MARINE ROBOTIC SYSTEMS WITH PLYMOUTH UNIVERSITY

Energy Based Control System Designs for Underactuated Robot Fish Propulsion

Daniel Thomas Roper

School of Marine Science and Engineering

University of Plymouth

A thesis submitted to the University of Plymouth in partial fulfilment of the requirements for the degree of:

Doctor of Philosophy

April 2013

Abstract

Energy Based Control System Designs for Underactuated Robot Fish Propulsion

Daniel Thomas Roper

In nature through millions of years of evolution fish and cetaceans have developed fast efficient and highly manoeuvrable methods of marine propulsion.

A recent explosion in demand for sub sea robotics, for conducting tasks such as sub sea exploration and survey has left developers desiring to capture some of the novel mechanisms evolved by fish and cetaceans to increase the efficiency of speed and manoeuvrability of sub sea robots.

Research has revealed that interactions with vortices and other unsteady fluid effects play a significant role in the efficiency of fish and cetaceans. However attempts to duplicate this with robotic fish have been limited by the difficulty of predicting or sensing such uncertain fluid effects. This study aims to develop a gait generation method for a robotic fish with a degree of passivity which could allow the body to dynamically interact with and potentially synchronise with vortices within the flow without the need to actually sense them.

In this study this is achieved through the development of a novel energy based gait generation tactic, where the gait of the robotic fish is determined through regulation of the state energy rather than absolute state position. Rather than treating fluid interactions as undesirable disturbances and 'fighting' them to maintain a rigid geometric defined gait, energy based control allows the disturbances to the system generated by vortices in the surrounding flow to contribute to the energy of the system and hence the dynamic motion.

Three different energy controllers are presented within this thesis, a deadbeat energy controller equivalent to an analytically optimised model predictive controller, a H_∞ disturbance rejecting controller with a novel gradient decent optimisation and finally a error feedback controller with a novel alternative error metric. The controllers were

tested on a robotic fish simulation platform developed within this project.

The simulation platform consisted of the solution of a series of ordinary differential equations for solid body dynamics coupled with a finite element incompressible fluid dynamic simulation of the surrounding flow. results demonstrated the effectiveness of the energy based control approach and illustrate the importance of choice of controller in performance.

Contents

List of Figures

Acknowledgements

I am deeply indebted to my supervisory team, Dr Sanjay Sharma, Prof Robert Sutton and Dr Philip Culverhouse without the support and advice of whom I would never have completed this project. I particularly wish to acknowledge the amazing patience of Dr Sharma and Prof Sutton for what must been an aeon they have spent on reading then giving me feedback and guidance on the all too slowly improving iterations I have produced of this work and all my other publications to date.

I would also like to thank my wonderful girlfriend Xiaoqing for her patience, support and pack lunches. She has truly mastered making the most of what is left of a candle burnt at both ends.

I would like to express my gratitude to my family who have been an endless font of support over the years, specifically I would like to thank my mother who's stubborn determination has always been an inspiration when my moral was low.

I would like to thank all the teachers I have had over the years who believed in me, often far more than I believed in myself.

Authors declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award.

Relevant scientific seminars and conferences were regularly attended at which work was often presented. Several papers have been published in refereed journals.

Signed: _____

Date: _____

Word count for the main body of this thesis: 30211

Nomenclature

Throughout this thesis the following convention has been followed for all algebraic symbols,

a	scalar valued identity
$\mathbf{a}$	vector or time series valued identity
$\mathbf{A}$	matrix valued identity or system identity

Unless otherwise all symbols and abbreviations found throughout this thesis will follow the bellow convention.

$\mathbf{I}$	identity matrix
Rf	linear reaction force
τ	pure torque
m_i	mass of body i
c_d	steady state coefficient of drag
θ_i	bearing of body i
$\mathbf{q}$	vector of body bearings
$\dot{\mathbf{q}}$	vector of body angular velocities
$\ddot{\mathbf{q}}$	vector of body angular accelerations
J	inertia of a body
$\mathbf{J}$	matrix of inertia
$\mathbf{e}$	measured error
$\hat{\mathbf{e}}$	control vector error
$\tilde{\mathbf{e}}$	best case control vector error
$\mathbf{u}$	controlable system input
$\mathbf{w}$	uncertain system input
$\mathbf{t}$	time
E	energy
cm	length centre meters
ms^{-1}	velocity in meters per second
ms^{-2}	acceleration in meters per second per second
Ls^{-1}	velocity in body lengths per second

$rad\ s^{-1}$	angular velocity radians per second
Hz	frequency in Hertz
W	energy in Watts
sec	time in seconds
MPa	pressure Mega Pascals
$RMSE$	root mean square error
UUV	unmanned underwater vehicle
ROV	remote operated vehicle
AUV	autonomous underwater vehicle
BCF	body and caudal fin
PMF	paired and median fin
CFD	computational fluid dynamics
LTI	linear time invariant
DC	direct current
SMA	shape memory alloy
CNT	carbon nano tube
IPMC	ionic polymer metal composites
DEA	dielectric elastomer actuator
PID	proportional integral derivative
BCCVE	best case control vector error

Chapter 1

Introduction

This chapter aims to introduce and justify this study.

1.1 Motivations

In recent years an increase in oceanographic engineering projects such as sub sea cables, pipelines and deep sea oil and gas drilling, combined with an increased interest in environmental awareness, has led to a demand for new tools for performing sub ocean tasks. Whilst manned submersibles have been in existence for some time, the consideration of human life support drives up complexity and cost. As well as placing limitations on maximum mission time. Furthermore certain sub sea tasks involve a high risk factor and it is desirable to remove the presence of humans. The solution has been the development of *unmanned underwater vehicles* (UUVs) and more recently a subclass of UUVs called *autonomous underwater vehicles* (AUVs) the definition of which being any self contained UUV that can operate without connection to an external power source or real time operational commands. The advantage of AUVs over more restricted tethered UUVs often referred to as *remote operated vehicles* (ROVs) is increased range, and reduced operating costs. AUV support vessels can operate in multiple sites as there is no need for real-time contact with the devices for control. In the offshore industry AUVs are used for tasks such as ocean floor topographical surveying, pipe or cable inspection and chemical pollution

sampling (Danson 2003). As well as private sector interests, in the public sector AUVs are used for military surveillance/reconnaissance, mine disposal, harbour patrolling, oceanographic seismology and ocean temperature monitoring (Corfield and Hillenbrand 2003). With such a wide variety of applications it is little surprise that AUVs come in a wide variety of shapes and sizes.

The design of AUVs has a direct effect on factors such as speed, manoeuvrability, range, reliability and general robustness. All factors which directly affect operational costs and thus commercial viability. In order to improve all of these factors developers are looking towards the growing subject of biomimetics. Biomimetics, also sometimes called biomimicry or bionics, broadly refers to the deliberate imitation of nature in man-made systems (Benyus 1997) , (Siochi et al. 2002).

The abundance of life in the Earth's oceans provides no shortage of suggestions for locomotion and manoeuvring tactics for a sub sea environment. Fish and cetaceans after millions of years of evolution have developed impressive speed and agility in sub sea locomotion. Tuna being an excellent example, able to outperform any man-made vehicle relative to its size in speed and turning ability (Triantafyllou and Triantafyllou 1995). This makes fish a natural choice for bioinspiration in the design of AUVs of the future.

Over the past few decades a significant number of prototype devices with biomimetic marine propulsion systems have been developed with various motives. These prototypes are loosely referred to in the literature as 'robotic fish'. Within this study robotic fish is used as a generic term applied to UUVs with designs incorporating sufficient bioinspiration from fish to be deemed 'fishlike'.

Research into the biomechanics of fish swimming has suggested that in nature fish can harvest energy from the turbulence in the surrounding fluid to reduce the effort of swimming (Beal 2003). Biomimicry of such an effect could dramatically increase

range and reduce operational costs of AUVs.

In order to achieve turbulent energy capture a gait generation tactic is needed that can take energy contribution from the disturbances supplied by the surrounding fluid. However the current traditional geometric positional control process effectively fights to resist the effects of external disturbances, meaning that external disturbances result in an increase in energy cost. By controlling the motion of a robotic fish through energy rather than through absolute positional commands, energy gained through fluid interactions will be allowed to contribute to the motion, thus reducing total energy cost.

The overall aim of this study is to develop biomimetic marine propulsion capable of mimicking the way fish and cetaceans in nature harness unsteady fluid effects to increase propulsion efficiency in terms of velocity and energetic cost of transport.

1.2 Objectives

This study has four key objectives in order to reach the above stated aim;

1. Review the existing state of the art of biomimetic propulsion systems, identifying key trends and omissions in the present art. This review should identify key characteristics of a robotic fish including morphology, actuation mechanisms and specific sources of bioinspiration.

2. Develop a model of an underactuated robotic fish of sufficient realism to be considered an effective controller test bed. The model should be as generalizable as reasonably possible so that it can be easily adapted to different designs.

3. Develop an energy based gait generation approach which can produce an effective swimming gait for an underactuated robotic fish

4. Develop and assess strategies for the control of energy. The output of the assessment should recommend a choice of energy controller for the application

of gait generation for an underactuated robotic fish.

1.3 Contributions

This study is considered to have made the following contributions to knowledge;

1. Proof of concept simulation study demonstrating that the control of state energy can result in effective swimming gait for an underactuated robotic fish.

2. Derivation of a differentiable explicit function for an upper bound of the H_∞ norm of a realisable linear time invariant system. The derivative of which being usable to asses parametric sensitivity of system norms and hence apply gradient based optimisation.

3. Definition of an alternative error metric for use with nonlinear control objectives that allows linear error feedback controllers to be applied to nonlinear control objective.

4. Derivation of an explicit function of state to determine the local value of the aforementioned alternative error metric for a state energy objective.

1.4 List of publications

Published Journal Articles :

Roper, D. Sharma, S. Sutton, R. & Culverhouse, P; *Oscillation and direction control strategies for a robotic fish* Underwater Technology Journal **vol. 31, no 2, pp. 67-76, 2013.**

Roper, D. Sharma, S. Sutton, R. & Culverhouse, P; *Energy-shaping gait generation for a class of underactuated robotic fish.* Marine Technology Society Journal. **vol. 46, no 3, pp. 34-43, 2012.**

Roper, D. Sharma, S. Sutton, R. & Culverhouse, P; *A review of developments*

towards biologically inspired propulsion systems for autonomous underwater vehicles. IMechE part M: Journal of Engineering for the Maritime Environment. **vol. 225, no 2, pp. 77-96, 2011.**

Published Conference Papers :

Roper, D. Sharma, S. Sutton, R. & Culverhouse, P; *Stratergies for control on a simplified model of a robotic fish* Proc IFAC Workshop - Navigation, Guidance and Control of Underwater Vehicles (NGCUV 2012) **Porto, Apr 10-12, 3302**

Roper, D. Sharma, S. Sutton, R. & Culverhouse, P; *Energy based gait control of an underactuated robotic fish* The 14^{th} International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR 2011) **Paris, Sep 6-8, 2011, pp. 135-141**

Roper, D. Sharma, S. Sutton, R. & Culverhouse, P; *A harmonic energy based gait production strategy for an underactuated robotic fish* UK Marine Technology Postgraduate Conference (UK MTPC 2011) **Southampton, Jun 9-10**

Submitted for Publication :

Roper, D. Sharma, S. Sutton, R. & Culverhouse, P; *A novel error metric for feedback energy control of a robotic fish* International Journal of Control **Submitted Dec 2012**

Roper, D. Sharma, S. Sutton, R. & Culverhouse, P; *H_∞ parametric sensitivity of LTI systems norms for gradient optimisation* IEEE Transactions on Automatic Control **Submitted Apr 2013**

1.5 Outline of thesis

The remainder of this thesis is divided into nine further chapters.

Chapter 2 contains an introduction to the mechanisms and theory of swimming of

fish and cetaceans in nature. Identifying and discussing key mechanisms employed by biological swimmers that are desirable to be emulated in biomimetic swimmers.

Chapter 3 presents a detailed review of the current state of the art in the field of biomimetic marine locomotion. Within trends in biomimetic propulsion systems are assessed and conclusions presented on the likely direction of future developments.

Chapter 4 begins with a generalised model for a simplified planar robotic fish. This generalised model is then used to provide a specific model for the geometry used throughout this thesis. Simulation methods used throughout this study to integrate fluid solid interaction are also described in detail.

Chapter 5 discusses the use of energy based control to regulate higher dimensional motions on an underactuated robotic fish. It is also demonstrated that an arbitrary realisable swimming gait can be described as a constant energy orbit.

Chapter 6 describes a deadbeat energy controller for the robotic fish. Results are presented which demonstrate that the energy control based gait generation approach can result in an effective swimming gait. However results presented also suggest that further optimisation of the energy controller could result in further improvement to the approach.

Chapter 7 presents an improved alternative controller for state energy based on robust H_∞ disturbance rejection and error feedback. The chapter goes on to describe novel parametric norm sensitivity approach. An explanation is given as to how this norm sensitivity approach was used further to optimise the robustness and fragility of the disturbance rejection.

Chapter 8 defines an alternative error metric specifically for use with error feedback control with nonlinear control objectives. The chapter also details an explicit function of state to quantify this alternative metric for the control of state energy.

Chapter 9 evaluates and compares results presented in chapters 6, 7 and 8. Comparing the suggested controllers fitness for the purpose of energy based gait generation for a robotic fish.

Chapter 10 finally presents concluding remarks and suggests areas for further study based on the findings presented within this thesis.

Chapter 2

Swimming in Nature

This chapter will introduce the mechanisms and theory of swimming of fish and cetaceans in nature.

2.1 Introduction

The aim of this study stated in chapter 1 is to develop biomimetic marine propulsion capable of mimicking the way fish and cetaceans in nature harness unsteady fluid effects to increase propulsion efficiency in terms of velocity and energetic cost of transport. The goal of this chapter is to establish a base understanding of the swimming mechanisms utilised in nature in order to select potential candidate mechanisms for biomimicry.

Fluid dynamic forces are generally either a result of viscosity effects or pressure; the comparative dominance of each in a given flow being determined by the Reynolds number (Re). Fluid effects with low Reynolds numbers i.e. $Re < 1$ are dominated by viscous forces whereas fluid effects with higher Reynolds numbers i.e. $Re \gg 1$ are dominated by pressure forces. In fluid dynamics this distinction is usually made by referring to low Reynolds number flows $Re < 1$ as belonging to the Stokesian realm and high Reynolds number flows $Re \gg 1$ as belonging to the Eularian realm referring to the dominant term in the Navier Stokes descriptive equations.

Although there are examples of biological swimmers that operate within the the

Stokesian realm, such swimmers are generally on a microscopic scale such as bacteria or sperm (Childress 1981). Therefore this study will focus on biological swimmers in the Eularian realm as the scale of most mechanical AUVs in current operation ensures that they operate well within the Eularian realm when operating in water. However Stokesian realm swimming mechanisms may well be of interest in developing devices for applications where high viscosity fluids such as residual fuel oils are involved.

There exists a significant body of literature relating to the bio-mechanics and hydrodynamics of fish and cetacean locomotion in the Eularian realm, based on observation of kinematics and surrounding flow through methods such as milk/ink plume tracing (Rosen 1959) and in the last few decades *digital particle image velocimetry* (DPIV) (Anderson 1996),(Wolfgang et al. 1999).

The remainder of this chapter is divided into six further sections. Section 2.2 will detail the categorization of swimming modes utilised in nature. Whereas section 2.3 will discuss the roles of lift and drag in swimming. Section 2.4 will discus methods employed by fish in nature to generate efficient periodic motion. Section 2.5 will mention some of the mechanism employed by fish for rapid acceleration and high speed manoeuvring. Section 2.6 will highlight the importance of vortices in the current theory of fish propulsion and finally section 2.7 will present some concluding remarks.

2.2 Summary of biological swimming modes

This section aims to introduce briefly some of the fundamental swimming modes found in nature.

The first distinction made between biological swimming modes is between *body and/or caudal fin* (BCF) type swimming and *paired or median fin* (PMF) swimming.

BCF swimming refers to swimming modes that generate thrust through the use of a translational wave propagated along a portion of the body and translated onto the caudal

Table 2.1: The classification of BCF swimmers Lindsey (1978)

Classification	% of body in wave
Anguilliform	$\geq 70\%$
Sub-Carangiform	50-70%
Carangiform	30-50%
Thunniform	≤ 30
Ostraciform	Caudal fin only

fin which acts as a propulsive surface.

BCF swimmers are often sub-categorized further according to the proportion of the body involved in the propulsive wave as shown in table 2.1. Typically with a BCF swimmer the larger the proportion of the body involved in the propulsive wave the greater the manoeuvrability. However the smaller the proportion of the body involved in the propulsive waves the greater the efficiency and speed of locomotion.

Typical BCF swimmers are capable of rapid swimming at speeds in the order of 10 *body length per second* (Ls^{-1}), (Bainbridge 1958), and rapid turning, often taking much less than 1 body length to turn 180 degrees.

PMF swimming refers to swimming modes that achieve locomotion through the actuation of paired pectoral fins, dorsal fins, anal fins or paired dorsal and anal fins, the classification of which can be found in Figure 2.1.

Typical PMF swimmers are capable of precision manoeuvring with 6 degrees of freedom, including station keeping and reversing manoeuvre.

A more complete review of fish locomotion modes can be found in, (Sfakiotakis et al. 1999).

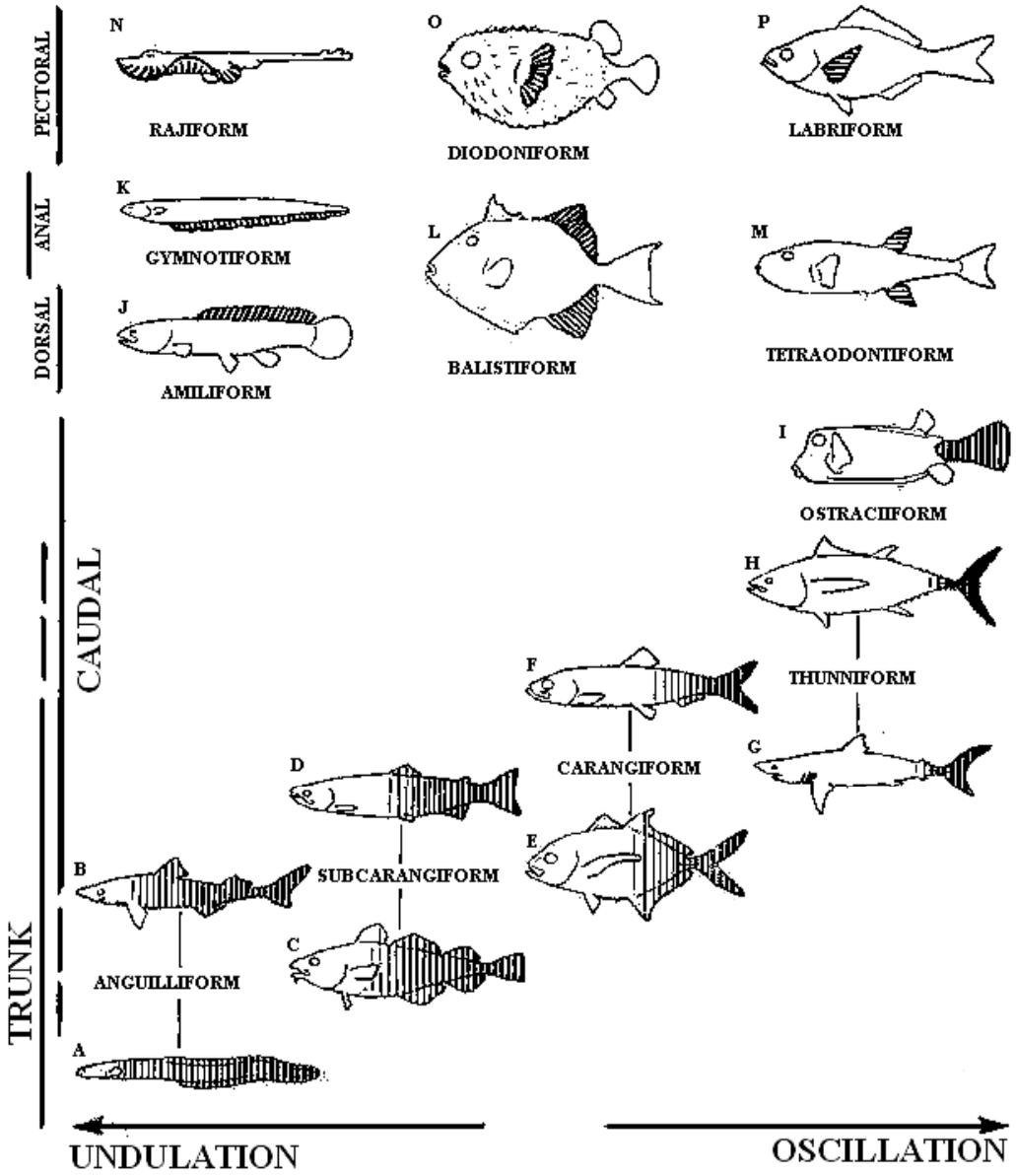


Figure 2.1: The Classification of PMF swimming by fin and movement type (Sfakiotakis et al. 1999) Copyright ©1979 Elsevier (Permission to reproduce this image has been obtained through RightsLink®)

Figure 2.2: Black Ghost Knife Fish (Aquarium 2011), Copyright ©2011 National Aquarium (Permission to reproduce this image has been granted by National Aquarium)

2.3 Lift and Drag

Lift and drag mechanisms play a large part in the efficiency of swimming motion (Fish 1996). Generally undulation based propulsion is more drag based and oscillation based propulsion is more lift based. Fins designed for lift based propulsions such as the caudal fin of the blue fin tuna are smooth and hydrodynamic.

However figure 2.2 shows a photograph of a black ghost knife fish which utilises drag gymnotiform type propulsion. As can be seen the black ghost knife fish has vertical ridges on the long anal fin almost orthogonal to the direction of motion. These ridges act to increase the drag. Suggesting that undulation based swimming methods may benefit from an increase in drag which reduces the slip of the fluid conveyor action.

2.4 Periodic Motion

Like most biological creatures, both BCF and PMF swimmers rely on a periodic motion or gait to generate locomotive thrust. It has been demonstrated that many fish use passive muscles to generate torsional elasticity along the body (Pabst 1996). This gives rise to the model of a fish as a spring oscillator. Estimates suggest that passive muscle elasticity can result in up to 30% energy savings (Harper et al. 1998).

It is thought that by altering the state of excitation of the muscle fish can control the stiffness of their body and hence change the frequency and amplitude of the gait (Long and Nipper 1996).

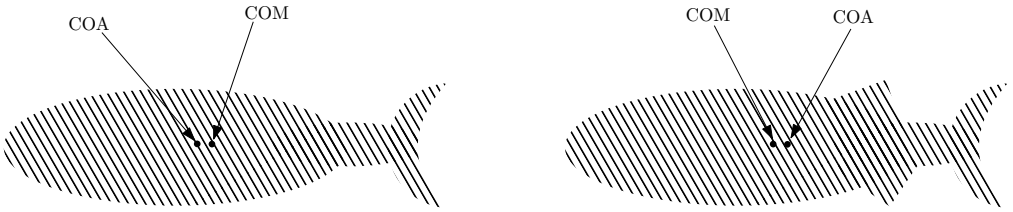
Since propulsion energy costs are a primary limiting factor on many vehicles any mechanisms that can reduce the cost of transport are of significant interest to biomimetic swimming mechanism design.

2.5 Manoeuvrability

Whilst PMF type swimming modes offer increased manoeuvrability at low speeds, many BCF fish employ novel kinematics to achieve rapid acceleration and high speed manoeuvres.

Cetaceans and other carangiform and subcarangiform swimmers often employ the so called ‘C’ or ‘S’ start swimming kinematics to achieve rapid acceleration (Spierts and Leeuwen 1999).

Whilst swimming at speed many BCF swimmers are or can make themselves dynamically unstable in yaw, allowing them to make rapid changes in direction (Weihs 2001), (Fish 2002). This can be achieved if the *centre of cross-section area* (COA) (i.e. the centre of area the longitudinal vertical planar central cross section) is forward of the *centre of mass* (COM). By employing variable dorsal and anal fins fish are able to



(a) (b)

Figure 2.3: Digaram showing *centre of area* (COA) relative to *centre of mass* (COM) ; (a) Fins retracted, (b) Fins deployed

adjust the position of the centre of cross section area relative to the centre of mass and hence the level of yaw stability as shown in figure 2.3.

Assuming that the friction and drag forces are aligned in the opposite direction to the direction of motion and act approximately at the COA. Also assuming the body pivots around the COM. Then if the COA is aft of the COM then the presence of yaw will cause the drag forces to create a course correcting moment. However if the COA is forward of the COM then the drag forces will cause a course destabilising moment.

This is of significant interest in the design phase of biomimetic swimming devices. By closely emulating the morphology of biological fish, biomimetic devices will be potentially able to achieve high yaw rates at speed dramatically increasing high speed manoeuvrability.

2.6 Vortecies in Swiming

Early experiments involving towing euthanized dolphins at anecdotal swimming speeds suggested that dolphins appear to have a 7 fold shortage of muscle mass to generate sufficient power to overcome drag forces (Gray 1936). Although it has been shown that these findings often referred to as 'Gray's Paradox' grossly over estimate the shortfall (Lee et al. 2009) and argued that many of the assumptions made in the original study were fundamentally flawed (Fish 2006). There is a general consensus that novel

drag reduction methods are utilised by dolphins and other BCF type swimmers to actively reduce drag. This consensus has lead to a great deal of research into BCF type swimming mechanism used in nature (Barrett et al. 1999).

Early theories of fish swimming were based on lamina lift and drag fluid interaction assumption, however the development of flow visualization techniques demonstrated that vorticity plays an important role in generation of propulsive forces.

Photographs of fish swimming in water over a thin layer of milk covering a dark background, have revealed the generation of vortices and their propagation along the fish's body. From these photographs it was theorised that these vortices act as pegs against which the fish can push its body to gain thrust (Rosen 1959).

It has been shown that not only can fish utilise vortices generated by their own movement to generate thrust, but they can also utilise vorticity present in the surrounding flow to reduce muscle expenditure. Experiments with live trout have demonstrated that if placed in a strong turbulent flow, the trout synchronize with vortices within the flow . Furthermore experiments with euthanized trout have revealed that the trout body was capable of swimming against the current of the flow without any muscular input. The experiment revealed that the trout body automatically synchronises with the vortices which stimulate a forward swimming gait without any input from the fish itself (Beal 2003).

This suggests not only that the dynamics of BCF swimming fish bodies are highly adapted to take advantage of vorticity in the flow. But also that fish employ a degree of passivity in their swimming in order to allow vorticies to affect their gait. The fact this can be achieved by dead fish suggests that sensing the vorticies is not essential to synchronisation. Naturally it is highly desirable for BFC swimming mechanism to be able to dynamically interact with vorticity in the surrounding fluid in order to reduce cost of transport (Anderson 1996).

2.7 Concluding Remarks

In this chapter some of the biological swimming mechanisms utilised by fish and cetaceans in nature were discussed, and their particular interest to the design of biomimetic swimming devices.

The use of passive muscle to generate elasticity within the body and the ability of fish bodies to interact with vortices within the surrounding fluid to reduce the energetic cost of transport is of particular interest for this study. The fact this achievable by dead fish suggests that sensing of the vortices is not essentialist for vortex synchronization.

The ability of dead fish to synchronise with turbulent flows to swim against the current suggests that the body dynamics are a critical factor in efficient swimming.

Chapter 3 will present a review of the available literature relating to biomimetic swimming devices and discuss how other studies have attempted to mimic the biological swimming mechanisms highlighted here.

Chapter 3

Literature Review

There are already a great number of unmanned and autonomous underwater vehicles in use, this chapter aims to provide a comprehensive review of the past and current trends in the development of biomimetic propulsion systems for autonomous underwater vehicles, drawing from the available literature that has been published.

3.1 Introduction

Chapter 2 described some of the swimming mechanisms utilised by fish and mammals in nature to generate thrust and manoeuvre in the marine environment. This chapter aims to provide a comprehensive review of previous attempts to emulate such mechanism in biomimetic devices drawing from the available published literature.

Biomimetic swimming devices will be divided into the two primary swimming modes discuses in chapter 2, *body and caudal fin* (BCF) type swimmers and *paired and median fin* (PMF) type swimmers. However prior to this it is necessary to discuss the construction of more traditional unmanned underwater vehicles in order to establish the relevant design and construction constraints for autonomous underwater vehicles.

This chapter will be divided into four further sections. Section 3.2 will give a brief overview of the design and construction of traditional unmanned underwater vehicles. Section 3.3 will give a history of the more notable biomimetic swimming machines

past and present. Section 3.4 will discuss some of the trends in the development and evolution of the biomimetic swimming machines. Finally section 3.5 will present concluding remarks drawn from this review of the available literature.

3.2 Traditional Unmanned Underwater Vehicles

Prior to discussing biomimetic underwater vehicles it is necessary to discuss the design of more traditional AUVs, in order to establish the associated design and construction constraints. This section will be divided into two further sub sections. The first will discuss the various AUV designs currently in commercial use. The second subsection will give details of the construction principals used to ensure that the devices can survive in harsh sub sea environments.

3.2.1 Design

The first and perhaps the simplest AUVs were the self propelled torpedoes first developed for military use in the 1800s (Blidberg 2001). The first torpedoes were developed with simple long thin bodies to help them maintain on a straight course and dual counter rotating screw propellers at the rear for propulsion, a design principal that to this day many AUVs developers still follow. The addition of a control plane behind the propeller gives course adjustment abilities and by adjusting the size more batteries can be included for longer mission durations and/or larger payloads can be accommodated. One such torpedo shaped AUV is the 7m long *Autosub*, developed by the National Oceanography Center Southampton (Griffiths et al. 2004). Designed for long range survey missions a large proportion of the hull mass is devoted to batteries (up to 700kg). On acoustic survey missions the electrical load of the sensors alone can be as much as 1700W (Collar et al. 1994). Energy is usually the main limiting factor in mission duration. Although torpedo shaped AUVs are known to be reasonably low in hydrodynamic drag and thanks to military research very well understood, their shape

originates from a design selected for its course holding characteristics giving them poor manoeuvrability. Some of these vehicles take several body lengths to perform a 180 degree turn, (Bandyopadhyay 2005). Speed wise most commercially available torpedo type AUVs have design speeds of between 1.5 and 3 ms^{-1} (Budiyono 2009).

For shorter range missions with smaller payloads and greater manoeuvrability requirements, there have been several AUVs developed with multiple cross axis thrusters. By employing thrusters arranged to deliver forces in all three Cartesian axes, such designs offer greater manoeuvrability and station keeping abilities. However such designs also result in the extra weight of multiple motors, the majority of which are redundant during forward locomotion. Also the designs often sacrifice hydrodynamic efficiency for stable thrust delivery platforms.

For situations where manoeuvrability is not an issue underwater gliders can be used. Gliders operate on a varying buoyancy drive. By taking on water the gliders reduce their net buoyancy force and sink. Whilst sinking the relative vertical motion of the water over the wing produces a forward lift force. By expelling the water at a given depth, the glider increases its net buoyancy and floats to the surface (Stommel 1989). By repeating this gliders can travel large distances using very little energy at speeds of around 0.5 ms^{-1} (Budiyono 2009). Steering is achieved either with a control plane or by an active roll mechanism that changes the resultant lift direction (Blidberg 2001). Although gliders offer a very efficient platform for long range survey missions, their low speed, lack of manoeuvrability and reliance on depth variation can be restrictive in terms of mission selection.

Although it has been shown that complex hull shapes can reduce drag, many AUV manufacturers are still using hull designs based on uniform diameter cylinder, which lend themselves to modular construction, for easy extendibility.

Details of some of the many AUVs currently commercially available can be found at

(AUVAC 2013).

3.2.2 Construction

Most AUVs are designed around cylindrical or spherical pressure vessels, (chosen for strength under compression), that house electrical or other pressure sensitive components. The remainder of the hull comprises of either flooded sections or buoyant foam, exposed to the outside pressure. Since some AUVs can reach depths of up to 6000m they must be able to withstand pressures of over 600 bar (Collar et al. 1994).

In order to minimize the amount of additional buoyant material needed and maximize the payload and battery weight allowance, there has been a great deal of research into lightweight materials for the construction of AUVs and their pressure vessels. Materials such as carbon fibre and ceramics have been investigated as well as new buoyant materials that maintain volume under high pressure (Stevenson and Graham 2003).

At great depths one of the largest problems is the protection of the prime mover from sea water. Surface vessels have simple sealing glands where the propeller shaft penetrates the hull to keep the water out. However such a gland would not be effective with a pressure differential of over 600 bar. One solution is to equalize the pressure by mounting the prime mover in a deformable hull section, and flooding it with a non compressible fluid tolerable to the prime mover. Certain mineral oils for instance are non conducting and non corrosive. Therefore such oils will not affect the operation of an electric motor when immersed. However such a solution does reduce running efficiency, as the motor must also overcome viscous forces within the fluid (Sharkh 2003). Some larger AUVs have employed synchronous magnetic couplings that allow the prime mover to be mounted within a pressure vessel with no penetrations, and the propeller shaft to be in a freely flooded compartment. But such couplings are large and

can become unsynchronized if subjected to a jarring force (Sharkh 2003).

3.3 Biomimetic Swimming Machines

Unlike biological evolution which over millions of years has tended to lead to a general increase in sophistication of biological swimmers, the evolution of biomimetic swimmers commenced with intricate complex mechanism and seems to be reducing in complexity with subsequent generations as the fundamental principals are distilled. The remainder of this section will be divided into five further subsections. Subsection 3.3.1 discusses the evolutionary path of robotic BCF type swimmers. Subsection 3.3.2 on the other hand discusses the evolutionary path of robotic PMF swimmers. Subsection 3.3.3 describes attempts made to combine BCF and PMF. Subsection 3.3.4 details some of the recent developments in the field of artificial muscle, and their application to robotic fish. Finally subsection 3.3.6 describes some of the present modelling methods used for robotic fish.

3.3.1 Body and caudal fin swimming machines

The ancestry of almost all biomimetic swimmers can be traced back to the Massachusetts Institute of Technology (MIT) *RoboTuna* an illustration of which can be found in figure 3.1. *RoboTuna* was a $1.2m$ towing tank replica of a real tuna. Built in order to better understand the mechanism involved in forward BCF swimming (Barrett 1994).

The tuna was chosen as a source of bioinspiration because they are one of the fastest swimming fish in nature. Capable of long periods of swimming at high speeds, implying likely use of novel hydrodynamic drag reducing techniques. Other factors that affected the decision to use a tuna for biological inspiration was that different subspecies of tuna have similar morphology despite differences in size. This was thought to imply that any resulting design would be easily scalable for future use as an

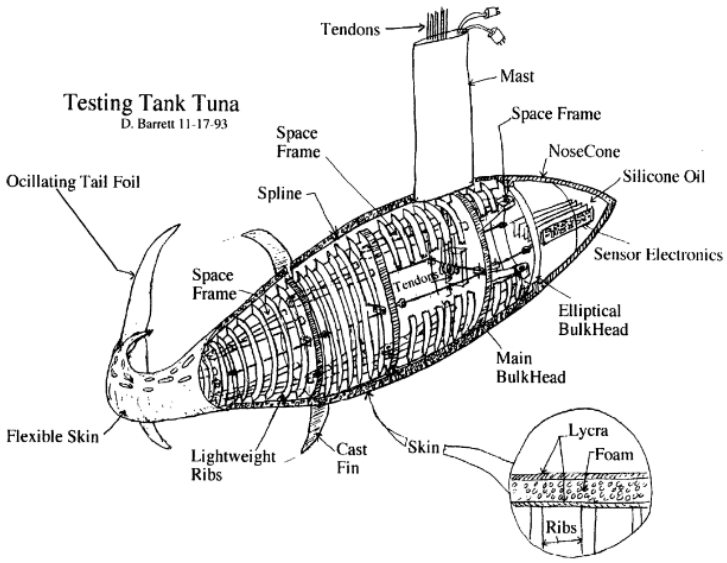


Figure 3.1: MIT Robo Tuna (Barrett 1994) Copyright ©1994 MIT (Permission to reproduce this image has been granted by MIT)

AUV. Furthermore the thunniform swimming mode that allowed a large proportion of the body to remain rigid, would allow for a larger payload.

When in operation *RoboTuna* was attached to an overhead sled by a towing mast in the position of the dorsal fin. The forward speed during runs was determined by the sled.

In order to ensure that *RoboTuna* could approximate actual tuna kinematics as closely as possible, every aspect of the design was over engineered. The shape itself was taken from a casting of a real blue fin tuna, using a custom built 3D profile meter, the shape was copied exactly. The tail movement came from a seven vertebrae backbone. The six joints were each actively actuated by cable tendons that fed through the body and up the mast. The tendons were driven by six large brush less DC servo motors mounted externally on the towing sled. The servo motors were deliberately over sized to avoid actuator saturation during more rigorous kinematics.

Two vertical flexible splines were fixed along the backbone, which transformed the discrete angles of the backbone to a smooth curve. Ribs were mounted onto these

smoothly curving splines at regular intervals to give shape. Over the ribs the tail was fleshed out with thick reticulated foam. Finally to give the whole body a smooth waterproof skin, a conformal Lycra sock was stretched over the entire length of the body. To avoid the complication of fully waterproofing flexible body, the entire hull was allowed to flood via vents cut into the outer skin.

Since the goal of the *RoboTuna* project was to better understand the underlying mechanisms of BCF swimming, a multitude of force sensors were incorporated, to measure the torque on the motors, the drag forces on the mast, and the pressure on the caudal fin. The large amount of sensors, and controllable parameters meant that in all five computers were directly involved in the control monitoring and recording of parameters during each run (Barrett 1996).

The study went on to determine seven key parameters involved in BCF swimming, and by running live experiments with *RoboTuna*, a genetic algorithm was used to produce an optimal set of swimming kinematics. Furthermore results from the *RoboTuna* project did indeed suggest a reduction in drag force for certain kinematics, agreeing with Gray's Paradox which states that some swimming creatures seem to have insufficient muscle mass to overcome linear drag forces at speeds at which they are known to swim (Gray 1936).

Following the success of the *RoboTuna* project, MIT in partnership with Draper Laboratories developed the *Vorticity Controlled Unmanned Underwater Vehicle* (VCUUV) (Anderson and Kerrebrock 2000) using many of the techniques developed during the *RoboTuna* project. The VCUUV was a self contained free swimming robotic tuna, built as a proof of concept prototype of a biomimetic AUV. Once again the morphology of a real blue fin tuna was used, however this time the shape was scaled up to $2.4m$ in length, comparable in size to some of the smaller conventional AUVs in use at the time.

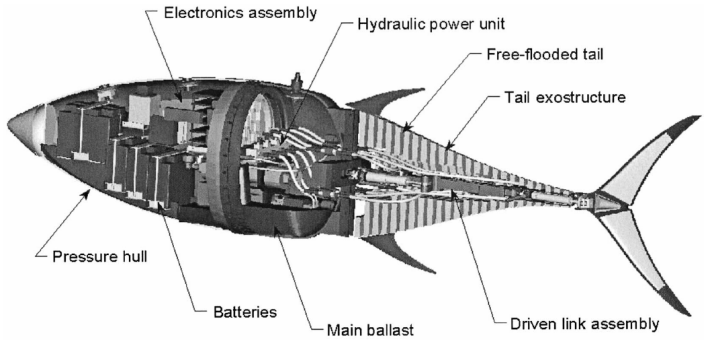


Figure 3.2: Draper Laboratories VCUUV (Anderson and Kerrebrock 2000) Copyright ©2002 Oxford University Press (Permission to reproduce this image has been obtained through RightsLink®)

VCUUV's tail movement came from a simplified five vertebrae backbone, with the four joints actively controlled by a closed loop hydraulic system (Cho 1997). The backbone in turn acted on a spline and rib structure similar to the one used in *RoboTuna*. The skin used was Lycra bonded to neoprene rubber. Like *RoboTuna* rather than trying to seal a flexible structure, the tail was allowed to be flooded.

The forward section of the body was constructed as a single pressure vessel, and contained the hydraulic system and the electronics. The hydraulic actuation system selected for its high power density, consisted of a reservoir, a small positive displacement pump, a pressure accumulation vessel, four servo valves and four cylinders. The cylinders being the only components outside the pressure vessel. The only penetrations needed through the pressure vessel walls were for hydraulic hoses. A diagram of the VCUUV layout can be found in figure 3.2.

By directly taking the forward swimming parameters derived during the *RoboTuna* project it was assumed that VCUUV would have a near optimal swimming kinematic without any further optimisation. The freedom from a towing tank sled meant that VCUUV could also be used as a testbed for turning manoeuvre kinematics.

During the testing of the VCUUV it was found that it was capable of turning rates of

up to $75°s^{-1}$, vastly outperforming conventional AUVs which usually have turning rates of approximatly $4°s^{-1}$. Unfortunately VCUUV was unable to reach its design speed of $1Ls^{-1}$ due to saturation of the actuator system at tail beat frequencies above $1Hz$, however a top speed of $0.61Ls^{-1}$ was achieved (Anderson and Chhabra 2001). The most recent direct application of the *RoboTuna* design can be seen in Boston Engineering's *GhostSwimmer* (Engineering 2009), which is a tuna based AUV currently being developed under commission from the US government for use in harbour monitoring.

The next generation of robotic swimmer to emerge from MIT was *RoboPike*. At approximately $80cm$ in length *RoboPike* was originally built as an undergraduate design project and later became an experimental test bed for experimentation on carangiform rapid manoeuvring kinematics (Kumph 2000). The pike was chosen as a source of bioinspiration because of the rapid manoeuvring, and acceleration abilities demonstrated by it in nature.

RoboPike's tail movement came from a further simplified four vertebrae backbone, with the three joints actively controlled by tendons driven by waterproofed brush less DC servo motors mounted in the midsection of the body. The backbone was connected to a novel helical wound fibreglass rib structure, stiffened in the vertical axis by a vertically mounted flexible spline. Over the rib structure a neoprene Lycra skin was stretched to form the outer hull. It was thought that the helical wound rib structure would give the tail elastic energy storing properties, similar to those reportedly used in real fish to increase metabolic efficiency while swimming (Pabst 1996). *Robopikes* forward section was constructed as a single pressure vessel housing batteries and electronic sub systems, however the tail and the mid body were flooded. Without parameter optimization *RoboPike* had a maximum speed of around $0.3Ls^{-1}$ at a tail beat frequency of $1Hz$.

The Japanese National Maritime Research Institute, (NMRI), developed a series of further simplified link based robotic fish, including a three link 34cm robotic sea bream denoted PF-300 to study turning performance (Hirata et al. 2000). The sea bream was selected as a source of bioinspiration because in nature its large side profile area and carangiform swimming style makes it an excellent fast turning fish. The two joints were actuated directly by brushless DC servo motors housed in small pressure vessels, the actuation mechanism penetrated the pressure vessel through a corrugated waterproof boot (Hirata 2000). The tail itself was left in a naked skeletal state, as it was thought that the majority of the propulsive force would be generated by the caudal fin, thus accurate representation of the rest of the body morphology was thought to be unnecessary. Servo control came from a standard radio control unit, A float held the aerial at the surface, and ensured that the PF-300 maintained a constant depth.

The PF-300 was able to produce tail beat frequencies of up to 2.3Hz, turning diameters as small as 75mm could be achieved and a top speed of approximately $0.6Ls^{-1}$. Subsequent robot swimmers developed by NMRI, include the 65cm four link PF-600, the 70cm four link PF-700, the 97cm three link UPF-2001, the 26cm two link PF-200, and the 57cm three link PF-550.

The PF-600, designed to study propulsion performance, had a cylindrical body housing two brushless DC servo motors, one servo motor actively controlled the two foremost joints, whilst the second servo motor was devoted entirely to actuating the caudal fin, allowing experimentation with phase angle between fin and tail movements.

The PF-700, was built for experiments on fast swimming, the body had a long slim cylindrical form designed for low drag. Through the use of a combination of brushless DC servo motors and a larger DC motor driving a Scotch yolk mechanism tail beat frequencies of up to 10 Hz could be achieved, resulting in a top speed of $1Ls^{-1}$.

The UPF-2001, was a simple three link robot designed as a multi purpose research

platform. The tail was driven by a single DC motor and Scotch yoke mechanism, driving both the tail joint and the fin joint with a phase difference generated through a novel mechanical mechanism, like the PF-700 a tail beat frequency of 10Hz was required to generate a speed of $1Ls^{-1}$. The PF-200 was a small proof of concept prototype that used a shifting mass mechanism to give active pitch control for manoeuvring in the vertical axis. The most recent robotic swimmer from NMRI is the PF-550, using a simple three link design, actuation came from two brush less DC servomotors, the entire tail mechanism is mounted on a rotating shaft, allowing the primary propulsor to be rotated to give agility in the vertical axis.

Like the PF-300 all the subsequent robot swimmers to emerge from NMRI relied on radio communication remote control, limiting them to operations on or near the surface. With the exception of the UPF-2001 all were constructed with open skeletal joints, however effort was made to approximate the profile of real fish tails by attaching moulded sections to the tail vertebra.

Figure 3.3 shows some of the robotic fish developed at NMRI. From the top the figure shows the PF-300, PF-600, PF 700, UPF-2001 and the PF-550.

This figure has been removed due to Copyright restrictions

Figure 3.3: Robitic fish developed by Japanese National Maritime Research Institute, (NMRI) (Hirata 2000) Copyright ©NMRI

The Tokyo Institute of Technology (TIT), developed two robotic dolphins aimed as prototypes toward the design of a biomimetically propelled AUV (Nakashima and Ono

2002). The first had a pneumatic actuation system, the second a DC servo motor.

Both robots had a three vertebrae design 1.5m in length with one active joint at the top of the tail and a passive joint at the caudal fin. By varying the stiffness of the passive joint it was found that a wide variety of tail beat kinematics could be achieved (Nakashima et al. 2004).

The bodies were constructed from an aluminium frame wrapped in carbon fibre reinforced plastic. The deformable tail shapes were made from fibre reinforced plastic rings connected by a non elastic waterproof membrane. The stiffness of the passive joint was adjusted by interchanging springs of variable elastic modulus. Through experimentation it was found that speeds of around $0.6Ls^{-1}$ could be achieved with tail beat frequencies of around 1.8 Hz. The popularity of 'Gray's Paradox' (Gray 1936) makes dolphins an obvious source of bioinspiration. So much work has been done investigating the paradox that dolphin swimming kinematics are among the best understood in nature (Fish 2006). Furthermore their size in nature is comparable to the size of existing AUVs.

Developers at the Istanbul Technical University (ITU), also developed a robotic dolphin AUV prototype. The aim was to improve upon the propulsion efficiency found in conventional AUV (Dogangil et al. 2005*a*). The Istanbul dolphin had a four vertebrae construction with each of the three joints actuated by an opposing bellows type pneumatic system. The Istanbul dolphin was constructed from sheet aluminium and polymerized formaldehyde plastic.

The flexing tail section was covered in a waterproof membrane supported by a flexible structure to allow the tail joints to remain dry. The caudal fin was made from cast silicon in order to mimic the flexibility of a real dolphin's caudal fin. Turning manoeuvrability was achieved using pectoral fins actuated in pitch. The Istanbul dolphin was able to swim at a speed of $1Ls^{-1}$, with a tail beat frequency of $1.35Hz$.

Figure 3.4: Photograph of Essex university G9 (Liu 2005) Copyright ©2005 Liu (Permission to reproduce this image has been granted by Liu

The University of Essex developed a series of multi link carangiform and sub-carangiform robot swimmers (Liu 2007). The latest of which the G9, based on a four vertebrae tail structure constructed using stereo lithography apparatus resin. The three joints were actively controlled by three powerful DC servo motors, capable of bending the body through an angle of 90° in 0.2s (Hu 2006). The G9 achieved manoeuvres in the vertical axis through a shifting mass mechanism, which moves the centre of gravity to alter the pitching moment, much like in the PF-200. No specific fish was chosen as a source of bioinspiration, instead an attempt was made to capture the more generalised principals of fish morphology.

The debut of the Essex fish at the London Aquarium in 2005 has made them perhaps the most well known robotic fish. They are currently being implemented in a collaborative project entitled, *Search and monitoring of harmful contaminants, other pollutants and leaks in vessels in port using a swarm of robotic fish* (SHOAL). Figure 3.4 shows a photograph of the G9 robotic fish that featured in the London Aquarium.

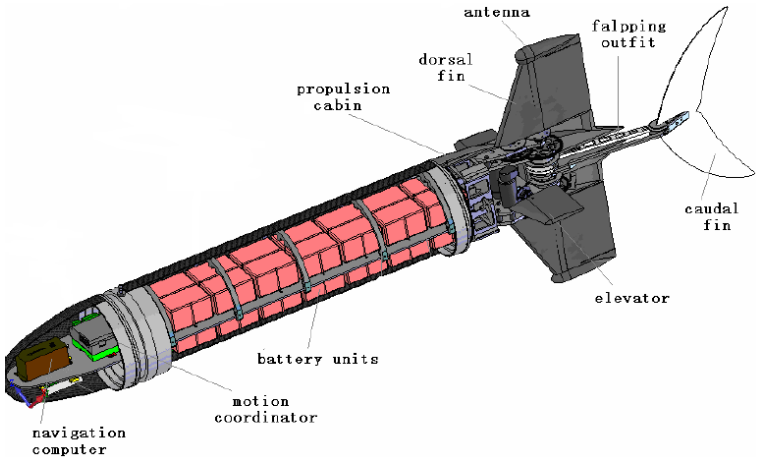


Figure 3.5: Beihang University SPC-III (Wang et al. 2010) Copyright ©2010 Elsevier (Permission to reproduce this image has been obtained through RightsLink®)

Beihang University Robotics Institute also developed a series of robotic fish based on non specific bioinspired morphology for use as UUVs. Among these the SPC-II and SPC-III had a common two joint BCF type propulsion module. The two joints were each actuated by a 150w brushless DC motor located within a sealed part of the vehicle. In water the DC servo motors were capable of driving the tail with beat frequencies of up to 2.5Hz.

SPC-II has a roughly fish like morphology designed like the PF-300 with a large side profile area for rapid turning ability, with an overall length of 1.2m. The forward part of the body was constructed as a rigid pressure vessel, and the tail mechanism was attached behind. The tail mechanism was capable of driving SPC-II at speeds of up to $1.2Ls^{-1}$, and producing yaw rates of up to $70°s^{-1}$. Despite having a maximum depth rating of only 5m SPC-II proved useful as a visual assistant in underwater archaeology (Liang et al. 2005).

SPC-III was constructed in many ways like a traditional AUV however in place of the propeller the two joint BCF tail was attached as shown in figure 3.5. With a 1.6m

rigid body section , the hydrodynamic shape enabled the propulsion system to drive the vessel at speeds of up to $1.17Ls^{-1}$ (Wang et al. 2010).

Following on from a biological study that demonstrated that dead fish exposed to harmonic stimulus could produce a forward swimming gait (Beal 2003). Researchers at MIT have developed a simplified compliant body method for generating BCF swimming gaits for small biomimetic AUVs suitable for multi agent survey tasks as shown in figure 3.6 .

