actual pitch rate provided by the non-linear model. Note that the bottom plot is the of the error between the two signals normalised by the RMS pitch rate.

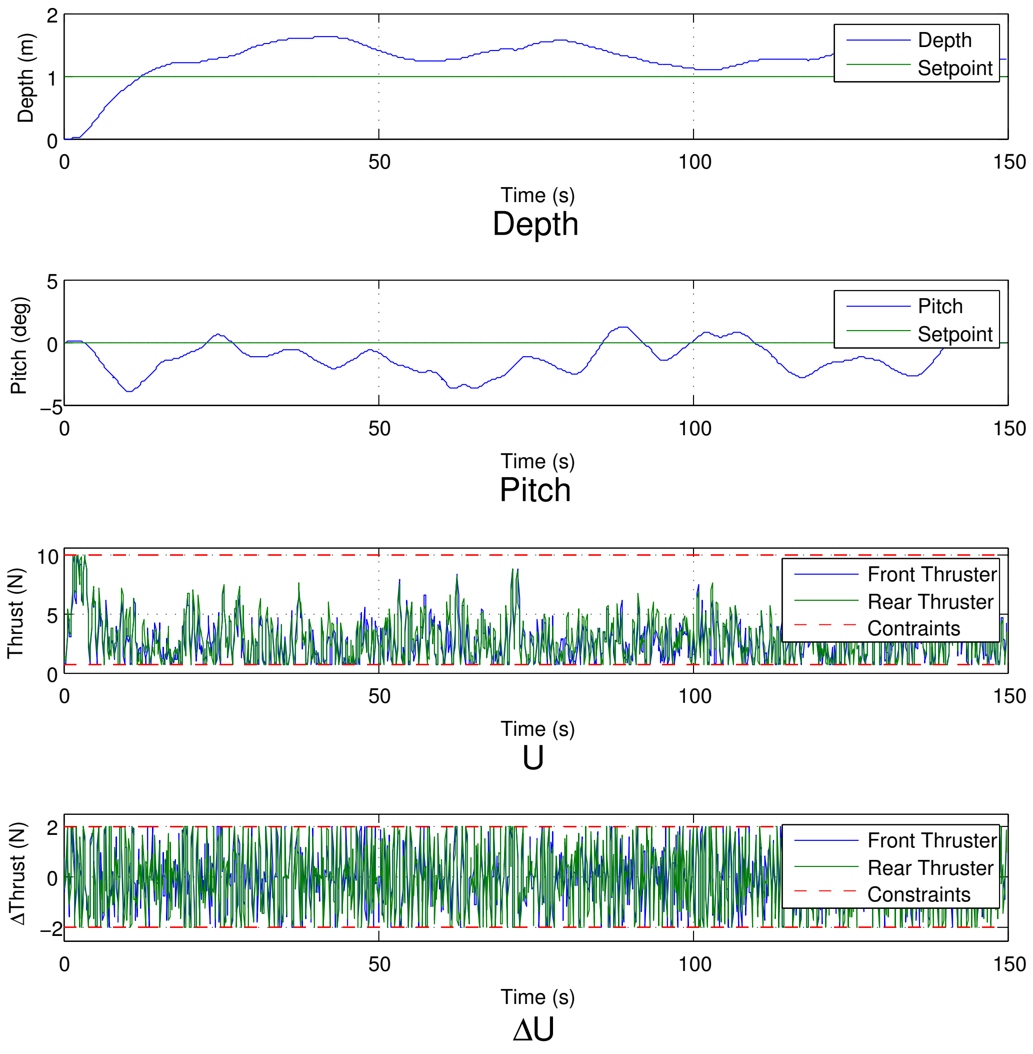


Figure 6.23: Simulation results of the depth and pitch filter using the polynomial type filter, with the addition of measurement noise on the depth and pitch signals. Note that performance is poor with noise dominating the controller output.

6.6.2 Observed Feedback

Given that the controller has been shown to be extremely sensitive to measurement noise, and that the proposed PT filter fails to provide acceptable performance, an alternative method of achieving full state-feedback needs to be investigated. One possible method, when measurement of all the states is not feasible or too noisy, is to use an observer or state estimator, see Section 3.3.4.

To implement the observer, the system needs to be checked to ensure that there are no unobservable states:

$$\text{Number of unobservable states} = dim(A) - rank \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} = 0 \tag{6.19}$$

As all the states are observable, an observer can be designed so as the estimated state-variable and output vectors are calculated as:

$$\hat{x}(k+1) = A\hat{x} + L\left[y(k) - \hat{y}(k)\right] + Bu(k) \tag{6.20}$$

$$\hat{y}(k) = C\hat{x} \tag{6.21}$$

The inclusion of the errors, between the measured and observed output vectors, within the estimation of the state-variable vector provides feedback to the observer. Assuming appropriate values for the L matrix are chosen, then the observer should be stable and all the states should converge to their correct values.

For this work, the MATLAB function 'dlqr' is used to calculate the L matrix. To calculate L, this function uses the A and C matrices along with two weighting matrices, Q and R, with dimensions of $m \times m$ and $q \times q$ respectively. These matrices help weight a quadratic function within the 'dlqr' function. For tuning simplicity the Q matrix has been fixed as the identity matrix, while R is an identity matrix multiplied by a scalar r_o. This scalar is used to tune the observer with smaller values favouring quicker convergence while larger values will provide slower convergence but with the benefit of better noise attenuation. To evaluate the controller performance with the observer, five r_o values are used; 0.1, 1, 10, 100 and 1000. The controller performance is analysed using the same four parameters used when tuning the controller (see Figure 6.3); depth overshoot, pitch disturbance, and the settling times of depth and pitch.

None of the tests converged to within the pitch tolerance of $\pm 0.1°$ however, the maximum pitch disturbance stayed within $0.5°$ for all the different r_o values yet showed no obvious trend with r_o. By visually examining the effect of r_o on the pitch control, values between 10 and 100 appeared to provide fast convergence and good stability.

The r_o value of 1000 did provide stable pitch performance but the pitch oscillated for some time near the beginning of the simulation. This may be due to the pitch or pitch rate failing to converge to the correct values quickly enough. Realistically, convergence to within $\pm 0.1°$, when there is measurement noise on the feedback signals, is unlikely. For normal operation maintaining the desired pitch within $\pm 0.5°$ should be acceptable, which the controller did achieve.

Figure 6.24 presents overshoot (upper plot) and depth settling time (lower plot) against r_o values. As can be seen, all the r_o values enable the controller to stay within the overshoot tolerance of 0.1m. The lowest overshoot was achieved by using an r_o value of 10 while the minimum depth settling time was achieved using an r_o value of 100.

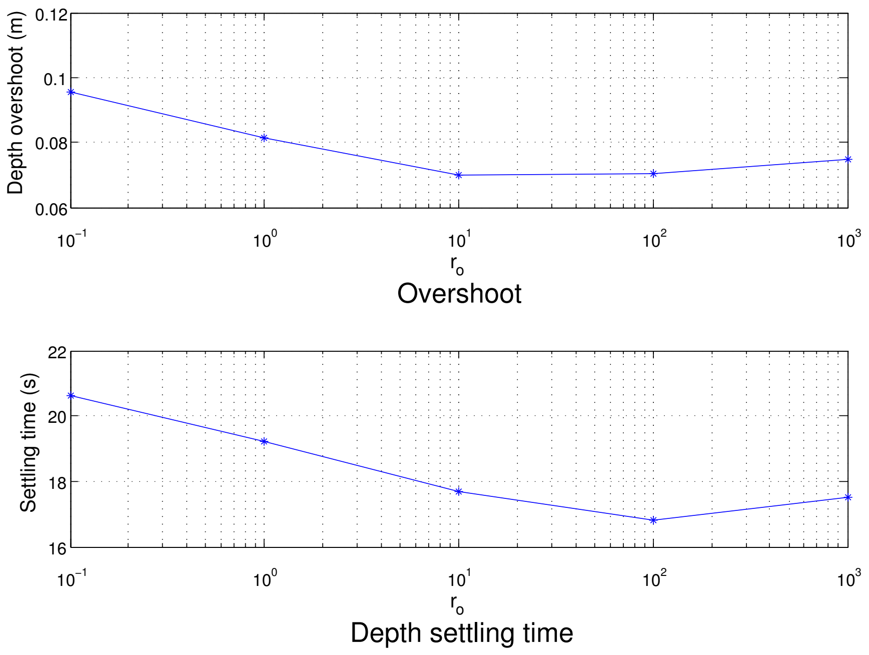


Figure 6.24: Effect of the observer gain, r_o, on the overshoot and depth settling time.

A compromise between depth and pitch performance is required and, from these simulations, it appears that an r_o value between 10 and 100 would work well. Although system performance does vary with r_o, it did not show any signs of system instability despite a wide range of values tested. This provides confidence that, when implemented on the real vehicle, the observer performance should be acceptable and that the operator can safely tune the observer without fear that the system may suddenly become unstable.

Figure 6.25 presents the simulation results using an r_o value of 10. As can be seen, the depth and pitch both converge quickly and smoothly to their appropriate set-points. The ΔU and U signals do show some noise, however, in comparison to Figure 6.23, the effect of the noise has been greatly reduced to within acceptable limits.

6.6.3 Summary

Clearly the addition of measurement noise severely effects the controller performance. The use of the polynomial type filter does show some potential at improving performance however it also includes risk of adding additional dynamics that can cause the controller performance to degrade. The use of the observer is essential, enabling the controller to perform well despite realistic noise levels.

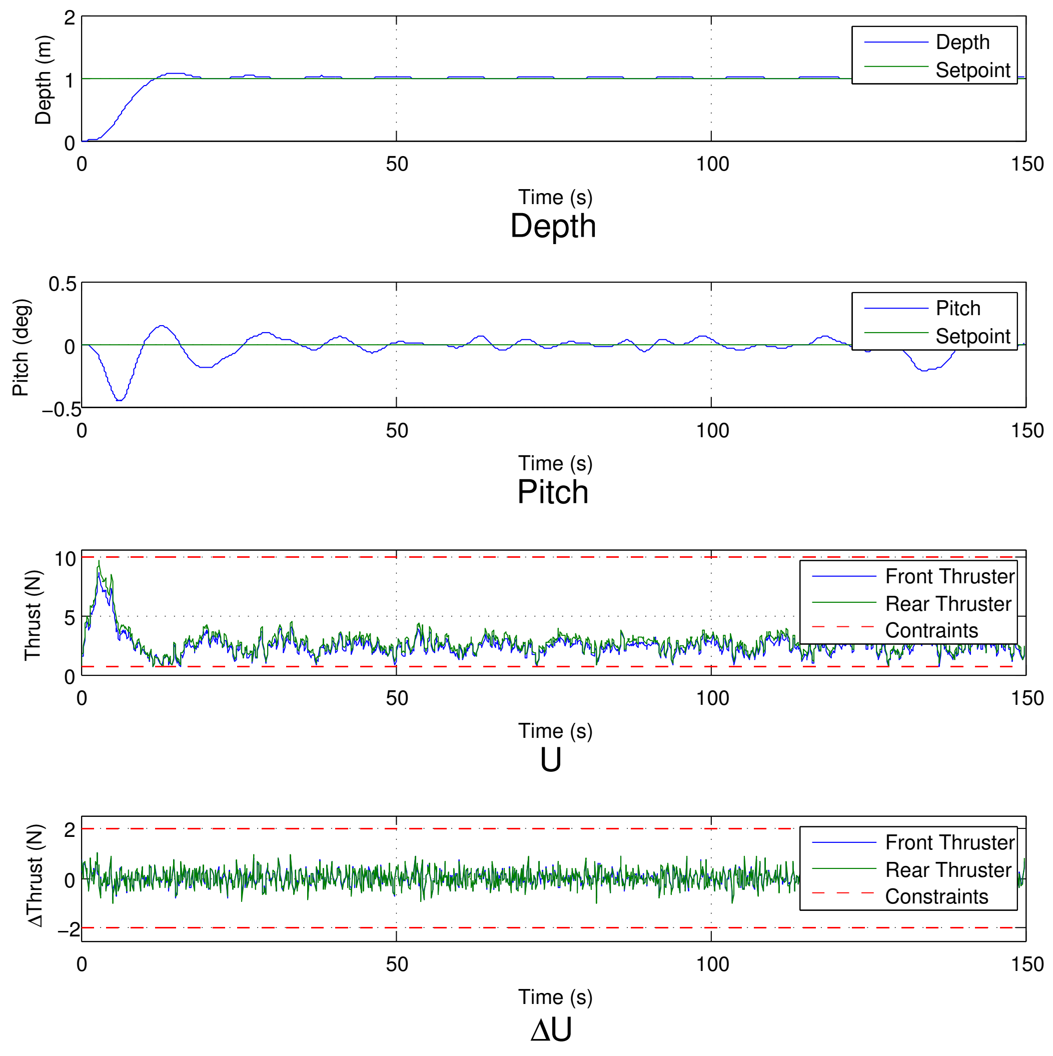


Figure 6.25: Simulation results of the MPC controller using the observer with an r_o value of 10. Note that the noise still effects the controller performance but within acceptable limits.

6.7 Experimental Verification

Before continuing onto a more difficult control problem in the next chapter, it is important that the controller is experimentally tested on the Delphin2 AUV. The controller, and the necessary tools used by the controller, are translated from MATLAB syntax into the Python programming language and implemented as a separate node within the Delphin2 AUV control software.

The purpose of evaluating the controller performance experimentally serves two purposes; first to verify that the simulation results, using the non-linear model of the AUV, closely match that of the experimental results thus ensuring the model is accurate, and also to ensure that the MPC controller performs within specifications.

The experiments are conducted in the A.B. Wood tank at the University of Southampton. Each test comprises of three depth set-points of 1.0 m, 3.0 m and 2.0 m. Once the AUV has stayed within ±0.2 m of the depth set-point for 60 seconds, it then moves onto the next depth set-point. The pitch set-point is fixed at 0.0° for all of the tests. An observer gain r_o of 50 is used for all of the tests to provide full state feedback.

One variable that often changes slightly on Delphin2 is the buoyancy force. This is due to variations in the payload on-board the vehicle. The simulations that have been performed in this chapter have used a buoyancy force of 5 N. On checking the buoyancy force (by calculating the mean average of the thrust values when stable at a depth set-point) the buoyancy force is found to be much lower at 3.35 N. The effect of this reduction in buoyancy will be discussed later in this section.

6.7.1 Effect of Tuning Parameters

Figures 6.26, 6.27 and 6.28 presents the experimental results for the sensitive, balanced and stable tuning parameter sets respectively (Table 6.5). In each Figure the depth, pitch, u and Δu signals are plotted against time.

Initial examination of the experimental results shows that all three controller parameter sets provide stable control of both depth and pitch. There is however significant overshoot of the depth demand when using any of the three parameter sets. The magnitude of the overshoot values, for the first depth demand of 1.0 m, follows a similar trend to that found in the tuning section of this chapter, see section 6.4. This trend results in lower overshoot values for higher N_p values, with a minimum overshoot of 0.15 m with an N_p value of 100. For the second depth demand of 3.0 m this trend is reversed, with the largest overshoot of 0.382 m using an N_p value of 100.

The cause of this reversal in trend is due to the magnitude of the step change in depth demand. By revisiting the conclusions from the tuning section, the lower N_p values cause the system to behave more aggressively with rapid changes in u values, whilst the higher N_p values along with higher r_w values cause the controller to perform in a more conservative manner. For the first step change of 1.0 m, the more aggressive controller with an N_p value of 60 causes the system to accelerate quickly, resulting in a large heave velocity which in turn results in the vehicle overshooting the depth demand. In contrast the more conservative controller parameters do not generate high heave velocities so their corresponding overshoots are lower.

This variation in controller 'aggressiveness' leads to the reversal in overshoot trends when the depth step change is greater. In this case all the controller parameters cause high heave velocities however, as the vehicle approaches the depth set-point the conservative controller parameters are slower to reduce the thrust magnitude and so their heave velocities decrease slower than the more aggressive controller parameters resulting in a greater overshoot.

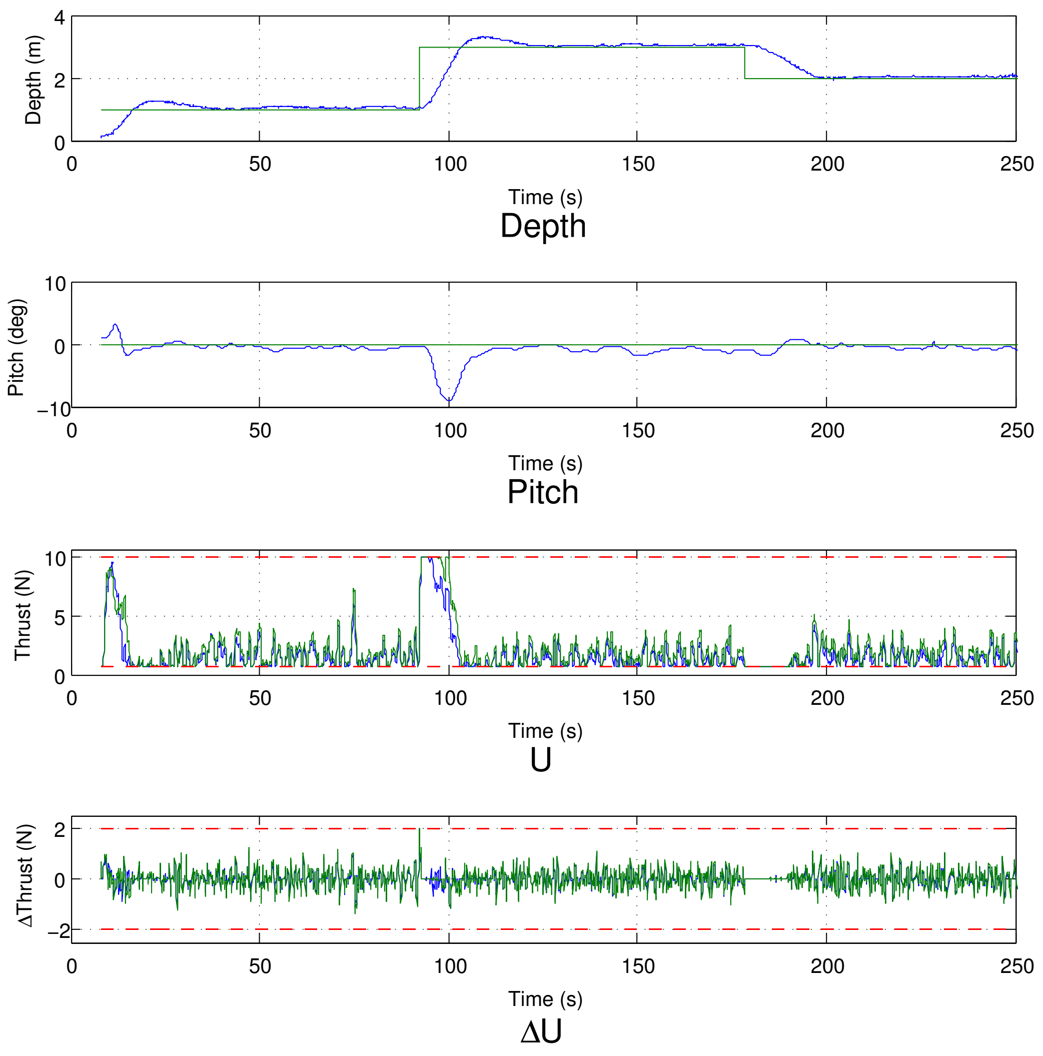


Figure 6.26: Experimental results of the MPC controller with depth set-points of 1 m, 3 m and 2 m. The pitch set-point remains fixed at 0.0°. Controller parameters: $N_p = 60$, $N_c = 8$, $r_w = 0.5$, $r_o = 50$.

All of the overshoots experienced by the vehicle, when increasing in depth, are beyond the acceptable 0.1 m tolerance defined earlier in this chapter. The simulations presented in the tuning section display little, if any, overshoot when diving from the surface to 1.0 m whilst experimentally there are significant overshoots. The cause of the overshoot from 1.0 to 3.0 m has been explained in the previous two paragraphs, here the first step change in depth demand will be discussed. The cause of this increased overshoot can be attributed to two variations between the earlier simulations and these experiments. Firstly the buoyancy of the vehicle, a disturbance force acting against the thruster

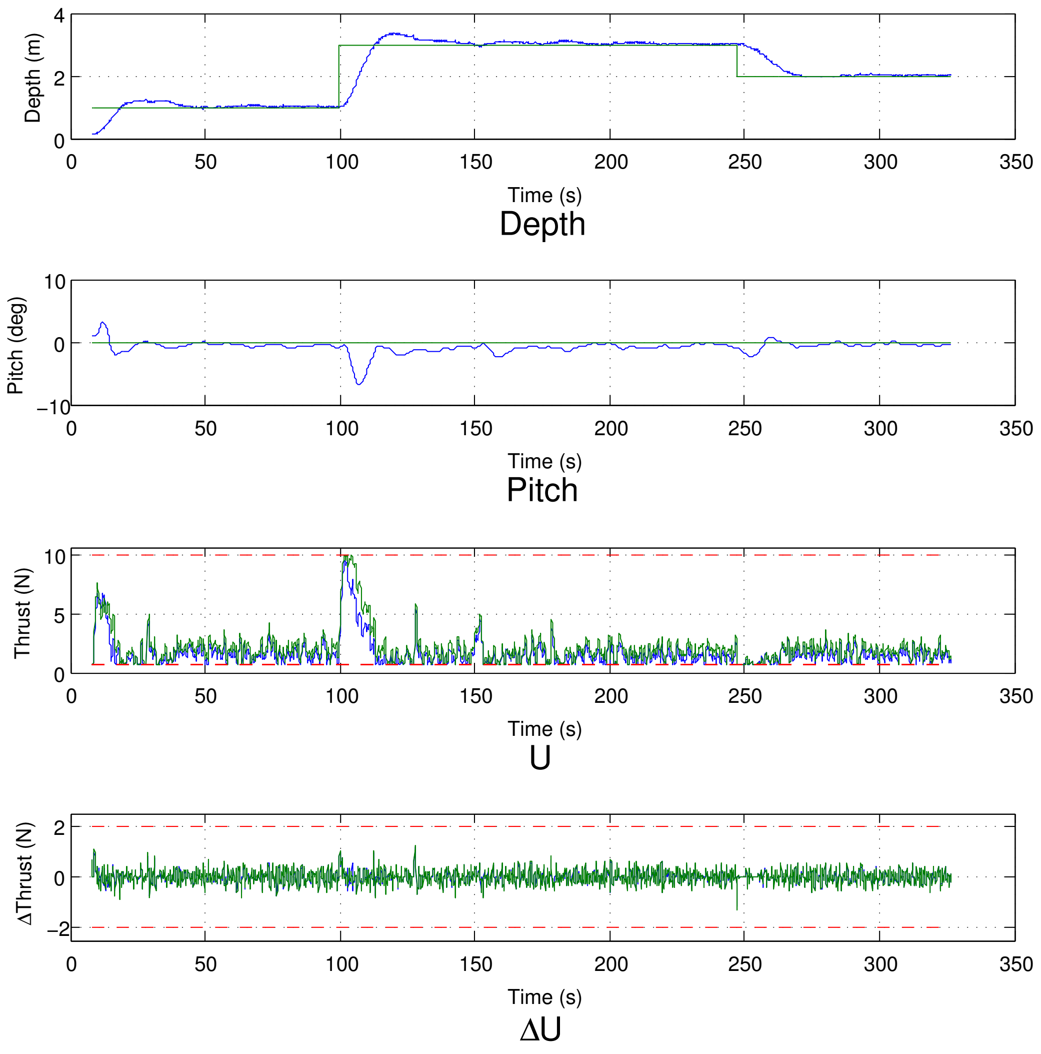


Figure 6.27: Experimental results of the MPC controller with depth set-points of 1 m, 3 m and 2 m. The pitch set-point remains fixed at 0.0°. Controller parameters: $N_p = 80$, $N_c = 8$, $r_w = 4.0$, $r_o = 50$.

forces, is 33% lower than in the earlier simulations. This means that when the controller attempts to slow down (its heave velocity) the force acting against it is lower, and so its negative acceleration is also lower thus causing the vehicle to overshoot its set-point. Examining the u signal for all three plots helps demonstrate this point. When the vehicle is stable, at each depth set-point, the magnitude of thrust is very close to the minimum thrust constraint. When the vehicle is decreasing its depth, from 3.0 m to 2.0 m, the controller has the freedom to slow the vehicle using the thrusters, and as a result, the overshoot is effectively zero. The effect of the low buoyancy can be easily understood if the

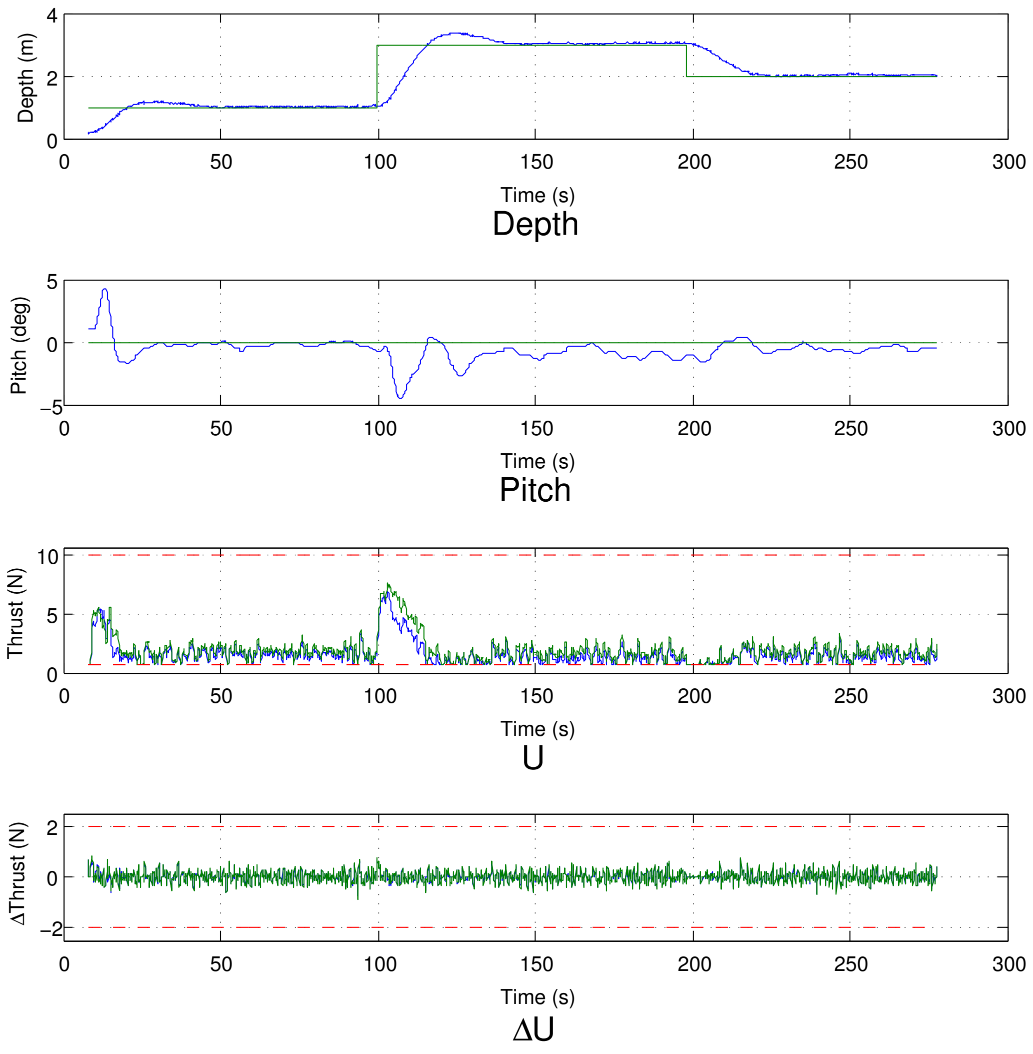


Figure 6.28: Experimental results of the MPC controller with depth set-points of 1 m, 3 m and 2 m. The pitch set-point remains fixed at 0.0°. Controller parameters: $N_p = 100$, $N_c = 8$, $r_w = 12.0$, $r_o = 50$.

buoyancy is imagined to equal zero then, with the thrusters constrained to develop positive forces, the system is guaranteed to overshoot and continue to dive far beyond its depth set-point. The reason that the thrusters were constrained to produce positive forces is to avoid non-linear behaviour due to the thruster motor dead-band.

Another factor that is not included within the simulation is the reduction in effective thrust when operating on or near the free surface, see Section 5.3.5. In the experiments this reduction in thrust near the surface causes the total thrust values, from the controller, to 'wind up,' such that the magnitude of thrust increases quickly. When the vehicle reaches a depth, at which the free surface no longer effects the thruster performance, the thrust demands are too high causing the vehicle to accelerate quickly and overshoot the set-point.

The controller's ability to maintain zero pitch worked reasonably well. The error was relatively small for most of the test duration however the vehicle failed to accurately converge to a zero pitch angle. This again can be partly attributed to the low vehicle buoyancy. As the vehicle requires a difference between the front and rear thrust values in order maintain zero pitch, due to the different lengths of each moment arm, each time both thrust values are constrained to their minimum values this difference is lost.

It can be speculated that with a high buoyancy force, thus moving the average thrust values away from the minimum constraints, and if the measurement noise on the feedback was lower, then the pitch would converge faster. Realistically, maintaining within one or two degrees should be sufficient for the majority of inspection type operations.

6.7.2 Model Verification

To ensure that the non-linear model (used to simulate the AUV) is accurate, simulations were performed using the same controller parameters and depth set-points. Figure 6.29 presents simulated and experimental results using the stable controller parameters. Note that the results shown in the Figure do not include the initial dive from the free surface as the reduction in effective thrust at shallow depths is not included within the non-linear model.

Both the simulated and experimental depth signals closely match, especially during the initial dive and overshoot. There is a slight error between both signals when the heave velocity is negative however this error is very small. This error is possibly due to asymmetry in the drag characteristics that is not included within the non-linear model.

The pitch signals do not agree as closely as the depth signals. The disturbances to pitch, whilst diving to the depth set-point of 3.0 m, show some resemblance but not accurately. It is believed that the pitch response is more sensitive to measurement noise than depth as the damping is lower, therefore the pitch signals will be more random and less likely to match.

6.8 Conclusions

In this chapter a relatively simple control problem has been addressed using the MPC algorithm. This problem required that the depth and pitch of the Delphin2 AUV be controlled using the front and rear vertical through-body tunnel thrusters.

In the first two sections, the non-linear depth and pitch system was examined and a linearised state-space model produced that approximates the vehicle dynamics. The controller was then built

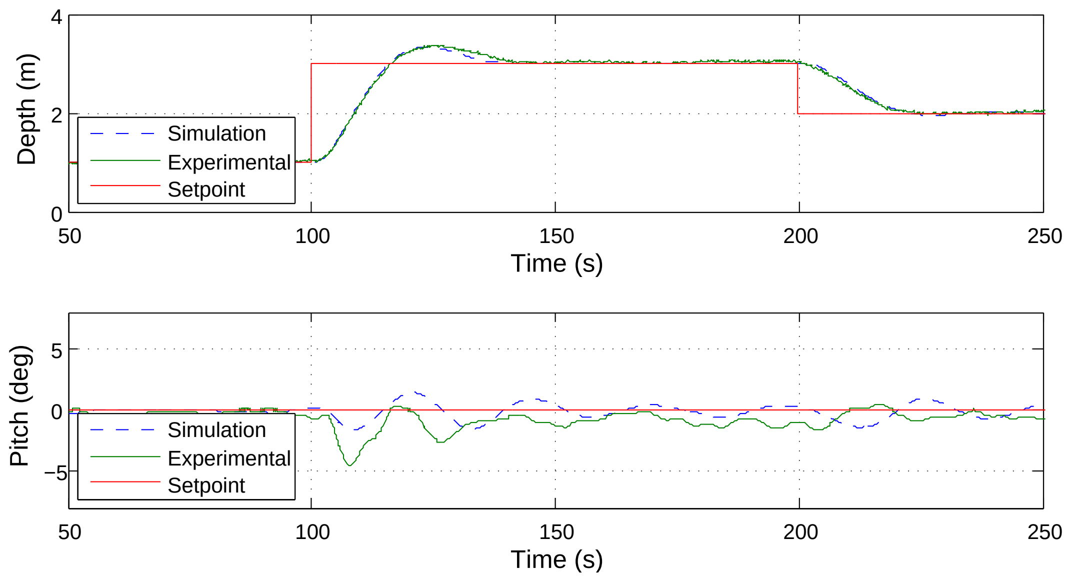


Figure 6.29: Comparison between simulated and experimental results of the MPC controller with depth set-points of 3 m and 2 m, whilst maintaining 0.0° pitch. Note both simulated and experimental depth control results are very similar with a slight divergence when the heave velocity is negative. The pitch does not agree as closely.

using the method described in Chapter 3 and input constraints applied to both front and rear thrusters.

A series of simulations were performed in order to evaluate the effects of the different controller parameters that can be used to tune the controller (the length of the prediction and control horizons, N_p and N_c, and the weighting scalar r_w). A simple method to estimate the minimum N_p value using the linearised model proved accurate, as simulation results with an N_p value below this value were unstable. The results of these simulations show that, in general, by using longer prediction horizons the probability that the controller will be stable increases and the sensitivity of the controller to the other two parameters is reduced.

The weighting scalar, r_w, can be thought of as the inverse to the gain used by a classical controller. Therefore, a high r_w value causes a slow reacting system and vice versa. Suitable r_w values increase exponentially with increasing N_p values. The control horizon has the least significant effect on the controller performance, for this system a length of four or more samples generally provides stable performance.

Another series of examinations were conducted to evaluate the effect of deliberately introducing errors within the linearised model. In an overview, it appears the MPC controller is very insensitive to modelling errors. Mass and inertia terms appear the most critical with errors greater than 35% causing significant performance degradation. The damping terms do not appear to have any effect and can actually be neglected from the model when using conservative controller parameters.

Before implementing the controller experimentally, simulations were performed to examine the effect of feedback quality on controller performance. Up to this point, full state-feedback was provided by the non-linear model used in the simulation. In reality only the depth and pitch signals are measurable, and both signals contain undesirable traits due to random errors and low resolution quantization. Direct use of the feedback to the controller caused the simulation to crash due to numerical errors. The use of a polynomial type filter enabled the controller to work but the performance was very poor. Finally a Luenberger observer was developed using the augmented model and a discrete linear quadratic regulator. This used both the depth and pitch signals as feedback and converged quickly to provide an accurate estimate of the state-variable vector. The controller performance when using the observer was good, however, the effect of the measurement noise was still noticeable. Realistically, measurement noise is always going to effect control systems, it is therefore up to the operator to decide whether the performance is adequate. From these simulations it was deemed that the controller performance was acceptable.

To verify that the controller works on the Delphin2 AUV the control algorithm was translated from MATLAB into Python and integrated into the vehicle's control software. All three controller parameter sets were tested and all provided stable depth and pitch control. The magnitude of the depth overshoots, for all the experimental tests, were greater than the predefined tolerance of 0.1 m, with the lowest overshoot at 0.15 m. This difference, between previous simulated results and the experimental results, was due to the vehicle's buoyancy force being significantly lower than normal due to variations in the payload. A comparison between an experimental result and a simulation with the lower buoyancy is shown in Figure 6.29, both outputs agree accurately.

This meant that as the controller attempted to slow the vehicle, the forces acting against the vehicle were low and thus caused the vehicle to overshoot. It is believed that increasing the vehicles buoyancy force would reduce the depth overshoots, however, as a consequence the power consumption of the thrusters will also increase. There is nothing within the controller design that directly penalises overshoot and therefore if a large enough step-change in depth set-point is implemented then there is a risk the vehicle will overshoot its set-point. Pitch control worked adequately well however it generally failed to accurately converge to the desired $0.0°$, again this was partly due to the vehicles low buoyancy. In conclusion, the depth and pitch controller did perform reasonably well with quick convergence to each of the set-points.

This chapter has examined and verified the suitability of the MPC algorithm for use in controlling the Delphin2 AUV whilst hovering. Both simulated and experimental results confirm that the controller can successfully manoeuvre the vehicle to each of the desired depth set-points with minimal overshoot or steady-state error. In order to reduce the risk of collision, caused by overshoot, a high level controller could be used to schedule two dive phases. For example, if the vehicle is to track an altitude above the seabed of 1.0 metres, then the vehicle could first dive and stabilise at 1.5 metres before diving to the final altitude.

Chapter 7

Flight Style Depth Control using MPC

7.1 Introduction

The majority of operational AUVs are of torpedo shape and utilise a method of depth control called flight-style. This method of operation has been described earlier in this thesis but the main points will summarised here. The flight-style method relies upon forces and moments, generated by the flow of fluid over the vehicle's hull and control surfaces, in order to manoeuvre. Therefore to generate these forces and moments, the vehicle is required to have a positive surge velocity relative to the fluid. This velocity is generally controlled by varying the amount of thrust produced by a propeller fitted to the rear of the vehicle.

The reason for choosing to operate an AUV using the flight-style method is because of its high energy efficiency. In comparison to using thrusters to control depth, flight-style depth control uses much less energy per kilometre. This assumes that the AUV is operating within surge velocities for which is was originally designed, and that the hull-form and control surfaces are suitable for flight-style operation.

In this chapter an MPC controller is developed to control the depth and surge velocity of the Delphin2 AUV operating using the flight-style method. This control problem is significantly more complex than the depth and pitch controller, developed in the Chapter 6, as one linear model would not be capable of accurately estimating the system dynamics throughout the full operational range.

First the flight-style system is described, identifying some of the strong non-linearities that may present difficulties when designing a suitable controller. To handle these non-linearities, a modification to the MPC algorithm is described. This requires a time-variant state-space model of the system, produced by linearising the non-linear model of the system, to be used within the controller design (as opposed to a time-invariant state-space model used by the depth and pitch controller).

In the second half of this chapter, the flight-style controller is tested in simulation and issues with the controller identified. Modifications to improve the controller performance are tested in simulation before finally the flight-style controller is experimentally tested using the Delphin2 AUV.

7.2 Flight-Style System

As is the case with most AUVs the Delphin2 is positively buoyant, meaning there is a constant negative force acting on the AUV along the vertical z axis. To maintain a fixed depth the AUV must generate a force equal and opposite to this buoyancy force. In Chapter 6 this was generated by the vertical through-body tunnel thrusters on the Delphin2 AUV.

A lift force, generated by flow over the hull, is used to counteract the buoyancy force when operating using the flight-style method. Note that other forces contribute to counteracting the buoyancy force, however, the lift force generated by the hull is the dominating force. Figure 7.1 illustrates some of the major forces and moments acting on the AUV when operating in flight style mode.

This hull lift force is a function of the angle of attack and the surge velocity squared, therefore to control the magnitude of this force, and thus the vehicle depth, requires that the AUV can adjust these variables using rear control surfaces and the rear propeller respectively. The flight-style controller that is developed in this chapter is tasked with controlling both the depth and surge velocity of the Delphin2 AUV. The flight-style system has three degrees of freedom: pitch, depth, and surge. These can be modelled as two second-order and one first-order systems respectively.

