An optimization framework for the co-design of controlled aerodynamic systems

Kuan Waey Lee[1], William Moase[2], Andrew Ooi[3] and Chris Manzie[4]
The University of Melbourne, Melbourne, Victoria 3010, Australia

Eric C. Kerrigan[5]
Imperial College London, London, SW7 2AZ, United Kingdom

Optimization studies of dynamic systems utilizing high-fidelity numerical models necessitate a trade-off between fidelity and the total computational time required during design. A gradient-based optimization framework is proposed for the aerodynamic shape and controller design of aerodynamic systems using computationally intensive high fidelity models. Subject to some general properties, the framework offers flexibility in the types of simulation models used and provides guarantees regarding closeness to an optimal design. A nested optimization loop is implemented, which allows for the partitioning of controller and plant architecture. The proposed framework exploits time-scale properties of the dynamic system model, closeness properties of partially converged iterative solutions of computational fluid dynamics models and the continuous adjoint method. Utilizing these methods, it is shown that this can improve the total computational time relative to finite differencing. An example of optimizing the aerodynamic body and control gains of a tail-fin controlled supersonic missile is presented.

[1] Student, Department of Mechanical Engineering, The University of Melbourne, Victoria, Australia, 3010
[2] Postdoctorate, Department of Mechanical Engineering, The University of Melbourne, Victoria 3010, Australia
[3] Professor, Department of Mechanical Engineering, The University of Melbourne, Victoria, Australia, 3010
[4] Professor, Department of Mechanical Engineering, The University of Melbourne, Victoria 3010, Australia
[5] Reader in Control Engineering and Optimization, Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, United Kingdom

Nomenclature

$D(t)$	Instantaneous drag, N
$e(t)$	Normal acceleration tracking error, ms^{-2}
$\tilde{E}$	Dimensionless specific total energy
$\mathbf{f}$	Dimensionless convective flux
$\hat{\mathbf{F}}$	Aerodynamic force vector, N
g	Acceleration due to gravity, ms^{-2}
G_m	System gain margin, dB
G_m^*	System gain margin lower bound, dB
I_{yy}	Second mass moment of inertia on pitch axis, kgm^2
J	Cost function
$\nabla\hat{J}$	Computed gradient
∇J	True gradient
K_A	Acceleration loop controller gain
K_{DC}	DC controller gain
K_R	Pitch rate loop controller gain
l_r	Characteristic length of tail fin
L	Lift force, N
m	Mass, m
M	Pitching moment, Nm
$\mathbf{n}$	Normal vector of each face
$\mathbf{n}_S$	Normal surface vector
N	Number of design variables
q	Pitch rate, rads^{-1}
$\mathbf{r}(t)$	Reference trajectory
s	Surface coordinate
S	Baseline surface
S'	Perturbed surface

S_i	Cell surface area of ith cell, m^2
$\mathbf{u}(t)$	Controller inputs
V	Missile speed, ms^{-1}
$\tilde{\mathbf{V}}$	Dimensionless fluid velocity vector
V_i	Cell volume of ith cell, m^3
w	Cost function weighting constant
W_I	Integral loop controller gain
$\mathbf{x}_a(t)$	"Slow" system states
$\mathbf{x}_b(t)$	Controller states
X	Shape deformation variable
$\mathbf{z}(t)$	"Fast" system states
$\bar{\mathbf{z}}(t)$	Quasi-steady solution of $\mathbf{z}(t)$
$\mathbf{z}_{h,i}(t)$	Volume averaged solution of $\mathbf{z}(t)$
α	Angle of attack, rad
δ	Achieved fin deflection, rad
δ_c	Commanded fin deflection, rad
$\dot{\delta}_{max}$	Maximum fin deflection rate, rads^{-1}
δ_z	Fin deflection rate, rads^{-1}
γ	Noise parameter
γ_f	Flight path angle, rad
γ_s	Ratio of specific heats
ϵ	Time-scale constant
ε	Interpolation mapping error
ζ_a	Actuator damping ratio
ζ_c	Controller damping ratio
η	Normal acceleration, ms^{-2}
η_c	Commanded normal acceleration, ms^{-2}
$\boldsymbol{\theta}_{ad}$	Design variables utilizing the adjoint method

$\boldsymbol{\theta}_f$	Design variables of the "fast" states
$\boldsymbol{\theta}_{f,U}$	Upper bound for design variables of the "fast" states
$\boldsymbol{\theta}_{f,L}$	Lower bound for design variables of the "fast" states
$\boldsymbol{\theta}_k$	Design variables at iteration k of optimizer
$\boldsymbol{\theta}_{ls}$	Design variables utilizing least squares finite difference
$\boldsymbol{\theta}_s$	Design variables of the "slow" states
$\boldsymbol{\theta}_{s,U}$	Upper bound for design variables of the "slow" states
$\boldsymbol{\theta}_{s,L}$	Lower bound for design variables of the "slow" states
$\tilde{\rho}$	Dimensionless density of fluid
τ_c	Controller rise time, s
$\boldsymbol{\Phi}$	Additive noise
ω_a	Actuator undamped natural frequency, Hz
ω_{cr}	Controller open-loop crossover frequency, Hz

I. Introduction

The sequential and compartmentalized design of controlled systems is widely practiced despite the potential for better overall system designs by considering the plant and controller as a combined multidisciplinary optimization (MDO) problem. This is primarily attributable to the increased computational burden of MDO relative to the sequential alternative.

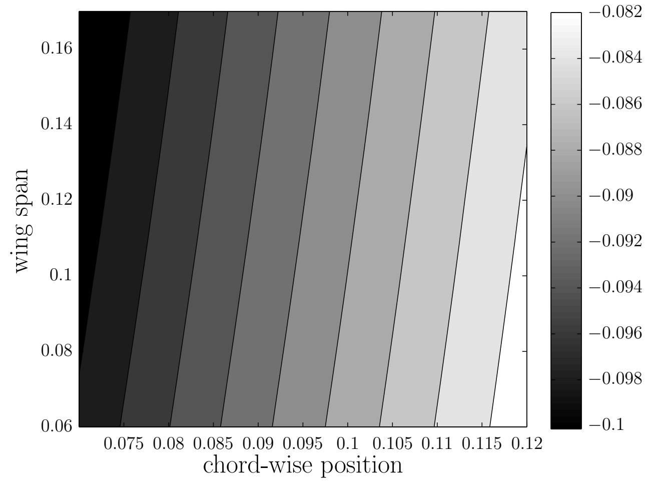
A wide range of (MDO) techniques exists [?], and can be classified as static or dynamic. The differentiation between the two is the concept of time. In static MDO, the main objective is system optimization under steady-state conditions, whereas central to dynamic MDO is the evolution of the system states. The optimization metric for dynamic systems can be an integral over a time period or Mayer-type term (e.g. mass) and is used to capture the dynamic system's performance. However, utilization of time-varying high fidelity numerical models poses a higher computational barrier than static models. It is for this same reason that, within the computational fluid dynamics (CFD) community, static properties such as lift and drag are routine computations, while dynamic properties like stability and control derivatives are not [?]. Even then, the computation requirements of static CFD solutions are not trivial.

While the benefits of high-fidelity modeling, such as CFD simulations are self-evident, one of the main barriers for their adoption in optimization is the combined computational requirements of evaluating and optimizing the design. Generally speaking, the computational requirements of simulation grow proportionally to the number of design variables. For optimization methods that use finite differences, much of the computation time is taken in the calculation of gradients. Finite differencing requires at least $N + 1$ simulations for a problem with N design variables.