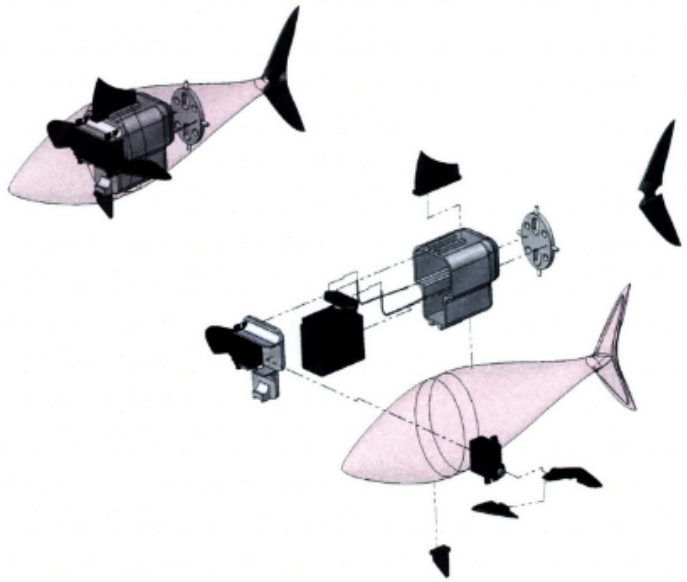


Figure 3.6: MIT Compliant Swimming Device (Alvardo 2007) Copyright ©2007 MIT (Permission to reproduce this image has been granted by MIT)

A simple DC servo motor driven mechanism embedded into a moulded silicon body, can produce a travelling body wave if activated periodically (Alvardo 2007). It was found that by doping the silicon the body could be given a different elastic modulus and hence produce different kinematics.

Various compliant body prototypes have been made, the largest being around $32cm$ in length, and reached a speed of $1Ls^{-1}$ at a tail beat frequency of $3.5Hz$. Reports indicate

that these simplified designs have proven to be fairly robust and give good longevity.

The University of Glasgow also developed a single actuator swimming mechanism for comparison experiments between biomimetic and conventional propulsion (Watts 2009). An $85cm$ robotic salmon named *RoboSalmon*, was built with a 10 vertebrae backbone actuated by a single tendon running down either side. A single DC servo motor drove the tendons to produce tail oscillations. However the resulting tail movement more closely resembled ocilliform swimming than carangiform.

The shape of the tail itself came from plastic ribs attached to intermittent vertebra forming a rib cage. This rib cage was covered in a waterproof membrane giving a dry tail. The second generation of the *RoboSalmon* also had a novel turning head section, that it was thought could compensate for excessive yaw oscillations observed in the original *RoboSalmon* during forward swimming and increase yaw rates during turning.

The poor tail beat kinematic resulted in a maximum speed of only $0.2Ls^{-1}$ at frequencies around $1Hz$. However experiments with the *RoboSalmon* did demonstrate that even such a sub-optimal kinematic *RoboSalmon* did produce more efficient propulsion than a propeller drive system at similar speeds, and was capable of far superior manoeuvrability.

A summary of BCF swimming Robots can be found in Table 3.1.

3.3.2 Paired and median fin propulsion

The excellent manoeuvrability and station keeping ability of PMF swimmers in nature inspired researchers at Tokai University to develop the robotic black bass (Kato 2000). The black bass was chosen because in nature it is a species known to use pectoral fins for low speed locomotion, and station keeping manoeuvres. By using two servomotors for each pectoral fin, the fins could be actuated in yaw and pitch axis respectively. By controlling the relative phase and magnitude of yaw and pitch

Table 3.1: A Summary of BCF Swimming Robots

Institution	Name	# Joints	Actuators	LOA	Year
MIT[1]	RoboTuna	6	6 DC servo	1.2m	1994
Draper	VCUUV	4	4 Hydraulic Piston	2.4m	1997
MIT	Robopike	3	3 DC Servo	80cm	2000
NMRI[2]	PF-300	2	2 DC Servo	34cm	1999
NMRI	PF-600	3	2 DC Servo	65cm	2000
NMRI	PF-700	2	1 DC motor +1 DC Servo	70cm	2001
NMRI	UPF-2001	2	1 DC motor	26cm	2001
NMRI	PF-550	2	2 DC Servo	57cm	2003
TIT[3]	Robot Dolphin 1	2	1 Pneumatic motor	1.8m	2002
TIT	Robot Dolphin 2	2	1 DC Servo	1.8m	2004
ITU[4]	Robot Dolphin	3	3 Pneumatic Paired Bellows	2m	2005
Essex	G9	3	3 Dc Servo	52cm	2007
BRI[5]	SPC-II UUV	2	2 DC Servo	1.2m	2005
BRI	SPC-III UUV	2	2 DC Servo	1.8m	2010
MIT	Compliant	N/A	1 DC Servo	$\leq$32cm	2007
Glasgow	RoboSalmon	10	1 DC Servo	85cm	2009

[1] *Massachusets Institute of Technology*
[2] *National Maritime Research Institute of Japan*
[3] *Tokyo Institute of Technology*
[4] *Istanbul Technical Universty*
[5] *Biehang University Robotics Institute*

oscillations, manoeuvring forces in the full six degrees of freedom could be achieved. Through experimentation it was demonstrated that precision docking manoeuvres could be performed in currents of up to $0.05ms^{-1}$. The black bass project could be looked at as the equivalent to PMF locomotion, as what *RoboTuna* was to BCF locomotion. Figure 3.7 shows a photograph of the third generation of Black Bass from Tokai University.

Figure 3.7: A photograph of Black Bass III (Kato 2000) Copyright ©2000 IEEE (Permission to reproduce this image has been obtained through RightsLink®)

Despite the success of Kato's robotic black bass using a single pair of propulsive fins several subsequent BCF propelled AUV designs have used multiple pairs of fins. *AQUA* was a six finned robot swimmer developed at McGill University (Georgiades et al. 2004). The swimming gait was roughly based on ostraciform swimming, with the six fins only actuated in the pitch axis. The fins were flipper shaped rather than wings shaped to produce a drag based thrust. The body itself was based on an earlier RHex terrestrial robot design, with expectations that it could be developed into a fully amphibious AUV platform.

The *AQUA* project demonstrated that using multiple simple single axis actuated flippers, heave, surge, pitch, roll and yaw motions can be achieved. A photograph of AQUA swimming can be found in figure 3.8.

MIT researchers have also developed a multi paired fin swimming AUV (Licht 2008). Based roughly on the morphology of a sea turtle, MIT's *RoboTurtle* was a four finned labriform type swimmer. The four finned design was selected as it was thought that the symmetry would provide a more stable and easier to control platform.

For simplicity of expansion *RoboTurtle's* fins were constructed as self contained modules (Licht et al. 2004). Each module contained a 190W DC brushed motor to

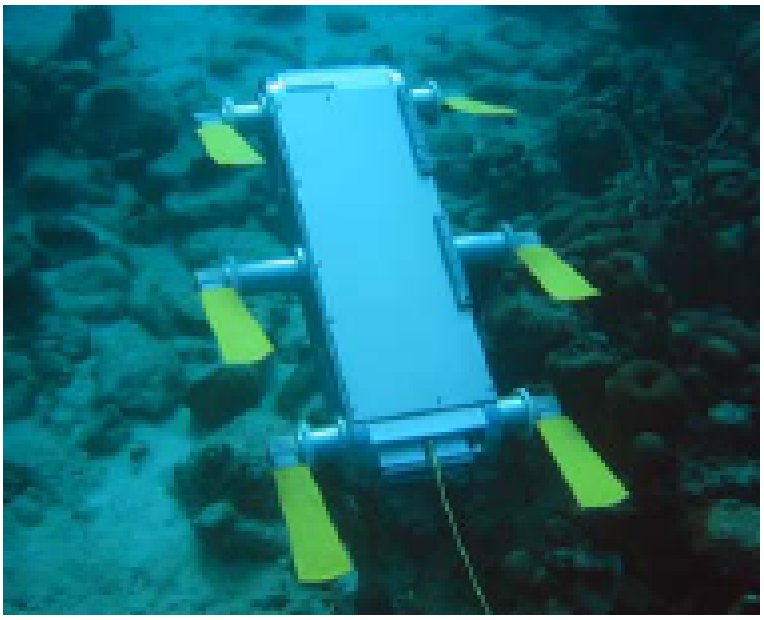


Figure 3.8: A photograph of AQUA swimming (Georgiades et al. 2004) Copyright ©2000 IEEE (Permission to reproduce this image has been obtained through RightsLink®)

provide actuation in roll and a 15W DC brushed motor to provide motion in pitch. All the corresponding motor control circuits and all the connections required to add one more fin module, giving devices a 'plus one' type infinite expandability. Like the robotic black bass manoeuvring forces were controlled by altering the phase and amplitude of oscillations. Figure 3.9 shows a CAD drawing of Robo Turtle with the modular fin unit.

A similar four finned modular design has been adopted by the commercially available *Transphibian* AUV from the iRobot Corporation (iRobot Corporation 2010), by using the fins as legs, the Transphibian is also able to produce limited terrestrial locomotion.

More recently a collaboration between The Pertoleum Institure in United Arab Emirates and University of South Queensland has focused on employing a similar fin mechanisms to on a 6 finned robot turtle with a goal of amphibious locomotion (Cubero 2012).

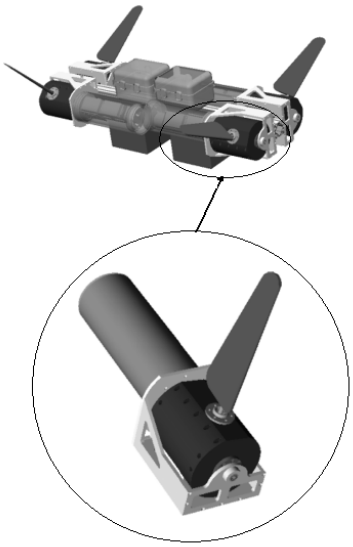


Figure 3.9: A CAD drawing of Robo Turtle adapted from (Licht et al. 2004) (Georgiades et al. 2004) Copyright ©2000 IEEE (Permission to reproduce this image has been obtained through RightsLink®)

Inspired by an observation that in nature, many amphibious animals despite having four limbs tend to only use only two for aquatic propulsion, developers at Vassar Collage developed a four finned swimming robot for experimentation regarding the specific advantage of four and two fin swimming gaits (Long et al. 2006) like the *AQUA* robot, the fins on the Vassar robot were operated as flippers with actuation only in pitch.

Experiments carried out using the Vassar robot comparing two and four fin swimming gaits indicted that although four fin gait did produce improved acceleration and breaking rates compared to two fin gaits, peak velocities achieved were the same for both two and four fin gaits. Furthermore the overall energy cost of transport for four fin gaits was more than double that of the two fin gaits.

Tufts University initiated the development of a large manta ray based swimmer/glider (Brower 2006). The manta ray was selected as a source of bioinspiration, because in nature manta rays are relatively fast, efficient swimmers and their large wing area make them a good candidate for glider design. The Tufts project looked at using pneumatic

pistons to actuate a multi-joint skeletal arm. The pistons were to be inserted into a moulded PVC ray wing, with the hope of producing a ray like swimming motion. The final design was to be approximately 50cm long with a total wing span of 1m. However after construction of one wing it was found that the combined effects of the inertia of the large wing span, the inertia of the water to be moved and the rigidity of the PVC meant that despite the use of a relatively powerful pneumatic system the actuators were unable to move the wing whilst it was in the water.

However two such manta ray based swimmer/glider AUVs have subsequently been developed in the commercial sector (Festo 2007; Logics 2013). Festo and Evo Logics have developed manta ray based AUVs, both using a novel fin ray effect designed by Evo Logics to maintain constant volume within the flexible structure. The Festo Ray shown in figure 3.10 uses a powerful hydraulic actuation system to control its wings, which have a 96cm span. The Evo Logics Ray comes in a variety of sizes from 1.5m up to 3.5m wing span, and incorporate a buoyancy driven glider mechanism, and a hydro jet propulsion system for precision manoeuvring as well as the ray like swimming motion.

Most recently researchers at the Robotics Institute of Beihang University have developed a robotic cow nosed ray (Cai et al. 2010). *Robo-ray II* was built over a single flexible rib, actuated by two Mckibben type pneumatic muscles, and a vertically flexing rudder section, also pneumatically actuated. This skeleton was then fitted with laterally oriented spines. Finally the whole structure was cast into silicon using a ray shaped mould. The overall design had a total wing span of 56cm and a total length of 32cm, and had a top speed of around $0.5Ls^{-1}$ with a wing beat frequency of 1.2Hz.

Festo have gone on to use the aforementioned fin ray effect in their penguin inspired AUVs shown in figure 3.10 (Festo 2009). Festo's *AquaPenguin* uses two pectoral fins in a labriform mode for propulsion. Both fins are driven in the roll plane by a single shared

DC motor, with mechanical gearing to give a synchronized roll oscillation. On each fin pitch control is achieved using a dedicated DC servo motor. The fins themselves are constructed on a sprung steel wire mesh embedded in moulded silicon polyamide, closely following the morphology of real penguin wings. The *AquaPenguins* have an overall length of about $77cm$ and a top speed of around $2Ls^{-1}$.

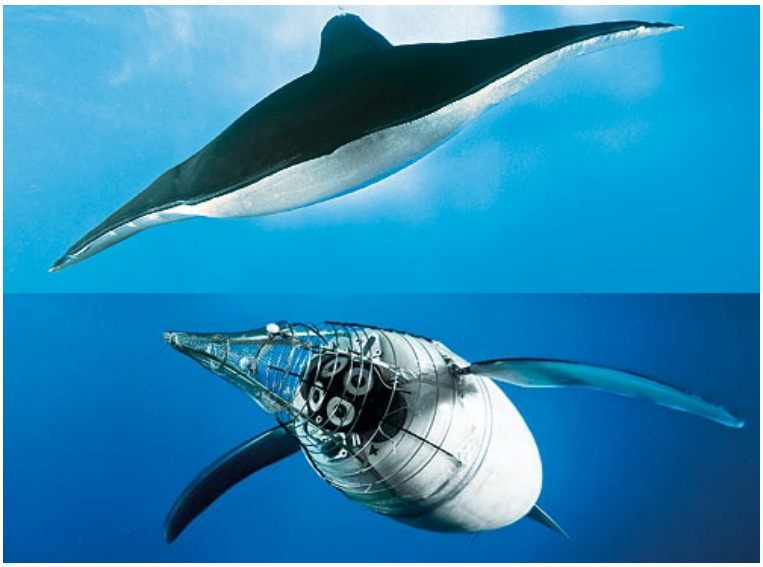


Figure 3.10: Top: Festo Aqua Ray, Bottom: Festo: Aqua Penguin; Copyright ©Festo AG & Co. KG, photographer Walter Fogel. (Permission to reproduce this image has been granted by Festo)

Whilst the majority of investigations into PMF swimming have focused on oscillatory type motions, there has been some interest in undulatory type PMF locomotion.

North Western University has developed a ribbon fin device based on the gymnotiform locomotion used by black ghost knifefish (Epstein et al. 2006). The knifefish was selected as a source of bioinspiration because of its ability to manoeuvre effectively in all six degrees of freedom, including reversing manoeuvres, despite having a relatively stiff body. The ribbon fin device comprised of a flexible membrane suspended between spines arranged in a line down the underside of the craft. The spines themselves are each oscillated in the roll plane. By controlling the phase and amplitude of the

Table 3.2: A Summary of PMF Swimming Robots

Institution	Name	# fins	Movement	Year
Tokai University	Robot Blackbass	2	Yaw and Pitch	2000
McGill University	AQUA	6	Pitch	2004
MIT	RoboTurtle	4	Roll and Pitch	2004
iRobot	Transphibian	4	Roll and Pitch	2010
Vassar colage	Madeleine	4	Pitch	2006
Festo	AquaRay	2	Roll	2007
Evo Logics	Subsea Glider	2	Roll	2007
Beihang University	Robo-ray II	2	Roll	2010
Festo	AquaPenguin	2	Roll and Pitch	2009
North Western University	Ribbon Fin	1	Undulatory	2006
Nayang Tech University	Knife Fish	1	Undulatory	2006
Nayang Tech University	Sting Ray	2	Undulatory	2006
Delft University	Galatea	2	Undulatory	2009

oscillations it was found that a propulsive wave could be propagated forwards or backwards and the introduction of various offsets could generate a great variety of manoeuvres.

Similar ribbon fin devices were used for propulsion systems in small scale experimental robots developed at the Nayang Technological University (Low and Willy 2006). A robot knifefish using gymnotiform propulsion and a robotic stingray using rajiform propulsion, where two such ribbon fin devices were mounted in the pectoral positions for propulsion.

The most practicaly scaled example of such a propulsion mechanism can be found in the Delft University's Galatea , a box like AUV using rajiform locomotion (Simons et al. 2009).

A summary of the more notable PMF swimming robots can be found in Table 3.2.

3.3.3 Combined BCF and PMF swimmers

Having developed both fast/efficient BCF swimming and precision stable PMF swimming mechanisms, the natural logical progression is to combine both to create a versatile AUV platform. Although several BCF swimming robots have incorporated actuated pectoral fins for vertical lift generation (Dogangil et al. 2005*a*; Hirata 2000). The availability of self contained modular pectoral fin arrangements as outlined in Licht et al. (2004), should make the incorporation of PMF swimming mechanism into existing BCF or even propeller driven devices relatively straight forward. Despite this there have been relatively few attempts to incorporate more sophisticated pectoral fin manoeuvring systems into BCF swimmers.

Peking University has been developing a 1.2m robotic dolphin, which combines BCF and PMF swimming. The BCF motion for the Peking University's dolphin comes from a novel adjustable amplitude Scotch yoke mechanism driven by a 150W DC brush motor driving the tail section and a 20W DC servo motor actively controlling the caudal fin (Yu et al. 2007). The tail itself is constructed with a three vertebrae design, covered in a steel rib structure. The ribs are connected with sprung steel joints that allow flexibility, and in turn are covered in a rubber waterproof membrane, to give a dry tail section (Hu et al. 2008).

The simple BCF mechanism is capable of tail beat frequencies in excess of $3Hz$, and manoeuvring can be achieved by reducing the DC motor in the tail to performing partial rotations.

The PMF motion comes from pectoral fin modules similar to those outlined in Licht et al. (2004) and enable the robotic dolphin to perform station keeping and reversing manoeuvres.

Other developers working on combined propulsion systems include the National

Taiwan University (Guo 2006), and the Chinese Academy of Science Beijing (Zhou et al. 2008).

The Taiwan University's *Biomimetic Autonomous Underwater Vehicle* (BAUV), is a $2.4m$ biologically inspired swimming robot that incorporates a three vertebrae BCF swimming mechanism, with two pectoral fins actuated in roll and pitch, to provide both high speed BCF swimming with PMF precision manoeuvring (Guo 2006).

Developers at the Chinese Academy of Science Beijing have developed a $78cm$ biomimetic swimmer that uses a novel mechanical linkage system similar to the one used in the UPF 2001. The mechanical linkage system is able to derive actuated control over two links from a single DC motor to provide BCF propulsion. The resultant prototype was capable of tail beat frequencies of up to $4Hz$ and a top speed of around $0.5Ls^{-1}$. PMC manoeuvring comes from two pectoral fins driven by a three motor arrangement similar to the one found in Festo's *AquaPenguin*, giving active roll and pitch control for labriform locomotion (Zhou et al. 2008).

3.3.4 Alternative actuators

Conventional actuators used in robotics, such as DC motors or servo motors and pneumatic or hydraulic pistons can be fairly restrictive when trying to synthesize the movement of biological mechanisms. Ideally developers of biomimetic devices would like biomimetic actuators, which emulate the behaviour of natural muscle.

Natural Muscle

Natural muscle is an elastic linear actuator, typical mammalian muscle can produce stress of around $0.35MPa$ and strains of around 20% (Mirfakhrai et al. 2007). Biomimetic actuators can be roughly divided into piezoelectric materials, shape memory material electro active polymers, chemo-mechanical actuators.

Piezoelectric materials

Some crystalline structures mostly found in ceramics, react to deformational stress by producing an electrical charge, and conversely react to an electrical charge by producing a deformation (Haertling 1999). Typical piezoelectric actuators can produce deformation forces in the order of $40MPa$, however with typical strain generation of less than 0.5% displacement amplification is needed for use in robotics, which in turn reduces force.

Shape memory materials

It was discovered that certain deformable polymers and alloy will return to a pre-trained shape when exposed to stimulus such as heat. Typical shape memory materials such as Ni Ti shape memory alloy (SMA) can produce contraction stresses in the order of $200MPa$, and strains of around 8% Shinjo (2005). When used in wire form SMA is widely accepted to be a good approximation of biological muscle, in terms of flexibility.

Electro active polymers

Several polymer constructions react to an electrical charge with large deformations; Dielectric elastometer actuators (DEAs) produce stress in the order of $1MPa$ and strains of up to 100%, however their reliance on high voltage can make them hazardous in larger applications that require higher currents; Ionic polymer-metal composites (IPMCs) can give strains of up to 2% and stress of $30MPa$, actuators are usually formed into bi-metallic strips, where linear contraction is translated into curvature; Carbon nano tubes (CNTs) have developed a lot of excitement in the field of actuators, with maximum strain levels typically just less than 2%. Incredible stresses can be generated with relatively low voltages, around 640 MPa at around 7V, and the incredible work density around $1MJm^{-3}$ make them a very attractive actuator, however with a current production cost of around $500,000\$kg^{-1}$ large scale implementations are cost

prohibitive (Mirfakhrai et al. 2007).

Chemo-mechanical actuators

Certain resins react to a change in PH with a change in volume. By using a Mckibben type actuator, this change in volume can be translated to linear contraction or expansion Tondu et al. (2010). Furthermore there have been developments towards solid state chemo mechanical actuators Choe and Kim (2006), however although PH driven actuators could potentially allow for greater energy storage densities, current technologies are limited by long actuation times, (in the order of minutes).

Alternative actuators in swimming machines

There have been a few attempts to implement solid state artificial muscle technologies in robotic swimmers. For example implementation of IPMC actuators on small scale swimmer was discussed in (Guo et al. 1998; Zhang et al. 2006) and (Hu et al. 2009). However the restriction of IPMC actuator sizes available make them a poor choice for large scale projects. Use of SMA to actuate a small tadpole robot was discussed in in Kim et al. (2005). The first investigation into the use of biomimetic actuators on a larger scale swimmer was carried out at the Florida Institute of Technology (Shinjo 2005). SMA was used to directly emulate the muscle structure of a $50cm$ South Atlantic Bonito. It was found that with sufficient independent lengths of SMA natural body movement could be accurately duplicated at tail beat frequencies of up to $1Hz$. The SMA selected was the NiTinol (Ni Ti) wire commercially available in a variety of diameters (GMBH 2013). A subsequent attempt to produce a simplified SMA actuated tuna was made by the University of Victoria Canada, using the configuration (Suleman and Crawford 2008), it was found that by flooding the tail, the actuators had better cooling and thus could maintain higher tail beat frequencies, up to $2Hz$, however the SMA design was eventually abandoned for a more conventional servo driven tail, as it

was found that the reliance of SMA on temperature fluctuation for activation made it thermodynamically inefficient, and the low cycle life of around 10^6 actuations would result in an unacceptably short operational lifespan (Suleman and Crawford 2009). Recently Researchers at the Swiss Federal Laboratories for materials research, have developed a fish like airship, that uses DEA sheets to produce a large scale carangiform motion (Jordi et al. 2010). Figure 3.11 shows a CAD drawing of the Swiss Federal Laboratories for materials research fish like airship. The construction is similar to the multi body kinematic chain structure used by many of the BCF type swimmers featured earlier in this chapter, however liner actuators are used rather than axial torque generating actuators. To date however there has been no attempt to realize this in an aquatic environment.

This figure has been removed due to Copyright restrictions

Figure 3.11: Swiss Federal Laboratories for materials research DEA fish like airship (Jordi et al. 2010) Copyright ©2010 IOPscience

3.3.5 Control Signal Generation

All of the biomimetic swimming devices featured within this chapter have relied on periodic oscillation or undulation to to generate forward thrust. The generation of such periodic motion is an area that has received a great deal of work in itself. Whilst the majority of devices reported relied on strict set of predefined kinematic patterns (Barrett 1996), (Watts 2009), (Liu 2007). The navigation control is then restricted to switching between a finite number pre defined gaits. Such an approach can obviously

be restrictive in terms of manoeuvrability. In the ideal world the motion should be adaptive to surroundings and higher level navigation commands. In an effort to achieve this researchers are attempting to generate gaits in real-time.

One common approach to real time gait generation is the use of *central pattern generators* (CPG)s. A CPG can be described as are neural network capable of providing a co-ordinated set of rhythmic output signals from a non rhythmic input signal (Crespi et al. 2008). From the biomimetic perspective it is thought that a CPG emulates the low level neuromechanical control of real fish in nature.

The Swiss Federal Institute of Technology have developed a simple three finned robotic fish with a single DV servo motor dedicated to each fin. A geometric position signal for the three fins was prescribed using the CPG. The Servos then attempted to match this position by using proportional differential control. Experiments demonstrated that such and approach can generate an effective multi-fin swimming gait in real time (Crespi et al. 2008).

Similarly researchers at China's National University of Defence Technology developed a gymnotiform robotic fish. The gymnotiform motion was generated by nine DV servo motors each controlled by a dedicated CPG. The CPGs for each servo were interconnected to ensure co-ordination of the motion. Experiments demonstrated that the approach could generate effective manoeuvring control for a undulatory type robotic fish with a high degree of actuation (Zhang et al. 2008).

Researchers at the National Natural Science Foundation of China have demonstrated that CPG gait generation can be effective for a fully actuated four joint BCF swimmer. DV servo motors positioned in each joint were responsible for actuation (Yu et al. 2011).

Whilst it is argued in (Crespi et al. 2008), (Zhang et al. 2008) and (Yu et al. 2011)

that CPGs represent an more accurate representation of the way fish generate there muscle stimulus, all three studies mentioned used the the CPG signal to prescribe geometric position. It has been suggested that prescribed geometric control is a poor representation of biological muscle as it precludes force driven or spasticity driven motions (Stefanini et al. 2006). A possible solution would be to use a CPG for open loop actuator excitation control however the absence of feedback could generate stability issues.

3.3.6 Modelling and Simulation

Parallel to the development of physical prototypes of robotic swimming devices, increases in the availability and sophistication of computers has led to an increase in the development of analytical and computational models, for the exploration and optimization of fish like swimming. Several studies have applied a purely analytical approach to the fluid modelling, such as the previously mentioned Harper et al. (1998), which used a quasi steady third order approximation of the Theodorsen method on a hydrofoil to approximate the flow over a caudal fin. However this model ignored all upstream hydrodynamic effects that would be created by the fish body. In Mason (2003), a quasi steady Joukowski transformation method was used to create a two dimensional model of a fish swimming. However the nature of Joukowski deformations does not allow for a standing body wave, characteristic of carangiform and sub carangiform swimming, so the model could at best been said to be a two dimensional approximation of ocilliform locomotion. The limitations of analytical modelling techniques stem from the dependence on geometries describable by mathematical functions. In order to model situations with more complex geometries computational numerical methods for solving the incompressible Navier Stokes equations of fluid motion are often employed. However as pointed out in Mittal (2004), to capture accurately all the features of the higher Reynolds number flows typical

for AUV size models such methods require high resolution numerical discretization, resulting in very large processing requirements. Furthermore for very high Reynolds number flows typical for larger AUVs or manned marine vessels areas of low pressure can be generated within the flow below the fluids vaporisation point, causing cavitation. Cavitation cannot be modelled with standard incompressible direct Navier Stokes solvers because they operate on the assumption that conservation of mass is sustained through conservation of volume, however cavitation can result in a local volume increase in the order of (104 :1). Since it has been shown that boundary layer cavitation can dramatically affect both lift and drag on a hydrofoil (Schnerr 2003), the modelling of cavitation has been a major research subject in naval architecture, several methods have been developed for modelling such cavitation (Schnerr 2003; Kinnas and Fine 1993). A further challenge arises when in comes to model validation, since the most complete experimental datasets such as the experimental data obtained from the RoboTuna project, were obtained using relatively shallow towing tanks, free surface effects will have been prevalent within the data. However most CFD modellers work with the assumption that the vehicle is at sufficient depth to ignore free surface effects, thus avoiding the added computational cost. Still further complication is added when considering interactions between cavitation and a free surface. A boundary element CFD method for the solution of such flows can be found in Bal and Kinnas (2002).

3.4 Discussion

3.4.1 Swimming Performance

The design history of biomimetic swimming devices has shown a strong general tendency towards simplification with fewer moving parts and simpler control requirements. In several cases this drive for simplification seems to have resulted in under engineered systems which have not lived up to their design expectations,

for instance the authors of Anderson and Kerrebrock (2000),Kumph (2000) and Watts (2009), all noted disappointment in the maximum speed demonstrated by their vehicles.

The shortfalls in speed among BCF swimming robots can be attributed to two possible causes. The possible causes are insufficient actuation power/speed or poor swimming kinematics. The authors of Anderson and Kerrebrock (2000) and Kumph (2000) suggested that actuator saturation was the major factor in limiting top speeds, each limited to a maximum tail beat frequency of $1Hz$. However a biological study carried in the 1950s demonstrated that for BCF swimmers, similar tail beat frequencies result in similar swimming speeds relative to body length regardless of specific morphology (Bainbridge 1958). Yet as can be seen from figure 3.12 there is little correlation among the tail beat frequencies and swimming speeds reported. For example the NMRI PF-700 and UPF-2001 each achieved tail beat frequencies of $10Hz$ resulting in a speed of $1Ls^{-1}$, whereas the Istanbul Technical University's Robot Dolphin managed $1Ls^{-1}$ at a tail beat frequency of only $1.35Hz$. This lack of cohesion between swimming speeds and tail beat frequencies suggests that many of the Biomimetic swimmers featured were operating with poor swimming kinematics. Assuming that studies featured to the left of the plot are more optimal it would then be desirable to ensure that future developed Biomimetic swimmers lie in approximate correlation which is roughly given by the line $Speed\ (L/S) = 0.5 \times Frequency\ (Hz)$.

It was shown in Triantafyllou and Triantafyllou (1995), that swimming efficiency is extremely sensitive to variation in kinematic parameters. Since propulsion efficiency was one of the main motives for developing biomimetic BCF swimming devices, it should be considered a high priority of future developments to ensure that good kinematics can be achieved.

The level of kinematic control available from a given design will inevitably determine

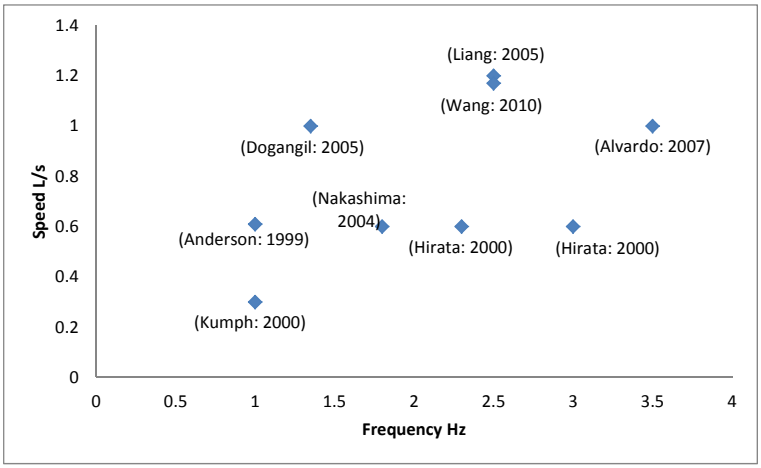


Figure 3.12: A graph showing the relationship between maximum tail beat frequency and resultant speed for BCF swimmers reported in this chapter

the kinematic parameter optimization approaches available to developers. Whilst complex designs that offer high levels of kinematic control are suitable for on line optimization. Simpler designs with relatively low levels of kinematic control require off line optimization in the design phase, to ensure that good kinematics are achievable.

Advances in fluid dynamic modelling have increased the effectiveness of off line kinematic parameter optimization. New techniques in computational fluid dynamics and analytical analysis of flapping foils has given developers a myriad of new tools. A review of research into modelling flapping foils can be found in Triantafyllou et al. (2003).

3.4.2 Actuator Selection

The choice of actuators is also a subject of some interest in the development of biomimetic swimming devices. Brushless DC servo motors are a popular choice with developers, the combination of high torque density and a high level of controllability

make them a good candidate for actuator selection. However high performance DC servo motors can be very expensive.

Several developers have opted for the combination of a continuously run DC motor and a mechanical oscillator such as a Scotch yoke. A scotch yoke can convert rotational motion to a simple harmonic oscillation and allow the use of relatively cheap motors. By using larger continuously run motors higher tail beat frequencies can be achieved. However such designs result in a reduction in controllability.

Pneumatic or hydraulic systems can offer excellent power densities and good controllability, depending on the arrangement of the control system. Hoses can easily penetrate pressure vessels without producing complex sealing problems making them an appealing choice for deep sea applications provided the ambient pressure has no effect on the operational cost. This could be problematic for McKibben type pneumatic actuators. A further benefit of piston or bellows type actuators is that they produce linear actuation similar to biological muscle, this makes them an appealing option from the biomimetics point of view. However such actuators require large support systems and may have lower actuation speeds than electrical systems.

It is widely expected that artificial muscle will play a role of increasing importance in biomimetic design. Several of the current technologies can vastly outperform biological muscle in strain and stress generating characteristics. However current technologies are not without their weaknesses; SMA has a short cycle life, and relies on thermal energy loss to work, resulting in low energy efficiency. IPMC and piezoelectric actuators are currently not available in large enough sizes for large scale robot actuation and Carbon nano tubes are currently cost prohibitive. Dielectric elastometer actuators possibly offer the most interesting solution at present, however their reliance on high voltages for actuation make the prospect of using them in aquatic environments unappealing.

As well as the choice of actuators, developers have been contemplating the number

of actuators required. After the early prototype BFC swimmers developed by MIT and Draper, most developers have opted for a two actuator design. With two active joints it is possible to actively move the caudal fin through a wide variety of swimming kinematics. Despite this there are some like the Istanbul Institute of technology robot dolphin team and the university of Essex robotic fish team who are still opting for a three or more actuator design. On the other hand there is the emergence of single actuator designs such as the Tokyo Institute of Technology's robotic dolphin series and MIT's compliant swimming devices which suggest that satisfactory kinematics can be achieved through the use of passive joints or compliant body sections of appropriately elastic modulus. Reduction in the number of required actuators has a dramatic cost and complexity saving implication

3.4.3 Hull Design

The design of flexible hull structures has also seen progress. From the early free flooded or open tail sections, there seems to be a trend towards closed dry tail sections. The development of flexible exostructures, that maintain internal volume such as the fin ray structure have enabled the use of sealed dry tail sections. However how well such structures can resist the high pressures of deeper operations remains to be seen. Moulded compliant bodies are also emerging as a popular option for flexible hull design.

The moulded body designs used on MIT's compliant body devices and Beihang University's Robo-Ray II, demonstrated that it can be a robust construction method and provided no air pockets are left by the moulding process can result in good pressure resistance. However for long duration deployments cooling of actuators could become a problem as such designs do not allow any internal thermal convection. Furthermore the lessons learned from the Tufts Ray project suggest that this construction method might not be suitable for larger constructions.

The tendency in traditional AUVs towards modular cylindrical designs suggests the possibility of a market for interchangeable propulsion modules, as used by Watts (2009). The PF-600, PF-700 and SPC-III have demonstrated that BCF propulsion can be effective with cylindrical body forms. SPC-III also demonstrated that reasonable turning performance can still be achieved with a torpedo shaped rigid bodied BCF swimmer.

3.4.4 PMF swimmers

Developers working on PMF swimming modes seem to have displayed a preference towards pectoral fin labriform type swimming. It has been shown that paired fins mounted orthogonal to the body and actuated with two degrees of freedom can produce propulsive forces for manoeuvres in a full six degrees of freedom. It has also been demonstrated that such pectoral fin propulsion systems can be constructed with a modular architecture, making them compatible for retrofitting existing rigid body AUVs (Bandyopadhyay 2005).

3.4.5 Comparison with traditional AUVs

PMF swimmers share many of the manoeuvrability characteristics of the multiple axis thruster AUVs in current operation. Their station keeping make them suitable for missions involving physical interaction such as sub sea welding, or mineral sampling on the sea bed. However the scalability may be an issue.

BCF like torpedo type AUVs require forward motion to achieve manoeuvrability. However once forward motion is established the manoeuvrability of BCF swimmers is far superior. The BCF type robotic swimmer is a good candidate for survey missions where manoeuvrability is critical, such as shallow water found on littoral zone, or in the presence of submersed obstacles. The necessity for body deformation would perhaps be a restrictive factor on the maximum depths achievable.

The complexity is also a major factor limiting commercial viability as higher complexity also affects both the cost of purchase and maintenance. The tendency towards fewer actuators reflects a desire to keep costs low and maintenance simple.

3.5 Concluding remarks

This chapter has presented a comprehensive review of the currently available literature relevant to biomimetic swimming devices.

Based on this review it has been concluded that further development of biomimetic swimming machines, requires development of offline optimisation methods for optimisation of design and control without capital expenditure associated with physical prototyping.

Furthermore it has been concluded that gait generation and controller selection are critical factors determining propulsion efficiency.

Whilst PMF swimmers offer greater manoeuvrability, by their nature they require multiple actuated fins. Whilst multiple fins may afford a degree of redundancy they also necessarily increase complexity and weight. As such the remainder of this study will focus on BCF swimming mechanisms.

Although moulded silicon type bodies such as utilised by MIT's Compliant swimmers offer the simplest design principal, the resultant devices are restricted in the available kinematics compared to multiple vertebrae type designs such as Essex G9. Furthermore it is thought that multiple vertebrae type designs have more scope for integration into existing modular AUV designs and offer better scalability.

In the next chapter the focus will be on the modelling of a multiple rigid vertebrae BCF type robotic fish.

Chapter 4

Modelling of a robotic fish

This chapter will describe a harmonic oscillator model of a simple robotic fish, capable of generating a body and caudal fin (BCF) locomotive gait.

4.1 Introduction

In chapter 3 the design and performance of numerous physical prototype robotic fish were discussed. Whilst ultimately physical prototypes offer the most robust proof of concept, the rigidity of physical prototypes limits the scope of experimentation. A mathematical model on the other hand can be more easily and quickly adapted and optimised.

The goal of this chapter is to develop an adaptable model for a robotic fish, suitable for an experimental study. The model must be generic to enable adaptation to changes in parameters and geometry, the model must bear an acceptable resemblance to reality i.e. must relate to a physically realizable device.

Mason (2003) modelled a fish as a 2 dimensional deformable Joukowski foil, solving fluid flow using semi-steady state analytical methods. Such an approach does not include unsteady fluid effects such as vortex interactions. Moreover by sacrificing the third dimension orthogonal to the plane of tail movement the effects of caudal, dorsal and anal fins were completely neglected. Furthermore, variations in body profile were also completely neglected. Whilst such omissions may be acceptable for

the modelling of anguilliform swimming the Joukowski foil method precludes the presence of multiple waves within the body, or varying of amplitude along the wave. Some studies have neglected the body dynamics altogether and simply modelled the fluid around the caudal fin (Pedro et al. 2003). Such an approach could be valid for applications where the size of the caudal fin is large in proportion to the profile of the anterior section of the body, however it does not include the effect of tail motion on heading. Terzopoulos et al. (1994) presented an intricate deformable fish body model based on 91 springs forming a deformable mesh. However this study was primarily for simulation purposes. It is unlikely that anyone would actually attempt to build a physical device of such complexity.

Dogangil et al. (2005*b*) used a 4 DOF kinematic chain combined with linear fluid lift and drag equations. The simplicity of such an approach makes it highly attractive. However, it could be argued that the omission of unsteady fluid effects may have resulted in an oversimplification.

The approach taken in this study is to integrate a kinematic chain model of the robotic fish, with a finite element solution of the Navier Stokes equations of motion on the surrounding fluid. This will hopefully result in the integration of unsteady fluid effects with the dynamics of the fish. The system will be modelled with torque generating actuators placed at the joints in keeping with the use of DC servo motors, which were identified as the most popular actuator choice for robotic fish in chapter 3. For simplicity of modelling cavitation effects will be omitted from the CFD simulation study allowing the use of an incompressible flow solving algorithm.

The remainder of this chapter is divided in to three further section. Section 4.2 will describe a generic model of a robotic fish as a free floating kinematic chain. Section 4.3 will give an overview of the specific simulation arrangement used throughout this study, describing both the physical geometry and the finite element analysis of the

surrounding fluid. Finally section 4.4 will present some concluding remarks.

4.2 Modelling a robotic fish as a free floating kinematic chain

A common feature of the majority of the BCF style robotic swimming machines surveyed in chapter 3 was a rigid multi-body construction with either actuated or passive joints between the vertebrae constrained to rotate about a single axis.

Such a vertebrae structure can be modelled as a planar free floating multi body kinematic chain as shown in figure 4.1.

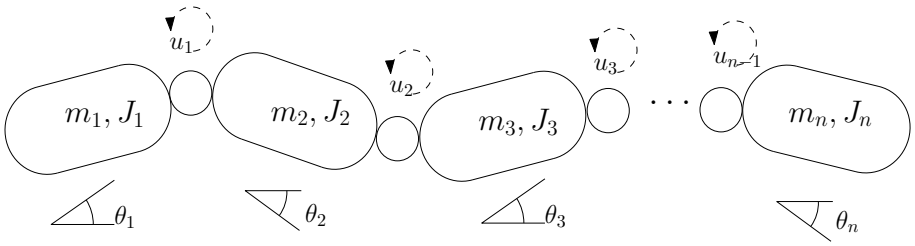


Figure 4.1: Free floating kinematic chain

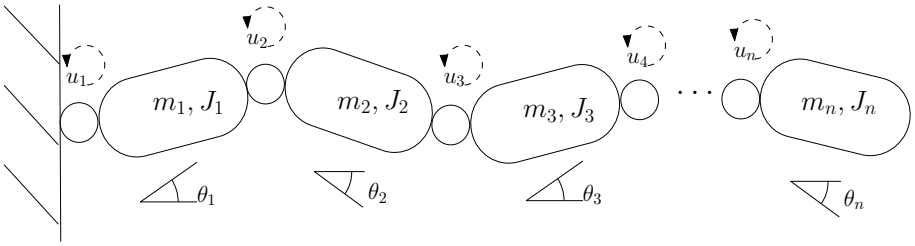


Figure 4.2: Anchored kinematic chain

Figure 4.2 shows an n bodied anchored kinematic chain. The position is able to move in n degrees of freedom. There are $n-1$ inputs in the form of torques between joints and 1 torque between the anchor point and the chain. It is logical to use the joint positions as the degrees of freedom so that inputs align with states.

Figure 4.1 shows an n body free floating kinematic chain, disregarding global translation but including global yaw the position of the bodies relative to the global

centre of mass is able to move in n degrees of freedom. There are $n-1$ inputs in the form of torques between the bodies. As there are fewer inputs than degrees of freedom, the system is underactuated and as such it is no longer logical to align states with inputs.

In this study the global bearing of each of the bodies is taken as a degree of freedom and will be denoted individually as θ_i or in vector form as $\mathbf{q}$.

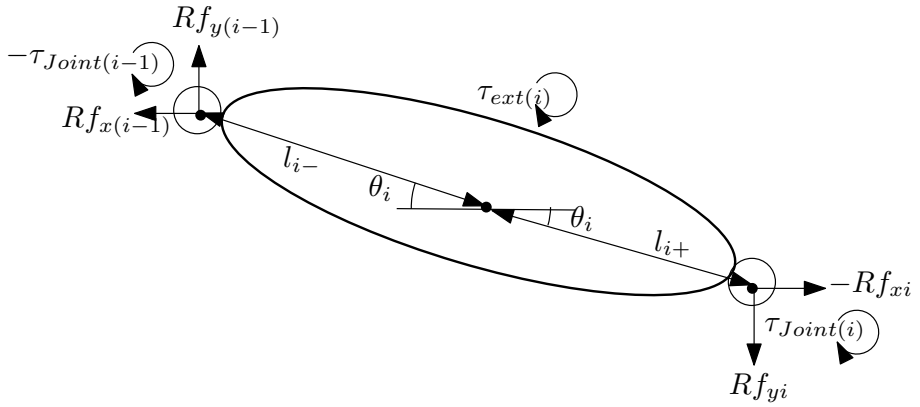


Figure 4.3: A summary of torques applied to a given body within a kinematic chain

The motion of each body can then be determined by summing all the torque forces and applying Newton's 2^{nd} Law. Figure 4.3 shows a summary of all the torques experienced by a given body i within a kinematic chain, where $\tau_{ext(i)}$ is the external torque experienced as a result of external stimulus, $\tau_{Joint(i)}$ is the torque experienced at joint i due to the summation of joint actuation torques and spring torques incorporated into this model due to observations made that fish in nature use passive muscle tension to generate spring forces in parallel with active muscle tension (Pabst 1996), finally the linear reaction forces at the joints $Rf_{y,(i)}$ and $Rf_{x,(i)}$ will generate torques. Also shown within the figure l_{i+} and l_{i-} are the lengths from the forward of the body and the aft of the body to the centre of mass respectively and θ_i is the bearing of the body.

$$\tau_{Rf} = \begin{bmatrix} -diag\{sin(\mathbf{q})\}\mathbf{L} & diag\{cos(\mathbf{q})\}\mathbf{L} \end{bmatrix} \begin{bmatrix} Rf_x \\ Rf_y \end{bmatrix} \tag{4.1}$$

Where $\mathbf{L}$ is the $n \times (n-1)$ matrix of lengths given by,

$$\mathbf{L} = \begin{bmatrix} l_{1-} & 0 & 0 & \dots & 0 \\ l_{2+} & l_{2-} & 0 & \dots & 0 \\ 0 & l_{3+} & l_{3-} & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & l_{n+} \end{bmatrix} \tag{4.2}$$

Resolving the reaction forces at the joints Rf_x and Rf_y first requires the resolution of the relative linear acceleration experienced by the bodies.

By taking the sum of the moments from the local centre of mass to the centre of mass of each of the other bodies, then dividing by the total mass. The coordinates of the local centre of mass of each body can be calculated relative to the global centre of mass as a function of the vector of degrees of freedom $\mathbf{q}$. The $\mathbf{x}$ and $\mathbf{y}$ coordinates of each body's centre of mass relative to the global centre of mass can be found as,

$$\begin{bmatrix} \mathbf{x} & \mathbf{y} \end{bmatrix} = \mathbf{M} \begin{bmatrix} cos(\mathbf{q}) & sin(\mathbf{q}) \end{bmatrix} \tag{4.3}$$

where $\mathbf{M}$ is the (nxn) matrix of moments divided by total mass given by,

$$M_{i,j} = -\frac{l_{j-}\left(\sum_{k=j+1}^{n}\gamma(i,k)m_k - \sum_{k=0}^{j}\gamma(i,k)m_k\right) + l_{j+}\left(\sum_{k=j}^{n}\gamma(i,k)m_k - \sum_{k=0}^{j-1}\gamma(i,k)m_k\right)}{\sum m} \tag{4.4}$$

and $\gamma(i,k)$ is the function,

$$\gamma(i,k) = \begin{cases} 0 & i = k \\ 1 & otherwise \end{cases} \tag{4.5}$$

The i^{th} row of $\mathbf{M}$ gives the position of the i^{th} body relative to the global centre of mass. l_{j-} and l_{j+} are the froward and aft lenghts of the j^{th} body relative to its centre of mass.

$\mathbf{M}$ is time invariant. Therefore the relative internal velocities and accelerations of the bodies can be found through differentiating equation 4.3 twice with respect to time as such,

$$\begin{bmatrix} \dot{\mathbf{x}} & \dot{\mathbf{y}} \end{bmatrix} = \mathbf{M} \begin{bmatrix} -sin(\mathbf{q})\dot{\mathbf{q}} & cos(\mathbf{q})\dot{\mathbf{q}} \end{bmatrix} \tag{4.6}$$

where $\begin{bmatrix} \dot{\mathbf{x}} & \dot{\mathbf{y}} \end{bmatrix}$ and $\begin{bmatrix} -sin(\mathbf{q})\dot{\mathbf{q}} & cos(\mathbf{q})\dot{\mathbf{q}} \end{bmatrix}$ is an $n \times 2$ vector. Similarly,

$$\mathbf{a} = \begin{bmatrix} \ddot{\mathbf{x}} & \ddot{\mathbf{y}} \end{bmatrix} = \mathbf{M} \begin{bmatrix} -sin(\mathbf{q})\ddot{\mathbf{q}} - cos(\mathbf{q})\dot{\mathbf{q}}^2 & cos(\mathbf{q})\ddot{\mathbf{q}} - sin(\mathbf{q})\dot{\mathbf{q}}^2 \end{bmatrix} \tag{4.7}$$

By Newtons 2^{nd} law the transposed $(n-1) \times 2$ matrix of reaction forces at the joints can be calculated as,

$$Rf^T = \begin{bmatrix} Rf_x^T & Rf_y^T \end{bmatrix} = \mathbf{M_2} \left(\mathbf{a} + \mathbf{A} - \frac{\mathbf{F}_{ext}}{\mathbf{m}} \right) \tag{4.8}$$

Where $\mathbf{F}_{ext}$ is the vector of external forces acting on each of the bodies, $\mathbf{m}$ is the vector of masses of the bodies, $\mathbf{a}$ is the local acceleration given by equation 4.7 and $\mathbf{A}$ is the acceleration experienced by the global centre of mass given by,

$$\mathbf{A} = \frac{\sum \mathbf{F}_{ext}}{\sum \mathbf{m}} \tag{4.9}$$

and $\mathbf{M_2}$ is the $(n-1)\times n$ matrix made up of masses given by,

$$\mathbf{M_2}(i,k) = \begin{cases} m_i & i \leq k \\ 0 & otherwise \end{cases} \tag{4.10}$$

$$= \begin{bmatrix} m_1 & 0 & 0 & \dots & 0 \\ m_1 & m_2 & 0 & \dots & 0 \\ m_1 & m_2 & \ddots & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ m_1 & m_2 & \dots & m_{n-1} & 0 \end{bmatrix} \tag{4.11}$$

By applying Newtons 2^{nd} law the motion can then be described by the second order system of ordinary differential equations,

$$\mathbf{J\ddot{q}} = \sum \tau = (\tau_{Rf} + \tau_{Ext} + \tau_u) \tag{4.12}$$

Substituting equations 4.1, 4.7 and 4.8 into the term τ_{Rf} yields,

$$\begin{aligned} \mathbf{J\ddot{q}} = &\left(diag\{sin(\mathbf{q})\}\mathbf{LM_2}\left(\mathbf{M}\left[-sin(\mathbf{q})\ddot{\mathbf{q}} - cos(\mathbf{q})\dot{\mathbf{q}}^2\right] + \mathbf{A_x} - \frac{\mathbf{F}_{ext_x}}{\mathbf{m}}\right)\right) \\ &- \left(diag\{cos(\mathbf{q})\}\mathbf{LM_2}\left(\mathbf{M}\left[cos(\mathbf{q})\ddot{\mathbf{q}} - sin(\mathbf{q})\dot{\mathbf{q}}^2\right] + \mathbf{A_y} - \frac{\mathbf{F}_{ext_y}}{\mathbf{m}}\right)\right) \\ &+ \tau_{Ext} + \tau_u \end{aligned} \tag{4.13}$$

This can be transformed into the standard manipulator form,

$$\mathbf{H}(\mathbf{q},\dot{\mathbf{q}})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{Kq} + \mathbf{D} + \mathbf{Bu} = \mathbf{0} \tag{4.14}$$

Where the matrix function $\mathbf{H}(\mathbf{q},\dot{\mathbf{q}})$ is given by,

$$\mathbf{H} = \mathbf{J} + \begin{bmatrix} diag\{sin(\mathbf{q})\} & diag\{cos(\mathbf{q})\} \end{bmatrix} \mathbf{LM_2M} \begin{bmatrix} diag\{sin(\mathbf{q})\} \\ diag\{cos(\mathbf{q})\} \end{bmatrix} \tag{4.15}$$

and the matrix function $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ is given by,

$$\mathbf{C} = \mathbf{C_{const}} + \begin{bmatrix} -diag\{sin(\mathbf{q})\} & diag\{cos(\mathbf{q})\} \end{bmatrix} \mathbf{LM_2M} \begin{bmatrix} diag\{cos(\mathbf{q})\} \\ diag\{sin(\mathbf{q})\} \end{bmatrix} diag\{\mathbf{q}\} \approx 0 \forall \mathbf{q} \in \mathbf{R_n} \tag{4.16}$$

$\mathbf{C_{const}}$ is the constant damping matrix corresponding to dampening coefficients of the joints in the form,

$$\mathbf{C_{const}} = \begin{bmatrix} c_1 & -c_1 & 0 & 0 & \dots & 0 \\ -c_1 & c_2+c_1 & -c_2 & 0 & \dots & 0 \\ 0 & -c_2 & c_2+c_3 & -c_3 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & -c_n & c_n \end{bmatrix} \tag{4.17}$$

and $\mathbf{K}$ is a spring matrix included to simulate the presence of axial springs at the joints, $\mathbf{K}$ will be in the form of an $n \times n$ time invariant singular matrix with the following structure,

$$\mathbf{K} = \begin{bmatrix} k_1 & -k_1 & 0 & 0 & \dots & 0 \\ -k_1 & k_2+k_1 & -k_2 & 0 & \dots & 0 \\ 0 & -k_2 & k_2+k_3 & -k_3 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & -k_n & k_n \end{bmatrix} \tag{4.18}$$

$\mathbf{D}$ will be the composition of the pure external disturbance torques and the torque

components of τ_{Rf} corresponding to the external linear forces.

