Abstract

The design of a missile system is a multi-disciplinary engineering activity that involves structural, aerodynamics, rocket propulsion, guidance, electronic, and closed-loop control engineering to name a few. In modern engineering practice, a systems engineering approach is utilised to manage the design of a missile, but this does not necessarily guarantee that the final design is optimal. The process may also be inefficient, requiring many iterations of design, prototyping and testing in order to achieve the required specifications.

In this thesis, multi-disciplinary optimisation frameworks are developed that target the aerodynamics and closed-loop control system of a supersonic tail-fin controlled missile. The aerodynamics and control system are highly coupled systems, but it is rare to see these subsystems optimised together in the literature. This is due in part to the computational requirements of the aerodynamic simulations and in part due to many control system design techniques that tend to treat the missile dynamics as immutable.

A model representing a supersonic tail-fin controlled missile is developed. The model utilises computational fluid dynamics (CFD) simulations in order to capture the aerodynamic behaviour and a state-space model for the dynamics of the missile. Control algorithms are utilised to perform the autopilot function of the missile. This model serves as a basis on which the aerodynamic shape and controller gains can be optimised.

Aerodynamic shape optimisation problems typically have large computational demands thus making them impractical to be used with global optimisation algorithms. The first optimisation framework developed is based on sample-based global extremum seeking. It is shown that under certain conditions, the convergence behaviour of CFD simulations can be viewed as plant dynamics and thus extremum seeking techniques can be applied to find the optimal aerodynamic shape. The results are a step toward obtaining globally optimal solutions within comparative computation times of gradient-based optimisers.

While useful for shape optimisation, the previous result would still struggle with combined aerodynamic shape and control optimisation problems. The next framework proposed is an adjoint-based gradient optimisation framework. The adjoint method has previously been utilised for static shape optimisation problems, but the result presented here is an extension for dynamic and controlled missile problems. The result shows that with appropriate time-scale separation between the actuator and flow states of the missile, the gradient of the cost function can be found with just two times the computational requirements of mapping the aerodynamic characteristics of the missile. This computational requirement is independent of the number of shape design variables and thus
shows its practicability. An example of a missile tail-fin profile and autopilot gain optimisation problem is presented.

There exists limitations of the adjoint based framework which prevent its use for certain missile geometries. Consequently, an implicit filtering framework is utilised in combination with the adjoint framework to cater for general missile geometries while still maintaining competitive computational speeds. This framework shows that general missile problems can be optimised without restriction. A number of optimisation examples involving a missile tail-fin profile and platform, missile nose cone and autopilot gains are presented.

Lastly, goal-oriented mesh adaptation which has often been utilised in the CFD community to refine their computational meshes is utilised in non-linear model predictive control (NMPC). Goal-oriented mesh adaptation is a result derived from the adjoint method. The control algorithm that is developed is computationally faster than the standard NMPC and therefore can be utilised in so-called “fast” systems.
Declaration

This is to certify that:

1. the thesis comprises only my original work towards the Ph.D.,

2. due acknowledgement has been made in the text to all other material used,

3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices

Kuan Wae Lee, May, 2018
Acknowledgements

I would like to thank my supervisor Professor Chris Manzie who encouraged me to pursue this research topic and for setting up the linkage project with BAE Systems Australia. I am grateful for the weekly meetings that were held early on in my candidature which helped immeasurably to structure my research. I am thankful for his patience and encouragement throughout my candidature.

I would like to thank Professor Andrew Ooi for his insights into fluid dynamics, in particular the computational aspects of the solvers. His support and review of my work is much appreciated.

Furthermore, thanks to Dr. William Moase whom I worked closely with on this linkage project and who was very thorough in reviewing my work and for always willing to help me to overcome the challenges I faced.

Special thanks Sei Zhen Khong who was a co-author on my first journal paper and who also happened to be my lecturer in Linear Systems. I am sure I would not have passed that subject without his teaching skills.

In the second year of my candidature, I was lucky enough to collaborate with Dr. Eric Kerrigan. I would like to thank him for providing his insights into implicit filtering optimisation methods. I found his excitement and passion for his research infectious.

I would also like to thank BAE Systems for their all of their support. Especially Dr. Jerome Vethecan and Dr. Paul Riseborough who helped me gain a good understanding of missile control.

Thanks to my fellow researchers in the control group - in particular - Chih Feng Lee, Vincent Bachtiar, Jalil Sharafi, Rohan Shekhar and Timothy Broomhead. It was always a fun and friendly work environment.

I would like to thank my parents and my sisters. They have been supportive of me throughout the entire process and for that I will always be grateful. Thank you to Prerana Sinha for proofreading my thesis. To So-Yeon Baik, thanks for loving me anyway. And to the rest of my family and to all my friends, thank you for your understanding.
## Contents

1 Introduction .................................................. 1
   1.1 Overview ............................................... 1
       1.1.1 Missile design ................................. 3
   1.2 Research goals ........................................ 4
   1.3 Thesis organisation .................................. 6

2 Literature Review .............................................. 9
   2.1 Overview ............................................... 9
   2.2 Optimisation in aerodynamic design ................. 10
       2.2.1 Gradient-free methods ....................... 11
       2.2.2 Gradient methods ............................. 13
   2.3 Optimisation in control design ..................... 14
   2.4 Nonlinear MPC and mesh adaptation ................. 16
   2.5 Summary ................................................ 19

3 System Modelling .............................................. 21
   3.1 Geometry models ...................................... 22
       3.1.1 Missile geometry ............................ 22
       3.1.2 Two-dimensional aerofoil profile ........... 23
       3.1.3 Three-dimensional missile part geometries .. 23
   3.2 State space model .................................... 25
   3.3 Aerodynamics model .................................. 27
       3.3.1 The compressible Euler equations .......... 27
       3.3.2 Semi-empirical aerodynamics model ........ 28
   3.4 Control system algorithms ........................... 29
       3.4.1 Three loop autopilot ......................... 30
       3.4.2 Non-linear model predictive control autopilot . 31
   3.5 Summary ................................................ 33
CONTENTS

4 Extremum Seeking for Shape Optimisation 35
  4.1 Introduction ........................................ 35
  4.2 Proposed approach ................................ 36
    4.2.1 The DIRECT algorithm for global optimisation .. 36
    4.2.2 The DIRECT algorithm for global extremum seeking .. 38
    4.2.3 CFD solver as a “plant”? ........................ 39
  4.3 Results ........................................... 43
  4.4 Conclusion .................................... 48

5 The Adjoint Method for Missile Optimisation 51
  5.1 Introduction .................................... 51
  5.2 Dynamic model .................................. 52
  5.3 Framework overview ................................ 56
    5.3.1 Cost function and design variables ............. 57
    5.3.2 Optimisation ................................ 59
    5.3.3 Cost and gradient calculation ................... 60
    5.3.4 Convergence ................................ 62
  5.4 Results ....................................... 62
  5.5 Conclusion .................................... 64

6 Implicit Filtering for Missile Optimisation 67
  6.1 Introduction .................................... 67
  6.2 Framework overview ................................ 68
    6.2.1 Cost function and design variables ............. 69
    6.2.2 Inner optimisation ............................. 70
    6.2.3 Outer optimisation .............................. 71
    6.2.4 Gradient calculation .......................... 73
    6.2.5 Convergence ................................ 74
  6.3 Results ....................................... 75
    6.3.1 Tail fin planform optimisation ................. 75
    6.3.2 Nose, tail-fin planform and profile optimisation .... 77
  6.4 Conclusion .................................... 80

7 Mesh Adaptation of Nonlinear MPC 83
  7.1 Introduction .................................... 83
  7.2 Direct collocation ................................ 84
  7.3 Goal oriented error estimation ..................... 86
  7.4 Proposed algorithm ................................ 89
  7.5 Results ....................................... 90
    7.5.1 Two-state system ............................ 90
    7.5.2 Missile pitch axis system .................... 93
CONTENTS

7.6 Discussion .................................................. 96

7.7 Conclusion .................................................. 97

8 Conclusion and Further Work ........................................ 99

8.1 Summary of contributions ...................................... 99

8.2 Contributions ................................................ 102

8.2.1 Journal papers ............................................. 102

8.2.2 Conference papers ......................................... 102

8.3 Further work ................................................ 103

A Gradient Methods .............................................. 105

A.1 The adjoint method .......................................... 105

A.2 Least squares finite difference method ...................... 106

B Optimisation Methods ......................................... 109

B.1 Implicit filtering ............................................. 109
List of Figures

1.1 Missile design process [26] showing the dependencies of subsystems and iteration of the design ........................................ 2

2.1 Model Predictive Control Scheme [8] .......................... 18

3.1 Tail-fin controlled missile geometry ............................ 22
3.2 Two-dimensional NACA0012 aerofoil ......................... 23
3.3 Three-dimensional missile nose and body .................... 24
3.4 Three-dimensional supersonic tail fin ......................... 24
3.5 Tail-fin controlled missile states ............................. 25
3.6 General Control Structure [75] ............................... 29
3.7 Classical “three loop” autopilot [69] ....................... 30
3.8 Missile pitch axis MPC autopilot ............................ 31

4.1 Proposed problem formulation ............................... 40
4.2 Residual error of fluid density states (Inset: showing initial divergence of residual error.) ............................ 43
4.3 Drag coefficient ($C_d$) surface with two design variables .... 45
4.4 Pressure coefficient $C_p$ across NACA0012 aerofoil surface. Upper surface (solid), lower surface (dashed) ....................... 45
4.5 Pressure coefficient $C_p$ across optimised aerofoil surface. Upper surface (solid), lower surface (dashed) ....................... 46
4.6 Aerofoil profiles ............................................. 47

5.1 Dynamic plant model ....................................... 52
5.2 Optimisation framework for the missile design problem .... 57
5.3 Continuity of cost function with respect to the geometry design variable ............................................ 58
5.4 Continuity of cost function with respect to controller parameters, $\tau_c$ and $\zeta_c$ ........................................ 59
5.5 Surface properties ........................................ 60
5.6 Step response of missile model .......................... 63
5.7 Optimisation convergence history .......................... 64
5.8 Tail-fin cross-section comparison (vertical scale stretched) .... 65

6.1 Comparison of the gradient of the lift coefficient with respect to wing span calculated via the adjoint method and finite differences 68
6.2 Optimisation framework for the missile design problem .......... 70
6.3 Residual error of CFD solver .................................. 72
6.4 Original and optimised shape of tail fin ......................... 76
6.5 Implicit filtering comparison with and without error control ... 77
6.6 Shape of original and optimised missile (body of missile omitted) 78
6.7 Cross sections of planform optimised tail fin shape (i.e. the chord shown here has not changed). The cross sections are labelled 1-5 as they move uniformly away from the missile body. .......... 79
6.8 Cross sections of planform and profile optimised tail fin shape. The cross sections are labelled 1-5 as they move uniformly away from the missile body. .......................... 79
6.9 Total optimisation time of nose and fin ........................ 80

7.1 Refinement relationship between $\tau$ and the states and inputs ... 85
7.2 Two state system ................................................. 91
7.3 Two state system discretisation level. The horizontal axis shows the discretisation level for the prediction horizon and the vertical axis shows the evolution of the prediction horizon at each controller time step. .......................... 92
7.4 Pitch axis missile using PI, SDRE and NMPC controllers ...... 96
7.5 Missile system discretisation level. The horizontal axis shows the discretisation level for the prediction horizon and the vertical axis shows the evolution of the prediction horizon at each controller time step. .......................... 97
## List of Tables

2.1 Missile control methods ........................................... 17

4.1 Comparison of optimisation results ................................. 48

6.1 Comparison of gradients for tail fin geometry calculated using various methods ............................................. 74

6.2 Optimal control parameters for original and optimised tail fin planform shape ............................................. 76

6.3 Optimal control parameters for original and optimised nose & fin shapes ..................................................... 78

7.1 Comparison of computational time and cost function evaluation for two state system (sec) ................................. 93

7.2 Comparison of computational time and cost function evaluation for missile example (sec) ................................. 95
Chapter 1

Introduction

1.1 Overview

The design of an autonomous missile system represents one of the most complex engineering tasks in modern aerospace engineering. The design must draw from many disciplines such as aerodynamics, propulsion, guidance and control, structural, safety and material engineering. The design must also meet objectives and constraints set by the customer’s requirements, where many of these objectives will conflict with each other and the constraints will also reduce the available design options.

Another important aspect of the design process is the management of employees, tools and processes. Gage [28] posits that the “[m]anagement of ... complex analysis modules in aerospace design is a challenging task”, and poor management and inefficiencies in the organisation can lead to missed opportunities.

A traditional approach to the engineering design task is to use systems engineering methodology. The aim of the methodology is to meet the requirements of the design in a systematic and manageable way within a defined time frame. At a high level, the process consists of defining the system requirements, establishing a baseline design by subdivision of the system into subsystems, and then continuously iterating the design until the design objectives are met. Some parts of the system must be designed first because of inherent dependencies. In practice, a performance “budget” is allocated to each subsystem at the start of the design and this then allows work to start across all subsystems simultaneously as each subsystem can be designed to the allocated “budget”.

The typical systems engineering process for a missile is illustrated in Figure 1.1. It shows the main subsystems and their dependencies, namely, the aerodynamics, propulsion, weight and control system.
INTRODUCTION

The systems engineering approach dictates that a system, due to its complexity, necessitates its division into parts. This division of labour is used despite the fact that many of the subsystems are tightly coupled and the impact of one with the other determines the performance of the overall system. Additionally, an unwanted side-effect of the systems engineering process is compartmentalised thinking [2], that is, for a department or group within an organisation to subconsciously pursue its own goals, rather than the organisational goals. Each discipline adopts their own models and methods in order to optimise their own processes and subsystems, with little regard for other groups in the organisation. This compartmentalisation can hinder the final design, and as stated in [14], “[t]he characteristics of knowledge that drive innovative problem solving within a function actually hinder problem solving and knowledge creation across
functions”

Furthermore, meeting all of the requirements may require many design iterations. This means the design process can take many man-hours to complete. The risk associated with failure can lead to a mind set that is highly reliant on existing or legacy designs.

Finally, the system engineering process provides no guarantee that the final design is locally or globally optimal in terms of any metric, as the design merely satisfies the requirements that were initially set.

Although no silver bullet, design optimisation, specifically multidisciplinary optimisation (MDO) is a tool that can aid in the design process. It is not the contention that MDO supplant the systems engineering process, but that it compliments the design process. MDO can aid in the design process by providing a roadmap from conceptual to final design. Indeed, its use is not limited to the early stages of design, but can also be used for refinement of detailed design. The caveat is that more representative and complex models have to be used and this will have an associated cost. It can also play an important role in breaking down domain “silos” that can develop within an organisation. Specifically, it can be used as a data-driven contract for disparate groups to agree to a design.

1.1.1 Missile design

A method to categorise missiles is by the roles that they play in the theatre of combat. Broadly speaking, these categories relate to the target and the launch platform of the missile. The four main categories are surface-to-surface (SSM), surface-to-air (SAM), air-to-air (AAM), and air-to-surface (ASM) missiles. The focus of this research is on short to medium range, surface-to-air interceptor missiles, however the methods developed here are general and can also be applied to the other categories.

SAMs were developed as a defensive countermeasure against incoming airborne threats. These surface-to-air missiles are characterised by their high manoeuvrability and high cruising speed as they are required to intercept high speed targets over a relatively short interception range. The requirement that the missiles have both high manoeuvrability and high speed are conflicting objectives, as the former requirement favours larger control surfaces while the latter demands smaller ones. Therefore, the system level design must consider the viable design space given the constraints and also the optimal design that maximises the mission objectives.

One of the primary drivers of the design of a missile, or any aerodynamic vehicle, is the geometry [91]. The aerodynamic behaviour, the physical size of the engine and fuel tanks, the warhead payload capacity and seeker sensor array
are all subsystems that depend on the shape and total internal volume of the missile. The total internal volume, in turn drives the size and shape of the wings, tail fins and control surfaces. This is the reason that the geometry is typically considered first [26].

Early examples of aerodynamic optimisation were based on analytical methods. For example, the Lanchester-Prandtl wing theory predicts that elliptical planform shaped wings produce minimal induced drag and this shape was used in the design of Britain’s Spitfire fighter aeroplanes [76]. In 1945, Lighthill [50] reported using analytical inverse field methods to determine the optimal shape of a aerofoil for a known pressure distribution. However, the application of analytical methods on the design of an aircraft is becoming increasingly rare, as closed-form theoretical results are often limited to very simple designs.

More recently, the trend toward simulation-based optimisation techniques has risen due to the wide adoption of computational fluid dynamics (CFD) models over analytical models. The benefits of CFD models are numerous, allowing for the simulation of detailed geometries, however these simulations require substantive computational resources to execute. Therefore, it is beneficial to use optimisation methods where the number of CFD executions are minimised or the computational load reduced in some way.

By itself, aerodynamic optimisation is a mature area of study. However less studied is MDO in the context of a missile system, where the control surfaces are coupled with a control system and the missile undergoes dynamic manoeuvres. This is because of the high computational cost of using CFD models which is exacerbated when employed in a dynamic environment. This motivates a new approach to MDO that attempts to reduce the computational resources required for CFD models and is one of the primary themes of this thesis. The findings in this research can also be applied to other computationally intensive optimisations, which this thesis will explore in a model predictive control (MPC) context.

1.2 Research goals

The scope of disciplines within aeronautical design are very large. To that end, this thesis will consider two of the missile’s subsystems, namely the aerodynamic geometry and the autopilot control subsystem. This is because both these subsystems are highly coupled and have a large influence on the manoeuvrability and speed of the missile.

The overall goal of this thesis is to improve the computational efficiency of the design of closed loop aerodynamic systems. This can encompass both global and local optimisation methods, and is required to consider the design operating over a finite time trajectory that exhibits representative
dynamic manoeuvres, rather than static operation. Computational efficiency is paramount, as high fidelity simulation models used within optimisation iterations can lead to infeasibly long solution times if naively applied. In a sense this mimics the trade off associated with using optimisation loops within an online control context such as model predictive control algorithms, and consequently some of the developments may be applicable to real-time controller deployment.

The research goals of this thesis are:

1. **Global optimisation of the geometry utilising computationally efficient methods**

Some of the main concerns in a design optimisation problem are the optimality of the solution, the accuracy of solution and the resources required to obtain the solution. These concerns depend on the optimisation strategy that is adopted and the fidelity of the model.

In general, global optimisation methods require the sampling of a large design space. This is problematic when coupled with CFD models where even a single sample requires large computational resources to process. Recent theoretical results in extremum seeking show that under certain conditions, there are guarantees regarding the solution of the global optimum when sampling the dynamic plant which has not yet reached steady state. This research aims to show that under similar conditions, the convergence of a numerical CFD solver to a steady state solution mimics traditional plant dynamics, and this then allows sampled data global extremum seeking to be used for shape optimisation problems. Additionally, a new algorithm called DIRECT-L is introduced and it is shown, under slightly stricter conditions, that the overall global optimisation time can be further reduced so that it is comparable with local optimisation techniques (at least for low-dimensional CFD problems).

2. **Local optimisation of the controller and geometry utilising computationally efficient methods**

For higher dimensional problems (and many aerodynamic and control optimisation problems fall into this category), a local optimal solution is usually a good starting point. In this thesis, the adjoint method, which has been used extensively in static aerodynamic optimisation problems is extended for use in a dynamic missile context. Again, the idea of partially converged CFD solutions is exploited to speed up the overall optimisation when it is coupled with noise tolerant gradient optimisers (such as implicit filtering). This leads to the main research goal of developing an optimisation framework for certain systems, such as a missile. The goal is to examine the appropriate gradient methods and exploit general
properties of the missile system in order to provide some guarantees in regards to the optimality of the design. The proposed framework exploits time-scale properties of the dynamic system model, closeness properties of partially converged iterative solutions of computational fluid dynamics models, noise-tolerant gradient optimisers and the continuous adjoint method.

3 To improve the computational speed of non-linear model predictive control
An interesting by-product of researching two disparate fields such as aerodynamic optimisation and control theory is that some methods which are well-known in one field, are not utilised in the other. Aerodynamic optimisation, as mentioned previously, faces unique challenges in terms of plant modelling and computation. The choice of discretisation of the flow field in the CFD model is driven by keeping the error uniformly small and the computational and memory load low as well. Mesh adaptation is one method in which the discretised mesh is refined in regions where the accuracy of the solution is demanded and coarsened where it is not. With adjoint mesh adaptation, the adaptation process is driven through a cost function. This research aims to explore the use of the adjoint mesh adaptation method within direct collocation non-linear model predictive control (NMPC). The discretisation of the plant model in NMPC shares similarities to the discretisation of the flow field in CFD, the cost function associated with adjoint mesh adaptation also lends itself naturally to the way NMPC problems are expressed. The goal is to see if mesh adaptation techniques can be used to speed up the computed solutions without compromising the accuracy and therefore allow NMPC to be applied to “fast” systems.

1.3 Thesis organisation

The structure of this thesis is as follows:
Chapter 2 is a brief review of the relevant literature in the area of optimisation in aerodynamic design, optimisation in control design and non-linear model predictive control. Chapter 3 provides an overview of the physical models and controller algorithms that are used in later chapters within the optimisation work. The main contributions of this thesis are presented from chapters 4 through to 7. In Chapter 4 the result of a global optimisation method for aerodynamic shape optimisation is presented. This is followed by an adjoint-based local optimisation framework for aerodynamic and control optimisation
1.3. **THESIS ORGANISATION**

In Chapter 5, the results of the previous chapters lead to Chapter 6 which is a generalised framework for aerodynamic shape and control optimisation. The final result in Chapter 7 utilises mesh-adaptation that is traditionally used in computational fluid dynamics and applies to non-linear model predictive control. In Chapter 8, the thesis contributions are summarised, further work is discussed and a summary of this thesis is presented.
Chapter 2

Literature Review

2.1 Overview

The role of design optimisation in the aerospace industry is long recognised and has a varied history. In [4], it was stated that over 8000 dissertations, journal articles, and reports have been written about the use of optimisation for aerospace applications. Apart from organisational barriers, there also exist technical barriers to the adoption of MDO in the design process. MDO poses additional challenges that single disciplinary optimisation does not. Foremost is the interaction between coupled subsystems [57, 6, 92], which are usually handled through creating smaller optimisation subproblems and defining the interaction through constraints. There is also a higher possibility of the inclusion of discrete and combinatorial design variables [73] in MDO as more and more subsystems are drawn into the design space. Systems that involve a control subsystem require the inclusion of time in simulation [3] in order to capture and evaluate the dynamic behaviour. This is a particular concern with CFD models, where time-dependent (unsteady flow) simulations are a few orders of magnitude greater in terms of computation compared to their steady-state counterparts. This indicates that the coupling and architecture of subsystems that are optimised affects the optimisers that can be effectively deployed.