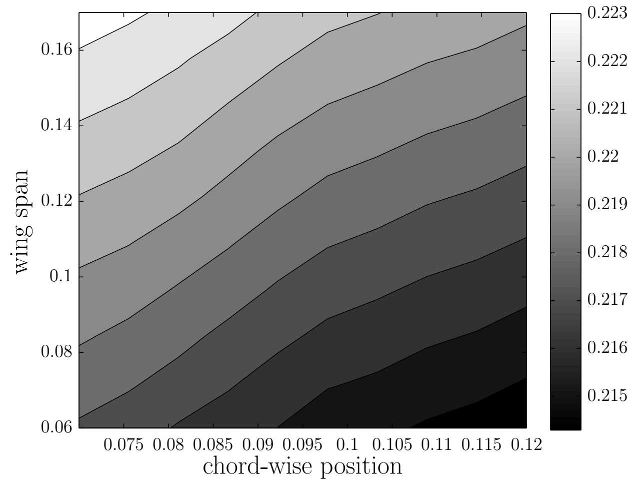
It was this problem with the computational cost of calculating the gradient that inspired Jameson to apply the adjoint method to CFD shape optimization [?] and demonstrate that gradient information could be retrieved with just two simulations regardless of the number of design variables. In CFD shape optimization problems, the two simulations are a primal system of partial differential equations (PDEs) representing the flow of the fluid, and an adjoint system derived from the primal system. 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 discretization and differentiation are performed. In either case, the gradients produced have been found to be the same when the discretization is sufficiently refined [?]. The adjoint method was first applied to single static operating conditions, and has since been extended to multi-point [?] and dynamic operating conditions [?].

A limitation of the *continuous* adjoint method is that it cannot handle design changes over sharp-edged geometries. Consider a simple example of the sensitivity of the lift of a supersonic fin. Figure 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 are not only of incorrect magnitude, but also of opposing sign. Gradient filtering and surrogate models have been proposed [?] 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 differentia-



(a) Adjoint method



(b) Finite differences

Fig. 1: Comparison of the gradient of the lift coefficient with respect to wing span calculated via the adjoint method and finite differences

tion 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 [**?**], and still requires some level of user interaction for the generated code to be computationally efficient [**?**]. Also, it is rare for multidisciplinary optimization problems to be programmed within the same simulator or in the same programming language. In the usual case, specialized simulators are coupled together in a co-design environment. This limits the applicability of the discrete adjoint method for multidisciplinary optimization problems and it is the reason it is not pursued here.

The computational requirements for global optimization techniques tend to be much higher than local techniques and this is because they require many more design evaluations in order to find the global optimum. Partial convergence of the CFD solver was utilized in a global optimization framework in order to speed up convergence to the optimal solution to a level comparable to gradient based methods [?]. However, global techniques do not scale well with the increase in design variables and cannot be used indiscriminately [? ?].

The motivation of this research is to develop an MDO framework for supersonic missiles that provides guarantees that the solution is optimal in some sense and allows arbitrary shapes to be incorporated into the design, while at the same time reducing the computational burden associated with high-fidelity model optimization. In order to provide the computational speed up, a number of properties of the system are necessary. Primarily, the system should display time-scale separation between the slow rigid body and fast flow field states. This allows for use of static CFD simulations in place of more costly dynamic CFD simulations.

The design optimization of a tail-fin controlled missile with a fixed controller architecture will be used throughout as a motivating example. The components of the model are shown in Figure 2. The design problem is posed as a nested optimization problem [?]. The design parameters are the

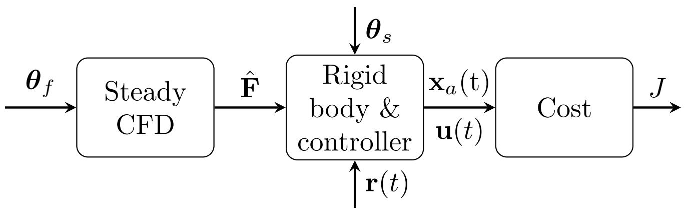


Fig. 2: Dynamic plant model

shape of the missile $\boldsymbol{\theta}_f$ and controller tuning parameters $\boldsymbol{\theta}_s$. Partitioning of the design variables will be made apparent in the following sections. Static CFD simulations are utilized to generate the aerodynamic forces $\hat{\mathbf{F}}$. Based on some reference trajectory $\mathbf{r}(t)$ the rigid body and controller model determines the evolution of the system states $\mathbf{x}_a(t)$ and controller inputs $\mathbf{u}(t)$ over some finite time, which is then used to calculate the cost function, J.

II. Dynamic Model

Consider a dynamic system model,

$$\dot{\mathbf{x}}_a(t) = \mathbf{f}_a(\mathbf{x}_a(t), \mathbf{x}_b(t), \mathbf{z}(t), \mathbf{u}(t), \epsilon) \tag{1a}$$

$$\dot{\mathbf{x}}_b(t) = \mathbf{f}_b(\mathbf{x}_a(t), \mathbf{x}_b(t), \mathbf{z}(t), \mathbf{r}(t), \boldsymbol{\theta}_s, \epsilon) \tag{1b}$$

$$\begin{aligned}\epsilon\,\frac{\partial \mathbf{z}(t)}{\partial t} = \mathbf{f}_z(\mathbf{x}_a(t), \mathbf{z}(t), &\frac{\partial \mathbf{z}(t)}{\partial x}, \frac{\partial \mathbf{z}(t)}{\partial y}, \frac{\partial \mathbf{z}(t)}{\partial z},\\ &\frac{\partial^2 \mathbf{z}(t)}{\partial x^2}, \frac{\partial^2 \mathbf{z}(t)}{\partial xy}, \ldots, \frac{\partial^2 \mathbf{z}(t)}{\partial z^2}, \ldots, \mathbf{u}(t), x, y, z, \boldsymbol{\theta}_f, \epsilon)\end{aligned} \tag{1c}$$

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

Property 1. *The control input* $\mathbf{u}(t) := \mathbf{g}(\mathbf{x}_a(t), \mathbf{x}_b(t), \mathbf{z}(t), \mathbf{r}(t), \boldsymbol{\theta}_s)$ *is designed such that* (1a)–(1b) *is asymptotically stable about a nominal reference trajectory* $\mathbf{r}(t)$*, uniform in* $\mathbf{z}(t)$*.*

If the system described by (1c) is asymptotically stable and there is time-scale separation between the coupled system $\{\mathbf{x}_a(t), \mathbf{x}_b(t)\}$ and $\mathbf{z}(t)$, then ϵ is small, and the coupled system is approximated by,

$$\dot{\mathbf{x}}_a(t) = \mathbf{f}_a(\mathbf{x}_a(t), \mathbf{x}_b(t), \mathbf{F}(\mathbf{x}_a(t), \mathbf{u}(t), \boldsymbol{\theta}_f), \mathbf{u}(t), 0) \tag{2a}$$

$$\dot{\mathbf{x}}_b(t) = \mathbf{f}_b(\mathbf{x}_a(t), \mathbf{x}_b(t), \mathbf{F}(\mathbf{x}_a(t), \mathbf{u}(t), \boldsymbol{\theta}_f), \mathbf{r}(t), \boldsymbol{\theta}_s, 0) \tag{2b}$$

$$\begin{aligned}0 = \mathbf{f}_z(\mathbf{x}_a(t), \mathbf{z}(t), &\frac{\partial \mathbf{z}(t)}{\partial x}, \frac{\partial \mathbf{z}(t)}{\partial y}, \frac{\partial \mathbf{z}(t)}{\partial z},\\ &\frac{\partial^2 \mathbf{z}(t)}{\partial x^2}, \frac{\partial^2 \mathbf{z}(t)}{\partial xy}, \ldots, \frac{\partial^2 \mathbf{z}(t)}{\partial z^2}, \ldots, \mathbf{u}(t), x, y, z, \boldsymbol{\theta}_f, 0)\end{aligned} \tag{2c}$$

Let the solution of (2a) and (2b) be defined as $\bar{\mathbf{x}}_a(t)$ and $\bar{\mathbf{x}}_b(t)$ respectively and let $\bar{\mathbf{z}}(t) := \mathbf{F}(\bar{\mathbf{x}}_a(t), \mathbf{u}(t), \boldsymbol{\theta}_f)$ be the quasi-steady solution of $\mathbf{z}(t)$.

Property 2. *The time scale separation parameter* ϵ *in* (1c) *is sufficiently small such that*

$$\mathbf{x}_a(t) - \bar{\mathbf{x}}_a(t) = O(\epsilon) \tag{3}$$

$$\mathbf{x}_b(t) - \bar{\mathbf{x}}_b(t) = O(\epsilon) \tag{4}$$

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

$$\mathbf{z}(t) - \bar{\mathbf{z}}(t) = O(\epsilon) \tag{5}$$

holds for $t > t_b$.

Singular perturbation theory has been previously applied to non-linear ordinary differential equation (ODE) systems [**?**] 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 utilized, that a similar time scale approximation can be made. To this end, the time scale separation parameter ϵ will be derived for the nominal operating conditions of supersonic flight and this will show that Property 2 is indeed a reasonable approximation.