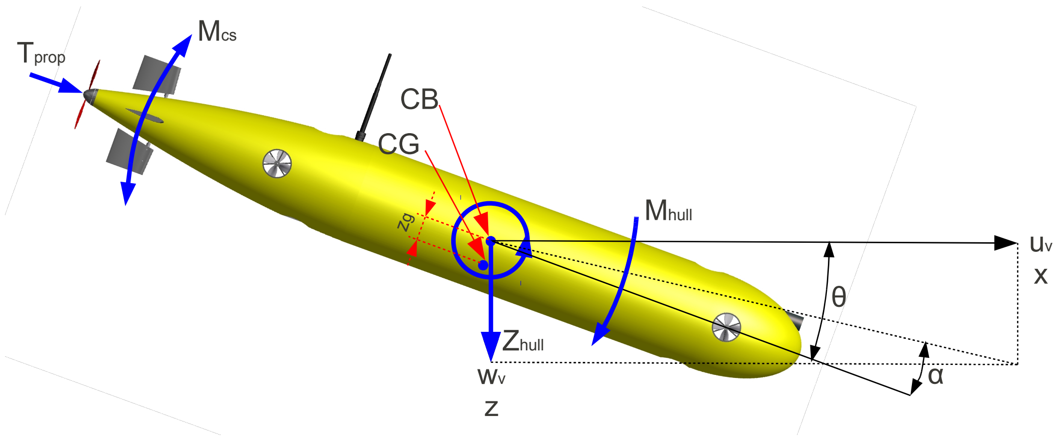


Figure 7.1: Illustration demonstrating the major forces and moments acting on the Delphin2 AUV when operating in flight style mode.

7.2.1 Surge

The acceleration along the x axis, equation (7.1), includes one mass term and three components; the thrust force from the rear propeller, and hydrodynamic damping from both the hull and control surfaces. The surge velocity is calculated by integrating the surge acceleration, equation (7.2).

The force from the propeller is used to control the surge velocity of the AUV and is a trigonometric function of the total propeller thrust and pitch angle. Both hydrodynamic damping terms are proportional to the square of surge velocity.

$$\dot{u_v} = \frac{1}{m_x}\left[\overbrace{(T_{prop}\cos\theta)}^{\text{Propeller force}} - \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_{Dh}(-\theta)|u_v|u_v\right)}^{\text{Drag force due to hull}} - \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_{Dcs}(\delta)|u_v|u_v\right)}^{\text{Drag force due to control surfaces}}\right] \quad (7.1)$$

$$u_v = \int_0^t \dot{u_v}\, dt \quad (7.2)$$

7.2.2 Depth

The heave acceleration, equation (7.3), includes one mass term and five components; the disturbance force due to the vehicle's buoyancy, the thrust force from the rear propeller, the lift forces generated by flow over both the hull and the control surfaces, and the hydrodynamic drag force. The depth is calculated by integrating the heave acceleration twice, equation (7.4). Note that in this thesis the heave velocity is the depth rate velocity in the earth axis.

The buoyancy force is a constant disturbance due to the difference in vehicle mass and displaced fluid. For the Delphin2 AUV this force is always negative, therefore accelerating the AUV towards the surface. The force acting along the z axis due to the propeller is a trigonometric function of the propeller thrust and vehicle pitch angle.

As discussed within the introduction to this section, the hull lift force is one of the most critical components within the flight-style system. This force is a function of the angle of attack of the hull, relative to the flow, and the surge velocity squared. As surge velocity tends towards zero the magnitude of the hull lift will also tend towards zero. There is therefore a critical surge velocity below which the vehicle cannot generate a sufficiently large enough lift force to overcome the buoyancy force and so the vehicle will become unstable in the z axis.

The control surfaces generate a lift force that is a function of control surface angle and surge velocity squared. This force, when coupled with a moment arm, creates the moment that is used to control the vehicle's pitch angle, see equation (7.5). Note that this force is very small in comparison with the lift force from the hull.

The final component is the hydrodynamic drag force and is a function of heave velocity squared. As the heave velocity is typically not very high when operating in flight-style operation, this force is generally low. If the AUV is diving quickly this model may over-estimate the drag force as this model does not accurately calculate the inflow angle, however, this should be sufficiently accurate for most operating conditions.

$$\begin{aligned}\dot{w_v} = \frac{1}{m_z}\Bigg[& \overbrace{(W-B)}^{\text{Buoyancy force}} + \overbrace{(T_{prop}\sin\theta)}^{\text{Propeller force}} + \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_{Lh}(\alpha)|u_v|u_v\right)}^{\text{Hull lift}} \\ & - \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_{Lcs}(\delta)|u_v|u_v\right)}^{\text{Control surface lift}} - \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_{Dw}|w_v|w_v\right)}^{\text{Damping force}}\Bigg]\end{aligned} \tag{7.3}$$

$$w_v = \int_0^t \dot{w_v}\ dt,\ z = \int_0^t w_v\ dt \tag{7.4}$$

7.2.3 Pitch

The pitch acceleration, equation (7.5), is calculated using an inertia term and four components; the moments generated by flow over both the hull and control surfaces, the restoring moment and a damping moment. The pitch of the AUV is calculated by integrating the pitch acceleration twice, equation (7.6).

The hull moment is destabilising and is a function of the angle of attack and surge velocity squared. The only system input, contributing to pitch acceleration, is the moment generated by the control surfaces and is a function of the control surface angle and the surge velocity squared. Note that as surge velocity tends towards zero then both the hull and control surface moments also tend towards zero. At a surge velocity of zero there is no effective system input, thus pitch is not controllable.

The restoring moment is a function of the pitch angle, vehicle weight and the distance along the z axis between the centre of gravity and centre of buoyancy. One point worth noting is that the restoring moment is not a function of surge velocity like the hull and control surface moments. When surge velocity is low, the moment generated by the control surfaces may be insufficient to overcome the restoring moment leading to vehicle instability.

The final component is due to hydrodynamic damping and is a function of pitch rate squared. As the pitch rate of the vehicle is typically low, the damping moment is generally small in comparison to the other components in equation (7.5).

$$\begin{aligned}\dot{q_v} \quad = \quad -\frac{1}{I_y}\Bigg[& \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_{Mh}(\alpha)|u_v|u_v\right)}^{\text{Hull moment}} + \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_{Mcs}(\delta_{cs})|u_v|u_v\right)}^{\text{Control surface moment}} \\ & + \overbrace{(z_g W \sin\theta)}^{\text{Restoring moment}} + \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_{Dq}|q_v|q_v\right)}^{\text{Damping moment}}\Bigg] \end{aligned} \tag{7.5}$$

$$q_v = \int_0^t \dot{q_v}\ dt,\ \theta = \int_0^t q_v\ dt \tag{7.6}$$

7.2.4 Discussion

Examining the three models shows that all of the equations have a strong dependence on surge velocity squared. This non-linear term is fundamental to flight style operation. Care must be taken when deriving a linearised model of the AUV as, for example, a model that is produced by linearising at a surge velocity of 0.5 m/s will be substantially different from another linearised at 1.0 m/s.

It has been mentioned when describing the surge, depth and pitch models that there is a critical speed below which the vehicle can become unstable. This is known as the Chinese effect [16] and occurs when the forces and moments, generated by the flow over the hull or control surfaces, are too low to overcome any disturbance, or restoring, forces and moments. The Chinese effect is most notable on AUVs with a positive buoyancy but it can also occur on vehicles that are neutrally buoyant.

A non-linear effect that has not been modelled here is stall. This generally occurs at low surge velocities, or at large angles of attack, and will result in the magnitude of the lift forces decreasing while the drag forces rapidly increase. The occurrence of stall increases the critical surge velocity at which instability, due to the Chinese effect, occurs.

7.3 Controller Design

In this section the design of a surge velocity and depth controller will be derived using the MPC method. The inputs to this controller are the thrust from the rear propeller, T_{prop}, and the control surface angle, δ.

Compared to the depth and pitch system, that was controlled in Chapter 6, the flight-style control problem is more complex due to a strong dependence on non-linear terms, in particular surge velocity squared. Here a method to handle these non-linearities, whilst still utilising the linear MPC algorithm, is proposed.

7.3.1 Time-Variant Model Predictive Control

One linearised state-space model was used within the depth and pitch controller developed in Chapter 6. This provided very good performance as the non-linear terms within the real system were generally stabilising (damping). The flight-style system has several non-linear terms that are destabilising which if linearised inaccurately, could cause system instability.

Fundamental to the MPC controller design is a linear state-space model of the plant that is to be controlled. If a linear state-space model was created by linearising the non-linear model at a surge velocity of 0.5 m/s but the system was operating at 1.0 m/s then the error between predicted and actual forces and moments could be up to 300%. Despite the ability of this MPC algorithm to handle model inaccuracies, the magnitude of these errors are expected to be too great to ensure good system performance.

In this chapter a time-variant MPC (TV-MPC) algorithm is developed and implemented. This utilises one linear model for all operational conditions. However, the linearised terms within the model are updated at the beginning of each sample. These terms, or coefficients, are linearised about the latest state-variable vector using the non-linear model. Therefore the dynamics of the linear model, used within the controller, should accurately approximate the real system at that instant. This type of linearisation is used by other control algorithms such as H^{∞}, [98]. It has been shown the literature, [98] and [99], that this method of linearisation can cause 'hidden couplings' that can cause control performance degradation. However, this effect is typically low for most systems and hence will not be directly addressed here.

One potential concern when operating with a TV-MPC controller is that the errors, between the future prediction horizon and the actual future trajectory of the vehicle, could increase significantly as the length of the prediction horizon increases. If the system states stay relatively constant, for example transiting at a fixed depth and surge velocity, then this should not be a problem, but if the system is accelerating then the linear model used at one instant may not accurately approximate the system dynamics at the end of the prediction horizon.

Figure 7.2 presents a flow-diagram for the TV-MPC algorithm. The basic structure is the same as with the standard MPC algorithm used in Chapter 6 with the exception that the procedures that occurred during initialisation now occur at the beginning of each new sample. These procedures include linearising the non-linear model, using the latest system feedback, and then rebuilding the augmented model. It should be noted that this does increase the computational load of the controller, but in comparison to calculating the optimal ΔU vector, the additional computational time is low.

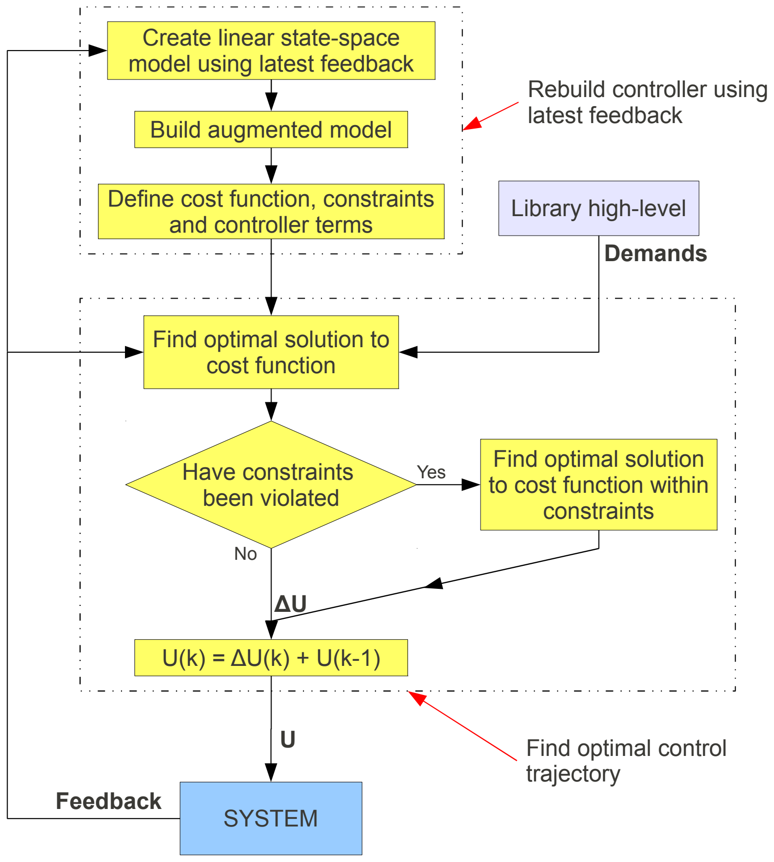

Create linear state-space model using latest feedback
Build augmented model
Define cost function, constraints and controller terms
Rebuild controller using latest feedback
Library high-level
Demands
Find optimal solution to cost function
Have constraints been violated
Yes
Find optimal solution to cost function within constraints
No
ΔU
U(k) = ΔU(k) + U(k-1)
U
Find optimal control trajectory
Feedback
SYSTEM

Figure 7.2: Flow diagram illustrating the TV-MPC algorithm. The difference between the normal MPC and TV-MPC algorithms is that the latter builds a new linearised model at the beginning of each control loop (in comparison to using just one fixed linear model).

7.3.2 Tuning

There are three controller parameters than can be used to tune the performance of the MPC controller; the lengths of the prediction horizon and control horizons, N_p and N_c, and the weighting scalars, r_w.

The prediction horizon is used to estimate the future trajectory of the system for a given change of future system inputs, ΔU. A fuller explanation of the control and prediction horizons can be found in Section 6.4.

In Chapter 6 both the system inputs, the front and rear vertical thrusters, are effectively identical. For this reason one scaler value, r_w, was used to penalise high values within the ΔU vector for both inputs. As this controller has two very different inputs, the propeller thrust and the control surface angle, it is desirable to have different weighting scalars for each input. For example to reduce the electrical power consumption of the propeller motor it would be beneficial to reduce the accelerations of this motor by using a high r_w value, while rapid changes in control surface angle could be deemed acceptable so a small r_w is used.

Recalling equation (3.21) the r_w scalar was used with an identity matrix to form the weighting matrix, $\bar{R}$, with dimensions of $mN_c \times mN_c$, were m is the number of inputs and N_c is the length of the control horizon:

$$\bar{R} = r_w I_{(mN_c \times mN_c)},$$

In order to build a new weighting matrix $\bar{R}$, with different r_w values for each input, it is defined as a block matrix with (block) dimensions of $N_c \times N_c$:

$$\bar{R}_{ij} = \begin{cases} 0_{m\times m} & \text{for } i \neq j \\ \begin{bmatrix} r_{w(1)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & r_{w(m)} \end{bmatrix} & \text{for } i = j \end{cases} \tag{7.7}$$

where $r_{w(1)}$ and $r_{w(m)}$ are the weighting scalars for the first and last inputs respectively.

Initial values for the tuning parameters of the flight-style controller developed in this chapter can be found in Table 7.1.

Table 7.1: Tuning parameters for flight-style controller

Parameter	**Value**
N_p	150
N_c	10
$r_{w(prop)}$	5.0
$r_{w(cs)}$	0.1

7.3.3 Constraints

The minimum and maximum absolute constraints for the propeller thrust are set at 1.0 N and 16.0 N respectively. The maximum is simply the maximum thrust that the propeller and motor can produce, while the minimum is the lowest value of thrust that can be produced before the propeller motor stalls. The rate of change of propeller thrust is initially set at ± 1.0 N per sample but this may be adjusted later in the chapter for tuning purposes.

The absolute constraints on control surface angles are set at their physical limitations of $\pm 30°$. The rate of change of control surface angle is approximated using the maximum speed of the linear actuators within the tail unit (see Chapter 4), resulting in constraints of $\pm 2°$ per sample. All of the constraints for the flight-style controller can be found in Table 7.2.

Table 7.2: Input constraints for flight-style controller

Parameter	**Value**
T_{prop} max	16.0 N
T_{prop} min	1.0 N
ΔT_{prop} max	1.0 N/sample
ΔT_{prop} min	-1.0 N/sample
δ max	30.0°
δ min	$-30.0°$
$\Delta\delta$ max	2.0°/sample
$\Delta\delta$ min	$-2.0°$/sample

7.4 Time-Variant State-Space Model

In this section a linear state-space model is derived that approximates the non-linear dynamics of the surge velocity, depth and pitch of the Delphin2 AUV, see Section 7.2. The input, output and state-variable vectors for the state-space model are defined then linear approximations, that are common throughout the system, are addressed and then each subsystem (surge velocity, depth and pitch) is linearised. Finally each of the subsystems are combined within one state-space model that can be used within the MPC controller design.

7.4.1 Input, Output and State-Variable Vectors

Before designing the linearised model, the input, output and state-variable vectors must be defined.

The input vector is comprised of whatever system inputs are available. Here this includes the thrust from the rear propeller and the angle of the rear horizontal control surfaces:

$$u(t) = \begin{bmatrix} T_{prop} \\ \delta \end{bmatrix} \tag{7.8}$$

The output vector is the two states that the controller is required to control; surge velocity and depth:

$$y(t) = \begin{bmatrix} u_v \\ z \end{bmatrix} \tag{7.9}$$

And the state-variable vector is defined using physically meaningful states that can either be measured or easily observed. For this model the state variable vector comprises of the surge velocity; depth; heave velocity; pitch; and pitch rate:

$$x_m(t) = \begin{bmatrix} u_v \\ z \\ w_v \\ \theta \\ q_v \end{bmatrix} \tag{7.10}$$

7.4.2 Linearisation Methods

In order for the TV-MPC algorithm to calculate the linearised coefficients, at the beginning of each new sample, requires standardised linearisation methods. The non-linearities that need to be linearised are trigonometric and quadratic functions.

Using small angle approximation the trigonometric functions can be approximated using:

$$\cos(\theta) = 1 \tag{7.11}$$

$$\sin(\theta) = \theta \tag{7.12}$$

As the heave velocity is generally very low, in comparison to surge, the angle of attack α for the hull can be approximated as:

$$\alpha = -\theta \tag{7.13}$$

Note this is accurate for steady-state conditions and will produce an error when changing depth however this error should be low.

In Chapter 6 the quadratic damping terms were linearised by minimising the square of the normalised errors. In this chapter the quadratic terms are directly linearised using the latest feedback. For example, a quadratic damping term such as:

$$X_D = \frac{1}{2}\rho V^{2/3} C_D |u_v| u_v \tag{7.14}$$

is linearised using the latest surge velocity at sample k:

$$X_D = \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_D |u_v(k)|\right)}^{\text{Linear coefficient}} \times u_v, \tag{7.15}$$

Figure 7.3 presents the damping forces against surge velocity for the non-linear and linearised terms (linearised at 0.5 m/s).

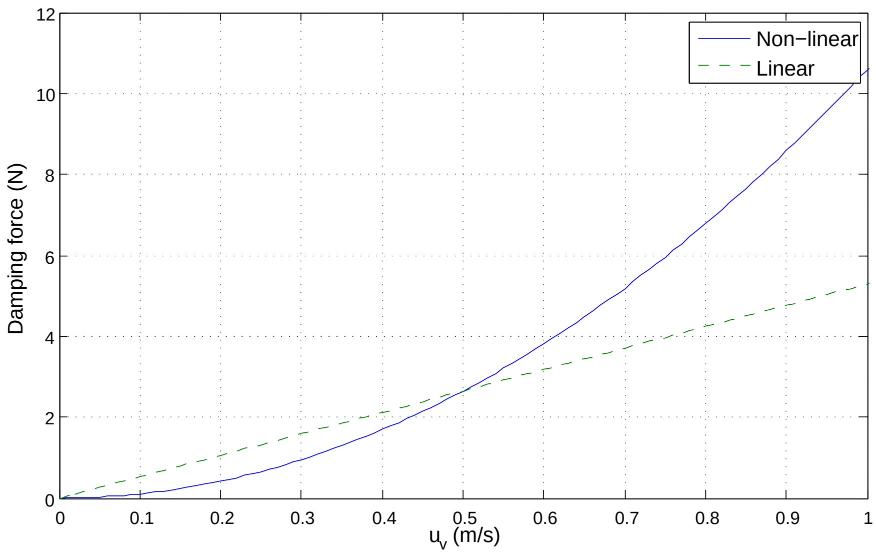


Figure 7.3: Comparison of drag forces calculated using the non-linear function and the linearised function (linearised at 0.5 m/s).

For non-linear terms that are a function of more than one state variable or input, an active variable must be defined. This is the variable that is deemed to have the most importance with regards to how the controller functions. For example, when calculating the moment generated by the control surfaces then their angle is taken to be the active variable while the other variables, such as surge velocity, are fixed using the latest feedback values. Therefore the controller 'realises' that it needs to adjust the angle of the control surfaces to adjust the moment generated by them.

7.4.3 Surge

Examining equation (7.1), one mass term, one input and two dynamic components can be identified. The mass term is specific to the x axis and includes added mass. The input is the thrust force produced by the rear propeller and is linearised by removing the dependence on pitch angle:

$$T_{prop} \cos\theta = T_{prop} \tag{7.16}$$

Both of the dynamic components are quadratic damping forces due to the hydrodynamic drag from the hull and control surfaces. Both of these forces are proportional to the square of surge velocity, whilst the drag coefficients for the hull and control surfaces are functions of α and δ respectively.

These terms will be approximated within the state-space models using surge velocity as the active variable. Linear drag terms are created using the latest surge velocity and the drag coefficients are calculated using the latest pitch and control surface angles. The hull damping term is therefore defined as:

$$k_{Dh} = -\frac{1}{2}\rho V^{2/3} C_{Dh}(-\theta(k))|u_v(k)| \tag{7.17}$$

and for the control surfaces as:

$$k_{Dcs} = -\frac{1}{2}\rho V^{2/3} C_{Dh}(\delta(k))|u_v(k)| \tag{7.18}$$

Thus the forces due to drag along the x axis can be calculated as:

$$X_{hull} = (k_{Dh}) \times u_v \tag{7.19}$$

$$X_{cs} = (k_{Dcs}) \times u_v \tag{7.20}$$

It could be desirable to have δ or θ as the active variables however this would cause an error if these angles change sign in the future prediction, resulting in a positive force for a drag term.

The dynamics and input matrices for the state-space model with the surge components therefore take the form:

$$\dot{x_m}(t) = \begin{bmatrix} \frac{(k_{Dh}+k_{Dcs})}{m_x} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_v \\ z \\ w_v \\ \theta \\ q_v \end{bmatrix}(t) + \begin{bmatrix} \frac{1}{m_x} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} T_{prop} \\ \delta \end{bmatrix}(t) \tag{7.21}$$

7.4.4 Depth

Examining equation (7.3), one mass, one disturbance, two inputs and two dynamic components can be identified. The mass term is specific to the z axis and includes added mass. As with the depth and pitch controller, developed in Chapter 6, the disturbance force caused by buoyancy will be omitted from the linearised model as the integrator within the controller is expected to compensate for this.

The first input is the force produced by the rear propeller acting along the z axis. This is a function of both the thrust force from the propeller and the pitch angle of the AUV. The linearised term uses the latest propeller force and θ is set as the active variable. Using small angle approximation, the sine function can be omitted and the force along the z axis can then be calculated using:

$$Z_{prop} = (-T_{prop}(k)) \times \theta \tag{7.22}$$

The second input is due to the lift generated by the rear control surfaces. This is linearised by creating a new coefficient that is updated on each sample using the latest surge velocities and the average gradient of the control surface lift coefficient against control surface angle $\dot{C}_{Lcs}$. This control surface lift term is defined as:

$$k_{Lcs} = \frac{1}{2}\rho L^2 \dot{C}_{Lcs} |u_v(k)| u_v(k), \tag{7.23}$$

and so the lift force generated by the control surfaces is calculated using:

$$Z_{cs} = (k_{Lcs}) \times \delta \tag{7.24}$$

The dynamic components are the lift force generated by flow over the hull and the hydrodynamic damping force. The hull lift is one of the most critical components within the linear model, without this information it is unlikely that the MPC controller would be able to successfully control depth. As there are two state variables within the hull lift equation, α (approximated here to equal $-\theta$) and u_v, an active variable must be chosen.

As the controller must 'know' to adjust the pitch angle of the vehicle, so as to alter its lift coefficient, the hull lift equation will use the pitch angle as its active variable. The equation is linearised by creating a new term using the latest surge velocity and the average gradient of the hull lift coefficient against angle of attack, $\dot{C}_{Lh}$. The hull lift term is defined as:

$$k_{Lh} = \frac{1}{2}\rho L^2 \dot{C}_{Lh} |u_v(k)| u_v(k) \tag{7.25}$$

and so the hull lift force is calculated using:

$$Z_{hull} = (k_{Lh}) \times \theta \tag{7.26}$$

The final dynamic component is the damping term due to the hydrodynamic drag. A linear coefficient is created using the drag coefficient and latest heave velocity:

$$k_{Dz} = -\frac{1}{2}\rho V^{2/3} C_{Dz} |w_v(k)| \tag{7.27}$$

thus the damping force along the z axis can be calculated as:

$$X_{hull} = (k_{Dz}) \times w_v \tag{7.28}$$

The dynamics and input matrices for the state-space model with the depth components therefore takes the form:

$$\dot{x}(t) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{(k_{Dz})}{m_z} & \frac{(T_{prop}(k)+k_{Lh})}{m_z} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_v \\ z \\ w_v \\ \theta \\ q_v \end{bmatrix}(t) + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & (\frac{k_{Lcs}}{m_z}) \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} T_{prop} \\ \delta \end{bmatrix}(t) \tag{7.29}$$

7.4.5 Pitch

Examining equation (7.5), one inertia, one input and three dynamic components can be identified. The inertia term is specific to rotation about the y axis and includes added inertia.

The input is the moment generated by the control surfaces. As the controller must 'realise' to adjust the angle δ, so as to change the moment generated by the control surfaces, it is chosen as the active variable. A linearised term is created using the latest surge velocity and the average gradient of the moment coefficient against control surface angle, $\dot{C}_{Mcs}$, and is defined as:

$$k_{Mcs} = \frac{1}{2}\rho L^3 \dot{C}_{Mcs} |u_v(k)| u_v(k), \tag{7.30}$$

and so the moment generated by the control surfaces can be calculated using:

$$M_{cs} = k_{Mcs} \times \delta \tag{7.31}$$

The first dynamic component to be addressed is the hull moment. This is a destabilising moment so it is important that the controller is aware of this. The active variable in this function is set to be θ. A linear term is created using the latest surge velocity and the average gradient of the moment coefficient against hull angle of attack, $\dot{C}_{Mh}$, and is defined as:

$$k_{Mh} = \frac{1}{2}\rho L^3 \dot{C}_{Mh} |u_v(k)| u_v(k), \tag{7.32}$$

and so the moment generated by the hull can be calculated using:

$$M_h = k_{Mh} \times \theta. \tag{7.33}$$

The restoring moment is linearised using small angle approximation as:

$$z_g W \sin\theta = z_g W \times \theta. \tag{7.34}$$

The last dynamic component is the damping term that is a function of pitch rate squared. In order to create a linearised term, the pitch rate is used both as the active component and the fixed term such that the linearised term is defined as:

$$k_{Dq} = -\frac{1}{2}\rho L^3 C_{Dq} |q_v(k)|, \tag{7.35}$$

and so the moment generated by hydrodynamic damping is calculated using:

$$M_D = k_{Dq} \times q_v \tag{7.36}$$

Combining the linear coefficients for the pitch model into the dynamics and input matrices results in:

$$\dot{x}(t) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{(k_{Mh}+z_g W)}{I_q} & \frac{k_{Dq}}{I_q} \end{bmatrix} \begin{bmatrix} u_v \\ z \\ w_v \\ \theta \\ q_v \end{bmatrix}(t) + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & (\frac{k_{Mcs}}{I_q}) \end{bmatrix} \begin{bmatrix} T_{prop} \\ \delta \end{bmatrix}(t) \tag{7.37}$$

7.4.6 Flight-Style State-Space Model

Combining the three state-space models created in the last three subsections, equations (7.21), (7.29) and (7.37), gives the flight-style model that will be used to build the MPC controller:

$$\dot{x}(t) = \overbrace{\begin{bmatrix} \frac{(k_{Dh}+k_{Dcs})}{m_x} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{(-k_{Dz})}{m_z} & \frac{(T_{prop}(k)+k_{Lh})}{m_z} & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{(k_{Mh}+z_gW)}{I_q} & \frac{k_{Dq}}{I_q} \end{bmatrix}}^{A_m} \begin{bmatrix} u_v \\ z \\ w_v \\ \theta \\ q_v \end{bmatrix}(t) + \overbrace{\begin{bmatrix} \frac{1}{m_x} & 0 \\ 0 & 0 \\ 0 & \frac{k_{Lcs}}{m_z} \\ 0 & 0 \\ 0 & \frac{k_{Mcs}}{I_q} \end{bmatrix}}^{B_m} \begin{bmatrix} T_{prop} \\ \delta \end{bmatrix}(t) \tag{7.38}$$

$$y(t) = \overbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}}^{C_m} x_m(t) \tag{7.39}$$

7.5 Observer Design

As with the depth and pitch controller developed in the previous chapter, an observer will be used to provide the state-variable vector that is used within the MPC algorithm as feedback. This observer is tasked with providing an accurate estimate of the state-variable vector using measured values of the system outputs with normal levels of measurement noise.

First the observability of the augmented model, used by the flight-style controller, is checked:

$$\text{Number of unobservable states} = dim(A) - rank \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} = 0 \tag{7.40}$$

As all the states are observable then the observer can be built, see Section 3.3.4. To calculate the observer feedback matrix, L, the discrete linear-quadratic state-feedback regulator (dlqr) MATLAB function is used. As the linear model is time-variant the L matrix must be calculated after each new model is built.

Using this observer the controller was tested in simulation. Despite various values of r_o, used for tuning the convergence rate of the observer, the overall system performance was very poor. This is because only surge velocity and depth measurements are provided as feedback to the observer causing the pitch and pitch rate to diverge from their real values. Figure 7.4 plots surge velocity, depth, heave velocity, pitch and pitch rate from both the non-linear model and the observer. The surge velocity, depth and heave velocity track the values from the non-linear model quite accurately however it is clear that the pitch and pitch rate are diverging from reality. These large errors cause the controller to completely fail, with no obvious convergence to any set-point.

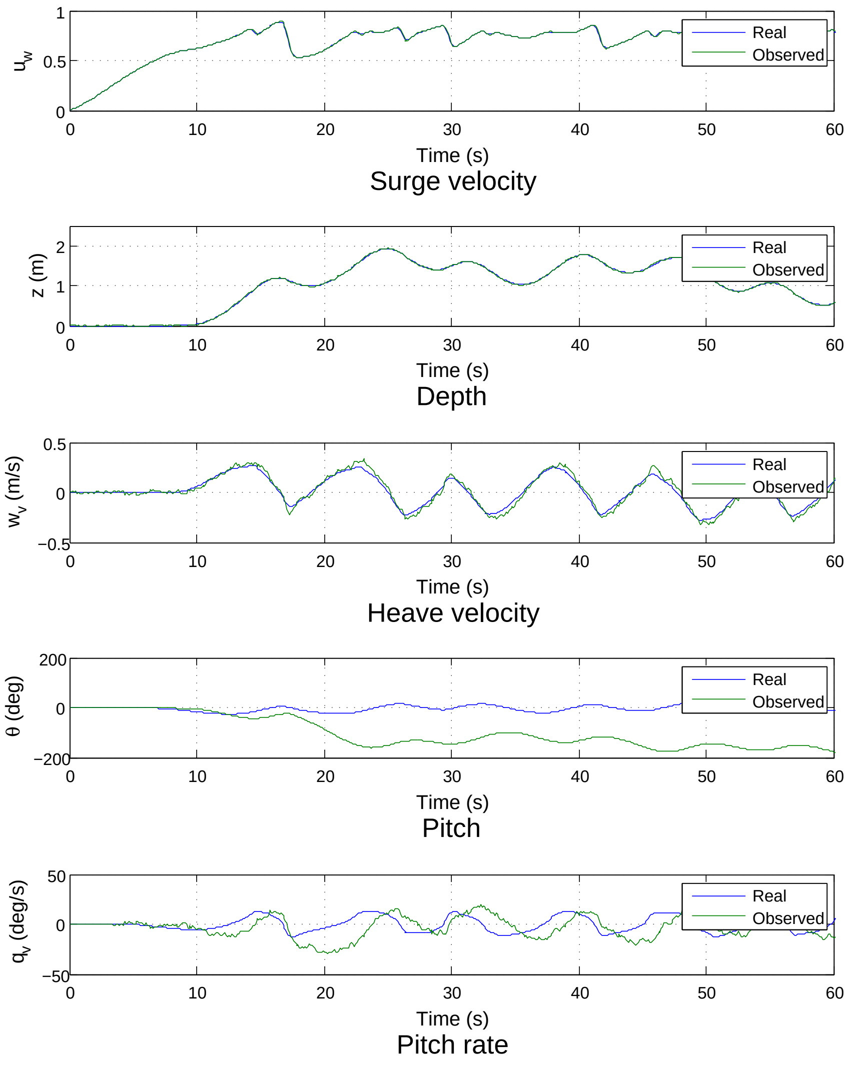


Figure 7.4: Comparison between observed and real state signals. Note that the observer has feedback of depth, resulting in the observed depth and surge velocity signals tracking the real signals accurately. However, as there is no feedback of the pitch state, both pitch and pitch rate diverge quickly from those of the real system causing the controller to fail.

To close the loop, and prevent pitch from diverging away from reality, a second augmented model of the system is built, identical to the one used by the controller but with the addition of pitch in the output vector. Figure 7.5 presents the same five states as in Figure 7.4 but using an observer with feedback of all three outputs. As can be seen the addition of pitch feedback has enabled the observed pitch to converge to the correct value. As would be expected all three states that are also outputs track the real value very closely. The two derivatives, the heave velocity and pitch rate, track the real values quite accurately though the measurement noise still has a small effect.

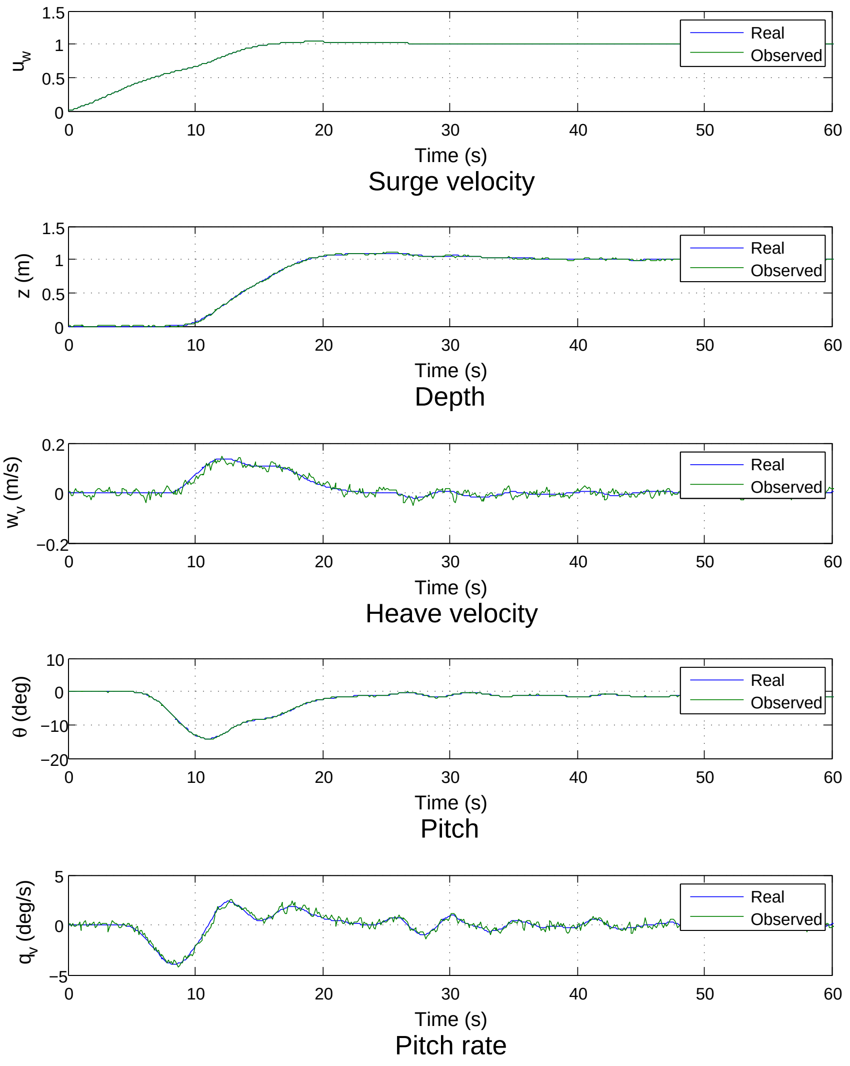


Figure 7.5: Comparison between observed and real state signals. In this observer, feedback of all three measured values is fed back into the observer resulting in the observer accurately tracking all five states.

The performance of the observer is tuned by varying the weighting matrix R. In Chapter 6 the observer, used by the depth and pitch controller, was tuned by an identity matrix and a scalar. For this controller, an effort is be made to tune the convergence rate of each individual state by using a diagonal matrix of different weighting scalars for each output:

$$R = \begin{bmatrix} r_o(u_v) & 0 & 0 \\ 0 & r_o(z) & 0 \\ 0 & 0 & r_o(\theta) \end{bmatrix} \tag{7.41}$$

where $r_o(u_v)$, $r_o(z)$ and $r_o(\theta)$ are the weighting scalars for the surge, depth and pitch signals.

Given that the feedback signal for surge velocity is coming from an open-loop non-linear model of the system within the dead-reckoner, there should be no measurement noise. Because of this, and that there are no derivative states (as with depth and pitch), the weighting scalar for the surge velocity observer can be near zero.

The scalars used for depth and pitch were found by trial and error. A visual comparison between the observer estimates and the real values from the non-linear model, along with controller performance, was used to decide on the weighting scalars. The scalars used for the remainder of this chapter are given in Table 7.3.

Table 7.3: Weighting scalars used to tune flight-style observer

Parameter	**Value**
$r_o(u_v)$	1.0×10^{-6}
$r_o(z)$	200.0
$r_o(\theta)$	10.0

7.6 Controller Performance in Simulation

In this section the non-linear model of the Delphin2 AUV is used to simulate the flight-style controller within the MATLAB environment. The key attributes of the controller performance are the ability to achieve the depth and surge velocity set-points within an acceptable length of time without significant overshoot. These simulations are performed in order to thoroughly test the basic performance and stability of the flight-style controller, and to identify any possible weaknesses associated with using the TV-MPC algorithm.

First the basic performance of the controller is analysed and discussed. Then a method of modifying the state-space model to improve controller performance is examined before finally identifying possible issues with the TV-MPC algorithm when faced with rapid accelerations.

7.6.1 Basic Performance

Figure 7.6 presents the system response to a step change of depth and surge velocity set-points. The simulation starts from zero initial conditions with set-points for depth and surge velocity of 1.0 m and 1.0 m/s respectively. Measurement noise has not been added to the feedback signals in this simulation.

The top two sub-plots in Figure 7.6 are surge velocity and depth against time, while the third and forth sub-plots are of the vehicle's pitch and the system inputs against time. The surge velocity increases quickly from 0.0 m/s to the set-point of 1.0 m/s within approximately 12 seconds and with no significant overshoot.