There are a number of papers that have explored multidisciplinary optimisation for aerostructures, aerodynamics and control [52, 51]. However, these papers have a strong focus on aeroelasticity rather than the fluid and control aspects. For papers that deal specifically with missile design, there have been few examples. Suzuki [95] looked at the simultaneous optimisation of a spaceplane’s fixed trajectory and aerodynamic geometry. Tekinalp and Bingol [99] did a similar study, but also included propulsion design parameters into their optimisation process. Yang et al. [106] studied range maximisation of a canard-
controlled missile and compared the case of unguided and guided flight. In all of the missile studies mentioned, the aerodynamic models made use of the semi-empirical models rather than CFD models. These previous studies have only considered fixed trajectory open loop flight and there has been no emphasis on tailoring the optimisation algorithm to solve the design problem.

This chapter reviews optimisation in aerodynamic design, both global and local optimisation methods, with a focus on model fidelity and computational aspects in the context of an optimisation framework. Also covered is the optimisation in missile control design, in particular, the application adjoint methods to adaptive modelling for NMPC.

2.2 Optimisation in aerodynamic design

The use of CFD simulations to capture the aerodynamic properties is common practice in modern aircraft design. Prior to the 1960s, the study of these aerodynamic forces and moments revolved around theoretical and experimental fluid dynamics. The Navier-Stokes equations are used to fully describe the physical nature of fluid flow. However, there are no known closed-form solutions to the Navier-Stokes equations, which limits the application of purely theoretical methods to aircraft geometries. Experimental fluid dynamics were also used extensively in the past and continue to be a core part of aeronautical engineering to the present day. However, conducting full-scale tests of a missile is an expensive exercise, and it is unsuitable at the conceptual stage of the design process.

Advances in computing power and numerical algorithms led to a third approach of CFD. While the term CFD can be defined to encompass all algorithms that implement some kind of fluid model, in this case it is defined to mean the algorithms that implement the Navier-Stokes or Euler equations that describe fluid flow. This definition is made to differentiate it from other computational models to be discussed.

The flow around an aerodynamic body can be fully described by the Navier-Stokes equations. However, considerable computational resources are required in order to capture the turbulent nature of the flow at all length and time scales. This is the approach of direct numerical simulation (DNS), where a very fine grid resolution is required to capture all the details of the turbulent flow. Indeed, so demanding is the computational cost that the use of DNS is not used to simulate real engineering problems, which in general are high Reynolds number flows that contain a large range of time and length scales. Instead, DNS is restricted to low Reynolds number flows [62] and is used as a means of validating approximate models.
Less computationally expensive, approximate solutions have been proposed. Large eddy simulation (LES), as the name implies, directly solves for the larger turbulent scales, and models the smaller scales using sub-grid scale models. Reynolds-averaged Navier-Stokes (RANS) separates the flow equations into the mean and fluctuating parts, the result is that these equations contain extra terms called Reynolds stresses. The Reynolds stresses are modelled using turbulence models. Indeed, a large amount of fluid dynamics research is dedicated to the development of turbulence models. Of the numerous turbulence models in the literature, the Spalart-Allmaras, $k-\omega$ and shear stress transport SST $k-\omega$ turbulence models are considered to be most suitable for external aerodynamics [102]. Other simplifications are also made within CFD modelling, although the validity of solutions derived from these simplifications is restricted to a particular flow regime or neglect certain physical phenomena.

Computationally speaking and regardless of simplifications, CFD models are not cheap to compute, as the equations that describe fluid flow are non-linear in nature and fluid particles interact over large time and length scales. Therefore, it is beneficial that optimisation utilising CFD simulations somehow account for these computational shortcomings.

### 2.2.1 Gradient-free methods

Global optimisation methods are processes that find the global solution (minimum or maximum) of a function subject to constraints, where that function may contain many local optima. There are both non-deterministic global methods (including evolutionary algorithms [32] and particle swarm methods [41]) along with deterministic approaches (including the Piyasvskii-Shubert algorithm [89] and the dividing rectangles (DIRECT) algorithm [39]). Non-deterministic methods such as genetic algorithms (GA) have been applied to shape optimisation, for example in [72], a multiple-objective genetic algorithm was executed on a NEC-SX4 vector supercomputer with 32 compute nodes running in parallel. It is clear from this example that while using brute force with CFD models is an option, it will not scale well with the increase in design variables and should not be used indiscriminately [56, 36].

An approach to overcoming the computation cost of CFD is to utilise lower order modelling techniques to characterise the flow and thus the forces and moments acting on the body. As long as certain assumptions in these lower order models hold (e.g. low angles of attack, subsonic or supersonic flow fields) these methods give a reasonable approximation of the flow. In panel methods, fluid singularities, sources and doublets are attached to discrete segments or “panels” on the surface of an aerodynamic body to approximately model the
flow. Rom [81] showed that these methods could be extended to high angles of attack as long as the flow separation point was accounted for beforehand. Even with such limitations, codes such as PANAIR [18] or VSAERO [58] were often used by the major aerospace companies until the 1990s [23]. Kroo [46] combined the use of panel methods with genetic algorithms. In this way, the computational burden associated with sampling in genetic algorithms are reduced, but at the expense of accuracy in the final result.

Another low-order modelling approach is the semi-empirical method. The semi-empirical method is based on the theory that the forces and moments acting on the body can be calculated by combining the contribution of each geometric component. Correction factors for wing-body interference and downwash are also included. The forces, moments and correction factors for each part are either derived from empirical formulae or are based on simple aerodynamic theory (e.g. slender body theory, linear theory) [63]. For missile design, two such databases in broad use are Aeroprediction [64] and Missile DATCOM [103]. Sooy and Schmidt [93] compared the results obtained from DATCOM and Aeroprediction against wind tunnel data for various missile configurations. They found that the lift force and pitching moment compared well, however the predicted axial forces had errors of greater than 10%. For optimisation purposes, a drawback to the above approach is that one is then restricted only to the parts that are provided in the database.

Another route is to characterise the cost function directly by sampling various values of the cost for the design variables of interest. These sampled points are then used to generate, for example, an interpolated polynomial response surface from which the optimisation is conducted. These methods are known as surrogate modelling or response surface modelling. Surrogate modelling has been applied to the optimisation of a high speed civil transport (HSCT) aircraft [59]. The requirement of taking samples to build the surrogate model can still be computationally demanding.

The approaches discussed thus far require accurate evaluations of the cost function (i.e. accurate simulations of the model) at each step of the optimisation process. For CFD models, this means a fully-converged solution. Partial convergence of CFD solutions have been explored in [27] in construction of a response surface, however no justification is given.

There have been a number of developments in the extremum seeking literature since the early 2000s [47] including a number of key theoretical developments [98, 96, 61], that have spawned numerous applications (see for example the references contained in [97]). The key difference between optimisers and extremum seeking algorithms is the presence of plant dynamics. More recently, the extremum seeking methodology has been extended to include global optimisers
2.2. **OPTIMISATION IN AERODYNAMIC DESIGN**

[43, 44, 45], which use sampled data approaches and consequently broaden the class of problems that can be tackled as the reliance on local gradients may be removed.

At a high level, there are clear parallels between the goal of global extremum seeking on dynamic plants and shape optimisation using CFD solvers. Indeed, the convergence of steady state CFD models mimic those of dynamic plants approaching stable equilibria.

### 2.2.2 Gradient methods

While the ideal situation is that the global optimum be found, this is not necessarily practical for higher dimensional problems such as applications that couple CFD-based shape optimisation with controlled dynamics. In many cases, local optimisation methods will give a solution that is good enough for conceptual design or for refining an existing design.

Local optimisation methods rely on calculation of the gradient of the cost function. Hicks et al. [34] first applied the finite difference method to a CFD model of a two-dimensional wing profile. This work was followed by Hicks and Henne in the optimisation of an entire wing [33]. One issue of finite differences is that it becomes computationally expensive when the number of design variables is large, as many executions of the CFD simulator are required to calculate the gradient. Finite differences would require at least $N + 1$ simulations, where $N$ is the number of design variables.

An approximation of the gradient can be obtained using, for example, simultaneous perturbation stochastic approximation (SPSA) [94]. In [105], the SPSA algorithm is applied to the optimisation of an aerofoil, and it was shown that there was a reduction in the number of function evaluations when compared with simulated annealing.

Jameson [38] was the first to apply adjoint methods to calculate the exact gradient using just two simulations. A **primal** simulation is used to capture the behaviour of the physical system and an **adjoint** simulation is used to calculate the gradient of a cost function with respect to all of the design variables. Adjoint methods are able to calculate the gradient independent of the number of design variables which makes it highly attractive for CFD applications. Jameson’s work was initially concerned with just optimisation of geometry at a single steady-state, but has since been extended for multi-point optimisation [80], rotorcraft blade optimisation [21] and aerofoil optimisation with a predefined pitching motion [22]. There are two main derivations of the adjoint system, namely the **continuous** adjoint and **discrete** adjoint method, which differ only in the order in which the discretisation and differentiation are performed. In
either case, the gradients produced have been found to be the same when the
discretisation is sufficiently refined [68]. A limitation of the continuous adjoint
method is that it cannot handle design changes over sharp-edged geometries.
Gradient filtering and surrogate models have been proposed [54] to manage
the inaccuracy problem. The main advantage of the discrete adjoint method
is that the derivation of the adjoint system can be automated via application
of reverse-mode automatic differentiation. Automatic differentiation relies on
either source-code transformation or operator overloading. The discrete adjoint
does not suffer from the limitations of the continuous adjoint method for sharp
geometry. Practically speaking however, automatic differentiation suffers from
a trade off between execution speed and memory consumption [25], and still re-
quires some level of user interaction for the generated code to be computationally
efficient [20]. Even with limitations, the adjoint method is still considered to
be state-of-the-art for aerodynamic optimisation because of its computational
advantage over other methods.

A straightforward method of calculating the gradient for dynamic systems
utilising CFD models is to use time-dependent CFD models. However, this
means the adjoint system is also time-dependent, and both these simulations
together can be very computationally expensive. Under certain conditions, time-
scale separation [42] of the fluid states, and, the controller and actuator states
is possible. This allows the fluid states and CFD simulations (both primal and
adjoint) to be treated in a pseudo-steady fashion.

The main cost of steady CFD evaluation is time-stepping the simulation to
a steady state solution. There are some optimisation methods, such as implicit
filtering [40], that are capable of utilising inaccurate evaluations of the cost.
These methods allow the utilisation partially converged CFD (both primal and
adjoint) to calculate the gradient.

Time-scale separation of dynamic systems, partial convergence of high-fidelity
CFD models, implicit filtering and the continuous adjoint method can be ex-
loited in the formulation of an optimisation framework for the design of a
missile’s geometry and controller.

2.3 Optimisation in control design

The control of a tail-fin missile system is a non-trivial control problem. Due to
the position of the tail-fins, the dynamics of the missile are of a non-minimum
phase system. The missile body and control surfaces are also subjected to
nonlinear aerodynamic forces and there are time varying changes to it’s mass
due to the burning of the rocket motor propellant during flight. The missile must
operate over large time scales and must satisfy demanding and safety critical
2.3. OPTIMISATION IN CONTROL DESIGN

performance objectives.

The missile guidance and autopilot are two subsystems that form part of the missile controller. Various control schemes have been applied to the problem of missile guidance and autopilot control in the past. In general, the design of the guidance system and autopilot have been treated as separate subsystems. The implicit assumption that the autopilot and guidance loop are spectrally separated [88]. Most missile guidance systems are based on minimising the line of sight (LOS) of the missile and its target. The guidance law produces acceleration commands which are then passed on to the autopilot to translate the acceleration commands into the missile's actuator commands.

The first rudimentary missile controllers were devised for the V-2 ballistic missile by the Germans during World War II. It utilised gyroscopes for lateral stabilisation and an accelerometer for velocity integration. It is the first instance of a crude, but effective, inertial integrating control system [30]. Proportional-integral (PI) or proportional-integral-derivative (PID) based controllers are still broadly utilised due to their simplicity, and only requiring the tuning of a few gain parameters. There are rule based tuning methods, such as Ziegler-Nichols [110], that rely on the output step response of the plant. There have also been coupling of optimisers to PID controllers in order to tune the gains [1, 29].

The classical “three loop” autopilot [107] is in fact a nested PI controller with some tuning rules to adjust the gains. In order to cope with the non-linear dynamics of the missile, a widely adopted method is to extend the “three loop” autopilot by using gain scheduled control [82]. Feedback linearisation and dynamic inversion is another method that has been used in the design of missile controllers [15]. In this method, the nonlinear dynamics of the plant are compensated for which then allows the use of linear feedback control. Sliding mode controllers [100, 109] where the missile system is driven to a so-called “sliding surface” or subspace of the system and then regulated to the desired set point have also been utilised although there are complications when applying these to non-minimum phase systems.

Optimal control theory has also been used to develop linear quadratic regulator (LQR) based missile guidance laws [12, 83]. A class of solution methods for non-linear optimal control are based on direct methods where the control problem is transcribed into a non-linear program (NLP). Transcription via direct methods is categorised into single shooting [86], multiple shooting [11] and collocation methods [10]. In non-linear optimal control, the optimal state trajectory and control inputs over the finite horizon are pre-computed offline. This can be computationally expensive, so in order to keep the costs low while maintaining uniformly small errors, a number of adaptive refinement methods have been proposed. $h$-refinement refers to adaptation of the mesh segment widths.
These methods include minimising the maximum integration error \cite{9} and the use of density functions based on control and state histories from previous mesh refinement iterations \cite{108}. In contrast, $hp$-refinement refers to simultaneous adaptation of both the mesh segment widths and the polynomial degree of the approximating functions. These include pseudospectral methods \cite{17} and adaptive wavelet methods \cite{16}. The main issue with adaptive refinement is that the refinement process generally itself a non-trivial optimisation problem.

Nonlinear model predictive control has also been utilised for aerospace applications \cite{53,90}. Due to the fast dynamics associated with missile systems, these predictive controllers do not use the standard formulation of online optimisation, but instead they rely on suboptimal approximations such as Taylor series expansion of the output and or by limiting the control order to zero (i.e. keeping the control input as a constant over the prediction horizon). These NMPC controllers also cannot explicitly handle constraints nor have the control input as part of the cost function.

A summary of control methods applied to missile systems is shown in Table 2.3. The table also describes the issues associated with each method.

### 2.4 Nonlinear model predictive control and mesh adaptation

Model predictive control (MPC) \cite{55} is another area where optimisation theory is heavily utilised. MPC is a discrete-time, model-based algorithm where an optimisation problem predicting the future behaviour of the plant and controller inputs are solved at each time step. A survey conducted by Qin and Badgwell \cite{77} in 1999 recorded over 4000 instances where MPC was used to solve industrial control problems, with the majority – over 67% – of the applications being in the chemical processing sector. This is mainly because chemical processes tend to have “slow” dynamics, where the relatively resource hungry computations of the MPC algorithm have enough time to be computed. At the time, only 13 applications of MPC were recorded for the aerospace and defence sector.

The model predictive controller solves an open-loop finite horizon optimal control problem at each time step, which produces a sequence of optimal control inputs, $\{u(t|t), u(t+1|t), \ldots, u(t+N_p|t)\}$, where only the first input of the sequence, $u(t|t)$, is used and the remaining values are discarded. At the next time step, the above procedure is repeated and a new input, $u(t+1|t+1)$, is applied. An overview of the MPC scheme is illustrated in Figure 2.1. $N_m$ represents the input horizon (or control horizon) over which the MPC algorithm can manipulate the input and $N_p$ is the output horizon (or prediction horizon)
Table 2.1: Missile control methods

<table>
<thead>
<tr>
<th>Control Method</th>
<th>Description</th>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gain scheduled control (LQR, PID, etc.) [107, 82, 12, 87, 83]</td>
<td>Develop a series of linear input-output controllers tuned with different “controller gains” on the output signal</td>
<td>Linearisation is restricted about equilibrium points of the plant and there are no guarantees of closed-loop stability.</td>
</tr>
<tr>
<td>Feedback linearisation and dynamic inversion [15]</td>
<td>Invert nonlinearities of the system and apply as nonlinear static feedback</td>
<td>Full knowledge of the nonlinearities are required to apply inversion. The controller cannot reject un-modelled or unknown disturbances. It also cannot deal with control saturation.</td>
</tr>
<tr>
<td>Sliding mode control [100, 109, 5]</td>
<td>Drive the dynamic system to a sliding surface and keep it in the neighbourhood of the surface</td>
<td>Due to the sliding mode switching mode behaviour of the controller, if it is too aggressively tuned, the controller can lead to chatter or excitation of un-modelled dynamics. Requires a dynamic sliding surface for non-minimum phase systems such as tail-fin controlled missiles.</td>
</tr>
<tr>
<td>Nonlinear model predictive control (Taylor series expansion) [53, 90]</td>
<td>Output of the nonlinear system is approximated as a Taylor series expansion</td>
<td>Suboptimal controller (due to approximation) Control order is restricted to be less than the order of the Taylor series approximation. Control input is not part of the cost function. Cannot explicitly deal with constraints.</td>
</tr>
<tr>
<td>Nonlinear model predictive control (direct methods) [19]</td>
<td>Solve online optimisation problem using a discretised version of the continuous model.</td>
<td>Computationally intensive process an optimisation problem has to be solved at each time step. Optimisation problem is non-convex, so only local minimum is guaranteed. If online solver terminates early, then control input may not be the local minimum solution.</td>
</tr>
</tbody>
</table>
over which the algorithm predicts the future states of the system.

Depending on whether it is a linear or non-linear variant of the algorithm either a quadratic program (QP) [13, 78] or a non-linear program is solved. The main advantage of MPC is that hard constraints can be explicitly taken into account, however the downside is that it is computationally more intensive compared to other control algorithms. Non-linear model predictive control (NMPC) typically utilise direct transcription where the control problem is restated as an NLP and is applied in an iterative fashion for each time step.

In CFD, mesh adaptation seeks to reduce the computational resources required by refining or coarsening the 2D or 3D computational grid of the flow field by application of some heuristics. Goal-oriented mesh adaptation [101] is an off-shoot of the adjoint method. A unique aspect of this method is that the mesh is adapted with respect to an objective function, not the state or input dynamics of the system. As in the adjoint method, there are two main approaches, that is, continuous [7] and discrete [31]. It can be considered as a class of \( h \)-refinement methods.

There are overlapping concepts with goal-oriented mesh adaptation and NMPC. Both ideas utilise dynamic models of the real system and both have objective functions that measure a cost. Adjoint mesh adaptation and NMPC
differ in their outputs, the former produces a model that can be efficiently computed, while the latter produces an optimal control input for the system actuators.

2.5 Summary

In this literature review, it has been shown that while CFD models provide a high-fidelity and accurate simulation of the aerodynamic properties of aircraft geometries, these computational models are substantive in their computational needs. Various global and local shape optimisation approaches have been adapted to address this issue and the benefits and limitations of these approaches have also been discussed.

The co-design of aerodynamic shape and controller optimisation is even more computationally demanding and usually involve many design variables. In many cases CFD models are not utilised or it is assumed that the computational aspects are not a concern. For instance, a search on SCOPUS\textsuperscript{1} for “aerodynamic” and “shape optimisation” returned 1,562 articles, whereas the addition of “controller” to the search reduced the document count to just 23 articles. Exploiting high fidelity CFD models in the context of co-design while maintaining computational tractability is an area of research that has not yet been widely explored.

From a controller algorithm perspective, model predictive control also requires non-trivial computational resources. Similarities exist between goal-oriented mesh adaptation and the formulation of an NMPC using direct-collocation methods. The combination of these two areas can be potentially exploited for more efficient NMPC computation.

\textsuperscript{1}Search conducted on May 2018
LITERATURE REVIEW
Chapter 3

System Modelling

A model is a simplified (usually mathematical) abstraction of the real system. For the purpose of analysis and optimisation of the system, a model defines a set of parameterised design variables that can be varied and produces a figure of merit of the system performance. This chapter describes the aerodynamic models and controller algorithms that are utilised by the missile system. The parameterised design variables that are utilised in the optimisation are also outlined. Finally, the figures of merit used to evaluate the performance of the system under various scenarios are described.

A model of a missile can be vastly different for different engineering disciplines. For example, a thermal management engineer and a seeker array engineer will use different models when studying the shape of a missile’s nose. Even within the same discipline, an engineer may choose different models depending on its purpose.

In order to capture the behaviour of the missile with respect to aerodynamics and control, a suitable system model is required. This model must encapsulate a number of facets of the missile, namely:

- The geometry of the missile. The geometry is the originating factor on which many other properties of the missile depend upon. The size of the missile will determine the missile’s payload capacity and range. Its shape will determine the aerodynamic properties, stability, and manoeuvrability. The overall missile geometry is defined along with a number of detailed two-dimensional and three-dimensional geometries of missile parts for CFD evaluation.

- The missile dynamics under external forces. The missile’s trajectory is subject to externally applied forces from the rocket motor and aerodynamic forces and moments acting on the missile. A pitch-axis dynamics
model will be developed to evaluate the pitching motion performance of the missile.

- The aerodynamics of the missile. This will determine the lift, drag and moments experienced by the missile. The aerodynamics models are based on a Missile DATCOM empirical model and, two-dimensional and three-dimensional CFD models.

- The controller dynamics under feedback. The missile trajectory is controlled using the autopilot which acts on command and sensory inputs to produce actuator outputs. Two autopilots will be developed, a “three-loop” autopilot and an MPC autopilot.

3.1 Geometry models

3.1.1 Missile geometry

The geometry of the missile is one of the main determining factors of the aerodynamic properties. For this study, the missile geometry is based on a tail-fin controlled missile described in [93]. The missile geometry consists of a secant ogive nose, cylindrical body and four cruciform tail-fin control surfaces. The profile of the tail-fins are symmetrical diamond shaped. The dimensions of the missile are shown in 3.1 with measurements scaled to the missile body’s diameter $d$. The true length of the missile’s body has been truncated in the diagram to better show the shape of the nose and tail fins.

![Figure 3.1: Tail-fin controlled missile geometry](image)
3.1. GEOMETRY MODELS

3.1.2 Two-dimensional aerofoil profile

Figure 3.2 is the two-dimensional profile of a NACA0012 aerofoil. This aerofoil profile is widely used as a standard in evaluating CFD solvers. It will be used here as the initial geometry on which new designs are generated. Hicks-Henne functions [33] are utilised to deform the aerofoil in the normal direction. The function consists of deformation parameters $a_i$, which defines the magnitude of the deformation, $b_i$ is the location of the centre of the bump (control point), $c = 3.0$ controls the width of the bump and $\chi$ is the horizontal coordinate of the aerofoil. The function is defined as,

$$
z(\chi) = a_i \left( \sin \left( \pi \chi \log \frac{\log 0.5}{\log \frac{b_i}{c}} \right) \right)^c, 0 \leq \chi \leq 1$$

For the purpose of profile shape optimisation, two control points are defined on the upper surface centred at one-third and two-thirds along the chord length (defined as the tip to the tail of the aerofoil) and spanning two-thirds of the chord. The other two control points are located on the lower surface at the same chord locations.