A closed form solution for the map $\mathbf{F}(\cdot, \cdot, \cdot)$ 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 $\mathbf{F}(\cdot, \cdot, \cdot)$ over all values of $\mathbf{x}_a(t)$ and $\mathbf{u}(t)$.

Property 3. *For any $\varepsilon > 0$, there is an interpolated mapping $\hat{\mathbf{F}}(\cdot, \cdot, \cdot)$ where*

$$\left\| \hat{\mathbf{F}}(\mathbf{x}_a(t), \mathbf{u}(t), \boldsymbol{\theta}_f) - \mathbf{F}(\mathbf{x}_a(t), \mathbf{u}(t), \boldsymbol{\theta}_f) \right\| < \varepsilon,\ \forall \mathbf{x}_a(t), \mathbf{u}(t). \tag{6}$$

The variable ε in Property 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 utilized. If Properties 1–2 are satisfied, then the model in the optimization framework can be approximated by (2). Note that open-loop trajectories that are designed such that Property 1 holds can also be considered. If Property 3 is satisfied, for example via sufficient sampling, then the interpolated mapping $\hat{\mathbf{F}}(\cdot, \cdot, \cdot)$ can be used in place of $\mathbf{F}(\cdot, \cdot, \cdot)$. These three properties are important in enabling accurate modeling of the missile system.

Application to a Supersonic Missile

An example of a dynamic system represented by (1a)-(1c) is a controlled missile undergoing pitch motion, so that

$$\dot{\mathbf{x}}_a(t) = \begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\delta} \\ \dot{\delta}_z \end{bmatrix} = \begin{bmatrix} \frac{-L(\alpha,\delta)+mg\cos\gamma_f}{mV} + q \\ \frac{M(\alpha,\delta)}{I_{yy}} \\ \delta_z \\ \omega_a^2\delta - 2\zeta_a\omega_a\delta_z + \omega_a^2\delta_c \end{bmatrix}, \tag{7}$$

where, α is the angle of attack, γ_f is the flight path angle, q is the pitch rate, L is the lift force, M is the pitching moment, m is the mass, V is the missile speed, I_{yy} is the second mass moment of inertia on the pitch axis and g is gravity. The fin actuators are modeled with second order dynamics, where, δ_c is the commanded fin deflection input, δ is the achieved fin deflection, δ_z is the fin deflection rate, ζ_a is the damping ratio and ω_a is the undamped natural frequency of the actuator. The normal acceleration for the missile is

$$\eta = \frac{V(q-\dot{\alpha})}{\cos\alpha}, \tag{8}$$

and this is the system output that is to track $\mathbf{r}(t)$. It is required that the commanded input, $\mathbf{u}(t) = \delta_c$, be subject to a control law such that Property 1 is satisfied. To that end, a three-loop autopilot is utilized as the controller [**?**], although any other stabilizing controller such as a tuned proportional integral (PI) controller could be chosen. The dynamic equations for the controller are

$$\dot{\mathbf{x}}_b(t) = -q + K_A(-\eta + K_{DC}\eta_c), \tag{9}$$

where $\mathbf{r}(t) = \eta_c$ is the reference normal acceleration, K_A and K_{DC} are controller gains. The output of the controller (and input $\mathbf{u}(t)$ into the plant) is the commanded fin deflection

$$\delta_c = -K_R q + W_I\mathbf{x}_b(t), \tag{10}$$

where K_R and W_I are two more controller gains. Suitable selection of K_A, K_{DC}, K_R and W_I is required to satisfy Property 1 as described in Zarchan [**?**].

Now define

$$\epsilon := \frac{\dot{\delta}_{max} l_r}{V}, \tag{11}$$

where l_r represents the characteristic length of the fin and $\dot{\delta}_{max}$ is the maximum fin deflection rate. The CFD model in this example is described by the dimensionless Euler equations, that is

$$\epsilon\frac{\partial \mathbf{z}}{\partial t} = -\tilde{\nabla}\cdot\mathbf{f}, \tag{12}$$

where $\mathbf{z} = \left[\tilde{\rho}\;\tilde{\rho}\tilde{\mathbf{V}}\;\tilde{\rho}\tilde{E}\right]^T$, with $\tilde{\rho} := \frac{\rho}{\rho_r}$ representing the dimensionless density of the fluid, $\tilde{\mathbf{V}} := \frac{1}{V}\left[v_1\; v_2\; v_3\right]^T$ is the dimensionless fluid velocity vector and $\tilde{E} := \frac{E}{V^2}$ is the dimensionless specific total energy. The divergence operator $\tilde{\nabla} := \dot{\delta}_{max} l_r \nabla$ and the dimensionless convective flux is $\mathbf{f} := \left[\mathbf{f_1}\;\mathbf{f_2}\;\mathbf{f_3}\right]^T$ with

$$\mathbf{f}_i = \begin{pmatrix} \tilde{\rho}\tilde{v}_i \\ \tilde{\rho}\tilde{v}_i\tilde{\mathbf{V}} + \tilde{p}\boldsymbol{\delta}_i \\ \tilde{\rho}\tilde{v}_i(\tilde{E} + \frac{\tilde{p}}{\tilde{\rho}}) \end{pmatrix}, \tag{13}$$

where $\tilde{v}_i := \frac{v_i}{V}$, $\boldsymbol{\delta}_i = \left[\delta_{i1}\;\delta_{i2}\;\delta_{i3}\right]^T$ is a vector of Kronecker delta functions and the dimensionless pressure is given by

$$\tilde{p} = \frac{\rho(\gamma_s - 1)}{\rho_r V^2}\left(E - \frac{1}{2}||\mathbf{V}||_2^2\right) \tag{14}$$

and γ_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 utilizing CFD and rigid body dynamics models [? ?], time-scale separation is assumed. We show here through the time scale parameter, ϵ , that this approximation is valid. Assuming a nominal speed of Mach 2 for the flow ($V = 680\,\mathrm{ms}^{-1}$) and the maximum actuator fin rate is $\dot{\delta}_{max} = 100\,\mathrm{rads}^{-1}$ and $l_r = 0.33\,\mathrm{m}$, then $\epsilon = 0.05$. This indicates that $O(\epsilon)$ is bounded on the order of 10^{-2} and therefore the approximation from Property 2 is good.

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 discretization, such as the finite volume method, to (12) is necessary. This results in a coupled system of ODEs, where for each discretised cell i

$$\frac{d\mathbf{z}_{h,i}}{dt} = -\frac{1}{V_i}\int_{S_i}\mathbf{f}.\mathbf{n}dS, \tag{15}$$

where $\mathbf{z}_{h,i}(t)$ is the volume averaged solution of $\mathbf{z}(t)$ for each cell, V_i is the cell volume, $\mathbf{n}$ 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 (15) to the steady state solution.

In order to populate the maps of the aerodynamic forces, for each missile shape design candidate, $\boldsymbol{\theta}_f$, steady state CFD simulations for a series of missile poses are evaluated. That is,

$$\hat{\mathbf{F}}(\alpha, \delta, \boldsymbol{\theta}_f) = \left[D(\alpha, \delta, \boldsymbol{\theta}_f)\ L(\alpha, \delta, \boldsymbol{\theta}_f)\ M_Y(\alpha, \delta, \boldsymbol{\theta}_f)\right]^T . \tag{16}$$

As stated in Property 3, the chosen interpolation scheme necessarily effects 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 α and δ was sufficient to keep the approximation error, ε, small.

III. Framework Overview

The optimization framework is shown in Figure 3. The framework consists of an inner and outer optimization loop. This two-level optimization 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 minimizing the overall optimization time.