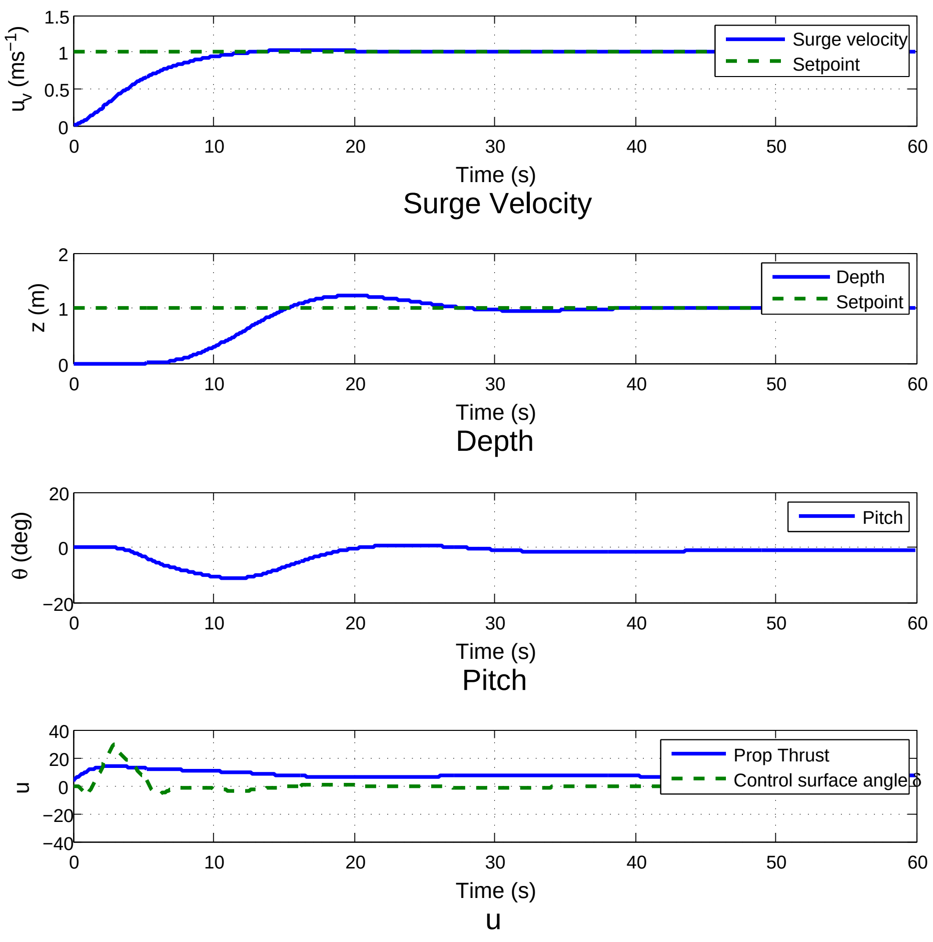


Figure 7.6: Simulation of the flight-style depth and speed controller. Note both outputs converge to the set-points however the depth signal does have an unacceptably large overshoot.

The depth response is much slower and less damped than the surge response, taking approximately 30 seconds to converge and with an undesirable overshoot of 0.22 m. The controller performs as would be expected, using the control surfaces to develop a negative pitch angle and thus dive the vehicle. As the pitch angle starts to decrease, at around 4 seconds, the control surface angle starts to wind back and eventually converges to approximately $-1.0°$ causing the vehicle's pitch to eventually converge to $-1.4°$.

The control of the vehicle's pitch angle, using the control surfaces, appears to be very conservative with little fluctuation of the control surface angle after 8 seconds. Clearly the controller predicts that the system will converge to the set-point using primarily the natural system dynamics, which it does, but it would be desirable if the depth control converged faster and had a smaller overshoot.

It would be expected that tuning the length of the prediction and control horizons, and the weighting scalers should provide improvements in the depth control performance. A wide range of these values, in particular a low weighting scalar for the control surfaces, does vary the convergence rate and overshoot but does not reduce the overshoot to within acceptable levels.

7.6.2 Controller Tuning by Model Adaptation

In this section the effect of varying the parameters used within the state-space model is investigated. The idea behind this is to manipulate the system model in such a way as to force the MPC controller to perform better.

For the depth and pitch controller, developed in Chapter 6, the damping terms for heave velocity and pitch rate were linearised using one fixed value. In the flight-style controller these values are linearised using the latest heave and pitch rate values. This should provide a better approximation to the real system dynamics at that instant, compared to one fixed term, but may not approximate the system dynamics as accurately throughout the future prediction. Comparing simulation results between the TV-MPC terms and those using the fixed damping terms, Table 6.2, the system performance is almost identical, suggesting that both models are similar.

To investigate why the depth control is so conservative the eigenvalues of the depth and pitch subsystems are examined. For the fixed damping terms, the poles of the depth and pitch subsystems are -0.0478 and -0.0173 respectively. Both of these values are very close to the real axis, suggesting the system is almost marginally unstable. Indeed, when using the TV-MPC method, and when both the heave velocity and pitch rates are zero, then their poles lie on the real axis. Having poles so close to the real axis is most likely the cause of the conservative input values from the controller.

To move these poles further to the left and force the MPC controller to act in a more aggressive manner, similar to the effect of high gain within classical control, the heave and pitch rate damping terms are multiplied by scalar gains. Using the fixed damping terms from Chapter 6, k_z and k_q, two gains are applied, G_z and G_q, to the heave and pitch rate damping respectively. Figure 7.7 presents the simulation results, for the same step-changes in depth and surge velocity as in Figure 7.6, but with damping gain values both equal to 40.0. Clearly the depth control has been improved dramatically and converges within approximately 16 seconds. The surge velocity control degrades a little with a small amount of overshoot. This overshoot has been caused by the rapid changes in pitch angle and thus drag of the vehicle such that when the vehicle levels out it quickly accelerates. This overshoot in surge velocity is not critical and is well within acceptable limits. The gain values were found using trial and error.

By artificially increasing the damping terms within the linear model, the poles have been moved far to the left thus imitating a more stable system. It is this increased stability that ‘gives’ the MPC algorithm warrant to execute larger control inputs causing the controller to behave in an overall more aggressive, high gain, manner.

This method of introducing inner gains could be deemed dangerous, with the obvious possibility of causing system instability if the gains are too high. Indeed by increasing both of the gains to 200.0 the depth control becomes unstable. It could be argued that if an MPC controller is performing

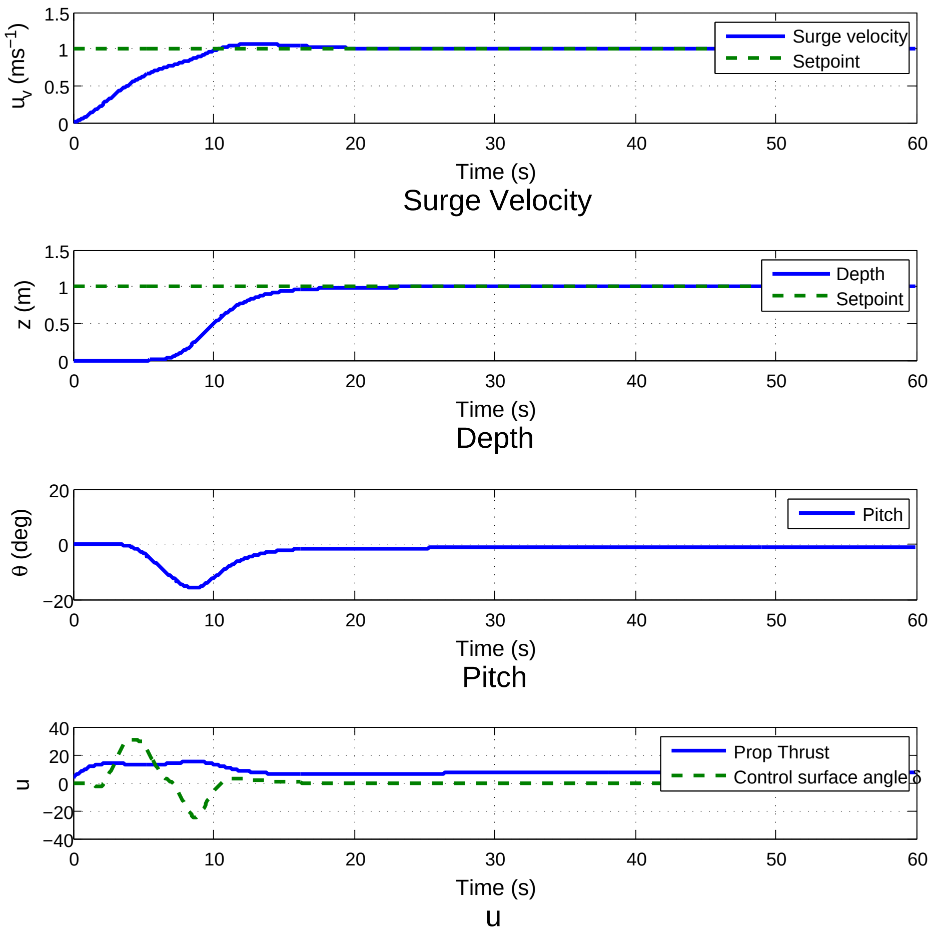


Figure 7.7: Simulation of the flight-style depth and speed controller with gains applied to the heave and pitch damping terms. Note the significantly faster convergence to the depth set-point with no overshoot.

too conservatively then the poles of the original dynamics matrix need to be investigated and then adjusted by means of gains within the model. It may also work in reverse if a system is performing too erratically, by moving the system poles to the right then the controller may act more conservatively. It is believed by the author that the model used by the controller should approximate the actual system as best as possible, only after efforts to tune the controller using normal methods should the operator start introducing inner gains.

Given the substantial benefit in controller performance, the fixed damping terms with their associated gains will be used for the rest of this chapter.

7.6.3 TV-MPC Limitations

One concern that was identified earlier in this chapter was that, due to the way the system is linearised, if any of the states are quickly accelerating then the future model prediction will be of poor quality and controller performance could degrade. As the majority of the dominant terms within the linear model are linearised using the latest surge velocity, e.g. hull lift force, then the controller will not appreciate that when it is accelerating (to a higher surge velocity) it needs to reduce the pitch angle to maintain a fixed depth.

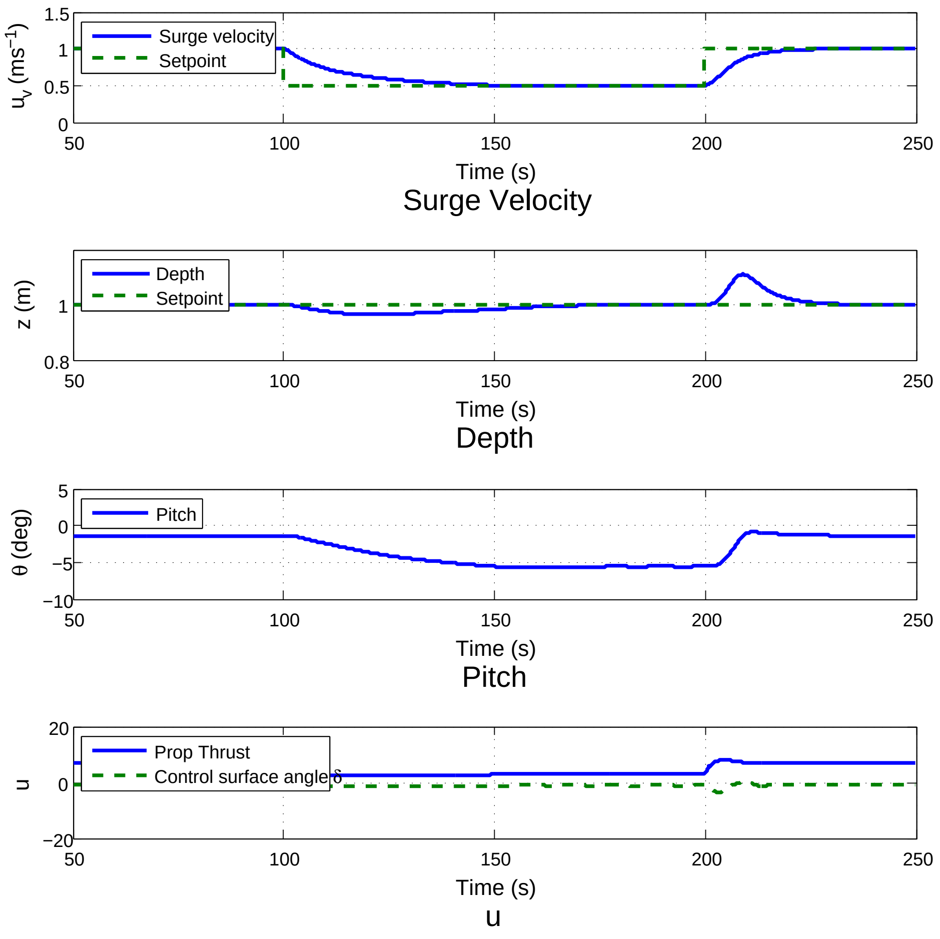


Figure 7.8: Simulation of the flight-style depth and speed controller with step changes in surge velocity set-points. Note that disturbances occur to the depth tracking when the surge velocity changes.

Figure 7.8 presents simulation results for step changes in surge velocity from 1.0 to 0.5 to 1.0 m/s. The controller does a very good job at achieving the surge velocity demands however disturbances do occur on the depth tracking. The first disturbance, due a decrease in surge velocity, is a decrease in vehicle depth before converging back to the set-point. The magnitude of this disturbance is small and, although undesirable, not of great concern. The second disturbance occurs when the surge velocity increases quickly from 0.5 to 1.0 m/s, causing a sudden overshoot. This overshoot of 0.11 m is just outside of the acceptable tolerance. It may be possible that a different linear model may help reduce or eliminate these overshoots. One issue to note is that the distance travelled to change speed from 1.0 to 0.5 m/s is quite high and so obstacle avoidance could be hindered whilst using this controller. If such an event occurred then the controller could be set into a different high gain mode to slow more abruptly.

During normal operation AUVs typically operate at a constant surge velocity, whereas adjusting its depth is quite common. Therefore it is more advantageous to keep the current linearisation scheme but reduce or constrain the accelerations along the x axis. There are two methods to reduce the surge accelerations; use larger $r_{w(prop)}$ values to penalise large changes in propeller thrust that in turn will reduce surge accelerations, and implement tighter constraints on the maximum and minimum changes in propeller thrust set-point. Both options will reduce accelerations in surge however, the latter option gives the ability to reduce positive accelerations (that cause increases in depth) while maintaining normal unconstrained negative accelerations that is desirable for obstacle avoidance. The use of constraints is more computationally expensive as it is likely that the Hildreth algorithm (used to identify the active constraints) will be required to complete more iterations for tighter constraints. Figure 7.9 presents the results of using a higher $r_{w(prop)}$ value of 250 (normally 5). As can be seen the acceleration has been reduced, as has the disturbances to within acceptable limits. A compromise between surge velocity response to depth disturbances needs to be evaluated by the AUV operator.

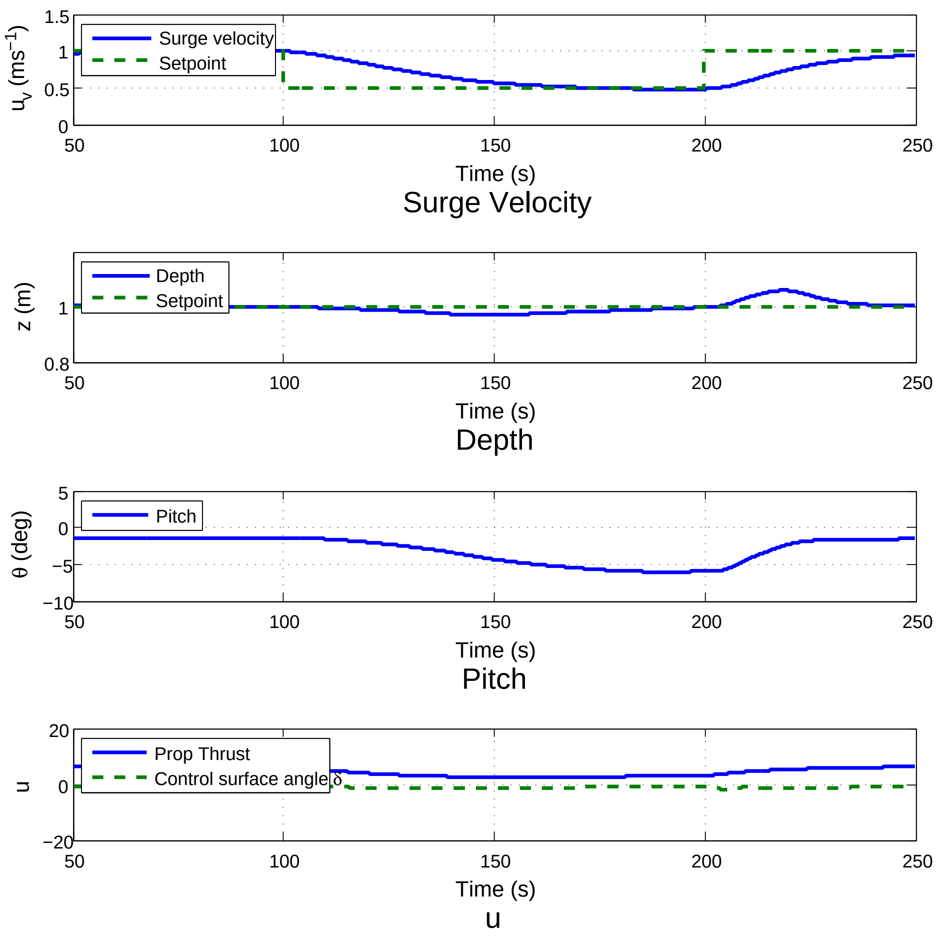


Figure 7.9: Simulation of the flight-style depth and speed controller with step changes in surge velocity set-points. A very large $r_{w(prop)}$ of 250 has been used to reduce the acceleration of the vehicle in the x axis.

7.7 Experimental Verification

To verify that the flight-style MPC controller, developed in this chapter, works on the vehicle requires that it is experimentally tested on the vehicle. The controller and observer have been translated into the Python programming language and integrated into the Delphin2 control software as a separate node. The vehicle was tested at Testwood Lake near Southampton (latitude: 50.937932, longitude: -1.507359), which measures approximately 270 m long, 125 m wide and has an average depth of 3.0 m.

7.7.1 Effect of Inner Gains

One of the significant alterations to the MPC controller that was developed during the simulation section of the chapter was the addition of inner gains within the state-space model. These gains are used to move the system poles further to the left of the real axis causing the MPC controller to behave more aggressively. In simulation, these gains drastically improved controller performance.

Figures 7.10, 7.11 and 7.12 present experimental results of the flight-style TV-MPC controller. Figure 7.10 presents results of the controller using an inner gain on the pitch damping, while Figures 7.11 and 7.12 are results of the controller without any inner gain on pitch damping and with no pitch damping at all.

All of the tests start from approximately zero initial conditions and have surge velocity and depth set-points of 1.0 m/s and 1.0 m respectively. Clearly the addition of the inner gain on pitch is highly detrimental to controller performance. The increased controller ‘aggressiveness’ is apparent from the ‘bang-bang’ type control surface angle demands, resulting in large fluctuations in pitch angle and no obvious convergence to the depth set-point. These large fluctuations in pitch angle cause the drag coefficient of the vehicle to be quite high and as a result the controller struggles to achieve the surge velocity set-point.

Figure 7.11 presents results using the normal pitch damping term without the inner gain. Here the depth control performs much better than with the gain, with convergence to the depth demand after approximately 32 seconds. There is however a large depth overshoot of 0.5 m. This overshoot behaviour is similar to the simulation results shown in Figure 7.6. Once the AUV converges to the depth set-point the vehicle continues to oscillate in pitch before stabilising at after 60 seconds however, slight oscillations do continue to occur.

Out of interest as to how the pitch damping term effects controller performance, one test was conducted with the pitch damping term set to zero, see Figure 7.12. From these results it is clear that the depth control degrades, with a very large depth overshoot of approximately 0.8 m and slow converge time of 60 seconds. Like the other results, the pitch oscillates throughout the test however it may be possible that the vehicle would stabilise to a fixed pitch angle over time.

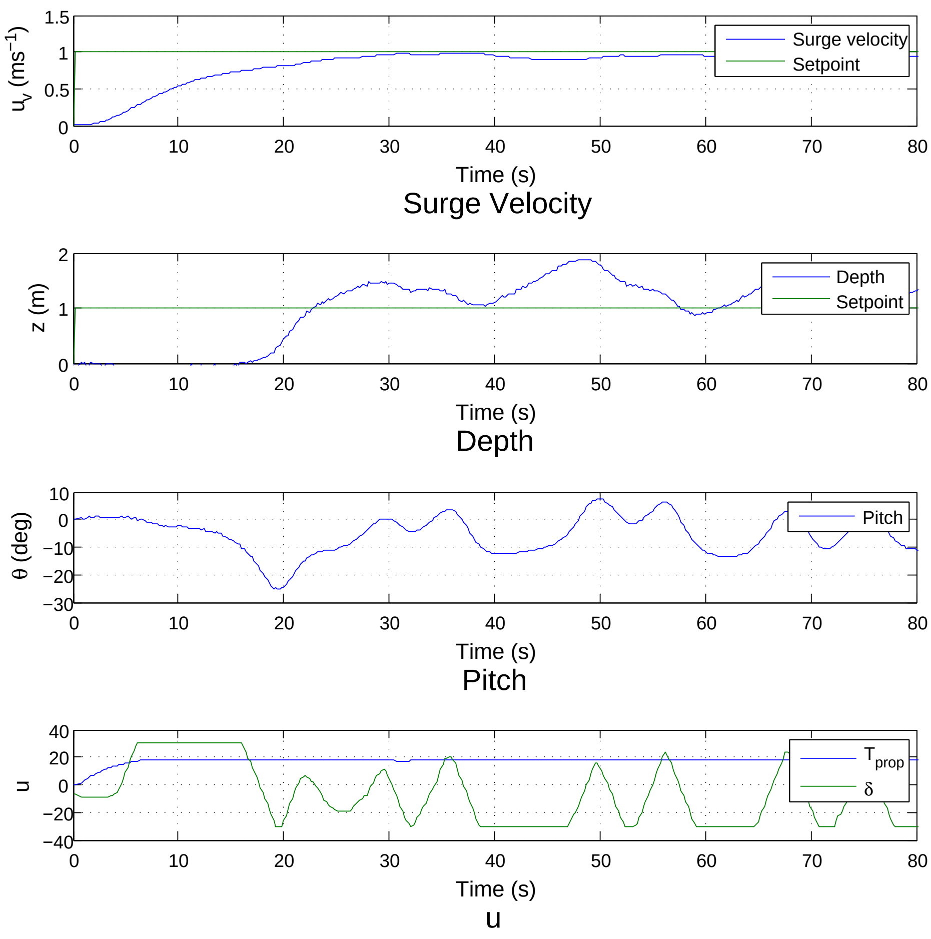


Figure 7.10: Experimental results of the TV-MPC flight-style controller with surge velocity and depth set-points of 1.0 m/s and 1.0 m respectively. The inner gain has been applied to both the depth and pitch damping terms within the state-space model.

The results shown here have demonstrated that the pitch damping term has a strong effect on flight-style depth control performance. The high inner gain on the pitch damping, that improved system performance in simulation, causes the controller to fail with no convergence to depth and large fluctuations in pitch. Examining the opposite conditions, with zero pitch damping, the controller performs very slowly with convergence to the depth set-point after a long period of time and after a large depth overshoot. With the normal pitch damping, as defined at the beginning of this chapter, depth control performance is better. With faster convergence to the depth set-point and with a smaller depth overshoot.

Therefore somewhere between zero and the high pitch damping (with the inner gain) lies an optimal pitch damping term that provides the best possible convergence time and minimal overshoot. The use of the damping coefficients from low speed hover operation may not provide an accurate approximation for flight-style operation.

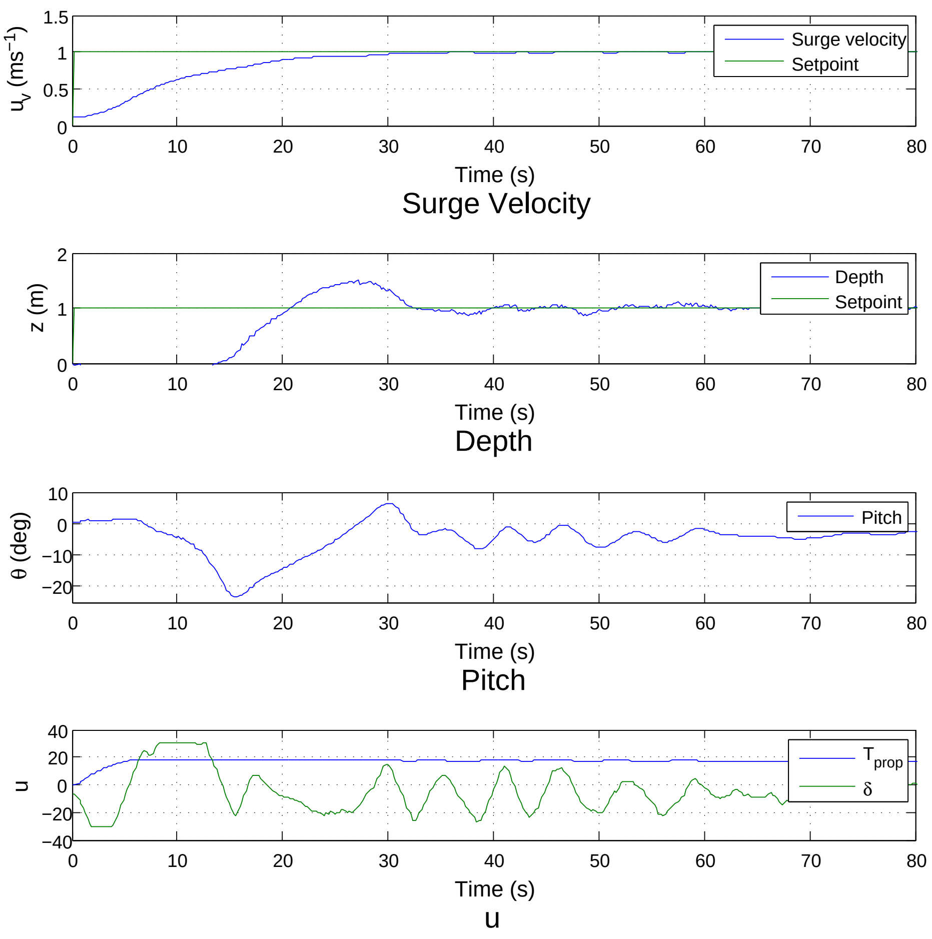


Figure 7.11: Experimental results of flight-style controller with surge velocity and depth set-points of 1.0 m/s and 1.0 m respectively. No inner gain has been applied to the pitch damping within the state-space model.

7.7.2 Depth Step-Changes

Using the normal pitch damping term, the flight-style controller was tasked with diving from the surface to 1.0 m then 2.0 m and finally back to 1.0 m, all at a fixed surge velocity of 0.75 m/s. Figure 7.13 presents the experimental results of this test.

The surge velocity quickly reaches the desired set-point of 0.75 m/s. Some small disturbances are noted on the surge velocity signal though these are small in magnitude and within acceptable limits.

The depth control convergences to the its respective set-points after approximately 40 seconds. The vehicle overshoots the depth set-points on each step-change and the vehicle experiences significant disturbances when stable at each depth. The pitch of the vehicle oscillates in an irregular manner with the exception of when the depth set-point is changed. Due to the strong coupling between depth and pitch it is difficult to deduce from Figure 7.13 what is causing these oscillations and disturbances.

Figure 7.14 presents the same data as in Figure 7.13 but focused on the period between 200 and

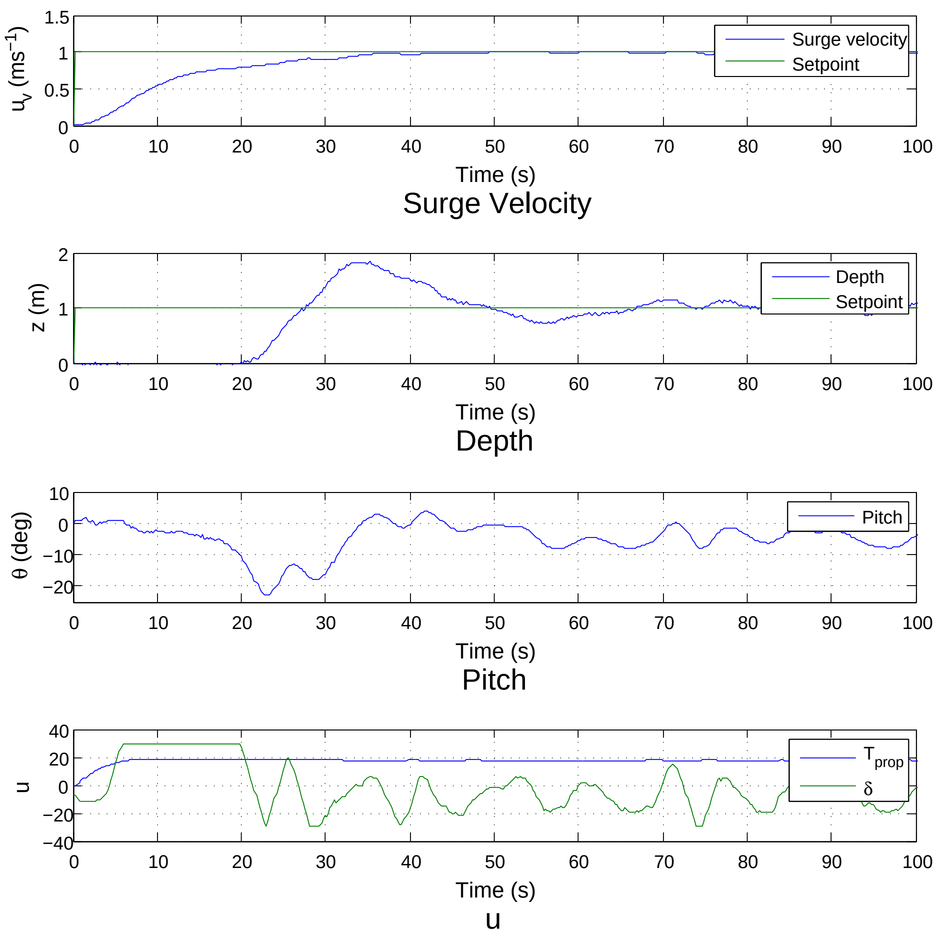


Figure 7.12: Experimental results of flight-style controller with surge velocity and depth set-points of 1.0 m/s and 1.0 m respectively. Pitch damping within the state-space model is set to zero.

250 seconds of the test. Before examining the data the basic flight style method will be reiterated. Acceleration in depth is controlled by varying the pitch angle which in turn varies the lift coefficient of the hull, and in doing so, the lift force generated by flow over the hull. Therefore, if the depth error is negative (set-point minus the current depth value) the pitch angle should increase and vice versa. Note the sign of the pitch angle.

Looking at the depth signal in Figure 7.14, there is a positive depth error between 200 and 228 seconds. To reduce this, the controller decreases the vehicle's pitch (increasing angle of attack) to generate more positive lift force from the hull. The pitch continues to decrease slowly until, at approximately 215 seconds, the depth error starts to decrease. Although the pitch angle does then start to decrease, it does not do so quickly enough and as a result the vehicle overshoots the depth set-point. The pitch of the vehicle appears to lag behind depth by approximately 2 seconds when ideally it should precede it so that the pitch angle is adjusted before the depth set-point is overshot. The prediction horizon used for this test was 120, equivalent to 12 seconds therefore should be sufficient to 'see' that the vehicle is going to overshoot and take appropriate measures to prevent this.

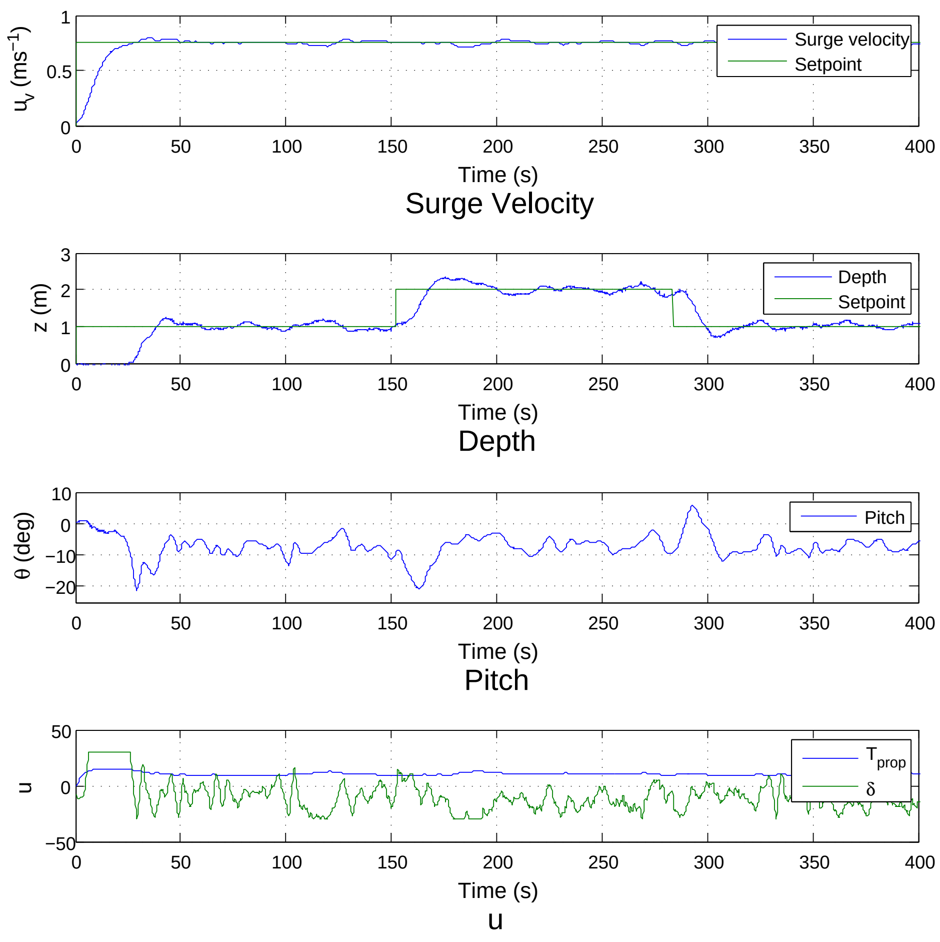


Figure 7.13: Experimental results of flight-style controller with a fixed surge velocity set-point of 0.75 m/s, and depth set-points of 1.0 m, then 2.0 m and finally back to 1.0 m.

7.8 Discussion

In this chapter a flight-style surge velocity and depth controller using the time-variant MPC (TV-MPC) algorithm has been developed. Simulation results, using the non-linear model of the vehicle as the system, showed that the controller worked but very conservatively and with large overshoots in depth demand. Inner gains were applied to the damping terms within the state-space model in order to move the poles of the depth and pitch subsystems further to the left of the real axis in the complex plane. Using this method the depth control was substantially improved, with much faster settling times and minimal overshoot. Without these additional gains on damping, the controller performance was unacceptably slow.

The experimental testing of the controller resulted in significantly different performance when compared to the simulated results. The use of the large inner gains on the pitch damping caused very large oscillations in pitch that in turn resulted in the vehicle failing to converge to the desired depth set-points. Without the gain on pitch damping, the vehicle does converge to the desired depth

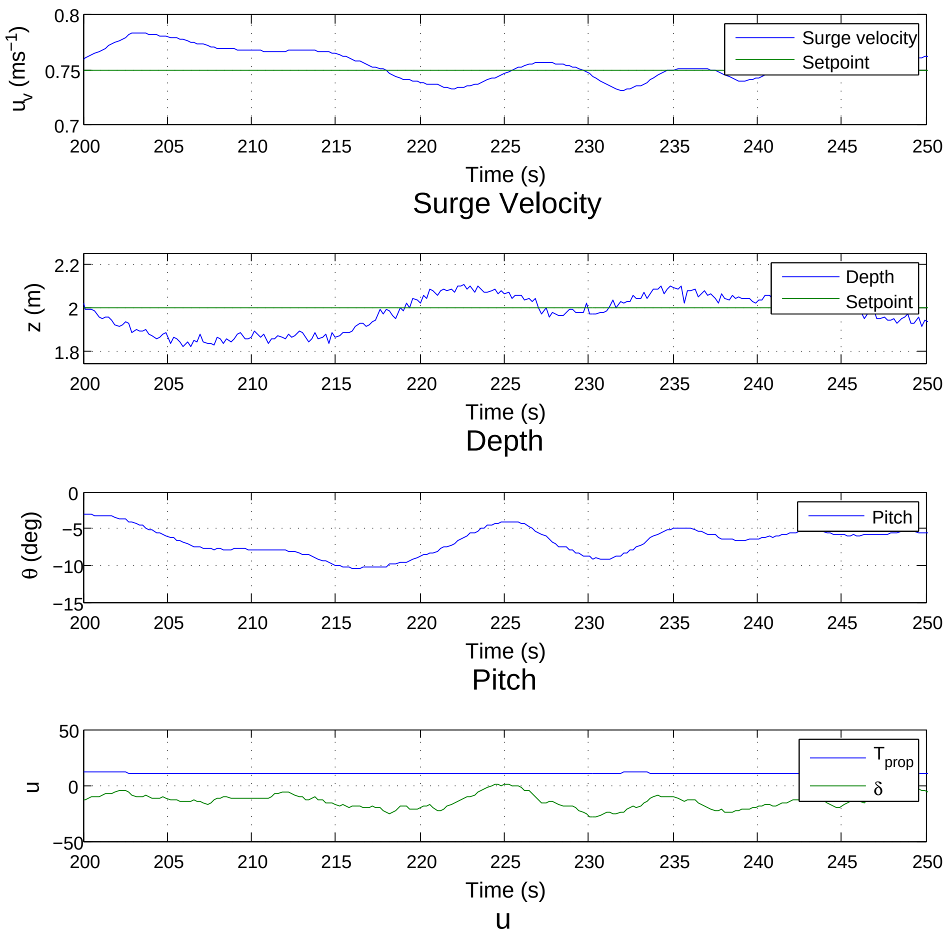


Figure 7.14: Closer look at the results shown in Figure 7.13.

set-point but only after a large overshoot of 0.5 m. Even then the vehicle experiences significant disturbances in depth tracking due to the vehicle oscillating with large amplitudes in pitch. Further to this, with no pitch damping term the overshoot and settling time both increased significantly, therefore the deliberate variation of the damping terms to improve controller performance could still prove a useful tool.

There are three factors that may contribute to the degradation in controller performance between simulation and experimental tests:

- Errors in surge velocity, as this is estimated not measured, could result in substantial errors in the forces and moments produced by the control surfaces and hull.
- The approximation of using the (negative) pitch angle as the angle of attack of flow over the hull may not be sufficiently accurate, along with the absence of flow-straightening effects on the performance of the control surfaces, may cause the forces and moments calculated within the linear state-space model to be inaccurate or even of the wrong sign.
- The heave and pitch damping coefficients are unknown for flight-style operation and have been

taken to be equivalent to those at low-speed hover operation which is unlikely to be particularly accurate.

The first of these factors, inaccuracies in estimated surge velocity, could cause the system to behave badly. For example, if the vehicle estimates its surge velocity at 0.75 m/s whilst in reality it is travelling at 1.0 m/s then the forces produced by the hull and control surfaces will be underestimated by 44%. By conducting simulations whereby the surge velocity, used within the linearisation process, is deliberately multiplied by a scalar to produce an error results in the vehicle oscillating in pitch and as a consequence the depth also oscillates. This effect is significant when the surge velocity is underestimated to be less than 10% of the true value however, overestimating the surge velocity has less of an effect on stability though convergence to the set-points is slightly increased. Estimating the surge velocity of the vehicle, especially as the flow will be very unsteady due to pitch oscillations, could easily result in errors of 25% or more. Without a substantial financial investment to acquire a Doppler velocity log (DVL), or a similar velocity sensor, in order to provide an accurate measurement of surge velocity, then these errors are unlikely to be significantly reduced. For now, it can be deemed better to underestimate the vehicle drag and so overestimate the surge velocity.