3.1.3 Three-dimensional missile part geometries

Figure 3.3 shows the three-dimensional geometry of the missile’s nose and body. As only the pitch axis motion of the missile is being considered, the geometry of the nose and body is modelled with symmetry assumed across the vertical
plane. Deformation vectors $X_1$, $X_2$ and $X_3$ are defined on the surface of the nose. Each shape deformation variable locally deforms the shape via a free-form deformation (FFD) algorithm [85]. While the missile body is modelled, but does not form part of the shape optimisation. Figure 3.4 shows the three-

![Figure 3.3: Three-dimensional missile nose and body](image)

Figure 3.3: Three-dimensional missile nose and body

dimensional geometry of the missile’s tail fin and the associated deformation vectors $X4$ through to $X11$.

![Figure 3.4: Three-dimensional supersonic tail fin](image)

Figure 3.4: Three-dimensional supersonic tail fin
3.2 State space model

A detailed modelling technique for capturing the dynamics of the flow field and a manoeuvring missile is to model the flow field states and utilise a moving mesh to model the missile’s geometry as it changes over time. However, when taking design optimisation into consideration, such a method becomes impracticable with modern computational hardware.

Another simpler approach is to describe the dynamics of a missile by using a six-degree of freedom (6DOF) state space model. The state space model describes the attitude, angular rates and accelerations, as well as, the linear position, velocities and accelerations of an idealised point mass subject to external forces and moments. These external forces and moments are then provided through separate aerodynamic models.

In order to simplify the analysis (and to reduce the modelling effort), only the pitch motion of the missile is considered. The important states relating to pitching motion of the missile are shown in Figure 3.5, where $V$ is the vehicle speed, $\theta$ is elevation angle, $\delta$ is the fin deflection angle, $\alpha$ is the angle of attack and $\eta$ is the normal acceleration.

![Figure 3.5: Tail-fin controlled missile states](image)

The equations of longitudinal pitch motion will be derived below. The rate change of angle of attack $\dot{\alpha}$ for 6DOF non-linear general vehicle in flight is given by,

\[ m\dot{\alpha}V \cos \beta = -F_T \sin \alpha - L + mg \cos(\theta - \alpha) + mV(q \cos \beta - p \sin \beta) \]  \hspace{1cm} (3.2)

where, $L$ is the lift force, $m$ is the mass, $\beta$ is the side slip angle, $F_T$ is the thrust force, $g$ is gravitational acceleration and $p$ is the yaw rate and $q$ is the pitch rate. The rate change in elevation angle $\dot{\theta}$ is given by,

\[ \dot{\theta} = q \cos \phi + r \sin \phi \]  \hspace{1cm} (3.3)

where, $\phi$ is the roll angle and $r$ is the roll rate. The moment equation is given
by,

\[ I_{yy}q = (I_{zz} - I_{xx})pr - I_{xz}^2(p^2 + r^2) + M \]  

(3.4)

where, \( M \) is the pitching moment, \( I_{yy}, I_{zz}, I_{xx}, I_{xz} \) are the second moment of inertias along various axes.

Restricting motion to pitching only, then \( \beta = 0 \) and \( p = r = 0 \). Set \( \phi = 0 \) and assume small angle approximation such that \( \sin \alpha = 0 \), then the longitudinal equations are,

\[
\begin{bmatrix}
\frac{d}{dt} \alpha \\
\frac{d}{dt} \theta \\
\frac{d}{dt} q
\end{bmatrix} =
\begin{bmatrix}
\frac{-L(\alpha, \delta) + mg \cos(\theta - \alpha)}{mv} + q \\
\frac{q}{M(\alpha, \delta)} \frac{M}{I_{yy}}
\end{bmatrix},
\]  

(3.5)

Equation 3.5 is a longitudinal model.

The pitch motion equations can be further simplified to a two state model according to [70] and is given by,

\[
\begin{bmatrix}
\frac{d}{dt} \alpha \\
\frac{d}{dt} q
\end{bmatrix} =
\begin{bmatrix}
K_a M C_n(\alpha, \delta) \cos(\alpha) + q \\
K_q M^2 C_m(\alpha, \delta)
\end{bmatrix},
\]  

(3.6)

where, \( M \) is the Mach number and, \( K_a \) and \( K_q \) are model constants.

The tail-fin actuators consist of electromechanical actuators which have their own control system and dynamics. To simplify the modelling effort, the actuators are modelled with second order dynamics. For deflection control,

\[
\begin{bmatrix}
\frac{d}{dt} \delta \\
\frac{d}{dt} \delta_z \\
\frac{d}{dt} \delta_c
\end{bmatrix} =
\begin{bmatrix}
\delta_z \\
-\omega_a^2 \delta - 2 \zeta_\omega \delta_z + \omega_a^2 u \\
-\delta_c
\end{bmatrix},
\]  

(3.7)

where, \( u \) is the commanded input, \( \delta \) is the achieved fin deflection, \( \delta_z \) is the fin deflection rate, \( \zeta_\omega \) is the damping ratio and \( \omega_a \) is the undamped natural frequency of the actuator. An alternative to deflection control, is deflection speed control which is modelled as follows,

\[
\begin{bmatrix}
\frac{d}{dt} \delta \\
\frac{d}{dt} \delta_z \\
\frac{d}{dt} \delta_c
\end{bmatrix} =
\begin{bmatrix}
\delta_z \\
-\omega_a^2 \delta - 2 \zeta_\omega \delta_z + \omega_a^2 \delta_c \\
\delta_c
\end{bmatrix},
\]  

(3.8)

where, \( \delta_c \) is the commanded fin deflection and \( u \) is the commanded fin deflection rate.

Finally, a useful output to measure the manoeuvrability is the normal acceleration for the three-state pitch axis missile given by,

\[ \eta = \frac{V(q - \dot{\alpha})}{\cos \alpha}, \]  

(3.9)
3.3. AERODYNAMICS MODEL

or for the two-state pitch axis missile,

$$\eta_r = K_z M^2 C_n(\alpha, \delta)$$  \hspace{1cm} (3.10)

where $K_z$ is a model constant.

A figure of merit of the pitching missile’s performance that combines both a measure of its manoeuvrability and aerodynamic performance is,

$$J = \int_0^T e(t)^2 dt + w \int_0^T D(t)^2 dt,$$  \hspace{1cm} (3.11)

where $e(t) = \eta_c - \eta$, is the instantaneous tracking error between the commanded and achieved normal acceleration and $D(t)$ is the instantaneous drag force. A weighting constant $w$ is used to adjust the relative cost of each of component.

3.3 Aerodynamics model

The modelling of an aerodynamic body in a fluid is about characterising the forces and moments that act on the body as it moves through the fluid. Two distinct aerodynamic models will be utilised in this research. The first model is based on the compressible Euler equations. The second model is based on semi-empirical formulations.

3.3.1 The compressible Euler equations

The compressible Euler equations describe adiabatic and inviscid flow. The fluid flow variables are defined as,

$$U = \begin{pmatrix} \rho \\ \rho V \\ \rho E \end{pmatrix}$$  \hspace{1cm} (3.12)

where $\rho$ is the density of the fluid, $V$ is a vector of the local velocities and $E$ is the specific internal energy. The flux across the boundary $dS$ is,

$$F_i = \begin{pmatrix} \rho v_i \\ \rho v_i V + p \delta_i \\ \rho v_i (E + \frac{p}{\rho}) \end{pmatrix}$$  \hspace{1cm} (3.13)

where, $v_i$ is the local fluid velocity in the $i^{th}$ direction, $\delta_i = [\delta_{i1} \delta_{i2} \delta_{i3}]^T$ is a vector of Kronecker delta functions and $p$ is the pressure and if an ideal gas is assumed,

$$p = \rho (\gamma - 1) \left( E - \frac{1}{2} \rho \left( ||V||^2 \right) \right)$$  \hspace{1cm} (3.14)
This leads to the Euler equations, that is
\[
\frac{\partial U}{\partial t} = -\nabla \cdot \mathbf{F},
\] (3.15)
where, the convective flux is \( \mathbf{F} := [F_1 F_2 F_3]^T \).

The lift, drag and pitching moment can be determined by taking the pressure integral over the surface of the aircraft’s geometry. Lift, drag and moment are defined as,
\[
L = \int p(\mathbf{x})\mathbf{n}(\mathbf{x}) \cdot \mathbf{k} dS 
\] (3.16)
\[
D = \int p(\mathbf{x})\mathbf{n}(\mathbf{x}) \cdot \mathbf{i} dS 
\] (3.17)
\[
M = \int p(\mathbf{x})\mathbf{n}(\mathbf{x}) \times (\mathbf{x} - \mathbf{x}_0) dS 
\] (3.18)
where, \( p(\mathbf{x}) \) is the pressure at location \( \mathbf{x} \), \( \mathbf{n}(\mathbf{x}) \) is the unit normal on the surface \( dS \), \( \mathbf{i} \) is the direction parallel to the freestream, \( \mathbf{k} \) is the direction perpendicular to the freestream, and \( \mathbf{x}_0 \) is reference position for the moment lever arm.

The Euler equations do not take into account viscosity effects, but these effects can be approximated based on the surface area of the missile. The equations also do not model flow separation and so can only capture manoeuvres where the angle of attack is relatively small. Again, supersonic missiles do not utilise high angle of attack manoeuvres. Typically, for subsonic flow at low Reynolds numbers the Navier-Stokes equations are necessary to fully model the flow field. However, for flow in the supersonic regime, where the dominant effects include shock waves and pressure drag, these effects are well described by the compressible Euler equations and are sufficient for the purpose of this research [48, 79].

### 3.3.2 Semi-empirical aerodynamics model

Semi-empirical methods can also be used to model the aerodynamic forces and moments. An example from [70] of the aerodynamic coefficients for a missile model undergoing pitching motion are provided below, where,
\[
C_n = \text{sgn}(\alpha)\left(a_n|\alpha|^3 + b_n|\alpha|^2 + c_n(2 - M/3)|\alpha| + d_n\delta\right) 
\] (3.19)
\[
C_m = \text{sgn}(\alpha)\left(a_m|\alpha|^3 + b_m|\alpha|^2 + c_m(-7 + 8M/3)|\alpha| + d_m\delta\right) 
\] (3.20)

where, \( C_n \) and \( C_m \) are the lift and pitching moment coefficients, \( a_n, b_n, c_n, d_n, a_m, b_m, c_m, \) and \( d_m \), are empirical constants used to “fit” the model to a pre-existing set of data.
Semi-empirical models, as the name suggests, are built from a predefined empirically-sourced catalogue of known missile parts and their associated aero-
dynamic properties. These parts are then combined together to produce the overall aerodynamic properties of a particular missile configuration. Semi-
empirical methods are rather restrictive for shape optimisation as they are only able to utilise their database of predefined shapes. Furthermore, the source of aerodynamic properties of these shapes have to be derived either through wind tunnel experimentation or CFD simulation. The underlying assumptions around the combination of parts in these methods also relies on linear theory and so semi-empirical methods are only suitable for low angle of attack scenarios.

Programs like Missile DATCOM [103] are based on semi-empirical methods. Missile DATCOM also allows the importation of external aerodynamic properties from steady-flow CFD evaluations. This means that aerodynamic properties derived from CFD models can be easily combined with the semi-empirical models in the situation where full CFD simulations of all component parts are impractical.

3.4 Control system algorithms

The typical modern control architecture for a missile control is shown in figure 3.6. The control architecture consists of three main parts, namely the guidance filter, the guidance law and the autopilot [37]. The seeker is an array of sensors that captures the target’s information such as the line of sight (LOS) angle and relative range. This information is passed to the guidance filter where the relative position and velocity vector of the target are estimated. The target’s state estimate is then used by the guidance law to determine the missile’s acceleration commands. The autopilot translates the acceleration commands into actuator commands which drives the missile’s control surfaces and propulsion. The missile’s motion is sensed by the inertial navigation system and this information is fed back to the guidance filter and autopilot.

![General Control Structure](image)

**Figure 3.6: General Control Structure [75]**
The autopilot is responsible for translating the guidance commands into actuator commands. As this is the lowest level of controller that is directly connected to the actuator, the focus of this research will be optimisation of the autopilot.

### 3.4.1 Three loop autopilot

An acceleration-based control system, commonly known as a “three-loop” autopilot is given by,

\[
\delta_c = -K_R q + W_I x_c \\
\dot{x}_c = -q + K_A (-\eta + K_{DC} \eta_c)
\]

(3.21)

where, the commanded normal acceleration, \( \eta_c \), is the input, the fin-deflection is the output, and the missile’s achieved pitch rate and acceleration form the feedback signals.

The “three loop” autopilot can be represented as a block diagram and is shown in Figure 3.7. The inner-most rate loop corrects for angular rate, the structural stability loop is added to compensate for statically unstable missile dynamics, and the outermost loop corrects for accelerations. This autopilot architecture is repeated for the pitch, yaw and roll dynamics of the missile, where additional logic is sometimes implemented to compensate for the cross-coupling between the pitch and yaw motion.

A linear technique for tuning the four gains of the controller follow from [107], where an equilibrium operating condition is chosen, along with the desired time constant \( \tau \), damping \( \zeta \) and open-loop crossover frequency. The tuning strategy provides a static mapping from \( \tau, \zeta \) and \( \omega_{cr} \) to the gains \( K_{DC}, K_A, W_I \) and \( K_R \).

The three-loop autopilot is simple to implement and does not require much in terms of computational resources. However, it does not provide explicit
3.4. CONTROL SYSTEM ALGORITHMS

control over the enforcement of constraints nor does it provide any guarantees in relation to closed-loop stability.

### 3.4.2 Non-linear model predictive control autopilot

A non-linear model predictive controller can also be used for the autopilot. Figure 3.8 shows the main components of an NMPC autopilot. The commanded normal acceleration $\eta_c$ is used as the setpoint and the optimiser’s cost function is chosen as the setpoint error, i.e. $e(t) = \eta_c - \eta$. There may also be explicit state and input constraints imposed on the optimiser. At each time step of the controller, the optimiser utilises a model of the missile and produces a series of future fin deflection inputs and missile states for some time into the future, such that the cost function is minimised and no constraints are violated. The controller then applies only the first fin deflection to the plant and the entire process is repeated in the following time step.

The NMPC can use the same dynamic models developed previously for the pitch-axis missile, that is, equations 3.5, 3.6, 3.7 and 3.8. The output equations 3.9 and 3.10 can be used as a reference for the commanded setpoint.

The MPC can be implemented using the direct collocation algorithm. For reference, it is derived as follows. In direct collocation methods both the control input and state variables are discretised. Consider a continuous dynamical system of the form,

$$\dot{x} = f(x, u, t),$$

with state and input constraints of the form,

$$x \in X$$  \hspace{1cm} (3.23a)

$$u \in U$$  \hspace{1cm} (3.23b)

where $X$ and $U$ represent the admissible sets of the states and inputs respectively.
Associated with this dynamical system is a continuous cost function,

\[ J_c = \phi[x(t_f), t_f] + \int_{t_0}^{t_f} L[x(t), u(t), t] dt. \]  \hspace{1cm} (3.24)

**Assumption 3.1.** The following assumptions are placed on the dynamic system:

(i) \( f : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n \) is a twice continuously differentiable function on \( X \times U \).

(ii) \( X \subseteq \mathbb{R}^n \) is closed and simply connected set.

(iii) \( U \subseteq \mathbb{R}^m \) is a compact and convex set.

(iv) A steady state equilibrium \( f(x^*, u^*) = x^* \) exists, and this is desired operating condition to which the NMPC regulator will drive the system to. Furthermore the point \( (x^*, u^*) \) is in \( X \times U \).

(v) In order to guarantee stability, some form of terminal constraint is also applied to the NMPC.

In direct collocation methods, the continuous time system and cost function can be reformulated as a nonlinear program (NLP). The standard formulation of the NLP is derived below for reference.

Let the prediction horizon \( T_f := t_f - t_0 \) be discretised using index \( j \in \{0, 1, 2, \ldots, N_c\} \). Time along the prediction horizon is then given by \( t_j = [t_0, t_1, \ldots, t_{N_c}] \), where, \( t_0 < t_1 < \ldots < t_{N_c} \). Also \( t_f = t_{N_c} \).

The dynamical system (3.22) is approximated by a polynomial of degree \( s \) that satisfies over an interval \([t_{j-1}, t_j] \),

\[ p(t_{j-1}) = x_{j-1} \]  \hspace{1cm} (3.25)

\[ p'(t_k) = f(t_k, p(t_k)) \]  \hspace{1cm} (3.26)

where, \( t_k = t_{j-1} + c_k h \) for \( k = 1, \ldots, s \) and \( t_j = t_{j-1} + h \).

For example, an implicit (backward) Euler approximation written in collocation form has degree \( s = 1 \), and,

\[ p(t) = f(t_j, p(t_j))(t - t_{j-1}) + p(t_{j-1}) \]  \hspace{1cm} (3.27)

The collocation approximations for each time interval are then incorporated
into the NLP as equality constraints,

\[
R(x, u) = \begin{cases} 
  x_0 - x_{\text{initial}} \\
  x_1 - p(t_1) \\
  \vdots \\
  x_j - p(t_j) \\
  \vdots \\
  x_{N_c} - p(t_{N_c})
\end{cases} = 0
\] (3.28)

The state and input constraints can be similarly discretised,

\[
x_j \in X \quad (3.29a)
\]
\[
u_j \in U \quad (3.29b)
\]

as can the cost function,

\[
J_c = \phi(x_{N_c}, t_{N_c}) + \sum_{k=1}^{N_c} L(x_j, u_j, t_j) \Delta t.
\] (3.30)

**Remark 3.1.** Discretisation of the state constraints in this way means that there are only guarantees for the constraints at the sampling time and not at the intersampling times. For practical reasons, the possible (albeit mild) violations at intersampling times are ignored.

If Assumption 3.1 holds for the dynamic system, then the NLP can be solved using iterative optimisation procedures at each time interval, for example, Sequential Quadratic Programming (SQP) or Interior Point (IP) methods. At each time interval, only the first control input of the solution to the NLP is applied to the plant.

### 3.5 Summary

In this Chapter, it can be seen that there are various modelling methods of differing degrees of fidelity that can capture the behaviour of a missile system. For the purpose of systems engineering, both physical and control system algorithm models are not only utilised within a single engineering discipline, but must exchange information and interact with other models to produce a cohesive systems model. The parameterisation of the design variables in these models and a systems level figure of merit makes them useful in optimisation.
Chapter 4

Extremum Seeking for Shape Optimisation

4.1 Introduction

Optimisation of aerodynamic shapes using computational fluid dynamics approaches has been successfully demonstrated over a number of years, however the typical optimisation approaches employed utilise gradient algorithms that guarantee only local optimality of the solution. While numerous global optimisation techniques exist, they are usually too time consuming in practice.

In this chapter, the conditions under which the (partial) convergence of a numerical CFD solver to the vicinity of a steady state solution mimics a traditional dynamic plant are outlined, and this will then allow sampled data global extremum seeking to be used for shape optimisation problems in fluid dynamics. The approach is demonstrated on a simple example involving drag minimisation on a 2D aerofoil. Additionally, a new modification of the DIRECT algorithm (DIRECT-L) is introduced that makes use of knowledge of the cost function’s Lipschitz constant in order to reduce the amount of sampling. It is shown that combining this with the partial convergence result further reduces the total computational effort of the optimisation process. The theoretical foundation for this proposed framework is also provided.

This Chapter appeared in part in IEEE Transactions on Control Systems Technology (Vol. 23, Issue 6, Nov. 2015) as “Aerodynamic Shape Optimization via Global Extremum Seeking”.

35
4.2 Proposed approach

4.2.1 The DIRECT algorithm for global optimisation

In this work, the DIRECT algorithm is proposed for use in a shape optimisation context. The algorithm is described in detail in [39], but is summarised again here for completeness.

A minimisation problem utilising the DIRECT algorithm is written as,

$$\min_{\theta \in \Omega} J(\theta)$$  \hspace{1cm} (4.1)

where, $J : (\Omega \subset \mathbb{R}^m) \to \mathbb{R}$, is a Lipschitz continuous objective function defined over a fixed domain and compact subset of $\mathbb{R}^m$. Let $L$ be the Lipschitz constant of $J$, ie. $|J(\theta_u) - J(\theta_v)| \leq L||\theta_u - \theta_v||_2$ for all $\theta_u, \theta_v$ in $\Omega$. $\Omega$ is an orthotope and without loss of generality is taken to be a unit hyper-rectangle throughout this paper, that is,

$$\Omega := \{ \theta \in \mathbb{R}^m | \theta_i \in [0, 1] \subset \mathbb{R}, i = 1, 2, ..., m \}$$  \hspace{1cm} (4.2)

Let $q$ denote the iteration number of the algorithm and $k$ the total number of samples.

**Step 1** (Initialisation): Initialise $q := 1$, $k := 0$ and $u_0$ as the centre of $\Omega$. Let the output $y_1 = J(\theta_0)$ and $\hat{y}_0 = y_1$. Let $S = \{ \Omega \}$.

**Step 2** (Identification): Let $d_i$ denote the distance from the centre point $\theta_i$ of the $i^{th}$ hyper-rectangle to its vertices. The set of hyper-rectangles, $S$, is a set that contains all $j$ hyper-rectangles identified to be potentially optimal if there exists an $\bar{L} > 0$, where the following conditions hold:

$$J(\theta_j) - \bar{L}d_j \leq J(\theta_i) - \bar{L}d_i, \forall i$$  \hspace{1cm} (4.3)

$$J(\theta_j) - \bar{L}d_j \leq \hat{y}_k - \epsilon|\hat{y}_k|$$  \hspace{1cm} (4.4)

(Note that $\epsilon$ is a tuning parameter used to adjust the global/local search bias of the algorithm).

**Step 3** (New hyper-rectangles): For every $j \in S$ divide the $j^{th}$ hyper-rectangle with centre $\theta_j$ according to the following rules:

(i) First identify the set of dimensions, $I$ in which $j^{th}$ rectangle has maximum side length, and let $\delta$ equal one third of this maximum side length and $e_i$ be the $i^{th}$ unit vector for each $i \in I$.

(ii) Sample $J$ at the points $\theta_j \pm \delta e_i$ for all $i \in I$.

(iii) Divide the box $j$ containing $\theta_j$ into thirds along the dimensions in $I$ starting with the dimension with the lowest value of $w_i := \min \{ J(\theta_j + \delta e_i), J(\theta_j - \delta e_i) \}$
4.2. PROPOSED APPROACH

and continuing to the dimension with the highest \( w_i \).

**Step 4 (Update):** Set \( k = k + \Delta k \), where \( \Delta k \) is the number of new points sampled during the \( q \)th iteration. Set \( q = q + 1 \) and the current estimate of the optima to be

\[
\hat{y}_q := \min_{i=1..k} y_i
\]  

(4.5)

**Step 5 (Loop):** Return to Step 2.

**Step 6 (Termination):** In [44], it is proposed that the termination of the DIRECT algorithm occur when it is within some tolerance of the estimate of the lower bound of \( J \). Suppose that there is some knowledge of the Lipschitz constant and that a constant \( \hat{L} \) is (conservatively) chosen such that \( \hat{L} > L \).