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

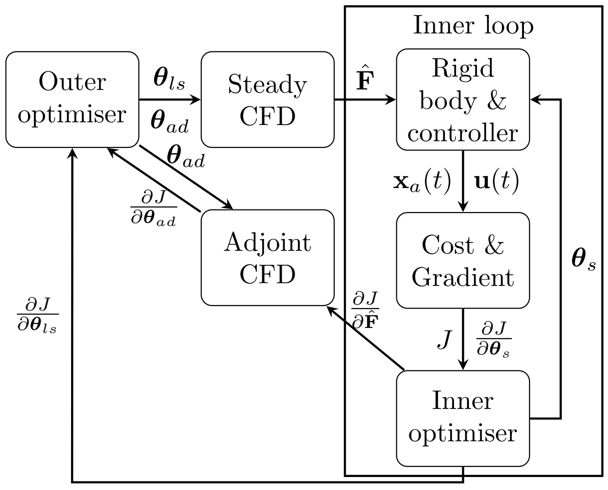
$$\begin{aligned} \boldsymbol{\theta}_s^*(\boldsymbol{\theta}_f) = \underset{\boldsymbol{\theta}_s}{\arg\min} \quad & J(\boldsymbol{\theta}_s, \boldsymbol{\theta}_f) \\ \text{subject to} \quad & g_{in}(\boldsymbol{\theta}_s) \geq 0 \quad \text{and} \\ & h_{in}(\boldsymbol{\theta}_s) = 0 \end{aligned} \tag{17}$$

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 (2a) and (2b). The outer optimization problem considers the design parameters associated with the fast states $\mathbf{z}(t)$ and with the restriction $\boldsymbol{\theta}_s = \boldsymbol{\theta}_s^*(\boldsymbol{\theta}_f)$. The optimal design variables of the outer loop are given by,

$$\begin{aligned} \boldsymbol{\theta}_f^* = \underset{\boldsymbol{\theta}_f}{\arg\min} \quad & J(\boldsymbol{\theta}_s^*(\boldsymbol{\theta}_f), \boldsymbol{\theta}_f) \\ \text{subject to} \quad & \boldsymbol{\theta}_{f,L} \geq \boldsymbol{\theta}_f \geq \boldsymbol{\theta}_{f,U} \\ & h_{out}(\boldsymbol{\theta}_f) = 0 \end{aligned} \tag{18}$$

where J is a cost function, $\boldsymbol{\theta}_{f,L}$ and $\boldsymbol{\theta}_{f,U}$ 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 (2c). Note that the framework can only handle inequality bound constraints in the outer loop.

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



Inner loop
Outer optimiser
$\boldsymbol{\theta}_{ls}$
$\boldsymbol{\theta}_{ad}$
Steady CFD
$\hat{\mathbf{F}}$
Rigid body & controller
$\boldsymbol{\theta}_{ad}$
$\frac{\partial J}{\partial \boldsymbol{\theta}_{ad}}$
$\mathbf{x}_a(t)$
$\mathbf{u}(t)$
Adjoint CFD
Cost & Gradient
$\boldsymbol{\theta}_s$
$\frac{\partial J}{\partial \boldsymbol{\theta}_{ls}}$
$\frac{\partial J}{\partial \hat{\mathbf{F}}}$
J
$\frac{\partial J}{\partial \boldsymbol{\theta}_s}$
Inner optimiser

Fig. 3: Optimization framework for the missile design problem

A. 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 optimizer that affect the value of the cost. A gradient-based optimization 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 $\boldsymbol{\theta}_f$ directly affect the

CFD simulations and the controller parameters $\boldsymbol{\theta}_s$ do not. Partitioning of the design variables in the optimization framework should result in a reduction in the overall simulation time by avoiding extra CFD evaluations (e.g. during a line search).

Application to a Supersonic 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, that is,

$$J = \int_0^T e(t)^2 dt + w \int_0^T D(t)^2 dt, \tag{19}$$

where $e(t)$ is the missile's normal acceleration tracking error; $D(t)$ is the instantaneous drag and w is a weighting constant that is used to tune to relative weighting on each component of the cost function. 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 maneuverability and range.

This study will consider two types of variables. First are shape design variables, represented by deformations away from a base design. Figure 4 shows the geometry of the tail fin missile, and Figures 5 and 6 show the CFD meshes and the associated deformation vectors. Each shape deformation variable locally deforms the shape via a free-form deformation (FFD) algorithm [**?**]. The second type of design variables are the control parameters. The three-loop autopilot has been

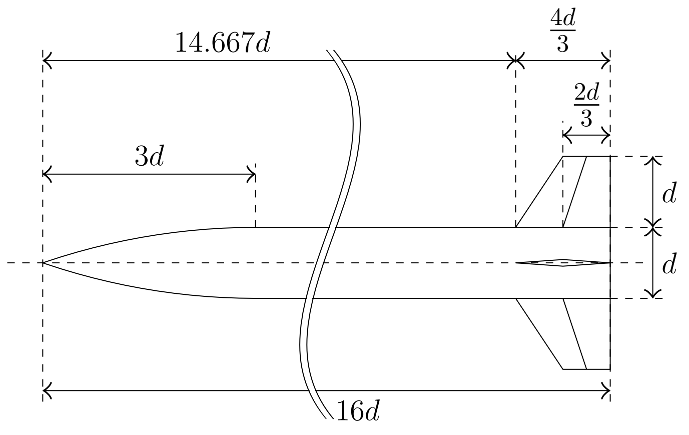


Fig. 4: Tail-fin controlled missile geometry

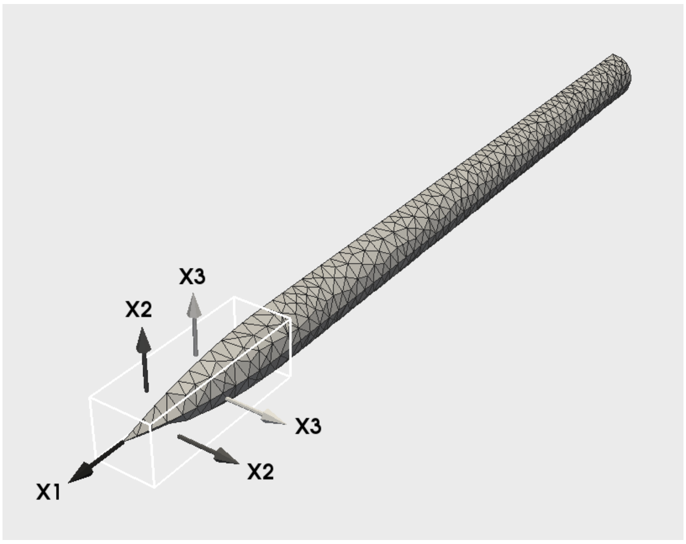


Fig. 5: Body mesh and shape deformation design variables

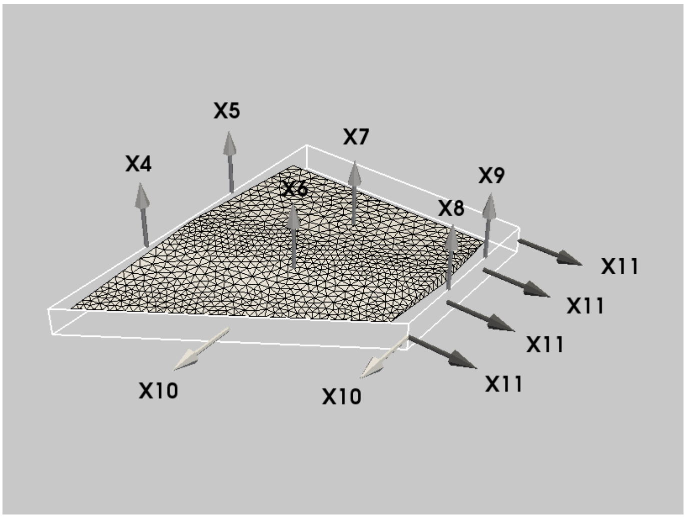


Fig. 6: Body mesh and shape deformation design variables

chosen as the controller. This autopilot has four gain variables, but rather than tuning each of these gains, Zarchan's tuning method [?] is utilized. Three controller characteristic parameters are specified in the tuning method, namely the rise time τ_c, damping ratio ζ_c and open-loop crossover frequency ω_{cr}. Let $\boldsymbol{\theta}_{ls} := [X_1\ X_2\ X_3\ X_{10}\ X_{11}]^T$ and $\boldsymbol{\theta}_{ad} := [X_4\ X_5\ \cdots\ X_9]$ be the vectors of shape deformation variables, and let $\boldsymbol{\theta}_s := [\tau_c\ \zeta_c\ \omega_{cr}]^T$ be the vector of characteristic parameters for the control tuning method.