The second factor regarding the angle of attack of the flow approaching the hull and control surfaces may also cause degradation of controller performance. Within the non-linear model, used for simulations, the angle of attack for the hull is calculated using a trigonometric function of surge and heave velocity while the angle of attack for the control surfaces is calculated using a function of the hull angle of attack and control surface angle δ. During steady flight, when both pitch and depth are stable, these approximations should be sufficiently accurate however when the vehicle is oscillating in pitch the angle of attack could have cause significant errors. It is not thought that errors in angle of attack are the primary cause for the pitch oscillations however they may contribute to controller degradation.

The damping coefficients for heave and pitch are not known for flight-style operation and have been assumed to be the same as those for low-speed hover operation. Given that the damping coefficients, in particular pitch, appear to have a strong influence on controller performance, in both simulation and experimentally, any inaccuracies will cause substantial degradation in controller performance. More accurate values for these coefficients could be derived from experimental data using a Planar Motion Mechanism (PMM) and Rotating Arm (RA) facility, as in [100], or found using computational fluid dynamics (CFD) as in [101]. Both these techniques are beyond the scope of this work. It is believed by the author that the inaccuracies in damping coefficients do contribute to the poor controller performance when the vehicle was tested experimentally. However, the oscillations in pitch that result in poor depth control are induced by inaccuracies in the estimated surge velocity. Therefore the inaccuracies in damping coefficients amplify other controller problems.

7.9 Conclusions

The flight-style surge velocity and depth controller has shown excellent performance in simulation however during experimental testing the performance was not as good. Convergence to the surge velocity set-points has generally been good, both in simulation and experimentally. However, the surge velocity feedback when operating experimentally is from an open-loop non-linear model, therefore large inaccuracies between estimated and actual surge velocities may exist though this is not due to the actual controller.

It is believed that these inaccuracies in the estimated surge velocity are the primary cause of the degradation in controller performance when tested experimentally, whilst other inaccuracies within the system model amplify these problems. Given the excellent performance when the controller is tested in simulation, it is thought that if both the mathematical model of the vehicle, and the method of calculating the surge velocity, are both improved then TV-MPC control algorithm could prove to be highly successful for flight-style operation.

Chapter 8

Depth Control within the Transitional Zone

8.1 Introduction

In Chapters 6 and 7, two controllers were developed for the Delphin2; one for low-speed hovering operation and the latter for medium to high-speed flight-style operation. The next generation of hover-capable AUVs require a vehicle that can operate throughout a speed range of 0 m/s (or negative) to the maximum surge velocity of the vehicle (for Delphin2 this is 1.2 m/s).

Overlapping both 'low-speed' operation (0 to 0.5 m/s) and 'medium-speed' operation (0.5 to 0.75 m/s) is a range of surge velocities when none of the actuator sets, used for depth control, can independently maintain depth stability of the Delphin2 AUV. This range of velocities is defined as the transitional zone and occurs when the effective force from the thrusters has decreased significantly, due to thruster jet interaction with the AUV hull (Figure 5.23), and when the lift forces from the control surfaces and hull are very low and uncertain due to the possibility of stall. The forces produced by the actuators when operating within the transitional zone are therefore both low in magnitude and high in uncertainty.

The low magnitude of the forces is countered by utilising both the flight-style and hover-mode actuator forces simultaneously. To successfully do this, control authority must be designated to one actuator set. This authority is declared so as to ensure that when one actuator set can independently control the system then the other actuator set is not utilised. For this work the control surfaces have control authority, such that the control surfaces are always active and the thrusters act to support the control surfaces at low speeds. Without this authority both the thrusters and control surfaces could be operating at high speed, reducing system efficiency and potentially leading to poor system performance.

The uncertainty associated with both the system inputs and system dynamics, when operating within the transitional zone, is one of the primary reasons for choosing this MPC algorithm for investigation. Within most low-level controllers that are required to handle model inaccuracies or uncertainties is an integrator that can, as time approaches infinity, help to reduce these problems. The MPC algorithm evaluated within this work has an embedded integrator that has been shown to handle these problems, e.g. compensating for the buoyancy force disturbance that is excluded from the linear depth model.

In this chapter a controller is developed tasked with controlling both depth and surge velocity of the Delphin2 AUV throughout the entire speed range of the Delphin2 AUV, from 0 m/s to 1.2 m/s. To achieve this four system inputs are used; the rear propeller, the horizontal control surfaces (treated as one input) and the front and rear vertical tunnel thrusters.

First the over-actuated model of the Delphin2 AUV and a linearised model are derived. Next, changes to the TV-MPC algorithm, in order to provide control authority to the control surfaces, is introduced and its performance evaluated under simulation. The controller is then experimentally tested on the Delphin2 AUV and the results discussed.

8.2 Over-Actuated AUV System

When the Delphin2 AUV is operating within the transitional zone it requires four actuators to control three degrees of freedom; the rear propeller, the horizontal control surfaces (counted here as one actuator), and the front and rear through-body tunnel thrusters. There are therefore more actuators than degrees of freedom, this is defined as over-actuated. Note that the Delphin2 AUV is over-actuated only when the AUV is not stationary (relative to the surrounding water) as at zero forward speed the control surfaces are not effective.

The mathematical model used to describe the motions of the AUV throughout the speed range is similar to that of the flight-style model but with the addition of the forces and moments generated by the thrusters. As with the flight-style controller, the three degrees of freedom that are modelled and included within the controller design are motion along the x and z (surge and depth) axes and rotation about the y axis (pitch).

8.2.1 Surge

The surge velocity is modelled as a first order system. One mass term, including added mass, is used along with two input and two dynamics components. The acceleration in surge is identical to that of equation (7.1) but with the additional force from the thrusters. This force is proportional to the sine of the vehicle's pitch. In normal operation the pitch of the AUV is generally low thus the force from the vertical thrusters acting along the x axis is also generally low.

One point worth noting is that if the propeller force was omitted from equation (8.1) and a controller was tasked with generating a positive surge velocity then the only method would be to use the thrusters to generate a positive force by setting the pitch angle of the vehicle to greater than zero degrees. As the surge velocity increases this positive pitch angle could potentially cause stability issues in depth control due to the lift force from the hull counter-acting the forces from the thrusters. Realistically the thrusters would not be tasked with, nor capable of, generating a surge velocity great enough to cause this issue but the control designer should be aware of this.

$$\begin{aligned} \dot{u_v} = \frac{1}{m_x} \Bigg[& \overbrace{(T_{prop}\cos\theta)}^{\text{Propeller force}} + \overbrace{((T_{vf}+T_{vr})\sin\theta)}^{\text{Thruster force}} \\ & - \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_{Dh}(-\theta)|u_v|u_v\right)}^{\text{Drag force due to hull}} - \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_{Dcs}(\delta)|u_v|u_v\right)}^{\text{Drag force due to control surfaces}} \Bigg] \end{aligned} \tag{8.1}$$

$$u_v = \int_0^t \dot{u_v}\ dt \tag{8.2}$$

8.2.2 Depth

The position of the vehicle along the z axis, depth, is modelled as a second-order system. One mass term, that includes added mass, is used within the heave acceleration equation along with three input, two dynamic and one disturbance components. Again this model is similar to that used for flight-style, equation (7.3), but with the addition of the thruster forces.

Within equation (8.3) are two dominating forces; the force from the thrusters, and the lift force from the hull. The dominance of these two forces is dependent on the surge velocity, therefore at low speeds the thrusters will be used while at higher speeds the hull force will dominate.

$$\begin{aligned}\dot{w_v} = \frac{1}{m_z}\Bigg[& \overbrace{(W-B)}^{\text{Buoyancy force}} + \overbrace{(T_{prop}\sin\theta)}^{\text{Propeller force}} + \overbrace{((T_{vf}+T_{vr})\cos\theta)}^{\text{Thruster force}} + \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_{Lh}(-\theta)|u_v|u_v\right)}^{\text{Hull lift}} \\ & - \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_{Lcs}(\delta)|u_v|u_v\right)}^{\text{Control surface lift}} - \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_{Dw}|w_v|w_v\right)}^{\text{Damping force}}\Bigg]\end{aligned} \tag{8.3}$$

$$w_v = \int_0^t \dot{w_v}\ dt,\ z = \int_0^t w_v\ dt \tag{8.4}$$

8.2.3 Pitch

The pitch angle of the vehicle is modelled as a second-order system. One inertia term, that includes added inertia, is used within the pitch acceleration along with two input and three dynamics components. Again, like the surge and depth models, the pitch equation is identical to that used for modelling flight-style operation, equation (7.5), but with the addition of the moments generated by the thrusters. These moments are derived by the forces from each thruster multiplied by the moment arm from the centre of gravity.

$$\begin{aligned}\dot{q_v} = -\frac{1}{I_y}\Bigg[& \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_{Mh}(\alpha)|u_v|u_v\right)}^{\text{Hull moment}} + \overbrace{(x_{Tvf}T_{vf} + x_{Tvr}T_{vr})}^{\text{Thruster moment}} + \overbrace{(z_g W \sin\theta)}^{\text{Restoring moment}} \\ & + \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_{Mcs}(\delta_{cs})|u_v|u_v\right)}^{\text{Control surface moment}} + \overbrace{\left(\frac{1}{2}\rho V^{2/3} C_{Dq}|q_v|q_v\right)}^{\text{Damping moment}}\Bigg]\end{aligned} \tag{8.5}$$

$$q_v = \int_0^t \dot{q_v}\ dt,\ \theta = \int_0^t q_v\ dt \tag{8.6}$$

8.3 Linearisation

The system model for the Delphin2 AUV operating in over-actuated mode is very similar to that of the flight-style model but with the addition of the thruster forces and moments. Rather than deriving the full linearisation of the system, the state-space model used within the flight-style controller will be modified to include the linearised components due to the thrusters. Please refer to section 7.4 for the derivation of the flight-style model.

8.3.1 Thruster components

One non-linearity that is not included within equations (8.1) to (8.5) is the influence of surge velocity on the effective force produced by each thruster, as discussed in Section 5.3.6. To approximate this reduction in effective thrust a scalar coefficient will be created using the latest surge velocity:

$$k_{Tu_w} = e^{-|u_w(k)|} \tag{8.7}$$

The force from the thrusters acting along the x axis is a function of both thruster inputs and the sine function of the vehicle pitch angle. First the sine function is omitted from the equation using small angle approximation. When the vehicle is operating with the thrusters at a fixed depth then the magnitude of the thrust should stay relatively constant. For this reason, and the fact that the controller must 'know' to adjust the vehicle's pitch to adjust the force along the x axis, the pitch angle is chosen as the active variable. The force acting along the x axis is therefore calculated using the latest thrust values:

$$X_T = k_{Tu_w}\left(T_{vf}(k) + T_{vr}(k)\right) \times \theta \tag{8.8}$$

Using small angle approximation the cosine function is taken to equal 1 and so the forces from the thrusters acting along the z axis are directly included within the B matrix:

$$Z_T = k_{Tu_w}\left(T_{vf} + T_{vr}\right) \tag{8.9}$$

The moments generated by the thrusters can be directly included within the B matrix using the moment arms:

$$M_T = k_{Tu_w}\left(x_{Tvf}T_{vf} + x_{Tvr}T_{vr}\right) \tag{8.10}$$

8.3.2 State-Space Model

Combining the forces and moments generated by the vertical tunnel thrusters, equations (8.7) to (8.10), into the flight-style state-space model, equation (7.14), the over-actuated state-space model is created:

$$\dot{x_m}(t) = \overbrace{\begin{bmatrix} \frac{(k_{Dh}+k_{Dcs})}{m_x} & 0 & 0 & \frac{k_{Tuw}\left(T_{vf}(k)+T_{vr}(k)\right)}{m_x} & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{(-k_{Dz})}{m_z} & \frac{(T_{prop}(k)+k_{Lh})}{m_z} & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{(k_{Mh}+z_g W)}{I_q} & \frac{k_{Dq}}{I_q} \end{bmatrix}}^{A_m} \begin{bmatrix} u_v \\ z \\ w_v \\ \theta \\ q_v \end{bmatrix}(t)$$

$$+ \overbrace{\begin{bmatrix} \frac{1}{m_x} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{k_{Lcs}}{m_z} & \frac{k_{Tuw}}{m_z} & \frac{k_{Tuw}}{m_z} \\ 0 & 0 & 0 & 0 \\ 0 & \frac{k_{Mcs}}{I_q} & \frac{x_{Tvf}k_{Tuw}}{I_q} & \frac{x_{Tvr}k_{Tuw}}{I_q} \end{bmatrix}}^{B_m} \begin{bmatrix} T_{prop} \\ \delta \\ T_{vf} \\ T_{vr} \end{bmatrix}(t) \tag{8.11}$$

$$y(t) = \overbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}}^{C_m} x_m(t) \tag{8.12}$$

8.4 Controller Design

The over-actuated controller developed in this Chapter uses the time-variant MPC algorithm (TV-MPC) first introduced in Chapter 7. This utilises one linear state-space model that has the parameters updated using the latest state-feedback before the optimal ΔU vector is calculated. For experimental verification, the Luenberger observer is used to estimate the state-variable vector. The design of this observer is discussed in Section 7.5.

8.4.1 Tuning

The weighting matrix, $\bar{R}$, is built using the same structure as with the flight-style controller such that different weighting scalars can be applied to the different actuators:

$$\bar{R}_{ij} = \begin{cases} 0_{m\times m} & \text{for } i \neq j \\ \begin{bmatrix} r_{w(1)} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & r_{w(m)} \end{bmatrix} & \text{for } i = j \end{cases} \tag{8.13}$$

where for this controller $r_{w(1)}$ to $r_{w(4)}$ are the weighting scalars for T_{prop}, δ, T_{vf} and T_{vr} respectively.

The lengths of the prediction and control horizons, N_p and N_c, will be found later using trial and error.

8.4.2 Constraints

Unlike the previous controllers developed in this Thesis, some of the constraints imposed on the over-actuated controller are scheduled depending on the state of the system. This has been done to help avoid detrimental effects produced by non-linearities associated with certain actuators. The constraints are set at the beginning of each new sample using the latest feedback and controller set-points.

For T_{prop}, the rate of change constraints (Δu) are fixed throughout the entire operating range at ± 1.0 N. However, the minimum and maximum absolute constraints are scheduled depending on the surge velocity set-point. Therefore, if this set-point is between 0.0 and 0.3 m/s then the minimum and maximum T_{prop} constraints are both set to 0.0 N. With surge velocity set-points greater than 0.3 m/s the constraints are set as 2.0 and 16.0 N respectively. The reason for doing this is to avoid the propeller motor dead-band. This non-linearity results in a minimum surge velocity of approximately 0.35 m/s below which the propeller motor stalls, therefore between 0.0 and 0.3 m/s the thrusters will be used to control surge velocity.

The rate of change constraints for both the front and rear thrusters, and the maximum absolute thrust constraint, are set at ± 2.0 N and 10.0 N respectively for the full operating range. The minimum thrust constraint is scheduled using the latest surge velocity such that when it is less than 0.4 m/s the constraints are set at 0.7 N, and if greater than or equal to 0.4 m/s the constraint is set at 0.0 N. The reasoning for this is that when operating at surge velocities below 0.4 m/s the thrusters will be required to help control depth, therefore constraining them at 0.7 N means that they are already spinning at a speed above their associated thruster motor dead-zone. At high surge velocities, it is desirable to completely switch off the thrusters to conserve energy, therefore the constraint is adjusted.

The constraints imposed on the control surfaces are the same as with the flight-style controller, with rate of change and absolute constraints of $\pm 2.0^\circ$ and $\pm 30^\circ$ respectively.

8.5 Controller Tuning by Model Adaptation

Before attempting over-actuated control, the performance of the MPC controller will be evaluated at surge velocities of 0.0 and 1.0 m/s. The reason for doing this is to evaluate whether one controller design will provide acceptable performance even when the dynamics matrix has changed substantially (due to the linearisation process).

When the Delphin2 AUV has a positive surge velocity both the thrusters and control surfaces can be used for depth control. In the next section a method to switch off the thrusters at high surge velocities is given. For now, it will be assumed that at high surge velocities the control surfaces will control depth, and so for this simulation the thrusters will be disabled by removing their components from the B_m matrix. Likewise at zero surge velocity the inputs from the control surfaces will be emitted from the B_m matrix.

8.5.1 Low Speed Operation

Using the TV-MPC algorithm the controller is tasked with achieving a depth set-point of 1.0 m while maintaining a surge velocity of 0.0 m/s. No inner gains have been applied to the system model as were used in the flight-style controller in the previous chapter.

Starting with an N_c value of 8, as used in the previous two chapters, the values for N_p and $r_w(T)$ were found using trial and error. Figure 8.1 presents the simulation results of the controller using N_p and $r_w(T)$ values of 150 and 4 respectively.

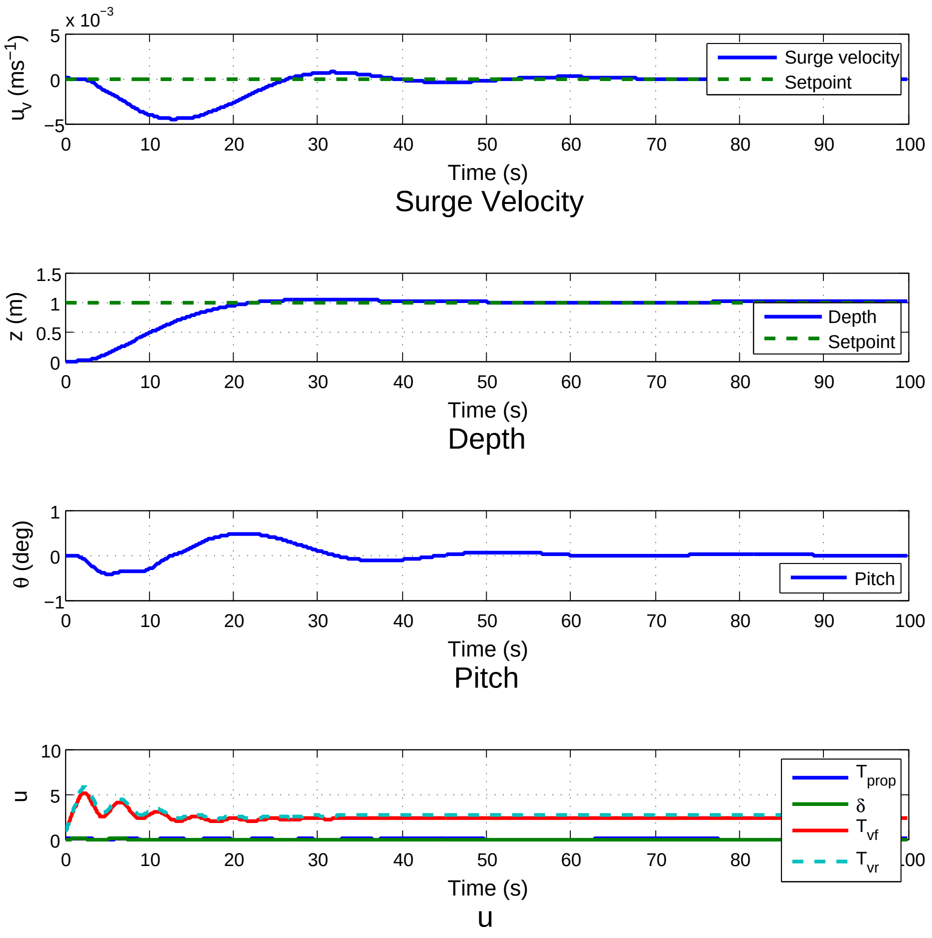


Figure 8.1: Simulation results of the controller, with only the thrusters active, with a step change in depth set-point from 0.0 to 1.0 m while maintaining a surge velocity of zero. Note a slight disturbance to surge velocity whilst diving phase but converges back to zero.

The performance of the controller is excellent, with minimal disturbance to the desired zero surge velocity and a small overshoot of less than 4 cm. In comparison to the depth and pitch controller, developed in Chapter 6, the convergence of depth to its set-point is approximately five seconds slower. This is due to the much longer control horizon, used by this controller, that is required to provide stability in surge control. The control input u appears under-damped with several oscillations before converging to fixed values. Increasing the $r_w(T)$ value reduces these oscillations but also further increases the depth convergence time.

8.5.2 High Speed Operation

Using the same N_p and N_c values as for low speed operation (150 and 8 respectively) and with an $r_w(cs)$ value of 0.1 (as used by the flight-style controller in Chapter 7) the TV-MPC controller was tested with surge velocity and depth set-points of 1.0 m/s and 1.0 m respectively. Figure 8.2 presents the results of these simulations, showing excellent surge velocity control with quick convergence and no overshoot. Unfortunately the depth control is poor, with very slow convergence to the set-point and an unacceptably large overshoot.

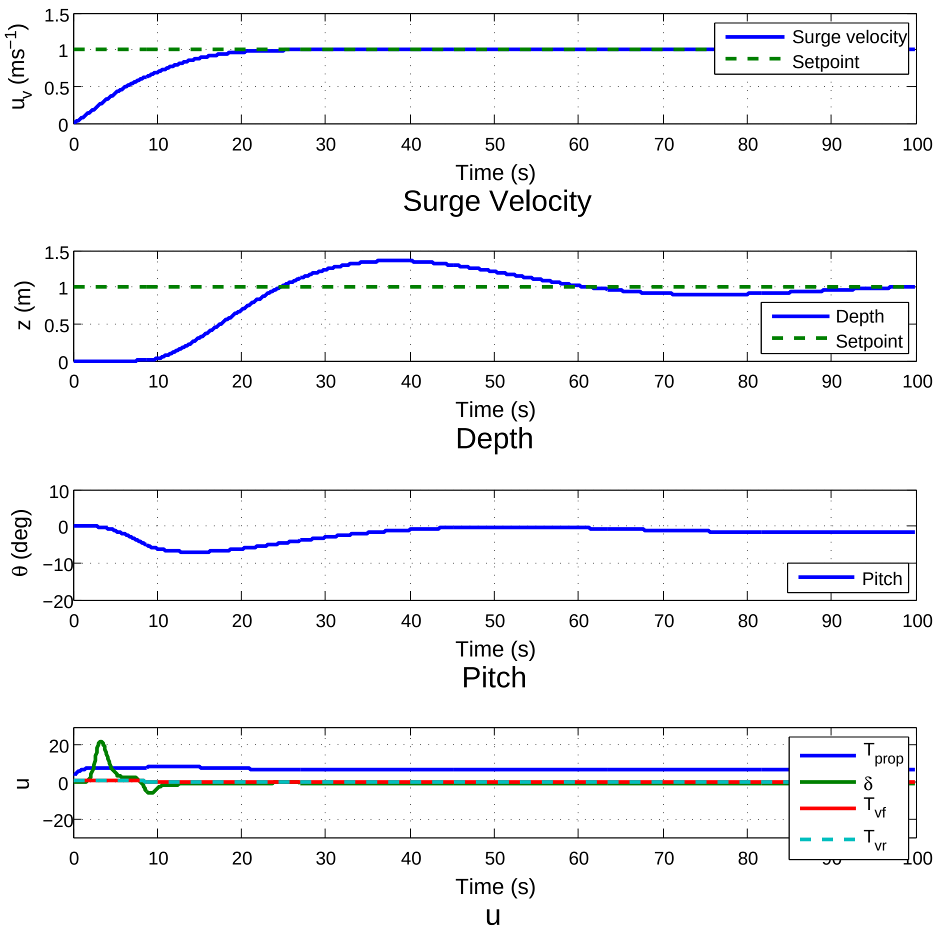


Figure 8.2: Simulation results using the standard over-actuated controller, with thrusters disabled. Note quick convergence to the surge velocity set-point with no overshoot.

The depth control performance in Figure 8.2 resembles that in Figure 7.6 when using the flight-style controller. To improve the performance of the flight-style controller, gains were added to the heave and pitch damping terms to move the poles of the system further to the left of the complex plane away from the real axis, see Section 7.6.2. This resulted in the flight-style controller acting in a more aggressive manner and greatly improved the performance of depth control when tested in simulation. However, when this was tested experimentally the gains appeared to have a detrimental effect on controller performance. This was attributed to the inaccurate estimation of the surge velocity that caused the controller to generally behave poorly. For this section, it will be assumed that if the surge velocity was accurately measured then the pole placement method would improve controller performance when tested experimentally.

Therefore using the same method as described, with the same fixed damping terms and gains as used in Section 7.6.2, the poles of the over-actuated model are moved to the left. Figure 8.3 presents the results for the same set-points as before but with the new model. Clearly the depth control has been drastically improved with very fast convergence and minimal overshoot.

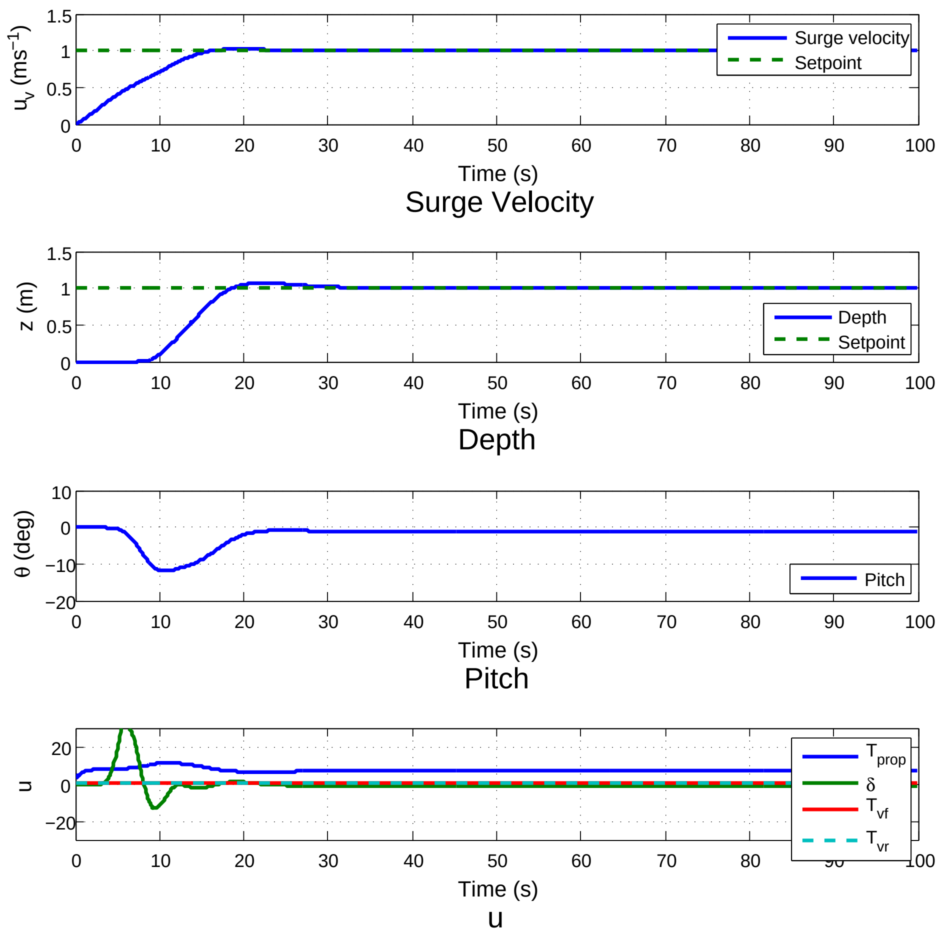


Figure 8.3: Simulation results using the over-actuated controller, with thrusters disabled, and gains applied to the heave and pitch damping terms (both equal to 40).

8.5.3 Effect of Inner Gains

In the previous two subsections it was shown that the standard TV-MPC controller works well in hover-mode at maintaining zero surge velocity and achieving depth set-points however, when operating in flight-style mode inner gains must be applied to the heave and pitch damping terms in order to achieve good performance. Figure 8.4 presents the performance of the controller in hover mode, using the same inner gains that made the flight-style performance better. Clearly the addition of the gains within the system model are highly detrimental to hover mode control with depth, pitch and surge velocity oscillating badly.

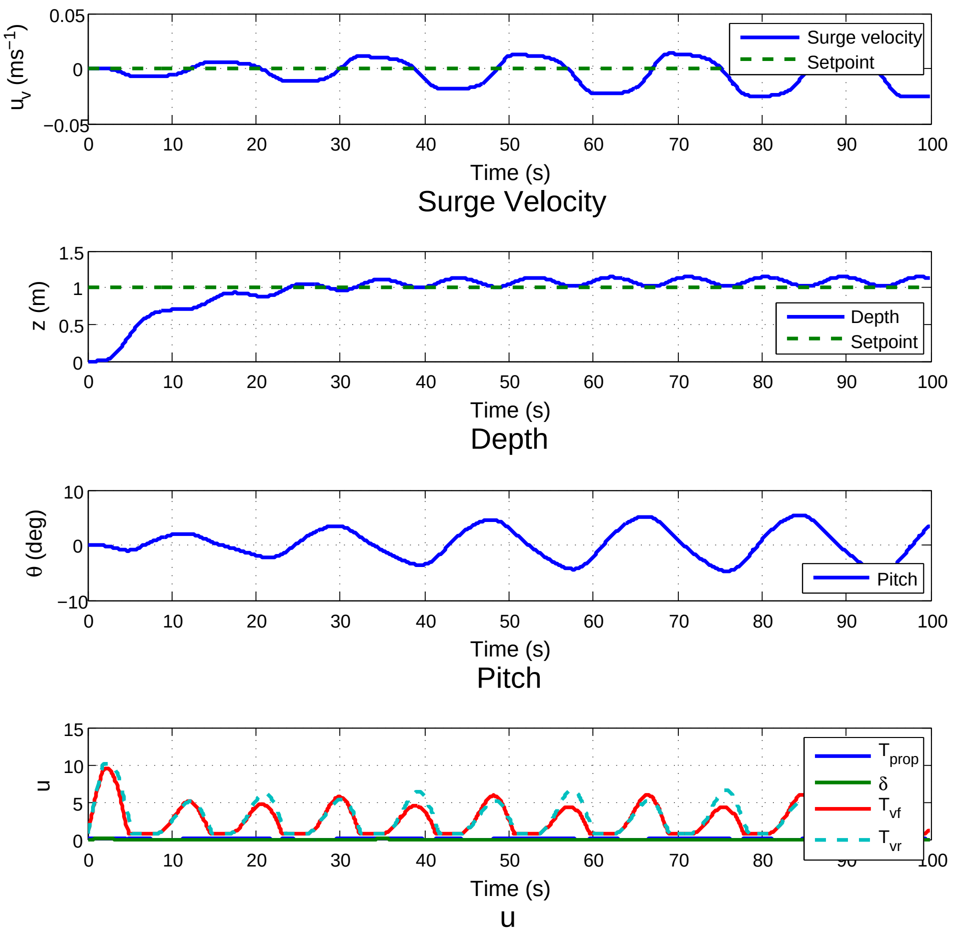


Figure 8.4: Simulation results using the over-actuated controller with gains applied to the heave and pitch damping terms (both equal to 40). A depth set-point of 1.0 m is applied whilst maintaining zero surge velocity.

It is not surprising that what works well in hover mode does not work for flight-style operations, and vice versa, as the two systems are significantly different. Given that the only difference between the two controllers is the inner gains then a method to schedule these gains depending on the current operational conditions is required.

The modified dynamics matrix, A_m, with the scheduled damping takes the form:

$$A_m = \begin{bmatrix} \frac{(k_{Dh}+k_{Dcs})}{m_x} & 0 & 0 & \frac{k_{Tuw}\left(T_{vf}(k)+T_{vr}(k)\right)}{m_x} & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \boldsymbol{\frac{(k_z+k_z|u_v(k)|G_z)}{m_z}} & \frac{(T_{prop}(k)+k_{Lh})}{m_z} & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{(k_{Mh}+z_gW)}{I_q} & \boldsymbol{\frac{(k_q+k_q|u_v(k)|G_q)}{I_q}} \end{bmatrix} \tag{8.14}$$

The changes to the dynamics matrix are highlighted in equation (8.14) in bold. These modifications include the fixed damping terms, used previously by both the depth and pitch controller and the flight-style controller, plus additional damping as a function of the absolute surge velocity and the gains. This will ensure that the gains are not active at low surge velocities when operating in hover mode but are active when operating in flight-style mode.

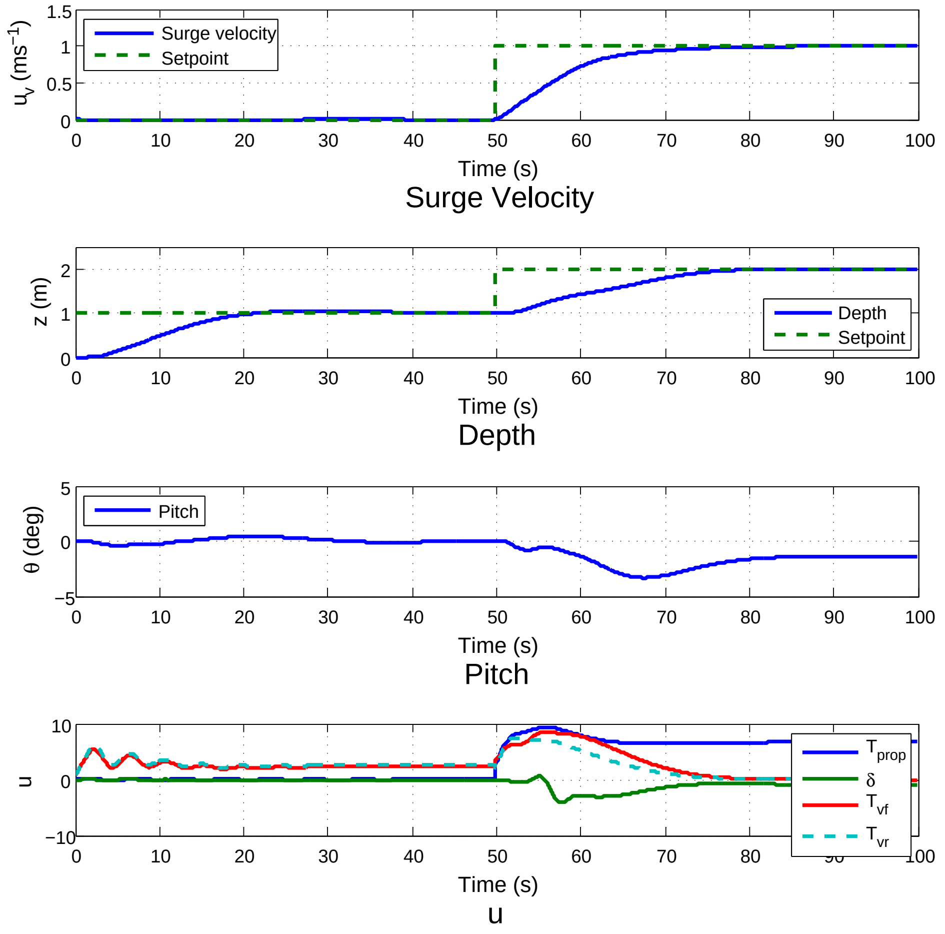


Figure 8.5: Simulation results demonstrating the effectiveness of using the surge velocity to schedule the inner gains applied to the system model.

8.6 Actuator Transition

As the thrusters and the control surfaces can both be used to control depth, then control authority must be designated to one actuator set so as to prevent all the actuator sets being used when one should suffice. The actuator set that has control authority is set as the primary input, therefore the supporting actuator sets are only active when the primary actuator set is unable to achieve the desired set-point or requires assistance to achieve it.

For Delphin2 it is desirable to give control authority to the control surfaces and have the thrusters as the supporting actuator set. The reason for this choice is that it would be very inefficient to use the thrusters to control depth at high surge velocities when the control surfaces should be capable of this independently. Figure 8.6 presents simulation results of the over-actuated system, without control authority, with surge velocity and depth set-points of 1.0 m and 1.0 m/s respectively. Although both surge velocity and depth converge quickly and without overshoot it should be noted that both thrusters are at or near their maximum constraints thus unnecessarily consuming a high amount of electrical energy.

The solution to the cost function used within the MPC algorithm, equation (3.20), is found by minimising the errors, between set-points and outputs, and minimising the magnitude of the changes in future inputs. The cost function is unaware of the absolute input values and so the option to directly penalise high thruster values is not available. Changing the cost function to solve the optimum absolute control inputs is not an option as this would remove the embedded integrator that has proved highly advantageous in this work so far. Instead, an additional component will be added to the current cost function.

Thus far, the ΔU vector has been found by solving:

$$\Delta U = \overbrace{(\phi^T\phi + \bar{R})^{-1}}^{\text{Penalise high }\Delta\text{u values}} \overbrace{(\phi^T R_s - \phi^T F x(k_i))}^{\text{Reduce control errors}}, \tag{8.15}$$

The first part of equation (8.15) uses the weighting matrix $\bar{R}$ to penalise high ΔU values, and the second part of the equation is used to reduce the control errors. The method proposed here, to penalise the use of the thrusters at higher surge velocities, requires an additional component in the second part of equation (8.15). The control authority component takes the form:

$$\eta = G_T|u_w(k)| \begin{bmatrix} 0, & 0, & T_{vf}(k), & T_{vr}(k), & 0_{(1\times(mN_c-m))} \end{bmatrix}^T \tag{8.16}$$

where G_T is a scalar used to tune the control authority. Note that the absolute value of u_w is used, as this is required to maintain stability. If the non-absolute value is used, and the surge velocity of the vehicle is negative, then the system will cause the thrusters to quickly 'wind-up' and cause the vehicle to dive uncontrollably.

Inserting equation (8.16) into (8.15) gives:

$$\Delta U = (\phi^T\phi + \bar{R})^{-1}(\phi^T R_s - \phi^T F x(k_i) - \eta) \tag{8.17}$$

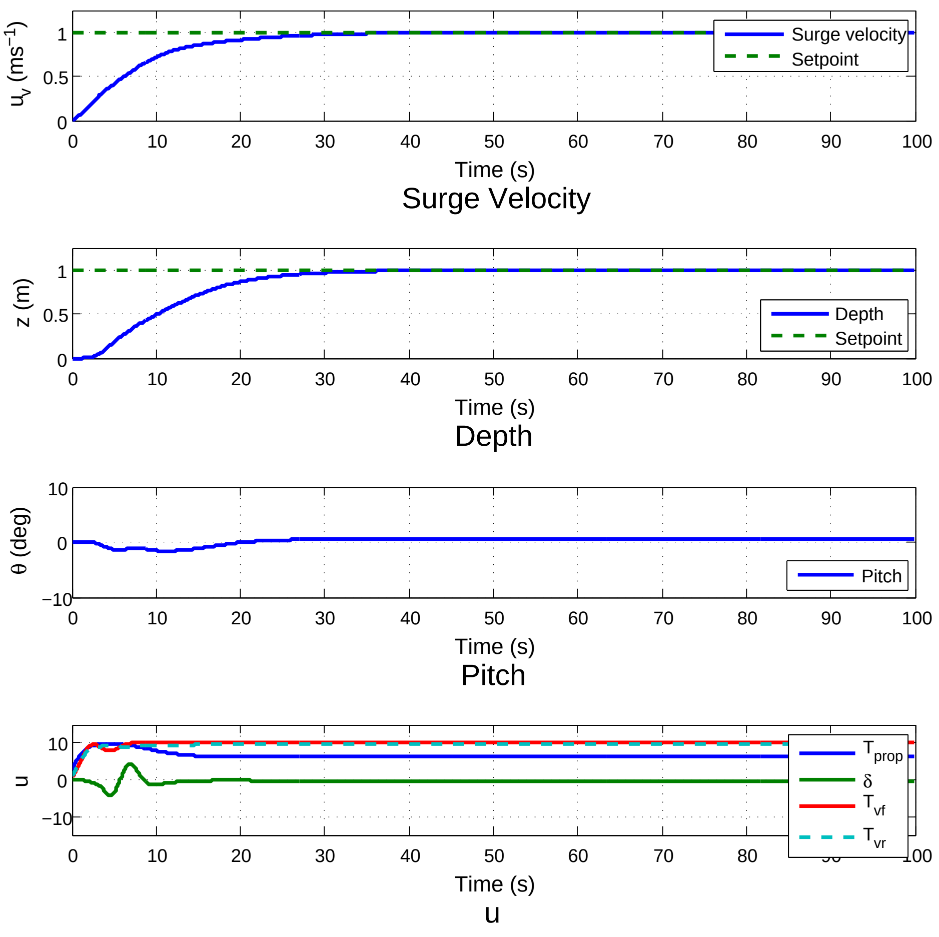


Figure 8.6: Simulation of over-actuated controller achieving surge velocity and depth set-points of 1.0m and 1.0m/s respectively. The controller has no designated control authority, and so the thruster demands are near their maximum constraints during high speed operating when the control surfaces control depth independently.