After \( q \) iterations, for some hyper-rectangle \( j \) with centre at \( \theta_j \) and distance to vertices \( d_j \), an estimate of the lower bound of \( J \) on the hyper-rectangle is given by \( J(\theta_j) - \eta_j \), with \( \eta_j := \hat{L}d_j \). As the hyper-rectangles are further divided in successive iterations, the estimate of the lower bound approaches the minimum of \( J \). DIRECT terminates when \( \eta := \max_j \eta_j \) drops below some specified tolerance, \( \eta^* \), where it is assumed \( \eta^* < \hat{L}d_0 \).

**Remark 4.1.** If no information about the Lipschitz constant is available, then an alternative termination criteria could be applied, such as an upper limit on the number of samples.

The DIRECT algorithm searches the space in a dense manner, that is, all hyper-rectangles will eventually be identified as potentially optimal if DIRECT is allowed to run to infinity. This implies that only a subsequence of the sampling points converges to the global optimum. In particular, the following result can be found in [39].

**Proposition 4.1.** Let \( y^* := \min_{\theta \in \Omega} J(\theta) \). The output sequence \( \{y_k\}_{k=1}^{\infty} \) of the DIRECT algorithm contains a subsequence which converges to \( y^* \). It follows that \( \lim_{q \to \infty} \hat{y}_q \to y^* \), where \( \hat{y} \) is as defined in (4.5).

Along with the termination criteria in Step 6 of the DIRECT algorithm, knowledge of the Lipschitz constant (or a conservative estimate thereof) can also be used to reduce the sampling of the DIRECT algorithm. Conditions (4.3) and (4.4) are modified such that \( \hat{L} \leq \tilde{L} \) is enforced when identifying potentially optimal hyper-rectangles. This modification is denoted as DIRECT-L. On the other hand, the restriction imposed by the DIRECT-L modification means that certain hyper-rectangles will be identified as non-optimal and omitted from the search, which effectively leads to less sampling. Furthermore, the sequence of sampling points converges to the global optimum as detailed below.
Proposition 4.2. The output sequence \( \{y_i\}_{i=1}^\infty \) of the DIRECT-L algorithm has the property that \( \lim_{i \to \infty} y_i = y^* \).

Proof. The convergence proof follows from that of the unmodified DIRECT algorithm given in [39]. All new hyper-rectangles are generated by first dividing existing potentially optimal hyper-rectangles into thirds along the hyper-rectangle’s longest side lengths. Therefore a rectangle can only have side lengths of \( 3^{-k} \) for \( k = 1, 2, 3, \ldots \). After \( r \) divisions, a hyper-rectangle will have \( j := r \mod m \) sides of length \( 3^{-(k+1)} \) and \( m-j \) sides of length \( 3^{-k} \) where \( k := (r-j)/m \).

The distance from centre to vertices is given by
\[
 d := \left( j3^{-2(k+1)} + (m-j)3^{(-2k)} \right)^{1/2}. \tag{4.6}
\]
Thus \( d \) approaches zero as \( k \) increases. Choosing \( \hat{L} > L \) ensures that the global minimum lies inside a hyper-rectangle that is considered to be potentially optimal, therefore the sequence will converge to the minimum.

Remark 4.2. Note that for DIRECT-L, the sequence of sampling points no longer forms a dense subset in the search domain as in the case of DIRECT, but instead concentrates its search around the minimum.

4.2.2 The DIRECT algorithm for global extremum seeking

Consider a time-invariant MISO dynamic system subject to bounded input \( u \),
\[
 \dot{x} = \hat{f}(x, \theta) \\
 y = h(x, \theta) \tag{4.7}
\]
Let \( x(t, \theta, x_0) \) and \( y(t, \theta, x_0) \) denote the solutions of (4.7) for \( x \) and \( y \) respectively at time \( t \) given initial conditions \( x_0 \) and input \( \theta \in \Omega \).

Assumption 4.1. (Paraphrased from [44]). The dynamic plant possesses the following qualities for any constant \( \theta \) in the allowed range:
(i) for any \( \theta \), there exists \( x^* \) such that \( x = x^* \) is a global asymptotic stable equilibrium of (4.7) uniformly in \( \theta \);
(ii) the static map \( J(\theta) := \lim_{t \to \infty} y(t, \theta, x_0) \) is locally Lipschitz.

In the same manner as [45], let the inputs and output of the dynamical system described by (4.7) be subject to a zero order hold and sampling operation respectively, \( \theta(t) := \theta_k \) for all \( t \in [kT_c, (k+1)T_c] \) and \( y_k := y(kT_c) \) for all \( k = 1, 2, \ldots \), where, \( T_c \) is the waiting time between samples of the plant. By utilising this
4.2. PROPOSED APPROACH

sample and hold approach, it is then possible to couple the DIRECT or DIRECT-L algorithms described in section 4.2.1 as an extremum seeker for the plant.

The DIRECT and DIRECT-L algorithms with the modified termination constraint has the following convergence property:

**Theorem 4.1.** Given $\Delta > 0$ and $\mu > 0$, let the termination threshold $\eta^* \in (0, \mu)$ and $\theta_b$ be the minimising input at termination of DIRECT(-L). There exists a critical sample time $T_c^* > 0$ such that for any $||x_0|| < \Delta$, the DIRECT and DIRECT-L algorithms will terminate with,

$$||y^* - J(\theta_b)|| \leq \mu$$

for all $T_c > T_c^*$.  \hspace{0.5cm} (4.8)

**Proof.** The proof of convergence for DIRECT-L follows directly from the proof for DIRECT found in [44]. The main steps are summarised here. Let $\nu := (\mu - \eta^*)/2$. If Assumption 4.1(i) holds for any $||x_0|| < \Delta$ then for any sufficiently large $T_c$,

$$|y_k - J(\theta_{k-1})| \leq \nu \text{ for all } k = 1, 2, \ldots$$

(4.9)

Proposition 4.2 states that as the DIRECT-L algorithm iterates it converges to $y^*$, then from Lemma 9 of [44],

$$|J(\theta_b) - y^*| \leq 2\nu + \eta^* = \mu.$$  \hspace{0.5cm} (4.10)

As discussed in Remark 4.2, an appropriate choice of $\hat{L}$ ensures the convergence of DIRECT-L to the global minimum. \hfill $\Box$

4.2.3 CFD solver as a "plant"?

Computational modelling of steady fluid flow over a body, for example, by the RANS equations or Euler equations which are partial differential equations, is achieved through spatial discretisation via the use of finite element/difference/volume methods. This leads to a very large (although finite) number of discrete cells representing the states of the fluid at different spatial locations in the simulation domain. The states are fluid properties such as density and velocity. An example of this discretisation for an aerofoil problem is shown in Figure 3.2 consisting of just over 10,000 cells. Resulting from the discretisation process are coupled non-linear ordinary differential equations of the form,

$$\frac{dx_h}{dt} = g(x_h, G(\theta)), \hspace{0.5cm} (4.11)$$
where, $x_h$ represents the fluid states in each cell, and the “input” to the CFD solver is considered to be the mesh, $\theta$. The mesh is generated from $u$ which is a set of parameters describing the deformation of the aerodynamic body. This mesh generation process is treated as a mapping $\theta = G(\theta)$. The “plant” is then considered to be the combination of the mesh generation process and the CFD solver. Note that the flow is assumed to be steady so that the states and output measurement approach constant values. These ordinary differential equations (4.11) can be solved to steady state through iterative time-marching methods.

The residual error is a measure of the error of the states of the numerical solution to the true solution. The convergence of the CFD solver is typically measured by the reduction of the residual error of all (or subset of) the states, where the solver is set to terminate when this error has dropped below some predefined threshold. Let the set of states at the interface between the fluid and the surface of the body be designated $I_B$, i.e. $i \in I_B$ if and only if $x_{h,i}$ is a cell adjacent to the surface of the body. The aerodynamic drag (or lift) on a body is then estimated through the summation involving a function of the states within $I_B$:

$$y = \sum_{i \in I_B} C(x_{h,i}, G(\theta)) \quad (4.12)$$

A global extremum seeker utilising the DIRECT method is proposed in the manner shown in Figure 4.1. In this arrangement, there are two effective time scales being utilised. The CFD solver utilises variable step time-marching iterations to converge towards a steady state solution, and these iterations are nominally denoted to have a variable time step $T_{CFD}$. It is noted that the wall-clock time associated with any given $T_{CFD}$ may be highly variable. The global extremum seeker samples the numerical solver after a waiting time expressed

![Figure 4.1: Proposed problem formulation](image-url)
4.2. **PROPOSED APPROACH**

as a multiple $N$ of the CFD iterations, i.e. $T_{ES} = NT_{CFD}$. With this description of the combined mesh generator and CFD solver as a plant, the following properties are required:

**Property 4.1.** For all allowable $\theta$, the mapping $G(\theta)$ is locally Lipschitz. Moreover the sets of indices of the mesh points on the fluid-surface interface ($I_B$) and on each boundary of the numerical domain are independent of $\theta$.

**Property 4.2.** The numerical CFD solver satisfies:

i. Let $\mathcal{D} := \{G(\sigma) : \sigma \in \Omega\}$. For all $\theta \in \mathcal{D}$, $\exists x^*_h(\cdot)$ such that $x^*_h(\theta)$ is an equilibrium of (4.11). Moreover, the equilibrium solution is semiglobally asymptotically stable uniformly in $\theta$.

ii. $l(\theta) := \sum_{i \in I_B} C(x^*_h, i, \theta)$, is locally Lipschitz.

iii. $x^*_h$ is initialised at each extremum seeker sample time, $T_{ES}$, with the final values of the states of the previous CFD solution.

Property 4.1 simply ensures that the mesh generation process is continuous with respect to the parameters describing the aerodynamic body. Notably, Property 4.1 prevents meshing techniques that add or remove cells from the numerical domain when the geometry is changed and this is the main reason for the use of mesh deformation rather than re-meshing for each new design $\theta$. Furthermore, for large $\theta$ the mesh deformation process can result in cells with large aspect ratios which can lead to numerical instability in the solver. There exists a number of pre-processing tools to test the geometric “health” of a mesh and these can be employed to check that Property 4.1 is met. Alternatively, one can bound the range of allowable $u$ to conservative values. Property 4.2 is related to the behaviour of the CFD numerics. Specifically, Property 4.2(i) is analogous to requiring stability of the dynamic plant, and ensures that Assumption 4.1(i) is satisfied. Property 4.2(ii) (in combination with Property 4.1) is analogous to requiring the steady-state behaviour of the dynamic plant to be Lipschitz, and ensures that Assumption 4.1(ii) is satisfied. Property 4.2(iii) essentially requires “hot starting” of each CFD run with the final state of the previous CFD run. This ensures the CFD numerics behave like a traditional dynamical system by avoiding a periodic resetting of the plant’s states at each sampling instant.

Although it is a comparatively straightforward task to ensure the mesh generation process and CFD solver satisfy Properties 4.1 and 4.2(iii), it is a less simple task to ensure Properties 4.2(i) and 4.2(ii) are satisfied. To achieve this, the stability of the solver is considered. Steady flow CFD solvers are built by discretising the *weak form* of the steady RANS or Euler equations. Stability of these non-linear equations can be shown by application of linear methods, such as von Neumann and matrix methods [35] or non-linear methods, such as the total variation diminishing (TVD) method [49]. Let $x_{h,0}$ be the state for which
the CFD solver is initialised. If $|x_{h,0}(\theta) - x_{h}^*(\theta)| \leq \Delta$, and if the solver is stable, then there exists a $\beta (\cdot, \cdot) \in K\mathcal{L}$ such that,

$$|x_{h}(t) - x_{h}^*| < \beta(|x_{h}(0) - x_{h}^*|, t)$$

(4.13)

In other words, (4.13) states that the solution of the CFD solver is semiglobally asymptotically stable [42, Lemma 4.5]. In addition to the stability, the solver must be consistent, that is, the discretisation error tends to zero, and convergent, the approximate solution tends to the exact solution, as the mesh, $\Delta x$, and the time-step, $\Delta t$, both tend to zero. It is assumed that the mesh and time-step have been sufficiently resolved to ensure that the aforementioned properties are satisfied.

**Proposition 4.3.** The cascade of mesh generation and CFD solver, ie. (4.11), can be modelled as (4.7). Furthermore it satisfies Assumption 4.1.

**Proof.** This is achieved via satisfaction of Properties 4.1 and 4.2 by the means discussed above.

The control scheme (ie. the global extremum seeker) takes a finite amount of time to find an input to the system which drives the steady-state behaviour into a neighbourhood of a global extremum. The size of this neighbourhood can be reduced at the expense of an increase in the waiting period $T_{ES}$ (through an increase in $N$). Coupled with a change in the termination criterion of DIRECT, a further reduction in the size of this neighbourhood can be obtained, but both these changes are at an expense of additional CFD computation.

**Proposition 4.4.** Let $\theta_{min}$ be the geometrical parameters that minimise $J$, and $\theta_{ES}$ be the optimal geometrical parameters resulting from the DIRECT-L optimisation. Then,

$$\forall \varepsilon > 0, \exists T_{c}^{*} \text{ s.t. } |J(\theta_{min}) - J(\theta_{ES})| < \varepsilon, \forall T_{ES} > T_{c}^{*}$$

(4.14)

**Proof.** In view of Proposition 4.3, the claim follows directly from Theorem 4.1.

Figure 4.2 shows the residual error plotted against the time-marching iterations of the CFD solver. The dynamics of the CFD code during stabilisation may lead to some initial divergence in the residual error. The choice of the number of CFD iterations may need to be conservative, as it is required not only for the convergence of the initial nominal system, but also for every subsequent shape perturbation within the optimisation process. Figure 4.2 is also useful in clearly highlighting the motivation for early termination of the CFD solver.
4.3 Results

To demonstrate the proposed approach, the drag minimisation of an aerofoil at an angle of attack of 1.25 degrees and a Mach number of 0.8 is considered. The CFD solver package used here is the SU2 open-source CFD code [74]. In addition to the CFD solver, the package incorporates mesh deformation and adjoint subroutines. The initial mesh, as shown in Figure 3.2, was also provided with the package and is assumed to be of good quality. Consider the Euler equations stated in (3.15) with dimensionality of two. Application of spatial discretisation, such as the finite volume method, to (3.15) results in a spatially coupled system of ordinary differentiation equations of the form,

$$\frac{dx_h}{dt} + \frac{1}{v_h} \int_{S_h} F \cdot n(x) dS = 0,$$  \hspace{1cm} (4.15)

(at around 10-50 iterations). Given the number of states involved in each iteration of the CFD code, there is a large computational burden associated with each iteration. Waiting for full convergence (ie. approximately 300 iterations) of the CFD at each iteration of the extremum seeker therefore induces a significant additional computational expense, particularly when the global search method necessarily will explore wide regions of the input space that may later be dismissed as non-viable.

Figure 4.2: Residual error of fluid density states (Inset: showing initial divergence of residual error.)
where \( x_h \) is the volume averaged solution of \( U \) for each cell, \( v_h \) is the cell volume, \( \mathbf{n}(\mathbf{x}) \) is the surface normal vector for each face and \( S \) is the total cell surface area. Performance measures of interest to an aerodynamicist include the aerodynamic drag coefficient which is defined as,

\[
C_d := \frac{1}{A} \int C_p \sin(\alpha + \phi) dA,
\]

(4.16)

with,

\[
C_p := \frac{p - p_\infty}{\frac{1}{2} \rho_\infty v_\infty^2},
\]

(4.17)

where \( \rho_\infty \) is the free stream density of the fluid, \( v_\infty \) is the free stream velocity, \( A \) is the surface area of the aerofoil, \( p_\infty \) is the free stream pressure, \( \alpha \) is angle of attack and \( \phi \) the angle of the surface normal relative to the vertical axis. Note that the discretised Euler equations (4.15) is the same form as (4.11) and the drag coefficient (4.16) is estimated by (4.12) defined previously. The drag coefficient is the cost to be optimised.

The initial geometry was taken to be the NACA0012 aerofoil and is deformed using Hicks-Henne functions defined in (3.1). Two control points are located on the upper surface centred at one-third and two-thirds along the chord length and spanning two-thirds of the chord. The other two control points are located on the lower surface at the same chord locations. The input is defined by \( \theta := [a_1, a_2, a_3, a_4] \).

To illustrate the presence of multiple local minima for these types of aerodynamic problems, the drag coefficient surface, where only two control points on the bottom surface of the aerofoil are allowed to vary, is plotted in Figure 4.3. The figure shows a Lipschitz continuous surface with a main basin in the middle where the drag is minimal and also a local minimum toward the top right. While the region of attraction of this local minimum is small, one cannot eliminate the possibility of becoming entrapped in one with gradient-based optimisers.

The plot of the pressure coefficient against the chord length (where \( \chi = 0 \) is the leading edge and \( \chi = 1 \) is the trailing edge) for the NACA0012 aerofoil is shown in Figure 4.4. The drag coefficient was calculated to be \( C_d = 0.021568 \) and from the plot it can be seen that the main contribution to the pressure drag is due to the stronger shock that developed over the top surface of the aerofoil indicated by the \( C_p \) values along the leading two-thirds of the chord.

To identify the best possible solution, and in order to benchmark the performance of the different approaches, full time scale separation between the plant and optimiser was introduced by waiting for the CFD to fully converge to a residual error tolerance of \( 1 \times 10^{-11} \) before updating the optimiser via
Figure 4.3: Drag coefficient ($C_d$) surface with two design variables

Figure 4.4: Pressure coefficient $C_p$ across NACA0012 aerofoil surface. Upper surface (solid), lower surface (dashed).

The DIRECT algorithm. In addition, the adjoint method was also considered as a “best-in-class” gradient-based local optimisation method in place of the DIRECT method. The aerofoil was initialised to the original shape with the next input parameters determined by using a sequential least-squares optimiser...
and the gradient at each optimisation step provided by the adjoint method. In implementing the global extremum seeking approach with partial convergence of the CFD, three different waiting times, $T_{ES}$ were considered. These were 10, 25 and 50 iterations of the CFD numerical solver. Lastly, the DIRECT-L optimiser is also tested with 10, 25 and 50 iterations of the CFD numerical solver and utilising a conservative value of 0.025 for the Lipschitz constant. The input domain of the DIRECT and DIRECT-L algorithms is the set $\theta = \{a \in \mathbb{R}^4 \mid a_i \in [-0.05, 0.05], i = 1, 2, ..., 4\}$.

Figure 4.5 shows the resulting $C_p$ values of all the optimisation strategies. In all cases, the shock on the upper surface is weakened, however as there are only four control points, there are structural limits to the shape deformation, and so the $C_p$ values (and thus the drag) can only be altered to a certain extent. The figure also shows the variation in $C_p$ values along the lower surface of the aerofoil for each optimisation strategy. Note that for the partially converged optimisation results, the values in the figure are produced by re-running the CFD solver until it is fully converged for the optimised shape that was found by the optimiser. This shows the “true” value of the pressure (and drag) coefficient as opposed to that observed by the optimiser due to early termination.

![Figure 4.5: Pressure coefficient $C_p$ across optimised aerofoil surface. Upper surface (solid), lower surface (dashed).](image)

Figure 4.6 shows the shape of all the optimised aerofoils, including the original NACA0012 aerofoil. There is partial asymmetry in the optimised aerofoil shape, primarily a consequence of the non-zero angle of attack. It can be seen
that the upper surface of all the optimised aerofoil shapes are practically the same, and the variation in shape is from the lower surface. This is indicative that the cost function with respect to the lower control points is similar to that shown earlier in Figure 4.3.

To quantify the results more thoroughly, the drag coefficients for all solutions are presented in Table 4.1. These include:

• the value of $C_d$ at termination of the optimisation, when the CFD has only run for $N$ iterations at the optimal geometry.

• the norm of the difference between the input parameters of each method compared to the optimal input parameters determined by the fully converged DIRECT optimisation.

• the “true” value of $C_d$ after running the CFD to convergence for the resulting optimal geometry of each method.

It is clear that the results predicted by Proposition 4.4 are maintained - i.e. the closeness of the extremum seeking solution to the global optimum can be controlled through the waiting time. It is also apparent that the proposed global extremum seeking approach does not converge particularly close to the global optimum when the waiting time is too short, as for $T_{ES} = 10T_{CFD}$. 

Figure 4.6: Aerofoil profiles
Table 4.1: Comparison of optimisation results

| Method      | N (number of CFD iterations) | k (number of samples) | Total time | $C_d$ measured at $T_{ES}$ | $|\theta - \theta^*|_2$ | True $C_d$ |
|-------------|-----------------------------|-----------------------|------------|---------------------------|--------------------------|-------------|
| Adjoint     | $\rightarrow \infty$       | 60                    | 34m        | n.a.                      | 0.0023                   | 0.006697    |
| DIRECT      | $\rightarrow \infty$       | 1809                  | 10hr 53m   | n.a.                      | 0                        | 0.006511    |
| DIRECT      | 50                          | 463                   | 1 hr 2m    | 0.006621                  | 0.0061                   | 0.006750    |
| DIRECT      | 25                          | 369                   | 41m        | 0.006672                  | 0.0083                   | 0.006928    |
| DIRECT      | 10                          | 285                   | 27m        | 0.007500                  | 0.0112                   | 0.007875    |
| DIRECT-L    | 50                          | 281                   | 37m        | 0.006782                  | 0.0081                   | 0.006832    |
| DIRECT-L    | 25                          | 203                   | 23m        | 0.006742                  | 0.0100                   | 0.006992    |
| DIRECT-L    | 10                          | 275                   | 26m        | 0.007440                  | 0.0114                   | 0.007762    |

For this example, it can be seen that the adjoint solution is closest to the fully converged DIRECT solution. There are however no guarantees that the adjoint or any other gradient-based method will converge to the global minimum if it is initialised from another point. As mentioned in the introduction, the adjoint method is known to produce inaccurate gradients, and this can also impact on the convergence to the optimum of the gradient-based optimiser. Differences in the optimised drag values may be more pronounced for different geometries or if a greater number of control points are used. The DIRECT extremum seeker does guarantee that the global minimum (or vicinity thereof) is found, whereas there are no guarantees that the adjoint method will converge to the global minimum.

Comparing the results of DIRECT with 10, 25 and 50 iterations and DIRECT-L with the respective iterations, it can be seen that DIRECT-L is able to converge to the same solution; in some cases with over 40% reduction in the number of samples and time taken. While DIRECT-L does reduce the number of samples taken, the main drawback is that knowledge of the Lipschitz constant of the cost function is necessary. In the case presented here, the Lipschitz constant was set arbitrarily but it could also be found or estimated, for example, via methods such as response surface modelling [67]. Also, as the number of control points increases, the difference between the computational effort of the adjoint method and DIRECT will be exacerbated as the regions the DIRECT algorithm is required to explore increase exponentially with input dimension. A major limiting factor is the time taken to re-initialise the mesh following the perturbation of the control input.