Figure 7 shows the cost function with respect to the nose tip geometry deformation. Figure 8

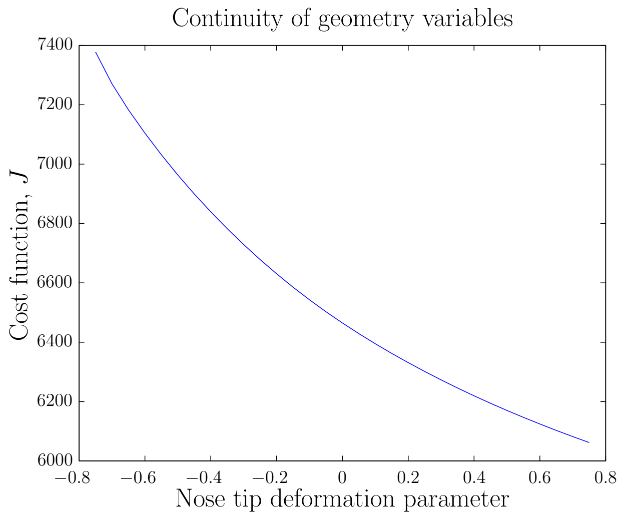


Fig. 7: Continuity of cost function with respect to nose tip geometry design variable

shows the cost function with respect to two of the controller parameters τ_c and ζ_c with ω_{cr} held constant. Figure 7 shows that the cost function is continuously differentiable with respect to the

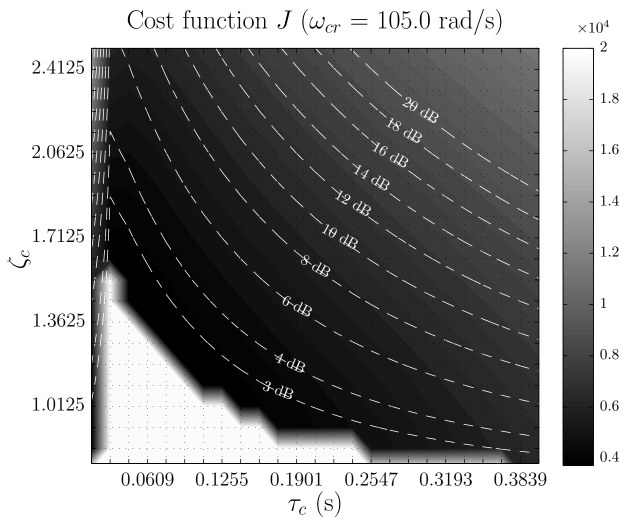


Fig. 8: Continuity of cost function with respect to controller parameters, τ_c and ζ_c

geometry design variables. It can also be seen from Figure 8 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.

B. Inner Optimization

The rigid body dynamics model and controller model, respectively (2a) and (2b), the design variables $\boldsymbol{\theta}_s$ and cost function J can be cast as a constrained optimization problem of the form (17). An optimizer 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 optimize only the controller design variables, that is, parameters in the controller (2b), the inner loop is also able to optimize other plant parameters from the slow dynamic model (2a).

Application to a Supersonic Missile

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

In the example presented here, MATLAB's[1] sequential quadratic program (SQP) solver is utilized 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. With this tuning method, there are no stability guarantees when optimizing the controller parameters and this can lead to critically stable or unstable closed-loop behavior. The cost function surface is shown in Figure 8 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 optimization. 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 on the inner optimization are then the dynamic equations (7) and (9) and the bounds

$$\begin{bmatrix} -\boldsymbol{\theta}_s + \boldsymbol{\theta}_{s,U} \\ \boldsymbol{\theta}_s - \boldsymbol{\theta}_{s,L} \\ G_m - G_m^* \end{bmatrix} \geq 0, \tag{20}$$

where $\boldsymbol{\theta}_{s,L}$, $\boldsymbol{\theta}_{s,U}$ are lower and upper bounds on the design variables, G_m is the gain margin and

G_m^* is the gain margin lower bound.

C. Outer Optimization

The system model (2), the design variables $\boldsymbol{\theta}_f$ and cost function J can be cast as a constrained optimization problem of the form (18). An optimizer 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 optimizer 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 modeled as additive noise, that is,

$$\nabla \hat{J}(\boldsymbol{\theta}, \gamma) = \nabla J(\boldsymbol{\theta}) + \Phi(\gamma) \tag{21}$$

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

Property 4. *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 γ. Thus, the parameter γ can be used to control the fidelity of the computed gradient. Let successive design iterations be found by an update law of the gradient based optimizer

$$\boldsymbol{\theta}_{k+1} = f(\nabla \hat{J}(\boldsymbol{\theta}_k, \gamma_k)), \tag{22}$$

where $\boldsymbol{\theta}_k$ is the design variables and γ_k is the error parameter at iteration k of the optimizer.

Property 5. *$\Phi(\gamma_k) \to 0$ as $k \to \infty$, so that the iterates of the design variables, $\boldsymbol{\theta}_k \to \boldsymbol{\theta}^*$, where $\boldsymbol{\theta}^*$ is an optimal solution.*

If the gradient of the cost function satisfies Property 4 and the optimizer is set up with an error control mechanism that satisfies Property 5, then the noise of the gradient will approach zero with

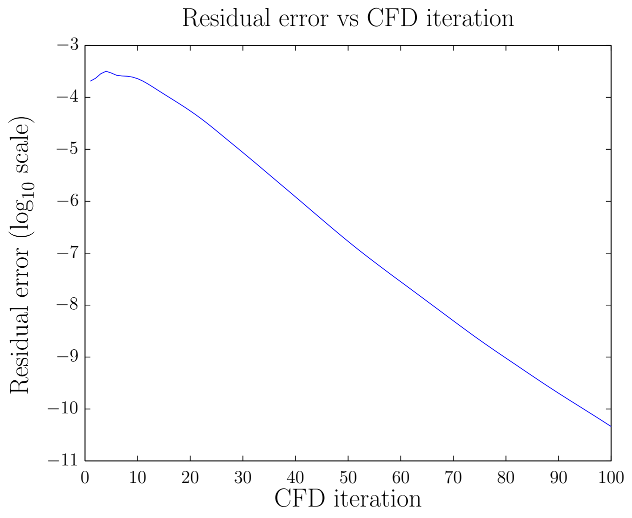


Fig. 9: Residual error of CFD solver

increasing iterations of the optimizer and the optimizer will converge to the vicinity of the optimal solution.

Application to a Supersonic Missile

The missile example utilizes 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 9 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 4 is modeled by the residual error of the CFD solver. The number of CFD iterations then corresponds to the parameter γ. In the context of the optimizer, as the optimization progresses the CFD iteration requires to be run to γ 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 discretization approaches the limit. This is known as the *consistency* property of the solver and with Property 4 only the closeness of the partially converged numerical solution to the ODE solution (and not the PDE solution) can be guaranteed.

A candidate optimizer that satisfies Property 5 is the implicit filtering algorithm [**?**]. Implicit filtering is a variation of the bound-constrained coordinate search (or direct search) optimization 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 paper, the implicit filtering algorithm was modified to neglect sampling in $\boldsymbol{\theta}_s$. Gradient information is found using the adjoint method instead. The modified algorithm is given in the Appendix.

D. 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 optimizer. 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, which represents the state of the art. However, this is not always possible, as shown in Figure 1 for sharp edged geometry variables. In this paper, two candidate gradient methods are utilized, namely the continuous adjoint method and least squares finite difference (LSFD). Further information regarding their derivation can be found in the Appendix.

Consider the surface of the aerodynamic body shown in Figure 10. Let S represent the baseline surface and S' be the deformation of the surface at coordinate s. The surface and its deformation

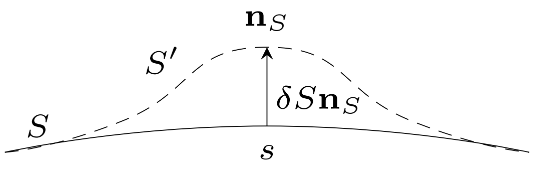


Fig. 10: Surface properties

is considered to be smooth if the following property is satisfied.

Property 6. *A normal vector $\mathbf{n}_S(s)$ is well defined at s such that*

$$S' = \{s + \delta S(s)\mathbf{n}_S(s), s \in S\} \tag{23}$$

represents the deformed smooth surface.

If Property 6 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.