If the robot fish n body system has $n-1$ controllable pure torque inputs at the joints, the corresponding $n \times (n-1)$ controllable input matrix $\mathbf{B}$ will be,

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & -1 \end{bmatrix} \tag{4.19}$$

or a singular input at joint i will correspond to an $n \times 1$ input matrix $\mathbf{B_i}$ where all the values are 0 except the i^{th} which equals 1 and the $(i+1)^{th}$ which equals -1.

4.3 Overview of Simulation Study

4.3.1 Structure

The starting point for the structural element of the simulation was a three dimensional model generated by using a CAD software package. The base geometry selected comprised of an approximately tuna shaped hull orientated facing forward along the X axis, symmetrical over the XY plane and the XZ plane. Added to the hull was a thin caudal fin orientated in the ZX plan, again symmetrical over the XY and XZ planes. Dorsal, anal and pectoral fins were omitted, a third angle view of the geometry can be found in figure 4.4. As can be seen the plan and profile cross sections of the body appear as smooth hydrofoils. The caudal fin is a lamina fin shape.

The key dimensions of the model were as follows; length $1m$, maximum beam $0.15m$, maximum height $0.3m$ and total volume $0.23849m^3$. This gave the fish a forward facing area of $\approx 0.0707m^2$.

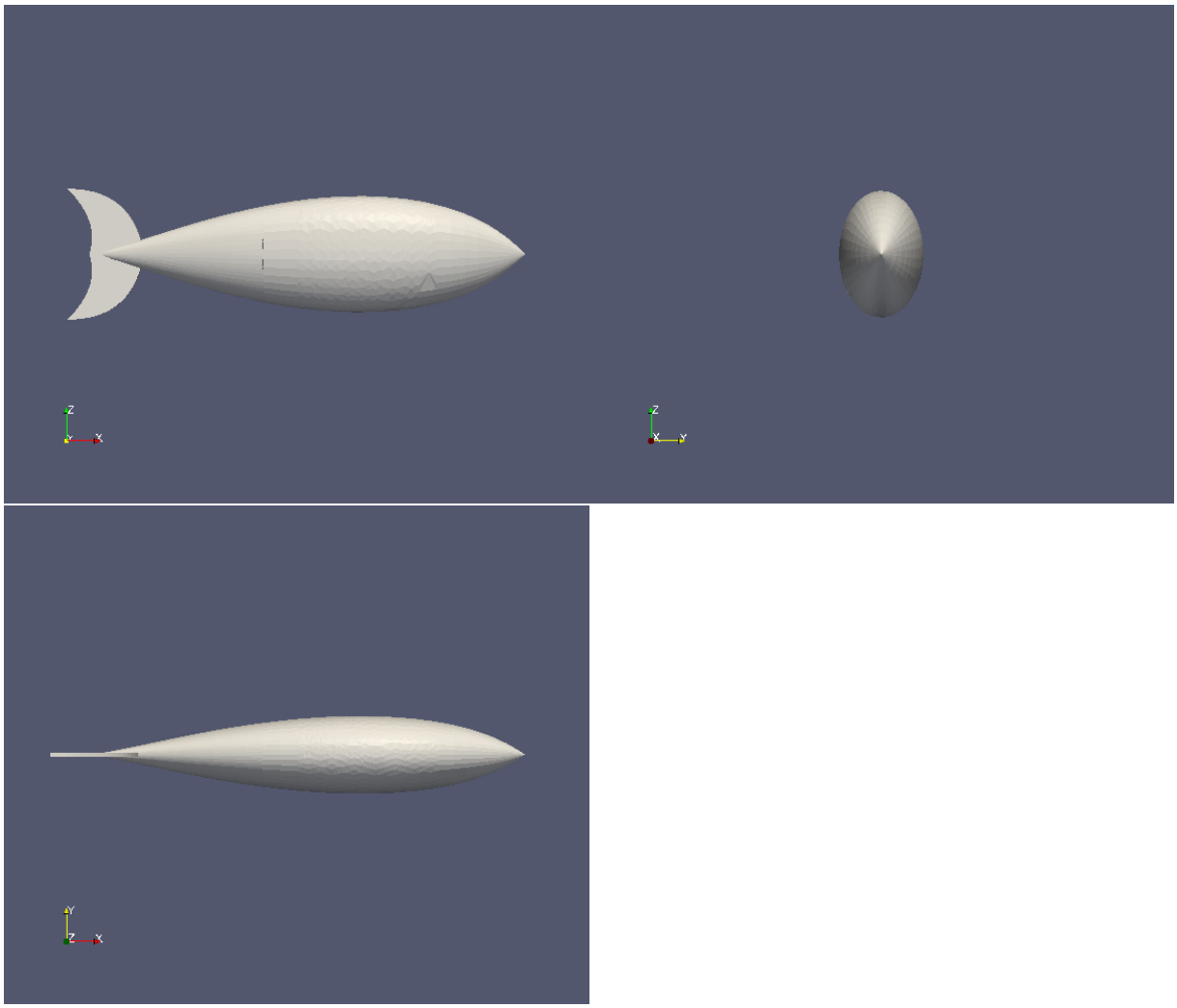


Figure 4.4: 3*D* geometry used for simulations

Steady state CFD simulation of the body stationary in a flow of $0.2ms^{-1}$ gave the drag force as $\approx 0.063176N$. The non dimensional coefficient of drag is given by

$$c_d = \frac{2F_d}{\rho v^2 A} \tag{4.20}$$

where F_d is the measured drag force, ρ is the density of the surrounding fluid and A is the cross sectional area. The nondimensional drag coefficient of the body was calculated as $c_d \approx 0.04468$ which is typical for a streamlined body in lamina flow.

The geometry was divided into four sections with divisions parallel to the ZY plane as

shown in figure 4.5, these four sections were then modelled as four rigid bodies within a kinematic chain with joints located on the X axis at the intersection on the sections.

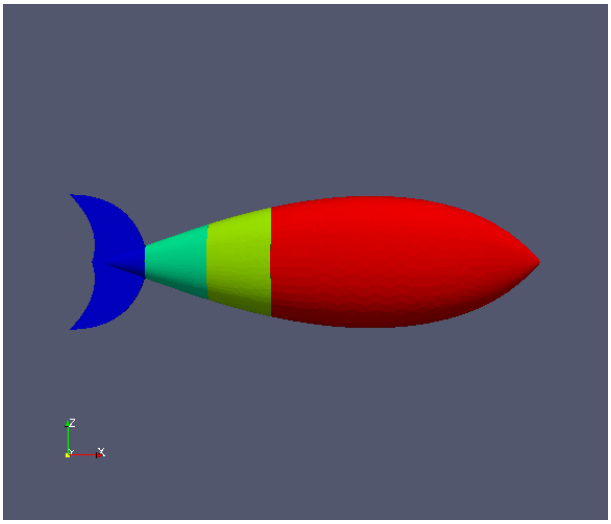


Figure 4.5: Sectional division of fish geometry

The forward section shown in red in the figure comprised of 60% of the total body length, in keeping with tunniform locomotion where movement is restricted to the anterior 40% of the body. Locating the joint between the forward most two segments at the origin, and the other two joints on the X axis at locations $x = [\ -0.14 \quad -0.28\ ]m$ respectively, the centre of mass of the four bodies was found on the X axis at locations $x = [\ 0.2422 \quad -0.06175 \quad -1.193 \quad -0.349\ ]m$ respectively. Assuming the body had uniform density equal to the density of fresh water the masses of the bodies were calculated as, $[\ 19.18 \quad 3.226 \quad 1.155 \quad 0.288\ ]Kg$ respectively. The rotational moment of inertia of each body around the Z axis at its centre of mass for each body was found to be

$$[\ 4.07e-1 \quad 1.336e-3 \quad 2.2e-4 \quad 1.797e-4\ ]Kg\ m^2$$

Substituting the mass and position data for the bodies into equation 4.4 gave the matrix

of moments ($\mathbf{M}$) as,

$$\mathbf{M} = \begin{pmatrix} 4.7416e-2 & 1.6824e-2 & 4.2574e-3 & 8.3324e-4 \\ -1.9478e-1 & -4.4926e-2 & 4.2574e-3 & 8.3324e-4 \\ -1.9478e-1 & -1.2318e-1 & -4.8743e-2 & 8.3324e-4 \\ -1.9478e-1 & -1.2318e-1 & -1.3574e-1 & -6.8167e-2 \end{pmatrix} \tag{4.21}$$

substituting the same mass and position data into equations 4.2 and 4.10 then taking the product yields,

$$\mathbf{LM_2} = \begin{pmatrix} 4.64540 & 0 & 0 & 0 \\ 2.68520 & 0.25243 & 0 & 0 \\ 2.68520 & 0.45164 & 0.10049 & 0 \\ 1.32342 & 0.22259 & 0.07969 & 0 \end{pmatrix} \tag{4.22}$$

These values for $\mathbf{M}$ and $\mathbf{LM_2}$ have been used throughout this study. A summary of the properties of the four sections can be found in table 4.1 and joint positions are shown in table 4.2.

Table 4.1: Summary of section properties

Section	1	2	3	4
mass (Kg)	19.18	3.226	1.155	0.288
J ($Kg\ m^2$)	4.07e-1	1.336e-3	2.2e-4	1.797e-4
COM $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ (m)	$\begin{bmatrix} 0.2422 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.06175 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -1.193 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.349 \\ 0 \\ 0 \end{bmatrix}$

Table 4.2: Joint location

Joint	1	2	3
Position $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ (m)	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.14 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.28 \\ 0 \\ 0 \end{bmatrix}$

4.3.2 Fluid

The fluid interactions were solved using the open source finite element solver openFoam^{TM}. In order to include fluid interactions with the motion of the body a dynamic mesh fluid solver was required. The native openFoam pimpleDyMFoam solver was selected which is a multiphase solver utilizing the PIMPLE algorithm, alongside a dynamic mesh solver.

The PIMPLE algorithm is a hybrid of the *semi implicit method for pressure linked equations* (SIMPLE) and *pressure implicit with splitting operators* (PISO) algorithms for the solution of the Navier Stokes equations of motion common to computational fluid solvers. The combination aims to maintain the robustness, accuracy and low computational cost of the PISO along with the comparability of SIMPLE with additional equations for turbulent transport terms (Barton 1998).

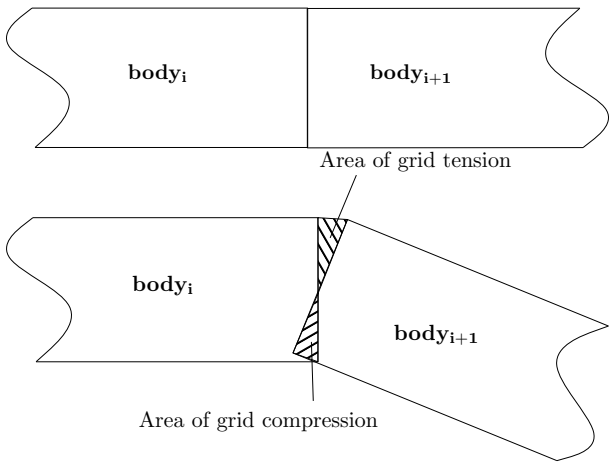


Figure 4.6: Localized grid tension and compression of rigid body movement

Limitations of rigid body mesh transformation proved restrictive. It was discovered that if rigid bodies were used the grid will suffer from localized stretching and compression around the joints. Figure 4.6 illustrates this problem, as the angle between the bodies increases the compression in the shaded area below the joint increases as

does the tension in the shaded area above. This resulted in significant restrictions on the maximum amplitude of joint movement in order to avoid grid elements becoming prohibitively small or large.

One possible solution considered was to regularly re-mesh whenever mesh elements became overly deformed, however such a method was deemed to be prohibitively computationally expensive.

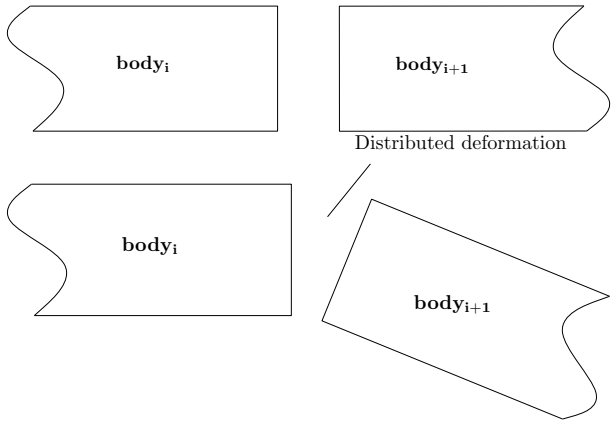


Figure 4.7: Disconnected body deformation

A second solution considered was to disconnect the segments providing space between the segments to allow for distribution of mesh deformation as illustrated by figure 4.7. However it was suggested that the discontinuity of the resultant body surface could cause additional turbulence and was unrealistic in terms of potential application.

The final solution arrived at was to include a deformable segment of the geometry between the rigid segments. This is illustrated in figure 4.8 where the deformable sections distribute the deformation. The result was that mesh deformations are then distributed over the deformable segment of the geometry rather than localized at the joint. This was considered in-keeping with the use of a flexible membrane to cover rigid vertebrae design principal used in several robotic fish featured in chapter 3.

Therefore the boundaries between rigid segments from figure 4.5 were separated by

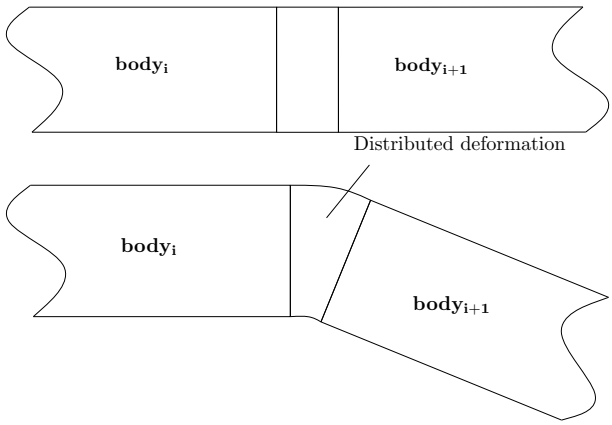


Figure 4.8: Distributed mesh deformation

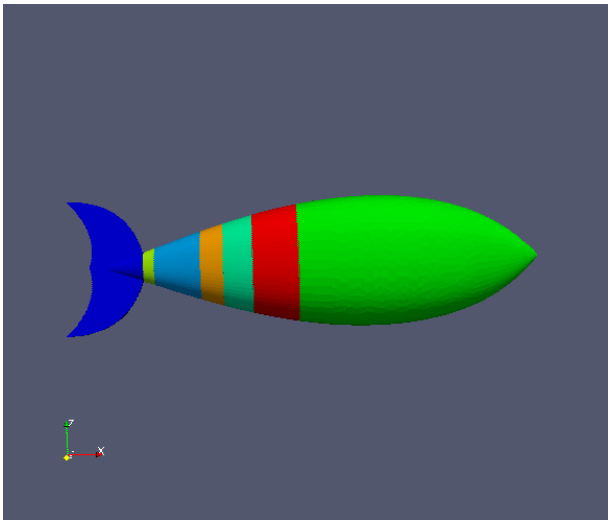


Figure 4.9: Segment map of geometry

deformable segments. Figure 4.9 shows the final segmentation of the geometry, with rigid segments alternating with deformable segments. The green, turquoise, light blue and dark blue segments are rigid and the red orange and yellow segments are deformable.

The fluid domain was selected as a cylinder orientated along the X axis, with centre at the origin, a total length of $6m$ and a radius of $2m$. Since the geometry was selected to be symmetrical about the XY plane and all rotation was constrained to be around the

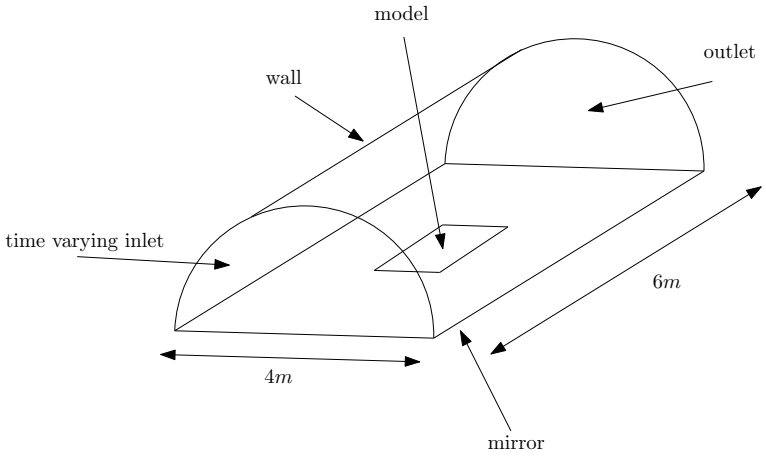


Figure 4.10: CFD domain

Z axis, it was assumed that the fluid motion would also be symmetrical about the XY plane. A diagram of the CFD domain complete with dimensions can be found in figure 4.10.

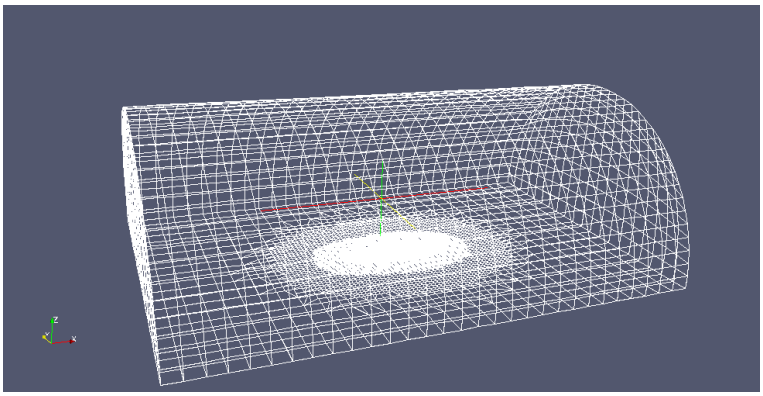


Figure 4.11: Meshed CFD domain

The original mesh was generated with the native openFoam mesh generation algorithm snappyHexMesh. As the primary interest was the fluid structure interactions rather than the far field flow, a coarse far field mesh was selected with graduated refinement towards the structure of interest. Figure 4.11 shows a 3D image of the mesh used, the densely meshed area in the centre contained the robotic fish.

Figure 4.12 demonstrates that the near field mesh was fine enough to capture turbulent

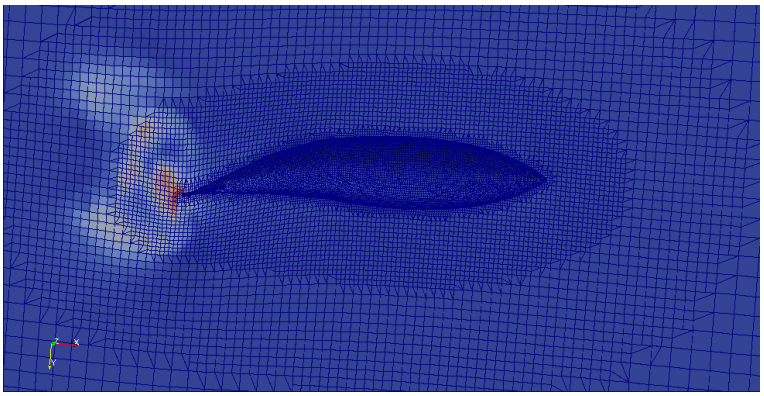


Figure 4.12: Vortices shed from caudal fin during motion

effects such as vortex formation. A pair of attached vortices can be seen at tail, and a further two shed vortices can be seen down stream.

The mesh deformations were solved using the native openFoam mesh deformation algorithm displacmentLaplacian with inverse distance diffusivity. Figure 4.13 shows the mesh surrounding the robotic fish in a flexed position. This demonstrates that the deformable segment approach taken allowed for significant body deformation whilst maintaining good uniformity of the surface mesh.

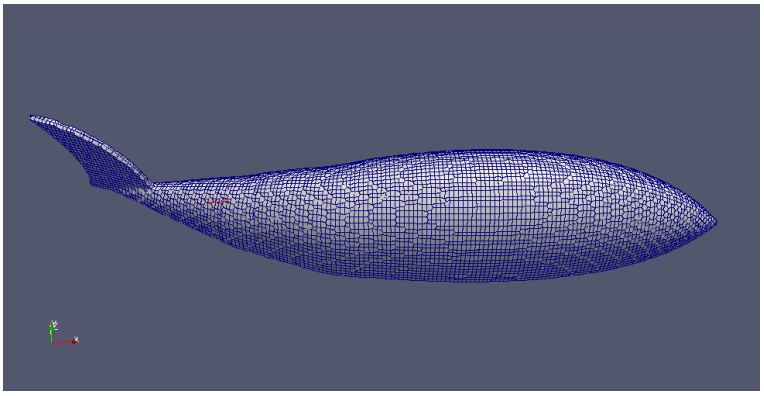


Figure 4.13: Surface mesh remains comparatively uniform throughout body motion

Exogenous inputs to the mechanical system could then be calculated at each time step through integration of pressure over the surface to calculate fluid induced forces and

torques.

4.3.3 Coupling fluid and structure

For simplicity an explicit step approach was used in conjunction with a coupling relaxation and a very small time step. The *ordinary differential equations* (ODEs) were resolved with a time step of the order of $1e-5s$ to compensate for the high degree of coupling between fluid and structure models. It was found that this approach resulted in a stable coupling between the two models although it is conceded that an implicit algorithm may have significantly reduced computational cost.

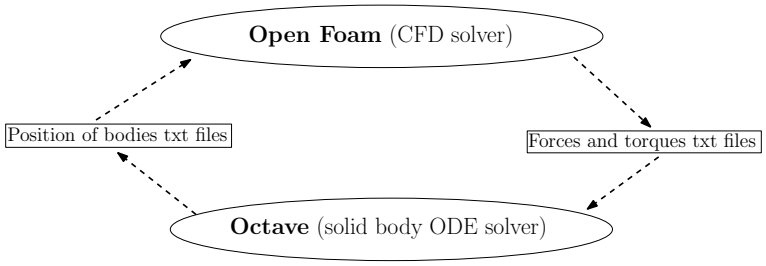


Figure 4.14: Flow diagram of solid body and CFD simulation

Figure 4.14 shows a flow diagram illustrating the coupling between the fluid and structure solvers. Full details of the hardware and software used including all code produced for this study can be found in appendix B.

4.4 Concluding Remarks

In this chapter a nonlinear state space model for a robotic fish modelled as an n bodied free floating kinematic chain has been derived.

The model incorporates a spring term, with the goal of storing energy during the non inertia phases of periodic motion in order to reduce the negative work characteristically associated with periodic motion.

Further to this fluid interactions have been solved on a dynamic meshed finite element

Navier Stokes solver, using the open source CFD software package openFoam^{TM}.

The geometry used is a closed shell with alternating rigid and flexible section to allow for smooth mesh deformations. This is in keeping with a sheathed vertebrae design used by several examples discussed in chapter 3.

Chapter 5 will go on to discuss methods for generating an effective propulsion gait for the model.

Chapter 5

Energy based gait generation for an underactuated robotic fish

This chapter will discuss gait generation for an underactuated body and caudal fin (BCF) type robotic fish based on the control of state energy.

5.1 Introduction

In chapter 4 a kinematic model of a free floating robotic fish was described. This chapter will focus on the production of an effective swimming gait for the model.

As illustrated in chapter 3 generation of an appropriate gait is critical in producing effective propulsion.

In chapter 4 it was mentioned that a free floating kinematic chain is necessarily underactuated by at least a degree of one, meaning that there exists at least one fewer control input than degrees of freedom. The degree of actuation of a system is determined by the number of actuators compared to the number of degrees of freedom. For instance if a robotic manipulator is designed to move in $\mathbf{R}_6$ space and possesses 6 actuators i.e. one for each degree of freedom. The robot is said to be fully actuated. If however the robot possesses < 6 actuators the robot is said to be underactuated. Conversely if the robot possesses > 6 actuators then it can be described as over actuated.

Over actuated devices have often been used in critical applications where a degree of redundancy is required. However the control of underactuated devices has been an area of growing interest. Traditionally robotic engineers have pursued a geometric approach to kinematic motion control, i.e. at time **t** position should be **x**. Whilst such a control approach can in theory generate any given kinematic pattern for fully actuated systems. When significant unmodelled disturbances are present accurate geometric control requires a high degree of stiffness in the control which results in highly energy intensive movement. Furthermore interactions with unsteady fluid effects, such as the vortex interactions as mentioned in chapter 2 require a degree of passivity for the gait to adapt to the surrounding flow. A more passive structure can be achieved by reducing the amount of active control inputs and relying more on the system dynamics to create the motion. This increases the degree of underactuation of the system. In the robot fish world MIT's compliant swimming devices (Alvardo 2007) and the Beihang University Robo-ray II (Cai et al. 2010) are both highly underactuated swimming devices; both devices use system dynamics to determine kinematic motion rather than active controllable inputs. Other applications where underactuation has been employed include underactuated manipulator arms for space vessels (Mukherjee 1993).

Methods for the control of underactuated systems have mostly focused on the cart pendulum or the PenduBot. Figure 5.1 shows a diagram of the typical cart pendulum arrangement. The system consists of a rigid pendulum mounted on a free moving cart. The cart pendulum as shown has two degrees of freedom, the angle of the pendulum θ and the horizontal displacement of the cart x. Typically the model is simulated with a single controllable input in the form of horizontal input force acting on the cart. Figure 5.2 shows a diagram of the typical PenduBot arrangement. The system consists of two rigid bodies connected in series to a fixed point pivot. The PenduBot as shown has two degrees of freedom measurable as either the orientation of the two bodies or the

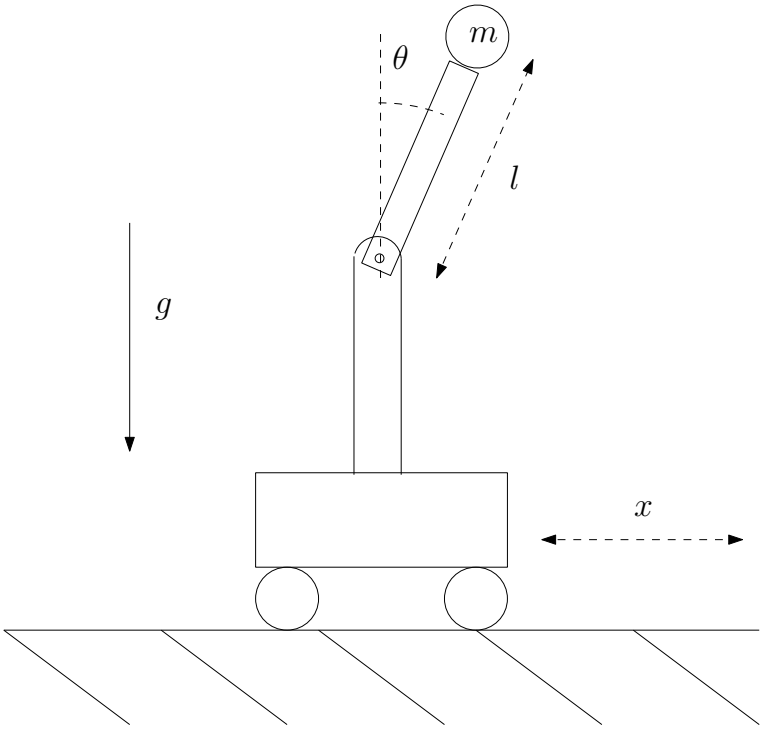


Figure 5.1: Inverted pendulum

angle of the two joints, θ_1 and θ_2. Typically the pendubot is controlled through a single torque input located at the pivot. Each of these devices can be described by a system of nonlinear ordinary differential equations similar in form to equation 4.14, with joint variables $\mathbf{q} = [\ \theta \quad x\ ]^T$ for the cart pendulum, $\mathbf{q} = [\ \theta_1 \quad \theta_2\ ]^T$ for the PenduBot.

A typical geometric control objective for the cart pendulum could be to '*swing up*' from an initial stable point to a dynamicaly unstable point. i.e. $[\ \pi \quad 0 \quad 0 \quad 0\ ]^T \rightarrow [\ 0 \quad 0 \quad 0 \quad 0\ ]^T$. Or for the PenduBot $[\ \pi \quad \pi \quad 0 \quad 0\ ]^T \rightarrow [\ 0 \quad 0 \quad 0 \quad 0\ ]^T$.

Spong (1996) presented a swing up control method for the Pendubot. Astrom and Furuta (2000) presented swing up control of the cart pendulum, similarly (Zhong and Rock 2001) presented a swing up method for the double cart pendulum, an expansion of the cart pendulum where two pendulums were placed on a single cart.

More of interest for this study is the control of a periodic motion over underactuated

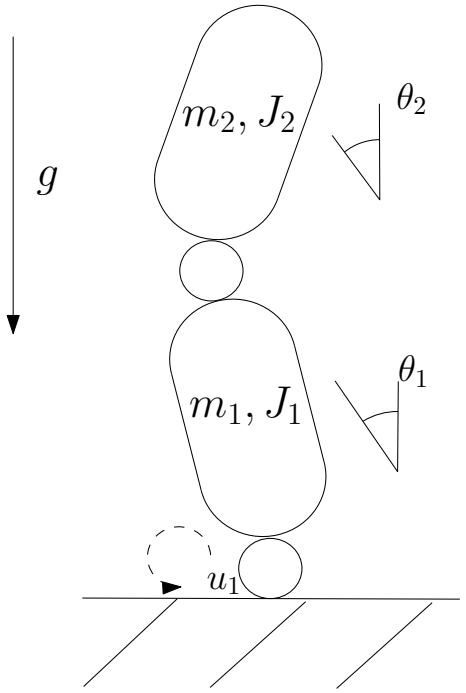


Figure 5.2: PenduBot

systems. Periodic motions can be described as an orbit in the state space around some point. Aracil et al. (2002),de Wit et al. (2002) and Shiriaev et al. (2005) all reported on orbital stabilization of cart pendulum.

The remainder of this chapter will be divided into two further sections. Section 5.2 discusses the theory of orbital control of a state space and how such a method could be used to generate a gait on an underactuated robotic fish. Whereas section 5.3 presents some concluding remarks.

5.2 State space orbit as a gait

BCF swimming locomotion like most biological locomotion relies on a periodic motion or gait. Any stable periodic motion can be plotted in state space as an enclosed path through the space. Figure 5.3 shows two such paths. Figure 5.3 (a) shows a symmetrical impulse orbit, the state travels at uniform velocity to a fixed point, then experiences

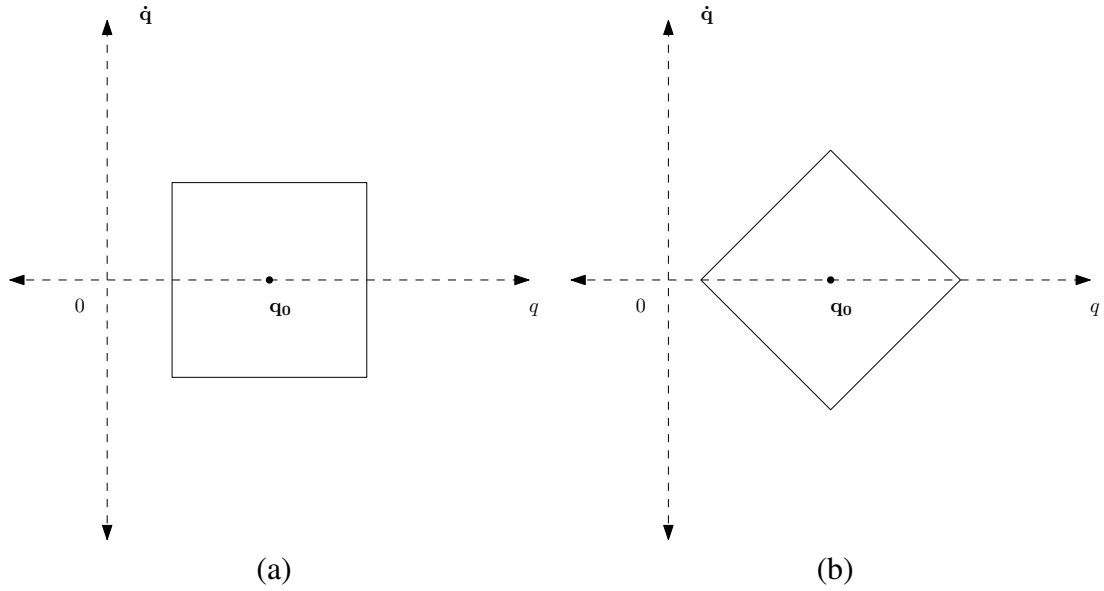


Figure 5.3: (a) Impulse orbit; (b) Bang bang orbit

instantaneous acceleration to a uniform velocity in the opposite direction. An impulse orbit could be analogous to a ping-pong ball being hit back and forth in the absence of air resistance and gravity. Figure 5.3 (b) shows a symmetrical bang bang orbit, the state accelerates uniformly in one direction to a fixed velocity then accelerates uniformly in the opposite direction. A bang bang orbit could be analogous to a binary oscillation of acceleration.

In addition to those shown in figure 5.3 there is an infinite number of possible shapes possible for a state space orbit. Each one corresponding to a different periodic motion. The goal is to find an orbit, or a group of orbits which will result in some degree of optimality in BCF swimming.

Many studies have confined the search space for optimum BCF gaits for a robotic tuna to those based around simple harmonic motions, i.e. $x(t) = a\sin(2\pi t/\lambda\alpha)$ where α is some phase angle, a is an amplitude and λ is a wave length (Barrett 1996; Watts 2009). Such a motion results in an elliptical orbit through the state space and as such this study will focus on elliptical orbits or orbits on the surface of a transformed hyper

sphere (hyper ellipse).

5.2.1 Orbital stabilisation control

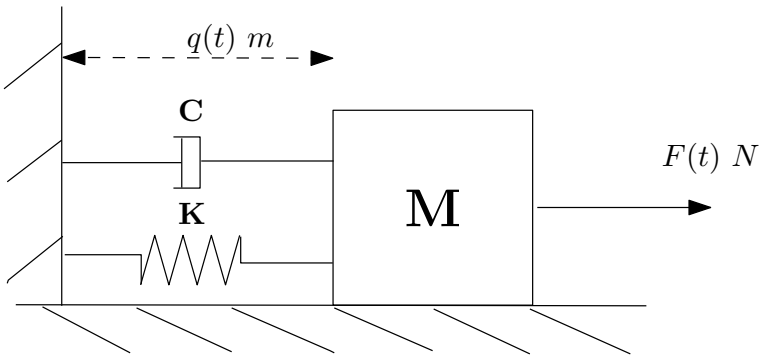


Figure 5.4: Mass spring damper system

Figure 5.4 shows a simple mass spring damper system, a mass is placed on a surface and is allowed to move along the horizontal axis. The mass is attached to a point on the horizontal axis via a spring providing linear force proportional to translation of the mass and a damper providing linear force proportional to translational velocity. The mass spring damper system can be described as a second order system with a single degree of freedom. Assuming the system is based on a frictionless surface and motion is orthogonal to gravity the system in figure 5.4 can be described by,

$$\begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\mathbf{M^{-1}K} & -\mathbf{M^{-1}C} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix} \mathbf{F} \tag{5.1}$$

where **K** is the linear spring constant, **C** is the dampening coefficient and **M** the mass.

A harmonic periodic motion of the mass can be described as,

$$\mathbf{q}(t) = \mathbf{q_0} \pm \sqrt{2}\sin(t+\alpha) \tag{5.2}$$

where $\mathbf{q_0}$ is an arbitrary point within the system and α is an arbitrary phase value. The

velocity time function must then be,

$$\dot{\mathbf{q}}(t) = \pm\sqrt{2}\cos(t) \tag{5.3}$$

Equations 5.2 and 5.3 can be plotted as an orbit around the point $\mathbf{q_0}$ as shown in figure 5.5. A classical geometric control approach to achieving this motion would be to prescribing the position at time t to equal the output of equation 5.2. However if it is sufficient that system merely follows the path from figure 5.5 the values α is completely arbitrary. For a given state error at one α value will be different from error at another α value. An incorrect choice of α can result in an error even when the state lies on the desired path, if that α corresponds to a different orbit.

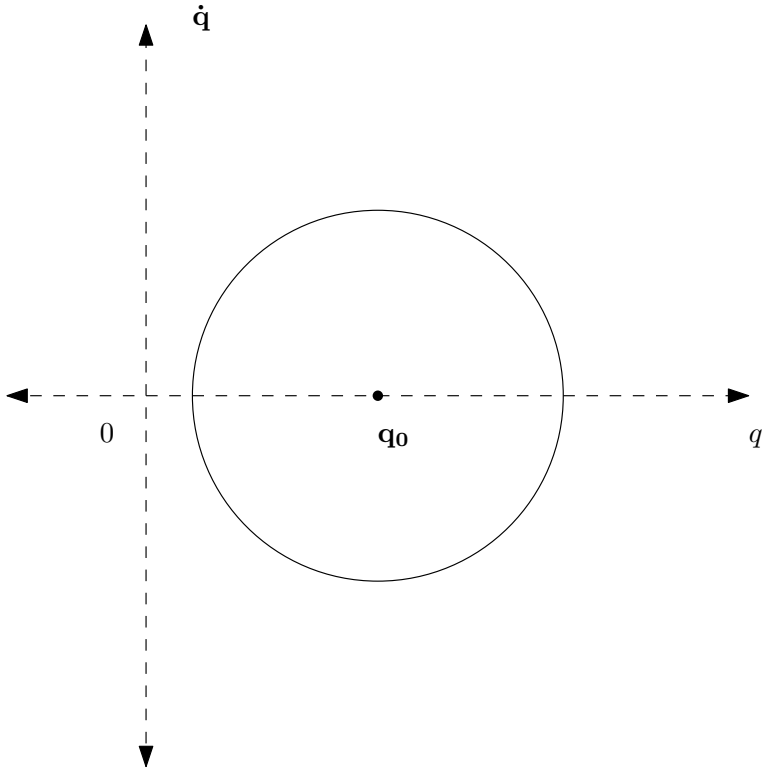


Figure 5.5: State space orbit

Figure 5.5 shows the orbital path of the motion described in equation 5.2. The motion follows a uniform elliptical path around $\mathbf{q_0}$.

The set of all possible orbits that follow the same path as equation 5.2 can be expressed as,

$$\begin{bmatrix} (\mathbf{q}-\mathbf{q_0}) & \dot{\mathbf{q}} \end{bmatrix} \begin{bmatrix} (\mathbf{q}-\mathbf{q_0}) \\ \dot{\mathbf{q}} \end{bmatrix} = \mathbf{x^T I x} = 2 \tag{5.4}$$

Equation 5.4 can be rewritten,

$$\frac{1}{2}\mathbf{p}^2 + \frac{1}{2}\dot{\mathbf{p}}^2 = 1 \tag{5.5}$$

where $\mathbf{p} = \mathbf{q} - \mathbf{q_0}$. Equation 5.5 is equivalent to the sum of stored spring and inertia energy for a system with spring constant $1Nmrad^{-1}$ and mass of $1KG$. Therefore equation 5.4 can be described as a unit spring mass state energy equation.

For an n dimensional system a unit spring mass state energy equation of the form $\mathbf{x^T I x} = 2$ can be satisfied by the set of all possible orbits on the surface of a n dimensional hypersphere. This set will include simple harmonic orbits describable as uniform elliptical motion through the state space as shown in figure 5.6 (a), and more complex orbits with multiple harmonics as shown in figure 5.6 (b) where the orbital path travels around the surface multiple times before repeating itself.

Furthermore it can be deduced that any possible stable elliptical state space orbit of system can be represented as a unit state energy equation with some state transformation.

$$\mathbf{x^T Q x} = 1 \tag{5.6}$$

where $\mathbf{Q} > 0$ is some full rank positive definite matrix.

Proof: If $\mathbf{T}$ is any real invertible matrix, then an orbit satisfying equation 5.4 in the space $\mathbf{z} = \mathbf{Tx}$ must necessarily satisfy $\mathbf{x^T T^T T x} = \mathbf{x^T Q x} = r$ in the space $\mathbf{x}$. It is also clear that $\mathbf{T^T T} = \mathbf{Q}$ is positive definite.

The control of state energy is a central concept in the control of underactuated systems.

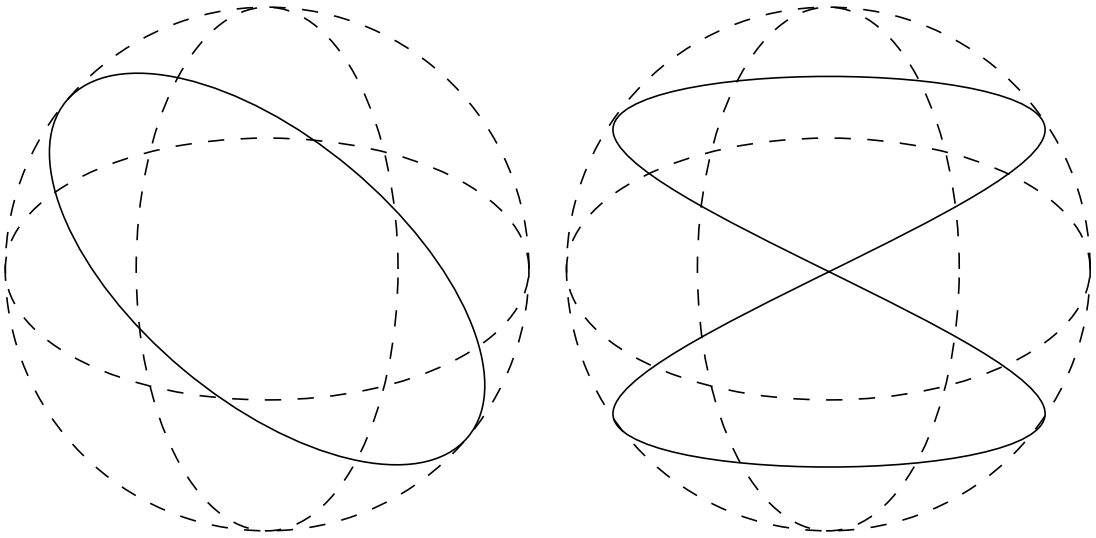


Figure 5.6: (a) Simple harmonic orbit on hypersphere; (b) Multiple harmonic orbit on hypersphere

Since the state energy level will guarantee that the state is on the orbital path the regulation of multi dimensional motions can be achieved through the control of a single output (Ortega et al. 2001).

Note there is no actual need for the physical energy of the system to correspond to the energy function used, for example the physical energy of the system in figure 5.4 is given by,

$$E = \frac{1}{2}\begin{bmatrix} \mathbf{q} & \dot{\mathbf{q}} \end{bmatrix}\begin{bmatrix} \mathbf{K} & 0 \\ 0 & \mathbf{M} \end{bmatrix}\begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} \tag{5.7}$$

however any positive definite $\mathbf{Q}$ could be described as an energy equation.

The difference between the physical and control energy equation may result in negative work for example. For the orbit described by equations 5.2 and 5.3 applied to the system described in equation 5.1, the state energy will be determined by the time dependent function,

$$\dot{E}_r = (\mathbf{K} - \mathbf{M})\sin(2t) \tag{5.8}$$

$\dot{E}_r < 0$ implies a reduction in total state energy is required. This can be achieved either through dissipation of energy or negative work form the controller. For the system described in equation 5.1 dissipation of energy can be calculated as,

$$\dot{E}_d = -D\dot{\mathbf{q}}^2 \tag{5.9}$$

therefore the required controller power input will be,

$$P_{in} = \dot{E}_r - \dot{E}_d = (\mathbf{K} - \mathbf{M})\sin(2t) + 2\mathbf{D}\cos^2(t) \tag{5.10}$$

In this study the search space for BCF swimming gaits will be restricted to gaits where the energy equation of the orbit is a close match to the physical energy coefficients. It is thought that this will improve efficiency through avoiding negative work, i.e. work that actively removes energy from the state. However it can be surmised that if the energy dissipation is at all times greater than any negative gradient of the systems physical energy during the orbit, then no negative work will occur. As the robotic fish is expected to operate in a fluid environment, there will be significant dissipation of inertia energy into the surrounding fluid. Therefore it is likely that a small underestimation of the inertia will not result in negative work. As such for this study fluid induced added inertia terms can be safely omitted from the inertia term in the control energy equation.

5.2.2 Gait and Orbit

Barrett (1996) experimentally derived an optimal gait for a tuna geometry. The gait was based on the restricted set of kinematics of the from,

$$y(x,t) = [c_1 x + c_2 x^2]\sin(\omega t + \frac{2\pi x}{\lambda}) \tag{5.11}$$

where x is longitudinal displacement along the tail, $[c_1 x + c_2 x^2]$ is the amplitude

envelope, ω is the body wave frequency, λ is the body wave length and t is time in seconds. The optimal values determined in Barrett (1996) are given in table 5.1.

Applying this kinematic to the model described in chapter 4, the lateral translation for the body centres of mass are given by the time lines shown in figure 5.7 (a). Figure 5.7 (b) shows a plot of the position of the centres of mass for the three bodies at sample points during the motion, the motion envelope can be seen to increase towards the anterior of the robotic fish. Equation 4.3 gave the forward kinematic between body bearing space and lateral displacement space as $\mathbf{y} = \mathbf{M}\sin(\mathbf{q})$. However $\det(\mathbf{M}) = 0$ implies that a global inverse kinematic does not exist. A least square error best fit for $\mathbf{q}$ is shown in figure 5.8. With the corresponding displacement time line shown in figure 5.9 (a). Figure 5.9 (b) shows a plot of body position at sample points during the motion for the least square kinematic. It can be noted that the motion envelope is marginally narrower than the original towards the end of the body. This is due to the physical constraints of the four body kinematic chain, by adding more links to the chain the kinematic could be more closely approximated.

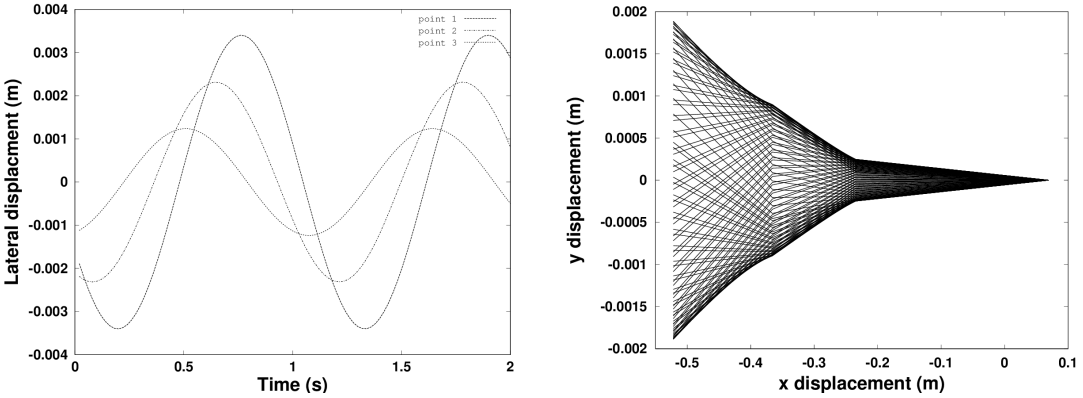
Table 5.1: Optimal tuna kinematic parameters (Barrett 1996)

c_1	0.00372
c_2	0.00483
ω	5.5401
λ	1.27

For the time series shown in figure 5.8 the following state energy equation was solved numerically for $\mathbf{Q}$,

$$\begin{bmatrix} \mathbf{q^T} & \dot{\mathbf{q}}^\mathbf{T} \end{bmatrix} \mathbf{Q} \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} = 1 \tag{5.12}$$

The resulting Q matrix found was,



(a) (b)

Figure 5.7: Optimal kinematic from Barrett (1996) (a) plot of lateral displacment of points against time, (b) plot of x and y position of points at sample times during the motion

$$\mathbf{Q} = \begin{bmatrix} 3.97 & 4.31 & 22.93 & 75.80 & 4.25 & 22.09 & 75.05 & -24.48 \\ 4.31 & 7.27 & 33.32 & 109.92 & 6.09 & 31.95 & 108.33 & -37.35 \\ 22.93 & 33.32 & 178.06 & 584.60 & 32.60 & 170.16 & 577.44 & -193.73 \\ 75.80 & 109.92 & 584.60 & 1934.80 & 109.02 & 564.40 & 1918.68 & -609.15 \\ 4.25 & 6.09 & 32.60 & 109.02 & 7.52 & 32.30 & 110.92 & -24.14 \\ 22.09 & 31.95 & 170.16 & 564.40 & 32.30 & 166.34 & 563.50 & -164.76 \\ 75.05 & 108.33 & 577.44 & 1918.68 & 110.92 & 563.50 & 1924.56 & -530.74 \\ -24.48 & -37.35 & -193.73 & -609.15 & -24.14 & -164.76 & -530.74 & 465.66 \end{bmatrix} \tag{5.13}$$

with a mean square error over the time series of $2.1867 \times 10^{-}18$. This demonstrates that the set of state space orbits described by $\mathbf{x^T Q x}$ for the system described in chapter 4 contains almost the exact motion described in figure 5.8.