4.4 Conclusion

The framework for the DIRECT-L optimisation algorithm was introduced and the conditions under which it could be applied to a computational fluid dynamics solver were stated. The algorithm was then employed to optimise the shape of
4.4. CONCLUSION

A two-dimensional aerofoil in order to minimise the aerodynamic drag in steady flow, and the results were contrasted against other global extremum seeking methods and the gradient-based adjoint method. It was found that by utilising DIRECT-L with a suitably chosen Lipschitz constant, that the number of samples and time taken could be reduced by up to 40\% when compared against the DIRECT algorithm, and that the solutions found were still in the vicinity of the true global optimum. The time difference between utilising global optimisation techniques (even with partially converged CFD results and exploiting knowledge of the Lipschitz constant) and local gradient-based approaches remains problematic, at least in the face of high dimensional optimisation requirements. This indicates that for multidisciplinary optimisation of aerodynamic systems that utilise three-dimensional models a local optimisation framework may be preferred.
Chapter 5

The Adjoint Method for Missile Optimisation

5.1 Introduction

It has been shown in Chapter 4 that global optimisation techniques can be made to be computationally competitive with local techniques in low-dimensional CFD aerodynamic shape optimisation. However, when tackling a large number of design variables, as in the case of optimising the performance of a missile undertaking a set of manoeuvres, a local optimisation technique is more effective.

Even when utilising local optimisation techniques, it may be tempting to simultaneously compute both the fluid dynamics and the feedback-controlled rigid-body dynamics. Such a task is, however, still computationally expensive. Moreover, a complication is introduced by the fact that the achieved aircraft trajectory depends upon the controller, which is typically tuned to the missile geometry.

In order to address these issues, this chapter will show that for dynamical systems with certain time-scale properties, simplifications can be made to the aerodynamic forces acting on a missile undergoing a dynamic manoeuvre such that they are adequately described by a pseudo steady-flow aerodynamic model.

The components of this dynamical system model are shown in Figure 5.1. The design parameters are the shape of the missile $\theta_f$ and controller tuning parameters $\theta_s$. Partitioning of the design variables will be made apparent in

---

This Chapter appeared in part in IFAC World Congress (2014) as “Adjoint assisted geometry design of a feedback controlled missile”.

51
Figure 5.1: Dynamic plant model

the following sections. The pseudo steady-flow aerodynamic model consists of a set of steady-flow CFD simulations conducted with the missile at various poses and flow conditions. The aerodynamic properties of the missile, such as lift, pitching moment and drag, are then extracted from the CFD simulations as \( F \), and then utilised in a rigid-body dynamic simulation of the aircraft undertaking some reference manoeuvre \( r(t) \). The rigid body and controller model determines the evolution of the system states \( x_a(t) \) and controller inputs \( u(t) \) over some finite time, which is then used to calculate the cost function, \( J \).

Further to this, the cost function gradients are calculated through a novel combination of an adjoint method for the CFD and a finite-differencing approach for the (computationally cheap) rigid-body dynamics. The combination of a pseudo steady-flow aerodynamic model and a modified adjoint method then allows for the efficient calculation of the gradient of the cost function. A gradient-based local optimiser is used to tune both the controller gains and the missile geometry in order to maximise the missile’s performance.

5.2 Dynamic model

Consider a dynamic system model,

\[
\begin{align*}
\dot{x}_a(t) &= f_a(x_a(t), x_b(t), z(t), u(t), \epsilon) \\
\dot{x}_b(t) &= f_b(x_a(t), x_b(t), z(t), r(t), \theta_s, \epsilon) \\
\epsilon \frac{\partial z(t)}{\partial t} &= f_z(x_a(t), z(t), \frac{\partial z(t)}{\partial x}, \frac{\partial z(t)}{\partial y}, \frac{\partial z(t)}{\partial z}, \frac{\partial^2 z(t)}{\partial x^2}, \frac{\partial^2 z(t)}{\partial x y}, \ldots, \frac{\partial^2 z(t)}{\partial z^2}, \ldots, u(t), x, y, z, \theta_f, \epsilon)
\end{align*}
\]  

(5.1a)  

(5.1b)  

(5.1c)

where \( \epsilon > 0 \) is constant; \( x_a(t) \) and \( z(t) \) respectively represent the plant states subject to controller input \( u(t) \). The states \( z(t) \) are related by a partial differential equation in spatial dimensions \( x \), \( y \) and \( z \). The controller states are represented by \( x_b(t) \) with reference trajectory \( r(t) \).

Property 5.1. The control input \( u(t) := g(x_a(t), x_b(t), z(t), r(t), \theta_s) \) is de-
5.2. DYNAMIC MODEL

Signed such that $(5.1a)-(5.1b)$ is asymptotically stable about a nominal reference trajectory $r(t)$, uniform in $z(t)$.

If the system described by $(5.1c)$ is asymptotically stable and there is time-scale separation between the coupled system $\{x_a(t), x_b(t)\}$ and $z(t)$, then $\epsilon$ is small, and the coupled system is approximated by,

\[
\dot{x}_a(t) = f_a(x_a(t), x_b(t), F(x_a(t), u(t), \theta_f), u(t), 0) \tag{5.2a}
\]

\[
\dot{x}_b(t) = f_b(x_a(t), x_b(t), F(x_a(t), u(t), \theta_f), r(t), \theta_s, 0) \tag{5.2b}
\]

\[
0 = f_z(x_a(t), z(t), \frac{\partial z(t)}{\partial x}, \frac{\partial z(t)}{\partial y}, \frac{\partial z(t)}{\partial z}, \ldots, \frac{\partial^2 z(t)}{\partial x^2}, \ldots, u(t), x, y, z, \theta_f, 0) \tag{5.2c}
\]

Let the solution of (5.2a) and (5.2b) be defined as $\bar{x}_a(t)$ and $\bar{x}_b(t)$ respectively and let $\bar{z}(t) := F(\bar{x}_a(t), u(t), \theta_f)$ be the quasi-steady solution of $z(t)$.

**Property 5.2.** The time scale separation parameter $\epsilon$ in $(5.1c)$ is sufficiently small such that

\[
x_a(t) - \bar{x}_a(t) = O(\epsilon) \tag{5.3}
\]

\[
x_b(t) - \bar{x}_b(t) = O(\epsilon) \tag{5.4}
\]

holds uniformly for all $t$. Furthermore, given an $\epsilon$ there exists a $t_b > 0$ such that

\[
z(t) - \bar{z}(t) = O(\epsilon) \tag{5.5}
\]

holds for $t > t_b$.

Singular perturbation theory has been previously applied to non-linear ordinary differential equation (ODE) systems [42] to separate the time scales of slow ODEs and fast ODEs. It will be shown that in the context of a supersonic missile model, where a coupled rigid body dynamics (ODE system) and a steady CFD model (PDE system) are utilised, that a similar time scale approximation can be made. To this end, the time scale separation parameter $\epsilon$ will be derived for the nominal operating conditions of supersonic flight and this will show that Property 5.2 is indeed a reasonable approximation.

A closed form solution for the map $F(\cdot, \cdot, \cdot)$ is typically not available and so must be approximated by numerical simulations. Surrogate models can be employed and a simple method is to introduce an interpolation scheme to approximate $F(\cdot, \cdot, \cdot)$ over all values of $x_a(t)$ and $u(t)$. 
**Property 5.3.** For any $\varepsilon > 0$, there is an interpolated mapping $\hat{F}(\cdot, \cdot, \cdot)$ where

$$
\left\| \hat{F}(x_a(t), u(t), \theta_f) - F(x_a(t), u(t), \theta_f) \right\| < \varepsilon, \forall x_a(t), u(t).
$$

(5.6)

The variable $\varepsilon$ in Property 5.3 is a measure of the approximation error of the aerodynamic map. Keeping this error small depends on the characteristics of the function as well as the approximation scheme being utilised. If Properties 5.1–5.2 are satisfied, then the model in the optimisation framework can be approximated by (5.2). Note that open-loop trajectories that are designed such that Property 5.1 holds can also be considered. If Property 5.3 is satisfied, for example via sufficient sampling, then the interpolated mapping $\hat{F}(\cdot, \cdot, \cdot)$ can be used in place of $F(\cdot, \cdot, \cdot)$. These three properties are important in enabling accurate modelling of the missile system.

**Application to the Missile**

An example of a dynamic system represented by (5.1a)-(5.1c) is a missile undergoing pitch motion described by (3.5) and a tail-fin actuator model described by (3.7), so that

$$
\dot{x}_a(t) = \begin{bmatrix}
\dot{\alpha} \\
\dot{\theta} \\
\dot{q} \\
\dot{\delta} \\
\dot{\delta}_z \\
\end{bmatrix} = \begin{bmatrix}
\frac{-L(\alpha, \delta) + mg \cos(\theta - \alpha)}{mV} + q \\
q \\
\frac{M(\alpha, \delta)}{I_{yy}} \\
\delta_z \\
-\omega_a^2 \delta - 2\zeta_a \omega_0 \delta_z + \omega_0^2 \delta \\
\end{bmatrix}.
$$

(5.7)

The normal acceleration for the missile is given by (3.9) and this is the system output that is to track $r(t)$. It is required that the commanded input, $u(t) = u$, be subject to a control law such that Property 5.1 is satisfied. To this end, a three-loop autopilot as described by (3.21) is utilised as the controller, although any other stabilising controller such as a tuned proportional integral (PI) controller could be chosen. The dynamic equations for the controller are

$$
\dot{x}_b(t) = -q + K_A(-\eta + K_{DC}\eta_c),
$$

(5.8)

where $r(t) = \eta_c$ is the reference normal acceleration. The output of the controller (and input $u(t)$ into the plant) is the commanded fin deflection

$$
\delta_c = -K_Bq + W_I x_a(t),
$$

(5.9)

Suitable selection of $K_A$, $K_{DC}$, $K_B$ and $W_I$ is required to satisfy Property 5.1.
5.2. DYNAMIC MODEL

as described in Zarchan [107].

Now define

\[ \epsilon := \frac{\delta_{\text{max}} l_r}{V}, \]  

(5.10)

where \( l_r \) represents the characteristic length of the fin and \( \delta_{\text{max}} \) is the maximum fin deflection rate. So that the time-scales of the dynamic system can be evaluated, the previously defined Euler equations (3.15) are rewritten as dimensionless quantities, that is

\[ \frac{\partial \mathbf{z}}{\partial t} = -\hat{\nabla} \cdot \mathbf{f}, \]  

(5.11)

where \( \mathbf{z} = [\hat{\rho} \hat{\rho} \hat{V} \hat{E}]^T \), with \( \hat{\rho} := \rho \rho_r \) representing the dimensionless density of the fluid, \( \mathbf{V} := \frac{1}{V} [v_1 v_2 v_3]^T \) is the dimensionless fluid velocity vector and \( \hat{E} := \frac{E}{\rho V^2} \) is the dimensionless specific total energy. The divergence operator \( \hat{\nabla} := \frac{\delta_{\text{max}} l_r}{V} \nabla \) and the dimensionless convective flux is \( \mathbf{f} := [f_1 f_2 f_3]^T \) with

\[ f_i = \left( \begin{array}{c} \hat{\rho} \hat{v}_i \\ \hat{\rho} \hat{v}_i (\hat{V} + \hat{E}) \end{array} \right), \]  

(5.12)

where \( \hat{v}_i := \frac{v_i}{V} \), \( \delta_i = [\delta_{i1} \delta_{i2} \delta_{i3}]^T \) is a vector of Kronecker delta functions and the dimensionless pressure is given by

\[ \hat{p} = \frac{\rho (\gamma_s - 1)}{\rho_r V^2} \left( E - \frac{1}{2} ||V||^2 \right) \]  

(5.13)

and \( \gamma_s \) is the ratio of specific heats.

For supersonic flight conditions the pitching and actuator states are an order of magnitude slower than those in the flow field. In other co-simulation papers of supersonic systems utilising CFD and rigid body dynamics models [60, 84], time-scale separation is assumed. We show here through the time scale parameter, \( \epsilon \), that this approximation is valid. Assuming a nominal speed of Mach 0.8 for the flow (\( V = 272 \text{ ms}^{-1} \)) and the maximum actuator fin rate is \( \delta_{\text{max}} = 100 \text{ rads}^{-1} \) and \( l_r = 0.33 \text{ m} \), then \( \epsilon = 0.1 \). This indicates that \( O(\epsilon) \) is bounded on the order of \( 10^{-1} \) and therefore the approximation from Property 5.2 is reasonable.

An iterative time-marching CFD solver is used to calculate the aerodynamic maps. In order to find an approximate solution to the partial differential equations, application of spatial discretisation, such as the finite volume method, to (3.15) is necessary. This results in a coupled system of ODEs, where for each discretised cell \( i \)

\[ \frac{d \mathbf{z}_{h,i}}{dt} = -\frac{1}{V_i} \int_{S_i} \mathbf{f} \cdot \mathbf{n}(\mathbf{x}) dS, \]  

(5.14)
where \( z_{h,i}(t) \) is the volume averaged solution of \( z(t) \) for each cell, \( V_i \) is the cell volume, \( n(x) \) is the surface normal vector for each face and \( S_i \) is the total cell surface area. The iterative solver is used to time-march the equations in (5.14) to the steady state solution.

In order to populate the maps of the aerodynamic forces, for each missile shape design candidate, \( \theta_f \), steady state CFD simulations for a series of missile poses are evaluated. That is,

\[
\hat{F}(\alpha, \delta, \theta_f) = [D(\alpha, \delta, \theta_f) \ L(\alpha, \delta, \theta_f) \ M_Y(\alpha, \delta, \theta_f)]^T.
\]

(5.15)

As stated in Property 5.3, the chosen interpolation scheme necessarily affects the approximation error. Missile pitch axis aerodynamic maps are typically quite linear, thus a linear interpolation scheme with relatively few sampling points may be used. For this missile model, it was found that four to five equidistant sampling points for each \( \alpha \) and \( \delta \) was sufficient to keep the approximation error, \( \varepsilon \), small.

5.3 Framework overview

An overview of the optimisation framework is provided in Figure 5.2. The design variables are separated into \( \theta_f \) which are variables associated with fast states and \( \theta_s \) which are variables associated with the slow states.

For each iteration of the optimiser, the following steps are performed:

1. The CFD solver is executed for each pose (i.e. \( \alpha \) and \( \delta \) combination) and this is used to calculate the lift, drag and moment aerodynamic values, \( \hat{F}(\alpha, \delta, \theta_f) \).
2. The maps of the aerodynamic forces are then utilised within a simulation of the feedback-controlled rigid-body dynamics for a series of commanded manoeuvres, and a cost function, \( J \), is evaluated.
3. The derivatives of \( J \) with respect to \( \theta_s \) and \( \hat{F}(\alpha, \delta, \theta_f) \) are calculated by means of finite differences (requiring multiple executions of step 2).
4. An adjoint CFD simulation is performed, using knowledge of both the primal CFD states and the derivatives of \( J \) with respect to \( \hat{F}(\alpha, \delta, \theta_f) \). The result of the adjoint CFD is the derivative of \( J \) with respect to \( \theta_f \).

Remark 5.1. Consider the isolated task of calculating the derivative of \( J \) with respect to \( \theta_s \) and \( \hat{F}(\alpha, \delta, \theta_f) \). In this work, the derivative is approximated using finite-difference, which requires many simulations of the feedback-controlled
5.3. FRAMEWORK OVERVIEW

![Diagram of optimisation framework]

**Figure 5.2:** Optimisation framework for the missile design problem

*rigid-body dynamics per derivative calculation. This computational cost could be reduced by instead using an adjoint based approach (without making any other change to the proposed framework). However, for the case at hand, the computational effort associated with simulations of the rigid-body dynamics is insignificant compared to that of the CFD. Thus the computational benefits of using an adjoint solver for the rigid body dynamics are small when weighed against the effort required to develop such a solver.*

### 5.3.1 Cost function and design variables

The cost function is a scalar quantity used to measure system performance. The design variables are the inputs to the optimiser that affect the value of the cost. A gradient-based optimisation algorithm is able to find a local optimum if the cost and constraint functions are differentiable and if a local optimum exists. The design variables have been partitioned based on whether or not they directly affect the CFD simulations. Clearly, the shape design variables $\theta_f$ directly affect the CFD simulations and the controller parameters $\theta_s$ do not.

**Application to the Missile**

For the tail-fin controlled missile example, one possible cost function is a combination of the normal acceleration tracking error and aerodynamic drag measured over a finite time given previously by (3.11). The test scenario chosen subjects the missile to track step changes in normal acceleration. While it cannot be claimed that this particular cost function and test scenario represents all desired performance characteristics and operating conditions, it does implicitly capture the trade off between the manoeuvrability and range.

This chapter will consider two types of variables. First are shape design variables, represented by deformations away from a base shape. The missile’s
tail fin geometry and associated deformation variables have been previously shown in Figure 3.4 and will be utilised here to show profile optimisation of the fin. The second type of design variables are the control parameters. The three-loop autopilot is chosen as the controller. Let $\theta_f := [X_4 X_5 \cdots X_9]$ be the vectors of shape deformation variables, and let $\theta_s := [\tau_c \zeta_c \omega_c]^T$ be the vector of characteristic parameters for the control tuning method.

Figure 5.3 shows the cost function with respect to one of the geometry design variables. Figure 5.4 shows the cost function with respect to two of the con-

![Continuity of geometry variables](image)

Figure 5.3: Continuity of cost function with respect to the geometry design variable

troller parameters $\tau_c$ and $\zeta_c$ with $\omega_c$ held constant. Figure 5.3 shows that the cost function is continuously differentiable with respect to the geometry design variables. It can also be seen from Figure 5.4 that the cost function is continuously differentiable within the stable region of the controller. It is assumed that the design variables are constrained within the limits of where the cost function is continuous.
5.3. FRAMEWORK OVERVIEW

Cost function $J (\omega_{cr} = 105.0 \, \text{rad/s})$

ζc
τc (s)
20 dB
18 dB
16 dB
14 dB
12 dB
10 dB
8 dB
6 dB
4 dB
3 dB
0.0609 0.1255 0.1901 0.2547 0.3193 0.3839
$\times 10^4$
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
1.0125
1.3625
1.7125
2.0625
2.4125

Figure 5.4: Continuity of cost function with respect to controller parameters, $\tau_c$ and $\zeta_c$

5.3.2 Optimisation

The optimal design variables are given by,

$$\theta^* = \arg \min_{\theta_f, \theta_s} J(\theta_s, \theta_f)$$

subject to $\theta_L \leq \theta \leq \theta_U$

$$g(\theta) \geq 0$$

(5.16)

where $\theta^* = \{\theta^*_f, \theta^*_s\}$ is the set of fast and slow optimal design variables, $J$ is a cost function, $\theta_L$ and $\theta_U$ are, respectively, the upper and lower bound constraints on the design variables and $g(\cdot)$ are inequality constraints. A gradient optimiser is then used to find an optimal design with respect to the design variables within the given design constraints.

Application to the Missile

MATLAB’s sequential quadratic program (SQP) solver is utilised to find the optimal design parameters. Note that with the Zarchan tuning method [107], there are no stability guarantees when optimising the controller parameters, $\theta_s$, and this can lead to critically stable or unstable closed-loop behaviour. The
cost function surface is shown in Figure 5.4 and it can be seen that the lowest cost region is also adjacent to the unstable region (represented as a high cost region) which violates the smoothness assumption required for gradient-based optimisation. A method to overcome this problem is to model the dynamic system as a linear system and to specify a lower bound on the gain margin. Lines of constant gain margin have been overlaid on the cost function surface. The constraints are then,

\begin{equation}
\begin{bmatrix}
-\theta_s + \theta_s,U \\
\theta_s - \theta_s,L \\
G_m - G_m^\ast
\end{bmatrix} \geq 0,
\end{equation}

where \( \theta_s,L, \theta_s,U \) are lower and upper bounds on the controller design variables, \( G_m \) is the gain margin and \( G_m^\ast \) is the gain margin lower bound.

### 5.3.3 Cost and gradient calculation

There are a number of methods to calculate the gradient of a cost function and the computational efficiency will depend on the application as well as the optimiser. For computationally expensive CFD simulations, careful consideration must be given to the chosen gradient method. The method chosen in this chapter is the adjoint method, which is considered to be the state of the art in terms of CFD shape optimisation. Derivation of the adjoint method is shown in Appendix A.1.

In order to utilise the adjoint method, the surface over which the deformation variable is defined must be smooth. Consider the surface of the aerodynamic body shown in Figure 5.5. Let \( S \) represent the baseline surface and \( S' \) be the deformation of the surface at coordinate \( s \). The surface and its deformation is

![Figure 5.5: Surface properties](image)

considered to be smooth if the following property is satisfied.

**Property 5.4.** A normal vector \( n_S(s) \) is well defined at \( s \) such that

\begin{equation}
S' = \{ s + \delta S(s)n_S(s), s \in S \}
\end{equation}

represents the deformed smooth surface.
If Property 5.4 is satisfied, which will be true for all design variables defined over smooth surfaces, then the adjoint method is applicable. Otherwise another gradient method will be required.

The adjoint method is well established for steady-state shape optimisation problems. However the adjoint method must be modified to accommodate the dynamic nature of the optimisation problem.

It can be seen that for \( k \) pseudo-steady poses of the missile, the gradient is,

\[
\frac{dJ}{d\theta_f} = \sum_k \frac{\partial J}{\partial \hat{F}(\alpha, \delta, \theta_f)} \frac{d\hat{F}(\alpha, \delta, \theta_f)}{d\theta_f},
\]

(5.19)

As previously mentioned, the first term of the product in (5.19) is calculated using finite differencing of the rigid body dynamics model. The second term is calculated using the adjoint method with the following modifications.

In aerodynamic studies, the properties of interest are predominantly some function of the pressure over the surface boundary \( S \) of the aircraft. Examples of these functions are given in (3.18). Let,

\[
F = \int_S p(x) \mathbf{n}(x) \cdot d\mathbf{d}S,
\]

(5.20)

where, \( F \in \hat{\mathbf{F}} \) and \( \mathbf{d} \) is a force projection vector. Now suppose that a force projection vector of the form,

\[
\mathbf{d} = \sum_k \frac{\partial J}{\partial \hat{F}(\alpha, \delta, \theta_f)} \mathbf{H},
\]

(5.21)

is used in (5.20). In order to calculate the gradient in (5.19) via the adjoint method, we require each row of \( \mathbf{H} \) in (5.21) to correspond to the forces and moments in the vector \( \hat{\mathbf{F}} \).

This means the derivative of \( J \) with respect to \( \theta_f \) is calculated using just two CFD simulations (one primal and one adjoint) per pose per optimiser iteration.

An alternative and perhaps more intuitive formulation is to calculate the gradients \( \frac{d\hat{F}(\alpha, \delta, \theta_f)}{d\theta_f} \) separately by executing an adjoint CFD simulation for each element in \( \hat{\mathbf{F}} \). However, this particular approach requires \( M+1 \) CFD simulations (one primal and \( M \) adjoint) per pose per optimiser iteration, where \( M \) is the number of elements in \( \hat{\mathbf{F}} \).