The η vector penalises the use of the thrusters and is proportional to surge velocity and thrust magnitude. The negative feedback acts to offset the control error so as to cause the controller to 'wind back' the thrust values. The η vector has no direct impact on the control surfaces, thus when the error is small and the vehicle is moving with a positive surge velocity the thrusters will be penalised.

At low surge velocities, when the control surfaces have little effect, the η vector may cause a small steady-state error in depth control. The choice of G_T, found by trial and error, should be made as a compromise between low speed steady-state errors and the speed at which the thrusters switch off. Therefore, a low G_T value should give a smooth transition, but at the expense of running the thrusters longer, compared to a high G_T value that has inferior low speed performance but transitions quicker to flight-style operation and so may prove more efficient.

Figure 8.7 presents simulation results of the over-actuated controller, with control authority and a G_T value of 0.1, with surge velocity and depth set-points of 1.0 m and 1.0 m/s respectively. This is the

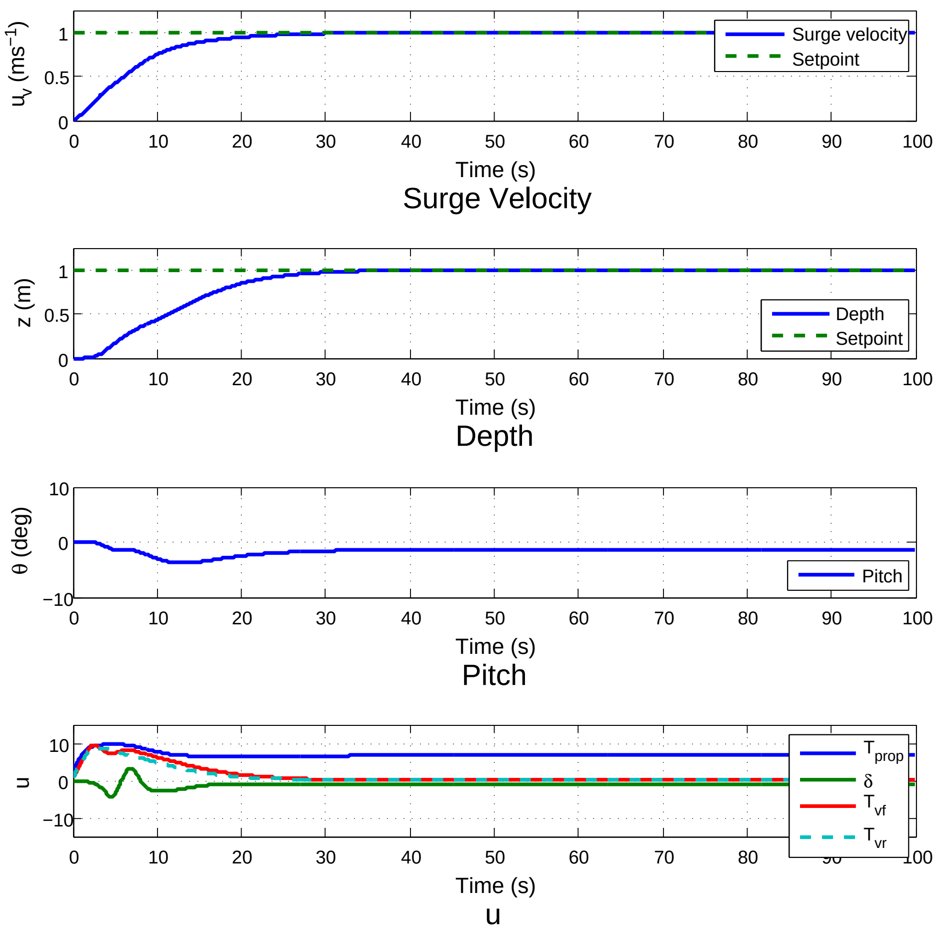


Figure 8.7: Simulation of over-actuated controller achieving surge velocity and depth set-points of 1.0m and 1.0m/s respectively. The control authority is designated to the control surfaces by penalising the use of the thrusters using equations (8.15) and (8.16).

same step changes and system as was used in Figure 8.6 yet with control authority the thrusters switch off when the depth error approaches zero. Note that the transition is smooth and has no noticeable adverse effects on the controller performance.

8.7 Experimental Verification

The over-actuated controller has been translated into the Python programming language and integrated into the Delphin2 control software as a separate node. The vehicle was tested at Testwood Lake near Southampton (latitude: 50.937932, longitude: -1.507359), which measures approximately 270 m long, 125 m wide and has an average depth of 3.0 m.

Note that the inner gains, used to adjust the heave and pitch damping coefficients, are not included within the controller design because of the stability issues noted when testing the flight-style controller, see Section 7.7. The controller, when first tested, performed adequately however it caused both of the AUV's CPUs to operate near to their maximum capacity. Therefore, to ensure soft real-time operation and 'free-up' CPU time for other processes the sampling rate was decreased from 10 Hz to 5 Hz. The controller parameters that were used to produce the results shown in this section can be found in Table 8.1. Note that the prediction horizon is approximately half of those used at 10 Hz. The rest of the parameters were found through trial and error in simulation.

Table 8.1: Tuning parameters for over-actuated controller

Parameter	**Value**
N_p	60
N_c	8
$r_{w(prop)}$	1.0
$r_{w(cs)}$	0.005
$r_{w(T)}$	0.2
G_T	0.05

Figure 8.8 presents the experimental results of the over-actuated controller controlling the surge velocity and depth of the Delphin2 AUV. The test starts with initial conditions of approximately 0.0 m/s and 0.0 m, and set-points of 0.0 m/s and 1.0 m, for surge velocity and depth respectively. The depth set-point remains constant at 1.0 m through-out the test whilst the surge velocity set-points vary from 0.5 to 0.75 and then back to 0.0 m/s. Note that unlike the previous Figures presenting controller results, the signal for δ has been plotted in a separate sub-plot at the bottom of the Figure so as to enable the reader to examine the different actuator settings in more detail. Also, vertical black dashed lines have been plotted on the bottom four sub-plots that correspond to the changes in surge velocity set-points.

Starting with the first set-points of 0.0 m/s and 1.0 m, the AUV dives, using only the thrusters, to the depth set-point in approximately 15 seconds with a small overshoot of 0.08 m. The vehicle is then stable at both set-points when the first step-change in surge velocity set-point occurs. As it begins to accelerate the vehicle loses depth and approaches the surface before slowly converging back to the depth set-point, the maximum disturbance was 0.3 m. As the vehicle approaches the depth set-point the thrusters start to wind-back to almost zero as the vehicle develops a large negative pitch angle to generate sufficient lift from the hull to overcome the buoyancy force. The cause of this initial disturbance can be attributed to three factors; the effective force from the thrusters quickly decreases with increasing surge velocity, the initial angle of the control surfaces caused the vehicle to momentarily generate a positive pitch angle thus generating a negative hull lift force, and the control authority component within the controllers cost function deemed it optimal to accept the depth error in favour of not increasing the magnitude of the forces demanded of the thrusters.

By the time the second step-change in surge velocity, to 0.75 m/s, occurs the AUV is operating almost entirely in flight-style mode. As the vehicle accelerates to meet the new surge velocity set-point it overshoots in depth by 0.39 m before quickly converging back to the depth set-point. This disturbance closely matches that experienced by the flight-style controller tested in the previous Chapter and is caused by the controller's inability to accurately predict the true optimal future pitch angle of the vehicle. Thus as the vehicle accelerates, from 0.5 m/s to 0.75 m/s, the pitch angle is not increased quickly enough and so the hull generates too much lift and the AUV dives beyond the depth set-point. After this disturbance the vehicle tracks both set-points reasonably well however the depth tracking is not smooth due to significant fluctuations in pitch angle.

The final step-change of surge velocity, from 0.75 to 0.0 m/s, occurs at 308 seconds into the test. This is a rapid change from flight-style to hover-mode operation. As the vehicle slows, both the thrusters and control surfaces try to pitch the vehicle down (negative pitch) so as to generate as much hull lift as possible but also to use the thrusters to actively brake. The thrusters quickly wind-up as the surge velocity decreases and the surge velocity and depth both converge to zero after 36 seconds. The momentary loss in depth, as the AUV starts to slow down is caused by the same factors as with the first step-change in surge velocity as discussed earlier in this Section.

The overall performance of the over-actuated controller has worked well, with relatively smooth transition from using primarily the thrusters to using the control surfaces. It can be seen the thruster signals are not exactly zero when operating in flight-style mode. This is a result of the unsteadiness of the flight-style controller, therefore requesting thruster forces to help damp out the oscillations in depth and pitch. It is believed that if the issues with the flight-style controller, primarily the addition of an accurate velocity measurement sensor to the vehicle, were resolved and the depth and pitch did not oscillate then the thrusters would switch off entirely.

Figure 8.9 presents the experimental results of the over-actuated controller handling a large step-change in surge velocity set-point. The AUV starts by diving using the thrusters to 1.0 m depth at a surge velocity of 0.0 m/s, then at 70 seconds into the test the surge velocity set-point changes to 1.0 m/s. As the vehicle quickly accelerates a very large disturbance of 0.69 m to the depth tracking occurs before converging back to 1.0 m. The disturbance is partly due to the same causes as in the previous Figure but with one additional factor. The controller uses the thrusters to generate a positive pitch angle so as to use the combined forces of the thrusters to accelerate the AUV along the x axis. However, as the surge velocity of the vehicle increases, this positive pitch angle generates a negative hull lift force that accelerates the vehicle towards the surface. As the surge velocity reaches approximately 0.5 m/s, the controller realises that the optimum solution is to transit with a negative pitch angle and utilises the control surfaces so as to achieve this. The vehicle stabilises at the depth set-point 20 seconds after the step-change is surge velocity set-point.

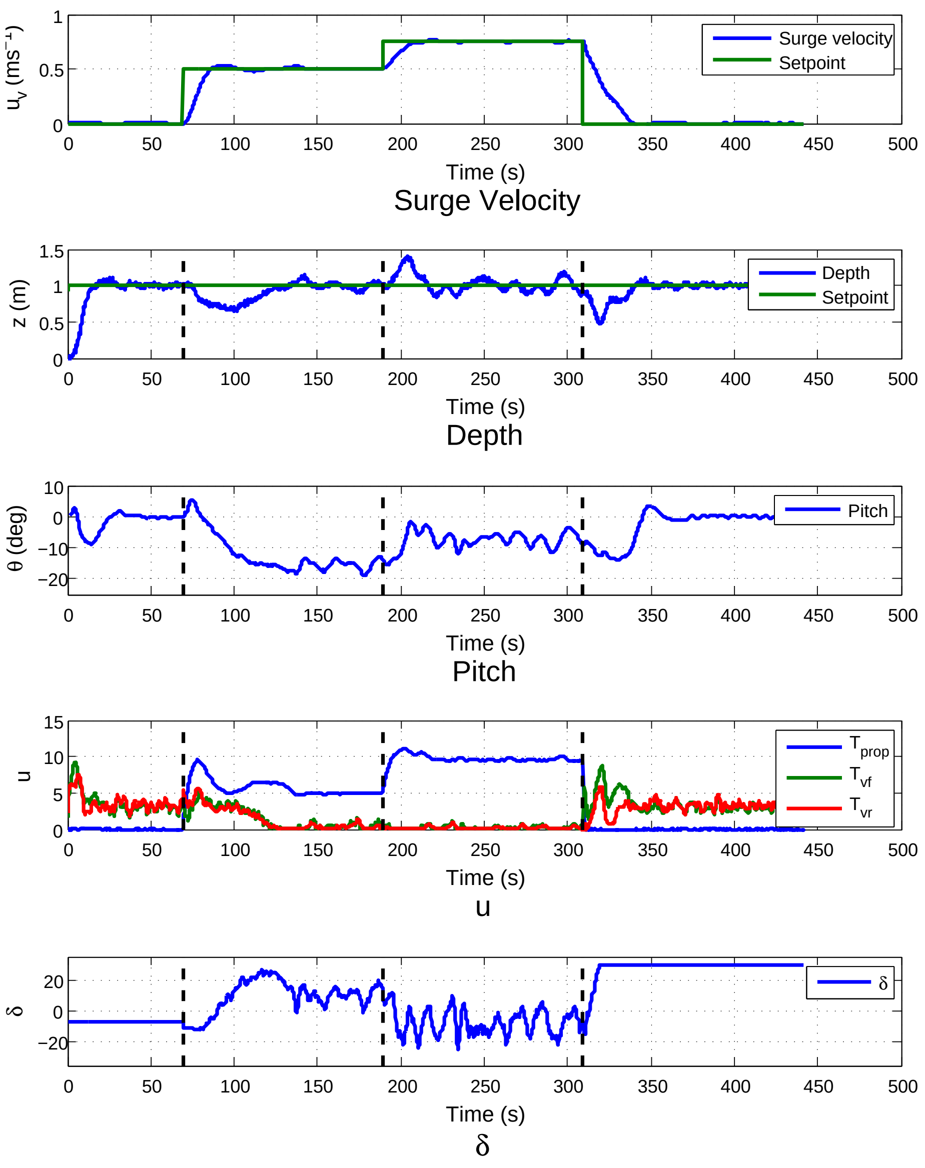


Figure 8.8: Experimental results of the over-actuated surge velocity and depth controller. One depth set-point of 1.0 m is used with step changes in surge velocity set-points from 0.0 to 0.5 to 0.75 and then back to 0.0 m/s. Initial surge velocity and depth are both approximately zero.

8.8 Conclusions

In this chapter the TV-MPC algorithm has been used to design a surge velocity and depth controller for the Delphin2 AUV. This controller is tasked with operating the vehicle efficiently throughout the full velocity range of the vehicle. The system is a combination of the dynamics from the hovering and flight-style systems modelled in Chapters 6 and 7 respectively. To ensure depth stability from low to high speeds, four actuator sets were required; the rear propeller, the horizontal control surfaces and the front and rear through-body tunnel thrusters.

Using the standard TV-MPC algorithm, a controller was designed and tested in simulation. This showed excellent performance at controlling both outputs by using all of the available actuators.

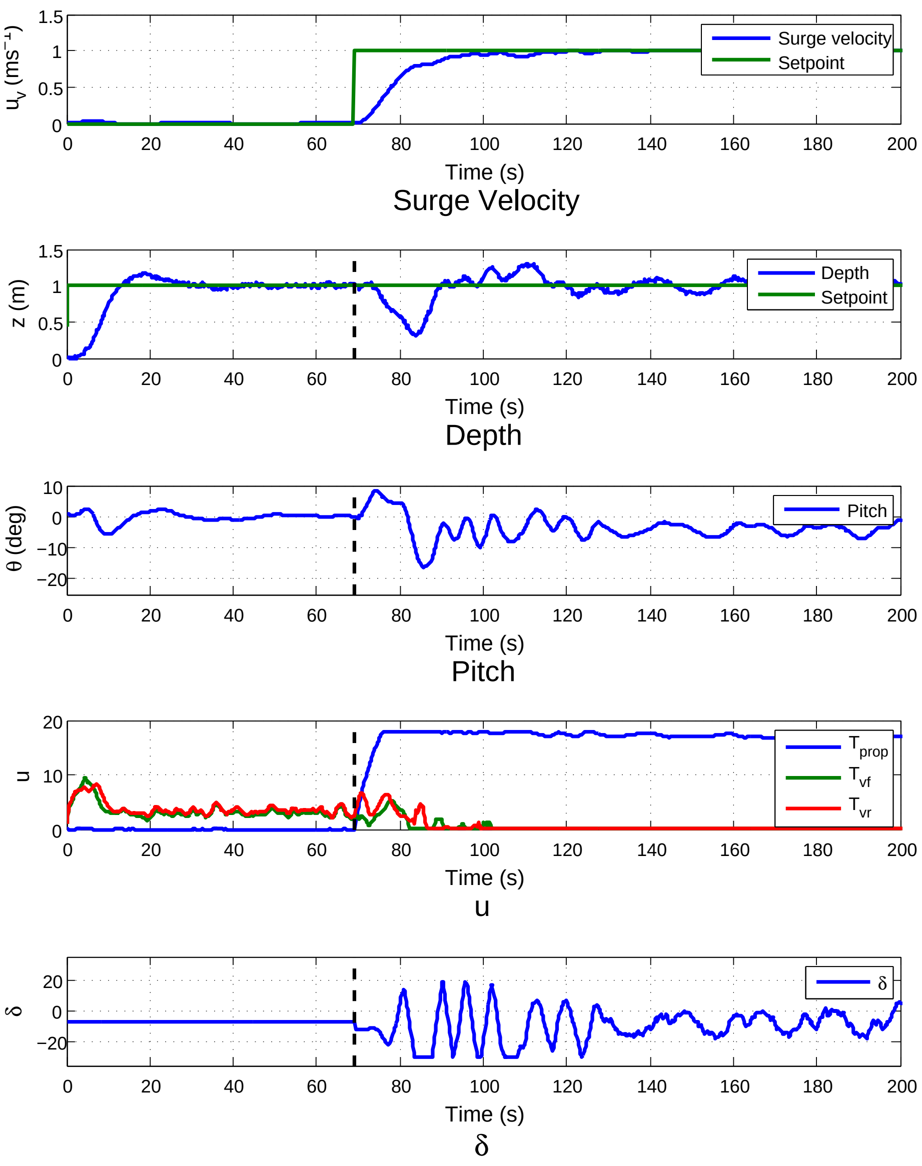


Figure 8.9: Experimental results of the over-actuated surge velocity and depth controller. One depth set-point of 1.0 m is used with an initial surge velocity set-points of 0.0 m/s, then at 70 seconds the surge velocity set-point is changed to 1.0 m/s.

However, it was noted that the thrusters were being utilised at high surge velocities which is both unnecessary and inefficient.

To solve this problem a method, defined in this work as the control authority, has been introduced. This method designates the control surfaces as the primary actuator set for depth control so that the thrusters are only utilised when the control surfaces require support in order to reduce the depth error. The control authority was achieved by modifying the cost function, within the MPC algorithm, so that the use of the thrusters is penalised at surge velocities greater than zero. This has been shown to work both in simulation and experimentally, with a weighting scalar providing a means to tune the rate of actuator transition.

The experimental results used to verify the controller design show great promise, however, further development is required. The controller did prove to be stable at all the surge velocities that the vehicle was tested at though it oscillated in pitch and depth when operating at higher speeds. These problems are common to both the over-actuated and flight-style controllers and are primarily caused by the inaccurate estimation of the vehicle's surge velocity. The controller was not tested at very low speeds due to limitations in the propeller control electronics.

The experimental results of the over-actuated controller operating at low speeds are better than those of the depth and pitch controller developed in Chapter 6. The primary difference is that the payload of the vehicle has been altered between these two tests. The buoyancy force of the AUV, when tested with the over-actuated controller, was 6.27 N in comparison to 3.35 N when the depth and pitch controller was tested. This significant increase in buoyancy results in very low overshoot values typically less than 0.14 m.

Another interesting comparison, between the depth and pitch controller and the over-actuated controller, is that the u thruster signals appear to be smoother when using the latter controller. This can be attributed to several factors; the observer has been tuned for each output, the controller outputs are depth and surge velocity, and the sampling rate has been halved. All three of these factors improve the signal noise ratio of the feedback and therefore the controller output, u, is less influenced by measurement noise.

A significant disadvantage of this controller, as has been seen with the flight-style controller too, is the disturbance to the depth tracking when accelerating in surge velocity. The coupling between surge and heave velocities of the vehicle is not adequately captured within the linearised state-space model. Thus, as the vehicle is accelerates in surge velocity, the estimated future trajectory of the vehicle will have significant inaccuracies, in particular towards the end of the prediction. These inaccuracies prevent the controller from adjusting the control surface and pitch angles to suitable values before accelerating in surge.

To remove, or at least reduce, these disturbances would require a different linearised model for use within the TV-MPC controller. Such a model must accurately approximate the coupling between the surge velocity and the rest of the system dynamics. To achieve this may require the use of a non-linear model predictive control (NMPC) algorithm rather than the TV-MPC algorithm. This may prove more computationally expensive than the already expensive TV-MPC algorithm, however it is worth investigating.

Chapter 9

Conclusions

In this chapter the conclusions from this Thesis will be presented and future research directions discussed. The first half is focused on the Delphin2 AUV, specifically the mechanical, electrical and software systems, and the second half focuses on the main contributions of this work, the MPC low-level controllers.

9.1 Delphin2 AUV

The Delphin2 AUV has been successfully designed and built at the University of Southampton, and is now a fully functional AUV. Over the past year, it has proven to be a reliable research platform that can be easily transported and tested in a wide variety of locations. Without the ability to experimentally evaluate the different components of this Thesis, many of the key contributions would most likely have been missed.

The vehicle typically weighs 54 kg, is 1.96 m long and has an operational endurance of up to 8 hours before the battery requires recharging. When dismantled, it can easily fit into a medium-sized hatchback car along with two people and its support equipment. The vehicle has operated at depths up to 14 m, therefore the verification of its 50 metre depth rating has yet to be achieved. The reason for this is that there has not been any necessity for the vehicle to dive any deeper and so the risk associated with diving to 50 m greatly exceeds any benefits.

Although not presented in this work, five of the six degrees of freedom have been successfully controlled, with the exception being roll. This degree of freedom is controllable, as all four control surface angles can be independently operated, however, there has never been a necessity to control roll. The heading and navigation (x, y) controllers are fully functional although they do require further development to improve their performance. The ability to transit at relatively high speeds and slow down to a hover, all whilst maintaining good depth control is one of the key contributions from this work and will be discussed in more detail later in this chapter.

In order to evaluate the competence of the Delphin2 AUV at performing hybrid AUV/ROV missions, as was the original concept, it was proposed that the vehicle be tasked with collecting scientific data that other AUVs would struggle to collect. To minimise risk it was decided that these missions occur in a large inland fresh water lake, rather than in the sea. Eventually, a mission statement was created in collaboration with the Agri-Food Biosciences Institute (AFBI) in Northern Ireland. The proposal involved the use of the Delphin2 AUV to perform a video survey of the bottom of Lower Lough Erne. This data was required to help the AFBI scientists investigate the population and coverage of Zebra mussels within the Lough. These fresh water shellfish are not native to Ireland yet have thrived in Lough Erne since they first arrived in the mid 1990s, see [102]. The primary difficulty with performing this mission was the poor water visibility which was typically less than 2 m.

In order to record video data of sufficient quality, the Delphin2 AUV was required to track the

Lough bottom at altitudes between 0.5 and 1.0 m, at a surge velocity of approximately 0.35 m/s. This velocity is within the transitional zone and therefore the tunnel thrusters and control surfaces were utilised simultaneously. The missions were conducted over two weeks and resulted in several hours worth of high quality data, showing that the mussels were more prevalent in shallower water depths than in the deeper and darker parts of the Lough.

Given the success of the Zebra mussel surveys, a second mission was performed, this time utilizing the Delphin2 AUV's ability to be operated as an ROV. This involved the AUV being used to capture close range video footage of eels that had been captured within a fyke net. Although some implementation issues were encountered, due to the bandwidth constraints of the umbilical tether, footage of an eel was successfully captured. More information on these science missions, including some of the video frames, can be found in Appendix B and on the BBC website, [103] and [104].

9.1.1 Vehicle Development

Before planning further research, it is recommended that several upgrades or modifications be made to the Delphin2 AUV. These should improve both the vehicle performance and the quality of the data it can collect, thus expanding the scientific capabilities of the vehicle and increasing the opportunity of collaborative research.

One of the key constraints on the Delphin2 AUV is the lack of an accurate speed measurement. This not only effects the low-level controller performance, as has been shown in Chapters 7 and 8, but also results in large navigation errors. In the worst case scenario these navigation errors could cause the vehicle to get lost, stuck or inadvertently enter a restricted area. Without accurate positional information, the quality of the scientific data that the AUV collects could be severely hindered.

For these reasons it is recommend that the Delphin2 AUV is lengthened and either a doppler velocity log (DVL) or an acoustic doppler current profiler (ADCP), be fitted to the vehicle. As will be discussed in the next section, the future use of the Delphin2 AUV may include operations in areas where strong tidal currents exist. In order to both successfully control and navigate the AUV, an ADCP would be the better choice as both fluid velocity and ground velocity can be measured. The physical size of the sensor is one of the key constraints, however, as the vehicle is unlikely to operate in deep water then the bottom track range need not exceed the vehicle's rating of 50 metres.

To further improve the navigational accuracy of the vehicle, a north seeking gyroscope should be integrated with the heading measurement from the compass sensor. This should reduce the effect of local variations of magnetic north due to the presence of large ferrous objects, for example, ships or certain geological minerals such as iron ore. The electronics for the rate gyroscope sensor have recently been finished, therefore it is only the development of a suitable algorithm that is required.

To improve controllability of the vehicle it is suggested that the motor controllers for the through-body tunnel thrusters and the rear propeller motor are upgraded. The motor controller that is used for controlling the thrusters has several limitations including; poor low speed thruster control, a slow serial communications protocol, and it is physically too large. To upgrade to a different controller may require integration of several single channel controllers with a micro-controller acting as the interface between them and the PC. The motor controller that is used to control the propeller motor is normally used for remote control boats. There is therefore no speed control, nor feedback, and there is a large dead-band that makes slow surge velocity control difficult. There are many commercially available controllers that would be suitable for this. It is also recommended that the motor and gearbox for the rear propeller are replaced as both have sustained a significant amount of wear.

The reliability of the cabling between the pressure vessel and the thrusters is poor. In 2010, all four thrusters failed within a few days of each other, all with the same fault. The thruster cables did not have adequate strain relief and as a result the metal wires snapped within their sheaths. Since then a number of temporary solutions have been used to fix these cables, with various degrees of success. A thorough examination of the fault is required and a permanent solution developed. In the event that new thrusters are purchased, it is recommended that the smaller 50 mm diameter thrusters be considered to replace the horizontal thrusters so as to refine the controllability of the heading control.

9.1.2 Future Research

The Delphin2 AUV has great potential as a research platform for both academic research and novel application projects. There is still a significant amount of research to be conducted on both the low-level control systems as well as higher level AI. The future development of the low-level controllers will be discussed in the next section.

A low-cost area of AI research is that of high-level planning. This is almost entirely software based development and so the material costs are low. Currently the mission planning on Delphin2 AUV is very linear, such that once a particular task finishes it moves onto the next predefined task, unless it has detected a fault upon which it returns to base. An active area of research that could improve the Delphin2 AUVs capabilities and efficiency is adaptive mission planning.

The basic idea of this is to prioritise the various mission tasks before starting the mission. Then using the current location, battery life, sensor suite etc, the mission planner decides what tasks it can perform and when to perform them. This algorithm is run continuously throughout the mission and so the plan may vary significantly with time so as to collect the optimum amount of data. By using a well designed adaptive planner the quality of the data can be improved whilst reducing the time in the water. Research is currently under-way at the University of Birmingham [105], with the intention to experimentally verify the algorithms in 2013 using the Delphin2 AUV.

The Delphin2 AUV is relatively ‘blind’ using its current sensor suite. This is due to the scanning-sonar having a relatively low resolution and that the water quality is rarely good enough to make use of the camera images. In order to start interacting with the subsea environment, the addition of new sensors, such as side-scan or multi-beam sonar systems, is required.

The high definition (HD) sonar systems that are commercially available, for example [106], would be ideally suited for use on-board the Delphin2 AUV. These sensors provide images of the seabed in great detail, but require that the target is within 5 metres range. For objects that require this level of detail, such as hydrothermal vents or subsea structures, it is unlikely that the majority of AUVs would be able to successfully get close enough, and at a slow enough speed, to capture high quality data.

Although a HD sonar system could be quickly integrated onto the Delphin2 AUV, to process and make use of the data in real-time would require a significant amount of research and effort. Processing of sonar images to find underwater objects, such as mines, is a current area of research around the world, see [107] and [108]. However, most of this research focuses on the detection of objects from a distance.

For the Delphin2 AUV to operate within a five metre range of the subsea structures would require the real-time processing of the sonar data. This processed data could be used to make intelligent decisions regarding whether the object is actually interesting, and more importantly, how to navigate in a highly complex environment such as a canyon.

The operational capability of the Delphin2 AUV has already been proven, Appendix B, however there are many different applications that it could perform. The key feature of Delphin2 is its ability to slow and operate at very close range to a subsea object or the sea floor. It is therefore suitable for inspection-type missions, especially those where there are multiple areas of interest within a large operational area.

An example of such a mission is the continuous monitoring of the foundations of offshore wind turbines. Due to the flow around these turbines, primarily due to tidal currents, scour can occur on both the foundations and the cabling between the different turbines, [109] and [110]. This scouring can cause the hydraulic loading on the structures to increase and significantly reduce the stability of the turbine foundations below safe levels. The Delphin2 AUV, equipped with a side-scan sonar, could continuously swim between each turbine and conduct a survey of each foundation. Then, using satellite communications or a data link on the turbines, the sonar data could be transmitted back to a central headquarters for processing. On installation of the the turbines a dedicated AUV docking station could be installed that was powered by the turbines. This could then be used to charge the Delphin2 AUVs batteries between missions. An example of an AUV docking station that enables underwater battery charging can be found in [111].

9.2 Model Predictive Control

In this Thesis the suitability of using the MPC algorithm as a low-level controller has been experimentally verified. Three control problems of increasing complexity were addressed. First, a depth and pitch controller was developed for low speed hovering control. Then a controller was designed for the highly non-linear flight-style problem before finally designing a controller for the over-actuated control problem. In this section, an overview of each controller along with their appropriate conclusions and contributions shall be given.

9.2.1 Low Speed Hovering Control

The first control problem tasked the MPC algorithm with controlling the AUV's depth, whilst maintaining a pitch angle of zero degrees, during low speed operations. The system is relatively linear and has two inputs; the front and rear vertical through-body tunnel thrusters.

The MPC algorithm was implemented in its standard form, as presented in Chapter 3, and utilized one linear state-space model within the controller design. The majority of system components were easily linearised and, by using the inverse of the thruster model, the actuator non-linearities were also avoided. Absolute and rate-of-change constraints were applied to the system inputs so as to reduce peak electrical power consumption and to help avoid specific actuator non-linearities.

The overall performance of this controller was excellent, with fast and smooth convergence to the desired set-points. The controller was tuned using a brute-force technique, whereby a large number of simulations were performed so that a wide range of controller parameter combinations could be evaluated. By post-processing each of the simulation results, the variation of each controller parameter could be directly compared to the different performance attributes, such as depth overshoot or pitch settling time.

It was shown that, in general, a longer prediction horizon (N_p) gave better overall system performance, while too short of a prediction horizon caused the system to become unstable. In comparison, the other two tuning parameters (N_c and r_w) where less intuitive to tune. The control horizon appears to have the least influence on controller performance whilst it has been shown that the weighting scalar r_w does have a significant effect. This scalar is used as a weighting component within the cost function so that a high r_w value will penalise large values within the ΔU vector. Consequently, smaller r_w values cause the system to react more aggressively by producing larger values within the ΔU vector. Generally the range of r_w values that gave good system performance increased with increasing length of prediction horizon. Three sets of tuning parameters were chosen after evaluating the simulations results. These sets were designed to provide; fast but potentially unstable performance, slow but stable performance, and a set that is approximately in the middle of the other two.

Using these three sets of tuning parameters, a further series of simulations were performed that examined the effect of deliberately introducing errors within the linear state-space model. The overall conclusion is that the controller is relatively insensitive to modelling inaccuracies. Controller performance did not degrade significantly when the mass and inertia errors were within $\pm 35\%$ and $\pm 50\%$ of their original values respectively, while less dominant terms, like the damping coefficients and restoring moments, could be completely eliminated from the model. The controller's insensitivity to modelling errors is highly beneficial. Designers of similar controllers, for AUVs or ROVs, can therefore have confidence that estimated model parameters should provide adequate system performance, and so experimental or simulation techniques used to provide the information to build the model could be avoided. This should reduce development time and costs.

Realistic levels of measurement noise were added to the depth and pitch feedback signals in simulation. An estimate of the state-variable vector was calculated by directly differentiating the various feedback signals. After several attempts it was noted that it was not possible to perform a full 60 second simulation using this form of feedback as numerical errors caused to program to crash. To address this problem, a polynomial type filter was introduced and implemented. This filter improved controller performance considerably, enabling the simulation to complete without crashing, however, the actual performance was still not of a suitably high standard.

The augmented state-space model, used within the controller design, was then used to design a Luenberger observer. This observer used the measured outputs, depth and pitch, to provide an estimate of the state-variable vector that could be used by the MPC controller as feedback. The observer feedback matrix was calculated using a Linear-quadratic state-feedback regulator function (dlqr) within MATLAB. This placed the eigenvalues of the observer on the left hand side of the complex plane so as to ensure observer stability. The performance of the controller and observer, with measurement noise added to the feedback signals, was excellent. There was still some degradation in performance but it was within acceptable limits. From these results, it can be concluded that an observer is an essential component when implementing the MPC algorithm on a real system.

The experimental results were not as good as the simulated results primarily because of the vehicle's payload at the time of testing. When the tests were conducted the AUV had a buoyancy of 3.3 N, compared to 5.0 N used by the simulations. This reduction in force, that acts against the thruster forces, meant that the AUV decelerated more slowly in heave when tested experimentally compared to in simulation. As a result, the vehicle overshot the depth set-points quite significantly. After the overshoot the AUV converged quickly to each depth set-point with a zero steady-state error. The pitch control performed adequately well and maintained a pitch angle close to the demanded zero degrees, however, the vehicle's low buoyancy did hinder the performance of the controller. In order for the AUV to achieve a level pitch angle, different magnitudes of thrust between the front and back thrusters are required. Every time both of the thruster demands reached their minimum absolute constraints, the difference between the thrusters was reset to zero.

When tested experientially all three tuning parameter sets were stable and provided excellent performance. The more aggressive controller set, with a short prediction horizon and low r_w scalar, converged more quickly to each set-point. Using the more stable parameters, the thruster demands were lower and as a result the convergence time was longer, however, the controller was less sensitive to measurement noise and as a result it experienced smaller disturbances to the depth tracking in comparison to the other sets.

From these results it is believed that the MPC algorithm could be successfully implemented, with minimal effort, on most UUVs that operate at low speeds using thrusters. As the controller has been shown to be relatively insensitive to modelling inaccuracies, the development of a suitable state-space model should not prove too difficult however the choice of the initial tuning parameters (N_p, N_c, and r_w) may be. By using the method described in Chapter 6, an estimate of the minimum length of prediction horizon can be calculated. For the Delphin2 AUV suitable control parameters can be estimated by using; an N_p value that is double that of the estimated minimum length for the prediction horizon, and N_c and r_w values that are equal to approximately 10% of the chosen N_p value. Further experimentation with different vehicles is required to verify this recommendation.

9.2.2 Flight-Style Control

Having shown with the hover-mode depth and pitch controller that the MPC algorithm has potential at controlling AUVs, a more challenging control problem was introduced. This was to control the depth and surge velocity of the Delphin2 AUV using the flight-style method. In order to maintain depth stability using this method the vehicle is required to operate with surge velocities greater than 0.4 m/s. The flight-style system contains much stronger non-linearities than the low-speed depth and pitch system.

The depth and pitch controller used one linear state-space model for all operating conditions. This proved sufficient as the system dynamics, when operating in hover-mode, could be approximated quite accurately as linear. For flight-style, the use of one linear model would be insufficient to accurately approximate the system dynamics throughout its operational range.

To handle the non-linearities of the flight-style system a time-variant MPC algorithm was proposed. This involved developing a linear time-variant state-space model of the flight-style AUV, within which the components of the model are linearised at the beginning of each sample using the latest state-feedback. Therefore terms within the model, such as the lift coefficient of the hull, are linearised using the latest surge velocity or other state-variable.

It was first shown in simulation that the controller did provide reasonable performance with convergence to both the depth and surge velocity set-points, but, the depth response was slow and had an unacceptably large overshoot. This slow response was due to the MPC algorithm not acting aggressively enough and instead relied on the natural system dynamics rather than using the system inputs.

An investigation showed that this was due to the low damping depth and pitch coefficients that the TV-MPC algorithm was calculating. This resulted in the respective system eigenvalues lying very near to the real axis of the complex plane. By significantly increasing the values of both the depth and pitch damping terms, by using gains placed within the state-space model, these eigenvalues were moved further to the left of the real axis and the depth control performance was greatly improved.

With the inner gains applied to the damping terms the depth control had effectively no overshoot and converged quickly to the appropriate set-points by using large fluctuations in control surface angles. The improvements in depth control did come at the slight sacrifice of surge velocity control. Here the convergence was slower than before and a small overshoot was often experienced, however, as the surge velocity is not as critical as depth this degradation in performance was deemed acceptable.

One weakness of the TV-MPC algorithm was identified when the accelerations in surge velocity were large. If, for example, the AUV is transiting at a fixed depth and a step change is surge velocity set-point was applied, then a disturbance is observed on the depth tracking as the vehicle accelerates. When the acceleration in surge was negative the AUV lost depth (approached the surface) but when the accelerations were positive the AUV gained depth and so overshot the depth set-point.

The loss in depth when decelerating is not deemed to be critical, however, the increase in depth could cause the AUV to crash into the sea floor. These disturbances were not large, typically less than 0.2 m, and could be reduced by imposing tighter constraints on the ΔT_{prop} values. This however increased the convergence time to its desired surge velocity, therefore a compromise between the depth disturbances and surge accelerations must be met. This is at the discretion of the AUV operator however it should be noted that tightening the constraints will likely increase the computational load of the controller which is obviously undesirable.

As with the depth and pitch controller, an observer was required when measurement noise was added to the feedback signals. This was built using the same method as before but with slight modifications to improve the observer performance. The first modification involved using a different C matrix for the augmented model used by the observer. This matrix resulted in the new augmented model having three outputs; surge velocity and depth, as before, along with the addition of pitch. The inclusion of pitch feedback into the observer greatly improved the estimation of the state-variable vector. A second modification was made to the R_o matrix used to tune the observer, whereby different scalars were designated for each output. These scalars were chosen by trial and error by performing simulations of the MPC controller with realistic levels of measurement noise on the feedback signals. Performance of the controller did show slight degradation when tested with the observer however this was within acceptable limits.

The performance of the flight-style controller, when tested experimentally, was significantly worse than that achieved in simulation. The controller did converge to the surge velocity and depth set-points, however, its performance at accurately and smoothly tracking the depth set-point was relatively poor. Although the standard deviation of the depth error was typically less than 0.08 m, which is poor but acceptable, the standard deviation of the pitch angle was often greater than 2^o. These fluctuations in pitch are unacceptable as they will significantly degrade the quality of data collected by the AUV, in particular acoustic or optical measurements.