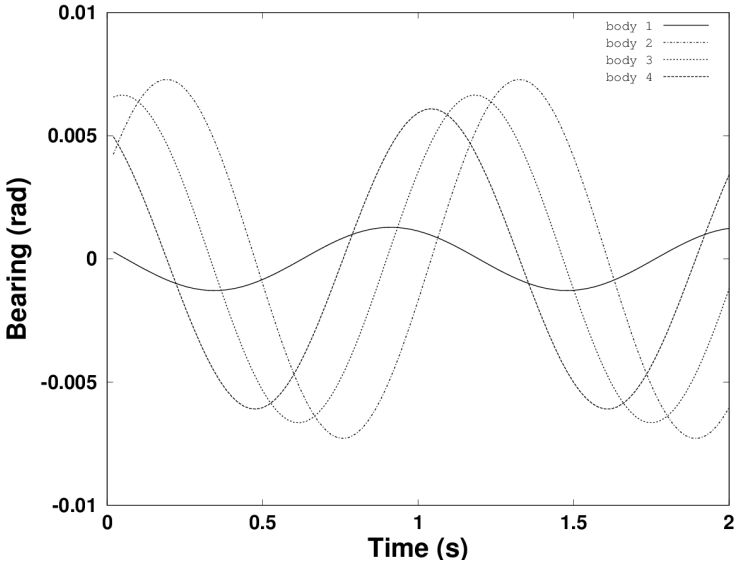


Figure 5.8: Least square best fit $\mathbf{q}$

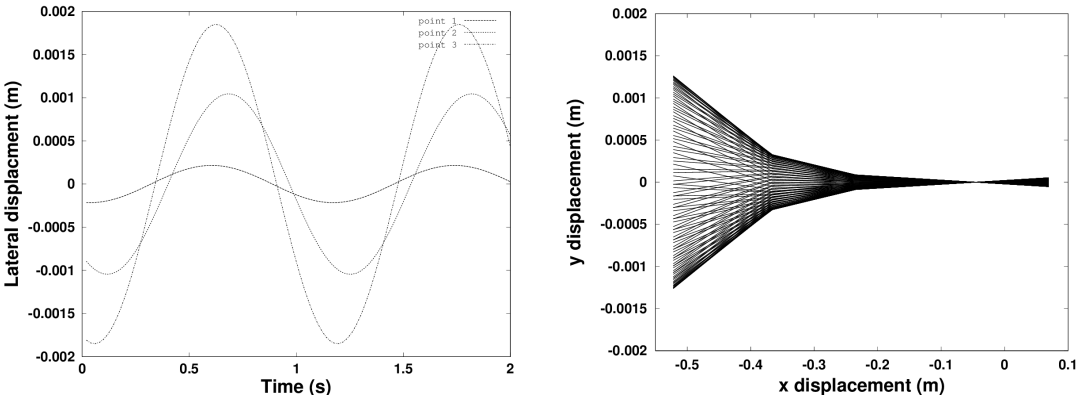
5.3 Concluding Remarks

In this chapter the concept of state energy orbits has been discussed for the control of motion on underactuated systems.

By controlling state energy rather than absolute position, high dimensional motion can be controlled with a single measurable output.

It was argued that a constant physical state energy orbit results avoid negative work, and as such could be considered most energy efficient. It was also argued that if sufficient inertia dissipation is present in the system underestimation of inertia energy will not result in negative work, as such fluid induced added inertia can be safely omitted from control state energy equations.

The next chapter will introduce a model based deadbeat control method for the control of state energy in order to apply the methods described here to the system described in chapter 4.



(a) (b)

Figure 5.9: Least square best fit kinematic (a) plot of lateral translation against time, (b) plot of x and y position of points at sample times during the motion

Chapter 6

Deadbeat state energy controller

This chapter will discuss a deadbeat strategy for the control of energy on a nonlinear manipulator system.

6.1 Introduction

In chapter 5 the use of state space orbits to describe periodic motion was discussed. Furthermore it was suggested that the periodic motion for a BCF swimming gait could be described as a state space orbit. The advantage being that high dimensional motion could be described by a single measurable parameter. In this chapter a deadbeat orbital stabilisation controller will be described. This chapter will discuss a proposed deadbeat state energy controller in order to control and stabilise the state space orbit. Deadbeat controllers for gait generation are common amongst terrestrial walking and hopping robots on unknown terrains. Where the necessity for balance requires a tightly controlled limit cycle with significant exogenous disturbances.

Deadbeat control is a discrete control method that effectively requires the reduction of the closed loop discrete transfer function for the deadbeat period between the reference and the control output to unity. Therefore effective deadbeat control requires accurate knowledge of system dynamics.

For terrestrial walking robots deadbeat controllers are based around known models for the common *spring loaded inverted pendulum* (SLIP) (Dai and Tedrake 2012).

Deadbeat controllers for underactuated single legged hopping robots have been described in Saranli et al. (1998),Ankarali et al. (2009) and Uyanic et al. (2011).

Asano and Xiao (2012) presented a robust deadbeat control for an underactuated spoked walker based on an extended SLIP model. Psudo-deadbeat control for a four legged robot again based on a known SLIP model was presented in Armada et al. (1993).

Deadbeat controllers are also common in the field of power electronics, used primarily in generation and transformation. Deadbeat controllers are often used within *pulse width modulator* (PMW) control for inverters Kawamura et al. (1988). Or to achieve accurate control over torque, flux or current in permanent magnet electric motors (Lorenz and Valenzuela 2012; Li et al. 2012).

As mentioned before deadbeat control is a purely discrete control method, as such there is no analogue equivalent. The time period for a deadbeat controller determines the accuracy, if the time period is to large then between each iteration of the control algorithm a significant amount of error may be allowed to accumulate this can have a significant negative effect on the performance of signal tracking, for instance a steady state disturbance would translate to a larger steady state error. However as the time period approaches zero the deadbeat controller has to make more 'aggressive' control moves to compensate the existing error within the period, meaning that a small error in modelling could result in significant overshoot, which in turn can lead to hunting and a complete destabilisation of the system. This means the shorter the deadbeat period the more fragile the closed loop system will be.

In this study a deadbeat controller is derived from the discrete system model for state energy. The approach is chosen as it encompasses the nonlinearity of the system without the addition of excessive complexity. This provides a reliable means of regulating state energy to test the hypothesis of energy based gait control for a robotic fish.

The remainder of this chapter will be divided into four further sections, section 6.2 will briefly introduce deadbeat control and discuss some of its advantages. In section 6.3 a deadbeat controller for state energy will be presented. Section 6.4 will present results from a simulation study applying the deadbeat controller to the model form chapter 4 .Finally section 6.5 will present some and concluding remarks.

6.2 Deadbeat control

A control response is considered deadbeat if and only if, for the defined deadbeat interval any state error present at the beginning of the interval is guaranteed to reduce to zero steady state error at the end of the interval, irrespective of control response during the interval. Therefore a deadbeat controller is automatically robustly stable. However satisfaction of the deadbeat criteria can be considered fragile as it is highly dependent on implementation accuracy. Implementation accuracy can be affected by uncontrollable factors such as actuator noise, resolution of available digital analogue conversion and accuracy of model.

This is mathematically equivalent to ensuring that the closed loop discrete transfer function for the deadbeat period between n reference signals and n controllable outputs is exactly equal to an $n \times n$ identity matrix regardless of the system coupling.

Put more simply in order to be considered deadbeat, a system with output $\mathbf{y} = \mathbf{Cz}$ given an input $\mathbf{r_n}$ must have $\mathbf{Cz_{n+1}} = \mathbf{r_n}$, therefore the discrete closed loop transfer function for the deadbeat period between $\mathbf{y_{n+1}}/\mathbf{r_n} \equiv \mathbf{I}$.

For an uncertain discrete state space system with a linear objective function of the form,

$$\begin{aligned} \mathbf{z_{n+1}} &= \mathbf{Az_n} + \mathbf{B_1 w_n} + \mathbf{B_2 u_n} \\ \mathbf{e_n} &= \mathbf{Cz_n} - \mathbf{r_n} \end{aligned} \tag{6.1}$$

where $\mathbf{w_n}$ and $\mathbf{u_n}$ are uncertain and controllable inputs respectively, and $\mathbf{e_n}$ is the error at time n. A deadbeat response can be achieved by selecting $\mathbf{u_n}$ that satisfies,

$$\mathbf{C}(\mathbf{A}\mathbf{z_n} + \mathbf{B_2}\mathbf{u_n}) = \mathbf{r_{n+1}} \tag{6.2}$$

At each time step the existing error will then be exactly corrected. The error at any time step will be constrained by the effect of exogenous disturbances during the previous deadbeat period,

$$\mathbf{e_{n+1}} = \mathbf{C}\mathbf{B_1}\mathbf{w_n} \tag{6.3}$$

For a continuous time system, the maximum magnitude of the error is then determined by size of the deadbeat period; the shorter the deadbeat period the smaller the error.

This form of deadbeat control could also be described as an analytically optimised model predictive controller with a prediction and control horizon of 1 (Rawlings and Mayne 2009). This strong parallel with model predictive control means that deadbeat control can be achieved for highly nonlinear systems provided that an analytical solution to the discrete time error function exits.

6.3 Deadbeat control of state energy

In the previous section an example of a simple deadbeat controller for a discrete time state space system with a linear objective function was given. This section will discuss a deadbeat controller for a system with a state energy objective function i.e. $\mathbf{e} = \mathbf{r} - \mathbf{x^T}\mathbf{Q}\mathbf{x}$.

For an uncertain discrete system with dynamics the form described in equation 6.1, the system dynamic equation can be expressed as the sum of three responses. The natural response given by $\mathbf{A}\mathbf{x_n}$, the response of the system to disturbance given by $\mathbf{B_1}\mathbf{w_n}$ and the controllable input response given by $\mathbf{B_2}\mathbf{u_n}$. Substituting the dynamic equation into

the objective function yields,

$$\mathbf{e_n} = \mathbf{r_n} - (\mathbf{Ax_n} + \mathbf{B_1 w_n} + \mathbf{B_2 u_n})^{\mathbf{T}} \mathbf{Q} (\mathbf{Ax_n} + \mathbf{B_1 w_n} + \mathbf{B_2 u_n}) \tag{6.4}$$

By separating the terms containing uncontrollable disturbance elements, the discrete state error function given in equation 6.4 can be divided into a controllable error function,

$$\mathbf{e_{cn}} = \mathbf{r_n} - (\mathbf{Ax_n} + \mathbf{B_2 u_n})^{\mathbf{T}} \mathbf{Q} (\mathbf{Az_n} + \mathbf{B_2 u_n}) \tag{6.5}$$

and an uncontrollable disturbance error function of which the system controller can have no knowledge,

$$\mathbf{e_{dn}} = -(\mathbf{2Ax_n} + \mathbf{B_1 w_n} + \mathbf{2B_2 u_n})^{\mathbf{T}} \mathbf{Q} (\mathbf{B_1 w_n}) \tag{6.6}$$

therefore if u_n is chosen such that equation 6.5 is equal to 0 a deadbeat response will be achieved.

6.3.1 Application to robotic fish

Deadbeat control of the model developed in chapter 4 requires the discretization of the dynamic equation 4.14 over some deadbeat interval.

For any given state $\mathbf{x} = [\ \dot{\mathbf{q}}^{\mathbf{T}} \quad \mathbf{q}^{\mathbf{T}}\ ]^T$, equation 4.14 can be locally linearised as,

$$\begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{H(x)^{-1}K} & \mathbf{H(x)^{-1}C(x)} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{H(x)^{-1}} \end{bmatrix} \mathbf{w} + \begin{bmatrix} \mathbf{0} \\ \mathbf{H(x)^{-1}B} \end{bmatrix} \mathbf{u} \tag{6.7}$$

where $\mathbf{w}$ is equivalent to the sum of the disturbance vector $\mathbf{D}$ from equation 4.14 and inaccuracies caused by model simplification. Equation 6.7 can be written as the

continuous linear state space equation,

$$\dot{\mathbf{x}} = \tilde{\mathbf{A}}\mathbf{x} + \tilde{\mathbf{B}}_\mathbf{1}\mathbf{w} + \tilde{\mathbf{B}}_\mathbf{2}\mathbf{u} \tag{6.8}$$

which can in turn be discretized over a period Δt as,

$$\mathbf{x}_{t+\Delta t} = \mathbf{e}^{\tilde{\mathbf{A}}\Delta t}\mathbf{x}_t + \tilde{\mathbf{A}}^{-\mathbf{1}}(\mathbf{I} - \mathbf{e}^{\tilde{\mathbf{A}}\Delta t})(\tilde{\mathbf{B}}_\mathbf{1}\mathbf{w}_t + \tilde{\mathbf{B}}_\mathbf{2}\mathbf{u}_t) \tag{6.9}$$

the state energy is then given by,

$$E = \mathbf{x}^\mathbf{T}\mathbf{Q}\mathbf{x} \tag{6.10}$$

Substituting Equation 6.9, gives a discrete equation for energy, in terms of current state, and input.

$$\begin{aligned} E_{t+\Delta t} = {} & \mathbf{n}^\mathbf{T}\mathbf{Q}\mathbf{n} + \mathbf{n}^\mathbf{T}\mathbf{Q}\mathbf{T}_\mathbf{2}\mathbf{u} + \mathbf{u}^\mathbf{T}\mathbf{T}_\mathbf{2}^\mathbf{T}\mathbf{Q}\mathbf{n} + \mathbf{u}^\mathbf{T}\mathbf{T}_\mathbf{2}^\mathbf{T}\mathbf{Q}\mathbf{T}_\mathbf{2}\mathbf{u} \\ & (\mathbf{u}^\mathbf{T}\mathbf{T}_\mathbf{2}^\mathbf{T} + \mathbf{n}^\mathbf{T} + \mathbf{w}^\mathbf{T}\mathbf{T}_\mathbf{1}^\mathbf{T})\mathbf{Q}\mathbf{T}_\mathbf{1}\mathbf{w} + \mathbf{w}^\mathbf{T}\mathbf{T}_\mathbf{1}^\mathbf{T}\mathbf{Q}(\mathbf{T}_\mathbf{2}\mathbf{u} + \mathbf{n} + \mathbf{T}_\mathbf{1}\mathbf{w}) \end{aligned} \tag{6.11}$$

Where $\mathbf{n}$ is the systems natural response given by,

$$\mathbf{n} = \mathbf{e}^{\tilde{\mathbf{A}}\Delta t}\mathbf{x_t}$$

and $\mathbf{T_1}$ and $\mathbf{T_2}$ are the uncertain and controllable input responses respectively given by,

$$\mathbf{T_1} = \tilde{\mathbf{A}}^{-\mathbf{1}}(\mathbf{I} - \mathbf{e}^{\tilde{\mathbf{A}}\Delta t})\tilde{\mathbf{B}}_\mathbf{1}$$

$$\mathbf{T_2} = \tilde{\mathbf{A}}^{-\mathbf{1}}(\mathbf{I} - \mathbf{e}^{\tilde{\mathbf{A}}\Delta t})\tilde{\mathbf{B}}_\mathbf{2}$$

The controllable state energy error function will be,

$$\mathbf{e_{c(t+\Delta t)}} = \mathbf{r} - \mathbf{n^T Q n} + \mathbf{n^T Q T_2 u_n} + \mathbf{u_n^T T_2^T Q n} + \mathbf{u^T T_2^T Q T_2 u_n} \tag{6.12}$$

Substituting $\mathbf{e}_{c(t+\Delta t)} = 0$ for a single input system equation 6.12 forms a quadratic equation in terms of u_n, the solution to which will be given as,

$$\mathbf{u_n} = \frac{\mathbf{n^T Q T_2} \pm \sqrt{\mathbf{n^T Q T_2^2} + \mathbf{4 T_2^T Q T_2 (n^T Q n - r)}}}{\mathbf{2 T_2^T Q T_2}} \tag{6.13}$$

This gives two possible solutions to the deadbeat objective provided that,

$$\mathbf{n^T Q T_2^2} + \mathbf{4 T_2^T Q T_2 (n^T Q n - r)} \geq 0 \tag{6.14}$$

Selection of the solution of least magnitude will result in the minimum cost deadbeat control response for the period Δt.

6.4 Results

The spring values were determined experimentally through trial and improvement, to select values that result in an effective forward swimming kinematic. Eventually spring constants for the three joints of $k_i = [\ 15 \quad 20 \quad 12.5\ ] Nm\ rad^{-1}$ were selected.

Giving the spring matrix,

$$K = \begin{bmatrix} 15 & -15 & 0 & 0 \\ -15 & 35 & -20 & 0 \\ 0 & -20 & 32.5 & 12.5 \\ 0 & 0 & -12.5 & 12.5 \end{bmatrix} \tag{6.15}$$

The control state energy equation was selected to reflect the physical energy of the system. The state energy matrix $\mathbf{Q}$ was given by,

$$\mathbf{Q} = \frac{1}{2}\begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0} & \mathbf{J} + \mathbf{M}'\mathbf{diag}(\mathbf{m})\mathbf{M} \end{bmatrix} \tag{6.16}$$

where $\mathbf{m}$ is the vector of body masses $\mathbf{J}$ is the diagonal matrix of rotational moments of inertia of the bodies and $\mathbf{M}$ is the matrix of moments given in chapter 4.

Because $\mathbf{Q}$ reflects the mechanical energy of the system the closed loop system response should follow on the system transients. The system is nonlinear, however for small perturbations in the state space, can be approximated locally by an LTI system. The eigenvalues of the central LTI approximation of system dynamic matrix are given by,

$$\begin{gathered} -23.4061 + 255.4551i \\ -23.4061 - 255.4551i \\ -1.2300 + 69.6848i \\ -1.2300 - 69.6848i \\ -0.0140 + 17.0379i \\ -0.0140 - 17.0379i \\ -0.0027 + 1.0459i \\ -0.0027 - 1.0459i \end{gathered} \tag{6.17}$$

Therefore the corresponding undamped system transient for the nonlinear system are

likely to be approximated by,

$$\begin{gathered} 40Hz \\ 11Hz \\ 2.7Hz \\ 0.17Hz \end{gathered} \tag{6.18}$$

The deadbeat controller described in the previous section was applied to the simulation described in chapter 4.

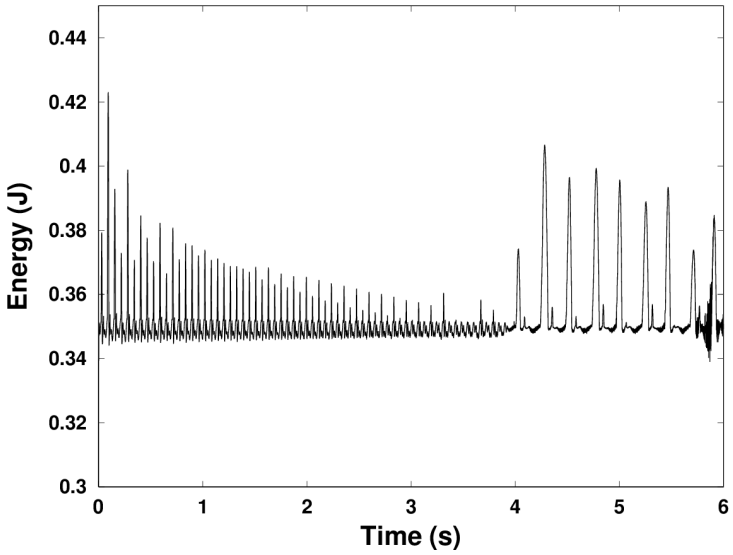


Figure 6.1: Plot of system state energy against time

An energy reference value of $0.35J$ was selected semi arbitrarily as it was large enough to produce distinct tail beat motions whilst remaining within the allowable deformations of the model. The simulation was run for 6 seconds with fluid and bodies starting from stationary. The deadbeat control interval was set to $2 \times 10^{-4}s$ with no modification of the control signal during the interval.

Figure 6.1 shows a plot of total system energy against time. The degradation of a high frequency transient can be seen over the initial 4 seconds after which a low frequency transient becomes dominant.The controller achieved the state energy control objective

with *root mean square error* (RMSE) of $\approx 0.0095J$.

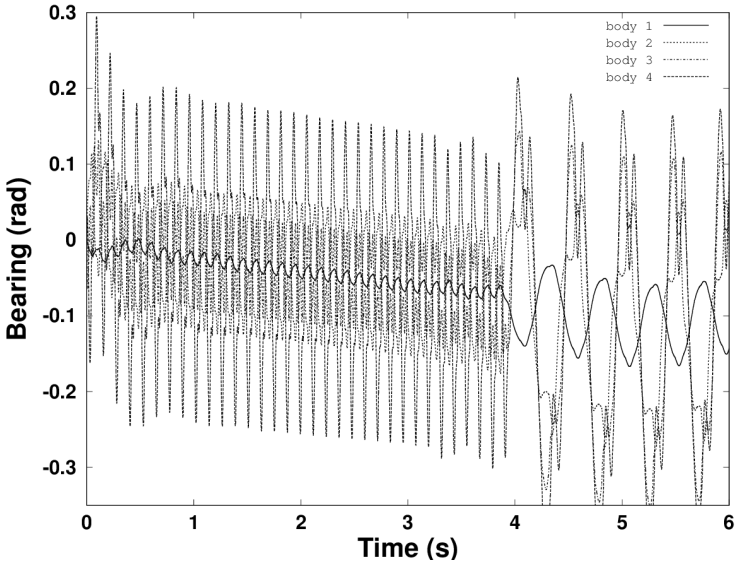


Figure 6.2: Plot of body bearing against time

Figures 6.2 shows plots of body bearing against time. It can clearly be seen that although the controller successfully restricted the model to the set of state space orbits described by $\mathbf{x^T Q x} \approx 0.35$ the resultant orbital path was not fixed. Initially the system established a stable orbit with frequency $\approx 10Hz$ however after $4seconds$ the orbital path switched to an alternative path over the surface $\mathbf{x^T Q x} \approx 0.35$ with frequency $\approx 2.5Hz$. This suggests that the system was initially dominated by the $11Hz$ undamped transient, then due to some change in condition or degradation the motion switched to be dominated by the $2.7Hz$ undamped transient. The general negative trend visible in figure 6.2 indicates a gradual yaw motion suggesting the presence of yaw instability. During the first 4 seconds the robotic fish experienced a mean yaw rate of $0.021rad\ s^{-1}$, which increased to $0.026rad\ s^{-1}$ during the motion between 4 and 6 seconds.

An artificial limit of $100Nm$ was placed on the control inputs to emulate controller saturation and prevent unrealistic system response. However after the initial rise time the controller did not hit this limit. Figure 6.3 shows a plot of controller input

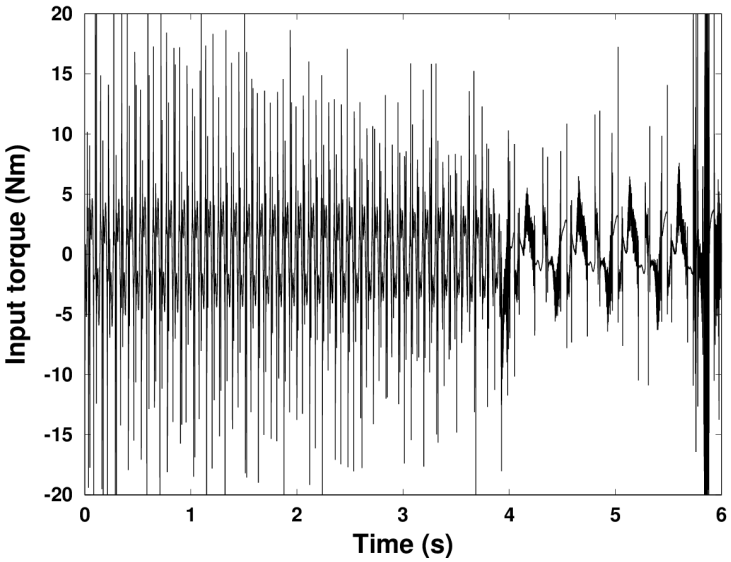


Figure 6.3: Plot of controller input against time

against time. The periodic spikes in control input correspond to changes in direction. The sudden large input at ≈ 6 seconds corresponds to the point at which the mesh deformations within the CFD solver reached the point where pressure could no longer be resolved within the allocated iterations. The total input work was calculated as the angular velocity of the actuated joint multiplied by the control input torque,

$$Work = \mathbf{B^T}\dot{\mathbf{q}}\mathbf{u} \tag{6.19}$$

Calculated actuator work values shown in figure 6.4 demonstrated that after initial establishment of motion, peak power requirement to maintain motion was $\approx 31W$. Mean absolute power requirement during the first 4 seconds t was calculated as $\approx 9.88W$ however during 4-6 seconds the requirement dropped to $\approx 3W$. Although as can be seen from figure 6.4 negative work was required during the periodic motion. The mean negative work during the two motions i.e. the average amount of work done to reduce total system energy was estimated as $\approx 0.98445W$ and $0.43W$ respectively

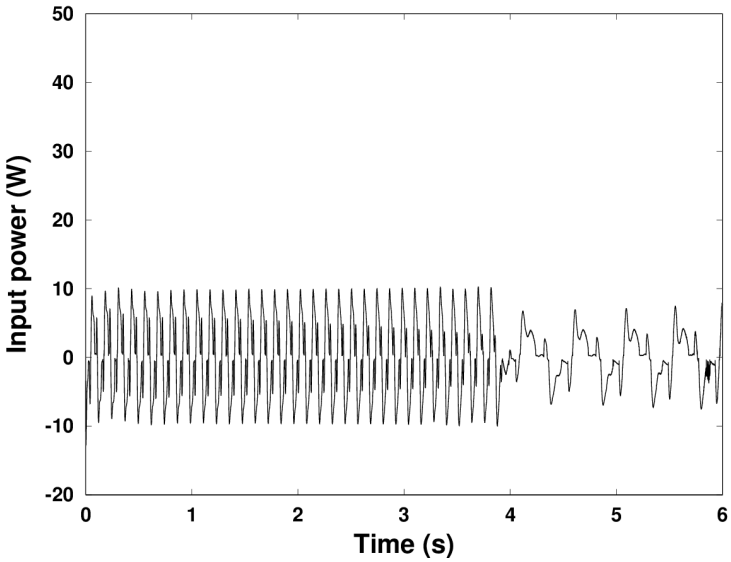
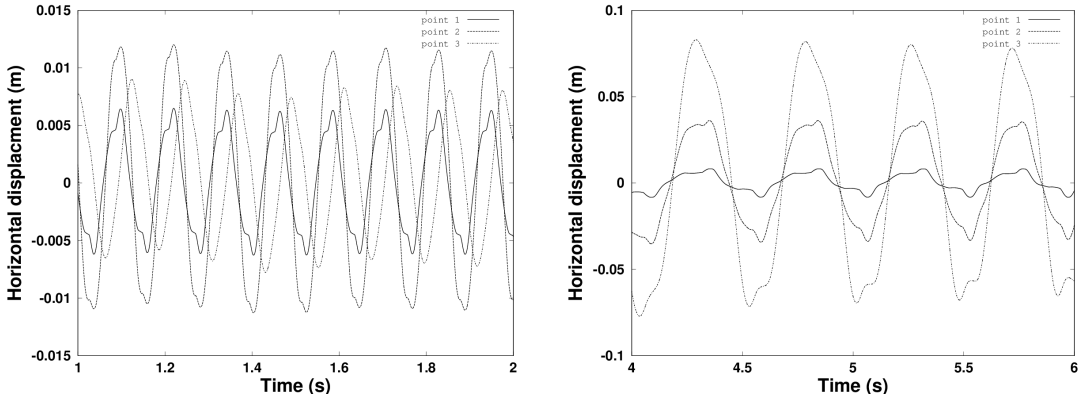


Figure 6.4: Plot of controller work against time

which was approximately 10% and 14% respectively of the total absolute work.

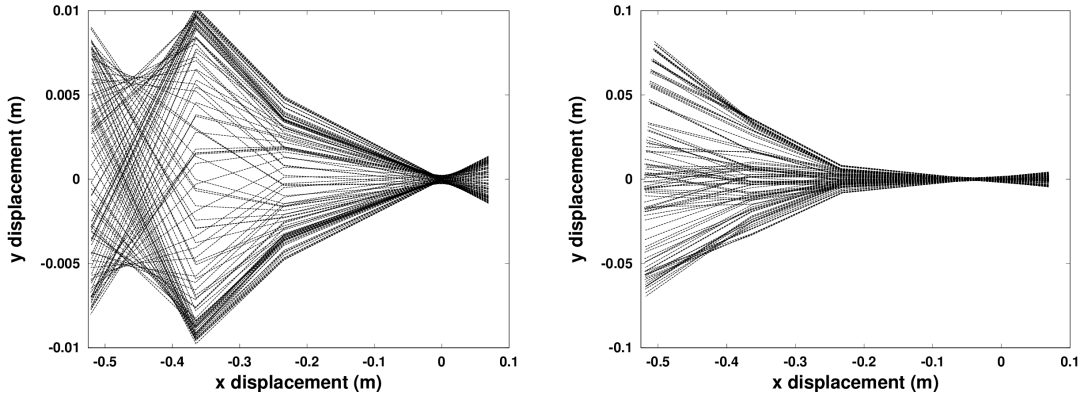


(a) (b)

Figure 6.5: Plot of body horizontal displacement against time (a) initial motion, (b) secondary motion

Figure 6.5 shows a plot of lateral displacement of the three points used in figures 5.7 and 5.9. It can be seen that the lateral motions become smoother towards the aft of the model. Suggesting that the caudal fin is experiencing smooth lateral translation. Also it can be observed that the lateral translations of the tail are significantly larger for the

lower frequency motion than the high frequency motion. As lateral translation of the caudal fin is essential for lift based BCF swimming this suggests that the low frequency motion is significantly better suited to generating lift based thrust.



(a) (b)

Figure 6.6: Plot of system kinematic (a) initial motion, (b) secondary motion

Figure 6.6 shows the position of the centres of mass of each of the bodies at sample points during the motion. It can clearly be seen that the motion envelope for the motion during the initial motion differed significantly from the motion envelop of the motion after 4 seconds. Whilst the motion envelope for the initial motion shows the lateral motion was greatest at the centre of the fish, the motion envelope of the motion after 4 seconds expanded towards the anterior of the body. This was more in keeping with optimal swimming kinematics described in Barrett (1996).

Figure 6.7 shows a plot of forward velocity of the simulated robotic fish against time, It can be seen that both kinematics resulted in forward thrust, however the latter kinematic resulted in significantly higher rate of acceleration $\approx 0.05ms^{-2}$ compared to $\approx 0.025ms^{-2}$.

Table 6.1 shows images of the fluid surrounding the robotic fish. Red indicates high velocity, and blue indicates low velocity. The high velocity areas that develop alongside the tail are indicative of vortices. The vorticies can be seen shedding and being carried

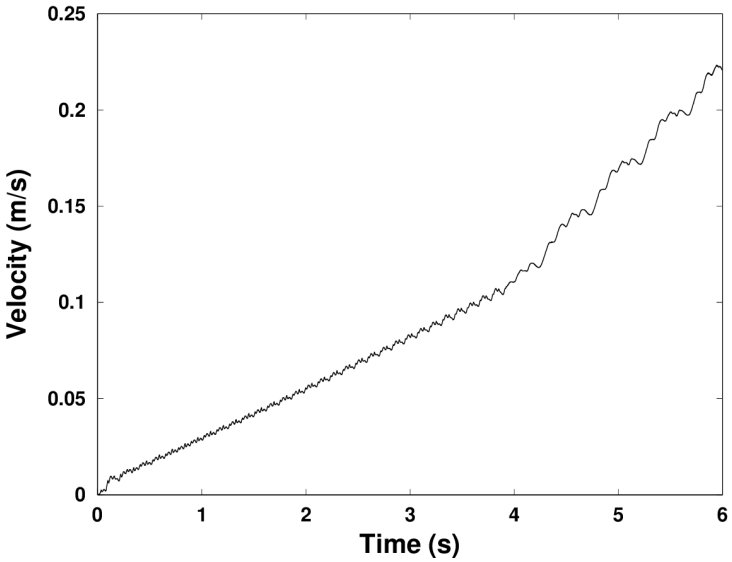


Figure 6.7: Plot of forward velocity against time

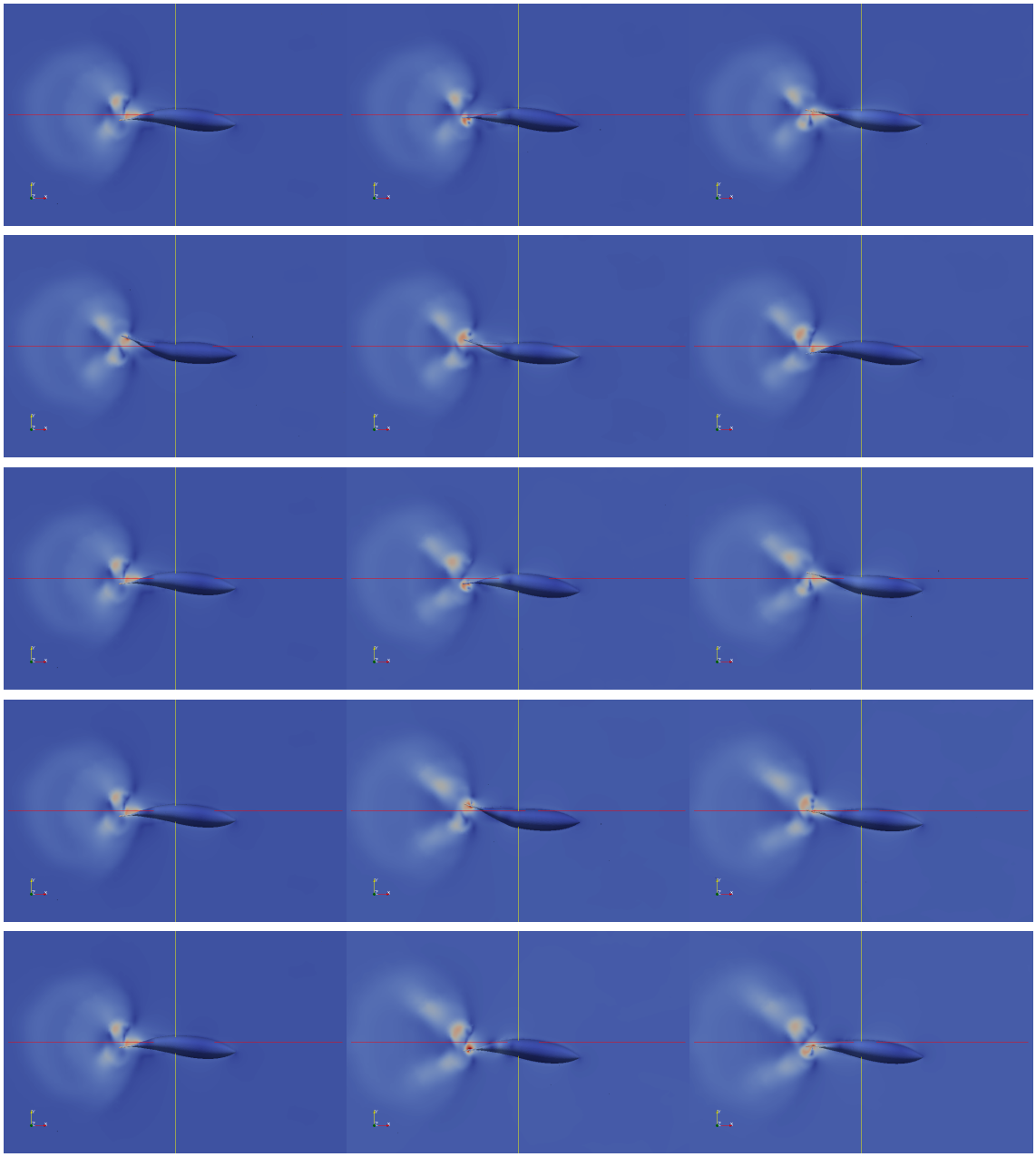
away down stream as observed with fish in nature. The full animation can be found at the following address (`http://youtu.be/O3malAc0Rfg`).

The vortices are focused mainly on the tail portion of the robotic fish body. However the downstream presence of vortices is so visible due to the limited far field resolution of the simulation study. The presence of attached vortices indicates that the robotic fish may be benefiting from vortex peg interaction, which may be contributing to the propulsive efficiency as suggested by Rosen (1959).

6.5 Concluding Remarks

In this chapter a deadbeat controller for state energy based on a local discretization of the plant model for an underactuated robotic fish was presented. Simulation results demonstrated that the controller resulted in periodic motion of the body of the simulated robotic fish. The system experienced two distinct periodic motions seeming to correspond to distinct system transients. Both periodic motions were on the surface of the objective region. Therefore it can be said that the controller successfully

Table 6.1: Velocity of fluid surrounding the robotic fish



bounded the system to the objective region.

Results demonstrated that both the periodic motions were effective swimming gaits. However the motion corresponding to the lower frequency transient resulted in significantly more thrust with reduced control input cost. The higher frequency

transient was associated with a significantly larger decay coefficient, the question arises as to why a transient with such a high decay coefficient remained prevalent within the system. A possible explanation of the initial dominance of the higher frequency transient was perhaps that due to the sharpness of the discrete control moves of the deadbeat controller provided high frequency stimulus which excited the transient which would have otherwise have decayed quickly.

These results can be considered a proof of concept that the control of state energy can lead to an effective swimming gait on an under actuated robotic fish. However the fact the deadbeat controller chosen did not discriminate between system transients and actively stimulated a less desirable transient suggests that the deadbeat energy controller is a poor choice of energy controller for this particular application.

The yaw instability observed in figures 6.2 suggests that some further work is needed in directional control. Strategies for controlling both yaw and state energy with a single controllable input would reduce the need for ancillary systems. It is likely in this case that the yaw instability was caused by the selection of a singular $\mathbf{Q}$ matrix for control. As a result the n dimensional surface to which the controller was fixing state orbit to was unbounded along the global yaw axis.

The presence of negative work in the controller input suggests that further improvement can be made in the efficiency of application. The potential for an additional $10\% - 15\%$ energy saving is a clear motivation for further study. As the $\mathbf{Q}$ matrix was selected to represent the physical energy of the system (mass and kinetic) the additional energy gained to necessitate negative work must have originated in exogenous force inputs from fluid interactions. The periodic nature of the negative work suggests that the fluid inertia effects form a significant part of the system dynamics, however it is very difficult to accurately model these effects in real time, furthermore such effects may also be dependent on variable external factors such as forward velocity. This is a major

problem since the deadbeat control's dependence on modelling accuracy makes the approach highly fragile.

In chapter 7 H_∞ continuous time robust non fragile control techniques will be investigated to minimise the effect of exogenous force inputs, modelling inaccuracies and imperfect controller implementation on total system energy. It is hoped that by giving the system a tolerance to modelling error, some of the negative work due to exogenous fluid effects can be avoided. Furthermore by applying a continuous time technique the control moves will be 'smoother' resulting in less high frequency stimulation.

Chapter 7

Design of a reduced fragility H_∞ observer feedback controller for the control of state energy

This chapter will discuss robust control techniques in conjunction with the parametric sensitivity of closed loop system norms, with the aim of assessing and reducing the fragility of closed loop systems.

7.1 Introduction

In chapter 6 deadbeat controller for state energy was derived from the system model which is simple and easy to use. However results indicated that the discrete nature of the controller resulted in a non smooth control input which excited undesirable system transients. This chapter will investigate the application of a continuous time robust control approach to the control of state energy applied to gait generation for the model of the robotic fish. Specifically H_∞ disturbance rejection feedback control will be used to reduce the dependence of the closed loop system on model accuracy.

Popular state space solutions for feedback control of linear time invariant (LTI) systems satisfying the H_∞ robustness criteria can be found in Doyle et al. (1992), and Skogestad and Postlethwaite (2007).

H_∞ control is widely used for noise rejection and robust position control in a wide range of complex structures subject to exogenous forces. Examples include geometric kinematic control of manipulator arms (Tumari et al. 2012). H_∞ control of periodic motion has revived a significant amount of interest albeit primarily with the interest of eradicating vibration, either in the end effector of manipulator robots (Douat et al. 2011) or in high-rise buildings (Zapaterio et al. 2011) rather than generation of periodic motion as in this application. H_∞ control has also been applied to gait generation for an ostraciiliform swimming robot (Hur et al. 2009).

However it has been suggested that some solutions to the H_∞ can be described as highly fragile (Keel and Bhattacharyya 1997). This means that small inaccuracies in the application of the controller or small parametric modelling error can result in total failure of the imposed robustness condition.

Several methods have been developed to incorporate additional non-fragility criteria with the robustness criteria into the controller design phase by including an operational envelope, where robustness is guaranteed for a specific parametric range (Kim and Oh 2007). However a critical issue with any design objective is quantification, Dorato et al. (1999) proposed the use of phase and gain margins to quantify the fragility of closed loop systems, however this method is limited to systems without infinite gain margin. i.e. systems that cross the zero axis on the Bode plot once.

This study however will present a novel quantification of fragility through a sensitivity analysis of the robustness criteria with respect to system parameters. For most systems there is no explicit equation for the H_∞ norm in terms of the system parameters, therefore an exact derivative is not available. However a novel approach is adopted in this study where an upper bound for the H_∞ norm is derived based on the equivalence with the furthest point from the origin of the Nyquist plot. This upper bound is expressible as an explicit function in terms of the system parameters and therefore an

exact parametric derivative exists.

The remainder of this chapter will be divided into five further sections. Section 7.2 discusses the application of H_∞ control methods to the control of energy. Section 7.3 details the derivation of an explicit function for the upper bound of the system H_∞ norm then go on to provide the derivative of said function with respect to the system dynamic matrix for state space cases. Section 7.4 discusses the use of gradient decent to increase the stability margin of robust closed loop systems. In section 7.5 results are presented comparing the gradient decent optimised H_∞ controller with the original H_∞ controller. Finally section 7.6 presents some concluding remarks.

7.2 H_∞ robust energy control for a robotic fish

The robotic fish described by equations 4.14 can be linearised as shown in equation 6.7 and rewritten as,

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B_1}\mathbf{w} + \mathbf{B_2}\mathbf{u} \\ E &= \mathbf{x}^\mathbf{T}\mathbf{Q}\mathbf{x} \end{aligned} \tag{7.1}$$

transforming this into the Laplace domain gives,

$$E = \begin{bmatrix} \mathbf{w}^\mathbf{T} & \mathbf{u}^\mathbf{T} \end{bmatrix} \begin{bmatrix} \mathbf{B_1^T} \\ \mathbf{B_2^T} \end{bmatrix} (s\mathbf{I} - \mathbf{A}^\mathbf{T})^{-1}\mathbf{Q}(s\mathbf{I} - \mathbf{A})^{-1} \begin{bmatrix} \mathbf{B_1} & \mathbf{B_2} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{u} \end{bmatrix} \tag{7.2}$$

The Laplace domain function for energy can then be expressed as the sum of three functions,

$$E = \mathbf{u}^\mathbf{T}\mathbf{B_2^T}\mathbf{G}\mathbf{B_2}\mathbf{u} + \mathbf{w}^\mathbf{T}\mathbf{B_1^T}\mathbf{G}\mathbf{B_1}\mathbf{w} + \mathbf{2}\mathbf{u}^\mathbf{T}\mathbf{B_2^T}\mathbf{G}\mathbf{B_1}\mathbf{w} \tag{7.3}$$

where,

$$\mathbf{G} = (s\mathbf{I} - \mathbf{A}^\mathbf{T})^{-1}\mathbf{Q}(s\mathbf{I} - \mathbf{A})^{-1} \tag{7.4}$$

Assuming $||\mathbf{B_2 u}||_\infty \gg ||\mathbf{B_1 w}||_\infty$ then the energy introduced due to the third term in equation 7.3 can be assumed to be negligible, therefore the system can be approximated by the first two terms. Figure 7.1 shows a block diagram of the open loop system between controllable and uncontrollable inputs and total system energy. The total energy is given by the summation of all the sub systems, however if $||\mathbf{B_2 u}|| \gg ||\mathbf{B_1 w}||$ then the contribution of the components in the dashed box will be relatively small therefore if the system components in the dashed box are neglected a reasonable approximation is given by the lower sub system.

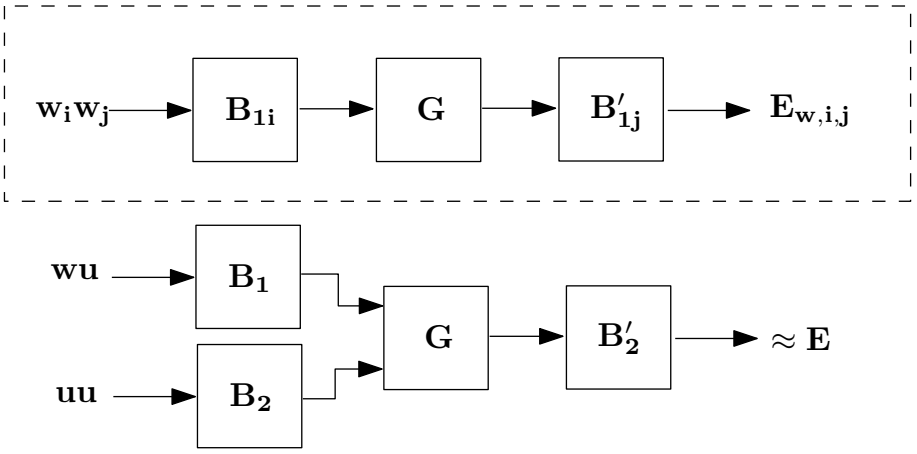


Figure 7.1: Open loop block diagram of state energy transfer function

The remaining open loop transfer function obtained by omitting the entirely uncontrollable terms from equation 7.2 can then be expressed as,

$$E \approx \mathbf{u^T B_2^T G} \begin{bmatrix} \mathbf{B_1} & \mathbf{B_2} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{u} \end{bmatrix} \tag{7.5}$$

The goal is to find some stabilising state feedback controller $\mathbf{K}$ which ensures robustness through bounded H_∞ norm of the gain between exogenous inputs $\mathbf{w}$ and state energy for the closed loop system. Substituting one of the $\mathbf{u}$s in equation 7.5 for

the controller output,

$$\mathbf{u} = \mathbf{KC_2x} \tag{7.6}$$

where $\mathbf{C_2x}$ is the observable output of the system. Then the closed loop system energy becomes,

$$E = \mathbf{u}\left[\mathbf{B_2^T GB_1} + \frac{\mathbf{B_2^T GB_2 KC_2 GB_1}}{\mathbf{1 + KC_2 GB_2}}\right]\mathbf{w} \tag{7.7}$$

By the multiplication inequality which holds for matrix and system ∞ norms, it can be stated,

$$\left|\left|\mathbf{u}\left[\mathbf{B_2^T GB_1} + \frac{\mathbf{B_2^T GB_2 KC_2 GB_1}}{\mathbf{1 + KC_2 GB_2}}\right]\right|\right|_\infty \leq ||\mathbf{u}||_\infty \left|\left|\left[\mathbf{B_2^T GB_1} + \frac{\mathbf{B_2^T GB_2 KC_2 GB_1}}{\mathbf{1 + KC_2 GB_2}}\right]\right|\right|_\infty \tag{7.8}$$

Since it is known that $||\mathbf{u}||_\infty$ will be bounded by the natural constraints of the system it remains only to find a controller $\mathbf{K}$ which satisfies,

$$\left|\left|\left[\mathbf{B_2 GB_1} + \frac{\mathbf{B_2 GB_2 KC_2 GB_1}}{\mathbf{1 + KC_2 GB_2}}\right]\right|\right|_\infty < \infty \tag{7.9}$$

Figure 7.2 shows the closed loop system block diagram relating uncontrollable inputs to $\mathbf{z}$. The observable output is connected to the controllable input through the controller $\mathbf{K}$ via positive feedback.

Satisfying the condition described in equation 7.8 is equivalent to ensuring that the closed loop system shown in figure 7.2 is H_∞ bounded. A standard solution to this problem can be found in Glover and Doyle (1988).

Signal tracking control requires the introduction of an additional reference input into the system giving the system input as,

$$\mathbf{u} = \mathbf{KC_2x} + \mathbf{a} \tag{7.10}$$

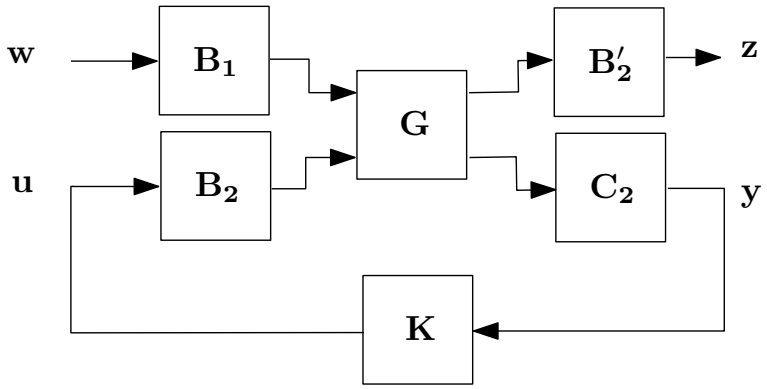


Figure 7.2: Closed loop noise rejection system block diagram

Where **a** is some additional input introduced. Figure 7.3 shows the closed loop system between uncontrollable inputs **w** and an abstract output **z** with an additional reference signal **a** added to the feed back signal.