Application to a Supersonic Missile

Table 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

Variable	Adjoint	LSFD	Central FD	Smooth
X_5	-0.2519	-0.2903	-0.2909	Yes
X_7	-0.2296	-0.2598	-0.2600	Yes
X_9	-0.1621	-0.1868	-0.1875	Yes
X_{10}	-0.01852	0.08832	0.08803	No
X_{11}	0.1048	0.2141	0.2123	No
Time (min)	3.79	19.69	101.1	
No. of simulations	2	24	24	

Table 1: Comparison of gradients for tail fin geometry calculated using various methods

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 γ), 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 utilized for design variables $\boldsymbol{\theta}_{ad}$ defined over smooth surfaces (i.e. tail fin profile), and LSFD is utilized for design variables $\boldsymbol{\theta}_{ls}$ defined over sharp edges (i.e. nose tip and tail fin planform).

E. Convergence

Properties 1–3 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 optimization problem. It has also been shown that the cost function is differentiable which is a necessary condition for the deployment of gradient based optimizers. Properties 4 and 5 relate to the tolerance of the optimizer to inaccurate gradients. Lastly, Property 6 dictates if the adjoint method is suitable for the gradient calculation. If all above hold, then the optimization 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 optimizer and the relationship between the iterate k and parameter γ. If $\Phi(\gamma)$ is small or if it decays quickly with increasing γ, 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 optimizer will approach the minimum more quickly. In particular, the optimizer may converge to the solution without necessarily requiring γ to be large, resulting in faster computation of the gradient and also the overall optimization 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.

IV. Results

Two examples[2] will be presented in this section. The first is a tail fin planform optimization 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 optimizes the missile tail fin's planform and profile and also the nose geometry. The adjoint method is utilized for the gradient calculation for a subset of design variables. The computational advantage of utilizing gradients derived from the adjoint method in this framework is also highlighted. The missile model is initialized 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 (19) 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 normalized according to

$$\bar{J} = \frac{J}{J_0} \times 100(\%) \tag{24}$$

where J_0 is the cost of the initial design.

A. Tail fin planform optimization

In this example, the design variables are the tail-fin leading edge and wing span (X_{10} and X_{11} in Figure 6). 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 $\delta = 0°, 2.5°, 5°, 7.5°, 10°$ are conducted. The control parameters are also optimized by the inner loop. Three optimization runs are conducted. In the first run, the residual error is controlled by $\gamma = k$ and increases the CFD iterations over successive implicit filtering optimizer 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 solver (which utilizes finite differencing) has been included for comparison. In the example, all three runs converged to the same optimal design.

Table 2 shows the optimized control parameters for the initial and final shape, that is, the control parameters after the inner-loop optimization. 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.

Shape	τ_c	ζ_c	ω_{cr}
Original	0.1165	1.7664	105
Optimized	0.1158	1.7626	105

Table 2: Optimal control parameters for original and optimized tail fin planform shape

The original and optimized tail fin shape are shown in Figure 11. It can be seen that the optimal planform shape favors 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 optimization was only able to minimize the drag component of the cost function.

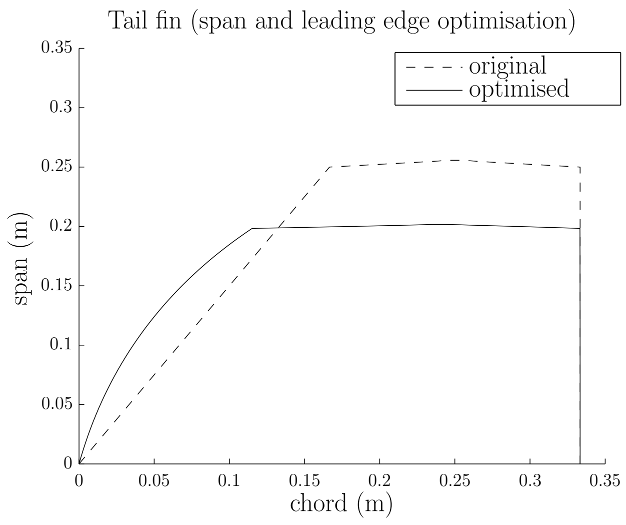


Fig. 11: Original and optimized shape of tail fin

Figure 12 shows the reduction in the normalized 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 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 optimization.

B. Nose, tail-fin planform and profile optimization

In this example, the nose and tail fin geometries are optimized. (X_1 to X_3 in Figure 5 and X_4 to X_{11} in Figure 6). CFD simulations are required for both the missile body and tail fin. For the nose, CFD simulations are conducted at $\alpha = 0°, 4°, 8°, 12°, 16°$, and for the tail fin, they are conducted at $\delta = 0°, 4°, 8°, 12°$. The gradients for the nose and tail fin planform design variables

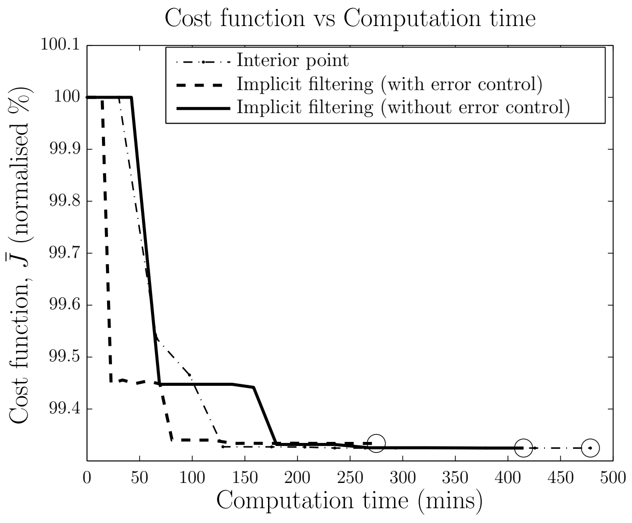


Fig. 12: Implicit filtering comparison with and without error control

are calculated using least squares finite differencing, while the gradients for the tail fin profile design variables are calculated using the adjoint method.

Two optimization cases are presented. Firstly, only the nose and the tail fin planform are optimized, that is, the design variables utilizing the adjoint method are omitted. Secondly, the nose, tail fin planform and profile are optimized. The result of the optimized nose and planform shape are the same for both cases and is shown in Figure 13. Note that the body of the missile, i.e. between $1.4m$ and $3.6m$, 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 optimized nose is sharper and that the shape of the optimized tail fin is the same as the optimized result in the previous example. In order to compare the profile shape, cross-sections through the tail fin are shown in Figures 14 and 15. The cross-sections are uniformly spaced and are labeled with increasing numbers to indicate the distance away from the missile body. Figure 14 shows the profile shape where only the planform has been optimized. Figure 15 shows the tail-fin where the planform and profile have been optimized. It can be seen that the height of the profile has been optimized 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 optimizer, and this suggests that there is a trade-off between maneuverability and aerodynamic drag for the profile design variables.

Table 3 shows that the controller parameters have changed from the original, and unlike in the

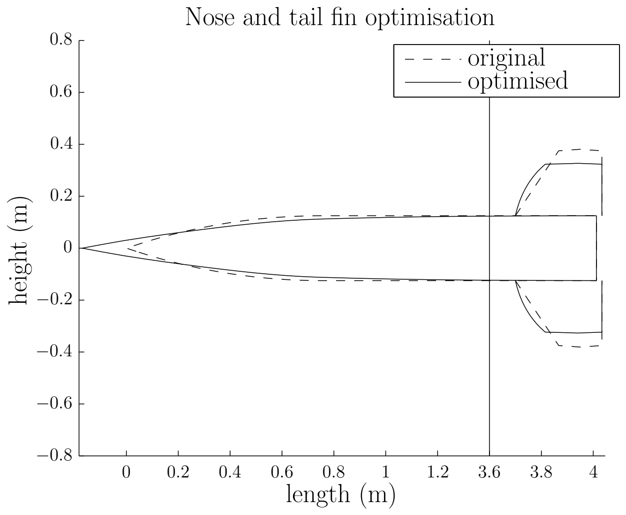


Fig. 13: Shape of original and optimized missile (body of missile omitted)

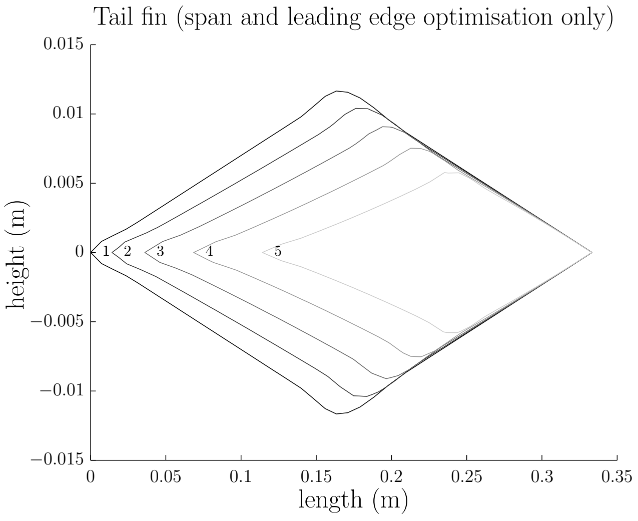


Fig. 14: Cross sections of planform optimized tail fin shape (i.e. the chord shown here has not changed). The cross sections are labeled 1-5 as they move uniformly away from the missile body.

previous example, this means that the geometry variables and controller parameters are coupled.