**Application to the Missile**

The tail-fins are the control surfaces of this missile. A change in the fins’ position, alters the flow field around the missile, this in turn changes the aerodynamic forces and moments and ultimately this affects the rigid-body dynamical
states. For pitch axis motion, it can be seen that the aerodynamic forces and moments depend on the states $\alpha$ and $\delta$. Rather than compute CFD (and adjoint) simulations over the entire $\alpha$-$\delta$ state space of the missile, DATCOM is used to generate the aerodynamic data for the nose and body of the missile, and only the tail-fin aerodynamic data at different $\delta$-deflections (poses) are generated using steady flow CFD (and adjoint) simulations.

With $\hat{F}(\alpha, \delta, \theta_f)$ defined as (5.15), this leads to a matrix,

$$
H = \frac{1}{C_\infty} \begin{bmatrix}
\cos \delta & \sin \delta & 0 \\
-\sin \delta & \cos \delta & 0 \\
0 & 0 & \frac{(x - x_0)}{L_{\text{ref}}}
\end{bmatrix},
$$

(5.22)

which is used in the adjoint CFD simulation, where, $C_\infty = \frac{1}{2} V^2 \rho_\infty^2 A_z$. $\rho_\infty$ is freestream density, $L_{\text{ref}}$ is the reference length and $A_z$ is the reference area. Also note that the geometric design variables, $\theta_f$, have all been defined over smooth surfaces, which means that Property 5.4 is satisfied.

### 5.3.4 Convergence

Properties 5.1–5.3 relate to the plant model and controller, and satisfaction thereof implies that the approximation as slow and fast dynamics sufficiently resemble the real dynamical system. Property 5.4 relates to the definition of the geometric design variables over smooth surfaces and is a necessary condition of the adjoint method. It has also been shown that the cost function is differentiable which is a necessary condition for the deployment of gradient based optimisers. If all above hold, then the optimisation framework will converge to the vicinity of a minimum of the cost function.

### 5.4 Results

The missile model is subjected to both a positive and negative step change in command acceleration of 5 ms$^{-2}$ over a 3 second period at a freestream speed of Mach 0.8. An example of the positive step command and response of the missile is shown in Figure 5.6. The fin deflection rate was recorded to be no more than 100 rad $\cdot$ s$^{-1}$ (equivalent to a fin tip rotational speed of 16.67 ms$^{-1}$). Comparing this value with the freestream Mach speed, it is reasonable to assume that there exists time scale separation between the actuator’s speed and the stabilisation of the transonic flow field and therefore justifies the use of steady flow CFD data.

The cost and gradients calculated using the adjoint method are used with an interior point optimisation algorithm [104] to determine the optimum tail-fin
design. Three optimisation test cases are conducted. In Test Case A only the geometry is optimised. In Test Case B the three control design variables $\tau$, $\zeta$ and $\omega_{cr}$, are optimised. In Test Case C both geometry and control parameters are optimised. A cost weighting $w = 3.0 \times 10^{-6}$ is used. The convergence histories for the test cases are shown in Figure 5.7.

The result shows that the optimiser was able to converge quite rapidly to the local minimum, requiring only a modest number of function evaluations. In Test Case A the cost was reduced by 1.4%, in Test Case B it is reduced by 3.8%, while in Test Case C the cost was reduced by 5.2%. It is noted that the drag was reduced in both cases A and C by about 2%. While this does not seem to be a significant reduction in absolute terms, it would still be a significant fuel saving. In Test Case A it can be seen that the geometry variables have only a small impact on the tracking error component, and vice versa for Test Case B, the impact of the control parameters on the drag component is small (in fact the drag force is slightly increased). This result indicates that the tracking error depends on the control parameters, and similarly, that the drag force depends on the geometric parameters.

Figure 5.8 show cut-through sections of the tail-fin at different stations along the span for the initial tail-fin and the two optimised tail-fins. It can be seen that both optimised tail-fins have very similar profiles. The overall height of the tail-fin has been reduced and the fin tapers in toward its skinniest point at the mid-span. It is interesting to note that the profile of the tip of the fin bulges out again, which emulates the aerodynamic benefits of some traditional wing-tip designs.
5.5 Conclusion

In this chapter, an optimisation framework integrating an adjoint-based cost sensitivity calculation into an environment consisting of CFD, rigid body dy-
dynamics and controller simulation has been developed. This framework illustrates that the adjoint method, which is typically used for static aerodynamic optimisation, can also be extended to dynamic optimisation problems. The system’s time-scale properties are exploited in order to perform efficient dynamic MDO. The time-scales of the system are identified and this then allows the use of pseudo-static aerodynamic maps. This greatly improves on the computation time of CFD simulations.

The framework is demonstrated through the design of a missile’s tail-fin, and as the main result is based on the adjoint method, it shows that a locally optimal design can be achieved with only a twofold increase in computational resources and is independent of the number of geometry design variables. As such, complex geometries with a large number of variables can be incorporated into the framework with minimal computational cost.
Chapter 6
Implicit Filtering for Missile Optimisation

6.1 Introduction

In Chapter 5, it was shown that the gradient of the cost with respect to the geometric design variables could be efficiently calculated using a modified continuous adjoint method. A limitation of the continuous adjoint method employed previously is that it cannot handle design changes over sharp-edged geometries [54]. Consider a simple example of the sensitivity of the lift of a supersonic fin. Figure 6.1 shows the gradient surface of the cost function with respect to deformation of the wing span calculated using finite differences and the continuous adjoint method. Comparing the two methods, it can be seen that the gradients not only have different magnitudes, but also have opposing sign.

The discrete adjoint is not limited to only smooth geometry. However, it is rare for multidisciplinary optimisation problems to be programmed within the same simulator or in the same programming language. In the usual case, specialised simulators are coupled together in a co-design environment. This limits the applicability of the discrete adjoint method for multidisciplinary optimisation problems and it is the reason it is not pursued here.

This chapter aims to extend the framework introduced in Chapter 5 for optimising missiles that provides guarantees that the solution is optimal in some sense and allows arbitrary shapes to be incorporated into the design, while at
Figure 6.1: Comparison of the gradient of the lift coefficient with respect to wing span calculated via the adjoint method and finite differences.

the same time reducing the computational burden associated with high-fidelity model optimisation.

6.2 Framework overview

The optimisation framework is shown in Figure 6.2. The framework consists of an inner and outer optimisation loop. This two-level optimisation strategy
is used to separate the design variables that rely on the computationally intensive CFD simulations from those that do not; with the aim of minimising the overall optimisation time. Partitioning of the design variables in the optimisation framework should result in a reduction in the overall simulation time by avoiding extra CFD evaluations (e.g. during a line search).

The inner optimisation problem considers only the design parameters associated with the slow states $x_a(t)$ and $x_b(t)$, where the design variables of the fast states $z(t)$ are held constant. The optimal design variables in the inner loop are given by

$$
\theta_s^* = \arg \min_{\theta_s} J(\theta_s, \theta_f) \\
\text{subject to } g_{in}(\theta_s) \geq 0 \text{ and } h_{in}(\theta_s) = 0
$$

where $J$ is a cost function, $g_{in}(\cdot)$ and $h_{in}(\cdot)$ are, respectively, inequality and equality constraints. The constraints $g_{in}(\cdot)$ includes bound constraints on the design variables and $h_{in}(\cdot)$ includes the slow system dynamics (5.2a) and (5.2b).

The outer optimisation problem considers the design parameters associated with the fast states $z(t)$ and with the restriction $\theta_s = \theta_s^*(\theta_f)$. The optimal design variables of the outer loop are given by,

$$
\theta_f^* = \arg \min_{\theta_f} J(\theta_s^*(\theta_f), \theta_f) \\
\text{subject to } \theta_{f,L} \geq \theta_f \geq \theta_{f,U} \\
h_{out}(\theta_f) = 0
$$

where $J$ is a cost function, $\theta_{f,L}$ and $\theta_{f,U}$ are, respectively, the upper and lower bound constraints on the design variables and $h_{out}(\cdot)$ are equality constraints that include the fast system dynamics (5.2c). Note that the framework can only handle inequality bound constraints in the outer loop.

Let the shape variables $\theta_f = \{\theta_{ls}, \theta_{ad}\}$, where $\theta_{ls}$ represents the variables whose gradients are calculated via least squares finite differencing and $\theta_{ad}$ represents the variables whose gradients are calculated via the adjoint method. Details of each component of the framework are described in the following subsections.

### 6.2.1 Cost function and design variables

The cost function (3.11) is the same as in Chapter 5. In this chapter, more geometry design variables (including those defined across sharp edges) are included.
Application to a Supersonic Missile

The cost function measures the range and manoeuvrability and is given by (3.11).

In Chapter 5, only the tail fin shape was used in the design optimisation. In this chapter, both the tail fin (shown in Figure 3.4) and nose (shown in Figure 3.3) will be used. The geometry design variables are defined over sharp edges on the tail fin, and this will make it suitable to trial the efficacy of the proposed framework.

The control parameter design remains unchanged and the three-loop autopilot will be utilised again.

Let \( \theta_{ls} := [X_1 \ X_2 \ X_3 \ X_{10} \ X_{11}]^T \) and \( \theta_{ad} := [X_4 \ X_5 \ \cdots \ X_9] \) be the vectors of shape deformation variables, and let \( \theta_s := [\tau_c \ \zeta_c \ \omega_{cr}]^T \) be the vector of characteristic parameters for the control tuning method.

6.2.2 Inner optimisation

The rigid body dynamics model and controller model, respectively (5.2a) and (5.2b), the design variables \( \theta_s \) and cost function \( J \) can be cast as a constrained optimisation problem of the form (6.1). An optimiser can then be used to find the optimum point with respect to the design variables related to the slow states. While the problem here has been set up to optimise only the controller design variables, that is, parameters in the controller (5.2b), the inner loop is also able to optimise other plant parameters from the slow dynamic model (5.2a).
6.2. FRAMEWORK OVERVIEW

Application to a Supersonic Missile

The inner loop takes a given missile geometry and optimises the controller characteristic parameters. If the plant constraints affected by the outer loop optimisation are to be dealt with explicitly by the controller, for example if model predictive control is utilised, then this may require additional information to be passed between the inner and outer loops.

In the example presented here, MATLAB’s sequential quadratic program (SQP) solver is utilised to find the optimal tuning parameters. As the computational requirement for the inner loop does not require CFD simulations, the gradient is simply calculated using finite differences. The same gain margin stability constraint (5.17) is applied so that the optimiser remains within the stable region.

6.2.3 Outer optimisation

The system model (5.2), the design variables $\theta_f$ and cost function $J$ can be cast as a constrained optimisation problem of the form (6.2). An optimiser can then be used to find the optimal point with respect to the design variables related to the fast states.

The calculation of the cost function and its gradients can be computationally costly. One method to mitigate computational load is to consider the use of noisy simulation models. Note that simulations do not have noise in the physical sense; however errors arising from the numerical approximation of solutions can be treated as noise. The simulation model is then coupled with a optimiser that can tolerate noise while still providing certain guarantees with regards to the optimal solution.

Let the error of the computed gradient of the cost function be modelled as additive noise, that is,

$$\nabla \hat{J}(\theta, \gamma) = \nabla J(\theta) + \Phi(\gamma) \quad (6.3)$$

where $\nabla \hat{J}(\theta, \gamma)$ is the computed (approximate) gradient, $\nabla J(\theta)$ is the true (analytical) gradient, and $\Phi(\gamma)$ is the additive error.

Property 6.1. There exists an error parameter $\gamma > 0$, such that $\Phi(\gamma) \to 0$ as $\gamma \to \infty$.

With iterative solvers, the computational time typically increases with $\gamma$. Thus, the parameter $\gamma$ can be used to control the fidelity of the computed gradient. Let successive design iterations be found by an update law of the
$\theta_{k+1} = f(\nabla \hat{j}(\theta_k, \gamma_k)),$ \hspace{1cm} (6.4)

where $\theta_k$ is the design variables and $\gamma_k$ is the error parameter at iteration $k$ of the optimiser.

**Property 6.2.** $\Phi(\gamma_k) \to 0$ as $k \to \infty$, so that the iterates of the design variables, $\theta_k \to \theta^*$, where $\theta^*$ is an optimal solution.

If the gradient of the cost function satisfies Property 6.1 and the optimiser is set up with an error control mechanism that satisfies Property 6.2, then the noise of the gradient will approach zero with increasing iterations of the optimiser and the optimiser will converge to the vicinity of the optimal solution.

**Application to a Supersonic Missile**

The missile example utilises an iterative CFD solver to calculate the map of aerodynamic forces. Within the solver, the residual error is used as a metric of the closeness of the approximate solution to the true solution and terminates the solver when the error drops below a certain threshold. Figure 6.3 shows the residual error against the CFD solver iterations. It can be seen that after 10 solver iterations, the residual error decreases. One of the inputs for calculating the gradient of the cost function is the map of aerodynamic forces, and so the additive error $\Phi(\gamma)$ in Property 6.1 is modelled by the residual error of the CFD solver. The number of CFD iterations then corresponds to the parameter...
6.2. FRAMEWORK OVERVIEW

In the context of the optimiser, as the optimisation progresses the CFD iteration requires to be run to \( \gamma \) CFD iterations. Note that there still exists an error between the PDE solution and the spatially discretised ODE solution and that this error tends to zero as the discretisation approaches the limit. This is known as the *consistency* property of the solver and with Property 6.1 only the closeness of the partially converged numerical solution to the ODE solution (and not the PDE solution) can be guaranteed.

A candidate optimiser that satisfies Property 6.2 is the implicit filtering algorithm [40]. Implicit filtering is a variation of the bound-constrained coordinate search (or direct search) optimisation method. Like coordinate search methods, implicit filtering samples points on a predefined stencil, however the algorithm then uses the sampled information to build a local model of the cost function. The local model provides gradient information for a line search algorithm. In this chapter, the implicit filtering algorithm was modified to neglect sampling in \( \theta_s \). Gradient information is found using the adjoint method instead. The modified algorithm is given in the Appendix.

6.2.4 Gradient calculation

There are a number of methods to calculate the gradient of a cost function and the computational efficiency will depend on the application as well as the optimiser. For computationally expensive CFD simulations, careful consideration must be given to the chosen gradient method.

Ideally, the gradient for all design variables should be found via the adjoint method. However, this is not always possible, as shown in Figure 6.1 for sharp edged geometry variables. In this chapter, two candidate gradient methods are utilised, namely the continuous adjoint method and least squares finite difference (LSFD). Further information regarding their derivation can be found in Appendix A.

If Property 5.4 is satisfied, which will be true for all design variables defined over smooth surfaces, then the adjoint method is applicable. Otherwise LSFD is used instead.

**Application to a Supersonic Missile**

Table 6.1 shows the gradient of the lift coefficient cost function with respect to a selection of tail fin design variables calculated using LSFD, the continuous adjoint method and with the central finite difference method as a benchmark for comparison. The total computational time and number of CFD simulations for each method is also shown. Note that 12 design variables are evaluated which results in 24 simulations for both the LSFD and central finite difference method.
It can be seen that the continuous adjoint method is the fastest method requiring only two CFD simulations, but is not applicable for design variables over sharp geometries. LSFD is applicable for all design variables and is also faster than the central finite difference method. Note that the speed up for the LSFD method is due to error control of the CFD iterations (i.e. the parameter $\gamma$), while the speed up for the adjoint method is due to both the nature of the method being independent of the number of design variables as well as the use of error control. The adjoint method is utilised for design variables $\theta_{ad}$ defined over smooth surfaces (i.e. tail fin profile), and LSFD is utilised for design variables $\theta_{ls}$ defined over sharp edges (i.e. nose tip and tail fin planform).

6.2.5 Convergence

Properties 5.1–5.3 relate to the plant model and controller, and satisfaction thereof implies that the approximation as slow and fast dynamics sufficiently resemble the real dynamical system. It has been shown that the design problem can be stated as a nested co-design optimisation problem. It has also been shown that the cost function is differentiable which is a necessary condition for the deployment of gradient based optimisers. Properties 6.1 and 6.2 relate to the tolerance of the optimiser to inaccurate gradients. Lastly, Property 5.4 dictates if the adjoint method is suitable for the gradient calculation. If all above hold, then the optimisation framework will converge to the vicinity of a minimum of the cost function.

The rate of convergence and accuracy of the solution will depend of the noise characteristics of $\Phi(\gamma)$, on the update law $f(\cdot)$ of the optimiser and the relationship between the iterate $k$ and parameter $\gamma$. If $\Phi(\gamma)$ is small or if it decays quickly with increasing $\gamma$, then the computed gradient is more accurate. Similarly, if the update law $f(\cdot)$ has a higher tolerance for inaccuracy in the gradient, then the optimiser will approach the minimum more quickly. In par-
ticular, the optimiser may converge to the solution without necessarily requiring \( \gamma \) to be large, resulting in faster computation of the gradient and also the overall optimisation process. However, no claim is made that this proposed framework is always faster than other gradient-based approaches, only that in practice it has been observed to be faster for the given missile example.

6.3 Results

Two examples will be presented in this section. The first is a tail fin planform optimisation with the focus on the computational cost of the proposed framework. The relationship between controlling the error in computing the gradient and the convergence rate will be discussed. The second example optimises the missile tail fin’s planform and profile and also the nose geometry. The adjoint method is utilised for the gradient calculation for a subset of design variables. The computational advantage of utilising gradients derived from the adjoint method in this framework is also highlighted. The missile model is initialised with zero normal acceleration with zero angle of attack at a freestream speed of Mach 2. A reference test case of a 10g positive and negative step change in commanded acceleration over a 3 second period is used, and fin deflection is saturated at 10 degrees. The cost function is taken from (3.11) and measures the integrated error between the achieved and commanded normal acceleration along with the integrated aerodynamic drag. The weighting parameter of the cost function is set to \( w = 5 \times 10^{-6} \). The cost function is normalised according to

\[
\bar{J} = \frac{J}{J_0} \times 100(\%)
\]

where \( J_0 \) is the cost of the initial design.

6.3.1 Tail fin planform optimisation

In this example, the design variables are the tail-fin leading edge and wing span (\( X_{10} \) and \( X_{11} \) in Figure 3.4). The aerodynamic data for the missile body are taken directly from DATCOM since there is no deformation over the body. CFD simulations of the tail fin at \( \delta = 0^\circ, 2.5^\circ, 5^\circ, 7.5^\circ, 10^\circ \) are conducted. The control parameters are also optimised by the inner loop. Three optimisation runs are conducted. In the first run, the residual error is controlled by \( \gamma = k \) and increases the CFD iterations over successive implicit filtering optimiser iterations (i.e. run with error control). In the second case, the CFD iterations are fixed so that the CFD simulation is considered fully converged and the residual error is small (i.e. run without error control). In the third case, MATLAB’s interior point
IMPLICIT FILTERING FOR MISSILE OPTIMISATION

solver (which utilises finite differencing) has been included for comparison. In the example, all three runs converged to the same optimal design.

Table 6.2 shows the optimised control parameters for the initial and final shape, that is, the control parameters after the inner-loop optimisation. There is very little change to the optimal controller parameters, which suggests that the control parameters do not influence the cost function in this example.

<table>
<thead>
<tr>
<th>Shape</th>
<th>τc</th>
<th>ζc</th>
<th>ωcr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>0.1165</td>
<td>1.7664</td>
<td>105</td>
</tr>
<tr>
<td>Optimised</td>
<td>0.1158</td>
<td>1.7626</td>
<td>105</td>
</tr>
</tbody>
</table>

Table 6.2: Optimal control parameters for original and optimised tail fin planform shape

The original and optimised tail fin shape are shown in Figure 6.4. It can be seen that the optimal planform shape favours a shorter wingspan and also a smaller wing sweep, which results in a low aspect ratio fin. A low aspect ratio result with control parameters that did not vary by much suggests that the optimisation was only able to minimise the drag component of the cost function.

![Figure 6.4: Original and optimised shape of tail fin](image)

Figure 6.4: Original and optimised shape of tail fin

Figure 6.5 shows the reduction in the normalised cost function against the computational time. It is noted that the reduction in the cost function is less than 1% and this suggests that the tail fin planform design variables are only very loosely related to the cost function for this example. The figure shows that implicit filtering with error control approaches the minimum in about 60% of the total time when compared to the interior point algorithm, and about 40% of
6.3. RESULTS

The total time when compared to implicit filtering without error control. This result shows that if the properties of model and framework hold, then there is a computational advantage for using error control for optimisation.

6.3.2 Nose, tail-fin planform and profile optimisation

In this example, the nose and tail fin geometries are optimised. (\(X_1\) to \(X_3\) in Figure 3.3 and \(X_4\) to \(X_{11}\) in Figure 3.4). CFD simulations are required for both the missile body and tail fin. For the nose, CFD simulations are conducted at \(\alpha = 0^\circ, 4^\circ, 8^\circ, 12^\circ, 16^\circ\), and for the tail fin, they are conducted at \(\delta = 0^\circ, 4^\circ, 8^\circ, 12^\circ\). The gradients for the nose and tail fin planform design variables are calculated using least squares finite differencing, while the gradients for the tail fin profile design variables are calculated using the adjoint method.

Two optimisation cases are presented. Firstly, only the nose and the tail fin planform are optimised, that is, the design variables utilising the adjoint method are omitted. Secondly, the nose, tail fin planform and profile are optimised. The result of the optimised nose and planform shape are the same for both cases and is shown in Figure 6.6. Note that the body of the missile, i.e. between 1.4\(m\) and 3.6\(m\), has been omitted in the diagram so that differences in the nose and tail fin shape is more easily observable. It can be seen that the shape of the optimised nose is sharper and that the shape of the optimised tail fin is the same as the optimised result in the previous example. In order to compare the profile shape, cross-sections through the tail fin are shown in Figures 6.7 and 6.8. The cross-sections are uniformly spaced and are labelled with increasing
IMPLICIT FILTERING FOR MISSILE OPTIMISATION

Nose and tail fin optimisation

height (m)

length (m)

0 0 .2 0.4 0.6 0.8 1 1.2 3.6 3.8 4

−0.8

−0.6

−0.4

−0.2

0

0.2

0.4

0.6

0.8

Figure 6.6: Shape of original and optimised missile (body of missile omitted)

numbers to indicate the distance away from the missile body. Figure 6.7 shows the profile shape where only the planform has been optimised. Figure 6.8 shows the tail-fin where the planform and profile have been optimised. It can be seen that the height of the profile has been optimised such that it is almost uniform, but with a slight bulge in the middle of the fin. Note that the height of the tail fin profile is not at the lower limits set in the optimiser, and this suggests that there is a trade-off between manoeuvrability and aerodynamic drag for the profile design variables.

Table 6.3 shows that the controller parameters have changed from the original, and unlike in the previous example, this means that the geometry variables and controller parameters are coupled.