To investigate the cause of this degradation in performance, between the experimental and simulated results, further simulations were performed. One key parameter that was found to severely effect controller performance was the introduction of an error between the real surge velocity and that used to create the linear state-space model. By reducing the surge velocity used within the linearisation process by 5%, below the real value, a significant oscillation in pitch was observed which in turn introduced disturbances on the depth tracking. It was noted however that overestimating the surge velocity had a much less detrimental effect on the controller performance than underestimating it. The reason for this controller degradation, and asymmetry, is that if the surge velocity is underestimated then so too are the forces and moments produced by the vehicle and its control surfaces. As a result, if there is a slight disturbance, either real or from measurement noise, the controller reacts with an unnecessarily large response which causes the vehicle to overcompensate for the disturbance. The vehicle will continue to overcompensate and so constantly oscillate. By overestimating the force, the controller will produce smaller responses to any disturbances, therefore the vehicle will converge to its set-points but more slowly.

Given that there is no speed measurement available on the Delphin2 AUV, the errors between the actual and estimated surge velocities are likely to be large. The addition of an accurate speed sensor, such as a DVL or ADCP, should improve the performance of the controller significantly. However, even with a measurement of the surge velocity, it is recommended that the linearisation process use an inflated value of surge velocity during initial trials of the controller so as to improve the controller stability. This exaggeration in surge velocity can then be incrementally reduced to improve system performance whilst maintaining stability.

9.2.3 Transitional Control

Perhaps the most important contribution from this Thesis is the over-actuated controller, developed in Chapter 7. This controller can be described as an 'enabling component' for a hover capable AUV as it provides the vehicle with the ability to operate freely throughout the entire velocity range of the vehicle.

The mathematical model of the system, used to approximate the vehicle dynamics, is effectively the combination of the depth and pitch model and the flight-style model, developed in Chapters 6 and 7 respectively. Therefore at low speeds, the thrusters are used to produce the necessary forces and moments in order to manoeuvre the vehicle, whilst at higher speeds the rear propeller and control surfaces are used. The over-actuated controller must therefore be able to manipulate all four of these actuators in order to control both the depth and surge velocity of the Delphin2 AUV.

The initial design of the over-actuated controller resembled that of the flight-style controller developed in Chapter 7. It was a MIMO system that utilised the TV-MPC algorithm. The only significant difference was that the system model had changed. Without further modifications, the controller was tested in simulation and achieved the various surge velocity and depth set-points quickly and smoothly. However, it was noted that the thrusters were utilised when the vehicle was operating at high surge velocities. Due to the high power consumption of the thrusters, especially at high speeds, it was deemed inappropriate to use the thrusters whenever the control surfaces should be capable of independently controlling the vehicle.

Due to the design of the MPC algorithm, the cost function does not directly penalise the magnitude of the different inputs but instead their rate of change. Therefore a method, defined as the control authority, was implemented and designated the control surfaces as the primary actuator set for depth control. This method required an additional component to be added to the cost function that penalised the use of the thrusters. This penalty component is a function of the latest thrust values, the absolute surge velocity of the vehicle and a scalar gain. The magnitude of the scalar gain was chosen by trial and error using simulation results as feedback.

The over-actuated controller when tested experimentally and in simulation showed excellent performance. At low speeds, the vehicle performed similarly to the depth and pitch controller. The buoyancy force, when the over-actuated controller was tested, was significantly higher than when the depth and pitch controller was tested. This resulted in better depth control as the magnitude of the depth overshoots was substantially lower.

When operating at high speeds, the control authority ensured that the thrusters switched off and the control surfaces were used to control the depth of the vehicle. Unfortunately, when operating with just the control surfaces the controller exhibited similar performance to the flight style controller, such that the pitch and depth oscillated significantly. This has been attributed to the inaccurate estimation of the surge velocity and has been discussed earlier in the chapter.

The most important attribute of this controller is its ability to operate within the transitional zone. Experimental results have been produced that show the controller maintaining depth while accelerating from 0.0 m/s to 0.5 m/s. Over the period of approximately 50 seconds, the pitch of the vehicle is slowly decreased so as to increase the hull lift force. Once the depth error is low and the pitch angle of the vehicle is relatively steady, the thrusters switch off smoothly. Unfortunately, due to restrictions imposed by the propeller motor controller, it is not possible to smoothly operate at speeds below 0.4 m/s and so the algorithm's ability to maintain stability within the transitional zone cannot be fully verified. To operate at speed lower than 0.4 m/s the propeller can be pulsed on and off though this does not provide smooth operation.

9.2.4 Final Remarks

The Delphin2 AUV is now fully functional and has the appropriate control systems to operate the vehicle from a hover to above 1.0 m/s. The over-actuated controller functions very well at low speeds and transitions from using the thrusters to the control surfaces smoothly. However it does still have some limitations that need addressing, in particular the marginal instability at high speeds.

9.3 Future Research

In this section ideas for future research that will further develop this work are proposed. It should be noted that the future research regarding the Delphin2 AUV was presented earlier in this chapter.

9.3.1 Surge Velocity Estimation

One of the major limitations of the Delphin2 AUV, that has hindered the performance of the controllers developed in this work, is the poor estimation of the vehicle's surge velocity. Earlier in this chapter, physical modifications to the vehicle, that should improve the accuracy of the estimated surge velocity, have been proposed. In the absence of these modifications, an alternative method must be devised that will improve the surge velocity estimation.

In this work, the surge velocity was estimated using a non-linear mathematical model of the vehicle, the components of which are derived from mostly experimental data. The data used to model the propeller performance may contain significant inaccuracies and, when coupled with the controllability issues of the propeller motor, results in large uncertainties on the magnitude of thrust generated by the rear propeller. It is therefore believed that the development of a more accurate propeller model should yield better estimations of the surge velocity.

If it is first assumed that the rest of the model is sufficiently accurate, then it may be possible to use an iterative learning algorithm to converge upon a better estimation of the propeller dynamics. To implement this would require feedback of either the vehicle's velocity or position, neither of which are available underwater. The only sensor that can provide this feedback is the GPS sensor but this only works on the surface. As was discussed in Section 5.5, the hydrodynamic drag acting on the vehicle is greatly increased when operating on the surface. Therefore performing iterative learning on the surface is unlikely to yield a better model for when the vehicle is submerged.

One possible option is to perform a series of underwater manoeuvres with the vehicle. By starting and finishing each manoeuvre from the surface, the learning algorithm can receive feedback of the start and finish locations from the GPS sensor. By designing the feedback law correctly, the learning algorithm should be able to converge onto a better estimate of the vehicle's navigation, which in turn should yield a more accurate estimate of the propeller dynamics and consequently the vehicle's surge velocity. It should be noted that the manoeuvres used in this process should be kept as simple as possible, starting with tests such as; dive, go straight for two minutes at a fixed propeller set-point, then return to the surface.

As the vehicle model is non-linear and has many coupled terms, only certain model parameters should be 'learnt.' Attempting to simultaneously learn multiple components of the model could cause the learning algorithm to converge to an incorrect set of values. The identification of these parameters should be examined at the same times as the rest of this research.

9.3.2 Six Degrees of Freedom Control

The final controller developed in this Thesis was the over-actuated surge velocity and depth controller for the Delphin2 AUV. Heading and sway control have not been addressed in this work but are required by the vehicle, especially heading control. Currently the Delphin2 AUV uses a gain-scheduled PI controller for heading control however performance is generally poor. There is no closed loop sway control implemented on the Delphin2 AUV at the time of writing this Thesis.

Given that the vehicle can be approximated as a body of rotation, then the heading dynamics can be approximated as the pitch dynamics (without the restoring moment) and the sway dynamics can be approximated as the heave dynamics. These could then be included within one large time-variant state-space model and implemented in the same manner as the over-actuated surge velocity and depth controller.

As with surge velocity, there is no measurement of the sway velocity of Delphin2. Therefore, the controller must receive feedback from a non-linear mathematical model. Given that sway control is only likely to be implemented when operating in hover mode, and that it is normally used to adjust the vehicle relative to a certain object, this form of feedback may prove adequate. For precise sway velocity control a velocity sensor will be required.

Assuming that the rate-gyroscope sensor is integrated onto the Delphin2 AUV then the heading controller will have measured feedback of both the yaw rate and yaw (heading) angle. One of the foreseeable difficulties with implementing the heading controller is the calculation of the heading error within the controller. This is due to the non-linear over-flow type feedback signal from the compass. One possible method would be to calculate the error outside of the MPC algorithm and then have the heading set-point fixed at zero degrees. Further thought is also required as to how a linear observer could handle the non-linear compass data.

Another issue that applies to the design of both the heading and sway controllers is how the non-linear dead-zone of the thruster motors is handled. This dead-zone, if not handled correctly, can cause cyclic oscillations about the heading set-point as the controller effectively acts in a 'bang-bang' manner. Therefore, the constraints imposed on the horizontal thrusters need to vary in accordance with the current sign and magnitude of the heading error. As discussed earlier in this chapter, a new motor controller (hardware) may be required to achieve fine low-speed heading control.

9.3.3 Depth Overshoot Reduction

The controllers developed in this work have been tuned so as to minimise the magnitude of the depth overshoot. Rather than rely on the tuning process, two alternative adaptations of the MPC controller could be utilised to minimise the overshoot; imposing a state constraint on the depth state, or forcing the controller to follow a predefined trajectory by controlling the heave velocity rather than the depth.

State constraints can be implemented using this form of MPC. Rather than using hard constraints, like those that have been imposed on the inputs, soft constraints should be used. These can be violated but at an expense within the cost function. By scheduling these constraints to be the depth set-point plus the acceptable tolerance then the controller should, in theory, avoid overshooting by using larger values within the Δu vector. The computational expense of imposing these constraints also needs investigating.

Controlling the heave velocity, so as to track a predefined trajectory to each depth set-point, may prove to be less computationally expensive and provide smoother control, in comparison to imposing state constraints. The method of calculating an optimal trajectory requires research. If the gradient of the chosen trajectory is beyond the system's capabilities it may lead to instability issues. Likewise, if the gradient is too low, then the vehicle may take too long to reach each set-point.

9.3.4 Fault Detection and Handling

The Delphin2 AUV has some relatively basic fault detection algorithms built into the control software. These check the performance of the different actuators to ensure that their feedback corresponds correctly to their inputs.

For vehicles that are over-actuated, such as the Delphin2 AUV, it may be possible to maintain stability without the use of all the actuators. Therefore, in the event that one of the actuators were to fail, for example a thruster motor burns out, then by updating the input constraints the faulty actuator could be eliminated from the model. By doing so, the controller will find the optimal solution to achieve its set-points using the available actuators.

This should be relatively easy to implement, however, the controllability of the vehicle needs investigation. For example, if a horizontal control surface were to jam at 30^o then the controller must realise that it should not exceed a certain surge velocity as the vehicle may dive uncontrollably.

9.3.5 Universal AUV Controller

One of the reasons for choosing the MPC algorithm, to control the Delphin2 AUV, was because the system dynamics are an integral part of the controller design. This feature means that, although tuning is required, it is not as critical to system performance as with a classical controller. It is proposed that a TV-MPC controller framework could be built for AUVs that, given some geometric and performance data, could be quickly and successfully implemented on any AUV. Here this concept is called the universal AUV controller.

The key component to the TV-MPC algorithm is the mathematical model of the vehicle. If the universal AUV controller is designed for use by torpedo shaped vehicles only, at least initially, then the derivation of a sufficiently accurate model should not be too difficult as:

- Lift and drag coefficients are non-dimensional and so can be scaled to the specific vehicles dimensions.
- The rigid-body mass and inertia values can be calculated using the CAD program used to design the vehicle.
- Performance data is normally available for off-the-shelf thrusters and motors.

If designed correctly, such a controller could prove extremely useful especially for smaller research groups that have specialities outside of low-level control. Eventually, the program could be developed to encompass all UUVs and UAVs that in turn could lead to standardisation of software throughout the unmanned vehicle industries.

9.3.6 Embedded Low-Level Control

A similar concept to the Universal AUV controller, is the use of a standardised electronics system that is dedicated to the low-level control of unmanned vehicles. In [112], a standardised computer system has been developed that can be easily configured for AUVs, UAVs or ASVs. This however is an all encompassing package that runs the entire control system. What is envisioned here is the development of an embedded low-level controller, such that the low-level control algorithms are run on a separate electronics board to the rest of the vehicle's control systems.

The majority of the computers that are used on-board unmanned vehicles are simply small form-factor PCs. These computers use central processing units (CPUs) that have been designed to perform a wide range of different types of calculations, all at average speeds. As low-level controllers are often heavily dependant on matrix operations, especially the MPC algorithm, choosing a board with a CPU that is specifically optimized for this type of calculation should improve system efficiency. Therefore, by opting to use an embedded low-level control system, the controller sampling rates and the electrical efficiency of the system could be improved.

How such a controller could be effectively and reliably integrated into the rest of the control system needs further research, as does the type of electronics and compatible software. By combining the universal AUV controller with embedded low-level control electronics, AUV developers could effectively buy a COTS low-level controller.

References

[1] SNAME. *Nomenclature for Treating the Motion of a Submerged Body Through a Fluid*, volume Technical and Research Bulletin No. 1-5. The Society of Naval Architects and Marine Engineers (SNAME)., 1950.

[2] S.M. Sharkh, S.R. Turnock, and A.W Hughes. Design and performance of an electric tip-driven thruster. *Proceedings of the Institution of Mechanical Engineers. Part M: Journal of Engineering for the Maritime Environment*, 217 (3):133–147, 2003.

[3] A.R. Palmer, G.E. Hearn, and P. Stevenson. A theoretical approach to facilitating transition phase motion in a positively buoyant autonomous underwater vehicle. *Transactions of RINA, Part A - International Journal of Maritime Engineering (IJME)*, 151, part A-3, 2009.

[4] A.D. Rogers, P.A. Tyler, D.P. Connelly, J.T. Copley, R. James, R.D. Larter, K. Linse, R.A. Mills, A.N. Garabato, R.D. Pancost, A.D. Pearce, N.V.C Polunin, C.R. German, T. Shank, P.H. Boersch-Supan, B.J. Alker, A. Aquilina, S.A. Bennett, A. Clarke, R.J.J. Dinley, A.G.C. Graham, D.R.H. Green, J.A. Hawkes, L. Hepburn, A. Hilario, V.A.I. Huvenne, L. Marsh, E. Ramirez-Llodra, W.D.K. Reid, C.N. Roterman, C.J. Sweeting, S. Thatje, and K. Zwirglmaier. The Discovery of New Deep-Sea Hydrothermal Vent Communities in the Southern Ocean and Implications for Biogeography. *PLoS Biol*, 10(1):e1001234, Jan. 2012.

[5] J. A. Huber, H. V. Cantin, S. M. Huse, D. B. Mark Welch, M. L. Sogin, and D. A. Butterfield. Isolated communities of Epsilonproteobacteria in hydrothermal vent fluids of the Mariana Arc seamounts. *FEMS Microbiology Ecology*, 73:538549, 2010.

[6] J.F. Holden, E.T. Baker, R.W. Embley, S.R. Hammond, T.M. Shank, S.L. Walker, and S.M. White. GALREX 2011: Initial Results of the 2011 NOAA Ocean Exploration Cruise to the Galpagos Rift Using Interactive Telepresence Technology. In *American Geophysical Union, Fall Meeting 2011*, 2011.

[7] P. Newman. Remotely Operated Vehicles involved in the Deepwater Horizon response. Online document: http:// www.douglas-westwood.com/ files/ files/ 583-Deepwater%20Horizon%20ROV%20%20AUV%20PN.pdf.

[8] B. J. Buckham, F. R. Driscoll, M. Nahon, and B. Radanovic. Three Dimensional Dynamics Simulation of Slack Tether Motion in an ROV system. In *The Thirteenth International Offshore and Polar Engineering Conference*. The International Society of Offshore and Polar Engineers, 2003.

[9] K.Q. Zhu, H.Y. Zhu, Y.S. Zhang, J. Gao, and G.P. Miao. A Multi-Body Space-Coupled Motion Simulation for A Deep-Sea Tethered Remotely Operated Vehicle. *Journal of Hydrodynamics, Ser. B*, 20(2):210 – 215, 2008.

[10] Z. Feng and R. Allen. Evaluation of the effects of the communication cable on the dynamics of an underwater flight vehicle. *Ocean Engineering*, 31(89):1019 – 1035, 2004.

[11] S. McPhail. Autosub6000: A Deep Diving Long Range AUV. *Journal of Bionic Engineering*, 6:55–62, 2009.

[12] B. Kieft R. McEwen M. Godin B. Hobson, J. G. Bellingham and Y. Zhang. Tethys-Class Long Range AUVs - Extending the Endurance of Propeller-Driven Cruising AUVs from Days to Weeks. In *IEEE AUV2012*, Southampton, UK, September 2012.

[13] R.P. Stokey, A. Roup, C. von Alt, B. Allen, N. Forrester, T. Austin, R. Goldsborough, M. Purcell, F. Jaffre, G. Packard, and A. Kukulya. Development of the REMUS 600 autonomous underwater vehicle. In *OCEANS, 2005. Proceedings of MTS/IEEE*, pages 1301–1304 Vol. 2, September 2005.

[14] D.P. Connelly, J.T. Copley, B.J. Murton, K. Stansfield, P.A. Tyler, C.R German, C.L. Van Dover, D. Amon, M.E. Furlong, N. Grindlay, N. Hayman, V. Hhnerbach, M. Judge, T. Le Bas, S. McPhail, A. Meier, K. Nakamura, V. Nye, M. Pebody, R.B. Pedersen, S. Plouviez, C. Sands, R.C. Searle, P. Stevenson, S. Tawsand, and S. Wilcox. Hydrothermal vent fields and chemosynthetic biota on the world's deepest seafloor spreading centre. *Nature Communications*, 620, January 2012.

[15] Y. Zhang, M.A. Godin, J.G. Bellingham, and J.P. Ryan. Using an Autonomous Underwater Vehicle to Track a Coastal Upwelling Front. *Oceanic Engineering, IEEE Journal of*, 37(3):338 –347, July 2012.

[16] R. Burcher and L. Rydill. *Concepts in submarine design.* Cambridge Ocean Technology Series, 1994.

[17] J. Liu, M.E. Furlong, A. Palmer, A.B. Phillips, S.R. Turnock, and S.M. Sharkh. Design and Control of a Flight-Style AUV with Hovering Capability. In *Proceedings of the International Symposium on Unmanned Untethered Submersible Technology (UUST 2009), Durham, New Hampshire, 23-26 August 2009*, page [9p]. Autonomous Undersea Systems Institute (AUSI), 2009. Proceedings issued on CDROM.

[18] W. J. Herr and K. Collins. MUST: A Large Versatile AUV Technology Testbed System. In *Underseas Defense '87*, San Diego Ca., October 1987.

[19] Atlas Elektronik. SeaCat configuration. Website: http://auvac.org/configurations/view/116, September 2012.

[20] T.L. Curtis, D. Perrault, C. Williams, and N. Bose. C-SCOUT: a general-purpose AUV for systems research. In *Proceedings of the International Symposium on Underwater Technology*, pages pp.73–77, 2000.

[21] L.V. Steenson, A.B. Phillips, E. Rogers, M.E. Furlong, and S.R. Turnock. Preliminary results of a hover capable AUV attempting transitional flight. In *Unmanned Untethered Submersible Technology (UUST)*, 2011a.

[22] A.F. Molland and S.R. Turnock. *Marine Rudders and Control Surfaces: Principles, Data, Design and Applications.* Butterworth-Heinemann, 2007.

[23] M.E. Furlong, S.D. McPhail, and P. Stevenson. A Concept Design for an Ultra-Long-Range Survey Class AUV. In *OCEANS 2007*, Aberdeen, June 2007.

[24] A.R. Palmer. *Analysis of the propulsion and manoeuvring characteristics of survey-style AUVs and the development of a multi-purpose AUV.* PhD thesis, University of Southampton, School of Engineering, Science and Mathematics, 2009.

[25] M.E. Furlong. *System identification of the hydrodynamic characteristics of underwater vehicles.* PhD thesis, University of Southampton, 2005.

[26] A.B. Phillips. *Simulations of a self propelled autonomous underwater vehicle.* PhD thesis, University of Southampton, April 2010.

[27] NATO Underwater Research Centre. Student Autonomous Underwater Challenge - Europe (SAUC-E). Website: http://sauc-europe.org/, October 2012.

[28] OCEANS 2025. The NERC Marine Centres' Strategic Research Programme 2007-2012: Theme 8: Technology Development. Online document: http://www.oceans2025.org/PDFs/Oceans_2025_Theme_8.pdf.

[29] G. Kim and S. Park. A Wireless Remotely Operated Vehicle Using Acoustic Communication. *Marine Technology Society Journal*, 46:44–49, 2012.

[30] I.C. Rust and H.H. Asada. A dual-use visible light approach to integrated communication and localization of underwater robots with application to non-destructive nuclear reactor inspection. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2445 –2450, May 2012.

[31] E. Omerdic, D. Toal, G. Dooly, L. Miller, and J. Coleman. Smart ROV LATIS in Action: Sea Trials. In *IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles (NGCUV)*, 2012.

[32] D. Yoerger, J. Newman, and J.-J. Slotine. Supervisory control system for the JASON ROV. *Oceanic Engineering, IEEE Journal of*, 11(3):392 – 400, July 1986.

[33] N.J. Cooke. Human Factors of Remotely Operated Vehicles. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 50, pages 166–169, October 2006.

[34] D.R. Yoerger, J.G. Cooke, and J.-J.E. Slotine. The influence of thruster dynamics on underwater vehicle behavior and their incorporation into control system design. *Oceanic Engineering, IEEE Journal of*, 15(3):167 –178, July 1990.

[35] L.E. Bird, A. Sherman, and J. Ryan. Development of an Active, Large Volume, Discrete Seawater Sampler for Autonomous Underwater Vehicles. In *OCEANS 2007*, pages 1 –5, Oct. 2007.

[36] W. Naeem, R. Sutton, J. Chudley, F. R. Dalgleish, and S. Tetlow. An online genetic algorithm based model predictive control autopilot design with experimental verification. *International Journal of Control*, 78(14):1076–1090, 2005.

[37] R. Stokey, T. Austin, B. Allen, N. Forrester, E. Gifford, R. Goldsborough, G. Packard, M. Purcell, and C. von Alt. Very shallow water mine countermeasures using the REMUS AUV: a practical approach yielding accurate results. In *OCEANS, 2001. MTS/IEEE Conference and Exhibition*, volume 1, pages 149 –156 vol.1, 2001.

[38] M. Kemp, M. Palanza, C. Skibski, J. Ormsby, and M. Estaphan. Persistence at Full Ocean Depth. In *Autonomous Underwater Vehicles 2012*, 2012.

[39] S. Kemna, M.J. Hamilton, D.T. Hughes, and K.D. LePage. Adaptive autonomous underwater vehicles for littoral surveillance: The GLINT10 field trial results. Technical report, NATO Underwater Research Center, May 2012. NURC-PR-2012-008.

[40] Sea trials prove AUV pipeline inspection value. Website: http://www.offshore-mag.com/articles/print/volume-67/issue-4/subsea/sea-trials-prove-auv-pipeline-inspection-value.html, 2006.

[41] E. Brhaug and P.E. Hagen. Record-Setting AUV Pipeline Inspection in Norway. *Pipeline & Gas Journal*, 238, 2011.

[42] D. McLeod, J. R. Jacobson, and S. Tangirala. Autonomous Inspection of Subsea Facilities-Gulf of Mexico Trials. In *Offshore Technology Conference*, April 2012.

[43] P. Egeskov, A. Bjerrum, A. Pascoal, C. Silvestre, C. age, and L.W. Smitt. Design, construction and hydrodynamic testing of the AUV MARIUS. In *Autonomous Underwater Vehicle Technology, 1994. AUV '94., Proceedings of the 1994 Symposium on*, pages 199 –207, July 1994.

[44] J. Eskesen, D. Owens, M. Soroka, J. Morash, F.S. Hover, and C. Chrysosstomidis. Design and Performance of Odyssey IV: A Deep Ocean Hover-capable AUV. In *Unmanned Untethered Submersible Technology (UUST)*, 2009.

[45] F. Dougherty, T. Sherman, G. Woolweaver, and G. Lovell. An autonomous underwater vehicle (AUV) flight control system using sliding mode control. In *OCEANS '88. A Partnership of Marine Interests. Proceedings*, pages 1265 –1270 vol.4, Oct. 1988.

[46] F. Dougherty and G. Woolweaver. At-sea testing of an unmanned underwater vehicle flight control system. In *Autonomous Underwater Vehicle Technology, 1990. AUV '90., Proceedings of the (1990) Symposium on*, pages 65 –73, June 1990.

[47] A.M. Bradley D.R. Yoerger and B.B. Warren. The Autonomous Benthic Explorer (ABE): an AUV optimized for deep seafloor studies. In *Proc. of the 7th International Symposium on Unmanned Untethered Submersible Technology (UUST91)*, pages 60–70, 1991.

[48] R. Christenson R. Jaffre F. Purcell M. Packard G.E., Stokey and R. Littlefield. Hull inspection and confined area search capabilities of REMUS autonomous underwater vehicle. In *OCEANS 2010*, pages pp 1–4, September 2010.

[49] R. Brown, J. Duchesney, J. Martz, P. Pietryka, and C. Alt. Development of the Remus 600 Hovering Autonomous Underwater Vehicle. In *Unmanned Untethered Submersible Technology (UUST) 2011*, 2011.

[50] C. Silvestre and A. Pascoal. Depth control of the INFANTE AUV using gain-scheduled reduced order output feedback. *Control Engineering Practice*, 15(7):883 – 895, 2007. Special Issue on Award Winning Applications, IFAC World Congress.

[51] V. Kanakakis, N. C. Tsourveloudis, and K. P. Valavanis. Design and testing of a fuzzy logic controller for an autonomous underwater vehicle. In *Proc. of the IARP Internat. Workshop on Underwater Robotics for Sea Exploration and Environmental Monitoring*, Riode, 2001.

[52] F. Santoso, M. Liu, and G. Egan. Linear Quadratic Optimal Control Synthesis for a UAV. In *AIAC12*, Melbourne, Australia, March 2007.

[53] R. McEwen and K. Streitlien. Modeling and Control of a Variable-Length AUV. Technical report, Monterey Bay Aquarium Research Institute, Bluefin Robotics, Corp., 2006.

[54] S.D. McPhail and M. Pebody. Autosub-1. A distributed approach to navigation and control of an autonomous underwater vehicle. In *Electronic Engineering in Oceanography, 1997. Technology Transfer from Research to Industry., Seventh International Conference on*, pages 16 –22, June 1997.

[55] S. D. McPhail. RRS Discovery Cruise 323 - First Deepwater Trials of the Autosub6000 AUV. Technical report, National Oceanography Centre, Southampton, 2008.

[56] M. Santhakumar and T. Asokan. A Self-Tuning Proportional-Integral-Derivative Controller for an Autonomous Underwater Vehicle, Based On Taguchi Method. *Journal of Computer Science 6*, 8:862–871, 2010.

[57] I. Boiko, L. Fridman, A. Pisano, and E. Usai. Analysis of Chattering in Systems with Second-Order Sliding Modes. *Automatic Control, IEEE Transactions on*, 52(11):2085 –2102, Nov. 2007.

[58] A.J. Healey and D. Lienard. Multivariable sliding mode control for autonomous diving and steering of unmanned underwater vehicles. *Oceanic Engineering, IEEE Journal of*, 18(3):327 –339, July 1993.

[59] A.J. Healey and D.B. Marco. Slow Speed Flight Control of Autonomous Underwater Vehicles: Experimental Results with the NPS AUV II. In *2nd International Offshore and Polar Engineering Conference (ISOPE)*, pages 523–532, San Francisco, CA., 1992.

[60] P. Bouffard, A. Aswani, and C. Tomlin. Learning-Based Model Predictive Control on a Quadrotor: Onboard Implementation and Experimental Results. In *Bouffard ICRA*, Saint Paul, MN, USA, May 2012.

[61] Y.C. Liu and C.Y. Lin. Model predictive control with integral control and constraint handling for mechatronic systems. In *Modelling, Identification and Control (ICMIC), The 2010 International Conference on*, pages 424 –429, July 2010.

[62] W. Naeem, R. Sutton, J. Chudley, F. R. Dalgleish, and S. Tetlow. An online genetic algorithm based model predictive control autopilot design with experimental verification. *International Journal of Control*, 78(14):1076–1090, 2005.

[63] A. Budiyono. Model Predictive Control for Autonomous Underwater Vehicle. *Indian Journal of Geo-Marine Sciences*, Vol. 40(2):pp. 191–199, 2011.

[64] M. Abdolhosseini, Y. Zhang, and C. Rabbath. Trajectory Tracking with Model Predictive Control for an Unmanned Quad-rotor Helicopter: Theory and Flight Test Results. *Intelligent Robotics and Applications*, 7506:411–420, 2012.

[65] T. Manrique, H. Malaise, M. Fiacchini, T. Chambrion, and G. Millerioux. Model predictive real-time controller for a low-consumption electric vehicle. In *Environment Friendly Energies and Applications (EFEA), 2012 2nd International Symposium on*, pages 88 –93, June 2012.

[66] J. M. Maciejowski. *Predictive control with constraints.* Prentice Hall, Essex, England, 2002.

[67] Clifford Hildreth. A quadratic programming procedure. *Naval Research Logistics Quarterly*, 4(1):79–85, 1957.

[68] L. Wang. *Model Predictive Control System Design and Implementation Using MATLAB.* Springer, 2010.

[69] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, New York, 2nd edition, 2006.

[70] David G. Luenberger. *Optimization by Vector Space Methods.* John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1997.

[71] K. Dutton, S Thompson, and B. Barraclough. *The Art of Control Engineering.* Pearson Prentice Hall, 1997.

[72] Numpy and Scipy Information. Website: http://docs.scipy.org/doc/, September 2012.

[73] CVXOPT Package. Website: http://abel.ee.ucla.edu/cvxopt/, September 2012.

[74] Yottalab Python Package. Online file: www.dti.supsi.ch/ $\tilde{}$bucher/ wp-content/ uploads/ 2011/ 03/ yottalab.py, September 2012.

[75] Slycot Package. Website: https://github.com/avventi/Slycot, September 2012.

[76] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[77] Ubuntu Operating System. Website: http://www.ubuntu.com/, July 2012.

[78] Seacon Worldwide. Wet-con connectors. Website: http:// seaconworldwide.com/ products/ electrical-wet-mate/ wet-con/, July 2012.

[79] TSL Technologies. TSL thruster controllers. Website: http:// www.tsltechnology.com/ marine/ controllers_data.htm, July 2012.

[80] Sensoray. S2255 Frame-grabber. Website: http://www.sensoray.com/products/2255.htm, July 2012.

[81] J. Feldman. DTNSRDC revised standard submarine equation of motion. Technical report, David W. Taylor Navel Ship Research and Development Center, June 1979.

[82] P. Ridley, J. Fontan, and P. Corke. Submarine dynamic modelling. In Jonathan Roberts and Gordon Wyeth, editors, *Australasian Conference on Robotics and Automation 2003*, Brisbane, December 2003. Australian Robotics & Automation Association. Proceedings of ACRA available on CD-ROM, at a cost of Aus$200 each.

[83] T.I. Fossen. *Guidance and control of ocean vehicles.* Wiley, 1994.

[84] M. Drela. XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils. 1989.

[85] J.P. Comstock, editor. *Principles of Naval Architecture (Second Revision), Volume III - Motions in Waves and Controllability.* Society of Naval Architects and Marine Engineers (SNAME)., 1989.

[86] J. Norris. Wind tunnel testing of Delphin2. Master's thesis, Engineering and the Environment, University of Southampton, June 2012.

[87] Wind tunnel balance details. Website: http:// www.windtunnel.soton.ac.uk/ balance/ balance_7x5highspd.html, July 2012.

[88] William H. Jr. RAE and Alan POPE. *Low-Speed Wind Tunnel Testing*, volume 3. Wiley, New York, 1999.

[89] A.R. Palmer, G.E. Hearn, and P. Stevenson. Experimental testing of an autonomous underwater vehicle with tunnel thrusters. In *SMP'09 - First International Symposium on Marine Propulsors*, pages 6–12, Trondheim, Norway, 2009.

[90] A. Hughes, S.M. Sharkh, and S.R. Turnock. CFD modelling of a novel electromagnetic tip-driven thruster. In *International Offshore and Polar Engineering Conference.*, pages 294 – 298, Seattle, USA, 2000.

[91] A. Saunders and M. Nahon. The effect of forward vehicle velocity on through-body AUV tunnel thruster performance. In *Oceans '02 MTS/IEEE.*, page 250 259, 2002.

[92] A.J. Healey, S.M. Rock, S. Cody, D. Miles, and J.P. Brown. Toward an improved understanding of thruster dynamics for underwater vehicles. volume 20, pages 354 –361, Oct. 1995.

[93] MATLAB. Curve fitting toolbox. Website: http:// www.mathworks.co.uk/ help/ toolbox/ curvefit/ bs9vw_1-1.html, July 2012.

[94] L.V. Steenson, A.B. Phillips, E. Rogers, M.E. Furlong, and S.R. Turnock. The performance of vertical tunnel thrusters on an autonmous underwater vehicle operating near the free surface in waves. In *Second International Symposium on Marine Propulsors*, Humburg, Germany, June 2011c. SMP11.

[95] S.I. Baniela. The Performance of a Tunnel Bow Thruster with Slow Speed Ahead: A Revisited Issue. *Journal of Navigation*, 62:631–642, 2009.

[96] M. Insel, A. F. Molland, and J. F. Wellicome. Wave resistance prediction of a catamaran by linearised theory. In *Fifth International Conference on Computer Aided Design, Manufacture and Operation. CADMO'94.*, 1994.

[97] D. Perrault, N. Bose, S. OYoung, and C.D. Williams. Sensitivity of AUV response to variations in hydrodynamic parameters. *Ocean Engineering*, 30(6):779 – 811, 2003.

[98] R.A. Nichols, R.T. Reichert, and W.J. Rugh. Gain scheduling for H-infinity controllers: a flight control example. *Control Systems Technology, IEEE Transactions on*, 1(2):69–79, 1993.

[99] Wilson J. Rugh and Jeff S. Shamma. Research on Gain Scheduling . *Automatica*, 36(10):1401 – 1425, 2000.

[100] N. I. Kimber and K. H. Scrimshaw. Hydrodynamic testing of a 3/4 scale Autosub model. In *Proc. Oceanology International '94*, Brighton, UK, March 1994.

[101] A.B. Phillips, M. E. Furlong, and Turnock S.R. Virtual planar motion mechanism tests of the autonomous underwater vehicle autosub. *STG-Conference / Lectureday "CFD in Ship Design"*, September 2007.

[102] D. Minchin, F. Lucy, and M. Sullivan. Ireland: a new frontier for the zebra mussel Dreissena polymorpha (Pallas). *Oceanological and Hydrobiological Studies*, 34:19–30, 2005.

[103] Julian Fowler. Voyage to the bottom of the lough. Website: http://www.bbc.co.uk/news/uk-northern-ireland-18369277, May 2012.

[104] Julian Fowler. Yellow submarine hunting for alien species in lough waters. Website: http://www.bbc.co.uk/news/uk-northern-ireland-18259919, May 2012.

[105] C. Harris. Contingency planning for long-duration AUV missions. In *Autonomous Underwater Vehicles (AUV)*, 2012.

[106] Sound Metrics. ARIS EXPLORER 3000. Website: http:// www.soundmetrics.com/ sites/ default/ files/ uploads/ products/ ARIS3000.pdf, October 2012.

[107] E. Galceran. A Real-time Underwater Object Detection Algorithm for Multi-beam Forward Looking Sonar. In *Navigation, Guidance and Control of Underwater Vehicles (NGCUV)*, April 2012.

[108] P. Chapple. Automated Detection and Classification in High-resolution Sonar Imagery for Autonomous Underwater Vehicle Operations. *Maritime Operations Division. Australian Government Department of Defence.*, DSTO-GD-0537, 2008.

[109] SP. Breton and G. Moe. Status, plans and technologies for offshore wind turbines in Europe and North America. *Renewable Energy*, 34(3):646 – 654, 2009.

[110] E. A. Hansen, H.J. Simonsen, A.W. Nielsen, J. Pedersen, and M. Hgedal. Scour protection around offshore wind turbine foundations. In *European wind energy conference proceedings*, Milan, Italy, 2007.

[111] B.W. Hobson, R.S. McEwen, J. Erickson, T. Hoover, L. McBride, F. Shane, and J.G. Bellingham. The Development and Ocean Testing of an AUV Docking Station for a 21 AUV. In *Oceans*, Vancouver, B.C., October 2007.

[112] J. Silva, A. Martins, and F. L. Pereira. A reconfigurable mission control system for underwater vehicles. In *Proceedings of the MTS/IEEE Oceans '99*, 1999.

Appendix A

MPC Code

A.1 MATLAB

In this appendix the Matlab code is given that simulates the over-actuated MPC controller developed in Chapter 8. In the second half of this appendix the python code, used on the Delphin2 AUV, is given.