Multiplying the system in figure 7.3 by **u** returns the system to the the closed loop system between **ua** and total state energy E.

$$E = \frac{\mathbf{B_2^T G B_2 C_2 G B_2}}{\mathbf{1 - C_2 G B_2}}\mathbf{ua} \tag{7.11}$$

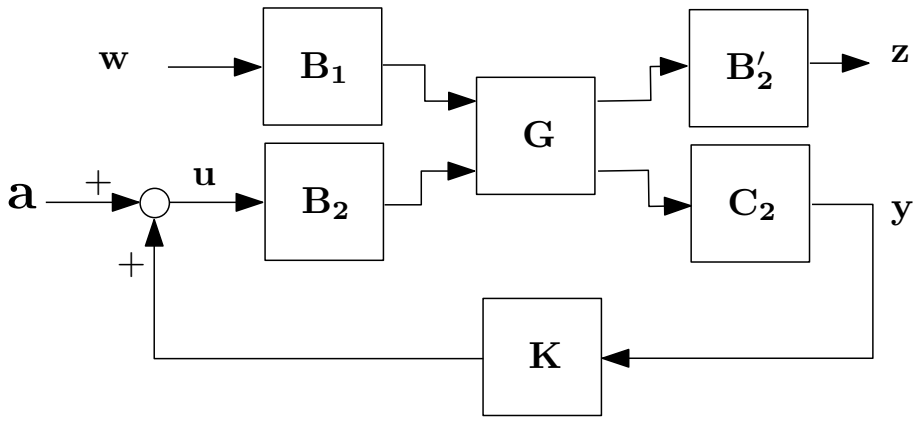


Figure 7.3: Closed loop noise rejection system with additional input block diagram

Error feedback energy control can then be achieved by implementing the closed loop system. Figure 7.4 shows a block diagram of the closed loop system between energy

reference signal $\mathbf{r}$ and energy E. The error signal $\mathbf{e}$ is generated with negative energy feedback. $\hat{\mathbf{G}}$ is the closed loop system between $\mathbf{au}$ and $\mathbf{E}$ given by,

$$\hat{\mathbf{G}} = \frac{\mathbf{B_2^T G B_2 C_2 G B_2}}{\mathbf{1 - C_2 G B_2}} \tag{7.12}$$

and $\mathbf{K_e}$ is some realisable error feedback controller. The additional input $\mathbf{a}$ can then be determined as $\mathbf{a} = \mathbf{u^{-1} K_e e}$. Substituting $\mathbf{a}$ into equation 7.10 gives,

$$\mathbf{u} = \mathbf{K C_2 x} + \frac{\mathbf{K_e e}}{\mathbf{u}} \tag{7.13}$$

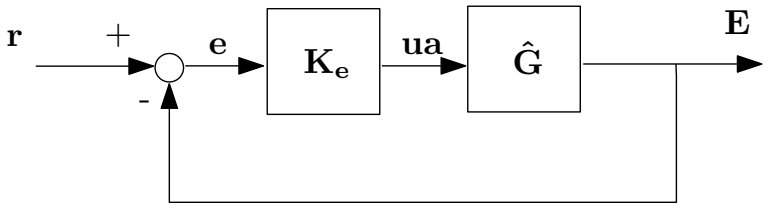


Figure 7.4: Closed loop error feedback control of energy

Equation 7.13 can be solved for $\mathbf{u}$ via the standard quadratic formula,

$$\mathbf{u} = \frac{1}{2}\left(\mathbf{K C_2 x} \pm \sqrt{(KC_2x)^2 + 4K_e e}\right) \tag{7.14}$$

Like the deadbeat controller featured in chapter 6 the $\pm$ gives the controller a decision element. For minimum cost control the least magnitude option can be taken. However by introducing a directional bias into the decision a degree of direction control can also be achieved. It was demonstrated in Roper et al. (2013) that such a directional bias can deliver a degree of yaw control to an underactuated robotic fish.

7.3 Parametric Sensitivity of H_∞ Norm

The ∞ norm of a system is defined as;

$$||\mathbf{G}||_\infty = \sup_\omega |\mathbf{G}(j\omega)| \tag{7.15}$$

and is equivalent to the maximum distance of the Nyquist plot of $\mathbf{G}$ from the origin. The H_∞ robustness criteria requires $||\mathbf{G}||_\infty < \infty$ for the closed loop system $\mathbf{y} = \mathbf{G_{w,y}w}$, where $\mathbf{w}$ is the vector of uncertain disturbances affecting the system.

The localized gradient of any continuous function along a given path can be found by the first derivative with respect to the vector of travel. So the local sensitivity of the H_∞ robustness condition to perturbations in the system could be described as,

$$\mathbf{S}(\mathbf{G}) = \frac{\partial ||\mathbf{G}||_\infty}{\partial \mathbf{G}} \tag{7.16}$$

However since there is no explicit function for $f(\mathbf{G}) = ||\mathbf{G}||_\infty$ in terms of the parameters of $\mathbf{G}$ (Doyle et al. 1992) an exact analytical value for $\mathbf{S}(\mathbf{G})$ cannot be found.

However using the equivalence of the ∞ norm to the maximum distance on the Nyquist plot it is possible to define an explicit function in terms of the system parameters of $\mathbf{G}$ that gives an upper bound on the ∞ norm.

Any n^{th} order LTI system $\mathbf{G}$ is expressible as a series of n partial first order systems, i.e.

$$\mathbf{G} = \sum_{i=1}^{n} \mathbf{G_i} \tag{7.17}$$

where $\mathbf{G_i}$ is of the form,

$$\mathbf{G_i} = \frac{b}{s+a} \tag{7.18}$$

where a and b belong to the set of complex values. The triangle inequality gives that,

$$||\mathbf{G}||_\infty \leq \sum_{i=1}^{n} ||\mathbf{G_i}||_\infty \tag{7.19}$$

therefore if the norms of each of the sub systems can be determined, an upper bound on the total system norm can be found. For a first order system the maximum distance of the Nyquist plot from the origin is simple to find. Transformation into the frequency domain yields.

$$\hat{\mathbf{G}}_\mathbf{i} = \frac{b}{\omega j + a} \tag{7.20}$$

The H_∞ norm of this sub system occurs at the ω that maximises the magnitude function. i.e. .

$$||\hat{\mathbf{G}}_\mathbf{i}||_\infty = sup_{\omega \geq 0} \left| \frac{b}{\omega j + a} \right| = |b| sup_k \left(\frac{1}{a_r^2 + (k^2 + a_i)^2} \right)^{0.5} \tag{7.21}$$

where a_r and a_r are the real and imaginary parts of a respectively and $k^2 = \omega$. This maximum must occur when the derivative with respect to k is equal to 0. Taking the differential of the part dependent on k,

$$\frac{d}{dk} \left(\frac{1}{a_r^2 + (k^2 + a_i)^2} \right)^{\frac{1}{2}} = \frac{-2k(k^2 + a_i)}{(a_r^2 + (k^2 + a_i)^2)^{\frac{3}{2}}} \tag{7.22}$$

which has zeros at $k = 0$ and if $a_i \leq 0$ $k = \pm(-a_i)^{\frac{1}{2}}$. By taking the second differential it is then trivial to show that, if $a_i < 0$ then the point at $k = 0$ is a minimum, otherwise $k = 0$ is a maxima. Substituting these values into equation 7.21 will yield $||\hat{\mathbf{G}}_\mathbf{i}||_\infty$

However it is well known that for a real system imaginary poles always come in conjugate pairs i.e.

$$\hat{\mathbf{G}}_\mathbf{i} + \hat{\mathbf{G}}_\mathbf{i}^* = \frac{b}{\omega j + a} + \frac{b^*}{\omega j + a^*} \tag{7.23}$$

For such a conjugate pair, since one must have a maximum at $\omega = 0$ and the other at

$\omega = |a_i|$ then the maximum of the sum must lie in the frequency range, $0 \leq \omega \leq |a_i|$.

Multiplying the partial fractions gives the second order transfer function,

$$\hat{\mathbf{G}}_\mathbf{i} + \hat{\mathbf{G}}_\mathbf{i}^* = \frac{ab^* + a^*b + (b+b^*)\omega j}{aa^* - \omega^2 - (a+a^*)\omega j} \tag{7.24}$$

It can be shown that for a second order transfer function of the form,

$$\hat{\mathbf{G}}_\mathbf{i} = \frac{b_2 + b_1\omega j}{a_2 - \omega^2 + a_1\omega j} \rightarrow a_1, a_2, b_1, b_2 \in \mathbf{R} \tag{7.25}$$

the differential of the magnitude function will be,

$$\frac{d|\hat{\mathbf{G}}_\mathbf{i}|}{d\omega} = \frac{-4\omega(b_2^2 + b_1^1)^{\frac{1}{2}}(\omega^2 + \frac{1}{2}a_1^2 - a_2)}{((a_2 - \omega^2)^2 + (a_1\omega)^2)^{\frac{3}{2}}} \tag{7.26}$$

which has zeros at $\omega = 0$ and $\omega = (a_2 - \frac{1}{2}a_1^2)^{\frac{1}{2}}$.

Again it is trivial to show by using the second derivative that if $a_2 > \frac{1}{2}a_1^2$ then $\omega = (a_2 - \frac{1}{2}a_1^2)^{\frac{1}{2}}$ will be a maximum. Substituting the values $a_2 = aa^*$ and $a_1 = -(a+a^*)$ gives, $\omega = real((a_i^2 - 2a_r^2)^{\frac{1}{2}})$, which implies that;

$$\arg\max_\omega |\hat{\mathbf{G}}_\mathbf{i} + \hat{\mathbf{G}}_\mathbf{i}^*| = \begin{cases} 0 & \sqrt{2}|a_r| > |a_i| \\ a_i^2 - 2a_r^2 & otherwise \end{cases} \tag{7.27}$$

also it follows that if a transfer function $\mathbf{G}$ has all poles inside the region $|z_i| \leq -\sqrt{2}z_r$, then $arg_\omega max|\hat{\mathbf{G}}| = 0$ and the ∞ norm can be calculated exactly as $||\hat{\mathbf{G}}||_\infty = \sum |\hat{\mathbf{G}}_\mathbf{i}(0)|$. Figure 7.5 shows a plot of the region $|z_i| \leq -\sqrt{2}z_r$, as z_r decreases the width of the region increases.

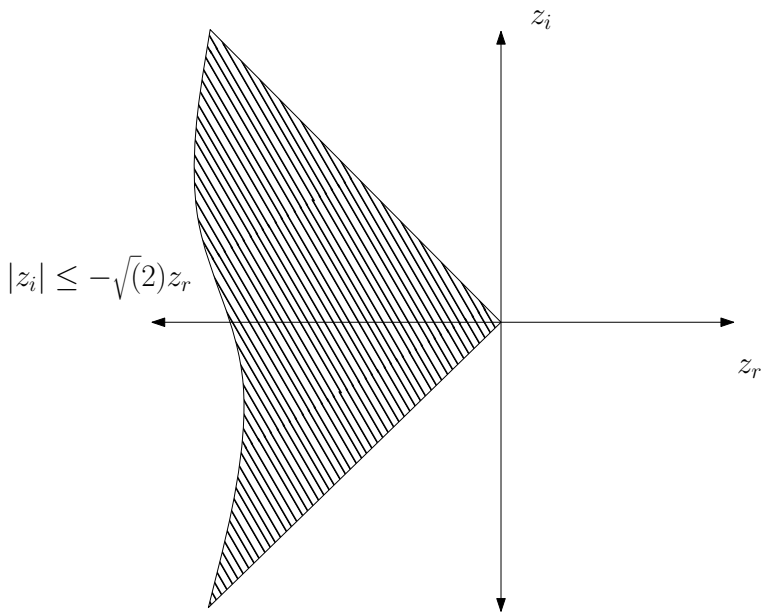


Figure 7.5: Complex region, which if all Eigenvalues fall within, system must have $arg_\omega max|\hat{\mathbf{G}}(\omega j)| = 0$

Otherwise, an upper bound on the infinity norm is given by,

$$||\mathbf{G}||_\infty \leq \sum_i |\hat{\mathbf{G}}_\mathbf{i}(\arg\max_\omega |\hat{\mathbf{G}}_\mathbf{i} + \hat{\mathbf{G}}_\mathbf{i}^*|)| \tag{7.28}$$

For the state space system,

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} \end{aligned} \tag{7.29}$$

Then if Λ is the diagonal matrix of eigenvalues and P is the matrix of eigenvectors the equivalent system,

$$\begin{aligned} \dot{\mathbf{z}} &= \Lambda\mathbf{z} + \mathbf{P}^{-\mathbf{1}}\mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{P}\mathbf{z} \end{aligned} \tag{7.30}$$

will be the diagonal form of the original. The frequency domain transfer function will be given by,

$$\hat{\mathbf{G}} = \mathbf{CP}(\omega j\mathbf{I} - \Lambda)^{-1}\mathbf{P^{-1}B} \tag{7.31}$$

Then if $-\sqrt{2}\lambda_r > |\lambda_i| \rightarrow \forall\lambda \in \mathbf{G}$, substituting $\omega = 0$ into equation 7.31 yields;

$$||\mathbf{G}||_\infty = -abs(\mathbf{CP})\Lambda^{-1}abs(\mathbf{P^{-1}B}) \tag{7.32}$$

where $abs(\mathbf{x})$ means the element by element absolute value of the vector $\mathbf{x}$. More generally the upper bound on the ∞ norm can be expressed as;

$$||\mathbf{G}||_\infty \leq abs(\mathbf{CP})abs([\Omega j - \Lambda])^{-1}abs(\mathbf{P^{-1}B}) \tag{7.33}$$

where,

$$\Omega = real\{(imag(\Lambda)^2 - 2real(\Lambda)^2)^{\frac{1}{2}}\} \tag{7.34}$$

Critically equations 7.33 and 7.34 give a value defined explicitly in terms of the system parameters and thus a sensitivity with respect to system parameters can be found.

If $F_\infty(\mathbf{G})$ is defined as the upper bound on $||\mathbf{G}||_\infty$ given by the right hand side of equation 7.33 then if $\mathbf{G}(\mathbf{K})$ is a system expressed as a function of the matrix $\mathbf{K}$

$$\begin{aligned}\frac{\partial F_\infty(\mathbf{G})}{\partial \mathbf{K}} = &\frac{\partial\ abs(\mathbf{C_1P})}{\partial \mathbf{K}}\left(\mathbf{I_c} \otimes abs(\Omega j - \Lambda)^{-1}abs(\mathbf{P^{-1}B_1}\right) \\ &+ (\mathbf{I_r} \otimes abs(\mathbf{C_1P}))\frac{\partial(\Omega j + \Lambda)^{-1}}{\partial \mathbf{K}}\left(\mathbf{I_c} \otimes abs(\mathbf{P^{-1}B_1}\right) \\ &\left(\mathbf{I_r} \otimes abs(\mathbf{C_1P})(\Omega j - \Lambda)^{-1}\right)\frac{\partial\ abs(\mathbf{P^{-1}B_1})}{\partial \mathbf{K}}\end{aligned} \tag{7.35}$$

where $\otimes$ is the Kronecker product, r and c are the number of rows and columns of $\mathbf{K}$

respectively, $\bar{U}$ is the matrix self differential. Ω is given by equation 7.34, and Λ and $\mathbf{P}$ are the matrix of eigenvalues, and eigenvectors of the system $\mathbf{A}$ matrix respectively.

For a guide to the differentiation eigenvalues and eigenvectors the interested reader is directed to Der et al. (2007). The assumption is then made that $\partial F_\infty(\mathbf{G})/\partial \mathbf{K} \approx \partial ||\mathbf{G}||_\infty / \partial \mathbf{K}$

7.4 Gradient decent to minimize parametric sensitivity

Closed loop system norm's between exogenous inputs and control outputs are often used for analysis of robustness. It may often not be desirable to minimize the said norms as this can lead to overly conservative controllers with poor signal tracking properties. However it is desirable to ensure that robustness conditions are non-fragile. The proposed method here is to employ a few iterations of gradient decent. The objective being to move away from steep regions of the space of parametric system norms.

Figure 7.6 demonstrates how a gradient decent method could increase the stability margin of a highly sensitive parameter. If f_0 is a parameter in a robust closed loop system and the dashed line represents the parametric value at which the robustness fails, movement down the gradient will move the parameter away from the point of failure and hence will increase the stability margin of the parameter increasing the allowable error before the robustness will fail.

7.5 Results

An initial H_∞ stabilising observer feedback controller was found using the state space method from Skogestad and Postlethwaite (2007). The controller consisted of a state observer,

$$\dot{\hat{\mathbf{x}}} = \tilde{\mathbf{A}}\hat{\mathbf{x}} + \mathbf{B_1}\gamma^{-2}\mathbf{B_1^T}\mathbf{X}_\infty\hat{\mathbf{x}} + \mathbf{B_2}\mathbf{u} + \mathbf{Z}_\infty\mathbf{L}_\infty(\mathbf{C_2}\hat{\mathbf{x}} - \mathbf{y}) \tag{7.36}$$

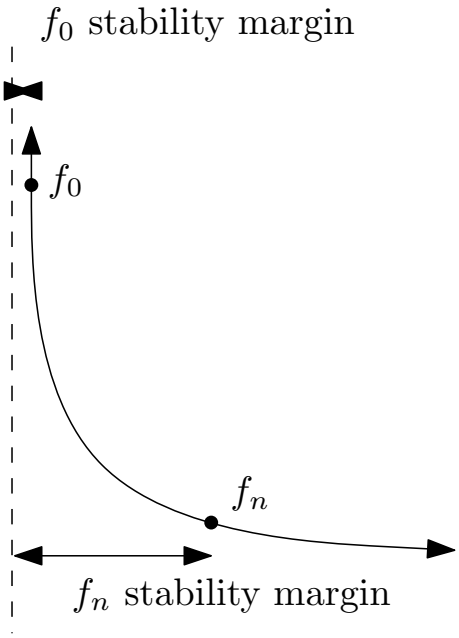


Figure 7.6: Gradient decent increasing stability margin

where $\tilde{\mathbf{A}}$ is the dynamic matrix representation of the subsystem **G** from equation 7.4 given by,

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A^T} & Q \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \tag{7.37}$$

$\mathbf{B_1}, \mathbf{B_2}$ and $\mathbf{C_2}$ were the system input and output matrices as shown in figure 7.2. $\mathbf{X}_\infty, \mathbf{Z}_\infty$ and $\mathbf{L}_\infty$ are matrix parameters derived using the method in Skogestad and Postlethwaite (2007), values for which can be found in appendix A. The value of $\gamma = 10$ was selected through trial and error as a working norm bound.

The input signal **u** was then generated with the feedback of observed state, $\mathbf{u} = \mathbf{F}_\infty \hat{\mathbf{x}}$. Where,

$$\mathbf{F}_\infty = \begin{bmatrix} -3.2e-02 & 1.2e-01 & 1.4e-01 & 1.5e-01 & -1.5e-02 & \dots \\ -9.3e-04 & 2.5e-01 & 1.1e-01 & -2.9e-03 & 1.7e-03 & \dots \\ 9.4e-04 & 2.3e-04 & -3.6e-03 & 1.9e-03 & 1.2e-03 & 3.9e-04 \end{bmatrix}$$

Assuming perfect state observation the upper bound on the closed loop system norm was found to be $||\hat{\mathbf{G}}_{\mathbf{w},\mathbf{z}}||_\infty < 12.96$. That this is larger than γ is no suprise as this is conservative upper bound. Where $\hat{\mathbf{G}}_{\mathbf{w},\mathbf{z}}$ represents the closed loop system between uncontrollable inputs $\mathbf{w}$ and output $\mathbf{z}$ featured in figure 7.3. The parametric sensitivity of the H_∞ norm upper bound was found to be,

$$\frac{\partial F_\infty(\hat{\mathbf{G}}_{\mathbf{w},\mathbf{z}})}{\partial \mathbf{F}_\infty} =$$

$$\begin{bmatrix} -1.6e+03 & -2.9e+02 & -9.0e+01 & -2.8e+01 & 3.8e+02 & \dots \\ 6.8e+01 & 2.3e+01 & 5.7e+00 & 1.4e+01 & 1.4e+01 & \dots \\ 1.3e+01 & 1.5e+01 & -3.2e+00 & -5.8e+01 & 6.2e+01 & -1.6e+02 \end{bmatrix}$$

The frobenius norm of the parametric sensitivity matrix was found to be ≈ 1697.7. Gradient decent was used to further optimise the F_∞ matrix,

$$\hat{\mathbf{F}}_\infty = \begin{bmatrix} -4.0e-02 & 1.2e-01 & 1.5e-01 & 1.5e-01 & 5.2e-02 & \dots \\ 1.1e-02 & 2.5e-01 & 1.1e-01 & -4.7e-03 & -2.1e-04 & \dots \\ -9.4e-04 & -1.6e-03 & -1.3e-02 & 4.9e-02 & 2.7e-02 & -1.4e-02 \end{bmatrix}$$

With the corresponding local parametric sensitivity,

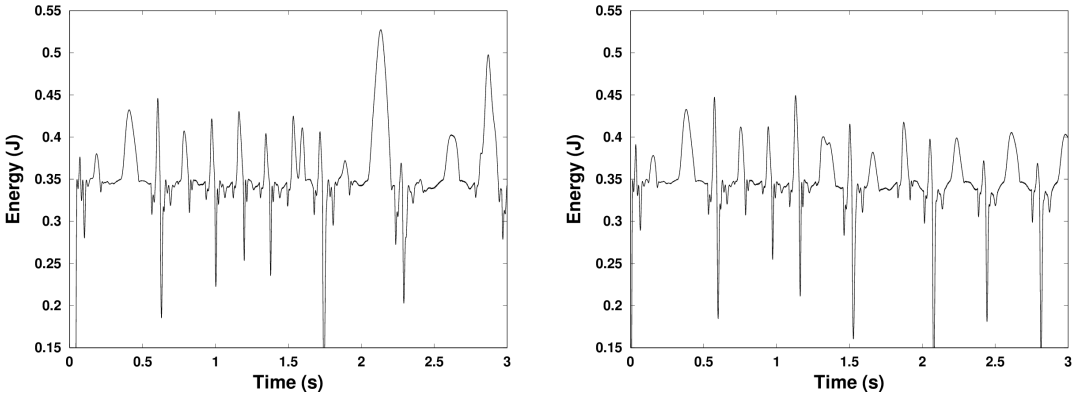
$$\frac{\partial F_\infty(\hat{\mathbf{G}}_{\mathbf{w},\mathbf{z}})}{\partial \mathbf{F}_\infty} =$$

$$\begin{bmatrix} 2.3e+00 & 3.7e-01 & 7.9e-02 & 3.7e-03 & -7.2e+00 & \dots \\ -1.3e+00 & -4.3e-01 & -1.0e-01 & 3.6e-01 & 3.9e-01 & \dots \\ 3.9e-01 & 3.9e-01 & 8.2e-01 & -1.5e+00 & -5.2e+00 & -6.5e+00 \end{bmatrix}$$

The frobenius norm of the local parametric sensitivity matrix was found to be ≈ 1.1477, representing a 1000 fold reduction in parametric sensitivity. Therefore observer inaccuracies will likely be tolerated significantly better.

A high gain proportional error feedback controller of $\mathbf{K_e} = 1000$ was selected for the error signal feedback as shown in figure 7.4.

As in chapter 6 a target of $0.35J$ was set as the objective region, the system spring, mass and inertia values were also the same as used in chapter 6. The controllers were implemented at a discrete time interval of $2e-4s$ and the simulation was run for 3 seconds. This shorter time interval was selected as it was long enough to visualise multiple instances of the periodic motion yet short enough to ensure that the mesh deformations did not prevent the pressure term in the CFD from resolving.



(a) (b)

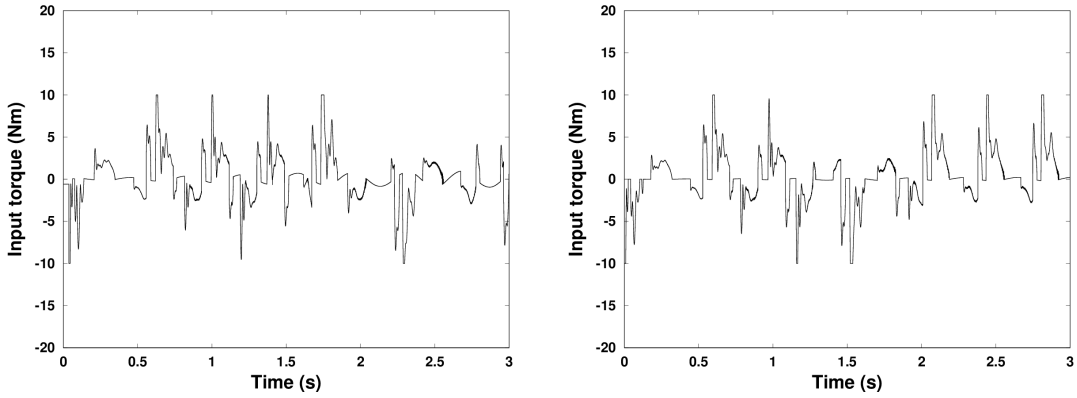
Figure 7.7: Plot of state energy against time (a) Original H_∞ state and error feedback controller, (b) Gradient optimised H_∞ controller.

The directional bias was implemented in the decision phase of the controller as follows,

$$\mathbf{u} = \begin{cases} \frac{\mathbf{F}_\infty\mathbf{x_o}+\sqrt{(\mathbf{F}_\infty\mathbf{x_o})^2+4\mathbf{K_e}\mathbf{e}}}{2} & \left|\frac{\mathbf{F}_\infty\mathbf{x_o}+\sqrt{(\mathbf{F}_\infty\mathbf{x_o})^2+4\mathbf{K_e}\mathbf{e}}}{2}\right| \leq \left|\frac{\mathbf{F}_\infty\mathbf{x_o}-\sqrt{(\mathbf{F}_\infty\mathbf{x_o})^2+4\mathbf{K_e}\mathbf{e}}}{2}\right| - 0.05\mathbf{q}_1 \\ \frac{\mathbf{F}_\infty\mathbf{x_o}-\sqrt{(\mathbf{F}_\infty\mathbf{x_o})^2+4\mathbf{K_e}\mathbf{e}}}{2} & otherwise \end{cases} \tag{7.38}$$

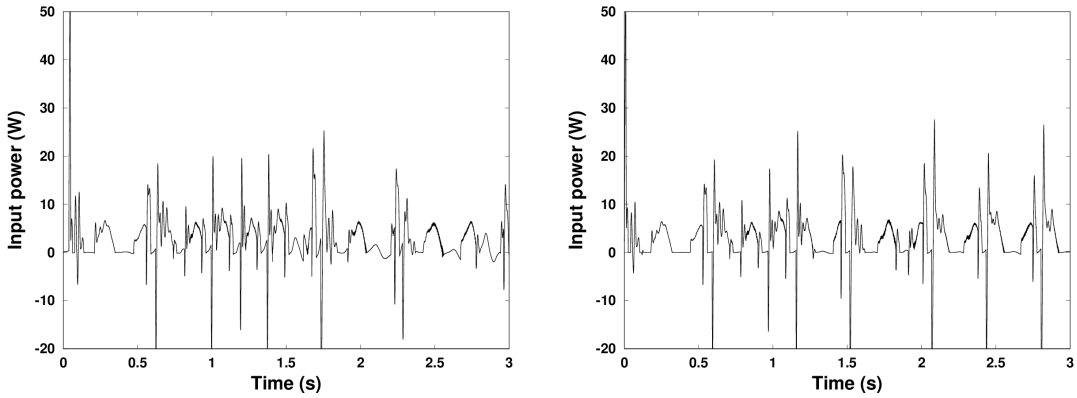
where $\mathbf{x_o}$ is the observer state, $\mathbf{e}$ is the energy error signal and $\mathbf{q}_1$ is the first element of the vector of bearings $\mathbf{q}$.

Figure 7.7 shows a plots of energy against time for the two controllers featured in this chapter. The original H_∞ state and error feedback controller resulted in a root mean square error of the energy signal of $0.0579J$, whereas the gradient optimised H_∞ controller resulted in a root mean square error of $0.0411J$.



(a) (b)

Figure 7.8: Plot of controller input signal against time (a) Original H_∞ state and error feedback controller, (b) Gradient optimised H_∞ controller



(a) (b)

Figure 7.9: Plot of controller input power against time (a) Original H_∞ state and error feedback controller, (b) Gradient optimised H_∞ controller

Figures 7.8 and 7.9 show plots of controller input signal and work against time for the two controllers. It can be seen that both controllers operated largely within the imposed torque constraints. The mean absolute work done by each of the controllers

was $3.5W$ for the original H_∞ and $3.5W$ for the gradient optimised H_∞ controller. The mean negative work was found to be 0.37 and $0.42W$ respectively. Representing approximately 10% and 12% negative work respectively.

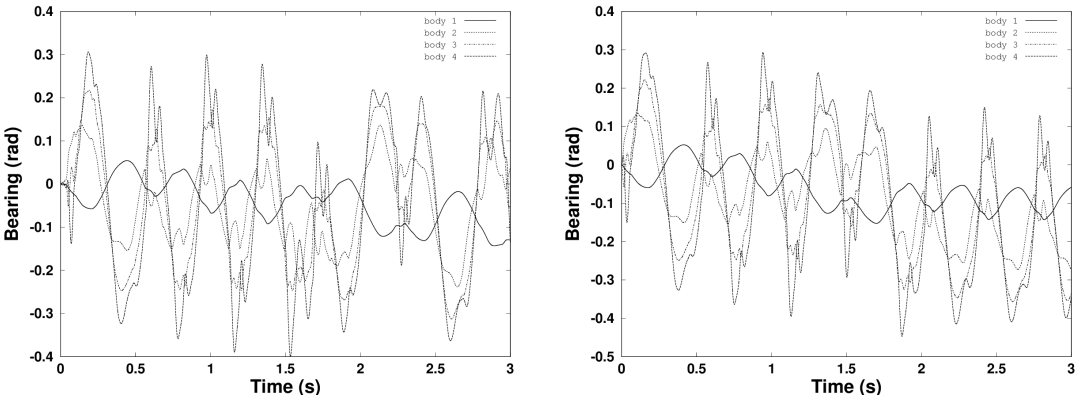
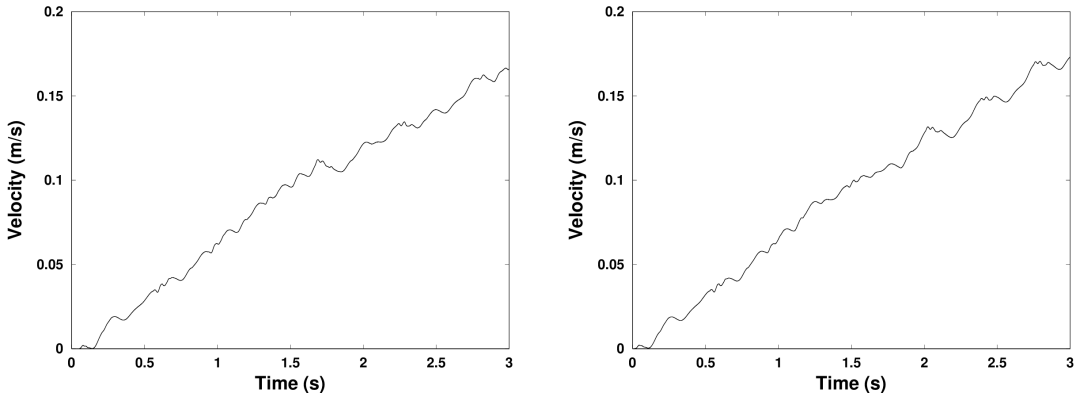


Figure 7.10: Plot of body bearing against time (a) Original H_∞ state and error feedback controller, (b) Gradient optimised H_∞ controller

The resultant body bearing motion of the robotic fish bodies is shown in figure 7.10. It can clearly be seen that the motion resulting from both controllers was non smooth, containing multiple transients, i.e. multiple peaks. It can also be seen that in both cases much of the yaw instability has been eradicated through the use of the decision bias. The original H_∞ resulted in a yaw rate of $\approx 0.029 rad\ s^{-1}$ whereas the optimised controller resulted in an initial yaw rate of $\approx 0.05 rad\ s^{-1}$ but stabilized after 2 seconds.

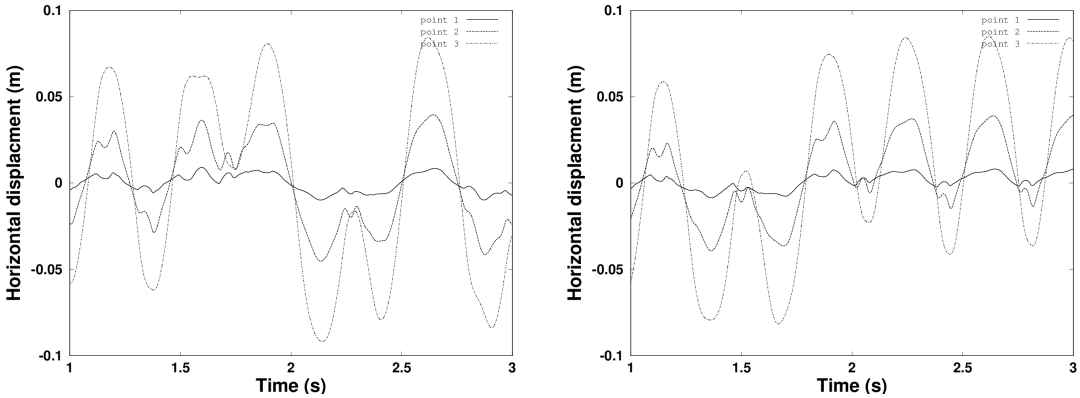
Figure 7.11 shows plots of forward velocity against time resulting from the periodic motion generated by each of the controllers. Both controllers resulted in forward thrust and acceleration from stationary of $\approx 0.055 ms^{-2}$ and $\approx 0.061 ms^{-2}$ respectively. However it can be seen that the acceleration from the original H_infty started to reduce significantly at around $1.75s$ however the acceleration optimised H_∞ remained approximately constant throughout the period shown.

Figures 7.12 and 7.13 show plots of body lateral displacement against time and lateral



(a) (b)

Figure 7.11: Plot of forward velocity against time (a) Original H_∞ state and error feedback controller, (b) Gradient optimised H_∞ controller



(a) (b)

Figure 7.12: Plot of lateral translation against time (a) Original H_∞ state and error feedback controller, (b) Gradient optimised H_∞ controller

displacement against longitudinal body position at samples within the time series respectively for the motion resulting from each of the controllers. The double peaks present in both figures 7.12 for the results from both controllers are likely an effect of the directional bias attempting to rectify the heading. It can be seen that the motion envelope is similar for the two controllers. Both have an expanding motion envelop in keeping with the optimal kinematic proposed in Barrett (1996).

Table 7.1 shows the velocity of the fluid around the robotic fish during the motion

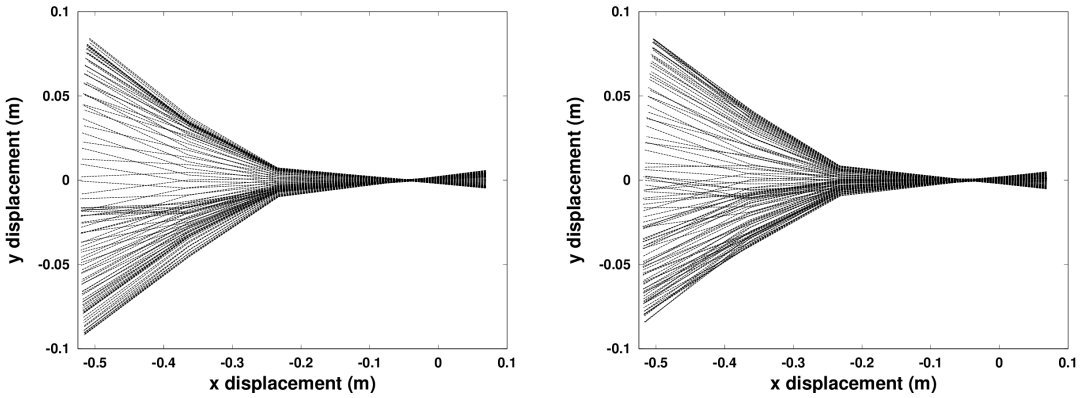


Figure 7.13: Resultant body kinematic (a) Original H_∞ state and error feedback controller, (b) Gradient optimised H_∞ controller

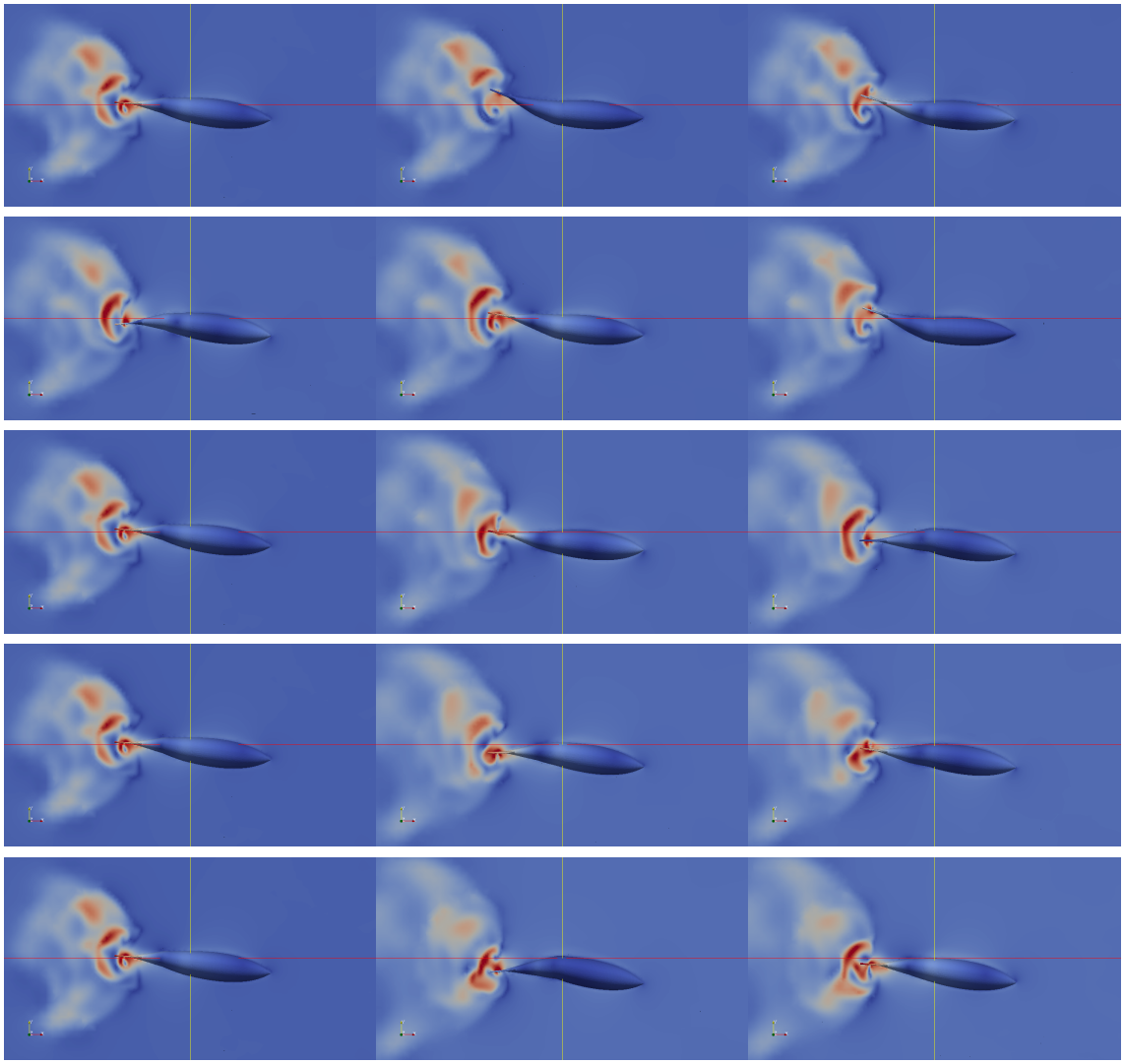
experienced due to the modified H_∞ controller. The vortices can be seen to be visibly more well defined than the vortices that developed due to the deadbeat controller shown in chapter 6. The vortices for the original H_∞ controller not shown here were similar in character. Videos of the two resultant motions can be found at (`http://youtu.be/TSdJPhPetvA`) and (`http://youtu.be/Dsl-wPzRSfM`) for the original H_∞ controller and for the optimised H_∞ respectively.

7.6 Concluding Remarks

In this chapter robust control of state energy was discussed. A closed loop disturbance rejection controller was proposed based on a popular state space solution to the H_∞ control problem for LTI systems.

Further to this an explicit parametric function was derived for an upper bound on LTI system H_∞ norm, based on an equivalence with the point of largest magnitude on the Nyquist plot. This explicit function was then used to derive a novel function for the parametric sensitivity of LTI system H_∞ norm. It was argued that gradient decent could be used to reduce the fragility of robust controllers. By moving parameters down the

Table 7.1: Velocity of fluid surrounding the robotic fish



norm gradients it was theorised that stability margins could be significantly increased.

Gradient decent optimisation of an H_∞ disturbance rejection controller resulted in a reduction in the local parametric sensitivity of the closed loop H_∞ norm by a factor of ≈ 1000.

Simulations were run with both the original H_∞ disturbance rejection controller and the gradient decent optimised version of the same controller.

Results demonstrated that both controllers resulted in periodic motion that resulted in forward thrust. However the optimised controller resulted in slightly improved disturbance rejection characteristics, due to a reduction in parametric reliance on observer accuracy and reduction in the norm of the closed loop system between disturbances and energy.

The controller demonstrated periodic inaccuracy. One possible explanation for the reduction in accuracy is the simplistic nature of error feedback control. Whilst for an n^{th} order LTI system with an $(n-1)^{th}$ order linear objective region the response of error to a given state space movement remains constant for all points in the state space. For lower dimensional or non-linear objective region such as the control of state energy the response of the error to a given state space movement is dependent on the state position relative to the objective region. This idea will be explored further in chapter 8.

Chapter 8

An alternative error energy control

This chapter will give details of a novel alternative error metric, for feedback control of system energy.

8.1 Introduction

In chapter 7 a method for reducing the parametric sensitivity of closed loop system norms was presented. By reducing the parametric sensitivity, the controller performance becomes less dependent on model and controller implementation accuracy. However the results presented in chapter 7 demonstrated significant periodic error suggesting that the linear controller applied was unable to fully compensate for the nonlinear nature of the objective region.

This chapter will attempt to derive an improved error feedback mechanism. Capable of adapting to nonlinear control objectives without adding unnecessary complexity to the controller.

Error feedback controllers such as *proportional integral and derivative* (PID) rely strongly on the consistency of the response of the error to a given input. However if the response of the error to a given input is inconsistent due to a nonlinearity in the objective region, i.e. at some points positive input results in positive error movement and at other points positive input results in negative error movement. Then a linear dynamic error feedback controller will not guarantee global stability. One possible

solution is to adapt the controller to the local input response (Wang et al. 2003). This can be achieved through sliding mode control (Yu and Kaynak 2009) or through online adaptive control (Jha et al. 2011). However the approach taken in this study is to adapt the definition of the error to ensure that the response of the error to input remains constant. This allows the error to be regulated through a single fixed controller. Furthermore this also allows the actual controller element to be easily interchanged whilst maintaining the ability to adapt to the local error response.

In this chapter a novel alternative error metric will be defined for use in feedback control tasks with nonlinear objective regions. An explicit equation for calculating this error metric for error in state energy will be derived and it will be demonstrated that a robust controller can be found using the proposed alternative error feedback.

The remainder of this chapter is divided into three further sections, section 8.2 defines 3 different measures of error and explain the choice of error metric for this chapter. Section 8.3 discusses robust control using the proposed alternative error feedback. Section 8.4 presents simulation results with the alternative error metric feedback controller. Finally section 8.5 presents a summary and concluding remarks.

8.2 Alternative Error Metrics for Feedback Control

For a given energy reference signal E_r, the objective region will be some nonlinear surface on R_n. Since it is known that a system cannot possess negative mechanical energy, it is necessarily true that $\mathbf{Q}$ is symmetrical positive semi-definite. As a result the problem can always be simplified by the state space transformation $\mathbf{z} = \mathbf{Q}^{\frac{1}{2}}\mathbf{x}$. The control objective region can then be expressed as $\mathbf{z^T z} = E_r$ which is equivalent to a hyper-sphere in R_n of radius $r = E_r^{0.5}$. So the objective is essentially converted to controlling radius of $\mathbf{z}$ from the origin i.e. $|\mathbf{z}| = \mathbf{r}$ where $\mathbf{r^2} = \mathbf{E_r}$. For clarity from this point forward the state space variable $\mathbf{z}$ will be used in place of $\mathbf{x}$ to signify

that this transformation has already been made. While numerous control techniques for linear control objectives already exist such as proportional integral differential (PID) error feedback or linear quadratic regulator (LQR). These techniques are not necessarily ideal for radius control objectives. Methods used to date for the control of energy in mechanical systems have been based heavily on linear control objective methods, such as linear state feedback (Spong 1996) or a PD error feedback with gravity compensation (Ortega et al. 2001). The problem with direct state or measured error feedback occurs when there is no available control move in the direction of least error or the response of the error to a given input is highly inconsistent. This can be seen more clearly if the two following definitions are made;

Definition 1: Measurable error (e); the minimum distance between the current state and the control objective region, equivalent to the numerical difference between output and reference signal.

Definition 2: Control vector error ($\hat{e}$); the minimum distance from the current state to the objective region along the direction of available controllable input vectors.

From Definition 1: measurable error for the radial control problem is given by the equation,

$$e = \mathbf{r} - |\mathbf{z}| \tag{8.1}$$

It can be easily seen that for a 1 dimensional $\mathbf{z}$ the response of the measured error to an increase in $\mathbf{z}$ is dependent on the sign of $\mathbf{z}$.

For a radial objective function it can be shown that the control vector error is governed by the equation,

$$\hat{e}(r, |\mathbf{z}|, \theta) = sign(\cos\theta)(r^2 - |z|^2 \sin^2\theta)^{\frac{1}{2}} - |\mathbf{z}|\cos\theta \tag{8.2}$$

As illustrated by figure 8.1, where θ is the minimum angle between available inputs and minimum direction of travel towards objective region. It can be seen that $|\hat{e}| \geq |e|$ for all combinations of system, objective region and state.

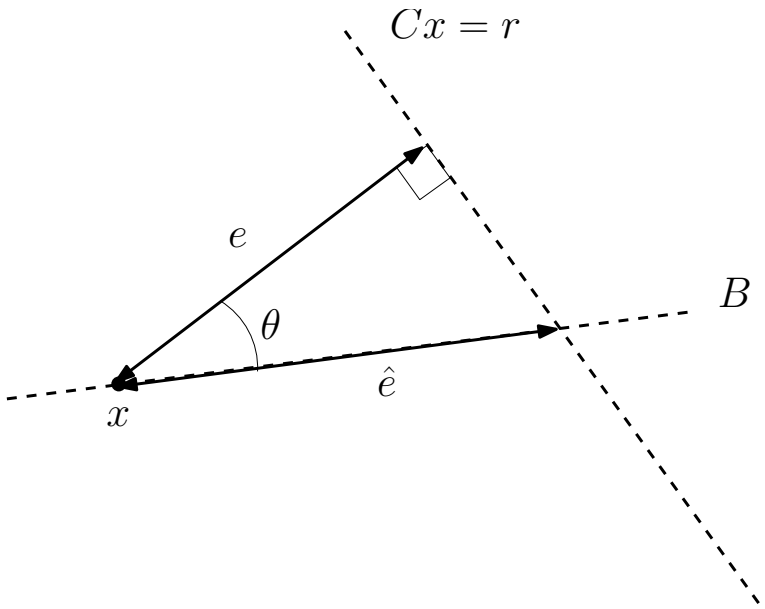


Figure 8.1: Linear state space objective problem

In the case of a linear control objective as illustrated by figure 8.1 the difference between e and $\hat{e}$ is easily calculable as $e = \hat{e}(\cos\theta)$. The response of e to a given movement along the vector B remains constant for all states. Therefore feeding back e will always result in an input proportional to the distance needed to be travelled through the state space to reach the objective region. However in the case of a radius control objective the relationship between measurable e and $\hat{e}$ is nonlinearly dependent on the current state. The response of e to a given movement along $\tilde{B}$ is nonlinearly dependent on the state. This suggests that e or any linear state combination would be a poor choice for feedback control. However it can be seen that the local response of $\hat{e}$ to a given movement along $\tilde{B}$ is constant for all states. This suggests feeding back $\hat{e}$ will result in the controller consistently providing an input proportional to the distance needed to be travelled through the state space to reach the objective. This is illustrated

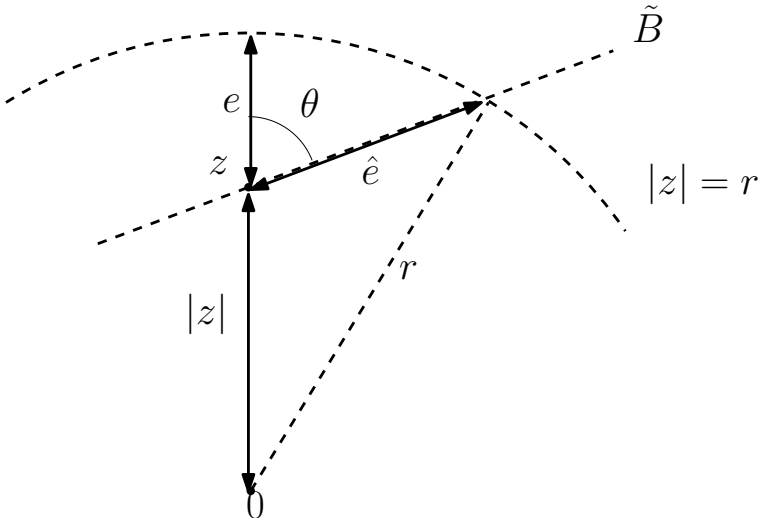


Figure 8.2: Quadratic state space objective problem

in figure 8.2 where measurable error e is the minimum distance to the curved objective function $|z| = r$ the available controllable direction of travel is $\tilde{B}$. The control vector error is the distance to be travelled along $\tilde{B}$ to reach the objective region.

As can be seen from equation 8.2 the control vector error becomes complex when $r^2 < |z|^2 \sin^2 \theta$ which is equivalent to a situation where no available input intersects with the objective region.

One could argue that if no available path intersects with the objective the best course would be to move along the path that minimizes the error. This is illustrated in figure 8.3 where the available input $\tilde{B}$ does not intersect with the objective region. Therefore $\tilde{e}$ represents a motion along $\tilde{B}$ to the point of minimum error. This can be reflected by a minor refinement of the error metric definition.

Definition 3: Best case control vector error (BCCVE) ($\tilde{e}$); the minimum distance from the current state to a point of minimum measured error along the direction of available controllable input vectors.