The normalized cost function versus the total optimization time are shown in Figure 16. By optimizing 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 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,

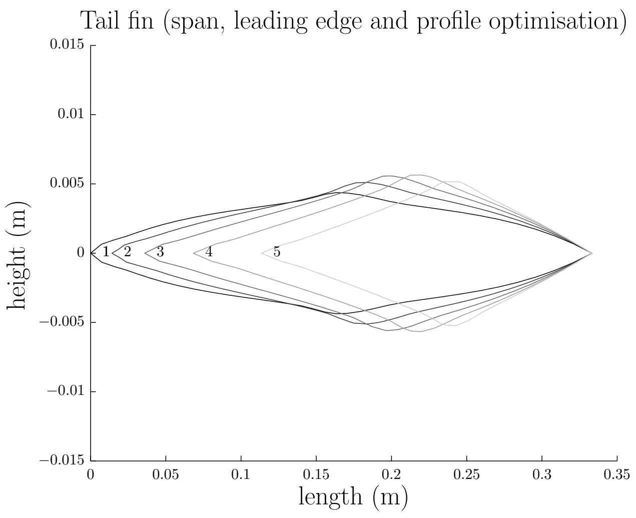


Fig. 15: Cross sections of planform and profile optimized tail fin shape. The cross sections are labeled 1-5 as they move uniformly away from the missile body.

Shape	τ_c	ζ_c	ω_{cr}
Original	0.0804	1.4759	105
Nose & planform	0.0854	1.5176	105
Nose, planform & profile	0.0850	1.5136	105

Table 3: Optimal control parameters for original and optimized nose & fin shapes

the number of design variables utilizing the adjoint method is more than twice that of the other geometry variables, yet the optimization time is not significantly increased. It can be seen that it is better to utilize the adjoint method where possible, that is, for design variables where Property 6 is satisfied. Indeed, one of the benefits of the optimization 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 optimizer means that this framework could lead to unexpected improvements in the optimized design.

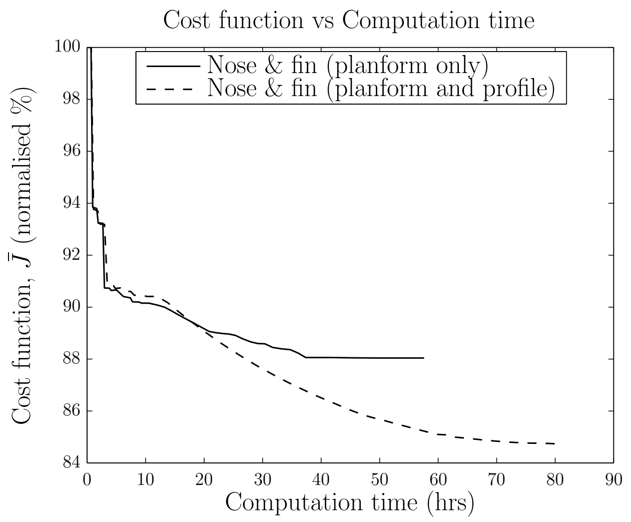


Fig. 16: Total optimization time of nose and fin

V. Conclusions

A framework for the optimization of a controlled aerodynamic system has been developed and it has been shown to successfully optimize a missile's shape and control design parameters. Two of the system's properties are exploited in order to perform efficient dynamic MDO. Firstly, the time-scales of the system are identified and a nested co-design optimization architecture is deployed. Secondly, surrogate models are utilized 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 optimization 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 specialized simulators (including closed source simulators).

It has been shown that controlling the error of the gradient computed by the optimizer leads to a speed up in the optimization time. This is particularly useful when utilizing computationally expensive models such as CFD simulations. Examples provided in the paper show that the framework is particularly useful in the optimization 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 equations can be implemented in the CFD solver, such as the Reynolds-averaged Navier-Stokes (RANS) equations with an associated turbulence model. Convexity is a property of the optimization metric and if the optimization 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 electromagnetic system where the shapes of the rotors and stators are optimized 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 optimization framework.

Appendix

The gradient methods and the implicit filtering optimization algorithm presented here are not new methods, but are reproduced here for the reader's convenience.

A. 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 $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_K\}$ 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 $\boldsymbol{\theta}_{ls}$ in the design space and points on the stencil $\mathbf{S}(\boldsymbol{\theta}_{ls}, h, \mathbf{V}) = \{\mathbf{Z} | \mathbf{Z} = \boldsymbol{\theta}_{ls} + h\mathbf{v}_k, 1 \leq k \leq K\}$, where h is the finite difference step size. This gives the gradient

$$\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\theta}_{ls}} = \frac{1}{h}\delta J(\boldsymbol{\theta}_{ls}, h, \mathbf{V})\mathbf{V}^{\dagger}, \tag{25}$$

where

$$\delta J(\boldsymbol{\theta}_{ls}, h, \mathbf{V}) = \begin{bmatrix} J(\boldsymbol{\theta}_{ls} + h\mathbf{v}_1) - J(\boldsymbol{\theta}_{ls}) \\ J(\boldsymbol{\theta}_{ls} + h\mathbf{v}_2) - J(\boldsymbol{\theta}_{ls}) \\ \vdots \\ J(\boldsymbol{\theta}_{ls} + h\mathbf{v}_K) - J(\boldsymbol{\theta}_{ls}) \end{bmatrix} \tag{26}$$

and $\mathbf{V}^{\dagger} := \mathbf{V}^T(\mathbf{V}\mathbf{V}^T)^{-1}$ is the pseudoinverse of $\mathbf{V}$. Note that a central differencing stencil requires $2N + 1$ simulations.

B. Continuous adjoint method [? ? ?]

The continuous adjoint method is typically applied to calculate the gradient of static steady-flow aerodynamic costs such as the lift or drag. In order to calculate costs of dynamic systems, the chain rule is utilized, where

$$\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\theta}_{ad}} = \sum_k \frac{\partial J}{\partial \hat{\mathbf{F}}(\mathbf{x}_{a,k}(t), \mathbf{u}_k(t))} \frac{\mathrm{d}\hat{\mathbf{F}}(\mathbf{x}_{a,k}(t), \mathbf{u}_k(t))}{\mathrm{d}\boldsymbol{\theta}_{ad}} \tag{27}$$

and where $\{\mathbf{x}_{a,k}(t), \mathbf{u}_k(t)\}$ are the k steady-state points over which the interpolated map $\hat{\mathbf{F}}(\cdot,\cdot)$ has been evaluated. On the right hand side of (27), the first term is the sensitivity of the cost with respect to the map and the second term is the sensitivity of the map with respect to the design variables. A straightforward, albeit naive, calculation of the second term in (27) is to calculate the gradient via the adjoint method for each value in the map $\hat{\mathbf{F}}(\cdot,\cdot)$ over the k points. In the missile example given in the paper, $\hat{\mathbf{F}}(\cdot,\cdot)$ has three components (see Equation (16)) and so will require a total of $3k$ adjoint CFD simulations. The total number of adjoint CFD simulations can be reduced to k simulations. First note that the lift, drag and pitching moment are a class of functions that can be written as

$$\hat{\mathbf{F}} = \int_S \mathbf{d} \cdot (p\mathbf{n}_S)\, ds, \tag{28}$$

where $\mathbf{d}$ is a force projection vector, p is the pressure and $\mathbf{n}_S$ is the local normal vector on the surface. Set

$$\mathbf{d} = \frac{\partial J}{\partial \hat{\mathbf{F}}(\mathbf{x}_{a,k}(t), \mathbf{u}_k(t))}\mathbf{G}, \tag{29}$$

where the derivatives of J with respect to $\hat{\mathbf{F}}(\mathbf{x}_{a,k}(t),, \mathbf{u}_k(t))$ are found using finite differences[3] and each row of $\mathbf{G}$ in equation (29) corresponds to the forces and moments in the map $\hat{\mathbf{F}}(\cdot,\cdot)$. For the tail-fin, where the pose only consists of the fin deflection δ, define

$$\mathbf{G} := \frac{1}{C_\infty}\begin{bmatrix} \cos\delta & \sin\delta & 0 \\ -\sin\delta & \cos\delta & 0 \\ 0 & 0 & \frac{-(x-x_0)}{L_{ref}} \end{bmatrix}, \tag{30}$$

where $C_\infty = \frac{1}{2}V^2\rho_\infty^2 A_z$. ρ_∞ is freestream density, L_{ref} is the reference length and A_z is the reference area. Equations (29) and (30) are implemented in (27) so that only k adjoint simulations are required.