<table>
<thead>
<tr>
<th>Shape</th>
<th>( \tau_c )</th>
<th>( \zeta_c )</th>
<th>( \omega_{cr} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>0.0804</td>
<td>1.4759</td>
<td>105</td>
</tr>
<tr>
<td>Nose &amp; planform</td>
<td>0.0854</td>
<td>1.5176</td>
<td>105</td>
</tr>
<tr>
<td>Nose, planform &amp; profile</td>
<td>0.0850</td>
<td>1.5136</td>
<td>105</td>
</tr>
</tbody>
</table>

Table 6.3: Optimal control parameters for original and optimised nose & fin shapes

The normalised cost function versus the total optimisation time are shown in Figure 6.9. By optimising the nose and tail-fin planform, there is an 11% reduction in the cost function relative to the original design. Including the tail fin profile yields an additional 4% improvement. This is an interesting result, as there is a marked improvement to the cost, yet the changes to the tail
6.3. RESULTS

Figure 6.7: Cross sections of planform optimised tail fin shape (i.e. the chord shown here has not changed). The cross sections are labelled 1-5 as they move uniformly away from the missile body.

Figure 6.8: Cross sections of planform and profile optimised tail fin shape. The cross sections are labelled 1-5 as they move uniformly away from the missile body.

fin profile are small relative to the entire missile (the deformation of the tail fin is of two orders of magnitude smaller compared to the total length of the missile). On the computational aspect, the number of design variables utilising the adjoint method is more than twice that of the other geometry variables, yet the optimisation time is not significantly increased. It can be seen that it is
Figure 6.9: Total optimisation time of nose and fin

better to utilise the adjoint method where possible, that is, for design variables where Property 5.4 is satisfied. Indeed, one of the benefits of the optimisation framework is that the designer need not be too worried about the computational effort when adding geometry design variables. Together with the flexibility of choosing the gradient method and the tolerance to noise in the optimiser means that this framework could lead to unexpected improvements in the optimised design.

6.4 Conclusion

In this chapter, a framework for the optimisation of a controlled aerodynamic system has been developed and it has been shown to successfully optimise a missile’s shape and control design parameters. In particular, this framework extends the work in Chapter 5 to geometry design variables over sharp edges.

Two of the system’s properties are exploited in this framework. Firstly, the identified time-scales of the system are arranged in a nested co-design optimisation architecture. Secondly, surrogate models are utilised in estimating the aerodynamic maps as well as the cost function to reduce the computational burden of CFD simulations.

This framework has provided flexibility in the setup of the optimisation problem, where various methods for the gradient calculation can be incorporated. It is also less restrictive than the discrete adjoint method, as it allows coupling of various specialised simulators (including closed source simulators). It has been shown that controlling the error of the gradient computed by the optimiser leads
to a speed up in the optimisation time. This is particularly useful when util-
isising computationally expensive models such as CFD simulations. Examples
provided in this chapter show that the framework is particularly useful in the
optimisation of supersonic vehicle shapes and is capable of dealing with both
sharp and smooth geometry design parameters.

To improve the fidelity of the model, more representative fluid flow equa-
tions can be implemented in the CFD solver, such as the Reynolds-averaged
Navier-Stokes (RANS) equations with an associated turbulence model. Con-
vestity is a property of the optimisation metric and if the optimisation surface is
non-convex, a multi-start approach can also be used to increase the likelihood of
finding the global optimum. Other candidate systems that satisfy the necessary
system properties can be applied to the framework. For example, an electro-
magnetic system where the shapes of the rotors and stators are optimised has a
similar time-scale separation between the electrical and mechanical states. They
also rely on electromagnetic field simulations that are computationally intensive.
Such systems are natural candidates for this optimisation framework.
IMPLICIT FILTERING FOR MISSILE OPTIMISATION
Chapter 7

Mesh Adaptation of Nonlinear Model Predictive Control

7.1 Introduction

In this chapter, mesh adaptation methods previously developed for CFD problems are utilised in the context of NMPC problems in order to speed up the online computational time. This makes NMPC more competitive when compared to other controller algorithms. Two examples will be presented in Section 7.5. The first example is a two-state nonlinear system and the second example is the longitudinal pitching motion of a missile. Linear and nonlinear model predictive controllers are rarely employed for missile autopilots. As discussed in Chapter 2.3 and shown in Table 2.3, other methods such as gain-scheduled control, the state dependent Ricatti equation (SDRE) approach, sliding mode control and feedback linearisation have been previously reported. When NMPC is utilised in an autopilot context, closed form suboptimal approximations (e.g. Taylor series) are employed in order to avoid the high computational requirements of solving an online optimisation problem at each time-step but are unable to handle constraints explicitly. The missile result in this chapter shows that mesh adaptation methods applied to direct collocated NMPC is a viable option for missile autopilots. The NMPC missile autopilot is also compared with PI

This Chapter appeared in part in *International Journal of Robust and Nonlinear Control (Accepted)* as “Mesh adaptation in direct collocated nonlinear model predictive control”.

83
and SDRE autopilots.

In direct collocation, first introduced in Chapter 3.4.2, both the control input and the state are discretised and treated as decision variables. This leads to a sparse finite dimensional optimisation problem where the constraints of the NLP consist of the constraints of the control problem along with the discretised system dynamics in the form of equality constraints.

Direct methods involve approximation of the control problem through discretisation of the control and/or state variables over the time horizon. The choice of discretisation is driven by keeping the error uniformly small and the computational and memory constraints of the controller hardware. A method that addresses both these issues and has previously been researched is adaptive refinement. Adaptive refinement seeks to refine the discretisation mesh in regions where the accuracy of the solution is demanded and to coarsen the mesh where it is not.

In consideration of NMPC as opposed to optimal control problems, the aforementioned adaptive refinement methods face an additional constraint. That is, that NMPC is a discrete control method and requires a fundamental time step on which control actions and sensor measurements are taken. The selection of the fundamental time step depends on a number of factors including the plant, sensor and actuator dynamics and the available computational resources of the controller hardware. This means that arbitrary refinement of the control input is not always possible.

The refinement method is based on goal-oriented mesh adaptation [101, 7] which has been extensively used in mesh refinement of partial differential equations particularly in the computational fluid dynamics (CFD) community. A unique aspect of this method compared to previous work is that the mesh is adapted with respect to the cost function and not the state or input dynamics. This leads to refinement dictated by the cost function in the control problem rather than being driven by the system dynamics.

7.2 Direct collocation

The direct collocation method for NMPC has been previously described in Section 3.4.2. In order to support an adaptive mesh scheme, the standard formulation of the NLP needs to be modified and is derived as follows.

Assumption 7.1. The fundamental time steps of the NMPC are invariant. That is to say, the discretised time intervals over which the control inputs remain constant. It follows that the refinement method is only applied within each fundamental time step and it is only useful to refine the state variables.
So that Assumption 7.1 holds, let the prediction horizon \( T_f := t_f - t_0 \) be discretised using two indices \( j \in \{0, 1, 2, \ldots, M\} \) and \( k \in \{0, 1, 2, \ldots, N\} \), where \( j \) is the control index and \( k \) is the state index. Time along the prediction horizon is then given by \( t_j = [t_0, t_1, \ldots, t_M] \), where, \( t_0 < t_1 < \ldots < t_M \) and \( t_k = [t_0, t_1, \ldots, t_N] \), where, \( t_0 < t_1 < \ldots < t_N \). Also \( t_f = t_M = t_N \). Let \( \tau \in \mathbb{Z}_{>0}^N \) be a vector where each element represents the current state discretisation level for the time interval \( t_j - t_{j-1} \). The relationship between the two indices are such that for \( t_{j-1} < t_k < t_j \),

\[
k = \begin{cases} 
0, & \text{if } j = 0 \\
m + \sum_{l=1}^{j-1} \tau_l, & \text{otherwise}
\end{cases} \tag{7.1}
\]

where \( m \in \{0, 1, \ldots, \tau_j\} \). An example of \( \tau \), \( x_k \) and \( u_j \) for a dynamic system is illustrated in Figure 7.1. As can be seen in the figure, the control index \( j \) is divided evenly over the prediction horizon and the state index \( k \) is determined by the value of \( \tau \) as well as the control index.

![Figure 7.1: Refinement relationship between \( \tau \) and the states and inputs](image)

The collocation approximations within each time interval in the NLP are
then,

\[
R(x, u) = \begin{cases} 
  x_0 - x_{\text{initial}} \\
  x_1 - p(t_1) \\
  \vdots \\
  x_k - p(t_k) \\
  \vdots \\
  x_N - p(t_N) 
\end{cases} = 0 \quad (7.2)
\]

Note that the state discretisation method chosen above follows the \( h \)-refinement methods for mesh adaptation. The alternative, and equally valid, \( hp \)-refinement methods could also be employed by altering the degree of the collocation polynomial \( p(\cdot) \).

The state constraints are discretised with the new index \( k \) as follows,

\[
x_k \in \mathcal{X} \quad (7.3)
\]

The cost function is also similarly discretised,

\[
J = \phi(x_N, t_N) + \sum_{k=1}^{N} L(x_k, u_j, t_k) \Delta t. \quad (7.4)
\]

**Remark 7.1.** Discretisation of the state constraints in this way means that there are only guarantees for the constraints at the sampling time and not at the intersampling times. For practical reasons, the possible (albeit mild) violations at intersampling times are ignored.

### 7.3 Goal oriented error estimation

A necessary step in adaptive mesh refinement is to establish a method of estimating the error in the approximation of the dynamical system (3.22). Specifically, an estimate of the error is required for each discretisation time interval in the prediction horizon so that the refinement algorithm can selectively refine (coarsen) only in the time intervals where the error is deemed to be high (low).

As discussed in Section 2.3, there are numerous error indicators that can be used to determine the error of the computed solution. For goal oriented error estimation, the error indicator relates the approximation error directly to the cost function. This is particularly useful in the context of NMPC as the error indicator can be used to target the refinement of the approximate solution only where it has an appreciable impact on the cost function.

There are two main methods in goal oriented error estimation. In the continuous method [7], the error estimate is derived from the continuous states of
7.3. GOAL ORIENTED ERROR ESTIMATION

the differential equations. In the discrete method [31], the states are first discretised and then the error estimation derived. The proposed approach in this paper follows the discrete method.

Firstly, let $C$ and $F$ denote some arbitrary coarse and fine discretisation respectively. Suppose the optimal state trajectory $x$ and input $u$ are generated by the NLP solver over the coarse and fine discretisations and are given by \{$x_C, u_C$\} and \{$x_F, u_F$\} respectively.

**Assumption 7.2.** $||x_C - x_F||_2 < \epsilon_x$ and $||u_C - u_F||_2 < \epsilon_u$.

**Remark 7.2.** The aim is to avoid utilising the (computationally expensive) NLP solver over the fine discretisation. Assumption 7.2 means that the solution to the NLP utilising the coarse discretisation cannot be too different from the fine discretisation, so that:

1. The system states calculated using the coarse discretisation can be interpolated onto the fine discretisation.

2. The optimal input trajectory calculated using the coarse discretisation can be used directly on the fine discretisation.

Note that Assumption 7.2 is similarly made in [9].

Denote the cost function (7.4) for the fine discretisation as $J_F(x_F, u)$ and the residual (7.2) as $R_F(x_F, u)$.

Now define $x_C^F := I^C_F x_C$ as the solution of the coarse discretisation that has been interpolated onto the fine discretisation.

Consider Taylor series expansion of the cost function for the fine discretisation,

$$J_F(x_F, u) = J_F(x_C^F, u) + \frac{\partial J_F}{\partial x_F} \bigg|_{x_C^F} (x_F - x_C^F) + \mathcal{O}((x_F - x_C^F)^2) \quad (7.5)$$

Similarly, the Taylor series expansion of the residual is,

$$R_F(x_F, u) = R_F(x_C^F, u) + \frac{\partial R_F}{\partial x_F} \bigg|_{x_C^F} (x_F - x_C^F) + \mathcal{O}((x_F - x_C^F)^2) = 0 \quad (7.6)$$

**Assumption 7.3.** There is a feasible solution to the NMPC and this implies that an inverse of $\frac{\partial R_F}{\partial x_F} \bigg|_{x_C^F}$ exists.

If Assumption 7.3 holds, define

$$\psi_F := \frac{\partial J_F}{\partial x_F} \bigg|_{x_C^F} \left[ \frac{\partial R_F}{\partial x_F} \bigg|_{x_C^F} \right]^{-1}. \quad (7.7)$$
Ignoring higher order terms as \((x_F - x_F^C)\) is small and combining (7.5), (7.6) and (7.7),
\[
J_F(x_F, u) = J_F(x_F^C, u) - \psi_F R_F(x_F^C, u)
\]
Solving (7.7) requires evaluation at the fine discretisation level, but let
\[
\psi_F = L_F^C \psi_C - \delta \psi_F,
\]
where, \(L_F^C \psi_C\) is an interpolated estimate of \(\psi_F\) and \(\delta \psi_F\) is the error of \(\psi\) between the fine and coarse estimate. Rewriting (7.8) and combining with (7.9),
\[
\Delta J_F := J_F(x_F, u) - J_F(x_F^C, u) = L_F^C \psi_C R_F(x_F^C, u) - \delta \psi_F R_F(x_F^C, u)
\]

**Remark 7.3.** Equation (7.10) is the error estimate of the cost function between the fine discretisation and the (coarse) interpolated discretisation. The implication of \(\Delta J_F\) is that the error between the coarse and fine discretisation can be made without having to calculate the cost function or solving the NLP at the level of the fine discretisation.

The first term of the right hand side of (7.10) is called the *computable correction*, so-called because its solution can be found without needing to solve on the fine discretisation level, and the second term is known as the *remaining error*.

It can be seen that,
\[
\|L_F^C \psi_C R_F(x_F^C, u)\| \leq N \sum_{k=0}^{N} \| (L_F^C \psi_C)_k (R_F(x_F^C, u))_k \| \tag{7.11}
\]
where, \((L_F^C \psi_C)_k\) and \((R_F(x_F^C, u))_k\) are the \(k^{th}\) components of each of the respective vectors.

Let \(\varepsilon_j\) be the error estimate of the cost function for each discretised time interval \(t_j - t_{j-1}\),
\[
\varepsilon_j := \sum_{k} \| (L_F^C \psi_C)_k (R_F(x_F^C, u))_k \| \tag{7.12}
\]
where, \(k\) is given by (7.1). Equation (7.12) estimates the discretisation error in each control time interval and can be used to alter the state discretisation level. Note that as in [66], the error estimate developed here is based on just the computable correction. Although it is also possible to estimate the remaining error, doing so incurs a higher computational cost.
7.4 Proposed algorithm

For each control iteration, the standard collocated NMPC algorithm consists of transcribing the optimal control problem into an NLP, solving the NLP and applying the first element of the input \( u \) as the control input to the plant. To reduce the computational burden of solving the NLP, the proposed algorithm modifies the standard algorithm by adapting the number of states \( x_k \) at each control iteration. The adaptation is driven by the error estimate of the cost function, and this means the accuracy of the solution with respect to the cost function should not suffer when calculating a (potentially) faster and smaller NLP solution.

The following description outlines the main steps of the proposed algorithm. Refinement and coarsening thresholds \( \varepsilon_{\text{lower}} \) and \( \varepsilon_{\text{upper}} \) are defined respectively. When the error estimate \( \varepsilon_j \) in (7.12) exceeds either of these thresholds, then an adaptation rule is applied to change the discretisation level. In order to limit the level of coarsening and refinement which is determined by the vector \( \tau \), \( \tau_{\text{min}} \) and \( \tau_{\text{max}} \) are also defined respectively.

**Algorithm 7.1.** The steps of the algorithm are described as follows:

1. Set \( \varepsilon_{\text{lower}}, \varepsilon_{\text{upper}}, \tau_{\text{min}} \) and \( \tau_{\text{max}} \).
2. Initialise each \( \tau_j \) to \( \frac{\tau_{\text{max}}}{2} \).
3. For each control iteration, \( i < i_{\text{max}} \):
   
   (a) Transcribe the optimal control problem defined by equations (3.22)–(3.24) into discretised form given by (7.2)–(7.4) using the current discretisation level given by \( \tau \).
   
   (b) Solve the optimal control problem to produce the optimal input and state trajectories, \( \{u_j, x_{C,k}\} \).
   
   (c) Apply the first element of \( u_j \) as the control input to the plant.
   
   (d) Assume that the state solution in each \( t_j - t_{j-1} \) time interval is to be refined and produce an interpolated solution \( \{u_j, x_{F,k}^C\} \) and \( \tau_{\text{int}} \).
   
   (e) Use the interpolated solution \( \{u_j, x_{F,k}^C\} \) to determine the error estimate \( \varepsilon_j \) using (7.12).
   
   (f) Determine the new refinement level for each element where,

   \[
   \tau_j = \begin{cases} 
   \tau_j - 1, & \text{if } \varepsilon_j < \varepsilon_{\text{lower}} \text{ and } \tau_j > \tau_{\text{min}} \\
   \tau_j + 1, & \text{if } \varepsilon_j > \varepsilon_{\text{upper}} \text{ and } \tau_j < \tau_{\text{max}} \\
   \tau_j, & \text{otherwise}
   \end{cases}
   \quad (7.13)
   \]
(g) Increment the control iteration $i$.

Steps (3a) to (3c) of Algorithm 7.1 form the standard steps in solving an NMPC problem. The refinement part is carried out in steps (3d) to (3f). In step (3d) the interpolated solution is found through interpolation methods and not by solving the optimal control problem at the refinement level of $\tau_{int}$. The refinement strategy (7.13) in step (3f) increases or decreases the discretisation level by one, but alternative strategies such as doubling or halving the discretisation level or varying the level proportionately to the magnitude of the error could also be adopted. The tuning parameters $\varepsilon_{\text{lower}}$ and $\varepsilon_{\text{upper}}$ control the sensitivity of the refinement strategy with respect the error estimate. The parameters $\tau_{\text{min}}$ and $\tau_{\text{max}}$ act as hard stops for the refinement strategy to prevent runaway conditions.

Remark 7.4. One method to speed up the computation of the solution is to “warm start” the solution by initialising the current time step with the solution calculated at the previous time step. Additionally, the solution at the previous time step can be shifted right during initialisation so that it better represents the solution at the current time. For a non-uniform mesh, some interpolation of the solution is required along with shifting. Similarly, the mesh refined at the previous time step can be used to “warm start” the mesh at the current time step. Indeed, to reduce the computational overhead the mesh is only refined once per control iteration.

7.5 Results

7.5.1 Two-state system

The first example is a two-state nonlinear system taken from [24].

\[
\frac{d}{dt} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.2x_2 + 0.1\sin(t) \\ 0.2\sin(x_1) + u \end{bmatrix} \tag{7.14}
\]

The cost function applied to this system is,

\[
J = \sum_k (t_k - t_{k-1})x_k^2 + \sum_j (t_j - t_{j-1})u_j^2. \tag{7.15}
\]

The system is subjected to a total simulation time of $T = 12.5$ seconds. The NMPC controller uses a prediction horizon of $T_f = 6$ seconds, control discretisation of $N = 25$, a minimum discretisation level of $\tau_{\text{min}} = 1$, a maximum discretisation level of $\tau_{\text{max}} = 10$, $\varepsilon_{\text{upper}} = 1 \times 10^{-3}$ and $\varepsilon_{\text{lower}} = 5 \times 10^{-4}$. 
7.5. RESULTS

Figure 7.2 shows the states and inputs against time and also $x_2$ against $x_1$. It can be seen that the controlled system stabilises at a limit cycle around the origin. Figure 7.3 shows the discretisation level $\tau$ represented as a shade with prediction horizon along the horizontal axis and time along the vertical axis. Note that the mesh is adapted quite aggressively during the first 2 seconds of simulation, but much less after this point. Also note that the higher level of discretisation at the end of the prediction horizon suggests that the adaptive mesh is accounting for the terminal limit cycle behaviour.

![Figure 7.2: Two state system](image-url)
In order to evaluate the computation load of the adaptive mesh algorithm, the computational time is measured. The algorithm is compared against an NMPC controller without adaptation with the discretisation level set to $\tau_{\text{max}}$. As this is an indirect measurement of the computational load, ten separate runs are conducted and the average and standard deviation time of the runs are shown in Table 7.1. Also shown is the open loop cost function evaluated using the closed loop data. The table shows that the average computational time for
### 7.5. Results

<table>
<thead>
<tr>
<th></th>
<th>Mean time (sec)</th>
<th>Standard dev. time (sec)</th>
<th>Cost function</th>
</tr>
</thead>
<tbody>
<tr>
<td>With adaptation using Algorithm 1</td>
<td>8.68</td>
<td>0.22</td>
<td>6.41</td>
</tr>
<tr>
<td>With constant temporal resolution</td>
<td>17.46</td>
<td>0.51</td>
<td>6.41</td>
</tr>
</tbody>
</table>

Table 7.1: Comparison of computational time and cost function evaluation for two state system (sec)

The adaptive mesh algorithm is approximately halved for the simple two state system. The variance of the computational time is also smaller with adaptation than without. The results also show that there are no differences in the cost function. Note that without adaptation the controller cannot run in real-time.

#### 7.5.2 Missile pitch axis system

The second example is a pitch axis missile model. The longitudinal pitching motion of a tail-fin missile is a non-minimum phase system with unstable zero-dynamics and this makes it a non-trivial control problem for many control methods. The performance of the NMPC controller with two other controllers will be conducted. That is, a PI controller and a SDRE controller.