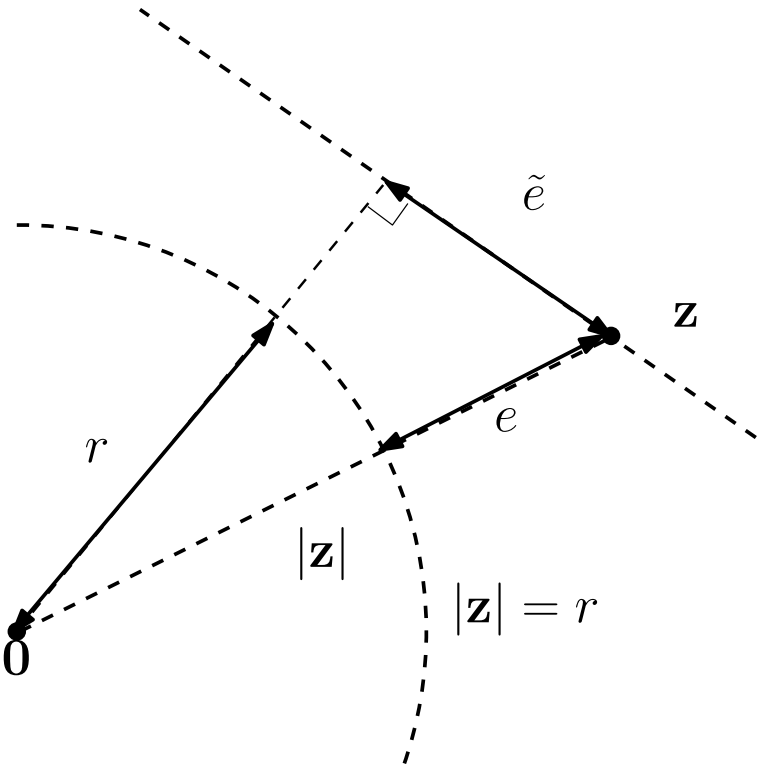


Figure 8.3: Quadratic state space objective problem with no controllable intersect

Thus whilst an intersect between the available controllable input vector and the objective region exists $\hat{e} \equiv \tilde{e}$, where no intersect exists $\tilde{e}$ gives the distance along the available input vector to the point of minimum error as shown in figure 8.3. Thus for a radial objective function it can be shown to be given by $-|z|\cos\theta$. Hence $\tilde{e}$ will then be governed by the equation,

$$\tilde{e}(r,|\mathbf{z}|,\theta) = sign(\cos\theta)real((r^2 - |\mathbf{z}|^2\sin^2\theta)^{\frac{1}{2}}) - |\mathbf{z}|\cos\theta \tag{8.3}$$

Figure 8.4 shows a three dimensional plot of $\tilde{e}$ against state magnitude for the region $|\mathbf{z}| \in \{0-3\}$ and angle for the region $\theta \in \{0-\pi\}rad$, with an objective of $r = 1$. From this plot three distinguishable regions of the state space can clearly be seen. The surface

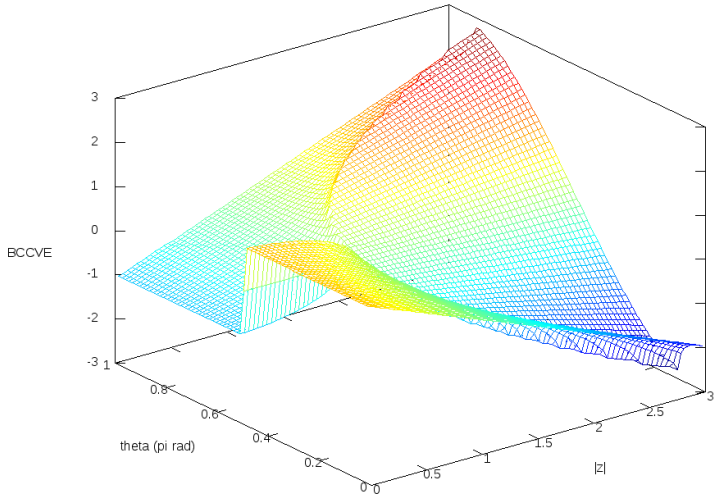


Figure 8.4: $\tilde{e}(1,|\mathbf{z}|,\theta)$

enclosed by a parabola to the right of the plot represents the region $r^2 < |\mathbf{z}|^2 \sin^2\theta$, where no intersecting solution exists. To the left hand side of the plot a distinction can be made between the region where the most positive intersection of the objective region is nearest, and where the most negative intersection of the objective region is nearest. Between these two regions lies a discontinuity, where both intersections are equidistant. It is also noteworthy that e is equivalent to the two dimensional cross section of this plot at $\theta = 0$ i.e. $e \equiv \tilde{e}(r,|\mathbf{z}|,0)$. For a single controllable input system equation 8.3 can be expressed as a function of state as;

$$\tilde{e} = (r,\mathbf{z},\mathbf{B}) = sign(\mathbf{B^T z})real\left(\left(r^2 - \mathbf{z^T z} + \frac{\mathbf{B^T z B^T z}}{\mathbf{B^T B}}\right)^{\frac{1}{2}}\right) - \frac{\mathbf{B^T z}}{|\mathbf{B}|} \tag{8.4}$$

In a case where multiple inputs are available equation 8.4 could be used to obtain a $\tilde{e}$ value for each available input, which would allow for multiple actuators to be employed

in energy signal tracking without the risk of negative work.

8.3 Alternative error robust feedback control

In the previous section distinctions between measurable error e, control vector error $\hat{e}$ and minimum best case control vector error $\tilde{e}$ were made and it was demonstrated that for a radial state space control objective the relationship between state $\mathbf{z}$ and $\tilde{e}$ is the non-continuous nonlinear dependency given in equation 8.4. Here the feedback of $\tilde{e}$ will be considered for the control of energy in a linear state space systems with unstructured uncertainty.

For a linear time invariant (LTI) differential inclusion, representing a mechanical system with unstructured uncertainty defined as,

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{B_1}\mathbf{w} + \mathbf{B_2}\mathbf{u} \tag{8.5}$$

Where $\mathbf{w}$ and $\mathbf{u}$ represent uncertain and controllable input signals respectively, $\mathbf{B_1}$ and $\mathbf{B_2}$ represent the uncontrollable and controllable input vectors respectively and the state vector $\mathbf{z}$ has already undergone a transformation as described in the previous section. The energy equation is given as a quadratic relation to state and is given by,

$$E = \mathbf{z^T z} \tag{8.6}$$

The objective region is $E = r^2$ which is equivalent to $|\mathbf{z}| = r$. However the available control actions are restricted to acting along the controllable input vector $\mathbf{B_2}$. The best case control vector error $\tilde{e}$ can then be calculated as a function of state with equation 8.4. Separating the linear and nonlinear components of equation 8.4 as,

$$\tilde{e}(r,\mathbf{z},\mathbf{B}) = \tilde{e}_{nl} + \tilde{e}_l \equiv sign(\mathbf{B^T z})real\left(\left(r^2 - \mathbf{z^T z} + \frac{\mathbf{B^T z B^T z}}{\mathbf{B^T B}}\right)^{\frac{1}{2}}\right) - \frac{\mathbf{B^T z}}{|\mathbf{B}|} \tag{8.7}$$

where,

$$\tilde{e}_{nl} = sign(\mathbf{B^T z}) real\left(\left(r^2 - \mathbf{z^T z} + \frac{\mathbf{B^T z B^T z}}{\mathbf{B^T B}}\right)^{\frac{1}{2}}\right) \tag{8.8}$$

and,

$$\tilde{e}_l = -\frac{\mathbf{B^T z}}{|\mathbf{B}|} \tag{8.9}$$

It can be observed that the nonlinear component lies in the region $r \leq \tilde{e}_{nl} \leq r$ and so can be expressed, $\tilde{e}_{nl} = r\delta s.t \delta \in R\{-1 \leq \delta \leq 1\}$. As $\tilde{e}_{nl}$ is bounded and when $|\mathbf{z}| \gg r$ only has a value if $\theta \approx 0$, then $|\mathbf{z}| \gg r$ implies $\tilde{e}_l \gg \tilde{e}_{nl}$. The nonlinear component $\tilde{e}_{nl}$ can thus be treated as a linearly independent bounded disturbance. Robust control of energy requires that there exists no possible disturbance that can cause the system energy to become unbounded. The square root of the average power of a time varying signal $\mathbf{u}(t)$ is denoted $pow(\mathbf{u}(t))$ and given by the function,

$$pow(\mathbf{u}(t)) = \lim_{T\to\infty}\left(\frac{1}{2T}\int_{-T}^{T}\mathbf{u}(t)^T\mathbf{u}(t)\,dt\right)^{\frac{1}{2}} \tag{8.10}$$

A time varying signal $\mathbf{u}(t)$ is described as a power signal if and only if the average power is bounded i.e. $pow(\mathbf{u}(t)) < \infty$. Robust control of the total system energy is equivalent to requiring that the signal of energy storing states $\mathbf{z}(t)$ is a power signal, i.e. $pow(\mathbf{z}(t)) < \infty$ for all possible exogenous input signals $\mathbf{w}(t)$.

If all exogenous inputs are mechanical disturbances it can be safely assumed that they will be finite power events, therefore it is known that $pow(\mathbf{w}(t)) < \infty$. For a linear time invariant system, $\mathbf{y}(t) = \mathbf{G}\mathbf{u}(t)$ the power signals are related by the system H_∞ norm i.e. $pow(\mathbf{y}(t)) = ||\mathbf{G}||_\infty pow(\mathbf{u}(t))$. The goal is then to find a controller K such that the closed loop system between exogenous inputs and the state vector $\mathbf{G_{z,w}}$ is H_∞ bounded. Although there are a number of well known methods (Doyle et al. 1992; Basar and Bernhard 2008; Skogestad and Postlethwaite 2007) for finding state

feedback controllers satisfying the closed loop H_∞ control objective. For simplicity in this study a 2^{nd} order realisable controller will be sought. Supposing K is a realisable LTI,

$$\dot{\mathbf{x}}_\mathbf{k} = \mathbf{A}_\mathbf{k}\mathbf{x}_\mathbf{k} + \mathbf{B}_\mathbf{k}\tilde{e} \tag{8.11}$$

Combining equations 4.14, 8.4 and 8.11 the closed loop system with $\tilde{e}$ feedback will have the form,

$$\begin{bmatrix} \dot{\mathbf{x}}_\mathbf{k} \\ \dot{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_\mathbf{k} & -\mathbf{B}_\mathbf{k}\mathbf{B}_\mathbf{2}^\mathbf{T}/|\mathbf{B_2}| \\ \mathbf{B_2}\mathbf{C}_\mathbf{k} & \mathbf{A} - \mathbf{B_2}\mathbf{D}_\mathbf{k}\mathbf{B}_\mathbf{2}^\mathbf{T}/|\mathbf{B_2}| \end{bmatrix} \begin{bmatrix} \mathbf{x}_\mathbf{k} \\ \mathbf{z} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_\mathbf{k} \\ \mathbf{B_2}\mathbf{D}_\mathbf{k} \end{bmatrix} \tilde{e}_{nl} + \begin{bmatrix} \mathbf{0} \\ \mathbf{B_1} \end{bmatrix} \mathbf{w} \tag{8.12}$$

which can be expressed as the sum of two LTIs,

$$\mathbf{z} = \begin{bmatrix} \mathbf{G}_{\mathbf{z},\tilde{e}_{nl}} & \mathbf{G}_{\mathbf{z},\mathbf{w}} \end{bmatrix} \begin{bmatrix} \tilde{e}_{nl} \\ \mathbf{w} \end{bmatrix} \tag{8.13}$$

As it is known that $|\tilde{e}_{nl}| \leq r \; \forall \mathbf{z} \in R_n$ it must necessarily be true that $pow(\tilde{e}_{nl}(t)) \leq r$, for signal tracking the objective is equivalent to ensuring $pow(\mathbf{z}(t)) = r$, therefore zero error energy tracking can be achieved only if $||\mathbf{G}_{\mathbf{z},\tilde{e}_{nl}}||_\infty \geq 1$.

8.4 Results

The robotic fish model was once again simulated with the same parameters used in chapter 6. The state energy reference signal of $0.35J$ was selected. The controller was simulated discretely with the control signal updated at intervals of $2 \times 10^{-4}s$.

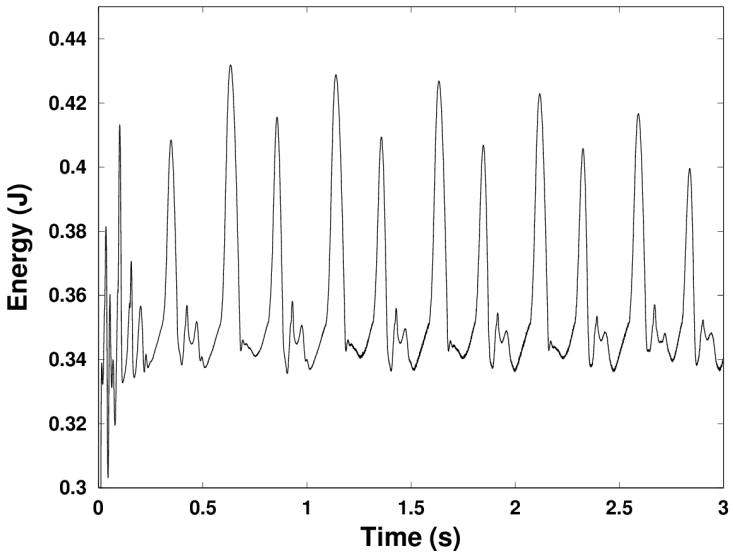


Figure 8.5: Plot of system energy against time

The BCCVE signal was fed into the control system defined by,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -17.123 & 2.709 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tilde{e}$$

$$\mathbf{u} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 87.847\tilde{e} \tag{8.14}$$

when combined with the robotic fish system this gives a closed loop infinity norm $||\mathbf{G}_{\mathbf{z},\mathbf{w}}||_\infty \leq 7.9$. It is noteworthy that the controller features a large direct proportional feedback element.

Figure 8.5 shows a plot of total system energy against time. The controller achieved the state energy control objective with *root mean square error* (RMSE) of $\approx 0.026356J$.

The input signal generated by the controller and the corresponding input work signal can be seen in figures 8.6 and 8.7 respectively. It can be seen that the required input torque did not exceed $10Nm$, and aside from a high initial work rate, once the periodic

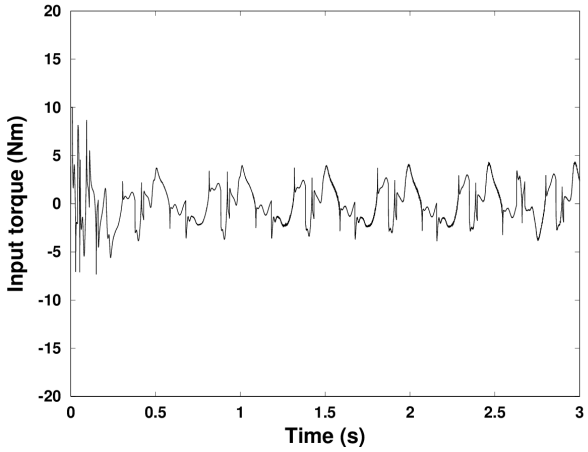


Figure 8.6: Plot of controller input against time

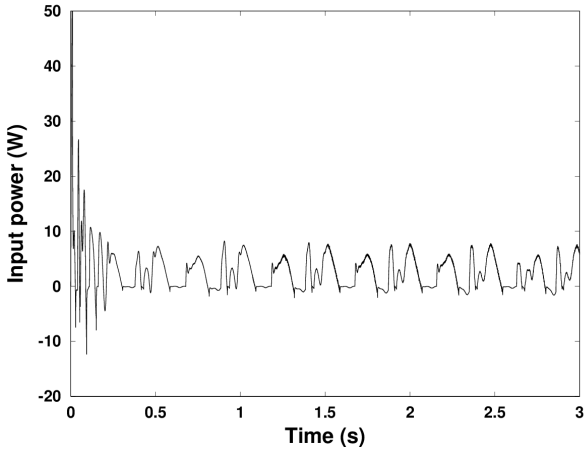


Figure 8.7: Plot of controller work against time

motion was established peak work was no more than $10W$. The mean absolute power input requirement was $\approx 2.98W$ and the mean negative work was $\approx 0.17W$. This constituted $\approx 5.7\%$ of the total absolute work.

The resultant motion like that experienced with the deadbeat controller demonstrated yaw instability. The rate of yaw for the selected time interval was $\approx 0.066 rad\ s^{-1}$. Figure 8.8 shows a plot of body bearing against time for the resultant motion, corrected for change in heading. It can be seen that the resultant motion was approximately $2.5Hz$.

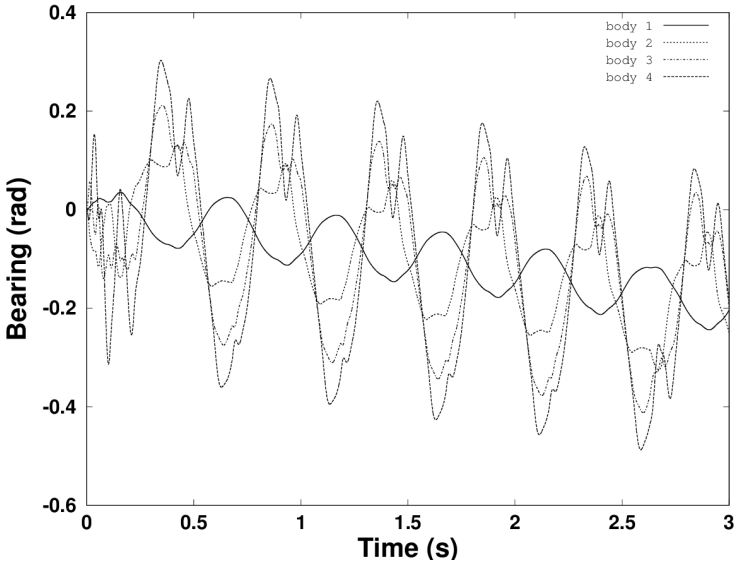


Figure 8.8: Plot of body bearing against time (corrected for global yaw).

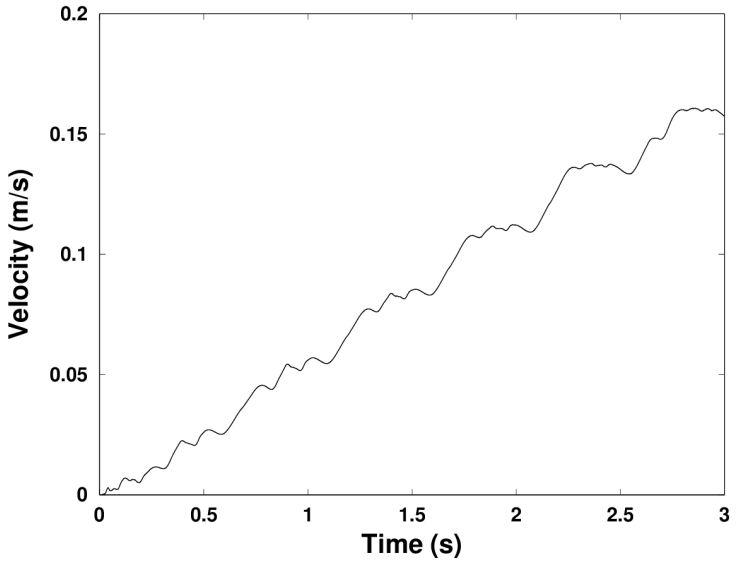
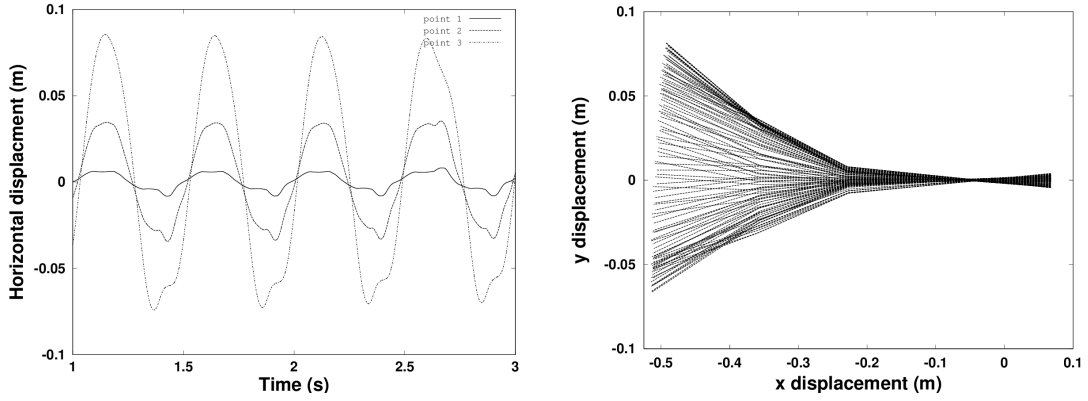


Figure 8.9: Plot of forward velocity against time

Figure 8.9 shows a plot of forward velocity against time. It can be seen that the kinematic resulted in effective forward thrust, resulting in an acceleration from rest of $\approx 0.056 ms^{-2}$.

Figure 8.10 shows plots of the body lateral displacement against time and the body



(a) (b)

Figure 8.10: (a) Plot of body lateral translation against time, (b) Plot of system kinematic

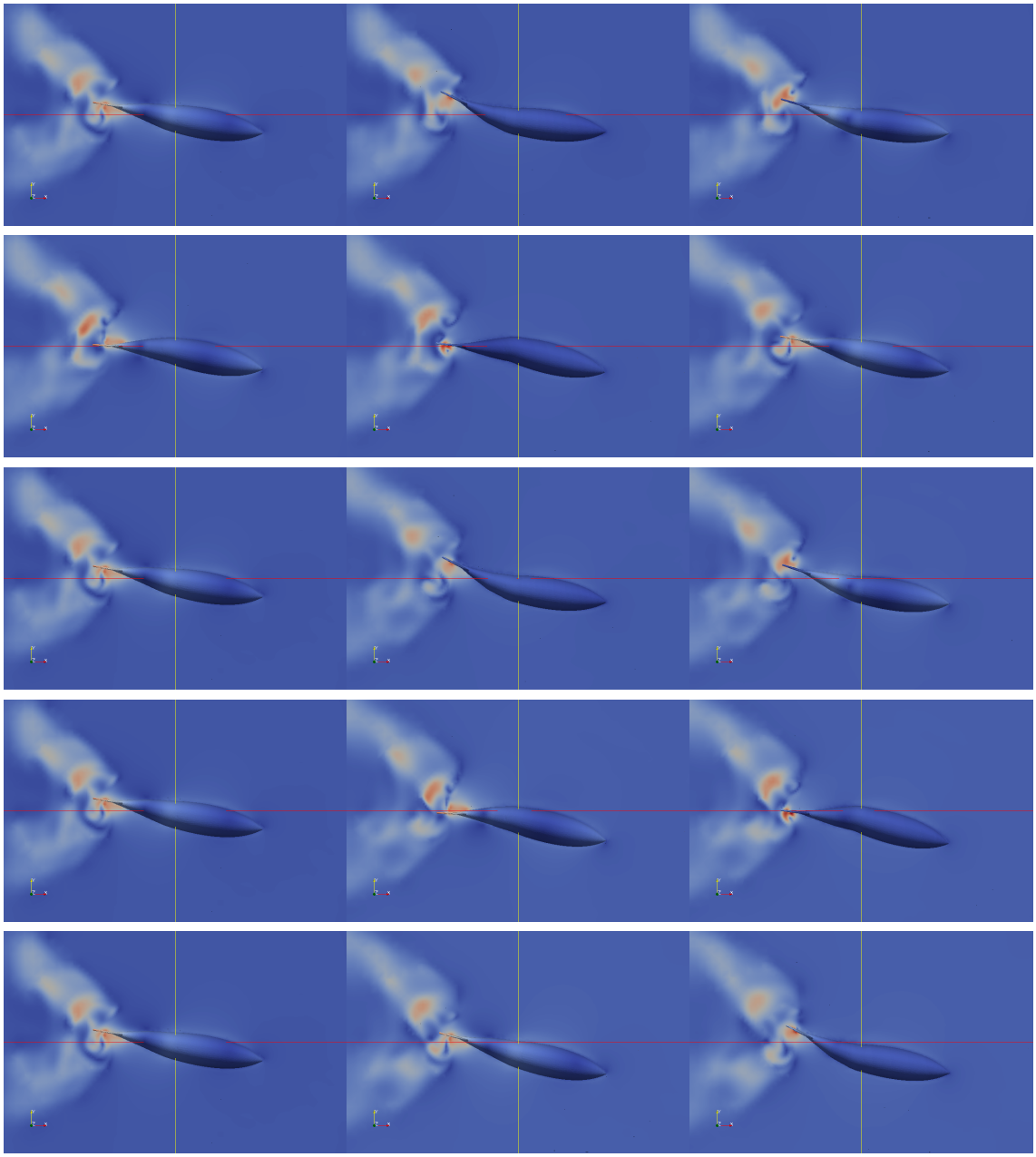
position time series for the resultant motion. It can be seen from figure 8.10 (b) that the motion envelope increases towards the anterior of the body in keeping with the suggested kinematic shape used in Barrett (1996).

Table 8.1 shows the velocity of the fluid around the robotic fish during the motion resultant from the alternative error feedback controller. Vortices can be seen shedding from the tail and travelling down stream in the flow. In keeping with observations of biological fish swimming (Rosen 1959). A video of the resultant motion can be found at (`http://youtu.be/jwAHzje-CqI`).

8.5 Conclusions

In this chapter a novel alternative error metric was defined for use with nonlinear state space objectives. It was argued that this error metric herein referred to as *best case control vector error* (BCCVE) forms a more logical choice for error feedback for control towards non-linear objective regions within the state space as it maintains a more constant response to control inputs across the state space than more traditional measured error metric.

Table 8.1: Velocity of fluid surrounding the robotic fish



Furthermore an explicit equation of state was derived for this metric for state energy control objectives. It was then demonstrated that robust control can be achieved using the proposed BCCVE feedback, provided that the closed loop system around the linear component of the error function was robustly stable with respect to both exogenous

inputs and nonlinear components of the error metric function.

The results demonstrated that the resultant controller bounded the system to the objective region more tightly than the robust state feedback controller from chapter 7. Without stimulating the high frequency transient like the deadbeat controller from chapter 6.

Furthermore there was a significant reduction in negative work from the deadbeat controller, suggesting improved efficiency.

The presence of yaw instability suggests that BCCVE feedback should be used in conjunction with an ancillary yaw stabilisation. The lack of a decision element in the control algorithm as featured in both the deadbeat and robust state feedback controller precludes the integration of decision based yaw stabilisation as used in chapter 7.

Chapter 9

Analysis of results

This chapter will analyse the results presented throughout this thesis with a goal of identifying the best choice of controller for energy based gait generation for a robotic fish.

9.1 Introduction

In chapter 4 a simple robotic fish was modelled as a free floating kinematic chain. Fluid solid interactions were interoperated into the model via a finite element *computational fluid dynamic* (CFD) simulation coupled with the *partial differential equation* (PDE) solver used to resolve the solid body dynamics. In chapter 5 a novel mechanism for gait generation over the simulated system was proposed, where the motion would be determined by a combination of system dynamics and the control of a state energy equation rather than explicit geometric control of motion. Whilst chapters 6, 7 and 8 presented three different state energy controllers. Each with distinct benefits and weaknesses.

This chapter aims to analyse the results presented within this thesis. The controllers will be compared against key performance criteria. With a goal of selecting the best controller for energy based gait generation for an underactuated robotic fish.

The remainder of this chapter is divided into three further sections. Section 9.2 discusses the limitations of the simulation study. Section 9.3 will compares and

contrast results from the previous three chapters, discussing possible causes and potential measures for improvement. Finally section 9.4 presents some concluding remarks based on the analysis presented herein.

9.2 Limitations of simulation study

The simulations study described in chapter 4 included a three dimensional CFD model coupled with a solver for the PDE's representing the solid mechanics of the robotic fish.

Limitations on available computing power and memory restricted the maximum spatial and time resolution of the simulations. The explicit coupling between the fluid and structure models also imposes limitations on the accuracy of the resultant model outputs. However it is assumed that the model produces an approximation of the expected dynamics of the system. No physical model validation has been carried out to verify the accuracy of the model.

Although the CFD component of the model resolves the flow around the body in three dimensions, symmetry was forced across the central horizontal plane. This artificially restricted the fluid induced torques acting on the bodies to yaw inducing torques. As a consequence no comment can be made from this study regarding the roll or pitch stability of the proposed approach.

The total time frame of each simulation was limited by a combination of the temporal resolution, and the maximum deformation allowable on the CFD mesh. Therefore it was decided that the simulations would focus on initial motion from rest. This proved that the motions generated could accelerate the robotic fish from static. Owing to the time frame of the sample only approximate comment can be made on what the maximum forward swimming speed would have been.

Although the methods presented in chapters 7 and 8 were continuous methods due to the constraints of computerised simulation all controllers within the thesis were

implemented discretely. The control interval was kept the same for all simulations making the comparisons drawn in this chapter like for like.

9.3 Comparison of controller performance

Within this thesis four different controllers were used to regulate the state energy of the robotic fish in simulations. The system response and resultant robotic fish behaviour differed significantly depending on the choice of controller.

In chapter 6 a deadbeat controller was put forward which at each iteration solved the discrete time energy equation to find the exact input $\mathbf{u}$ to move to a zero error state from the existing state. This deadbeat controller ignored the presence of system disturbances thus made no allowance to compensate for them at each time. However as deadbeat control does not allow the propagation of existing error it can therefore could be considered universally robust.

In chapter 7 popular robust control techniques were adapted to generate an H_∞ noise rejecting error feedback controller for the control of state energy. This controller was then further optimised using a novel parametric sensitivity of norm based gradient decent to produce a reduced fragility disturbance rejection error feedback controller, with the aim of reducing the effect of parametric uncertainty on the system response to disturbance.

Finally in chapter 8 an alternative error metric controller was defined, which used a novel error metric definition to ensure that the response of error to a given input remained constant despite nonlinearity of the objective region.

Simulations were run for a period of $3s$ with the exception on the deadbeat controller where reduced initial velocity allowed the simulation to run for $6s$.The controllers were each allowed to provide a single control input to be updated at intervals of $2 \times 10^{-4}s$. All the controllers were given identical control objectives, i.e. to regulate the state

energy given by $E = \mathbf{x}^{\mathbf{T}}\mathbf{Q}\mathbf{x}$ to $0.35J$, where the matrix $\mathbf{Q}$ was kept constant. The passive objective of the controllers was to produce an effective/efficient swimming gait for the robotic fish.

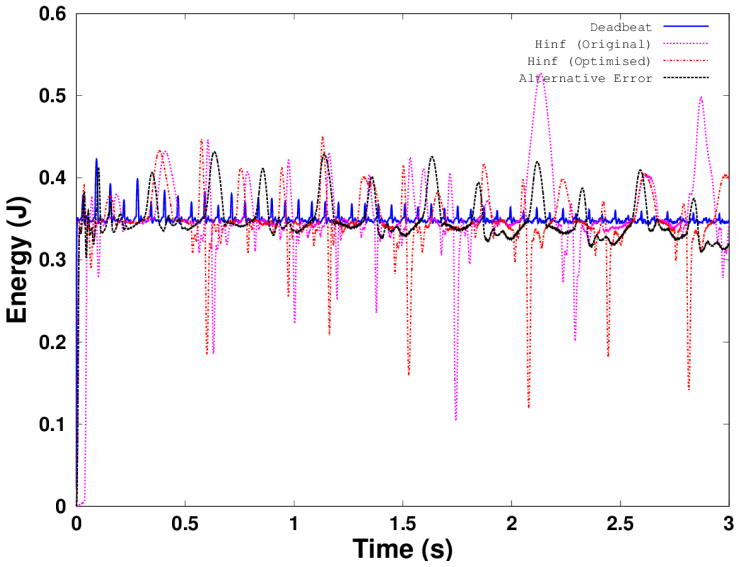


Figure 9.1: Plot comparing energy against time for candidate controllers over 3 seconds

Comparing the controllers on performance the stated control objective reveals that the deadbeat controller was the most effective regulator of energy returning a *root mean square error* (RMSE) of $\approx 0.0095J$. The alternative error metric controller performed second best with RMSE $\approx 0.0263J$. The original H_∞ disturbance rejection error feedback controller and the gradient optimised H_∞ disturbance rejection error feedback controllers performed worst with RMSE of ≈ 0.0579 and ≈ 0.0411 respectively. The poor performance of the original H_∞ and the gradient optimised H_∞ controllers was perhaps largely due to periodic disturbance caused by the nonlinearity of the control objective region which causes variation in the error response to a given input. Figure 9.1 shows a plot of state energy against time for the four controllers. It can clearly be seen that the dead band for the energy response to the deadbeat controller is significantly smaller than the dead band for the other controllers. The two H_∞

controllers have the widest dead band.

Figure 9.2 shows a close-up of the initial rise of each of the energy signals from rest. It can be seen that with the exception of the deadbeat controller all the other controllers result in approximately the same initial ramp up. The alternative energy controller however falls short of the objective at the first peak.

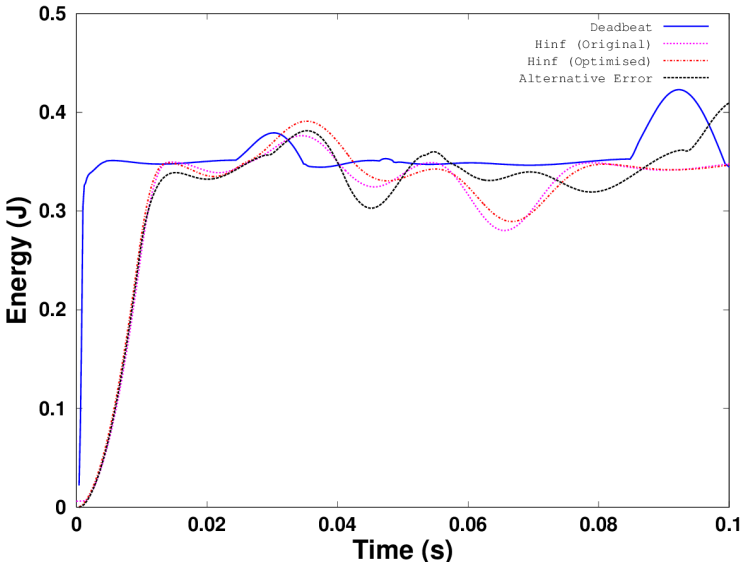


Figure 9.2: Plot comparing energy against time for candidate controllers from rest

Despite the excellent performance of the deadbeat controller on the stated objective it arguably performed worst on the passive objective. By introducing high frequency stimulation into the system through 'non smooth' control. High frequency transients which would have otherwise dissipated quickly were maintained in the motion. As a result the orbital path followed by the deadbeat controller for the first 4 seconds required a mean energy expenditure of $\approx 10W$ significantly more than the orbital paths followed by the other controllers. The original and the optimised H_∞ required an average of $\approx 3.5W$ and $\approx 0.355W$ respectively. The alternative error controller required only $2.98W$. After a period of $4s$ the deadbeat controller did however eventually allow the higher frequency transient to dissipate, after which the controller followed a similar orbital

path to the other controllers with a mean energy requirement of only $2.56W$. This lower energy cost being perhaps due to the remaining presence of turbulence in the surrounding fluid from the earlier high frequency motion.

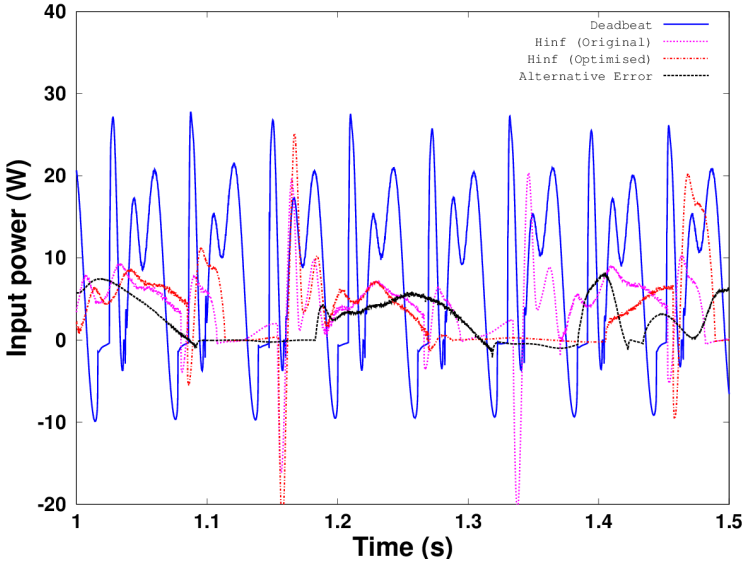


Figure 9.3: Plot comparing work against time for candidate controllers

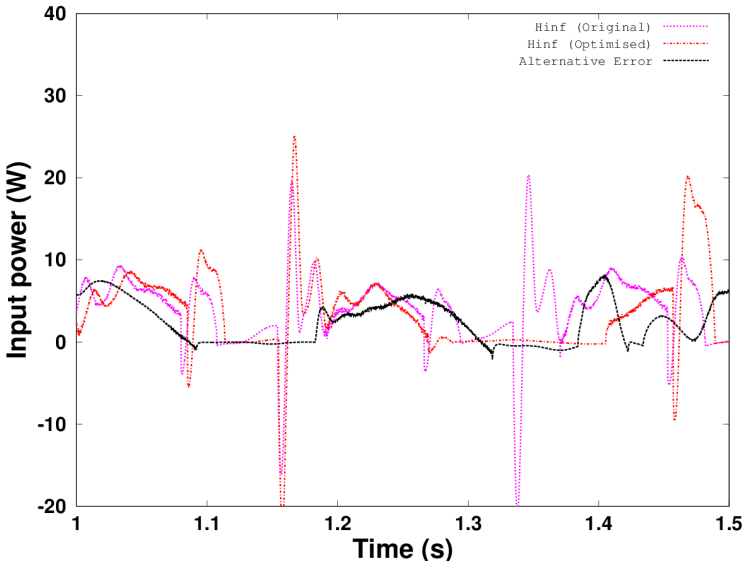


Figure 9.4: Plot comparing work against time for candidate controllers (without deadbeat)

Figures 9.3 and 9.4 show plots of the input power of each of the controllers against

time for a selected time period. It can be seen that the alternative error controller power series lacks the positive and negative spikes characteristic in the power series from the H_∞ controllers.

The avoidance of negative work was one of the motivations stated for using physical energy to regulate the motion. However simulation results revealed that choice of controller had a significant effect on how successfully the system avoided negative work. The initial orbital path followed by the deadbeat controller resulted in an average of $\approx 0.984W$ of negative work, constituting $\approx 10\%$ of the total absolute work input of the controller. After the switch in orbital path this dropped to $\approx 0.43W$ of negative work however this now constituted $\approx 15\%$ of the mean absolute work. The best performing controller in terms of eradicating negative work was the alternative error metric controller, which resulted in $\approx 0.17W$ of mean negative work, which constituted 5.7% of the total work. The original and optimised H_∞ controller resulted in mean negative work of $\approx 0.37W$ and $\approx 0.33W$ respectively constituting $\approx 12\%$ and $\approx 10\%$ negative work respectively. The high level of negative work in the deadbeat controller was likely due to overshoot caused by modelling inaccuracy. On the other hand the negative work from the H_∞ controller was likely originated in overshoots due to the inconsistency of the error response to a given response. The reduced fragility H_∞ controller performed marginally better than the original H_∞ controller in this respect.

Figure 9.5 shows a plot of the bearing of the robotic fish head for each of the simulations. It can clearly be seen that the alternative error feedback results in significantly increased rates of yaw. It is interesting to note also that it is the only one of the four controllers to initiate with positive yaw oscillation, yet still results in yaw in the same direction. This could suggest that the mesh chosen has a tendency to cause negative yaw motion.

The yaw stability was also identified as a significant issue. For forward swimming it is

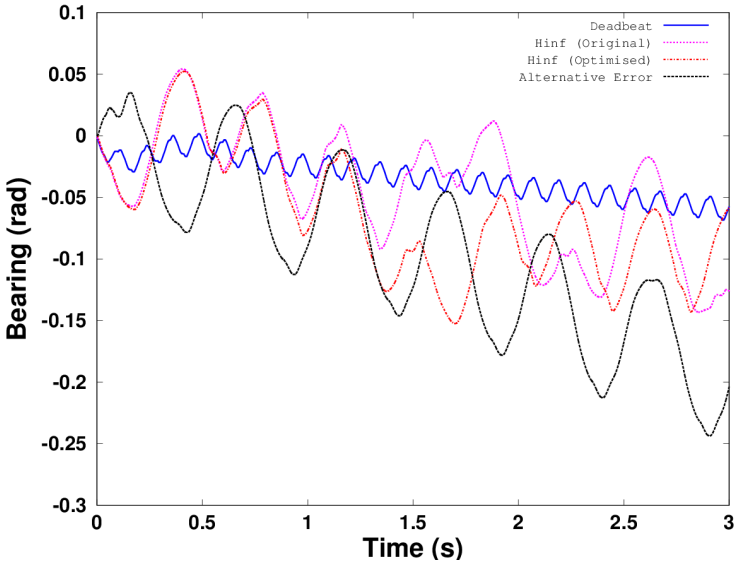


Figure 9.5: Plot comparing forward section bearing against time for candidate controllers

desirable to ensure the global yaw is kept to a minimum. However all the controllers to some degree experienced global yaw in the resultant motion. The largest yaw rate was experienced by the alternative error controller which experienced a global yaw rate of $0.067 rad\ s^{-1}$.The deadbeat controller resulted in an initial yaw rate of $0.021 rad\ s^{-1}$ during the initial motion, increasing to $0.026 rad\ s^{-1}$ after the switch of orbital path. In order to try to address the yaw issue the H_∞ controllers included a direction bias in the decision element of the controller. The original H_∞ controller resulted in a yaw rate of $0.029 rad$ however after experiencing an initial yaw rate of $0.02 rad\ s^{-1}$ the optimised H_∞ controller seemed to stabilise in yaw due to the directional bias. The yaw rate is thought to be determined by a combination of the yaw stability and the initial yaw inputs generated when establishing the periodic motion.

Figure 9.6 shows a plot of forward velocity against time for the four controllers. As all three controllers resulted in motion that caused forward locomotion all three controllers could be said to have produced 'effective' swimming gaits. However

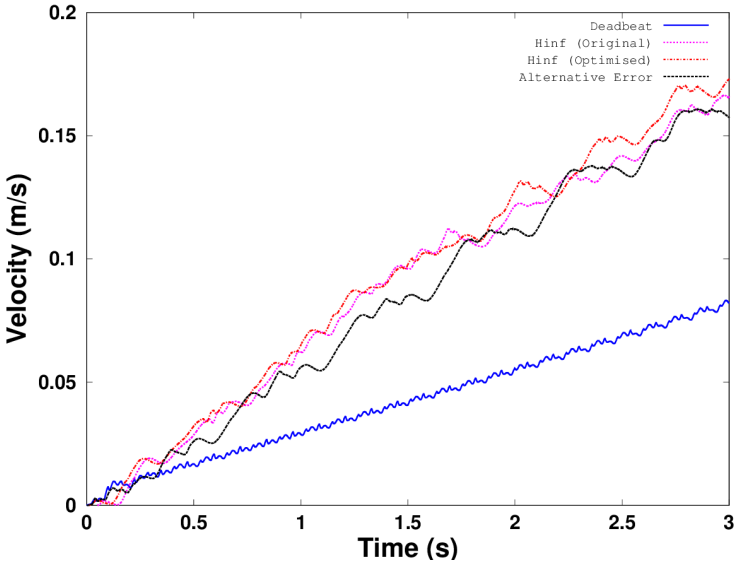


Figure 9.6: Plot comparing forward velocity against time for candidate controllers

comparing the velocities $3s$ into the simulation reveals that the choice of controller had significant effect on the resultant swimming speed. The fastest was with the optimised H_∞ controller reaching $0.173ms^{-1}$ followed by the original H_∞ controller which reached $0.170ms^{-1}$ and the alternative error metric controller reaching $0.157ms^{-1}$ and finally the deadbeat controller reaching $0.0825ms^{-1}$. However correcting for direction of heading gives, $0.188ms^{-1}$ for the modified H_∞, $0.1836ms^{-1}$ for the alternative error, $0.171ms^{-1}$ for the original H_∞ and $0.0825ms^{-1}$ for the deadbeat.

Figure 9.7 shows a plot of heading corrected velocity. It can be seen that whilst the H_∞ controllers seem to initially have resulted in a faster swimming motion, after around $1.5s$ the alternative error controller catches up. This is perhaps partially due to interference from the course correction element integrated into the H_∞ controllers causing a less optimal gait.

Comparing cost of transport over the first 3 seconds, given by,

$$cost\ of\ transport = \frac{\int_0^3 |work|\ dt}{\int_0^3 velocity\ dt} \tag{9.1}$$

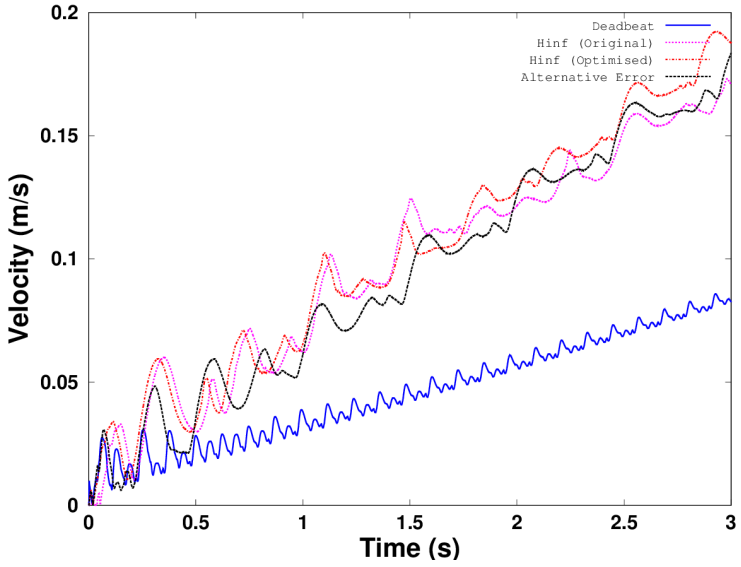


Figure 9.7: Plot comparing velocity corrected for heading against time for candidate controllers

unsurprisingly revealed that the deadbeat controller performed worst $\approx 222Jm^{-1}$. The original and modified H_∞ controllers resulted in a cost of transport of $\approx 37.8Jm^{1}$ and $\approx 38.1Jm^{-1}$. The alternative error metric controller performed best with $\approx 31.89Jm^{-1}$. However it must be observed that the H_∞ controllers may have performed better in the absence of the directional bias in the controller.

Figure 9.8 Compares the resultant vortices from the four controllers. It can be seen that the deadbeat controller resulted in much less well defined vortices. This could be due to the high frequency input from the discrete controller disrupting vortex formation.

Based on the rate of acceleration experienced by the fish during the 3 seconds simulated and the steady state coefficient of drag of the geometry given in chapter 4 as $c_d \approx 0.04468$. It is possible to project the final velocity that the fish would have reached before drag forces balanced with thrust would have been $\approx 0.95ms^{-1}$.

Figure 9.9 shows how this projected velocity fits in with the reported BCF swimmers in chapter 3. It can be seen that the projected velocities lie in good correlation with the

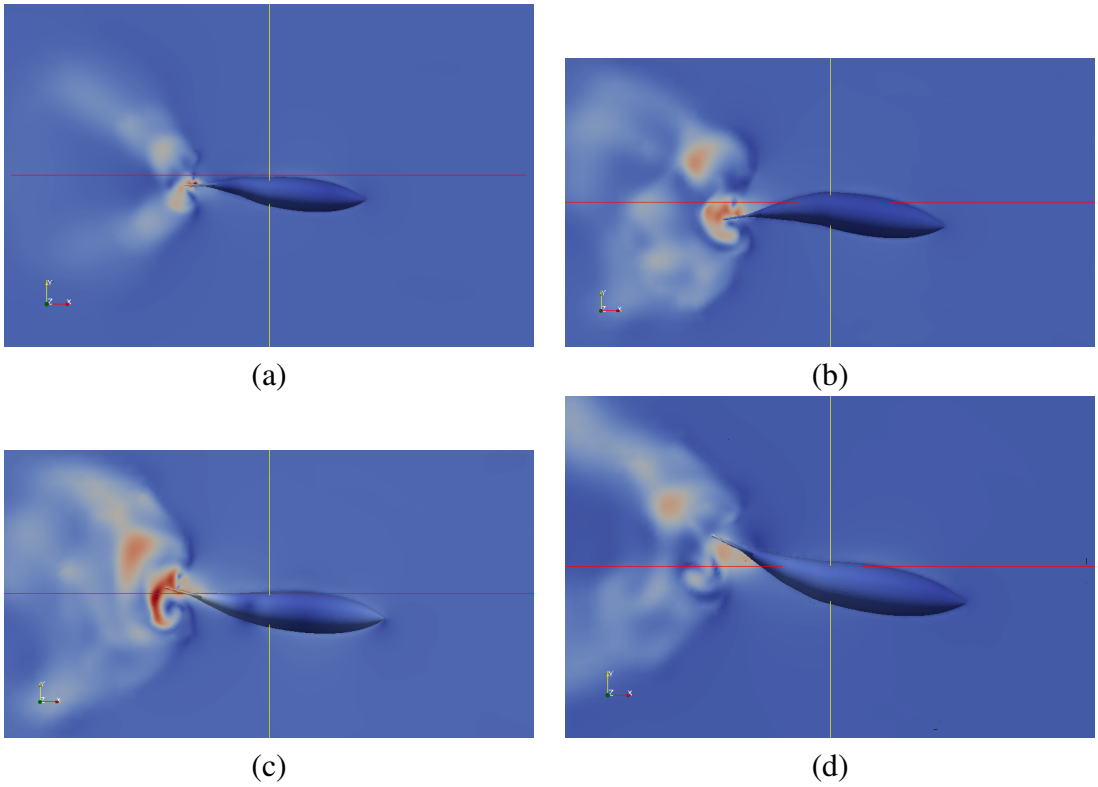


Figure 9.8: A comparison of resultant vortices (a) Deadbeat controller, (b) H_∞ (original), (c) H_∞ (optimised), (d) alternative error

more sucessful robotic BCF swimmers reported. Furthermore the projections suggest that the swimming gait is significantly more optimal than those achieved by other underactuated devices.

A summary of the results presented within this thesis can be found in table 9.1.

9.4 Concluding Remarks

In this chapter a comparison was made between the simulation results presented in chapters 6, 7 and 8. The comparison demonstrated that the choice of controller has a significant effect not only on the accuracy of the energy control, but also on the resultant orbital path through the state space. This in turn has a significant effect on the fitness for purpose as an energy based gait regulator for a robotic fish.

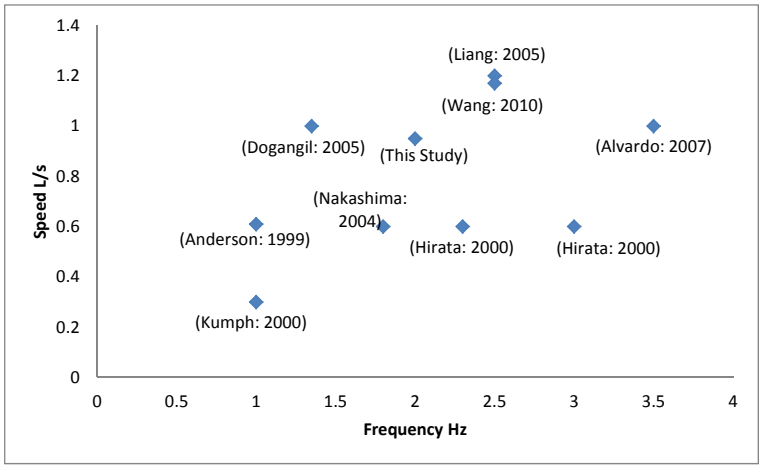


Figure 9.9: A graph showing the relationship between maximum tail beat frequency and resultant speed for BCF swimmers reported in this thesis

Table 9.1: Summary of Controller Performance

	Deadbeat	**H_∞** (original)	**H_∞** (gradient optimised)	**Alternative Error**
$RMSE\ (J)$	0.0095	0.0579	0.0411	0.0263
$\lvert\bar{work}\rvert\ (W)$	9.88/2.56	3.5	3.55	2.98
$\bar{work}^{-ve}\ (W)$	0.984/0.43	0.37	0.33	0.17
$yaw\ (rad\ s^{-1})$	0.021/0.026	0.029	0.02	0.066
$v^*\ (ms^{-1})$	0.0825	0.171	0.188	0.1836
$\bar{a}\ (ms^{-2})$	0.027/0.058	0.054	0.061	0.056
$COT\#\ (Jm^{-1})$	222	37.6	38.2	31.9

** velocity after 3 seconds*
 # Cost of transport

The deadbeat controller offered the most accurate control of state energy, and therefore in terms of the stated control objective function could be described as the most successful controller. However the deadbeat controller performed poorly with regards to the passive objective of generating an efficient effective stable gait for a robotic fish. Whilst the gait produced was effective in generating propulsion the efficiency

was relatively poor and proved to be unstable in terms of fixing the orbit. The high level of negative work and the excitation of undesired transients resulting from the non smooth input make the deadbeat control a poor choice for energy based control of an underactuated robotic fish.