Now, the derivation of the adjoint method is described. The interpolated map $\hat{\mathbf{F}}(\cdot,\cdot)$ is a function of the fast states, $\mathbf{z}$, and design variables, $\boldsymbol{\theta}_{ad}$. By calculus of variations, a change in $\boldsymbol{\theta}_{ad}$ results in a change in the map,

$$\delta\hat{\mathbf{F}} = \frac{\partial\hat{\mathbf{F}}}{\partial\mathbf{z}}\delta\mathbf{z} + \frac{\partial\hat{\mathbf{F}}}{\partial\boldsymbol{\theta}_{ad}}\delta\boldsymbol{\theta}_{ad}. \tag{31}$$

It can be expensive to compute variations in the fast states, $\delta\mathbf{z}$, that is, each variation will require an additional CFD simulation. The aim of the adjoint approach is to eliminate this term in (31). Suppose that partial differential equations of the form (2c) is introduced as an equality constraint

$$R\left(\mathbf{z},\boldsymbol{\theta}_{ad}\right) = 0. \tag{32}$$

The variation in (32) is

$$\delta R = \left[\frac{\partial R}{\partial\mathbf{z}}\right]\delta\mathbf{z} + \left[\frac{\partial R}{\partial\boldsymbol{\theta}_{ad}}\right]\delta\boldsymbol{\theta}_{ad} = 0. \tag{33}$$

Equation (31) can be combined with (33) via a Lagrange Multiplier ψ, which gives

$$\begin{aligned}\delta\hat{\mathbf{F}} =& \frac{\partial\hat{\mathbf{F}}}{\partial\mathbf{z}}\delta\mathbf{z} + \frac{\partial\hat{\mathbf{F}}}{\partial\boldsymbol{\theta}_{ad}}\delta\boldsymbol{\theta}_{ad} - \psi\left(\left[\frac{\partial R}{\partial\mathbf{z}}\right]\delta\mathbf{z} + \left[\frac{\partial R}{\partial\boldsymbol{\theta}_{ad}}\right]\delta\boldsymbol{\theta}_{ad}\right)\\ =& \left\{\frac{\partial\hat{\mathbf{F}}}{\partial\mathbf{z}} - \psi\left[\frac{\partial R}{\partial\mathbf{z}}\right]\right\}\delta\mathbf{z} + \left\{\frac{\partial\hat{\mathbf{F}}}{\partial\boldsymbol{\theta}_{ad}} - \psi\left[\frac{\partial R}{\partial\boldsymbol{\theta}_{ad}}\right]\right\}\delta\boldsymbol{\theta}_{ad}.\end{aligned} \tag{34}$$

Choose ψ such that

$$\left[\frac{\partial R}{\partial\mathbf{z}}\right]\psi = \frac{\partial\hat{\mathbf{F}}}{\partial\mathbf{z}}. \tag{35}$$

Equation (35) is called the adjoint equation. With an appropriately defined flow field and boundary conditions, this adjoint PDE can be implemented and solved using components of the same code used to solve (2c).

Substituting (35) into (34) eliminates the term $\delta\mathbf{z}$. The change in the cost functional is

$$\delta\hat{\mathbf{F}} = \frac{\mathrm{d}\hat{\mathbf{F}}}{\mathrm{d}\boldsymbol{\theta}_{ad}}\delta\boldsymbol{\theta}_{ad}, \tag{36}$$

where the gradient is given by

$$\frac{\mathrm{d}\hat{\mathbf{F}}}{\mathrm{d}\boldsymbol{\theta}_{ad}} = \frac{\partial\hat{\mathbf{F}}}{\partial\boldsymbol{\theta}_{ad}} - \psi\left[\frac{\partial R}{\partial\boldsymbol{\theta}_{ad}}\right]. \tag{37}$$

Implicit Filtering [?]

Algorithm 1.

1. Note the upper and lower bounds of all design variables and scale them such that $\Omega = \{\boldsymbol{\theta}_f \in R^N | 0 \leq \theta_j \leq 1\}$ for $j = \{1, 2, \ldots, N\}$. Let the initial design be scaled to $\boldsymbol{\theta}_{f,0} \in \Omega$. Scale the cost function $J(\cdot)$ by $\frac{1}{1.2J(\boldsymbol{\theta}_{f,0})}$.

2. Let Q be the maximum number of iterations of the algorithm. Let $\boldsymbol{\Gamma} = [\gamma_1, \gamma_2, ..., \gamma_Q]$ be an error vector to control the error of the computed cost function. It is required from Property 4 that the selection of successive γ values be increasing so that the noise level decrease as the algorithm proceeds.

3. Initialize the current design point $\boldsymbol{\theta}_c = \boldsymbol{\theta}_{f,0}$, the step size $h = 0.5$, error control $\gamma = \gamma_1$, the Hessian $\mathbf{H} = \mathbf{I}$ 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 $J(\boldsymbol{\theta}_c, \gamma)$.

   (b) Compute the gradient $\nabla J(\boldsymbol{\theta}_c, \gamma)$. The gradient can be calculated, for example, by using either least squares finite differencing (25) or the adjoint method (27). If least squares finite differencing is used, then store the value of the cost function at each stencil point as well.

   (c) If $||\boldsymbol{\theta}_c - \mathcal{P}(\boldsymbol{\theta}_c - \nabla J(\boldsymbol{\theta}_c, \gamma))|| < \tau h$ go to step (5f), where $\mathcal{P}(\cdot)$ is the projection operator.

   (d) Set the line search design point $\boldsymbol{\theta}_+ = \boldsymbol{\theta}_c$.

   (e) While $||\boldsymbol{\theta}_+ - \mathcal{P}(\boldsymbol{\theta}_+ - \nabla J(\boldsymbol{\theta}_+, \gamma))|| \geq \tau h$:

      i. Update the model Hessian $\mathbf{H}$ using the BFGS quasi-Newton method [?].

      ii. Determine the search direction,

$$\mathbf{s} = -\mathbf{H}^{-1}\nabla J(\boldsymbol{\theta}_+)$$

iii. Perform a backtracking line search

$$\boldsymbol{\theta}_+ = \mathcal{P}(\boldsymbol{\theta}_+ - \lambda \mathbf{s}), \tag{38}$$

where λ is initialized as 1 and reduced by a factor of β 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, $\boldsymbol{\theta}_{f,min}$ from the stencil (if least squares finite differencing is used) or from the successful line search and set $\boldsymbol{\theta}_c = \boldsymbol{\theta}_{f,min}$, $h = h/2$, $\gamma = \gamma_{k+1}$ and $k = k + 1$.

Acknowledgments

The authors would like to thank the generous support of the Australian Research Council under the Linkage Projects funding scheme (LP110200025). Support of BAE Systems Australia and the Defence Sciences Institute is also acknowledged.

References

[1] http://www.mathworks.com

[2] Both examples were executed on a Intel Core$^{\text{TM}}$ i7-2600 3.4GHz desktop PC with 8GB of RAM

[3] Comparatively speaking, using finite differences in the rigid body dynamic model does not significantly increase the computational burden as compared to the computation required for CFD simulations.