A.1.1 Over-Actuated MPC Controller

```
clear; clc; pause(0.1); tic

%%%% Controller parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Np          =  60;      % Length of prediction horizon
Nc          =  8;       % Length of control horizon
Tgain       =  0.05;    % Thruster penalty term

gain = zeros(4,4);
gain(1,1) = 1.0;        % Prop gain
gain(2,2) = 0.005;      % Foil gain
gain(3,3) = 0.2;        % T0 gain
gain(4,4) = 0.2;        % T1 gain

inputs = 4;

for i = 1:Nc-1
gain((i*inputs+1):(i*inputs+inputs),(i*inputs+1):(i*inputs+inputs)) ...
        = gain(1:inputs,1:inputs);
end

%%%% Simulation parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
setpoint    =  [1.0 1.0]';  % Controller set-points; surge vel, depth
dt          =  0.2;         % Controller sampling time (s)
sim_length  =  50;          % Length of simulation (s)

%%% Initial Conditions %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
delta       = [0 0 0 0]';
delta_demand= [0 0 0 0]';
T_demands   = [0 0 0 0]';
T0x         = [0 0]';
T1x         = [0 0]';
T2x         = [0 0]';
T3x         = [0 0]';
prop        = 0;
velX        = 0;
velY        = 0;
velH        = 0;
velP        = 0;
P           = 0;
velZ        = 0;
Z           = 0;
u           = [0 0 0 0 0 0 0 0 0 0 0 0]';
y           = [0 0]';
X           = 0;
```

```
Y           = 0;
H           = 0;

%%%% Create arrays %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
time     = 0:dt:sim_length;
sp       = zeros(sim_length/dt,2);
x        = zeros(5,sim_length/dt);
pitch    = zeros(sim_length/dt,1);
depth    = zeros(sim_length/dt,1);
t        = zeros(sim_length/dt,1);
U        = zeros(sim_length/dt,inputs);
dU       = zeros(sim_length/dt,inputs);
propSP   = zeros(sim_length/dt,1);
vel_x    = zeros(sim_length/dt,1);
vel_p    = zeros(sim_length/dt,1);
vel_y    = zeros(sim_length/dt,1);
X_pos    = zeros(sim_length/dt,1);
Y_pos    = zeros(sim_length/dt,1);
heading  = zeros(sim_length/dt,1);

%%%% Get first model %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
getMIMO_DepthSpeed2;

%%%% More initialisation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xm       = zeros(length(A),1);
Xf       = zeros(n,1);
Xfo    = zeros(n+1,1);

%%%% Build constraints matrices %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
conMatrices
lambda = zeros(32,1);

i = 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Start of Simulation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for kk=[1:sim_length/dt];

    %%% Get latest linearised model %%%
    getMIMO_DepthSpeed2;

    %%%% Set Constraints %%%%
    conMatrices;

    %%%% Find Optimal DeltaU %%%%
    F = -(Phi_R*setpoint-Phi_F*Xf - ...
        [0; 0; u(3:4); zeros(Nc*inputs - inputs,1)]*Tgain*abs(velX));
    [DeltaU, km(kk), lambda] = QPhild(E,F,M,gamma,lambda);


    %%%% Optimum %%%%
    dprop  = DeltaU(1);
    ddelta = DeltaU(2);
    dT0    = DeltaU(3);
    dT1    = DeltaU(4);

    u(1,:) = u(1,:) + dprop;
    u(2,:) = u(2,:) + ddelta;
    u(3,:) = u(3,:) + dT0;
    u(4,:) = u(4,:) + dT1;

    prop         = u(1,:);
    delta_demand = u(2,:);
    T_demands    = [u(3,:) u(4,:) 0 0]';

    %%%% Nonlinear Vehicle Model %%%%
    [accP, velP, P, accZ, velZ, Z, velX, X, velH, H, ...
        velY, Y, delta, T0x, T1x,T2x,T3x] = ...
    AUV2(prop, delta, delta_demand,T0x,T1x,T2x,T3x, ...
        T_demands, velP,P, velZ,Z, velX, X, velH, H, velY, Y);

    Zraw = Z;        % + randsample(depth_noise(:,1),1);
    Praw = P*pi/180; % + randsample(pitch_noise(:,1),1))*pi/180;

    %%%% Estimate State-variable Vector %%%%
    y   = [velX Zraw Praw]';
    Xfo = A_o*Xfo+K_ob*(y-C_o*Xfo)+B_o*DeltaU(1:4);
    Xf  = Xfo(1:end-1);

    XF(:,kk) = Xf;

    %%%% Save Data %%%%
    t(kk)        = time(kk);
    sp(kk,:)     = setpoint;
    U(kk,:)      = u(1:inputs);
    dU(kk,:)     = DeltaU(1:inputs);
    depth(kk)    = Z;
    pitch(kk)    = P;
    vel_x(kk)    = velX;
    vel_p(kk)    = velP;
    vel_z(kk)    = velZ;

    %%%% Percentage Done %%%%
    fprintf('Percentage done = %f \n',(kk/(sim_length/dt))*100)

end
%%%% End of Simulation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
toc


%%%% PLOT RESULTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
scrsz = get(0,'ScreenSize');
figure('Name','Over-actuated MPC2','Position',[50 0 700 700])

subplot(4,1,1)
```

```
plot(t,vel_x,t,sp(:,1),'--','linewidth',2);

xlabel({'\fontsize{12}Time (s)';'\fontsize{16}Surge Velocity'})
ylabel('\fontsize{12}u_v (ms^{-1})')
legend('Surge velocity','Setpoint','location','northeast')
grid 'on'
% ylim([0 1.5])
% xlim([50 250])

subplot(4,1,2)
plot(t,depth,t,sp(:,2),'--','linewidth',2);
xlabel({'\fontsize{12}Time (s)';'\fontsize{16}Depth'})
ylabel('\fontsize{12}z (m)')
legend('Depth','Setpoint','location','southeast')
grid 'on'
ylim([0 1.5])
% xlim([50 250])

subplot(4,1,3)
plot(t,pitch,'linewidth',2);
xlabel({'\fontsize{12}Time (s)';'\fontsize{16}Pitch'})
ylabel('\fontsize{12}\theta (deg)')
legend('Pitch','location','northwest')
grid 'on'
% ylim([-20 10])
% xlim([50 250])

subplot(4,1,4)
plot(t,U(:,1),t,U(:,2),t,U(:,3),t,U(:,4),'--','linewidth',2);
xlabel({'\fontsize{12}Time (s)';'\fontsize{16}u'})
ylabel('\fontsize{12}u')
legend('T_{prop}','\delta','T_{vf}','T_{vr}','location','northeast')
grid 'on'
% ylim([-30 30])
% xlim([50 250])
```

getMIMO_DepthSpeed2 Script

```
%%%% CONSTANTS %%%%
massZ   = 167.5;
massX   = 85;
massBG  = 55;
I       = 70;
Volume  = 0.08;
LTf     =-0.55;
LTr     = 0.49;
Lr      = 1;
L       = 2;
BG      = -0.005;
m       = massBG*9.81;
rho     = 1000;
G       = m*BG;

%%%% FOILS %%%%
```

```
fCl =  0.0001578;                   % Lift coefficient per degree
fCd =  3.421e-5*delta(1)^2;         % Drag coefficient per degree squared
fCm =  8.057e-5;                    % Moment coefficient per degree

%%%% HULL %%%%
hCl =  0.001512;                    % Lift coefficient equation per degree
hCd =  0.0005719*P^2 + 0.07265;     % Drag coefficient equation per degree squared
hCm =  0.0001596;                   % Moment coefficient equation
hCdp=  0.027;                       % Pitch rate damping coefficient
hCdz=  3.33;                        % Heave rate damping coefficient

%%%%% FOILS %%%%%%%%%
FL   = -0.5*rho*(L^2)*(fCl)*velX*abs(velX);     % Times alpha to get lift
FDx  = -0.5*rho*(Volume^(2/3))*(fCd)*abs(velX); % Times velX to get drag
FM   = -0.5*rho*(L^3)*(fCm)*velX*abs(velX);     % Times alpha to get moment

%%%%% HULL %%%%%%%%%
HL   = -0.5*rho*(L^2)*(hCl)*velX*abs(velX)*180/pi; % Times pitch to get lift
HDx1 = -0.5*rho*(Volume^(2/3))*(hCd)*abs(velX);    % Times velX to get drag
HM   =  0.5*rho*(L^3)*(hCm)*velX*abs(velX)*180/pi; % Times pitch to get moment

%%%%% THRUSTERS %%%%%
T0   = T_demands(1);
T1   = T_demands(2);
Told = T0 + T1;
T_coeff = 1/exp(abs(velX));

%%%%% PROP %%%%%%%%%
prop = prop;

%%%%% MODS %%%%%%%%%
HDz  = (-0.0862 * 0.5*1000*0.08^(2/3)) + ...
       (-0.0862 * 0.5*1000*0.08^(2/3))*40*abs(velX);
HDp  = (-0.01303 * 0.5*1000*0.08^(2/3)) + ...
       (-0.01303 * 0.5*1000*0.08^(2/3))*40*abs(velX);

%%%%% AUV STATE SPACE MODEL %%%%%%%%%%
A = [(HDx1 + FDx)/massX 0 0 Told/massX 0; 0 0 1 0 0; ...
    0 0 HDz/massZ (-prop+HL)/massZ 0; 0 0 0 0 1;0 0 0 (HM + G)/I HDp/I];
B = [1/massX 0 0 0; 0 0 0 0; 0 FL/massZ T_coeff/massZ T_coeff/massZ; ...
    0 0 0 0; 0 FM/I T_coeff*LTf/I T_coeff*LTr/I];
C = [1 0 0 0 0; 0 1 0 0 0];
D = zeros(2,4);

[A, B, C, D] = c2dm(A, B, C, D, dt);

%%%% Create augmented model %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[Phi_Phi, Phi_F,Phi_R, A_e, B_e, C_e, Phi, F2, Ky, Kmpc] = mpcgain5(A,B,C,Nc,Np);
[n,n_in] = size(B_e);

E = (Phi_Phi + gain);

%%%% Observer %%%%
```

```
C_om = [1 0 0 0 0; 0 1 0 0 0; 0 0 0 1 0];
[A_o, B_o, C_o, K_ob] = obsv_matrices(A,B,C_om);
```

Set_Contraints Script

```
clear M
clear gamma0
clear gamma

%%%% NORMAL CONTRAINTS %%%%%
Propmin     =  1.5;
Propmax     =  17.8;
dPropmax    =  1;
dPropmin    = -1;
CSmax       =  30;
CSmin       = -30;
dCSmax      =  4;
dCSmin      = -4;
Tmax        =  10;
Tmin        =  0.4;
dTmax       =  2.0;
dTmin       = -2.0;

%%%% VARIABLE CONSTRAINTS %%%%
if setpoint(1) == 0
    Propmax = 0;
    Propmin = 0;
end

if velX > 0.0;
    Tmin = 0;
end

%%% BUILD M MATRIX %%%%
constraints = [dPropmax dPropmin Propmax Propmin; dCSmax dCSmin CSmax CSmin; dTmax dTmin Tma

[in, c] = size(constraints);

Mdef = [1 -1 1 -1]';

for Un = 1:in
    for NC = 1:Nc

        %%% M MATRIX %%%
        rowstart = ((Un-1)*Nc*4) + ((NC-1)*4 + 1);
        col      = (NC-1)*in + Un;

        M(rowstart:rowstart+3, col) = Mdef;

        %%% GAMMA0 %%%
        gamma(rowstart:rowstart+3,1) = diag(Mdef)*constraints(Un,:)' + [0; 0; -u(Un); u(Un)]

    end
end
```

Hildreth Programming Procedure - QPhild

```
function [eta,km,lambda]=QPhild(H,f,A_cons,b,lambda)

km = 0;
eta=-H\f;

kk = length(find(A_cons*eta>b));
if (kk==0); return; end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
P=A_cons*(H\A_cons');   %H
d=(A_cons*(H\f)+b);     %K

[n,m]=size(d);          %Dimensions of K
x_ini=zeros(n,m);       %Zero matrix with dimensions of K

lambda=x_ini;           %lambda zero = zeros
al=10;                  %Convergence check value

for km=1:100
lambda_p=lambda;

for i=1:n
w= P(i,:)*lambda-P(i,i)*lambda(i,1);
w=w+d(i,1);
la=-w/P(i,i);
lambda(i,1)=max(0,la);
end

al=(lambda-lambda_p)'*(lambda-lambda_p);


if (al<10e-8); break; end
end
eta=-H\f -H\A_cons'*lambda;
```

Augmented Model Script - mpcgain5

```
function [Phi_Phi,Phi_F,Phi_R,A_e, B_e,C_e,Phi,F2,Ky,Kmpc] = ...
  mpcgain5(Ad,Bd,Cd,Nc,Np)

[m1,n1]=size(Cd);
[n1,n_in]=size(Bd);
A_e=eye(n1+m1,n1+m1);
A_e(1:n1,1:n1)=Ad;
A_e(n1+1:n1+m1,1:n1)=Cd*Ad;
B_e=zeros(n1+m1,n_in);
B_e(1:n1,:)=Bd;
B_e(n1+1:n1+m1,:)=Cd*Bd;
C_e=zeros(m1,n1+m1);
C_e(:,n1+1:n1+m1)=eye(m1,m1);
```

```
P = zeros(m1*Np,n_in);
F = zeros(m1*Np,n1+m1);

n=n1+m1;

P(1:m1,1:n_in)=C_e*B_e;
F(1:m1,1:n)=C_e*A_e;

[rP,cP] = size(P);
[rF,cF] = size(F);

for kk=1:Np-1
    F(kk*m1+1:(kk+1)*m1,:)= F(kk*m1-1:kk*m1,:)*A_e;
    P(kk*m1+1:(kk+1)*m1,:) = F(kk*m1-1:kk*m1,:)*B_e;
end

Phi=zeros(Np*m1,Nc*n_in);
Phi(:,1:n_in) = P;

for i=1:Nc-1
    Phi(:,i*n_in+1:(i+1)*n_in)=[zeros(m1*i,n_in);P(1:m1*Np-i*m1,:)]; %Toeplitz matrix
end

Phi_Phi= Phi'*Phi;
Phi_F= Phi'*F;
Phi_R=  Phi_F(:,6:7);

F2   = F;
Ky   = a*inv(Phi_Phi+0.01*eye(length(Phi_Phi)))*(Phi'*ones(length(Phi'),1));
Kmpc = a*inv(Phi_Phi+0.01*eye(length(Phi_Phi)))*(Phi_F);
```

AUV Non-Linear Model

```
function [accP, velP, P, accZ, velZ, Z, velX, X, velH, H, velY,...
            Y, delta, T0x, T1x,T2x,T3x] = ...
         AUV2(prop, delta, delta_demand,T0x,T1x,T2x,T3x, T_demands, ...
            velP,P, velZ,Z, velX, X, velH, H, velY, Y)

dt_sim = 0.05;
massZ  = 167.5;
massX  = 85;
massBG = 55;
buoyancy = -5;
I      = 70;
Volume = 0.08;
LTf    =-0.55;
LTr    = 0.49;
LTfh   =-0.65;
LTrh   = 0.59;
Lr     = 1;
L      = 2;
BG     = -0.005;
m      = massBG*9.81;
```

```
rho    = 1000;


trimErrorMoment = 0;

%%%% Actuator model %%%%
Ta = [0.9017 -0.0792; 0.076 0.9968];
Tb = [0.038; 0.0015];
Tc = [0 2.0833];


%%%% FOILS %%%%
mean_delta = mean(delta);
fCl = -0.0001574;                                  % Lift coefficient per degree
fCd =  3.351e-5*mean_delta^2 + 0.0002868*mean_delta;% Drag coefficient per degree squared
fCm = -8.057e-5;                                   % Moment coefficient per degree
lim =  10;

%%%% HULL %%%%
hCl_dot = -0.001512;                               % Lift coefficient equation per degree
hCdp    =  0.027;
hCdh    =  3.33;
hCdz    =  3.33;


for k = 1:4

    %%%%% FLOW ANGLES %%%%%
    if velZ && velX > 0
    try
        inflow_angle = (180/pi)*atan(velZ/velX);
    end

    else
        inflow_angle = 0;
    end

    %%%%% THRUSTERS %%%%%
    T0x = Ta*T0x + Tb*T_demands(1);
    T0  = Tc*T0x;

    T1x = Ta*T1x + Tb*T_demands(2);
    T1  = Tc*T1x;

    T2x = Ta*T2x + Tb*T_demands(3);
    T2  = Tc*T2x;

    T3x = Ta*T3x + Tb*T_demands(4);
    T3  = Tc*T3x;

    T0  = T0/exp(abs(velX));
    T1  = T1/exp(abs(velX));
    T2  = T2/exp(abs(velX));
```

```
T3  = T3/exp(abs(velX));

%%%%% FOILS %%%%%%%%%
delta_delta = delta_demand - delta;
delta_delta(delta_delta > lim) = lim;
delta_delta(delta_delta < -lim) = -lim;

delta       = delta + delta_delta;

alphaSP = (delta + P + inflow_angle);

foil_liftZ   =  0.5*rho*(L^2)*(fCl*(alphaSP(2)+alphaSP(4))/2)*velX*abs(velX);
foil_liftY   = -0.5*rho*(L^2)*(fCl*(delta(1)+delta(3))/2)*velX*abs(velX);

foil_drag    = -0.5*rho*(Volume^(2/3))*(3.421e-5*mean_delta^2)*velX*abs(velX);
foil_momentP =  0.5*rho*(L^3)*(fCm*(alphaSP(2)+alphaSP(4))/2)*velX*abs(velX);
foil_momentH =  0.5*rho*(L^3)*(fCm*(delta(1)+delta(3))/2)*velX*abs(velX);

%%%%% HULL %%%%%%%%%%
hull_angle = P + inflow_angle;

hull_liftZ   =  0.5*rho*(L^2)*(hCl_dot*hull_angle)*velX*abs(velX);
hull_liftY   =  0.5*rho*(L^2)*(hCl_dot*(velH*180/pi))*velX*abs(velX);

hull_dragX   = -0.5*rho*(Volume^(2/3))*(0.000519*hull_angle^2 ...
                + 0.07265)*velX*abs(velX);

hull_momentP =  0.5*rho*(L^3)*(0.0001596*hull_angle)*velX*abs(velX);
hull_momentH =  0.5*rho*(L^3)*(0.0001596*(velH*180/pi))*velX*abs(velX);

hull_dragP = -0.5*rho*(L^2)*(hCdp)*velP*abs(velP);
hull_dragH = -0.5*rho*(L^2)*(hCdh)*velH*abs(velH);
hull_dragY = -0.5*rho*(Volume^(2/3))*(hCdz)*velY*abs(velY);
hull_dragZ = -0.5*rho*(Volume^(2/3))*(hCdz)*velZ*abs(velZ);

%%%%% PROP %%%%%%%%%%
prop = prop;

%%%%% AUV %%%%%%%%%%%
%% pitch %%
accP = (trimErrorMoment + hull_momentP + foil_momentP ...
       + BG*sin(P*pi/180)*m + T0*LTf + T1*LTr  + hull_dragP)/(I);
velP = accP*dt_sim + velP;
P    = velP*dt_sim*180/pi + P;

%% heading %%
accH = (hull_momentH + foil_momentH + T2*LTfh + T3*LTrh + hull_dragH ...
       -20*velH)/(I);
velH = accH*dt_sim + velH;
H    = velH*dt_sim*180/pi + H;

if H > 360; H = H - 360; end
if H < 0; H = H + 360; end
```

```
    %% forward speed %%
    accX = (prop*cos(P*pi/180) + hull_dragX + foil_drag + T0*sin(P*pi/180) ...
           + T1*sin(P*pi/180))/massX ;
    velX = accX*dt_sim + velX;

    %% sway %%
    accY = (hull_liftY + foil_liftY + T2 + T3 + hull_dragY - 20*velY)/massZ ;
    velY = accY*dt_sim + velY;

    %% depth %%
    accZ = (prop*sin(-P*pi/180) + foil_liftZ + hull_liftZ + T0*cos(P*pi/180) ...
           + T1*cos(P*pi/180) + buoyancy + hull_dragZ )/(massZ); %- 20*velZ
    velZ = accZ*dt_sim + velZ;
    Z    = velZ*dt_sim + Z;

    if Z < 0
        velZ = 0;
        Z = 0;
    end

    %% Location %%
    if H >= 0 && H < 90;    X = X + (velX*sin(H*pi/180))*dt_sim;
        Y = Y + (velX*cos(H*pi/180))*dt_sim; end
    if H >= 90 && H < 180;  X = X + (velX*cos((H-90)*pi/180))*dt_sim;
        Y = Y - (velX*sin((H-90)*pi/180))*dt_sim; end
    if H >= 180 && H < 270; X = X - (velX*sin((H-180)*pi/180))*dt_sim;
        Y = Y - (velX*cos((H-180)*pi/180))*dt_sim; end
    if H >= 270 && H < 360; X = X - (velX*cos((H-270)*pi/180))*dt_sim;
        Y = Y + (velX*sin((H-270)*pi/180))*dt_sim; end

    if H >= 0 && H < 90;    X = X + (velY*cos((H-90)*pi/180))*dt_sim;
        Y = Y - (velY*sin((H-90)*pi/180))*dt_sim; end
    if H >= 90 && H < 180;  X = X - (velY*sin((H-180)*pi/180))*dt_sim;
        Y = Y - (velY*cos((H-180)*pi/180))*dt_sim; end
    if H >= 180 && H < 270; X = X - (velY*cos((H-270)*pi/180))*dt_sim;
        Y = Y + (velY*sin((H-270)*pi/180))*dt_sim; end
    if H >= 270 && H < 360; X = X + (velY*sin((H-360)*pi/180))*dt_sim;
        Y = Y + (velY*cos((H-360)*pi/180))*dt_sim; end

end
```

A.2 Python

```
#!/usr/bin/python
import roslib; roslib.load_manifest('DelphinROSv2')
import rospy
import time
import numpy as np
import scipy
from scipy import linalg
from math  import *
from yottalab import *
```

```
import cvxopt
from cvxopt import matrix, solvers

from DelphinROSv2.msg import tsl_setpoints
from DelphinROSv2.msg import tail_setpoints
from DelphinROSv2.msg import dead_reckoner
from DelphinROSv2.msg import depthandspeed_MPC
from DelphinROSv2.msg import ForcesAndMoments #Only temporary
from DelphinROSv2.msg import compass
from std_msgs.msg import Float32
from std_msgs.msg import Bool
from std_msgs.msg import Int8

################################################################################
########## MAIN CONTROL LOOP ###################################################
################################################################################

def main_control_loop():

    global onOFF                            # Controller on or off (bool)
    global speed_demand                     # Surge velocity set-point
    global depth_demand                     # Depth set-point

    #### SOLVER PARAMETERS ####
    solvers.options['show_progress'] = False # Display solver information
    solvers.options['maxiters']      = 100   # Solver interations (default = 100, tune to st

    #### CONTROLLER PARAMETERS ####
    Np          = 60
    Nc          = 8
    gain_P      = 1.0
    gain_F      = 0.005
    gain_T      = 0.2
    Tgain       = 0.05
    delta_t     = 0.2
    Thrust_Smin = 0
    Thrust_Smax = 2500

    #### BUILD GAIN MATRIX ####
    inputs      = 4
    gain_matrix = np.zeros([inputs*Nc,inputs*Nc], dtype = np.float)
    gain_matrix[0,0] = gain_P
    gain_matrix[1,1] = gain_F
    gain_matrix[2,2] = gain_T
    gain_matrix[3,3] = gain_T
    i = 0
    while i < Nc:
        gain_matrix[i*inputs:i*inputs+inputs,i*inputs:i*inputs+inputs] =
                                                gain_matrix[0:inputs,0:inputs]
        i = i + 1

    #### INITIAL CONDITIONS ####
    [velX, velZ, velP]    = [0, 0, 0]
    [pitch, depth]        = [0, 0]
    [prop, delta, T0, T1] = [0, 0, 0, 0]

    #### BUILD AUGMENTED MODEL ####
    [Ad, Bd, Cd, Dd, FL, FDx, FM, HL, HDx, HM, HDp, HDz, G] =
                        get_model(velX,velZ,velP,pitch,delta,T0,T1,prop,delta_t)

    [Phi_Phi, Phi_F, Phi_R, A_e, B_e, C_e] =
                        mpc_gain(Np, Nc, Ad, Bd, Cd, Dd)# Create augmented model

    #### OBSERVER ####
    Co = np.matrix([[1, 0, 0, 0, 0],
                    [0, 1, 0, 0, 0],
                    [0, 0, 0, 1, 0]],  dtype = np.float)
    [A_o, B_o, C_o, K_ob] = obsv_matrices(Ad,Bd,Co)

    #### INITIALISE ####
    time_zero = time.time()
    [q, n]    = np.shape(Cd)
    [n, m]    = np.shape(Bd)
    xm        = np.zeros([n,1],float)
    Xf        = np.zeros([n+q,1],float)
    Xfo       = np.zeros([n+q+1,1],float)
    u         = np.matrix([[0.0],[0.0],[0.0],[0.0]])
    Delta_u   = np.matrix([[0.0],[0.0],[0.0],[0.0]])
    gammaN    = np.zeros([32,1],float)
    pn        = 0
    demands   = np.matrix([[speed_demand],[depth_demand]])

    #### SET CONTRAINTS ####
    [gamma, M] = set_constraints(Nc,  m, velX, speed_demand, u)

################################################################################
################################################################################
    print 'Depth controller ready'

    while not rospy.is_shutdown():

        dt = time.time() - time_zero     # Calculate time since last calculation
        ########################################################################
        if dt >= delta_t and onOFF == True:

            time_zero = time.time()

            #### Get latest feedback ###########################################
            velX  = DR.velX
            depth = DR.Z
            pitch = DR.pitch

            #### Get setpoints #################################################
            demands = [[speed_demand],[depth_demand]]              # Set demands

            #### Build model ###################################################
```

```
[Ad, Bd, Cd, Dd, FL, FDx, FM, HL, HDx, HM, HDp, HDz, G] =
            get_model(velX,velZ,velP,pitch,delta,T0,T1,prop,delta_t)
[Phi_Phi, Phi_F, Phi_R, A_e, B_e, C_e] =
            mpc_gain(Np, Nc, Ad, Bd, Cd, Dd)

#### Set constraints ##############################################
[gamma, M] = set_constraints(Nc,  m, velX, speed_demand, u)

#### Find optimal #################################################
neg = np.zeros([Nc*m,1])
neg[2,0:1] = u[2] * Tgain*abs(velX)
neg[3,0:1] = u[3] * Tgain*abs(velX)

E = Phi_Phi + gain_matrix
F = -(np.dot(Phi_R,demands) - np.dot(Phi_F,Xf) - neg)
km = 0

t = time.time()
E = matrix(E)
F = matrix(F)
M = matrix(M)
b = matrix(gamma)

try:
    sol = solvers.qp(E, F, M, b)
    Delta_u = sol['x']
except:
    Delta_u = Delta_u
    print 'solution failed!'

#### Set control values ###########################################
dprop  = float(Delta_u[0,0])
ddelta = float(Delta_u[1,0])
dT0    = float(Delta_u[2,0])
dT1    = float(Delta_u[3,0])

prop   = u[0] = float(u[0] + dprop)
delta  = u[1] = float(u[1] + ddelta)
T0     = u[2] = float(u[2] + dT0)
T1     = u[3] = float(u[3] + dT1)

if depth_demand == 0.0:
    u[1] = -7.0
    delta = -7.0

#### Estimate state variable vector ###############################
[A_o, B_o, C_o, K_ob] = obsv_matrices(Ad,Bd,Co)

y   = [[velX],[depth],[pitch*np.pi/180.0]]
Xfo = np.dot(A_o,Xfo) + np.dot(K_ob, (y - np.dot(C_o, Xfo)))
                                        + np.dot(B_o, Delta_u[0:4])
Xf  = Xfo[0:7]
```

```
        #### Convert thrust to setpoint demand ############################
        if velX > 0.4 and T0 < 1.25:
            thruster0 = 0
        else:
            thruster0 =  int(limits((1.13*np.sign(T0)*(60.0*(np.abs(T0)
                    /(1000*0.46*0.07**4))**0.5)), Thrust_Smin, Thrust_Smax))

        if velX > 0.4 and T1 < 1.25:
            thruster1 = 0
        else:
            thruster1 =  int(limits((1.13*np.sign(T1)*(60.0*(np.abs(T1)
                    /(1000*0.46*0.07**4))**0.5)), Thrust_Smin, Thrust_Smax))

        #### Convert prop thrust to setpoint demand #######################
        if prop > 2.0 and speed_demand > 0.1:
            propSP = int(round((prop + 13.9434)/1.5116))
        else:
            propSP = 0

        if propSP > 22:
            propSP = 22

        if propSP < -22:
            propSP = -22

        #### Calculation time ##############################################
        calc_time = time.time() - time_zero     # Time taken to do calculations
        ###################################################################

        ### Publish values ################################################
        pub_tsl.publish(thruster0 = thruster0, thruster1 = thruster1)
        pub_tail.publish(cs0 = -u[1], cs1 = -u[1])
        pub_prop.publish(propSP)
        pub_C.publish(onOFF = onOFF, depth = depth, depth_demand = depth_demand,
            pitch = pitch, speed = velX, speed_demand = speed_demand,
            prop = prop, dprop = dprop, ddelta = ddelta, delta = delta,
            dT0 = dT0, T0 = T0, T1 = T1, thruster0 = thruster0,
            thruster1 = thruster1, prop_gain = gain_P, delta_gain = gain_F,
            thruster_gain = gain_T, Np = Np, Nc = Nc, delta_t = delta_t,
            calc_time = calc_time, km = km, propSP = propSP, sim_time = sim_time,
            Xf1 = Xfo[0], Xf2 = Xfo[1], Xf3 = Xfo[2], Xf4 = Xfo[3],
            Xf5 = Xf[4], Xf6 = Xfo[5], Xf7 = Xfo[6], Xf8 = Xfo[7])

        sim_time   = sim_time + delta_t
        simulation = False
        ###################################################################
    else:
        time.sleep(0.001)

######## END OF LOW LEVEL CONTROLLER ##########################################
################################################################################
################################################################################
```

```
def set_constraints(Nc, n_in, velX, speed_demand, u):


    #### SET CONSTRAINTS ####
    if speed_demand > 0.2:
        Propmin  = 2.0
        Propmax  = 17.8
    else:
        Propmax  = 0
        Propmin  = 0

    dPropmax = 1.0
    dPropmin =-1.0

    Tmax     = 10.0
    Tmin     = 0.7

    dTmax    = 2.0
    dTmin    =-2.0

    CSmax    = 30
    CSmin    =-30

    dCSmax   = 4.0
    dCSmin   =-4.0

    if velX >= 0.3:
        Tmin = 0

    #### COMBINE CONSTRAINTS ####
    constraints = np.matrix([[dPropmax, dPropmin, Propmax, Propmin],
                             [dCSmax, dCSmin, CSmax, CSmin],
                             [dTmax, dTmin, Tmax, Tmin],
                             [dTmax, dTmin, Tmax, Tmin]])
    [i, c]      = np.shape(constraints)

    Mdef  = np.matrix([1, -1, 1, -1])
    M     = np.zeros([Nc*i*c,Nc*i])
    gamma = np.zeros([Nc*i*c,1])

    Un = 1
    NC = 1

    #### BUILD M AND GAMMA MATRICES ####
    while Un <= i:

        while NC <= Nc:

            rowstart = ((Un-1)*Nc*c) + ((NC-1)*c + 1) - 1
            col = (NC-1)*i + Un - 1
            M[rowstart:rowstart+c,col] = Mdef
            gamma[rowstart:rowstart+c,0:1] = np.diag([1.0, -1.0, 1.0, -1.0]) *
                    np.transpose(constraints[Un-1,:]) +
                    np.matrix([[0],[0],[-u[Un-1]],[u[Un-1]]])

            NC = NC + 1

        NC = 1
        Un = Un + 1

    return [gamma, M]

################################################################################
################################################################################

def get_model(velX,velZ,velP,pitch,delta,T0,T1,prop,delta_t):

    #### FIXED MODEL VALUES ####
    massZ   = 167.5
    massX   = 85.0
    massBG  = 55.0
    I       = 70.00

    volume  = 0.08
    Ltf     =-0.55
    Ltr     = 0.49
    Lf      = 1.0
    L       = 2.0
    BG      =-0.011
    rho     = 1000.0
    G       = massBG*9.81*BG
    #####################

    #### VERIABLE MODEL VALUES ####
    ## FOILS ##
    fCl     = 0.0 #0.0001574
    fCd     = 3.351e-5*abs(delta)**2
    fCm     = 7.563e-5

    ## HULL ##
    hCl     = 0.001512
    hCd     = 0.07265
    hCm     = 0.0001596
    hCdp    =-0.01303 # Linear terms
    hCdz    =-0.0862  # Linear terms

    ## THRUSTERS ##
    Told    = T0 + T1
    T_coeff = 1.0/exp(abs(velX))

    #### UPDATED COEFFICIENTS ####
    ## FOILS ##
    FL      =-0.5*rho*(L**2)*fCl*velX*abs(velX)
    FDx     =-0.5*rho*(volume**(2.0/3.0))*fCd*abs(velX)
    FM      =-0.5*rho*(L**3.0)*fCm*velX*abs(velX)

    ## HULL ##
    HL   = -0.5*rho*(L**2.0)*(hCl)*velX*abs(velX)*180.0/np.pi
    HDx  = -0.5*rho*(volume**(2.0/3.0))*(hCd)*abs(velX)
    HM   =  0.5*rho*(L**3.0)*(hCm)*velX*abs(velX)*180.0/np.pi
    HDp  = (hCdp * 0.5*rho*L**2) + (hCdp * 0.5*rho*rho*L**2)*abs(velX)*0.0
    HDz  = (hCdz * 0.5*rho*volume**(2.0/3.0)) +
                            (hCdz * 0.5*rho*volume**(2.0/3.0))*abs(velX)*0.0

    #### STATE SPACE MODEL ####
    Ac = np.matrix([[(HDx + FDx)/massX, 0, 0, Told/massX, 0],
                   [0, 0, 1, 0, 0],
                   [0, 0, HDz/massZ, (HL)/massZ, 0],
                   [0, 0, 0, 0, 1],
                   [0, 0, 0, (HM + G)/I, HDp/I]], dtype = np.float)

    Bc = np.matrix([[1/massX, 0, 0, 0],
                   [0, 0, 0, 0],
                   [0, FL/massZ, T_coeff/massZ, T_coeff/massZ],
                   [0, 0, 0, 0],
                   [0, FM/I, (Ltf*T_coeff)/I, (Ltr*T_coeff)/I]],  dtype = np.float)

    Cc = np.matrix([[1, 0, 0, 0, 0],[0, 1, 0, 0, 0]],  dtype = np.float)

    Dc = np.zeros([2,4], dtype = np.float)

    SYS = tuple([Ac, Bc, Cc, Dc])
    [Ad, Bd, Cd, Dd] = cont2discrete(SYS,delta_t)
    return [Ad, Bd, Cd, Dd, FL, FDx, FM, HL, HDx, HM, HDp, HDz, G]

################################################################################
################################################################################

def QPhild(E,F,M,gamma,delta_t,time_zero): # Find optimum control horizon within constraints
    iter = 75

    km      = 0                           # Hildreth iterations equals zero
    [n1,m1] = np.shape(M)
    eta     = -np.dot(np.linalg.inv(E),F) # Find optimum control horizon

    x = True in (np.dot(M,eta) > gamma)   # Check if constraints have been violated

    if not x:                             # If constaints have not been violated with optimu
        return [eta, km]

    H = np.dot(M,(np.dot(np.linalg.inv(E),np.transpose(M)))) # Calculate H matrix
    K = np.dot(M,(np.dot(np.linalg.inv(E),F))) + gamma       # Calculate K matrix

    [n,m] = np.shape(K)

    Lambda =  np.zeros([n,m],float)   # Initialise correction term lambda
    Lambda_p = np.zeros([n,m],float)  # Initialise 'old' correction term lambda_p
    al = 10                            # Initialise convergance criterium to greater than 10e-8

################################################################################
    t = time.time()
    while (km < iter) and (al > 10e-8) and
                      ((time.time() - time_zero) < (delta_t - 0.005)):

        Lambda_p[:,0] =  Lambda[:,0]
        i = 0

        while i < n:
            w  = np.add(np.dot(H[i,:],Lambda), -np.dot(H[i,i],Lambda[i,0]))
            w  = np.add(w,K[i,0])
            la = np.divide(-w,H[i,i])


            Lambda[i,0] = np.max([0.0, la])
            i = i + 1

        al = np.dot(np.transpose(Lambda - Lambda_p),(Lambda - Lambda_p))

        km = km + 1

################################################################################

    eta = -np.dot(np.linalg.inv(E),F) -
                np.dot(np.linalg.inv(E),(np.dot(np.transpose(M),Lambda)))

    return [eta, km] # Return solution and number of Hildreth algorithm iterations

################################################################################
################################################################################
def mpc_gain(Np,Nc,Ad,Bd,Cd,Dd): # Create augmented model

    [m1,n1]   = np.shape(Cd)
    [n1,n_in] = np.shape(Bd)

    A_e = np.eye(n1+m1,n1+m1)
    A_e[0:n1,0:n1] = Ad
    A_e[n1:n1+m1,0:n1] = np.dot(Cd,Ad)

    B_e = np.zeros([n1+m1,n_in],float)
    B_e[0:n1,:] = Bd;
    B_e[n1:n1+m1,:] = np.dot(Cd,Bd)

    C_e = np.zeros([m1,n1+m1],float)
    C_e[:,n1:n1+m1] = np.eye(m1,m1)

    n = n1+m1
    P = np.zeros([Np*m1,n_in],float)
    F = np.zeros([Np*m1,n1 + m1],float)
    P[0:m1,0:n_in] = np.dot(C_e,B_e)
    F[0:m1,0:n]  = np.dot(C_e,A_e)

    kk = 1
    while kk < Np:
        F[kk*m1:(kk+1)*m1,:] = np.dot(F[(kk-1)*m1:kk*m1,:],A_e)
        P[kk*m1:(kk+1)*m1,:] = np.dot(F[(kk-1)*m1:kk*m1,:],B_e)
        kk = kk+1

    Phi = np.zeros([Np*m1,Nc*n_in],float)
    Phi[:,0:n_in] = P

    i = 1
    while i < Nc:
        Phi[0:i*m1,i*n_in:(i+1)*n_in] = np.zeros([i*m1,n_in],float)
        Phi[i*m1:Np*m1,i*n_in:(i+1)*n_in] = P[0:Np*m1-i*m1,0:n_in]
        i = i + 1

    Phi_Phi = np.dot(np.transpose(Phi),Phi)
    Phi_F = np.dot(np.transpose(Phi),F)

    [mF,nF] = np.shape(Phi_F)
    Phi_R   = Phi_F[:,(nF-m1):nF]

    return [Phi_Phi, Phi_F, Phi_R, A_e, B_e, C_e]
################################################################################
################################################################################

def cont2discrete(sys, dt, method="zoh", alpha=None):

    if len(sys) == 4:
        a, b, c, d = sys
    else:
        raise ValueError("First argument must either be a tuple of 4 (ss) arrays.")

    if method == 'zoh':
        # Build an exponential matrix
        em_upper = np.hstack((a, b))

        # Need to stack zeros under the a and b matrices
        em_lower = np.hstack((np.zeros((b.shape[1], a.shape[0])),
                              np.zeros((b.shape[1], b.shape[1])) ))
        em = np.vstack((em_upper, em_lower))
        ms = expm(dt * em)

        # Dispose of the lower rows
        ms = ms[:a.shape[0], :]
        ad = ms[:, 0:a.shape[1]]
        bd = ms[:, a.shape[1]:]
        cd = c
        dd = d

    else:
        raise ValueError("Unknown transformation method '%s'" % method)

    return ad, bd, cd, dd

################################################################################
################################################################################
def obsv_matrices(Ad,Bd,Cd):
    [m1,n1]   = np.shape(Cd)
    [n1,n_in] = np.shape(Bd)

    A_e = np.eye(n1+m1,n1+m1)
    A_e[0:n1,0:n1] = Ad
    A_e[n1:n1+m1,0:n1] = np.dot(Cd,Ad)

    B_e = np.zeros([n1+m1,n_in],float)
    B_e[0:n1,:] = Bd;
    B_e[n1:n1+m1,:] = np.dot(Cd,Bd)

    C_e = np.zeros([m1,n1+m1],float)
    C_e[:,n1:n1+m1] = np.eye(m1,m1)

    A_o = A_e
    B_o = B_e
    C_o = C_e

    [r,c] = np.shape(C_e)
    Q = np.eye(c,c,dtype = 'float')

    R = np.diag([1.0, 500.0, 20.0])

    [K, So, Eo] = dlqr(np.transpose(A_o), np.transpose(C_o), Q, R)
    K_ob        = np.transpose(K)

    return [A_o, B_o, C_o, K_ob]

################################################################################
################################################################################
def limits(value, min, max):
    if value < min:
       value = min
    elif value > max:
       value = max
    return value

################################################################################
################################################################################

def depth_demand_cb(depthd):
    global depth_demand
    depth_demand = depthd.data

def speed_demand_cb(speed_d):
    global speed_demand
    speed_demand = speed_d.data

def dead_reckoner_cb(data):
    global DR
    DR  = data

def depth_onOff_cb(onOff):
    global onOFF
    onOFF = onOff.data

def compass_cb(data):
    global cur_compass
    cur_compass  = data

def simulation_cb(data):
    global simulation
    simulation = data.data
################################################################################
#### INITIALISATION ############################################################
################################################################################

if __name__ == '__main__':

    rospy.init_node('MPC_Depth_controller')

    global onOFF
    global dead
    global speed_demand
    global depth_demand
    global compass
    global backseat_driver_flag
    global simulation

    backseat_driver_flag = 0
    simulation  = False
    cur_compass = compass()
    onOFF       = False
    [depth_demand, speed_demand] = [0.0,0.0]
    DR = dead_reckoner()

    rospy.Subscriber('depth_demand', Float32, depth_demand_cb)
    rospy.Subscriber('speed_demand', Float32, speed_demand_cb)
    rospy.Subscriber('dead_reckoner', dead_reckoner, dead_reckoner_cb)
    rospy.Subscriber('Depth_onOFF', Bool, depth_onOff_cb)
    rospy.Subscriber('compass_out', compass, compass_cb)
    rospy.Subscriber('simulation_flag', Bool, simulation_cb)
    pub_tsl   = rospy.Publisher('TSL_setpoints_vertical', tsl_setpoints)
    pub_tail  = rospy.Publisher('tail_setpoints_horizontal', tail_setpoints)
    pub_C     = rospy.Publisher('DepthandSpeed_MPC_values', depthandspeed_MPC)
    pub_prop  = rospy.Publisher('prop_demand',Int8)
    pub_FandM = rospy.Publisher('Model_ForcesAndMoments',ForcesAndMoments)

    rospy.loginfo("Depth controller online")
```

```
main_control_loop()
```

Appendix B

Video survey of Lough Erne using the Delphin2 AUV

B.1 Introduction

An overview of the Delphin2 AUVs first scientific mission is given in this Appendix. The missions were organised in collaboration with the Agri-Food Biosciences Institute (AFBI) and were performed in Lower Lough Erne, Northern Ireland, Figure B.1. The primary objective for these missions was to collect video data of the Lough bottom, in various locations and depths, in order to study to the population and coverage of Zebra mussels. These are an invasive species of fresh-water shellfish that arrived in the Lough in the mid-1990's and have already significantly altered the lake ecology. The missions started on the 21st of May 2012 and lasted two weeks.