The H_∞ disturbance rejecting error feedback controller, resulted in a more predictable motion, however at the cost of controller accuracy. The sensitivity optimised H_∞ controller, performed better than the original H_∞ disturbance rejection controller, improving both accuracy and reducing the amount of negative work. It is suggested that whilst such a controller may be a good choice for swimming in a lamina flow, the disturbance rejection aspect may prevent the robot fish from benefiting from vortex interactions in a turbulent flow, negating a significant proportion of the original motivation behind the selection of energy based control.

The alternative error metric controller outperformed both H_∞ controllers in terms of the accuracy of the error control, achieving a further reduction in the negative work, whilst avoiding stimulation of undesired transients, this controller suffered the greatest degree of yaw instability, this was perhaps due to the lack of a decision phase where a direction bias could be introduced. It is clear that the alternative error metric controller is the best choice for energy based control of an underactuated robotic fish gait however an ancillary yaw stabilisation such as adjustable dorsal/anal fins as used by fish in nature may be required. A possible alternative considered is the reshaping of the objective region.

The speed projections although somewhat crude are very promising comparing very favourably against other BCF type robotic fish reviewed in the literature, significantly out performing other single actuator underactuated swimmers.

Chapter 10

Conclusions and further work

This chapter presents conclusions drawn form this study and recommendations for future work.

10.1 Summary of thesis and contributions

The stated aim of this study was to develop biomimetic marine propulsion capable of mimicking the way fish and cetaceans in nature harness unsteady fluid effects to increase efficiency. Towards this goal the following contributions have been made,

Chapter 3 presented an extensive review of the available, assessing trends in the development of biomimetic propulsion systems. The review revealed a generalised trend in BCF robotic swimmers towards simpler under actuated devices and identified gait generation as a critical factor in biomimetic propulsion.

Chapter 4 described a simulation platform to enable offline experimentation and optimisation of a simplified BCF type robotic fish. The model was based on a four link free floating kinematic chain surrounded by an incompressible fluid. Fluid interactions were integrated through a 3D finite element CFD simulation. This model was then used in the simulations featured throughout the thesis.

Chapter 5 detailed a novel method for the generation of gaits for underactuated robotic fish based on the control of energy. By treating the gait as a state space orbit, high dimensional periodic motion could be regulated through a single observable output.

Chapter 6 presented a proof of concept study demonstrating that the aforementioned novel gait generation tactic can produce an effective swimming gait for a BCF type robotic fish. A simple model based deadbeat controller was derived to regulate the energy. Results presented demonstated that an energy regulated state space orbit could produce an effective swimming gait for a robotic fish.

Chapter 7 described an improved state energy controllers based on robust control techniques and a novel technique to determine the parametric derivative of system norms. The resultant controller was robust and with non fragile disturbance rejection allowing for significant unmodeled disturbance from unsteady fluid effects and unmodelled dynamics.

Chapter 8 featured a further improved state energy controller based on a novel alternative error metric allowing error feedback linear controllers to more effectively regulate systems with nonlinear objectives. By altering the measure of error to ensure that the response of the system error to a given control input remains constant the number of control problems solvable with a simple error feedback controller is vastly increased.

Chapter 9 provided an assessment of the four proposed controllers featured within this thesis with a view to determine fitness for purpose. Findings demonstrated the importance of controller selection in resultant system performance.

Further to these contributions towards the stated aim during the course of this study the following Contributions to Knowledge have been made which it believed may be of wider interest.

A parameter differentiable upper bound on the H_∞ norm of an LTI system was derived. Allowing the local parametric gradient of the H_∞ norm to be approximated. This is thought to be of wider interest to the robust control community for applications such as

the design of nonfragile robust systems with infinite phase margin.

Also an alternative error metric for error feedback control of systems with nonlinear objective region was defined and an explicit function of state to quantify the aforementioned error metric for a state energy control objective was derived. This is though to be of wider interest to the control of linear dynamic systems with nonlinear objectives.

10.2 Concluding remarks

This study has followed a bottom up approach to biomimicry in that rather than copying a swimming gait from nature which could be considered a top down approach. This study has focused on a method to mimic the mechanisms used in nature to produce an effective swimming gait.

10.3 Recommendations for future work

Yaw stabilisation is key to developing the methods described within this thesis to practical applications. Whilst with the H_∞ controllers presented direction bias did go some way towards yaw stabilisation, full control over yaw is desirable. This could possibly be achieved through the reshaping of the objective region to discourage global yaw either through alternative choice of the energy matrix $\mathbf{Q}$ or possibly through the integration of two of more intersecting energy objective functions. It is possible that the best case control vector error definition could be expanded to achieve this to some degree with a single controllable input.

Experimental validation of the simulations presented within this thesis would also help to strengthen the arguments presented herein.

The results presented within this thesis only included acceleration from rest and did not give an accurate indication of maximum speed achievable. Further investigation into

swimming at speeds would be desirable.

It is believed that further optimisation of the spring values and energy objective function could yield more optimal swimming kinematics.

Appendix A

H_∞ controller Parameters

Parameters for H_∞ controller presented in chapter 7

$$
\mathbf{X}_{\infty} = \begin{bmatrix}
2.4e+00 & 1.6e+00 & 1.4e+00 & 1.3e+00 & 1.7e-02 & -4.6e-02 & -1.2e-01 & -1.1e-01 & 1.5e-04 & -1.1e-04 & -3.8e-05 & -1.6e-06 & 7.7e-01 & 1.3e-01 & 4.3e-02 & 1.0e-02 \\
1.6e+00 & 4.7e+00 & 4.9e+00 & 4.6e+00 & -9.7e-02 & 3.0e-01 & 6.6e-01 & 3.5e-01 & -7.4e-04 & 8.2e-04 & 2.1e-05 & -1.2e-04 & 7.4e-01 & 1.6e-01 & 5.8e-02 & 1.5e-02 \\
1.4e+00 & 4.9e+00 & 6.9e+00 & 8.0e+00 & -3.9e-02 & 2.2e-02 & 3.8e-01 & 8.7e-01 & -7.0e-04 & -4.5e-05 & 4.9e-04 & 2.4e-04 & 7.3e-01 & 1.6e-01 & 6.5e-02 & 1.7e-02 \\
1.3e+00 & 4.6e+00 & 8.0e+00 & 1.0e+01 & -4.7e-02 & 2.9e-01 & -1.4e-01 & 3.5e-01 & -5.6e-04 & -7.5e-04 & 6.8e-04 & 6.2e-04 & 7.2e-01 & 1.6e-01 & 6.8e-02 & 1.9e-02 \\
1.7e-02 & -9.7e-02 & -3.9e-02 & -4.7e-02 & 6.6e+01 & -3.4e+02 & -7.4e+01 & 8.4e+01 & -3.1e+00 & 1.9e+00 & 8.6e-01 & 1.7e-01 & 1.2e-03 & -1.4e-03 & -7.1e-04 & -1.9e-04 \\
-4.6e-02 & 3.0e-01 & 2.2e-02 & 2.9e-01 & -3.4e+02 & 2.6e+03 & -1.4e+03 & -2.8e+03 & 1.2e+01 & -1.1e+01 & -1.7e+00 & 7.2e-01 & -2.4e-03 & 4.2e-03 & 2.1e-03 & 5.8e-04 \\
-1.2e-01 & 6.6e-01 & 3.8e-01 & -1.4e-01 & -7.4e+01 & -1.4e+03 & 4.7e+03 & 2.4e+03 & 1.3e+01 & -2.6e+00 & -7.1e+00 & -3.2e+00 & -9.9e-03 & 9.3e-03 & 4.6e-03 & 1.1e-03 \\
-1.1e-01 & 3.5e-01 & 8.7e-01 & 3.5e-01 & 8.4e+01 & -2.8e+03 & 2.4e+03 & 1.9e+04 & 1.2e+01 & 4.3e+00 & -9.4e+00 & -7.2e+00 & -1.1e-02 & 8.3e-03 & 5.7e-03 & 1.8e-03 \\
1.5e-04 & -7.4e-04 & -7.0e-04 & -5.6e-04 & -3.1e+00 & 1.2e+01 & 1.3e+01 & 1.2e+01 & 3.5e-01 & -1.8e-01 & -1.2e-01 & -3.5e-02 & 9.3e-06 & -1.2e-05 & -7.0e-06 & -2.0e-06 \\
-1.1e-04 & 8.2e-04 & -4.5e-05 & -7.5e-04 & 1.9e+00 & -1.1e+01 & -2.6e+00 & 4.3e+00 & -1.8e-01 & 1.2e-01 & 5.2e-02 & 9.3e-03 & -5.9e-06 & 8.7e-06 & 2.8e-06 & 1.4e-07 \\
-3.8e-05 & 2.1e-05 & 4.9e-04 & 6.8e-04 & 8.6e-01 & -1.7e+00 & -7.1e+00 & -9.4e+00 & -1.2e-01 & 5.2e-02 & 4.6e-02 & 1.7e-02 & -4.8e-06 & 2.8e-06 & 2.8e-06 & 1.1e-06 \\
-1.6e-06 & -1.2e-04 & 2.4e-04 & 6.2e-04 & 1.7e-01 & 7.2e-01 & -3.2e+00 & -7.2e+00 & -3.5e-02 & 9.3e-03 & 1.7e-02 & 8.6e-03 & -1.5e-06 & 1.4e-07 & 1.1e-06 & 6.5e-07 \\
7.7e-01 & 7.4e-01 & 7.3e-01 & 7.2e-01 & 1.2e-03 & -2.4e-03 & -9.9e-03 & -1.1e-02 & 9.3e-06 & -5.9e-06 & -4.8e-06 & -1.5e-06 & 2.8e-01 & 4.8e-02 & 1.6e-02 & 3.9e-03 \\
1.3e-01 & 1.6e-01 & 1.6e-01 & 1.6e-01 & -1.4e-03 & 4.2e-03 & 9.3e-03 & 8.3e-03 & -1.2e-05 & 8.7e-06 & 2.8e-06 & 1.4e-07 & 4.8e-02 & 9.0e-03 & 3.2e-03 & 7.8e-04 \\
4.3e-02 & 5.8e-02 & 6.5e-02 & 6.8e-02 & -7.1e-04 & 2.1e-03 & 4.6e-03 & 5.7e-03 & -7.0e-06 & 2.8e-06 & 2.8e-06 & 1.1e-06 & 1.6e-02 & 3.2e-03 & 1.2e-03 & 2.9e-04 \\
1.0e-02 & 1.5e-02 & 1.7e-02 & 1.9e-02 & -1.9e-04 & 5.8e-04 & 1.1e-03 & 1.8e-03 & -2.0e-06 & 1.4e-07 & 1.1e-06 & 6.5e-07 & 3.9e-03 & 7.8e-04 & 2.9e-04 & 7.5e-05
\end{bmatrix}
$$

$$\mathbf{Z}_{\infty}\mathbf{L}_{\infty}\begin{bmatrix}
1.5e-06 & 9.0e-07 & 5.4e-07 & 2.0e-07 & 5.2e-06 & 9.5e-07 & 3.4e-07 & 8.5e-08 \\
1.5e-06 & 9.2e-07 & 5.5e-07 & 2.0e-07 & 5.2e-06 & 9.6e-07 & 3.4e-07 & 8.6e-08 \\
1.6e-06 & 9.2e-07 & 5.5e-07 & 2.0e-07 & 5.2e-06 & 9.6e-07 & 3.4e-07 & 8.6e-08 \\
1.6e-06 & 9.2e-07 & 5.5e-07 & 2.0e-07 & 5.3e-06 & 9.7e-07 & 3.4e-07 & 8.7e-08 \\
-3.8e-06 & -2.3e-06 & -1.6e-06 & -3.8e-07 & 4.8e-06 & 7.6e-07 & 2.2e-07 & 4.5e-08 \\
-3.8e-06 & -2.4e-06 & -1.6e-06 & -3.9e-07 & 4.9e-06 & 7.7e-07 & 2.3e-07 & 4.6e-08 \\
-3.8e-06 & -2.4e-06 & -1.6e-06 & -3.9e-07 & 4.9e-06 & 7.7e-07 & 2.3e-07 & 4.6e-08 \\
-3.8e-06 & -2.4e-06 & -1.6e-06 & -3.9e-07 & 4.9e-06 & 7.7e-07 & 2.3e-07 & 4.6e-08 \\
-2.3e+01 & 2.5e+01 & -6.1e+00 & 2.2e+00 & -2.5e-01 & -6.5e-01 & -1.5e-01 & -2.3e-02 \\
2.5e+01 & -9.5e+01 & 8.7e+01 & -1.8e+01 & 1.9e-01 & 1.8e-01 & -4.6e-01 & -3.0e-02 \\
-6.1e+00 & 8.7e+01 & -1.7e+02 & 8.8e+01 & 9.5e-02 & 3.4e-01 & 2.7e-01 & -3.2e-01 \\
2.2e+00 & -1.8e+01 & 8.8e+01 & -7.3e+01 & 2.9e-02 & 8.9e-02 & 3.1e-01 & 3.7e-01 \\
-2.5e-01 & 1.9e-01 & 9.5e-02 & 2.9e-02 & -1.2e+00 & -1.8e-01 & -5.4e-02 & -1.2e-02 \\
-6.5e-01 & 1.8e-01 & 3.4e-01 & 8.9e-02 & -1.8e-01 & -9.3e-02 & -3.3e-02 & -6.2e-03 \\
-1.5e-01 & -4.6e-01 & 2.7e-01 & 3.1e-01 & -5.4e-02 & -3.3e-02 & -3.1e-02 & -9.2e-03 \\
-2.3e-02 & -3.0e-02 & -3.2e-01 & 3.7e-01 & -1.2e-02 & -6.2e-03 & -9.2e-03 & -8.2e-03
\end{bmatrix}$$

Appendix B

Simmulation Study

This appendix contains details of the specific hardware and software used for all simulation studies featured throughout this the thesis.

Table B.1: Hardware

model	Dell Latitude E6220
processor	Intel core i3 2.1GHz
ram	4GB

Table B.2: Software

Operating System	Ubuntu 12.04.02
CFD	OpenFoam version 2.0.1
ODE	Octave version 3.6.2

B.1 OpenFoam Case files

The following section gives the Octave case files used in verbatim form. Approximate runtime for simulation on the above described hardware was $2 days\ s^{-1}$.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
application     snappyHexMesh;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         100;

deltaT          1;

writeControl    runTime;

writeInterval   1;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

}

// ************************************************************************* //


/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      snappyHexMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Which of the steps to run
castellatedMesh true;
snap            true;
addLayers       true;


// Geometry. Definition of all surfaces. All surfaces are of class
// searchableSurface.
// Surfaces are used
// - to specify refinement for any mesh cell intersecting it
// - to specify refinement for any mesh cell inside/outside/near
// - to 'snap' the mesh boundary to the surface
geometry
{

Fish.stl
{
type triSurfaceMesh;
name Fish;
}


//    Head.stl
 //   {
  //      type triSurfaceMesh;
   //     name Head;
   // }
// Middle.stl
 //   {
  //      type triSurfaceMesh;
   //     name Middle;
   // }
  // Tail.stl
  //  {
  //        type triSurfaceMesh;
//   name Tail;
 //   }

};
```

```
// Settings for the castellatedMesh generation.
castellatedMeshControls
{

    // Refinement parameters
    // ~~~~~~~~~~~~~~~~~~~~~

    // If local number of cells is >= maxLocalCells on any processor
    // switches from from refinement followed by balancing
    // (current method) to (weighted) balancing before refinement.
    maxLocalCells 3000000;

    // Overall cell limit (approximately). Refinement will stop immediately
    // upon reaching this number so a refinement level might not complete.
    // Note that this is the number of cells before removing the part which
    // is not 'visible' from the keepPoint. The final number of cells might
    // actually be a lot less.
    maxGlobalCells 3000000;

    // The surface refinement loop might spend lots of iterations refining just a
    // few cells. This setting will cause refinement to stop if <= minimumRefine
    // are selected for refinement. Note: it will at least do one iteration
    // (unless the number of cells to refine is 0)
    minRefinementCells 10;

    // Allow a certain level of imbalance during refining
    // (since balancing is quite expensive)
    // Expressed as fraction of perfect balance (= overall number of cells /
    // nProcs). 0=balance always.
    maxLoadUnbalance 0.10;


    // Number of buffer layers between different levels.
    // 1 means normal 2:1 refinement restriction, larger means slower
    // refinement.
    nCellsBetweenLevels 3;



    // Explicit feature edge refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    // Specifies a level for any cell intersected by its edges.
    // This is a featureEdgeMesh, read from constant/triSurface for now.
    features
    (
        //{
        //    file "someLine.eMesh";
        //    level 2;
        //}
    );



    // Surface based refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~

    // Specifies two levels for every surface. The first is the minimum level,
    // every cell intersecting a surface gets refined up to the minimum level.
    // The second level is the maximum level. Cells that 'see' multiple
    // intersections where the intersections make an
    // angle > resolveFeatureAngle get refined up to the maximum level.

    refinementSurfaces
    {
Fish
{
level (5 5);
}
 //       Head
 //       {
           // Surface-wise min and max refinement level
 //          level (4 4);
  //      }
// Middle
// {
//  level (4 4);
// }
// Tail
// {
//  level (4 4);
// }
    }

    // Resolve sharp angles
    resolveFeatureAngle 30;
```

```
    // Region-wise refinement
    // ~~~~~~~~~~~~~~~~~~~~~

    // Specifies refinement level for cells in relation to a surface. One of
    // three modes
    // - distance. 'levels' specifies per distance to the surface the
    //   wanted refinement level. The distances need to be specified in
    //   descending order.
    // - inside. 'levels' is only one entry and only the level is used. All
    //   cells inside the surface get refined up to the level. The surface
    //   needs to be closed for this to be possible.
    // - outside. Same but cells outside.

    refinementRegions
    {
Fish
{
mode distance;
// Hress
//levels 4((0.15 5)(0.3 4)(0.6 3)(1.2 2));
//LRes
levels 3((0.2 4)(0.5 3)(1 2));
}

//       Head
 //       {
 //           mode distance;
  //          levels ((0.5 3));
   //     }
// Middle
 //       {
  //          mode distance;
   //         levels ((0.5 3));
    //    }
// Tail
  //      {
   //         mode distance;
    //        levels ((0.5 3));
     //   }
    }


    // Mesh selection
    // ~~~~~~~~~~~~~~

    // After refinement patches get added for all refinementSurfaces and
    // all cells intersecting the surfaces get put into these patches. The
    // section reachable from the locationInMesh is kept.
    // NOTE: This point should never be on a face, always inside a cell, even
    // after refinement.
    locationInMesh (1.11 0.01 0.23);

 allowFreeStandingZoneFaces true;

}



// Settings for the snapping.
snapControls
{
    //- Number of patch smoothing iterations before finding correspondence
    //  to surface
    nSmoothPatch 3;

    //- Relative distance for points to be attracted by surface feature point
    //  or edge. True distance is this factor times local
    //  maximum edge length.
    tolerance 4.0;

    //- Number of mesh displacement relaxation iterations.
    nSolveIter 30;

    //- Maximum number of snapping relaxation iterations. Should stop
    //  before upon reaching a correct mesh.
    nRelaxIter 5;
}



// Settings for the layer addition.
addLayersControls
{
    // Are the thickness parameters below relative to the undistorted
    // size of the refined cell outside layer (true) or absolute sizes (false).
    relativeSizes true;

    // Per final patch (so not geometry!) the layer information
    layers
```

```
    {

        Fish_Link_1
        {
            nSurfaceLayers 1;
        }
Fish_Link_2
        {
            nSurfaceLayers 1;
        }
Fish_Link_3
{
    nSurfaceLayers 1;
}
Fish_Link_4
{
   nSurfaceLayers  1;
}
Fish_Joint_1
{
nSurfaceLayers 1;
}
Fish_Joint_2
{
nSurfaceLayers 1;
}
Fish_Joint_3
{
nSurfaceLayers 1;
}
    }

    // Expansion factor for layer mesh
    expansionRatio 1.0;

    //- Wanted thickness of final added cell layer. If multiple layers
    //  is the thickness of the layer furthest away from the wall.
    //  See relativeSizes parameter.
    finalLayerThickness 0.3;

    //- Minimum thickness of cell layer. If for any reason layer
    //  cannot be above minThickness do not add layer.
    //  Relative to undistorted size of cell outside layer.
    minThickness 0.1;

    //- If points get not extruded do nGrow layers of connected faces that are
    //  also not grown. This helps convergence of the layer addition process
    //  close to features.
    nGrow 1;


    // Advanced settings

    //- When not to extrude surface. 0 is flat surface, 90 is when two faces
    //  make straight angle.
    featureAngle 30;

    //- Maximum number of snapping relaxation iterations. Should stop
    //  before upon reaching a correct mesh.
    nRelaxIter 3;

    // Number of smoothing iterations of surface normals
    nSmoothSurfaceNormals 1;

    // Number of smoothing iterations of interior mesh movement direction
    nSmoothNormals 3;

    // Smooth layer thickness over surface patches
    nSmoothThickness 10;

    // Stop layer growth on highly warped cells
    maxFaceThicknessRatio 0.5;

    // Reduce layer growth where ratio thickness to medial
    // distance is large
    maxThicknessToMedialRatio 0.3;

    // Angle used to pick up medial axis points
    minMedianAxisAngle 130;

    // Create buffer region for new layer terminations
    nBufferCellsNoExtrude 0;

    // Overall max number of layer addition iterations
    nLayerIter 50;
}


// Generic mesh quality settings. At any undoable phase these determine
```

```
// where to undo.
meshQualityControls
{
    //- Maximum non-orthogonality allowed. Set to 180 to disable.
    maxNonOrtho 65;

    //- Max skewness allowed. Set to <0 to disable.
    maxBoundarySkewness 20;
    maxInternalSkewness 4;

    //- Max concaveness allowed. Is angle (in degrees) below which concavity
    //  is allowed. 0 is straight face, <0 would be convex face.
    //  Set to 180 to disable.
    maxConcave 80;

    //- Minimum projected area v.s. actual area. Set to -1 to disable.
    minFlatness 0.5;

    //- Minimum pyramid volume. Is absolute volume of cell pyramid.
    //  Set to a sensible fraction of the smallest cell volume expected.
    //  Set to very negative number (e.g. -1E30) to disable.
    minVol 1e-10;
minTetQuality 1e-30;
    //- Minimum face area. Set to <0 to disable.
    minArea -1;

    //- Minimum face twist. Set to <-1 to disable. dot product of face normal
    //- and face centre triangles normal
    minTwist 0.02;

    //- minimum normalised cell determinant
    //- 1 = hex, <= 0 = folded or flattened illegal cell
    minDeterminant 0.001;

    //- minFaceWeight (0 -> 0.5)
    minFaceWeight 0.02;

    //- minVolRatio (0 -> 1)
    minVolRatio 0.01;

    //must be >0 for Fluent compatibility
    minTriangleTwist -1;


    // Advanced

    //- Number of error distribution iterations
    nSmoothScale 4;
    //- amount to scale back displacement at error points
    errorReduction 0.75;
}


// Advanced

// Flags for optional output
// 0 : only write final meshes
// 1 : write intermediate meshes
// 2 : write volScalarField with cellLevel for postprocessing
// 4 : write current intersections as .obj files
debug 0;


// Merge tolerance. Is fraction of overall bounding box of initial mesh.
// Note: the write tolerance needs to be higher than this.
mergeTolerance 1E-6;


// ************************************************************************* //


/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pimpleDyMFoam;
```

```
startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         10;

deltaT          2e-4;

writeControl    adjustableRunTime;

writeInterval   1e-2;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

adjustTimeStep  false;

maxCo           0.05;

libs (
"libOpenFOAM.so"
"libExeFunc.so"
"libFilePPvf.so"
"libBodyInfo.so"
"libmytvfriv.so"
"libMyFilePPvf.so"
);

functions
{
 body1
    {
        type                bodyInfo;
        functionObjectLibs  ( "libBodyInfo.so" );
        outputControl       timeStep;
        outputInterval      1;
        patch              (Fish_Link_1);
        pName               p;
        UName               U;

        rhoName             rhoInf;
        log                 true;
        rhoInf              1000;
    }
 body2
    {
        type                bodyInfo;
        functionObjectLibs  ( "libBodyInfo.so" );
        outputControl       timeStep;
        outputInterval      1;
        patch              (Fish_Link_2);
        pName               p;
        UName               U;

        rhoName             rhoInf;
        log                 true;
        rhoInf              1000;
    }
 body3
    {
        type                bodyInfo;
        functionObjectLibs  ( "libBodyInfo.so" );
        outputControl       timeStep;
        outputInterval      1;
        patch              (Fish_Link_3);
        pName               p;
        UName               U;

        rhoName             rhoInf;
        log                 true;
        rhoInf              1000;
    }
 body4
    {
        type                bodyInfo;
        functionObjectLibs  ( "libBodyInfo.so" );
        outputControl       timeStep;
```

```
        outputInterval      1;
        patch             (Fish_Link_4);
        pName               p;
        UName               U;

        rhoName             rhoInf;
        log                 true;
        rhoInf              1000;
    }
   executable
    {
        type                ExeFunc;
        functionObjectLibs  ( "libExeFunc.so" );
        outputControl       timeStep;
        outputInterval      1;
       Executable "octave OctExe/Fish.m";
    }


}

// ************************************************************************* //


/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default Euler;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss linearUpwind grad(U);
    div(phi,k)      Gauss limitedLinear 1;
    div(phi,omega)  Gauss limitedLinear 1;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear limited 0.5;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    pcorr           ;
    p;
}

// ************************************************************************* //


/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
```

```
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    pcorr
    {
        solver           GAMG;
        tolerance        0.02;
        relTol           0;
        smoother         GaussSeidel;
        nPreSweeps       2;
        nPostSweeps      8;
        cacheAgglomeration on;
        agglomerator     faceAreaPair;
        nCellsInCoarsestLevel 10;
        mergeLevels      1;
    }

    p
    {
        $pcorr
        tolerance        1e-7;
        relTol           0.01;
    }

    pFinal
    {
        $p;
nPostSweeps     10;
        tolerance        1e-7;
        relTol           0;
    }

    "(U|k|omega)"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-06;
        relTol          0.1;
    }

    "(U|k|omega)Final"
    {
        $U;
        tolerance       1e-06;
        relTol          0;
    }

    cellDisplacement
    {
        solver          GAMG;
        tolerance       1e-5;
        relTol          0;
        smoother        GaussSeidel;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
    }
}

PIMPLE
{
    correctPhi          yes;
    nOuterCorrectors    2;
    nCorrectors         8;
    nNonOrthogonalCorrectors 0;
}

relaxationFactors
{
    p               0.3;
    "(U|k|omega)"   0.7;
}

cache
{
    grad(U);
}
```

```
// ************************************************************************* //


/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      motionProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dynamicFvMesh      dynamicMotionSolverFvMesh;

motionSolverLibs ("libfvMotionSolvers.so");

solver            displacementLaplacian;

diffusivity      inverseDistance 7(Fish_Joint_1  Fish_Joint_2  Fish_Joint_3 Fish_Link_1 Fish_Link_2 Fish_Link_3 Fish_Link_4);


// ************************************************************************* //


/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      RASProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

RASModel        kOmegaSST;

turbulence      on;

printCoeffs     on;


// ************************************************************************* //


/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  RASModel;


// ************************************************************************* //


/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
```

```
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#include        "include/initialConditions"

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform $turbulentKE;

boundaryField
{

    bound_2
    {
        type            inletOutlet;
        inletValue      $internalField;
        value           $internalField;
    }

    bound_3
    {
         type            kqRWallFunction;
        value           $internalField;
    }
    bound_1
    {

type            fixedValue;
        value           $internalField;
    }

mirror
{
  type symmetryPlane;
}



    "Fish_.*"
    {
        type            kqRWallFunction;
        value           $internalField;
    }


}


// ************************************************************************* //


/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    "Fish_.*"
    {
        type            nutkWallFunction;
        value           uniform 0;
    }

    bound_1
    {
```

```
        type            calculated;
        value           uniform 0;
    }
 bound_2
    {
        type            calculated;
        value           uniform 0;
    }
 bound_3
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
mirror
{
  type symmetryPlane;
}
}
}
}


// ************************************************************************* //



/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#include        "include/initialConditions"

dimensions      [0 0 -1 0 0 0 0];

internalField   uniform $turbulentOmega;

boundaryField
{

   bound_2
    {
        type            inletOutlet;
        inletValue      $internalField;
        value           $internalField;
    }

    bound_3
    {
   type            omegaWallFunction;
        value           $internalField;
    }
    bound_1
    {

type            fixedValue;
        value           $internalField;
    }

   mirror
{
  type symmetryPlane;
}




    "Fish_.*"
    {
        type            omegaWallFunction;
        value           $internalField;
    }


}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];
#include        "include/initialConditions"
internalField  uniform $pressure;

boundaryField
{


    "Fish_.*"
    {
        type            zeroGradient;
    }

 bound_1
    {
       type            zeroGradient;
    }

    bound_2
    {
       type            fixedValue;
        value           $internalField;
    }

 bound_3
    {
       type            zeroGradient;
    }
mirror
{
  type symmetryPlane;
}

}

// ************************************************************************* //


/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    "Fish_.*"
    {
        type          movingWallVelocity;
        value           uniform (0 0 0);
    }

   bound_1
    {
    type          myTimeVaryingUniformInletOutlet;
    timeDataFileName     "OctExe/Flow.dat";

    value $internalField;
    }
```

```
    bound_2
    {
      type            inletOutlet;
        inletValue      uniform (0 0 0);
        value           $internalField;
    }
    bound_3
    {
    type fixedValue;
value uniform (0 0 0);
    }
  mirror
{
  type symmetryPlane;
}



}

// ************************************************************************* //


/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       pointVectorField;
    location    "0.01";
    object      pointDisplacement;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 0 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
  Fish_Joint_1
        {
type fileBend;
timeDataFileName   "OctExe/Joint1.dat";
timeDataFile2Name  "OctExe/Joint1V.dat";
bdist         0.1;
angle0        0;
axis         (0 0 1);
origin      (0 0 0);
value uniform (0 0 0);
    }
  Fish_Joint_2
        {
type fileBend;
timeDataFileName   "OctExe/Joint2.dat";
timeDataFile2Name  "OctExe/Joint2V.dat";
bdist         0.05;
angle0        0;
axis         (0 0 1);
origin      (-0.14 0 0);
value uniform (0 0 0);
    }
  Fish_Joint_3
        {
type fileBend;
timeDataFileName   "OctExe/Joint3.dat";
timeDataFile2Name  "OctExe/Joint3V.dat";
bdist         0.025;
angle0        0;
axis         (0 0 1);
origin      (-0.28 0 0);
value uniform (0 0 0);
    }
    Fish_Link_1
        {
type fileDisplacment;
timeDataFileName   "OctExe/bod1.dat";
angle0        0;
axis         (0 0 1);
origin      (0.242194 0 0);
value uniform (0 0 0);
    }
```

```
    Fish_Link_2
    {
type fileDisplacment;
timeDataFileName   "OctExe/bod2.dat";
angle0        0;
axis         (0 0 1);
origin      (-0.0618 0 0);
value uniform (0 0 0);
    }
   Fish_Link_3
   {
type fileDisplacment;
timeDataFileName   "OctExe/bod3.dat";
angle0        0;
axis         (0 0 1);
origin      (-0.193 0 0);
value uniform (0 0 0);

    }
   Fish_Link_4
   {
type fileDisplacment;
timeDataFileName   "OctExe/bod4.dat";
angle0        0;
axis         (0 0 1);
origin      (-0.349 0 0);
value uniform (0 0 0);

    }
    "bound.*"
    {
  type fixedValue;
value uniform (0 0 0);
    }
mirror
{
  type symmetryPlane;
}


}


// ************************************************************************* //
```

B.2 Octave ODE solving script

The following section gives the octave script files used in this thesis in verbatim form.

Fish.m

```
1;
cd OctExe
dir
fishModel

Dt = dt/ceil(dt/1e-6);
HinfReader
%contQuad
%fuzzyCont
for  i=1:ceil(dt/1e-6)
Hinf

Plant

endfor
ObsWR
record
```

fishModel.m

```
1;
state = myReader("state");
bod1= myReader("../body1/bodyInfoOut.dat");
bod2= myReader("../body2/bodyInfoOut.dat");
bod3= myReader("../body3/bodyInfoOut.dat");
bod4= myReader("../body4/bodyInfoOut.dat");
B = [1;-1;0;0];
dt=bod1(1,1);

coms = [0.2422 -0.06175 -0.193 -0.349];
mass = [19.18 3.226 1.155 0.288];
J=[4.07e-1 0 0 0;0 1.336e-3 0 0;0 0 2.2e-4 0;0 0 0 1.797e-4];
```

```
Hinge=[0 -0.14 -0.28];
l= [(coms(1)-Hinge(1)) (Hinge(1)-coms(2)) (coms(2)-Hinge(2)) (Hinge(2)-coms(3)) (coms(3)-Hinge(3)) (Hinge(3)-coms(4))];
L =[l(1) 0 0 0;
 l(2) l(3) 0 0;
   0 l(4) l(5) 0;
      0 0 l(6) 0];
mm = [mass(1) 0 0 0;mass(1) mass(2) 0 0;mass(1) mass(2) mass(3) 0];

T2= L(:,1:3)*mm;

M= [sum(mass(2:4))*l(1) sum(mass(2:4))*l(2)+(mass(3)+mass(4))*l(3) (mass(3)+mass(4))*l(4)+mass(4)*l(5) mass(4)*l(6);
-mass(1)*l(1) -mass(1)*l(2)+(mass(3)+mass(4))*l(3) (mass(3)+mass(4))*l(4)+mass(4)*l(5) mass(4)*l(6);
-mass(1)*l(1) -mass(1)*l(2)-(mass(1)+mass(2))*l(3) -(mass(1)+mass(2))*l(4)+mass(4)*l(5) mass(4)*l(6);
-mass(1)*l(1) -mass(1)*l(2)-(mass(1)+mass(2))*l(3) -(mass(1)+mass(2))*l(4)-(sum(mass(1:3)))*l(5) -sum(mass(1:3))*l(6)];

M*=1/sum(mass);

K = [16   -15   0 0;-15   35  -20 0;0  -20   32.5 -12.5;0 0 -12.5 12.5];
Ks= K;

Ks(1,1)=-Ks(1,2);


LinAM = [17.8528    4.8774    2.5494    7.8239];
RotAM = [2.777319   0.180589   0.070902   0.344297];



Q= 0.5*[Ks zeros(4);zeros(4) J+M'*(diag(mass))*M];
Qc = 0.5*[K zeros(4);zeros(4) J+M'*(diag(mass))*M];;

damp = 0.01*[0.5 0 0 0;0 1 -1 0;0 -1 2 -1;0 0 -1 1];
t= state(1);
q= state(2:5)';
dq = state(6:9)';
dx = state(10);
dy = state(11);
vx = state(12);
vy = state(13);
tcost = state(14);
ybar = state(15);
u=state(16);
x=[q;dq];
oldF= myReader("Fstore");
lax=5;
f1 =(lax*oldF(1:3)+bod1(2:4)+bod1(5:7))/(lax+1);
f2 =(lax*oldF(4:6)+bod2(2:4)+bod2(5:7))/(lax+1);
f3 =(lax*oldF(7:9)+bod3(2:4)+bod3(5:7))/(lax+1);
f4 =(lax*oldF(10:12)+bod4(2:4)+bod4(5:7))/(lax+1);
m1 =(lax*oldF(13:15)+bod1(8:10)+bod1(11:13))/(lax+1);
m2 =(lax*oldF(16:18)+bod2(8:10)+bod2(11:13))/(lax + 1);
m3 =(lax*oldF(19:21)+bod3(8:10)+bod3(11:13))/(lax + 1);
m4 =(lax*oldF(22:24)+bod4(8:10)+bod4(11:13))/(lax + 1);


% Write Fstore
fid = fopen("Fstore","w") ;
fprintf(fid,"%g ",[f1 f2 f3 f4 m1 m2 m3 m4]);
fclose(fid);

Ax = (f1(1)+f2(1)+f3(1) + f4(1))/sum(mass);
Ay = (f1(2)+f2(2)+f3(2) + f4(2))/sum(mass);
```

Plant.m

```
1;
Ht = J+ sin(diag(q))*T2*M*sin(diag(q))+diag(cos(q))*T2*M*diag(cos(q));
Ct =-sin(diag(q))*T2*M*diag(cos(q))+diag(cos(q))*T2*M*sin(diag(q));
Ct = Ct*diag(dq) - damp ;

Rfpx= -mm*(ones(4,1)*Ax -[f1(1);f2(1);f3(1);f4(1)]./mass');
Rfpy= -mm*(ones(4,1)*Ax -[f1(2);f2(2);f3(2);f4(2)]./mass');

D = [m1(3);m2(3);m3(3);m4(3)]-sin(diag(q))*L(:,1:3)*Rfpx + diag(cos(q))*L(:,1:3)*Rfpy ;

if abs(det(Ht))>1e-10
H2 = Ht^(-1);
else
H2= (J+ T2*M)^(-1);
endif
At = [zeros(4) eye(4);-H2*K H2*(Ct)];
Ats = [zeros(4) eye(4);-H2*Ks H2*(Ct)];
EA = real(e^(Ats*Dt));
xt=EA*x;
xt+= real(At^(-1)*(EA-eye(8))*([zeros(4,1);H2*D] +[zeros(4,1);H2*B*u]));

%for i=1:3
%if (abs(xt(i+3)*D(i)/x(i+3))<1e2)
```

```
%D(i)= xt(i+3)*D(i)/x(i+3);
%endif
%endfor
x=EA*x;
x+= real(At^(-1)*(EA-eye(8))*([zeros(4,1);H2*D] +[zeros(4,1);H2*B*u]));


dx += vx*Dt + 0.5*Ax*Dt^2;
dy += vy*Dt + 0.5*Ay*Dt^2;

vx+= Ax*Dt;
vy+= Ay*Dt;
q=x(1:4);
dq=x(5:8);

t+=Dt;
tcost = (tcost + 500*Dt*u*[1 -1 0 0] *dq)/(1+500*Dt);
ybar = (ybar +300*Dt*x'*Q*x)/(1+300*Dt);
```

record.m

```
xdiss = M *(cos(q)-ones(4,1));
ydiss =  M*sin(q);



% Write State
fid = fopen("state","w") ;
fprintf(fid,"%g ",[t,x',dx,dy,vx,vy,tcost,ybar,u]);
fclose(fid);

% Write coms
fid =  fopen("../body1/bodyInfoIn.dat","w");
fprintf(fid,"(");
fprintf(fid,"%g ",[coms(1)+dx+xdiss(1) dy+ydiss(1) 0]);
fprintf(fid,")");
fclose(fid);
fid =  fopen("../body2/bodyInfoIn.dat","w");
fprintf(fid,"(");
fprintf(fid,"%g ",[coms(2)+dx+xdiss(2) dy+ydiss(2) 0]);
fprintf(fid,")");
fclose(fid);
fid =  fopen("../body3/bodyInfoIn.dat","w");
fprintf(fid,"(");
fprintf(fid,"%g ",[coms(3)+dx+xdiss(3) dy+ydiss(3) 0]);
fprintf(fid,")");
fclose(fid);

%  Write point displacments
fid =  fopen("bod1.dat","w");
fprintf(fid,"(");
fprintf(fid,"%g ",[x(1) dx+xdiss(1) dy+ydiss(1)]);
fprintf(fid,")");
fclose(fid);
fid =  fopen("bod2.dat","w");
fprintf(fid,"(");
fprintf(fid,"%g ",[x(2) dx+xdiss(2) dy+ydiss(2)]);
fprintf(fid,")");
fclose(fid);
fid =  fopen("bod3.dat","w");
fprintf(fid,"(");
fprintf(fid,"%e ",[x(3) dx+xdiss(3) dy+ydiss(3)]);
fprintf(fid,")");
fclose(fid);
fid =  fopen("bod4.dat","w");
fprintf(fid,"(");
fprintf(fid,"%e ",[x(4) dx+xdiss(4) dy+ydiss(4)]);
fprintf(fid,")");
fclose(fid);
fid =  fopen("Joint1V.dat","w");
fprintf(fid,"(");
fprintf(fid,"%e ",[x(1)-x(2) 0 0]);
fprintf(fid,")");
fclose(fid);
fid =  fopen("Joint2V.dat","w");
fprintf(fid,"(");
fprintf(fid,"%e ",[x(2)-x(3) 0 0]);
fprintf(fid,")");
fclose(fid);
fid =  fopen("Joint3V.dat","w");
fprintf(fid,"(");
fprintf(fid,"%e ",[x(3)-x(4) 0 0]);
fprintf(fid,")");
fid =  fopen("Joint1.dat","w");
fprintf(fid,"(");
fprintf(fid,"%e ",[x(1)-(x(1)-x(2))/2 dx+xdiss(1)-l(1)*cos(x(1))+l(1) dy+ydiss(1)-l(1)*sin(x(1))]);
fprintf(fid,")");
```

```
fclose(fid);
fid =  fopen("Joint2.dat","w");
fprintf(fid,"(");
fprintf(fid,"%e ",[x(2)-(x(2)-x(3))/2 dx+xdiss(2)-l(3)*cos(x(2))+l(3) dy+ydiss(2)-l(3)*sin(x(2))]);
fprintf(fid,")");
fclose(fid);
fid =  fopen("Joint3.dat","w");
fprintf(fid,"(");
fprintf(fid,"%e ",[x(3)-(x(3)-x(4))/2 dx+xdiss(3)-l(5)*cos(x(3))+l(5) dy+ydiss(3)-l(5)*sin(x(3))]);
fprintf(fid,")");
fclose(fid);
fid =  fopen("record.dat","a");
fprintf(fid,"(");
fprintf(fid,"%g ",[t+dt,dt,x',dx,dy,vx,vy,u]);
fprintf(fid,") \n");
fclose(fid);

fid = fopen("Flow.dat", "w");
fprintf(fid, "inletValue uniform (%g 0 0);", -5*dx);
fclose(fid);
fid = fopen("Dist.dat","a");
fprintf(fid, "(")
fprintf(fid, "%g ", [t+dt D']);
fprintf(fid,") \n");
fclose(fid);
```

B.2.1 Deadbeat Controller

contQuad.m

```
1;


Ht = J+ sin(diag(q))*T2*M*sin(diag(q))+diag(cos(q))*T2*M*diag(cos(q));
Ct =-sin(diag(q))*T2*M*diag(cos(q))+diag(cos(q))*T2*M*sin(diag(q));
Ct = Ct*diag(dq) - damp ;



if abs(det(Ht))>1e-10
H2 = Ht^(-1);
else
H2= (J+ T2*M)^(-1);
endif
At = [zeros(4) eye(4);-H2*K H2*(Ct)];
Ats = [zeros(4) eye(4);-H2*Ks H2*(Ct)];
EA = real(e^(Ats*dt));
xt=EA*x;
Bt= real(At^(-1)*(EA-eye(8))*([zeros(4,1);H2*B]));


a= Bt'*Q*Bt;
b= 2*Bt'*Q*xt;
c= xt'*Q*xt -0.35;

rot =(b^2-4*a*c)^(0.5);
if isreal(rot)
uP= (-b+rot)/(2*a);
uM= (-b-rot)/(2*a);

if abs(uP)<abs(uM)
u=uP;
else
u=uM;
endif

else
u= -0.1*b/(2*a);
endif

if abs(u)>100
u=100*u/abs(u);
endif
```

B.2.2 H_∞ Controller

HinfMat.m

```
H=J+ T2*M;

Damp = 0.1*diag(mass);
A1 = [zeros(4) eye(4);-H^(-1)*K -H^(-1)*(0.0001*Ks)];
```

```
Q= 0.5*[K zeros(4,4);zeros(4,4) J+M'*diag(mass)*M];
A2= [A1' Q;zeros(8) A1];
qs = [0.1 0.3 0.3 0.3];

difH=diag(cos(qs))*T2*M*diag(cos(qs))+diag(sin(qs))*T2*M*diag(sin(qs))-T2*M;
Kdiff = (H +difH)^(-1)*difH*H^(-1)*K;

Kerr =abs(diag(sum(Kdiff'))*max(qs));

%B1=[[zeros(8,4);Kerr] zeros(12,4);zeros(4) 0.1*H^(-1)*diag(mass)];


B2 =  [zeros(12,1);H^(-1)*B];
B1= [[zeros(8);eye(8)]];

C1=[[zeros(4,1);H^(-1)*B]' zeros(1,8)];
C2=[zeros(8) eye(8)];
```

HinfSetup.m

```
fishModel
HinfMat

con1=con2=con3=0;
gam=100;
while(con1+con2+con3)<3
[r p] = size(B1);
[r q] = size(B2);
R = [-gam^(-2)*eye(p) zeros(p,q);zeros(q,p) eye(q)]^-1;
 X22= MyCare(A2,[B1 B2],C1,R);
con1=defPos2(X22);
[p r] = size(C1);
[q r] = size(C2);
R = [-gam^(-2)*eye(p) zeros(p,q);zeros(q,p) eye(q)]^-1;
 Y22= MyCare(A2,[C1' C2'],B1',R);
con2 = defPos2(Y22);
F22= -B2'*X22;
con3 = (max(real(eig(A2+B2*F22)))<0);
gam *= 1.5;
endwhile
gam/=1.5;
L22= -Y22*C2';
Z22= (eye(16)-gam^(-2)*Y22*X22)^(-1);


function printArray(in,fid)
[a b]= size(in);
for i=1:a
fprintf(fid,"%g ",in(i,:))
fprintf(fid,"\n")
endfor
endfunction

fid = fopen("HinfPara","w") ;
printArray(gam,fid);
printArray(F22,fid);
printArray(X22,fid);
printArray((Z22*L22)',fid);
fclose(fid);
```

HinfReader.m

```
gub = myReader("HinfPara");
[a b]=size(gub);
gam=gub(1,1);
F22= gub(2,:);
X22=gub(3:18,:);
ZL = (gub(19:a,:))';
HinfMat
Nbar =  rscale(A2,B2,C1,0,F22);
A4=A2+B2*F22;
EA4 = real(expm(A4*Dt));
obsu = myReader("observer");
obs=obsu(1:16)';
u=obsu(17);
```

Hinf.m

```
1;

sk = -C1*A4^(-1)*B2;
Ed = u*B'*dq - dq'*Damp*dq;
```

```
error = (0.35-x'*Q*x)*1000;
de = max(abs(B'*dq),0.1);

if ((F22*obs)^2+4*(error)>=0)

uP=0.5*real(F22*obs) +0.5*((F22*obs)^2+4*(error))^0.5;
uM=0.5*real(F22*obs) -0.5*((F22*obs)^2+4*(error))^0.5;
else
uP=uM=real(F22*obs);
endif


LS = -0.05*[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]*obs;

if abs(uP)<(abs(uM)+LS)
u=uP;
else
u=uM;
endif

if u>10
u=10;
endif
if u<-10
u=-10;
endif

Corr=x;

if t>0.5
Corr -=0.025*[1;1;1;1;0;0;0;0];
endif
if t>1
Corr -=0.05*[1;1;1;1;0;0;0;0];
endif
if t>2
Corr -=0.1*[1;1;1;1;0;0;0;0];
endif
obs=EA4*obs;

obs+= A4^(-1)*(EA4-eye(16))*([B1 B2]*[gam^(-2)*B1'*X22*obs;u]+ZL*(C2*obs - Corr));
```

ObsWR.m

```
fid =  fopen("observer","w");
fprintf(fid,"(");
fprintf(fid,"%g ",[obs;u]);
fprintf(fid,")");
fclose(fid);
```

B.2.3 H_∞ Optimiser

script.m

```
1;
Dt =1;
fishModel;
HinfSetup;
HinfReader;

SysCL = ss(A2+B2*F22,B1,C1);
Estimator = ss(A2+B1*(gam^-2)*B1'*X22+ZL*C2+B2*F22,ZL,F22);
AP  = [A2 B2*F22;-ZL*C2 Estimator.a];
BP  = [B1;B1*0];
CP = [C1 C1*0];



dACLdF22=[];
for i=1:length(F22)
dACLdF22 = [dACLdF22 zeros(16,i-1) B2 zeros(16,16-i)];
endfor


count = 0;
delta  =0.01;
for i=1:50
[ddA norms]= dHinfdA(A2+B2*F22,B1,C1);
ddF = matChainRule(ddA,dACLdF22,A4);
F22 -= delta*max(abs(F22))*ddF/max(abs(ddF));
SysCL = ss(A2+B2*F22,B1,C1);
```

```
cost(i) = norm(SysCL,inf);
fragility(i) = norm(ddF,"fro");
if cost(i)>cost(i)
count +=1;
elsedd
count =0;
endif
if count>5
count = 0;
delta *=0.1;
endif

endfor

plot(cost)

function printArray(in,fid)
[a b]= size(in);
for i=1:a
fprintf(fid,"%g ",in(i,:))
fprintf(fid,"\n")
endfor
endfunction

fid = fopen("HinfPara","w") ;
printArray(gam,fid);
printArray(F22,fid);
printArray(X22,fid);
printArray((Z22*L22)',fid);
fclose(fid);


function [out norms] = dHinfdA(A,B,C)
[r c]=size(A);
[k n]=size(B);
[m f]=size(C);
[P L] = eig(A);
[dL dV V]=dEVdA(A);
B1 = zeros(size(B(:,1)));
for s=1:n
B1 +=B(:,s);
endfor
B=B1;
[k n]=size(B);

Omeg = real((imag(L)^2 - 2*real(L)^2).^0.5);

norms = abs(C*P)*abs((Omeg*i-L)^-1)*abs(P^-1*B);