The pitch axis model is a combination of (3.6) and (3.8) and is given by,

\[
\frac{d}{dt} \begin{bmatrix} \alpha \\ q \\ \delta \\ \dot{\delta} \\ \delta_c \end{bmatrix} = \begin{bmatrix} K_aMC_a(\alpha,\delta) \cos(\alpha) + q \\ K_qM^2C_m(\alpha,\delta) \\ \dot{\delta} \\ -\omega_a^2 \delta - 2\zeta\omega_a \dot{\delta} + \omega_a^2 \delta_c \\ u \end{bmatrix}
\]

(7.16)

The missile’s normal acceleration is given by (3.10). Furthermore, let the pitch rate, \(q\), be constrained to,

\[-0.2 \leq q \leq 0.2\]

(7.17)

#### 7.5.2.1 PI autopilot

The first autopilot is a variant of a PI controller with pitch rate as feedback. The control input for the PI controller is given by,

\[
u_{PI} = P_c \eta + \int I_c \eta dt + P_q q
\]

(7.18)

where, \(P_c\) is the proportional gain of the acceleration, \(I_c\) is the integral gain of the acceleration and \(P_q\) is the proportional gain for the pitch rate.
7.5.2.2 SDRE autopilot

The second controller is based on the state dependent Ricatti equation approach. The state vector for this controller is given by,

\[
x_{SD} = \begin{bmatrix} \alpha & q & \delta & \dot{\delta} & \delta_c \end{bmatrix}^T \tag{7.19}
\]

Define the state error as,

\[
e_{SD} = x_{SD} - x_d \tag{7.20}
\]

where, \(x_d\) is the desired state for a given \(\eta\) normal acceleration. The general state-dependant coefficient form is,

\[
\dot{e}_{SD} = A(e_{SD})e_{SD} + B(e_{SD})u_{SD} \tag{7.21}
\]

Using direct parameterisation on Equation 7.16,

\[
A(e_{SD}) = \begin{bmatrix}
K_n M (a_n \alpha^2 + b_n |\alpha| + c_n (2 - \frac{a_n}{d_n})) \cos(\alpha) & 1 & d_n \cos(\alpha) & 0 & 0 \\
K_n M^2 (a_m \alpha^2 + b_m |\alpha| + c_m (-7 + \frac{a_m}{d_m})) & 0 & d_m & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{7.22}
\]

\[
B(e_{SD}) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T \tag{7.23}
\]

with,

\[
Q(e_{SD}) = \begin{bmatrix} 40 + \alpha^2 & 0 & 0 & 0 & 0 \\
0 & 10 + q^2 & 0 & 0 & 0 \\
0 & 0 & 20 + \delta^2 & 0 & 0 \\
0 & 0 & 0 & 1 + \dot{\delta}^2 & 0 \\
0 & 0 & 0 & 0 & 1 + \delta_c^2
\end{bmatrix} \tag{7.24}
\]

\[
R(e_{SD}) = 1 \tag{7.25}
\]

The control is given by,

\[
u_{SD} = -R(e_{SD})^{-1}B(e_{SD})^TP(e_{SD})e_{SD} \tag{7.26}
\]

where \(P(e_{SD})\) is the unique, positive-definite solution to the algebraic state dependent Ricatti equation and satisfies,

\[
P(e_{SD})A(e_{SD}) + A(e_{SD})^TP(e_{SD}) - 
P(e_{SD})B(e_{SD})R(e_{SD})^{-1}B(e_{SD})^TP(e_{SD}) + Q(e_{SD}) = 0 \tag{7.27}
\]
Note that in order to meet the state constraint defined by Equation 7.17, the PI and SDRE controllers need to be suitably “de-tuned” so that the pitch rate does not exceed the imposed limits.

### 7.5.2.3 NMPC autopilot

For the NMPC controller, the pitch rate constraints can be incorporated explicitly into the formulation. The cost function for the controller is,

\[
J = \sum_{k} (t_{k} - t_{k-1}) \left( \beta_{1}(\alpha_{k} - \alpha_{c})^2 + \beta_{2}(q_{k} - q_{c})^2 + \beta_{3}(\delta_{k} - \delta_{c})^2 + \beta_{4}\dot{\delta}_{k}^2 \right) + \\
\sum_{j} (t_{j} - t_{j-1}) \left( \beta_{5}(u_{j} - \delta_{c})^2 \right)
\]

(7.28)

where the \( \beta \)'s are weights associated with each of the states.

### 7.5.2.4 Simulation Results

The NMPC controller has a prediction horizon \( T_{f} = 10 \) seconds and control discretisation of \( N = 30 \). A minimum discretisation level of \( \tau_{\text{min}} = 1 \) and a maximum discretisation level \( \tau_{\text{max}} = 10 \) is utilised. The error thresholds are set to \( \varepsilon_{\text{upper}} = 2.5 \times 10^{-5} \) and \( \varepsilon_{\text{lower}} = 5 \times 10^{-6} \). The desired normal acceleration is set to \( \eta_{c} = 10g \).

The total simulation time for all three controllers is \( T = 10 \) seconds. These results are shown in Figure 7.4. Note that all three controllers achieve the desired normal acceleration and remain within the pitch rate constraint.

Figure 7.5 shows the discretisation level \( \tau \) and it can be seen that aggressive discretisation occurs in the first 5 seconds of the controller where there is rapid actuation and state changes and much less discretisation is required once the missile is in steady state.

The computational load of the proposed algorithm for the missile example is also compared against an NMPC controller without adaptation. The average and standard deviation time of ten separate runs are shown in Table 7.2. The

<table>
<thead>
<tr>
<th></th>
<th>Mean time (sec)</th>
<th>Standard dev. time (sec)</th>
<th>Cost function</th>
</tr>
</thead>
<tbody>
<tr>
<td>With adaptation using Algorithm 1</td>
<td>13.13</td>
<td>0.33</td>
<td>3.09</td>
</tr>
<tr>
<td>With constant temporal resolution</td>
<td>19.29</td>
<td>0.47</td>
<td>3.06</td>
</tr>
</tbody>
</table>

Table 7.2: Comparison of computational time and cost function evaluation for missile example (sec)
Table shows that the average computational time for the adaptive mesh is two-thirds of the non-adapted time for the missile system. The variances again are lower in the adapted case over the non-adapted algorithm and there are minimal differences in the cost function.

7.6 Discussion

The PI autopilot is the simplest controller to implement and this controller can also be “gain scheduled” in order to extend the performance for nonlinear systems. However as shown in the results, in order to meet the constraints there are some trade-offs with the controller’s transient performance. Furthermore, tuning of the gains for each operating condition can be a labour intensive process which results in extensive system verification and validation. The controller also requires that the pitch rate be measured or estimated and then fed back into the control input [107]. There are also no guarantees of closed-loop stability when using this method.

The SDRE autopilot faces similar gain tuning issues as the PI autopilot. Indeed, there are many more parameters in \( Q \) and \( R \) matrix that may need to defined for different operating conditions. The SDRE controller also requires full-state feedback [65], with the state information gathered either through mea-
7.7 Conclusion

In this chapter, an adaptive mesh algorithm used primarily on CFD problems is applied to direct collated non-linear model predictive control problems. The refinement is based on goal-oriented error estimation which means the refinement of the control problem relates directly to the objective function rather than refinement based on the system dynamics or other heuristics. This is desirable because the cost function for NMPC problems relate directly to the performance of the closed-loop system.
The presented algorithm is applicable to a range of NMPC problems without restriction. Improvements of the computational time of up to 50% was observed in the given examples and this shows that a tangible benefit for real-world NMPC problems. In particular, this result builds toward the practical use of the NMPC algorithm for missile autopilots where the computation time of the autopilot is reduced by a third. The advantages of NMPC with its explicit handling of constraints over other control methods cannot be understated.

A natural extension for this mesh adaptation algorithm is in moving horizon estimator (MHE) problems. Missile sensors, such as gyroscopes and accelerometers, could very well benefit from a mesh adapted MHE. Further work could also consider refinement for each state separately (i.e. refinement on separate channels). $hp$-refinement strategies may also be incorporated into the refinement algorithm.
Chapter 8

Conclusion and Further Work

This thesis has explored utilising multidisciplinary optimisation frameworks in order find the optimal design for aerodynamic systems. Currently, the systems engineering methodology is widely used in the aerospace industry to manage the design process of complex aerodynamic systems, however a severe limitation is that there are no guarantees that resulting designs are optimal in any sense.

The optimisation frameworks developed in this thesis have targeted two important aspects of a missile system, namely, the aerodynamic shape and the autopilot controller. These are two highly coupled systems that have often been designed separately by different engineering teams. This has often limited the final design as the competing constraints of the aerodynamics and manoeuvrability cannot be explicitly accounted for in the early stages of the design process.

The contributions in this thesis show that there are clear and practical advantages of utilising MDO to automate the design process. Furthermore, by harnessing the properties of the aerodynamic system under design, the frameworks in this thesis show an improvement to standard optimisation frameworks in terms of computational speed.

8.1 Summary of contributions

1. Efficient application of a global optimisation algorithm to CFD aerodynamic problems

CFD-based aerodynamic shape optimisation problems have rarely made efficient use of global optimisation techniques in the past. These problems have heavy computational burdens that require many CFD simulations to be evaluated and
computing a global solution in a timely manner has often proved impractical. The advantage of a global optimiser is that it does not become entrapped in local minima.

In Chapter 4, the conditions where the convergence of an iterative CFD solver mimics that of a dynamic plant were outlined. Combining the convergence property of the CFD solver with global extremum seeking has then allowed its application in aerodynamic shape optimisation problems. To the author’s knowledge this is the first application of extremum seeking in this context. The results show that extremum seeking improved the total computational time when compared against traditional global optimisation techniques.

A global optimisation method called DIRECT-L that utilises knowledge of the cost function’s Lipschitz constant was also developed and this has further reduced the computational requirements in these types of problems with the solution guaranteed to be within the vicinity of the true global minimum. DIRECT-L has made global optimisation comparable to local optimisation techniques in terms of computation without its inherit limitations at least for small dimensional problems.

2. Multi-disciplinary optimisation of coupled missile system using gradient-based optimisers

In the context of MDO for coupled aerodynamic shape and autopilot design, a continuous adjoint based optimisation framework was developed in Chapter 5 that optimised the shape of the missile as well as the controller parameters. Adjoint-based aerodynamic optimisation which has primarily been used for static optimisation has now been extended to dynamic systems.

The framework exploits time-scale separation between the fluid states and the actuator states so that steady CFD simulations can be utilised in a dynamic scenario. As the framework is based on the adjoint method, the calculation of the gradient of the cost function in this dynamic context has been shown to require only double the number of CFD simulations (when comparing to the primal CFD simulations) irrespective of the number of design variables. This means that the dynamic missile optimisation framework is able to cater for realistic problems with complex multivariate geometries where the design is guaranteed to be a local optimal solution.

One limitation of the continuous adjoint based framework is its inability to deal with sharp edged geometry and this problem is elaborated upon in Chapter 6. Thus the dynamic optimisation framework was further extended by incorporating implicit filtering (a noise-tolerant gradient optimisation algorithm) and exploits closeness properties of partially converged iterative CFD solutions so that both sharp and smooth geometries can be efficiently optimised. The extended framework is both computationally efficient and is capable of optimising
aerodynamic shape and controller design problems with little restrictions. This makes the extended framework suitable for general closed-loop aerodynamic optimisation problems. If the properties of the plant and controller hold, the framework guarantees that the solution will converge to a local minimum of the cost function.

The examples shown in Chapter 5 and 6 are representative of realistic missile optimisation problems. It is the optimisation of the shape of a tail-fin missile undergoing single-axis pitching motion. The framework can also cater to other missile parts and manoeuvres.

3. Development of adaptive mesh for non-linear model predictive control
Model predictive control has relatively high computational requirements when compared with other control algorithms. Nonlinear model predictive control has even higher computational requirements. However, the advantages of NMPC is that they are capable of modelling non-linear systems such as missiles more accurately, and like linear MPC they can explicitly take system constraints into account.

In Chapter 5, an algorithm was developed that dynamically optimises the model in the NMPC in order to reduce the computational cost of running the controller. This result was based on adjoint-based mesh adaptation techniques that are well known in the fluid-dynamics community and have been re-purposed in an NMPC context. Goal-oriented mesh adaptation, which adapts the mesh in CFD problems so that they are optimised for some fluid property (i.e. cost function), was applied to adapt the prediction horizon of a collocated NMPC. It was shown, though examples, that the computation of the non-linear model is up to 50% more efficient by adapting the mesh using goal-oriented mesh adaptation than by leaving the horizon discretisation fixed. This brings non-linear model predictive control a step closer to being used in so-called “fast systems” such as missile controllers.

4. High fidelity modelling of a tail-fin controlled missile
The results shown in Chapter 5 and 6 required the development of a high fidelity dynamics model that not only modelled the aerodynamics, flight dynamics and control of a tail-fin controlled missile, but could also take in geometric and controller design variables that were automatically manipulated and the performance evaluated by the optimisers. The missile model developed in Chapter 3 couples together a 3DOF state space model of the missile dynamics, deformable 3D models of the missile’s geometry, CFD-based aerodynamic models and control system algorithms. It has the capability of driving the missile under various manoeuvring scenarios.

The model was developed using the following software packages, namely,
MATLAB, Missile DATCOM, SU2, OpenVSP, gmsh and python. Apart from MATLAB, all the remaining software are open source software packages which lends to its attractiveness to be adopted by other research groups and by the aerospace industry.

The missile model developed is realistic and extensible. By a minor modification of the 3-DOF to a full 6-DOF model and simply altering of the missile geometry, design variables and commanded manoeuvres it has the capacity to cater to a wide variety of missile types and operating conditions.

8.2 Contributions

The following journal and conference papers have been submitted as part of the research contributions to the literature.

8.2.1 Journal papers


8.2.2 Conference papers


8.3 Further work

It is this author’s hope that the research presented herein will encourage further adoption of high-fidelity modelling in engineering design so as to minimise inefficiencies in manual design processes. Aerospace and defence engineering firms are always looking for an advantage to shorten the time-to-market on products under development. Not only is timing important, these products need to be better than those developed by competitor firms. The MDO frameworks described can give a competitive advantage and this thesis contributes toward gaining this advantage. However much more work is necessary to bring the tools and ideas developed here to a commercial engineering design environment.

In terms of research, further work and various extensions to the optimisation frameworks presented are possible. The DIRECT and DIRECT-L algorithms are but few examples within sampled-data extremum seeking. Other modifications or algorithms can be investigated within the global optimisation space. Additionally, this thesis has explored the use of the continuous adjoint method and implicit filtering for the gradient-based optimisation frameworks. Other gradient methods, such as the discrete adjoint method can also be investigated. The general properties that are necessary for these frameworks to work means that they can also be easily adapted for problems other than aerodynamic shape optimisation. For instance, shape optimisation of rotors and stators for controlled electric motors or for inlet and outlet shapes of jet engines.

It is also hoped that the efficiency gains from the adaptive NMPC result encourages adoption of NMPC for so-called “fast systems”. There are many aspects of adaptive mesh refinement that can be further investigated for use in receding horizon control and estimation. In this thesis, adaptive refinement was only applied to refining the plant model of a missile autopilot. Other aerospace models such as inertial navigation systems could benefit from the refinement algorithm developed here. The adaptation was only applied to refine the plant states in prediction horizon. Further work could be carried out to see if the control inputs could be refined in the same way.
Appendix A

Gradient Methods

A.1 The adjoint method

The adjoint method is a means of calculating the gradient of a scalar function with respect to the design variables. The derivation of the adjoint equations following from [68] and [21] are reproduced here.

Consider an scalar aerodynamic property $F$, that is a function of the flow-field quantities, $U$, and design variables $\theta_f$.

$$ F = F(U, \theta_f). \quad (A.1) $$

By calculus of variations a change in $\theta_f$ results in a change in the scalar function,

$$ \delta F = \frac{\partial F}{\partial U} \delta U + \frac{\partial F}{\partial \theta_f} \delta \theta_f. \quad (A.2) $$

It is expensive to compute variations in the flow-field quantities, $\delta U$, that is, each variation will require an additional CFD simulation. The aim of the adjoint approach is to eliminate this term in (A.2). Suppose that the governing equations of the flow are introduced in the form of an equality constraint,

$$ R(U, \theta_f) = 0. \quad (A.3) $$

For example, $R(U, \theta_f)$ could be the conservative form of the compressible Euler equations. The variation in (A.3) is,

$$ \delta R = \left[ \frac{\partial R}{\partial U} \right] \delta U + \left[ \frac{\partial R}{\partial \theta_f} \right] \delta \theta_f = 0. \quad (A.4) $$

Equation (A.2) can be combined with (A.4) via a Lagrange Multiplier, $\psi$, which
GRADIENT METHODS

\[ \delta F = \frac{\partial F}{\partial U} \delta U + \frac{\partial F}{\partial \theta_f} \delta \theta_f - \psi \left( \left[ \frac{\partial R}{\partial U} \right] \delta U + \left[ \frac{\partial R}{\partial \theta_f} \right] \delta \theta_f \right) \]

\[ = \left\{ \frac{\partial F}{\partial U} - \psi \left[ \frac{\partial R}{\partial U} \right] \right\} \delta U + \left\{ \frac{\partial F}{\partial \theta_f} - \psi \left[ \frac{\partial R}{\partial \theta_f} \right] \right\} \delta \theta_f. \tag{A.5} \]

Suppose \( \psi \) is chosen such that,

\[ \left[ \frac{\partial R}{\partial U} \right] \psi = \frac{\partial F}{\partial U}. \tag{A.6} \]

Equation (A.6) is called the adjoint equation, and with an appropriately defined flow field and boundary conditions, this adjoint PDE can be implemented and solved using the same numerical computer code that is used to solve the governing equations of the flow.

Substituting (A.6) into (A.5) eliminates the term \( \delta U \). The change in the cost functional is,

\[ \delta F = G \delta \theta_f, \tag{A.7} \]

where the gradient, \( G \), is given by,

\[ G = \frac{\partial F}{\partial \theta_f} - \psi \left[ \frac{\partial R}{\partial \theta_f} \right]. \tag{A.8} \]

A.2 Least squares finite difference method

The least squares finite difference method calculates the gradient based on a linear least squares fit of sampled points. Let \( \mathbf{V} = \{v_1, \ldots, v_K\} \) be the stencil search directions. For a central differencing stencil, the search directions are \( \mathbf{V} = \{I_N, -I_N\} \) and \( K = 2N \). To calculate the gradient, the cost function is evaluated at some point \( \mathbf{\theta} \) in the design space and points on the stencil \( \mathbf{S}(\mathbf{\theta}, h, \mathbf{V}) = \{Z|Z = \mathbf{\theta} + h\mathbf{v}_k, 1 \leq k \leq K\} \), where \( h \) is the finite difference step size. This gives the gradient

\[ \frac{dJ}{d\mathbf{\theta}} = \frac{1}{h} \delta J(\mathbf{\theta}, h, \mathbf{V}) \mathbf{V}^\dagger, \tag{A.9} \]

where

\[ \delta J(\mathbf{\theta}, h, \mathbf{V}) = \begin{bmatrix} J(\mathbf{\theta} + h\mathbf{v}_1) - J(\mathbf{\theta}) \\ J(\mathbf{\theta} + h\mathbf{v}_2) - J(\mathbf{\theta}) \\ \vdots \\ J(\mathbf{\theta} + h\mathbf{v}_K) - J(\mathbf{\theta}) \end{bmatrix}. \tag{A.10} \]
and $V^\dagger := V^T(VV^T)^{-1}$ is the pseudoinverse of $V$. Note that a central differencing stencil requires $2N + 1$ simulations.
Appendix B

Optimisation Methods

B.1 Implicit filtering

Implicit filtering is a local optimisation method that utilises the gradient of the cost function to build a quadratic model. The quadratic model allows the method to build up an approximate gradient even if the underlying gradient is noisy.

1. Note the upper and lower bounds of all design variables and scale them such that \( \Omega = \{ \theta \in \mathbb{R}^M | 0 \leq \theta_j \leq 1 \} \) for \( j = \{1, 2, \ldots, M\} \). Let the initial design be scaled to \( \theta_0 \in \Omega \). Scale the cost function \( J(\cdot) \) by \( \frac{1}{1 + J(\theta_0)} \).

2. Let \( Q \) be the maximum number of iterations of the algorithm. Let \( \Gamma = [\gamma_1, \gamma_2, \ldots, \gamma_Q] \) be an error vector to control the error of the computed cost function. It is required that the selection of successive \( \gamma \) values be increasing so that the noise level decrease as the algorithm proceeds.

3. Initialise the current design point \( \theta_c = \theta_0 \), the step size \( h = 0.5 \), error control \( \gamma = \gamma_1 \), the Hessian \( H = I \) and iteration level \( k = 1 \).

4. Let \( \tau > 0 \) be the termination constant. The value \( ||\tau h|| \) is used to determine when to terminate the search at each iteration level \( k \).

5. While \( k \leq Q \):

   (a) Evaluate the cost function at \( J(\theta_c, \gamma) \).

   (b) Compute the gradient \( \nabla J(\theta_c, \gamma) \). The gradient can be calculated, for example, by using either least squares finite differencing (A.9) or the adjoint method (A.8). If least squares finite differencing is used, then store the value of the cost function at each stencil point as well.
(c) If $||\theta_c - P(\theta_c - \nabla J(\theta_c, \gamma))|| < \tau h$ go to step (5f), where $P(\cdot)$ is the projection operator.

(d) Set the line search design point $\theta_+ = \theta_c$.

(e) While $||\theta_+ - P(\theta_+ - \nabla J(\theta_+, \gamma))|| \geq \tau h$:
   i. Update the model Hessian $H$ using the BFGS quasi-Newton method [71].
   ii. Determine the search direction,
   \[ s = -H^{-1}\nabla J(\theta_+) \]
   iii. Perform a backtracking line search
   \[ \theta_+ = P(\theta_+ - \lambda s), \]  
   where $\lambda$ is initialised as 1 and reduced by a factor of $\beta$ at each line search iteration. There exists a maximum number of iterations and if this is exceeded the line search is said to have failed.

(f) Select the best point, $\theta_{f,min}$ from the stencil (if least squares finite differencing is used) or from the successful line search and set $\theta_c = \theta_{f,min}$, $h = h/2$, $\gamma = \gamma_{k+1}$ and $k = k + 1$. 
Bibliography


[70] Robert A. Nichols and Robert T. Reichert. “Gain Scheduling for $H$-
Infinity Controllers: A Flight Control Example”. In: Transactions on
Control Systems Technology 1.2 (June 1993), pp. 69–79.


[72] Shigeru Obayashi et al. “Multiobjective Evolutionary Computation for
Supersonic Wing-Shape Optimization”. In: IEEE Transactions on Evolu-

[73] S. L. Padula and R. K. Kincaid. Aerospace Applications of Integer and

[74] F. Palacios et al. “Stanford University Unstructured (SU2): An open-
source integrated computational environment for multi-physics simula-
tion and design”. In: 51st AIAA Aerospace Sciences Meeting including
the New Horizons Forum and Aerospace Exposition. AIAA. Grapvine,
TX, Jan. 2013.

[75] Neil F. Palumbo, Ross A. Blauwkamp, and Justin M. Lloyd. “Modern
Homing Missile Guidance Theory and Techniques”. In: John Hopkins

2010.

[77] S. Joe Qin and Thomas A. Badgwell. “A survey of industrial model pre-
dictive control technology”. In: Control Engineering Practice 11.7 (July

Methods to Model Predictive Control”. In: Journal of Optimization The-

Optimization Using an Adjoint Formulation”. In: International Mecha-

[80] James Reuther et al. “Constrained Multipoint Aerodynamic Shape Op-
timization Using an Adjoint Formulation and Parallel Computers”. In:


[82] Wilson J. Rugh and Jeff S. Shamma. “Research on gain scheduling”. In:


Author/s:
Lee, Kuan Waey

Title:
Optimisation of closed-loop aerodynamic systems

Date:
2018

Persistent Link:
http://hdl.handle.net/11343/214509

File Description:
Optimisation of closed-loop aerodynamic systems

Terms and Conditions:
Terms and Conditions: Copyright in works deposited in Minerva Access is retained by the copyright owner. The work may not be altered without permission from the copyright owner. Readers may only download, print and save electronic copies of whole works for their own personal non-commercial use. Any use that exceeds these limits requires permission from the copyright owner. Attribution is essential when quoting or paraphrasing from these works.