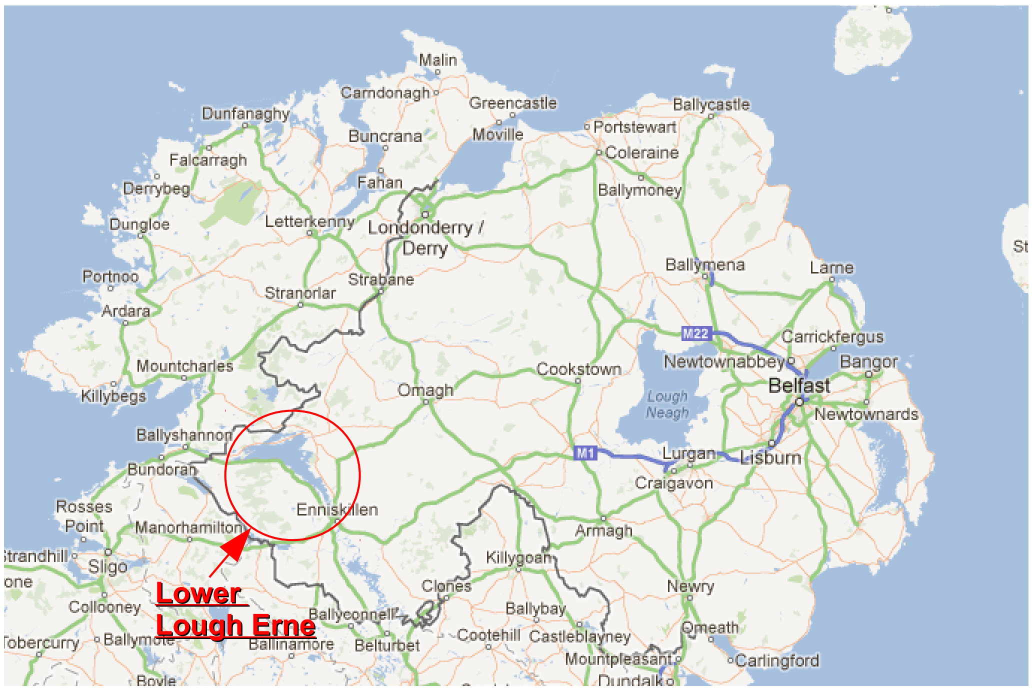


Figure B.1: Map of Northern Ireland indicating the location of Lower Lough Erne that hosted Delphin2 AUVs first science mission. Image from Google Maps.

The scientific team, three AUV operators and one marine scientist, were based at the south-east of the Lough, Figure B.2. The majority of effort was focused in Carrickreagh Bay where three successful transects of the bay were conducted. Near to the end of the two weeks, a second series of missions were performed by operating the Delphin2 AUV in ROV mode. By operating in this mode, close range video footage was acquired of an eel caught in a fyke net and of a large Zebra mussel population underneath the jetty in Carrickreagh Bay.

AFBI provided a boat and skipper for the two weeks. This served as the mission headquarters during day time, and also provided transportation for the team and AUV around the lough, Figure B.3.

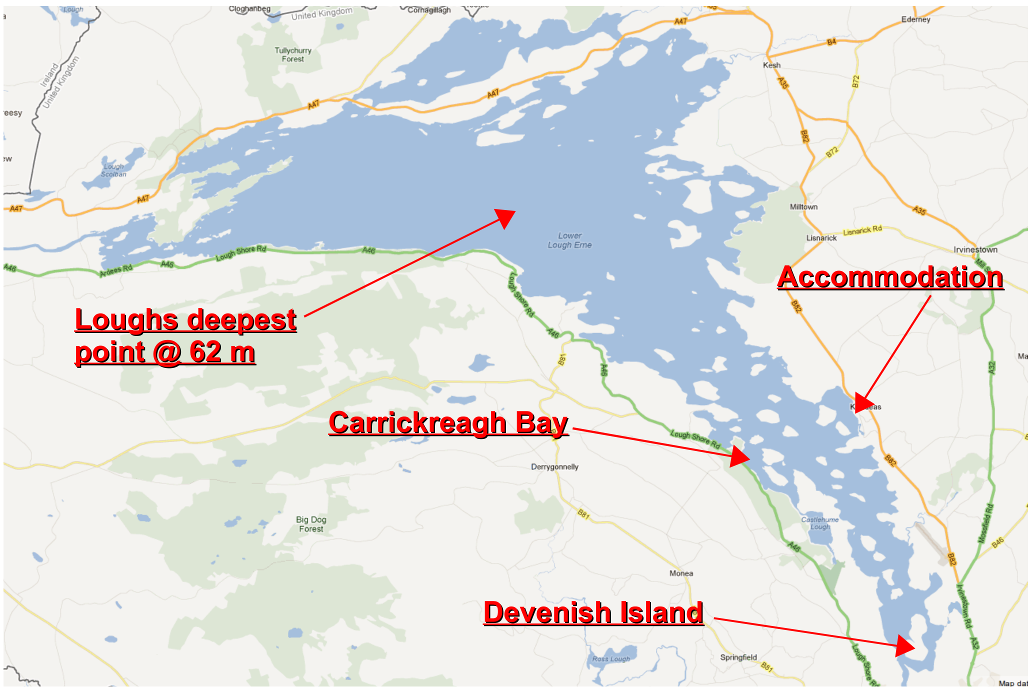


Figure B.2: Map of Lower Lough Erne, indicating were the team was based and the two main areas of study; Carrickreagh Bay and Devenish Island. Image from Google Maps.

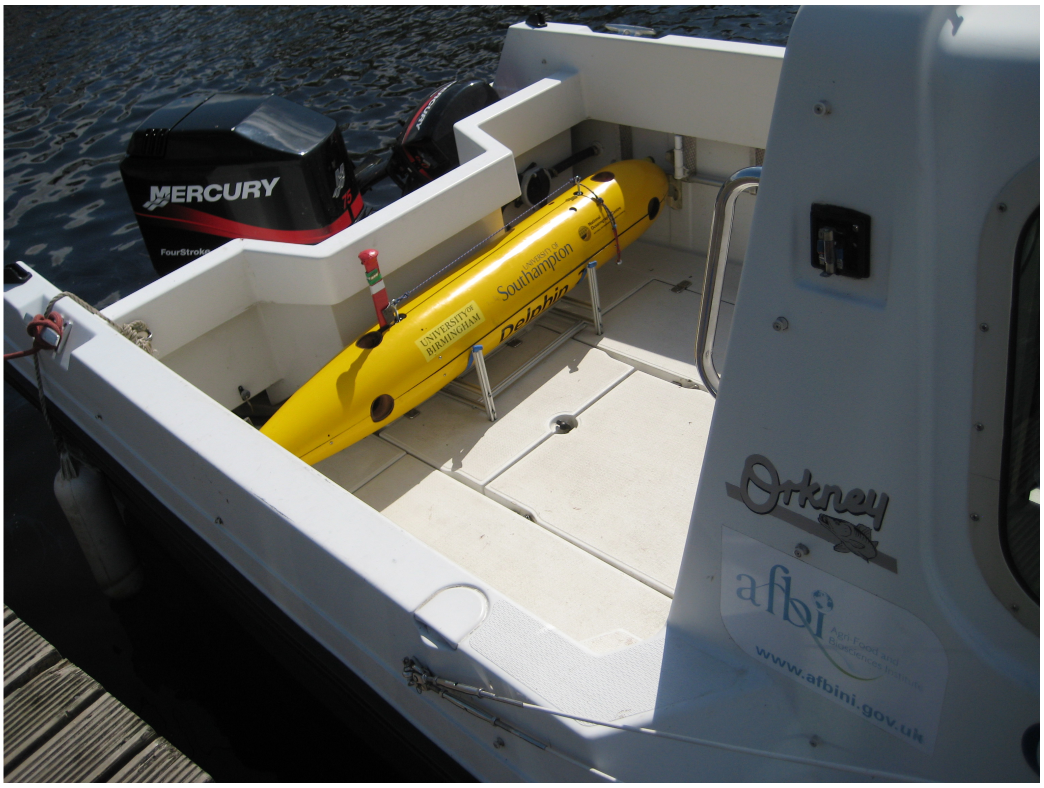


Figure B.3: The Delphin2 AUV in the back of the AFBI operated boat before sailing to the test location.

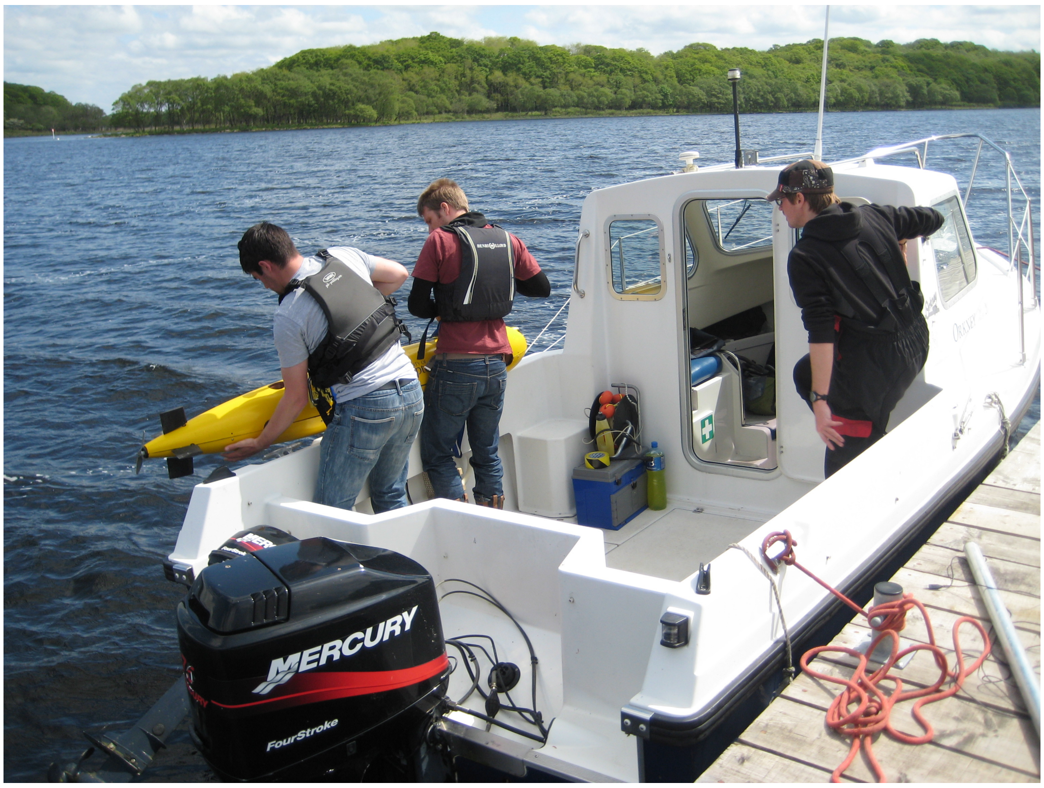


Figure B.4: Delphin2 AUV being launched off the side of the AFBI boat at Carrickreagh Bay.

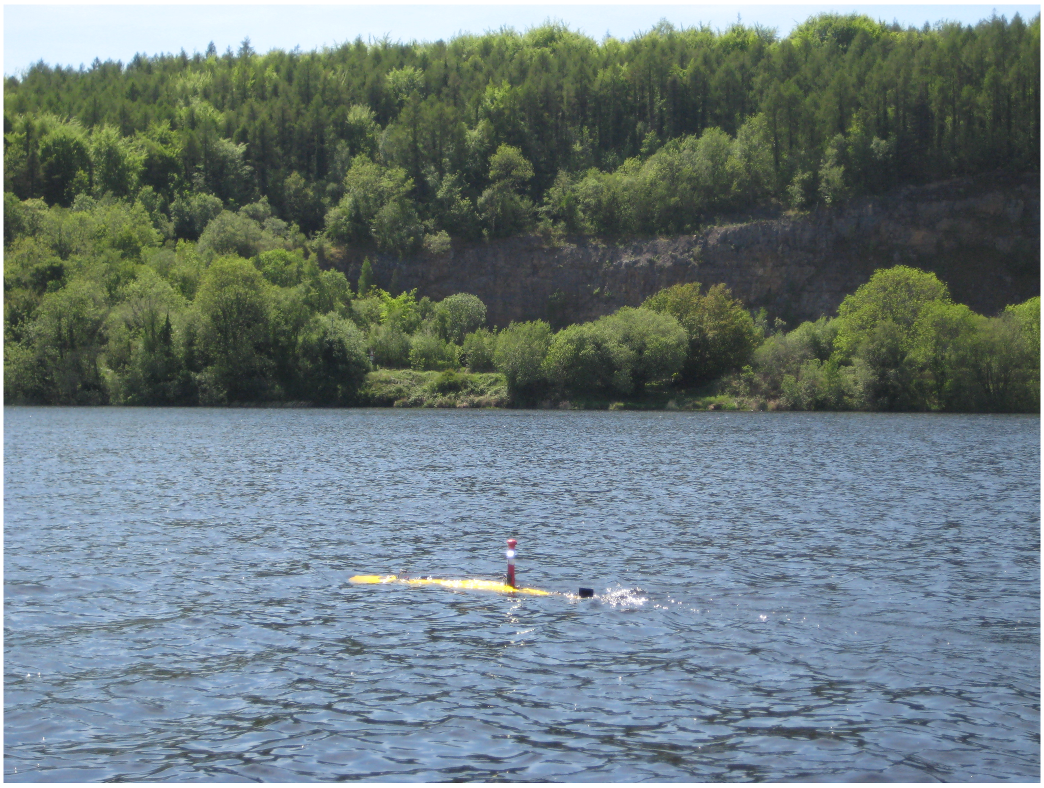


Figure B.5: Delphin2 AUV returning to the jetty in Carrickreagh Bay after a successful video survey of the lough bottom.

B.2 Video Survey of Carrickreagh Bay

The main area of effort for the two weeks was Carrickreagh Bay, Figure B.2. For all of the missions, the Delphin2 AUV was launched and recovered off the side of the boat, Figure B.4, by two people using lifting strops. The jetty was used as the start and end location for all the missions.

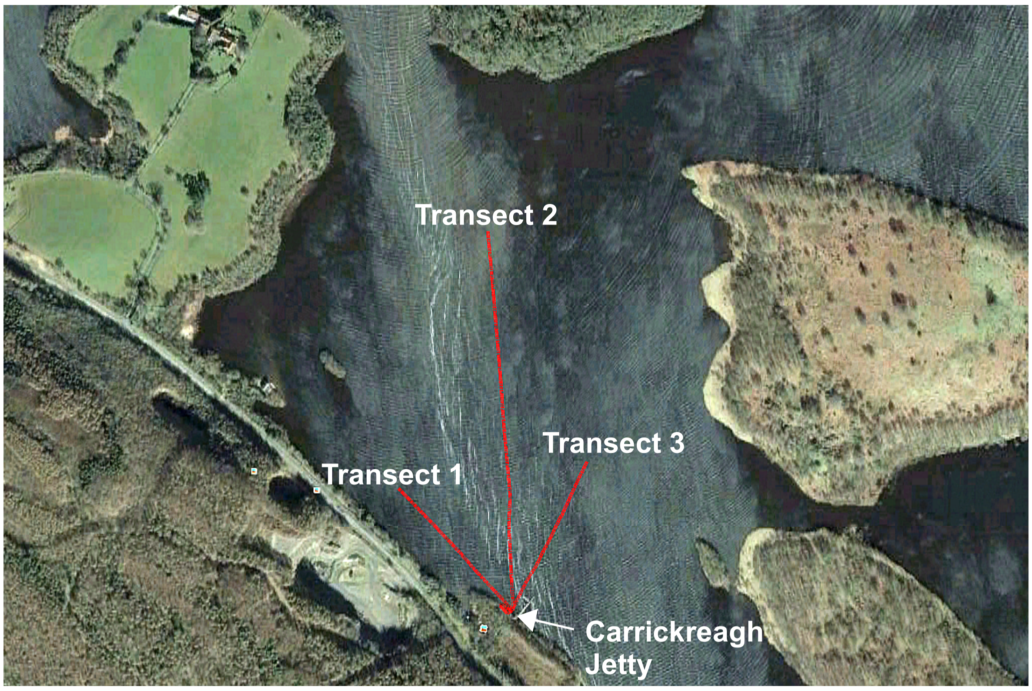


Figure B.6: Satelite image of Carrickreagh Bay. The three transects that were performed are shown in the Figure. Note the location of the jetty that served as the start and end location of each mission. Image from Google Maps.

In order to study where the Zebra mussels favoured living, three transects of the bay were performed, Figure B.6. These were chosen so as to follow the lough edge, track into the middle of the bay, and track across a small valley.

Each transect mission was split into several tasks that the AUV was required to perform:

1. Navigate, on the surface, to the start location using GPS.
2. Check depth is greater than 2.0 metres (the start location was manually chosen therefore this was a secondary check that the AUV was at the correct location).
3. Dive, with a zero surge velocity, to 1.5 metres depth.
4. Check altitude is within the expected range of 0.5 to 3.0 metres.
5. Adjust depth demand so as to achieve an altitude of 0.75 metres.
6. Switch on cameras and lights

7. Proceed at 0.4 m/s on the predefined bearing whilst maintaining 0.75 metres altitude.
8. Once a time limit for that transect has completed, switch off cameras and lights then return to the surface and await a valid GPS fix.
9. Return, on the surface, to the jetty.
10. Mission complete.

B.2.1 Vehicle Performance

Figures B.7, B.8 and B.9 present the results of the Delphin2 AUVs altitude tracking, along with water temperature data, for transects 1, 2 and 3 respectively. Note that depth has been inverted so as to enable better visualisation of the data. All three Figures show that the AUV performed extremely well at maintaining an average altitude of 0.75 metres, especially when the water depth was increasing.

When the AUV was tasked with tracking the lough floor and the as the water depth started to decrease, the altitude controller did not perform as well. As a result, a few collisions with the lough bottom did occur but thanks to the low speed of the vehicle and the relatively soft lough bottom, the AUV was not damaged. The cause of this is that only the echo sounder was being used to measure altitude as the scanning sonar was not fully integrated. Therefore, the AUV did not have any obstacle avoidance and so only reacted after it had overshot its altitude demand.

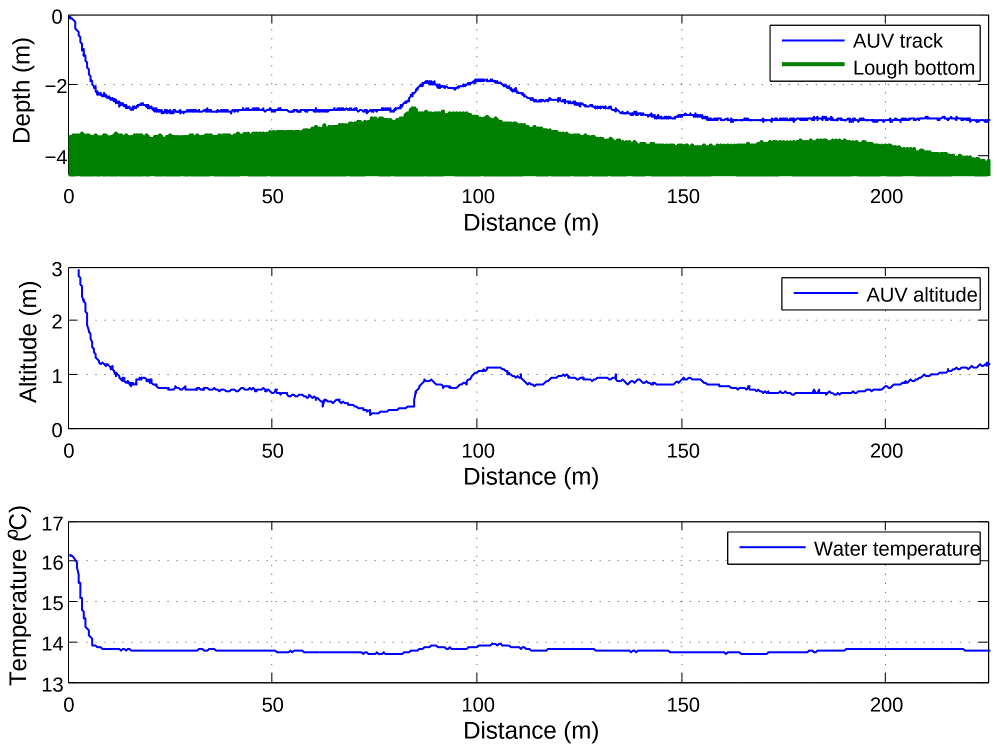


Figure B.7: Depth, altitude and temperature data during transect 1.

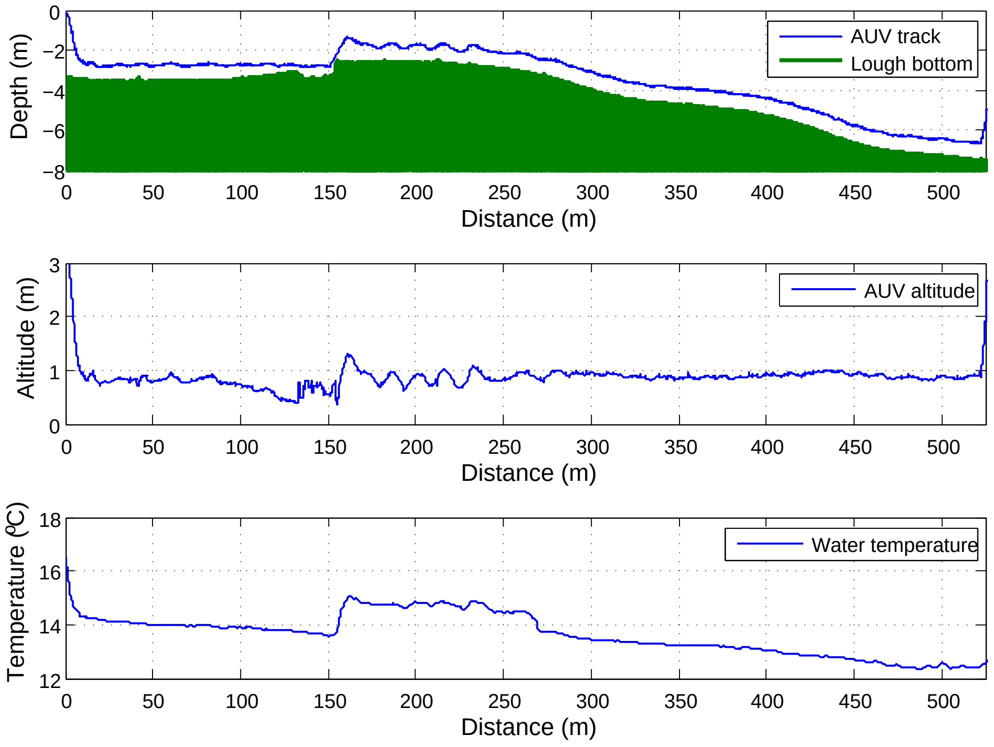


Figure B.8: Depth, altitude and temperature data during transect 2.

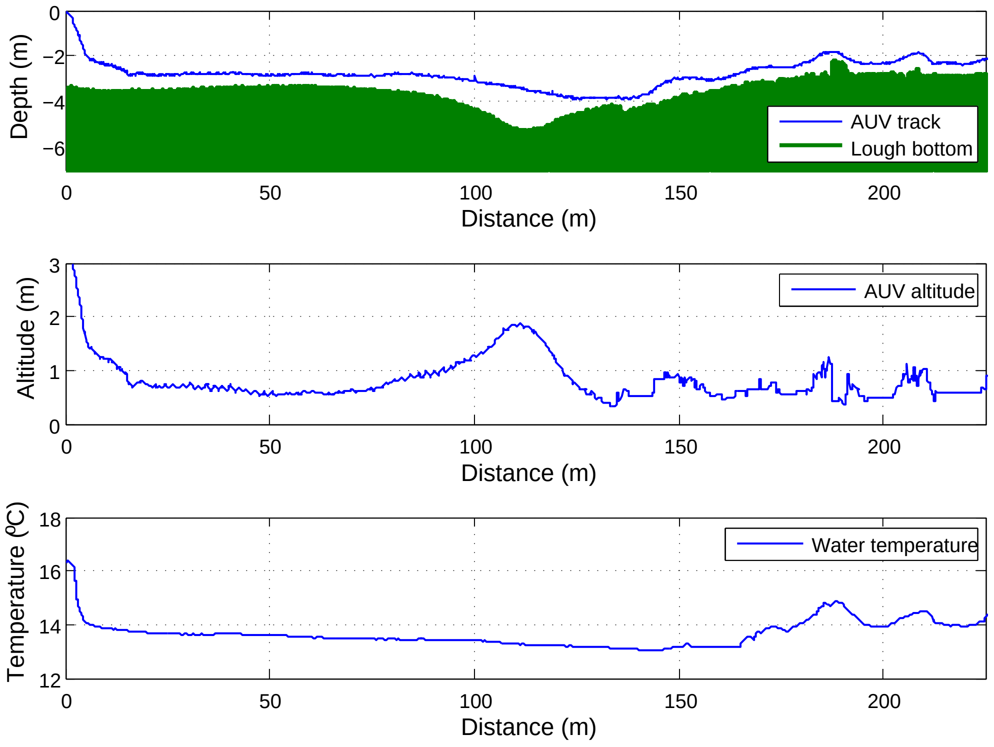


Figure B.9: Depth, altitude and temperature data during transect 3.

For future surveys that require altitude tracking, information from the forward scanning sonar must be effectively utilised by the altitude control algorithm. It is also recommended that the echo sounder be upgraded to a model that provides reliable altitude information when operating near to the bottom. The current echo sounder starts to output erroneous information when altitude falls below 0.5 metres. These errors can cause the altitude control algorithm to fail.

B.2.2 Video Data

The purpose of these transects was to collect video data of the Lough bottom. Figures B.10 and B.11 present video frames, from the downwards and forwards facing cameras, captured during the three different transects.

It was not difficult to locate the Zebra mussels underwater. Almost every frame, when the lough bottom was visible, included a Zebra mussel. Frame 656 in Figure B.10 shows four small fresh-water sponges and a small bed of Zebra mussels. In Frame 731, a bed of feeding Zebra mussels are shown with their funnels out. Frame 4426 is a mix of dead and alive Zebra mussels and Frame 9902 shows a large 'carpet' of Zebra mussels.

In Figure B.11 four frames from the forwards looking camera are presented. The information in these frames is similar to that in Figure B.10, showing fresh-water sponges and scattered beds of Zebra mussels. The forward looking camera did show better three-dimensional perspective in comparison to the downwards camera however, due to the longer range of each shot, it was more susceptible to poor water visibility.

B.3 Delphin2 ROV Operations

A secondary mission that the Delphin2 AUV was tasked with was to operate as an ROV in order inspect the underneath of the Carrickreagh jetty and examine a Fyke net that had been laid for 24 hours. Both tasks required remote control by a human operator neither navigation under the jetty, nor locating the fyke net, were possible given the Delphin2 AUVs current sensors and AI.

An ethernet cable was used to directly connect the AUV to the operators laptop and a gaming controller was used as the user interface. Both the forwards and downwards video streams were available to the operator as well as position and actuator information.

Figure B.12 presents images taken by the Delphin2 AUV whilst operating in ROV mode. The top two images are the stanchions of the Carrickreagh jetty and show large beds of Zebra mussels, in particular at the bottom of the stanchions. The bottom two images are of a fyke net that had successfully caught an eel.

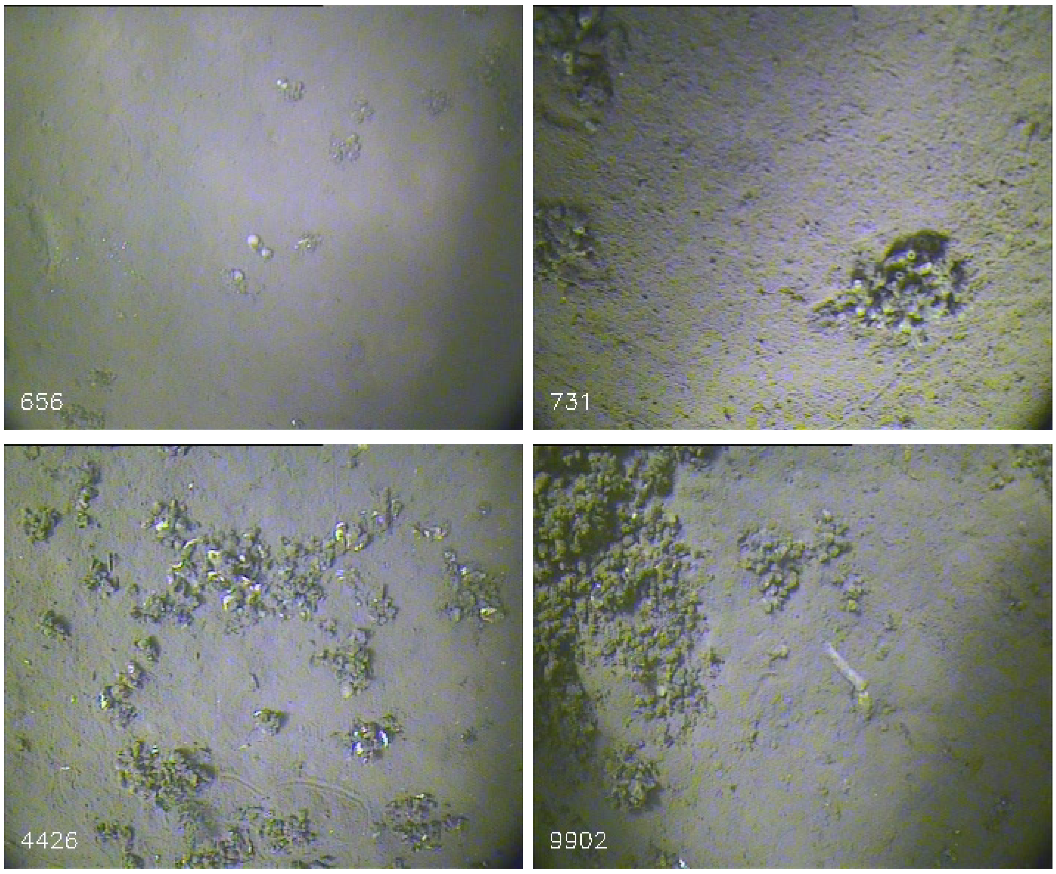


Figure B.10: Video frames, from the downwards looking camera, from the three transects. Frame 656: Small beds of Zebra mussels and a type of fresh-water sponge. Frame 731: Beds of Zebra mussels feeding using their funnels. Frame 4426: Bed of Zebra mussels. Note the white shells are the remains of dead mussels. Frame 9902: Large bed of Zebra mussels and an unidentified object.

B.4 Conclusions

The Delphin2 AUV showed excellent competence at performing scientific missions. The ability to track the lough bottom, at a very low altitude and speed, led to some very interesting underwater footage. Given further development of the navigation and planning parts of the vehicles software, it is thought that the Delphin2 AUV could perform similar missions without the aid of a boat.

The data that was collected, although interesting, is not of great scientific importance for two primary reasons; the cameras are very low quality (resolution of 352x288 pixels) and the navigation inaccuracies are too great. However, the concept that Delphin2 was designed for, has been verified.

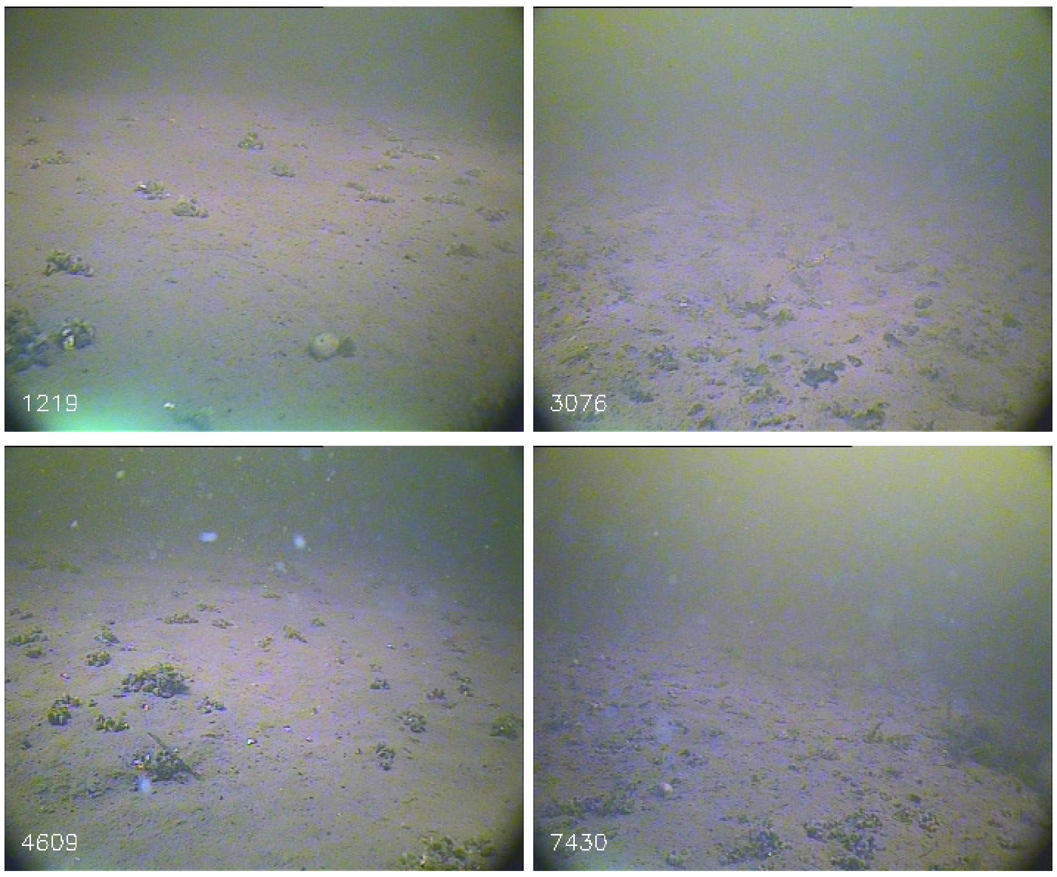


Figure B.11: Video frames, from the forwards looking camera, from the three transects. Frame 1219: White object at the bottom of the image is a fresh-water sponge. Frames 3076, 4609, and 7430: Small scattered beds of Zebra mussels. Note the forwards facing camera gives better 3D perspective than the downwards camera.

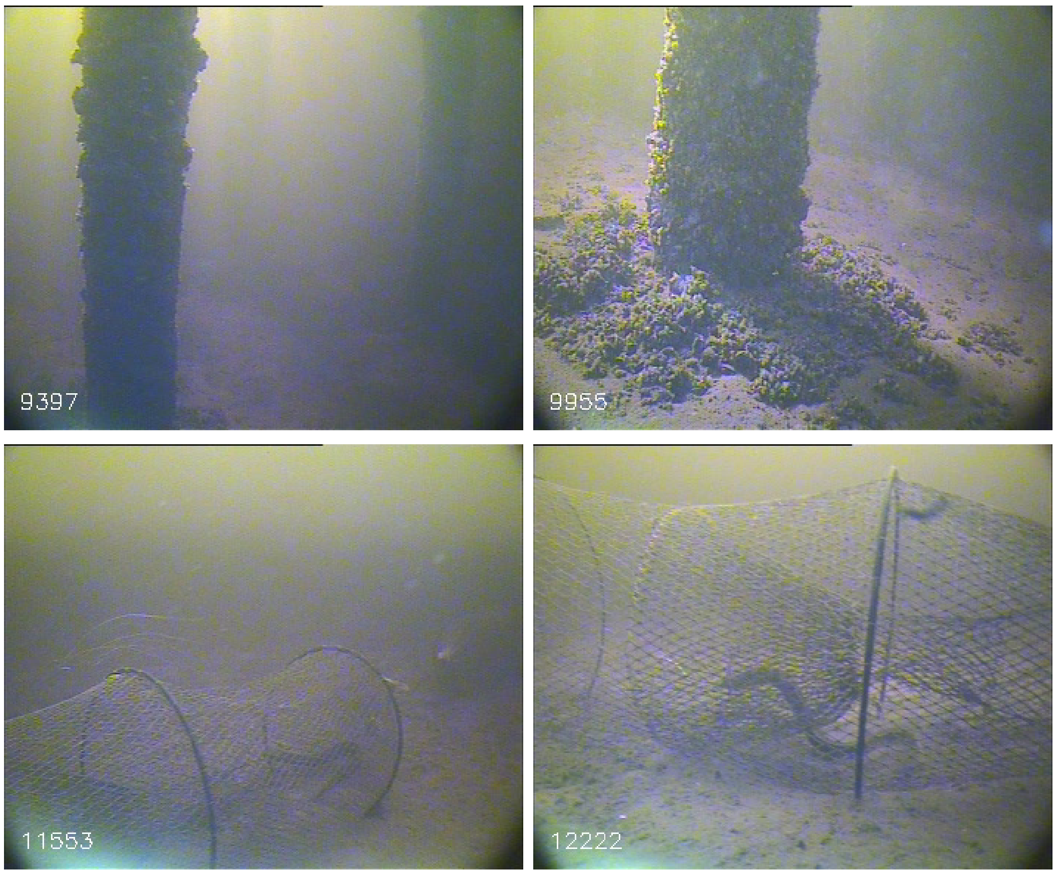


Figure B.12: Video frames, from the forwards looking camera, whilst operating the Delphin2 AUV in ROV mode. Frames 9397 and 9955: Carrickreagh jetty stanchions. Note the vast coverage of Zebra mussels, in particular at the bottom of the stanchion. Frames 11553 and 12222: Fyke net, located to the north of Devenish Island, with a captured eel.