[dump index]= sort(norms);

nor=dump(m);
C = C(index(m),:);
[m f]=size(C);

if norm(Omeg,"fro")==0
dOmeg = zeros(c^2);
else
dOmeg =   kron(eye(r) ,Omeg^-1)*(kron(eye(r),imag(L))*imag(dL) - 2*kron(eye(c),real(L))*real(dL));
endif

dOmeg;
[dr di]= dabsdA((Omeg*i-L)^-1);
size(dr);
F1r = matChainRule(dr,real(dmAdA((Omeg*i-L))),(Omeg*i-L)^-1);
F1i = matChainRule(di,imag(dmAdA((Omeg*i-L))),(Omeg*i-L)^-1);
F2r = matChainRule(F1r,real(dOmeg*i -dL),Omeg*i-L);
F2i = matChainRule(F1i,imag(dOmeg*i -dL),Omeg*i-L);


Component1=kron(eye(r),abs(C*P))*(F2r +F2i)*kron(eye(c),abs(P^-1*B));
[dr di]= dabsdA(C*V);

F1r = matChainRule(dr,real(kron(eye(r),C)*dV),C);
F1i = matChainRule(di,imag(kron(eye(r),C)*dV),C);
Component2=(F1r+F1i)*kron(eye(c),abs((Omeg*i-L)^-1)*abs(V^-1*B));

[dr di]= dabsdA((V^-1)*B);

F1r = matChainRule(dr,real(dmAdA(V)*kron(eye(c),B)),B);
F1i = matChainRule(di,imag(dmAdA(V)*kron(eye(c),B)),B);

F2r = matChainRule(F1r,real(dV),V);
F2i = matChainRule(F1i,imag(dV),V);

Component3= kron(eye(r),abs(C*V)*abs((Omeg*i-L)^-1))*(F2r+F2i);

out=Component1 + Component2 + Component3;
endfunction
```

```
function [dL dV V]=dEVdA(A)
[r c]=size(A);
[P L] = eig(A);
X=P;
R=P^-1;
Y=conj((P^-1)');
dL=zeros(c^2);
for i=1:c
[trash ind]=sort(abs(X(:,i)).*abs(Y(:,i)));
m=ind(c);
gam(i)=1/X(m,i);
M(i)=m;
endfor

for i=1:c

TdL = dEdA(A,i);

for l=1:r
for m=1:c
dL((l-1)*c +i,(m-1)*c+i) += TdL(l,m);
endfor
endfor
endfor
for f=1:r
for g=1:c
dA=E(r,c,f,g);
for k=1:r
for l=1:c

if k!=l
C(k,l)=R(k,:)*dA*P(:,l)*gam(l)/(gam(k)*(L(l,l)-L(k,k)));
else
C(k,l)=0;
endif

endfor

endfor
for K=1:r
C(k,k)= -sum(X(M(k),:).*gam.*C(:,k)');
endfor
dV((f-1)*c+1:f*c,(g-1)*c+1:g*c)=X*C;
endfor
endfor
V= P*diag(gam);
endfunction



function out=dEdA(In,k)

[V E]= eig(In);
k;
R=V;
L=(R^-1);

out= (R(:,k)*L(k,:))';
%out=(V(:,1)*((V^-1)(1,:)))';
endfunction



function out = dmAdA(A)
[r m] =size(A);

out = -kron(eye(r),A^-1)*kronMat(r,m)*kron(eye(m),A^-1);

endfunction



function [dreal dimag]=dabsdA(A)
[n m]=size(A);

ref1 = sign(real(A));
ref2 = sign(imag(A));
for i=1:n
for j=1:m
if abs(real(A(i,j)))==0
dreal((i-1)*n+i,(j-1)*m+j)=0;

else
dreal((i-1)*n+i,(j-1)*m+j)=real(A(i,j))/abs(A(i,j));

endif
if abs(imag(A(i,j)))==0
dimag((i-1)*n+i,(j-1)*m+j)=0;
```

```
else
dimag((i-1)*n+i,(j-1)*m+j)=imag(A(i,j))/abs(A(i,j));

endif
endfor
endfor



endfunction




function out=matChainRule(dAdB,dBdM,B)
[a b]=size(dBdM);
[r c]=size(B);
[f g]=size(dAdB);
n=f/r;
m=g/c;
k=a/r;
l=b/c;


out = kron(eye(k),dAdrowB(dAdB,B))*kron(dcolBdM(dBdM,B),eye(m));
endfunction
```

B.2.4 Alternative error metric control

```
1;
ContState = myReader("contState")';
target =0.35;

Ht = J+ T2*M;
Bt=[zeros(4,1);Ht^-1*B];
dprod = (x'*Qc*Bt)/(norm(Qc^0.5*x,2)*norm(Qc^0.5*Bt,2));


orth = 1-acos(abs(dprod))/pi;
par =acos( abs(dprod))/pi;
orth =0;
par = 1;
en = sqrt(x'*Qc*x);
err = orth*sqrt(target-en^2)+par*(target^0.5-en);

et  = -Bt'*Qc*x/(Bt'*Qc*Bt)^0.5 +real(sign(Bt'*Qc*x)*(target-x'*Qc*x +Bt'*Qc*x*Bt'*Qc*x/(Bt'*Qc*Bt))^0.5);


cont =[17.1282   -2.7093   87.8466];
%cont=[-5.9877e-01   1.9421e+00   4.9839e+01];

Cont = ss([0 1;-cont(1:2)],[0;1],[1 0],cont(3));
ContState = real(expm(Cont.a*Dt)*ContState + Cont.a^-1*(eye(2)-expm(Cont.a*Dt))*Cont.b*et);
if abs(ContState(1))>1e5
ContState(1)=0;
endif
if abs(ContState(2))>1e5
ContState(2)=0;
endif

u=(Cont.c*ContState + Cont.d*et);



fid = fopen("contState","w") ;
fprintf(fid,"%g ",ContState');
fclose(fid);

if u>10
u=10;
endif
if u<-10
u=-10;
endif
```

B.3 Additional OpenFoam Source code

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
```

```
    \\  /    A nd           | Copyright (C) 1991-2010 OpenCFD Ltd.
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

\*---------------------------------------------------------------------------*/

#include "bodyInfo.H"
#include "volFields.H"
#include "dictionary.H"
#include "Time.H"

#include "incompressible/singlePhaseTransportModel/singlePhaseTransportModel.H"
#include "incompressible/RAS/RASModel/RASModel.H"
#include "incompressible/LES/LESModel/LESModel.H"

#include "basicThermo.H"
#include "compressible/RAS/RASModel/RASModel.H"
#include "compressible/LES/LESModel/LESModel.H"

#include "IFstream.H"



// * * * * * * * * * * * * * * * Static Data Members * * * * * * * * * * * * * * //

namespace Foam
{
    defineTypeNameAndDebug(bodyInfo, 0);
}

// * * * * * * * * * * * * * * Private Member Functions  * * * * * * * * * * * * //









// * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * //

Foam::bodyInfo::bodyInfo
(
    const word& name,
    const objectRegistry& obr,
    const dictionary& dict,
    const bool loadFromFiles
)
:
    name_(name),
    obr_(obr),
    active_(true),
    log_(false),
     pName_(word::null),
    UName_(word::null),
    rhoName_(word::null),
patchname_(word::null)

{



    read(dict);


}


// * * * * * * * * * * * * * * * * * Destructor  * * * * * * * * * * * * * * * * //

Foam::bodyInfo::~bodyInfo()
{}
```

```
// * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * * //


// Read the dictionary and substitute values
void Foam::bodyInfo::read(const dictionary& dict)
{
    if (active_)
    {
        log_ = dict.lookupOrDefault<Switch>("log", false);
        const fvMesh& mesh = refCast<const fvMesh>(obr_);
        // ***Patches for solid bodies****//
   //     patchSet_ =
   //          mesh.boundaryMesh().patchSet(wordList(dict.lookup("patch")));

        dict.readIfPresent("directForceDensity", directForceDensity_);

        if (directForceDensity_)
        {
            // Optional entry for fDName
            fDName_ = dict.lookupOrDefault<word>("fDName", "fD");

            // Check whether fDName exists, if not deactivate forces
            if
            (
                !obr_.foundObject<volVectorField>(fDName_)
            )
            {
                active_ = false;
                WarningIn("void forces::read(const dictionary& dict)")
                << "Could not find " << fDName_ << " in database." << nl
                    << "    De-activating forces."
                    << endl;
            }
        }
        else
        {
            // Optional entries U and p
            pName_ = dict.lookupOrDefault<word>("pName", "p");
            UName_ = dict.lookupOrDefault<word>("UName", "U");
            rhoName_ = dict.lookupOrDefault<word>("rhoName", "rho");

            // Check whether UName, pName and rhoName exists,
            // if not deactivate forces
            if
            (
                !obr_.foundObject<volVectorField>(UName_)
             || !obr_.foundObject<volScalarField>(pName_)
             || (
                    rhoName_ != "rhoInf"
                 && !obr_.foundObject<volScalarField>(rhoName_)
                )
            )
            {
                active_ = false;

                WarningIn("void forces::read(const dictionary& dict)")
                    << "Could not find " << UName_ << ", " << pName_;

                if (rhoName_ != "rhoInf")
                {
                    Info<< " or " << rhoName_;
                }
                Info<< " in database." << nl << "    De-activating forces."
                    << endl;
            }
            // Reference density needed for incompressible calculations
            rhoRef_ = readScalar(dict.lookup("rhoInf"));
            // Reference pressure, 0 by default
            pRef_ = dict.lookupOrDefault<scalar>("pRef", 0.0);
        }
patchname_= wordList(dict.lookup("patch"));
//dict.readIfPresent("COM", COM_);
}
}

void Foam::bodyInfo::makeFile()
{
    // Create the forces file if not already created
    if (OutFilePtr_.empty() || InFilePtr_.empty() )
    {
        if (debug)
        {
            Info<< "Creating bodyInfo file." << endl;
        }

        // File update
        if (Pstream::master())
        {
            fileName bodyInfoDir;
```

```
            word startTimeName =
                obr_.time().timeName(obr_.time().startTime().value());

            if (Pstream::parRun())
            {
                // Put in undecomposed case (Note: gives problems for
                // distributed data running)
                bodyInfoDir = obr_.time().path()/".."/name_;
            }
            else
            {
                bodyInfoDir = obr_.time().path()/name_;
            }

            // Create directory if does not exist.
            mkDir(bodyInfoDir);

            // Open new file at start up

        InFileName_ = bodyInfoDir/(type() + "In.dat");
        OutFileName_ = bodyInfoDir/(type() + "Out.dat"); // Changed


            // Add headers to output data
           writeFileHeader();
        }
    }
}


void Foam::bodyInfo::writeFileHeader()
{
//InFilePtr_() << "(0 0 0)" << endl;
}




void Foam::bodyInfo::execute()
{
    // Do nothing - only valid on write
}


void Foam::bodyInfo::end()
{
    // Do nothing - only valid on write
}


void Foam::bodyInfo::write()
{
    if (active_)
    {
        // Create the forces file if not already created
        makeFile();
scalar DT =obr_.time().deltaTValue();

        if (Pstream::master())
        {
            IFstream dataStream(InFileName_);
           vector COF
                    (
                        dataStream
                    );

            OutFilePtr_.reset(new OFstream(OutFileName_));

            OutFilePtr_()  << "Delta T:  " << DT << " Forces and Moments :" << getForceMoments(COF) << endl;

        }
    }
}



//***calculate the forces for the given body****//
Foam::forces::forcesMoments Foam::bodyInfo::getForceMoments(vector COF)
{
dictionary forcesDict;
    forcesDict.add("patches", patchname_);
    forcesDict.add("rhoInf", rhoRef_);
    forcesDict.add("rhoName", rhoName_);
    forcesDict.add("CofR", COF);
    forces f("forces", obr_, forcesDict);
    forces::forcesMoments fm = f.calcForcesMoment();
return fm;
```

```
}
```

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 1991-2010 OpenCFD Ltd.
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Class
    Foam::bodyInfo

Description
    Calculates the forces and moments by integrating the pressure and
    skin-friction forces over a given list of patches.

    Member function calcForcesMoment()calculates and returns the forces and
    moments.

    Member function forces::write() calls calcForcesMoment() and writes the
    forces and moments into the file \<timeDir\>/forces.dat

SourceFiles
    forces.C
    IOforces.H

\*---------------------------------------------------------------------------*/

#ifndef bodyInfo_H
#define bodyInfo_H

#include "List.H"
#include "vector.H"
#include "Vector2D.H"
#include "primitiveFieldsFwd.H"
#include "volFieldsFwd.H"
#include "HashSet.H"
#include "Tuple2.H"
#include "OFstream.H"
#include "Switch.H"
#include "pointFieldFwd.H"
#include "forces.H"
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{

// Forward declaration of classes
class objectRegistry;
class dictionary;
class mapPolyMesh;

/*---------------------------------------------------------------------------*\
                          Class forces Declaration
\*---------------------------------------------------------------------------*/

class bodyInfo
{
public:

    // list of vectors

typedef List<vector> vectorVec;
typedef List<wordList> PatchList;
typedef Tuple2<vector, tensor> positionState;
typedef Tuple2<positionState, vectorVec> stateData;
//typedef struct sixMat { scalar e[6][6];} sixMat;
//typedef struct sixVec { scalar e[6];}sixVec;
```

```
//typedef required for force calculations

  typedef Tuple2<vector, vector> pressureViscous;

    // Tuple which holds the forces (.first()) and moment (.second)
    // pressure/viscous forces Tuples.
    typedef Tuple2<pressureViscous, pressureViscous> forcesMoments;


protected:

    // Private data

        //- Name of this set of forces,
        //  Also used as the name of the probes directory.
        word name_;
        const objectRegistry& obr_;
        //- on/off switch
        bool active_;



        //- Switch to send output to Info as well as to file
        Switch log_;
        // Read from dictionary
labelHashSet patchSet_;

        wordList patchname_;


            //- Name of pressure field
            word pName_;
            //- Name of velocity field
            word UName_;
            //- Name of density field (optional)
            word rhoName_;
            //- Is the force density being supplied directly?
            Switch directForceDensity_;
            //- The name of the force density (fD) field
            word fDName_;
            //- Reference density needed for incompressible calculations
            scalar rhoRef_;
            //- Reference pressure
            scalar pRef_;
            //- Centre of rotation






        // File writing info
fileName InFileName_;
        fileName OutFileName_;
      ;
        //- Forces/moment file ptr
        autoPtr<OFstream> InFilePtr_;
autoPtr<OFstream> OutFilePtr_;




    // Private Member Functions

        //- If the forces file has not been created create it
        void makeFile();
        void getForces();



        //- Output file header information
        virtual void writeFileHeader();

       //- Calculates the present system energy (not including added mass)

// Calclate state data objects from q vectors



        forcesMoments getForceMoments(vector);

        //- Disallow default bitwise copy construct
        bodyInfo(const bodyInfo&);

        //- Disallow default bitwise assignment
        void operator=(const bodyInfo&);
```

```
public:

    //- Runtime type information
    TypeName("bodyInfo");


    // Constructors

        //- Construct for given objectRegistry and dictionary.
        //  Allow the possibility to load fields from files
        bodyInfo
        (
            const word& name,
            const objectRegistry&,
            const dictionary&,
            const bool loadFromFiles = false
        );


    //- Destructor
    virtual ~bodyInfo();


    // Member Functions

        //- Return name of the set of forces
        virtual const word& name() const
        {
            return name_;
        }

        //- Read the forces data
        virtual void read(const dictionary&);

        //- Execute, currently does nothing
        virtual void execute();

        //- Execute at the final time-loop, currently does nothing
        virtual void end();

        //- Write the forces
        virtual void write();

        //- Calculate and return forces and moment
    //    virtual void calcfunctionVariable() const;



        //- Update for changes of mesh
        virtual void updateMesh(const mapPolyMesh&)
        {}



        //- Update for changes of mesh
        virtual void movePoints(const pointField&)
        {}
};


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

} // End namespace Foam

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#endif

// ************************************************************************* //


/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 1991-2010 OpenCFD Ltd.
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
```

```
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

\*---------------------------------------------------------------------------*/

#include "bodyInfoFunctionObject.H"

// * * * * * * * * * * * * * * * * Static Data Members * * * * * * * * * * * * * * * * //

namespace Foam
{
    defineNamedTemplateTypeNameAndDebug(bodyInfoFunctionObject, 0);

    addToRunTimeSelectionTable
    (
        functionObject,
        bodyInfoFunctionObject,
        dictionary
    );
}

// ************************************************************************* //


/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 1991-2010 OpenCFD Ltd.
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Typedef
    Foam::bodyInfoFunctionObject

Description
    FunctionObject wrapper around forces to allow them to be created via the
    functions entry within controlDict.

SourceFiles
    forcesFunctionObject.C

\*---------------------------------------------------------------------------*/

#ifndef bodyInfoFunctionObject_H
#define bodyInfoFunctionObject_H

#include "bodyInfo.H"
#include "OutputFilterFunctionObject.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{
    typedef OutputFilterFunctionObject<bodyInfo> bodyInfoFunctionObject;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#endif

// ************************************************************************* //


/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 1991-2010 OpenCFD Ltd.
     \\/     M anipulation  |
```

```
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

\*---------------------------------------------------------------------------*/

#include "ExeFunc.H"
#include "volFields.H"
#include "dictionary.H"
#include "Time.H"

#include "incompressible/singlePhaseTransportModel/singlePhaseTransportModel.H"
#include "incompressible/RAS/RASModel/RASModel.H"
#include "incompressible/LES/LESModel/LESModel.H"

#include "basicThermo.H"
#include "compressible/RAS/RASModel/RASModel.H"
#include "compressible/LES/LESModel/LESModel.H"


// * * * * * * * * * * * * * * * * Static Data Members * * * * * * * * * * * * * * * //

namespace Foam
{
    defineTypeNameAndDebug(ExeFunc, 0);
}

// * * * * * * * * * * * * * * * Private Member Functions  * * * * * * * * * * * * * //









// * * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * * //

Foam::ExeFunc::ExeFunc
(
    const word& name,
    const objectRegistry& obr,
    const dictionary& dict,
    const bool loadFromFiles
)
:
    name_(name),
    obr_(obr),
    active_(true),
    log_(false),


    ExeFuncFilePtr_(NULL)
{



    read(dict);
}


// * * * * * * * * * * * * * * * * * * Destructor  * * * * * * * * * * * * * * * * * //

Foam::ExeFunc::~ExeFunc()
{}


// * * * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * * //

void Foam::ExeFunc::read(const dictionary& dict)
{
    if (active_)
    {
        log_ = dict.lookupOrDefault<Switch>("log", false);
```

```
        Executable_ = dict.lookupOrDefault<string>("Executable", "myfun.m");


    }
}


void Foam::ExeFunc::makeFile()
{
    // Create the forces file if not already created
    if (ExeFuncFilePtr_.empty())
    {
        if (debug)
        {
        //    Info<< "Creating ExeFunc file." << endl;
        }

        // File update
        if (Pstream::master())
        {
            fileName ExeFuncDir;
            word startTimeName =
                obr_.time().timeName(obr_.time().startTime().value());

            if (Pstream::parRun())
            {
                // Put in undecomposed case (Note: gives problems for
                // distributed data running)
                ExeFuncDir = obr_.time().path()/".."/name_/startTimeName;
            }
            else
            {
                ExeFuncDir = obr_.time().path()/name_/startTimeName;
            }

            // Create directory if does not exist.
      //      mkDir(ExeFuncDir);

            // Open new file at start up
    //        ExeFuncFilePtr_.reset(new OFstream(ExeFuncDir/(type() + ".dat")));

            // Add headers to output data
      //      writeFileHeader();
        }
    }
}


void Foam::ExeFunc::writeFileHeader()
{
    if (ExeFuncFilePtr_.valid())
    {
        ExeFuncFilePtr_()
            << "# This" << tab
            << "Not Strictly Necessary"
            << endl;
    }
}


void Foam::ExeFunc::execute()
{
    // Do nothing - only valid on write
}


void Foam::ExeFunc::end()
{
    // Do nothing - only valid on write
}


void Foam::ExeFunc::write()
{
    if (active_)
    {
        // Create the forces file if not already created
    //    makeFile();
        // Call the calculation Function
        system(Executable_);

        if (Pstream::master())
        {


            if (log_)
            {
```

```
            }
        }
    }
}

Foam::ExeFunc::functionVariable Foam::ExeFunc::calcfunctionVariable() const
{
scalar t=obr_.time().value();
    functionVariable fm
    (
         vector
  (
            5*Foam::sin(4*t),
            3*Foam::sin(2*t),
            5*Foam::sin(4*t)
        ),
  vector
  (
            5*Foam::sin(4*t),
            3*Foam::sin(2*t),
            5*Foam::sin(4*t)
        )

    );


    return fm;
}



// ************************************************************************* //


/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 1991-2010 OpenCFD Ltd.
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Class
    Foam::ExeFunc

Description
    Calculates the forces and moments by integrating the pressure and
    skin-friction forces over a given list of patches.

    Member function calcForcesMoment()calculates and returns the forces and
    moments.

    Member function forces::write() calls calcForcesMoment() and writes the
    forces and moments into the file \<timeDir\>/forces.dat

SourceFiles
    forces.C
    IOforces.H

\*---------------------------------------------------------------------------*/

#ifndef ExeFunc_H
#define ExeFunc_H

#include "List.H"
#include "vector.H"
#include "Vector2D.H"
#include "primitiveFieldsFwd.H"
#include "volFieldsFwd.H"
#include "HashSet.H"
#include "Tuple2.H"
#include "OFstream.H"
#include "Switch.H"
#include "pointFieldFwd.H"
```

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{

// Forward declaration of classes
class objectRegistry;
class dictionary;
class mapPolyMesh;

/*---------------------------------------------------------------------------*\
                           Class forces Declaration
\*---------------------------------------------------------------------------*/

class ExeFunc
{
public:

    // Tuple which holds the pressure (.first()) and viscous (.second) forces
    typedef Tuple2<vector, vector> functionVariable;

    // Tuple which holds the forces (.first()) and moment (.second)
    // pressure/viscous forces Tuples.
   // typedef Tuple2<pressureViscous, pressureViscous> forcesMoments;

    //- Sum operation class to accumulate the pressure, viscous forces and moments
    class sumOp
    {
    public:

        functionVariable operator()
        (
            const functionVariable& fm1

        ) const
        {
            return functionVariable
            (
                functionVariable
                (
                    fm1
                )

            );
        }
    };


protected:

    // Private data

        //- Name of this set of forces,
        //  Also used as the name of the probes directory.
        word name_;

        const objectRegistry& obr_;

        //- on/off switch
        bool active_;

        //- Switch to send output to Info as well as to file
        Switch log_;

        // Read from dictionary

      string Executable_;

        //- Forces/moment file ptr
        autoPtr<OFstream> ExeFuncFilePtr_;


    // Private Member Functions

        //- If the forces file has not been created create it
        void makeFile();

        //- Output file header information
        virtual void writeFileHeader();







        //- Disallow default bitwise copy construct
        ExeFunc(const ExeFunc&);

        //- Disallow default bitwise assignment
```

```
        void operator=(const ExeFunc&);


public:

    //- Runtime type information
    TypeName("ExeFunc");


    // Constructors

        //- Construct for given objectRegistry and dictionary.
        //  Allow the possibility to load fields from files
        ExeFunc
        (
            const word& name,
            const objectRegistry&,
            const dictionary&,
            const bool loadFromFiles = false
        );


    //- Destructor
    virtual ~ExeFunc();


    // Member Functions

        //- Return name of the set of forces
        virtual const word& name() const
        {
            return name_;
        }

        //- Read the forces data
        virtual void read(const dictionary&);

        //- Execute, currently does nothing
        virtual void execute();

        //- Execute at the final time-loop, currently does nothing
        virtual void end();

        //- Write the forces
        virtual void write();

        //- Calculate and return forces and moment
        virtual functionVariable calcfunctionVariable() const;

        //- Update for changes of mesh
        virtual void updateMesh(const mapPolyMesh&)
        {}

        //- Update for changes of mesh
        virtual void movePoints(const pointField&)
        {}
};


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

} // End namespace Foam

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#endif

// ************************************************************************* //


/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 1991-2010 OpenCFD Ltd.
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.
```

```
    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

\*---------------------------------------------------------------------------*/

#include "ExeFuncFunctionObject.H"

// * * * * * * * * * * * * * * * * Static Data Members * * * * * * * * * * * * * * * //

namespace Foam
{
    defineNamedTemplateTypeNameAndDebug(ExeFuncFunctionObject, 0);

    addToRunTimeSelectionTable
    (
        functionObject,
        ExeFuncFunctionObject,
        dictionary
    );
}

// ************************************************************************* //



/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 1991-2010 OpenCFD Ltd.
     \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Typedef
    Foam::ExeFuncFunctionObject

Description
    FunctionObject wrapper around forces to allow them to be created via the
    functions entry within controlDict.

SourceFiles
    forcesFunctionObject.C

\*---------------------------------------------------------------------------*/

#ifndef ExeFuncFunctionObject_H
#define ExeFuncFunctionObject_H

#include "ExeFunc.H"
#include "OutputFilterFunctionObject.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{
    typedef OutputFilterFunctionObject<ExeFunc> ExeFuncFunctionObject;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#endif

// ************************************************************************* //
```

List of references.

Alvardo, P. V. y. (2007), Design of Biomimetic Compliant Devices for Locomotion in Liquid Environments, PhD thesis, Massachusetts Institute of Technology.

Anderson, J. M. (1996), Vorticity Control for Efficient Propulsion, PhD thesis, Massachusetts Institute of Technology.

Anderson, J. M. and Chhabra, N. K. (2001), Maneuvering and stability performance of a robotic tuna, *in* 'Symposium on Stability and Manuveuverability: Anual meeting of the Scociety for Integrative and Comparative Biology', Chicago, Illinois, USA, pp. 118–126.

Anderson, J. M. and Kerrebrock, P. A. (2000), 'The Vorticity control Unmanned Undersea Vehicle (VCUUV); An Autonomouse Robot Tuna.', *Draper Technology Digest* pp. 63–70.

Ankarali, M. M., Arslan, O. and Saranli, U. (2009), A analytical solution to the stance dynamics of pasive spring-loaded inverted pendulum with damping, *in* '12th Int Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR' 09)', Istanbul, Turkey.

Aquarium, N. (2011), 'Black ghost knifefish'.
URL: *http://nationalaquarium.wordpress.com/*

Aracil, J., Gordillo, F. and Acosta, J. A. (2002), Stabilization of oscillations in the inverted pendulum, *in* L. Basañez and J. A. de la Puente, eds, 'Proceedings of the 15th IFAC World Congress, 2002', Vol. 15, IFAC, Barcelona, Spain.

Armada, M. A., Aliane, N., de Santos, P. G. and Jimenez, M. A. (1993), Controller tuning for a four-legged locomotion robot, *in* 'Proceedings of International Conference on Systems, Man and Cybernetics 'Systems Engineering in the Service of Humans", IEEE, Madrid, Spain, pp. 89–93.

Asano, F. and Xiao, X. (2012), Output deadbeat control approaches to fast convergent gait generation of underactuated spoked walker, *in* 'IEEE/SICE International Symposiup on System Integration', Fukuoka, Japan, pp. 265–270.

Astrom, K. J. and Furuta, K. (2000), 'Swinging up a pendulum by energy control', *Automatica* **36**, 287–295.

AUVAC (2013), 'Autonomous Undersea Vehicle Applications Center'.
URL: *http://auvac.org/explore-database/*

Bainbridge, R. (1958), 'The Speed of Swimming of Fish As Related to Size and to the Frequency and Amplitude of the Tail Beat', *Journal of Experimental Biology* **35**, 109–133.

Bal, S. and Kinnas, S. A. (2002), 'A bem for the prediction of free surface effects on the cavitating hydrofoils', *Computational Mechanics* **28**, 260–274.

Bandyopadhyay, P. R. (2005), 'Trends in Biorobotic Autonomous Undersea Vehicles', *IEEE Journal of Oceanic Engineering* **30**, 109–139.

Barrett, D. S. (1994), The Design of a Flexible Hull Undersea Vehicle Propelled by an Oscillating Foil, Master's thesis, Department of Ocean Engineering Massachusetts Institute of Technology.

Barrett, D. S. (1996), Propulsive Efficiency of a Flexible Hull Underwater Vehicle, PhD thesis, Department of Ocean Engineering Massachusetts Institute of Technology.

Barrett, D. S., Triantafyllou, M. S., Yue, D. K. P., Grosenbaugh, M. A. and Wolfgang, M. J. (1999), 'Drag reduction in fish-like locomotion', *Jounal of Fluid Mechanics* **392**, 183–212.

Barton, I. E. (1998), 'Comparison of simple and psio-type algorithms for transient flows', *International Journal for Numerical Methods in Fluids* **26**, 459–483.

Basar, T. and Bernhard, P. (2008), H_∞ *- optimal control and related minmax design problems*, 2nd edn, Birkhauser, Boston.

Beal, D. N. (2003), Propulsion through Wake Synchronization using a Flapping foil, PhD thesis, Massachusetts Institute of Technology.

Benyus, J. M. (1997), *Biomimicry*, Harper Collins, New York, USA.

Blidberg, D. R. (2001), The Development of Autonomous Underwater Vehicles (AUV); A Breif Summary, *in* 'IEEE ICRA', Vol. 4, Seoul, Korea.

Brower, T. P. L. (2006), Design of a Manta Ray Inspires Underwater Propulsive Mechanism For Long Range, Low Power Operation, Master's thesis, Tufts University.

Budiyono, A. (2009), 'Advances in Unmanned Underwater Vehicles Technologies: Modeling, control and guidance perspectives', *Indian Journal of Marine Sciences* **38**(3), 282–295.

Cai, Y., Bi, S. and Zheng, L. (2010), 'Design and Experiments of a Robotic Fish Imitating Cow-Nosed Ray', *Journal of Bionic Engineering* **7**, 120–126.

Childress, S. (1981), *Mechanics of Swimming and Flying*, Cambridge University Press, Cambridge, UK.

Cho, J. L. (1997), Electronic Subsystems of a Free-Swimming Robotic Fish, Master's thesis, Massachusetts Institute of Technology.

Choe, K. and Kim, K. J. (2006), 'Polyacrylonitrile linear actuators: Chemomechanical and electro-chemomechanical properties', *Sensors and Actuators A* **126**, 165–172.

Collar, P. G., Babb, R. J., Michel, J.-L., Brisset, L. and Kilpatrick, I. M. (1994), Systems research for unmanned autonomous underwater vehicles, *in* 'OCEANS '94. 'Oceans Engineering for Today's Technology and Tomorow's Preservation.", Vol. 1, Brest, France, pp. I/158–I/163.

Corfield, S. and Hillenbrand, C. (2003), *Technology and Applications for Unmanned Underwater Vehicles*, Taylor and Francis, London, chapter Defence Applications for Unmanned Underwater Vehicles, pp. 161–178.

Crespi, A., Latch, D., Pasquire, A. and Ijspeerd, A. J. (2008), 'Controlling swimming and crawling in a fish robot using a central pattern generator', *Autonomous Robots* **25**(1-2), 3–13.

Cubero, S. N. (2012), 'Design concepts for a hybrid swimming and walking vehicle', *Procedia Engineering* **41**, 1211–1220.

Dai, H. and Tedrake, R. (2012), Optimizing robust limit cycles for legged locomotion on unknown terrain, *in* 'Proceedings of the 51st IEEE International Conference on Decission and Control', Maui, Hawaii, pp. 1207–1213.

Danson, E. F. S. (2003), *Technology and Applications of Autonomous Underwater Vehicles*, Taylor and Francis, London, UK, chapter AUV Tasks in the Offshore Industry, pp. 127–138.

de Wit, C. C., Espiau, B. and Urrea, C. (2002), Orbital stabilization of underactuated mechanical systems, *in* L. Basañez and J. A. de la Puente, eds, 'Proceedings of the 15th IFAC World Congress,', Vol. 15, IFAC, Barcelona, Spain.

Der, N. P. V., Morsche, H. G. T. and Mattheij, R. R. M. (2007), 'Computation of eigenvalue and eigenvector derivitives for a general complex-valued eigensystem', *Electronic Journal of Linear Algebra* **16**, 300–314.

Dogangil, G., Ozcicek, E. and Kuzucu, A. (2005*a*), Design, Construction, and Control of a Robotic Dolphin, *in* 'IEEE International Onference on Robotics and Biomimetics', Shatin, Hong Kong, pp. 51–56.

Dogangil, G., Ozcicek, E. and Kuzucu, A. (2005*b*), Modeling, Simulation, and Development of a Robotic Dolphin Prototype, *in* 'IEEE International Conference on Mechatronics and Automation', pp. 952–957.

Dorato, P., Famularo, D. and Abdallh, C. T. (1999), 'Analytic phase margin design', *IEEE Transactions on Automatic Control* **44**(10), 1894–1900.

Douat, L. R., Queinnec, I., Garcia, G., Michelin, M. and Pierrot, F. (2011), h_∞ control applied to vibraion minimization of the parallel robot par2, *in* 'International Conference on Control Applications (CCA)', IEEE, Denver, CO, USA, pp. 947–952.

Doyle, J. C., Francis, B. A. and Tannenbaum, A. R. (1992), *Feedback Control Theory*, Dover Publications.

Engineering, B. (2009), 'GhostSwimmer'.
URL: *http://www.boston-engineering.com/media/pdfs/TS1166-Robo_Atherton.pdf*

Epstein, M., Colgate, J. E. and MacIver, M. A. (2006), Generating Thrust with a biologically inspired robotic ribbon fin, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems,', Beijing, China, pp. 2412–2417.

Festo (2007), 'Aqua Ray'.
URL: *http://www.festo.com/rep/en-us_us/assets/pdf/Aqua_ray_en.pdf*

Festo (2009), 'AquaPenguin; Brochure'.
URL: *http://www.festo.com/net/en-us_us/downloads/Download.ashx?lnk=29113/AquaPenguin_en*

Fish, F. E. (1996), 'Transitions from Drag-based to Lift-based Propulsion in Mammalian Swimming', *American Zoologist* **36**, 628–641.

Fish, F. E. (2002), Balancing Requiremants for Stability and Manuverability in Cetaceans, *in* 'Symposium on Stability and Manuverability; Annual Meeting of the Society fro Integrative and Comparative Biology', Vol. 42, Chicago, Illinois, USA, pp. 85–93.

Fish, F. E. (2006), 'The myth and reality of Gray's paradox: implication of dolphin drag reduction for technology.', *Bioinspiraion and Biomimetics* **1**, R17–R25.

Georgiades, C., German, A., Hogue, A., Liu, H., Prahacs, C., Ripsman, A., Sim, R., Torres, L. A., Zhang, P., Buehler, M., Dudek, G., Jenkin, M. and Milios, E. (2004), AQUA: an aquatic walking robot, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems,', Sendai, Japan, pp. 3525–3531.

Glover, K. and Doyle, J. C. (1988), State-space formulae for all atabilizing controllers that satisfy an H_{\infty}-norm bound and relation to risk sensitivity, *in* 'Systems & Control Letters 11', pp. 167–172.

GMBH, M. (2013), 'NiTinol The metal with a Mind'.
URL: *http://www.memory-metalle.de/*

Gray, J. (1936), 'Studies in animal locomotion: VI. the propulsive powers of dolphin', *Journal of Experimental Biology* **13**, 192–199.

Griffiths, G., Jamieson, J., Mitchell, S. and Rutherford, K. (2004), Energy Storage for long endurance AUVs, *in* 'Proc. Advances in Technologies for Underwater Vvehicles conference, IMarEST', London, UK.

Guo, J. (2006), *Advances in Unmanned Marine Vehicles*, The Institution of Electrical Engineers, Stevenage, UK, chapter 12; Guidance and control of a biomimetic-autonomous underwater vehicle, pp. 256–276.

Guo, S., Fukuda, T., Kato, N. and Oguro, K. (1998), Development of underwater microrobot using ICPF actuators, *in* 'Proceedins of the 1998 IEEE International Conference on Robotics and Automation', Leuven, Belgium, pp. 1829–1834.

Haertling, G. H. (1999), 'Ferroelectric ceramics; history and technology', *Journal of the American Ceramics Society* **82**(4), 797–818.

Harper, K. A., Berkemeier, M. D. and Grace, S. (1998), 'Modeling the Dynamics of Spring-Driven Oscillating-Foil Propulsion', *IEEE Journal Of Ocianic Engineering* **23**(3), 285–297.

Hirata, K. (2000), 'Fish Robot home page.'.
URL: *http://www.nmri.go.jp/eng/khirata/fish/*

Hirata, K., Takimoto, T. and Tamura, K. (2000), Study on turning performance of a fish robot, *in* 'First International Symposium on Aqua Bio-Mechanisms', pp. 287–292.

Hu, H. (2006), Biologically Inspired Design of Autonomous Robotic Fish at Essex, *in* 'BioIEEE SMC UK-RI Chapter Conference, on Advances in Cybernetic Systems', Sheffield, UK, pp. 3–8.

Hu, Q., Hedgepeth, D. R., Xu, L. and Tan, X. (2009), A Framework for Modeling Steady Turining of Robotic Fish, *in* 'IEEE International Conference on Robotics and Automation', Kobe, Japan, pp. 2669–2674.

Hu, Y., Wang, L., Yu, J., Huo, J. and Jia, Y. (2008), Development and Control of Dolphin-like Underwater Vehicle, *in* 'American Control Conference', Seattle, WA, USA, pp. 2858–2863.

Hur, M., Kang, T., Chan, W. L. and Choi, J.-M. (2009), 'h_∞ controller design of an ostraciiform swimming fish robot', *Indian Journal of Marine Science* **38**, 302–307.

iRobot Corporation (2010), 'Transphibian'.
URL: *http://www.irobot.com/gi/maritime/Transphibian*

Jha, N., Singh, U., Saxena, T. K. and Kapoor, A. (2011), 'Online adaptive contro for nonlinear processes under the influence of external disturbance', *International Journal of Artificial Intelligence and Expert Systems (IJAE)* **2**(2), 36–46.

Jordi, C., Michel, S. and Fink, E. (2010), 'Fish-like propulsion of an airship with planar membrane dielectric elastometer actuators', *Bioinspiration and Biomimetics* **5**, 26007.

Kato, N. (2000), 'Control Performance in the Horizontal Plane of a Fish Robot With Mechanical Pectoral Fins', *IEEE Journal Of Ocianic Engineering* **25**(1), 121–130.

Kawamura, A., Haneyoshi, T. and Hoft, R. G. (1988), 'Deadbeat controlled pwm inverter with parameter estimation using only voltage sensor', *Power Electronics, IEEE Transactions on* **3**(2), 118–125.

Keel, L. H. and Bhattacharyya, S. P. (1997), 'Robust, fragile, or optimal?', *IEEE Transactions on Automatic Control* **42**(8), 1098–1105.

Kim, B., Kim, D.-H., Jung, J. and Park, J.-O. (2005), 'A biomimetic undulatory tadpole robot using ionic polymerï£¡metal composite actuators', *Smart Materials and Structures* **14**, 1–7.

Kim, J.-H. and Oh, D.-C. (2007), 'Robust non-fragile h_∞ control for descriptor systems with parameter uncertains and time delay', *International Journal of Control, Automation, and Systems* **5**(1), 8–14.

Kinnas, S. A. and Fine, N. E. (1993), 'A numerical analysis of the flow around two and three dimensioal partially cavitating hydrofoils', *Journal of Fluid Mechanics* **254**, 151–181.

Kumph, J. M. (2000), Maneuvering of a Robotic Pike, Master's thesis, Massachusetts Intstitute of Technology.

Lee, H.-J., Jong, Y.-J., Change, L.-M. and Wu, W.-L. (2009), 'Propulsion stratergy analysis of high-speed swordfish', *Transactions of the Japan Society for Aeronautical and Space Science* **52**(175), 11–20.

Li, N., Ming, Y. and Dian-guo, X. (2012), Deadbeat predictive current control for pmsm, *in* 'Power Electronics and Motion Control Conference (EPE/PEMC)', IEEE, Novi Sad, Serbia, pp. LS6b.1–1 – LS6b.1–6.

Liang, J., Wang, T., Wang, S., Zou, D. and Sun, J. (2005), Experiment of Robofish Aided Underwater Archaeology, *in* 'IEEE International Conference on Robotics and Biomimetics', Shatin, China, pp. 499–504.

Licht, S. C. (2008), Biomimetic Oscillating Foil Propulsion to Enhance Underwater Vehicle Agility and Manuverability, PhD thesis, Massachusetts Institute of Technology.

Licht, S., Polidoro, V., Flores, M., Hover, F. S. and Triantafyllou, M. S. (2004), 'Design and Projected Performance of a Flapping Foil Auv', *IEEE Journal Of Ocianic Engineering* **29**(3), 786–794.

Lindsey, C. C. (1978), *Fish physiology*, Academic Press, New York, chapter Form, function, and locomatory habits in fish., pp. 1–100.

Liu, J. (2005), 'Essex robotic fish'.
URL: *http://cswww.essex.ac.uk/staff/hhu/jliua/*

Liu, J. (2007), Modelling and Online Optimization of Robotic Fish Behaviours, PhD thesis, University of Essex.

Logics, E. (2013), 'Subsea Glider with Fin Ray Effect'.
URL: *http://www.evologics.de/en/products/glider/index.html*

Long, J. H. J., Schumacher, J., Livingston, N. and Kemp, M. (2006), 'Four flippers or two? Tetrapodal swimming with an aquatic robot', *Bioinspiraion and Biomimetics* **1**, 20–29.

Long, J. H. and Nipper, K. S. (1996), 'The importance of body stifness in undulatory propulsion', *American Zoologist* **36**, 678–694.

Lorenz, R. D. and Valenzuela, M. A. (2012), Time optimal and loss minimizing deadbeat-direct torque and flux control or interior permanent magnet sysncronous machines, *in* 'Energy Conversion Congress and Exposition (ECCE)', IEEE, Masison, WI, USA, pp. 2568–2575.

Low, K. H. and Willy, A. (2006), 'Biomimetic Motion Planning of an Undulating Robotic Fish Fin', *Journal of Vibration and Control* **12**, 1337–1359.

Mason, R. (2003), Fluid Locomotion and Trajectory Planning for Shape-Changing Robots, PhD thesis, California Institute of Technology.

Mirfakhrai, T., Madden, J. D. W. and Baughman, R. H. (2007), 'Polymer artificial Muscles', *Materials today* **10**(4), 30–38.

Mittal, R. (2004), 'Computational Modeling in Bio-Hydrodymanics: Trends, Chalenges and Recent Advances', *IEEE Journal Of Ocianic Engineering* **29**, 595–604.

Mukherjee, R. (1993), 'Control of free-flying underactuated space manipulators to equilibrium manifolds', *IEEE Transactions on Robotcs and Automation* **9**(5), 561–570.

Nakashima, M. and Ono, K. (2002), *Neurotechnology for Biomimetic Robots*, MIT Press, Cambridge, USA, chapter Development and Experiment of Two-Joint Dolphin Robot, pp. 309–324.

Nakashima, M., Takashi, Y. and Ono, K. (2004), *Bio-mechanisms of Swimming and Flying*, Springer, chapter Three-Dimensional Maneuverability of the Dolphin Robot, pp. 79–92.

Ortega, R., van der Schaft, A. J., Mareels, I. and Maschke, B. (2001), 'Putting energy back in control', *IEEE Control Systems Magazine* pp. 18–33.

Pabst, D. A. (1996), 'Springs in Swimming Animals', *American Zoologist* **36**, 723–735.

Pedro, G., Suleman, A. and Djilali, N. (2003), 'A numerical study of the Propulsive efficiency of a flapping hydrofoil', *International Journal for Numerical Methods in Fluids* **42**, 493–526.

Rawlings, J. B. and Mayne, D. Q. (2009), *Model Predictive Control: Theory and Design*, Nob Hill Publishing, Madiosn, Wisconsin, USA.

Roper, D., Sharma, S., Sutton, R. and Culverhouse, P. (2013), 'Oscillation and dirrection contorol stratergies for a robotic fish', *Underwater Technology Journal* **31**(3), 67–76.

Rosen, M. W. (1959), Water Flow about a Swimming Fish, Master's thesis, University of California.

Saranli, U., Schwind, W. J. and Koditschek, D. E. (1998), Toward control of a multi-jointed, monoped runner, *in* 'International Conference on Robotics and Automation', Leuven, Belgium, pp. 2676–2682.

Schnerr, G. H. (2003), *Numerical Simulations of Incompressible Flows*, Worl Scientific Publishing, chapter Modeling and ccomputation of unsteady cavitating flows based on bubble dynamics, pp. 544–574.

Sfakiotakis, M., Lane, D. M. and Davies, J. B. C. (1999), 'Review of Fish Swimming Modes for Aquatic Locomotion', *IEEE Journal of Oceanic Engineering* **24**(2), 237–252.

Sharkh, S. M. A. (2003), *Technology and applications of Autonomous Underwater Vehicles*, Taylor and Francis, chapter Propulsion Systems for AUVs, pp. 109–125.

Shinjo, N. (2005), Investigations into the use of Shape Memory Alloy for Biomimetic Propulsion of Underwater Vehicles, PhD thesis, Florida Institute of Technology.

Shiriaev, A., Perram, J. W. and de Wit, C. C. (2005), 'Constructive tool for orbital stabilization of underactuated nonlinear systems: Virtual constraints approach', *IEEE Transactions on Automatic Control* **50**(8), 1164–1176.

Simons, D. G., Bergers, M. M. C., Henrion, S., Hulzenga, J. I. J., Jutte, R. W., Pas, W. M. G., van Schravendijk, M., Vercruyssen, T. G. A. and Wilken, A. P. (2009), A highly versatile autonomous underwater vehicle with biomechanical propulsion, *in* 'IEEE Oceans 2009-Europe', Delft, Netherlands, pp. 1–6.

Siochi, E. J., John B. Anders, J., Cox, D. E., Jegley, D. C., Robert L. Fox and Katzberg, S. J. (2002), Biomimetics for NASA Langley Research Center – Year 2000 Report of Findings From a Six-Month Survey, Technical memorandum, NASA Langley Research Center.

Skogestad, S. and Postlethwaite, I. (2007), *Multivariable Feedback Control; Analysis and Design*, 2nd edn, Wiley.

Spierts, I. L. Y. and Leeuwen, J. L. V. (1999), 'Kinematics and muscle dynamics of c and s-starts of carp (cyprinus carpio l.)', *Journal of Experimental Biology* **202**, 393–406.

Spong, M. (1996), Energy based control of a class of underactuated mechanical systems., *in* 'IFAC world Congress', San Francisco, United States, pp. 431–435.

Stefanini, C., Orlandi, G., Menciassi, A. and Ravier, Y. (2006), A mechanism for biomimetic acutationin lampray-like robots, *in* 'Biomedical Robotics and Biomechantronics', Pisa, Italy, pp. 579–584.

Stevenson, P. and Graham, D. (2003), *Technology and applications of Autonomous Underwater Vehicles*, Taylor and Francis, chapter Advanced materials and their influence on the structural design of AUVs, pp. 77–91.

Stommel, H. (1989), 'The Slocum Mission', *Oceanography* **2**, 22–25.

Suleman, A. and Crawford, C. (2008), 'Design and testing of a biomimetic tuna using shape memory alloy induced propulsion', *Computers & Structures* **86**(3-5), 491–499.

Suleman, A. and Crawford, C. (2009), *Underwater Vehicles*, INTECH, chapter Studies on Hydrodynamic Propulsion of a Biomimetic Tuna, pp. 459–486.
URL: *http://sciyo.com/books/show/title/underwater_vehicles*

Terzopoulos, D., Tu, X. and Grzeszczuk, R. (1994), 'Artificial fishes; autonomous locomotion, perception, behavior, and learning in a simulated physical world', *Artificial Life* **1**(4), 327–350.

Tondu, B., S.Mathe and Emirkhaniankibben, R. (2010), 'Low pH-range control of McKibben polmeric artificial muscle', *Sensors and Actuators A* **159**, 73–78.

Triantafyllou, M. S., Techet, A. H. and Hover, F. S. (2003), Review of Experimental Work in Biomimetic Foils, Technical report, Office of Naval Research (USA).

Triantafyllou, M. S. and Triantafyllou, G. S. (1995), 'An Efficient Swimming Machine', *Scientific American* **272**(3), 64–71.

Tumari, M. Z. M., Saealal, M. S., Ghazali, M. R., Zawawi, M. A. and Shah, L. H. A. (2012), 'H-infinity controller based on lmi region for flexible robot manipulator', *Resrarch Journl of Applied Science* **7**, 275–281.

Uyanic, I., Saranli, U. and Morgul, O. (2011), Adaptive control fo a spring-mass hopper, *in* 'IEEE International Conference on Robotics and Automation', Shanghai, China, pp. 2138–2143.

Wang, T., Wen, L., Liang, J. and Wu, G. (2010), 'Fuzzy Vorticity Control of a Biomimetic Robotic Fish Using a Flapping Lunate Tail', *Journal of Bionic Engineering* **7**, 56–65.

Wang, Y., Guan, Z.-H. and Wang, H. O. (2003), 'Feedback and adaptive control for the syncronization of chen system via a single variable', *Prhysics Letters A* **312**, 34–40.

Watts, C. M. (2009), A Comparison Study of Biologically Inspired Propulsion Systems for an Autonomous Underwater Vehicle, PhD thesis, University of Glasgow.

Weihs, D. (2001), 'Stabiity versus manuverability in aquatic locomotion', *Integrative and Comparative Biology* **42**(1), 127–134.

Wolfgang, M. J., Anderson, J. M., Grosenbaugh, M. A., Yue, D. K. P. and Triantafyllou, M. S. (1999), 'Near-Body Flow Dynamics in Swimming Fish', *Journal of Experimental Biology* **202**, 2303–2327.

Yu, J., Hu, Y., Huo, J. and Wang, L. (2007), An Adjustable Scotch Yoke Mechanism for Robotic Dolphin, *in* 'IEEE International Conference on Robotics and Biomimetics', Sanya, China, pp. 513–518.

Yu, J., Wang, M., Su, Z., Tan, M. and Zhang, J. (2011), Dynamic modeling and its application for a cpg-coupled robotic fish, *in* 'IEEE International conference on Robotics and Automation', Shanghai, China, pp. 159–164.

Yu, X. and Kaynak, O. (2009), 'Sliding-mode control with soft computing: A survey', *IEEE Transactions on Industrial Electronics* **56**(9), 3275–3285.

Zapaterio, M., Karimi, H. R. and Luo, N. (2011), 'Semiactive vibration control of nonlinear structures through adaptive backstepping techniques with h_∞ performance', *International Journal of System Science* **42**, 853–861.

Zhang, D., Hu, D., Shen, L. and Xie, H. (2008), 'Design of an artificial bionic neural network to control fish-robot's locomotion', *Neurocomputing* **71**, 648–654.

Zhang, W., Guo, S.-X. and Asaka, K. (2006), 'A New Type of Hybrid Fish-like Microrobot', *International Journal of Automation and Computing* **4**, 358–365.

Zhong, W. and Rock, H. (2001), Energy and passivity based control of the double inverted pendulum on a cart., *in* 'IEEE International Conference on Control Applications', Mexico City, pp. 896–901.

Zhou, C., Tan, M., Gu, N., Cao, Z., Wang, S. and Wang, L. (2008), 'The Design and Implementation of a Biomimetic Robotic Fish', *International Journal of Advanced Robotic Systems* **5**(2), 185–192.

Bound copies of published papers.