Optimization Framework for Codesign of Controlled Aerodynamic Systems

Kuan Waey Lee,* William Moase,† Andrew Ooi,‡ and Chris Manzie‡

University of Melbourne, Melbourne, Victoria 3010, Australia

and

Eric C. Kerrigan‡

Imperial College London, London, England SW7 2AZ, United Kingdom

DOI: 10.2514/1.J054711

Optimization studies of dynamic systems using high-fidelity numerical models necessitate a tradeoff 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 that allows for the partitioning of controller and plant architecture is implemented. 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. It is shown that combining these methods 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.

Nomenclature

\[ D(t) \] = instantaneous drag, N

\[ E(t) \] = dimensionless specific total energy

\[ e(t) \] = normal acceleration tracking error, \( m \cdot s^{-2} \)

\[ F \] = aerodynamic force vector, N

\[ g \] = acceleration due to gravity, \( m \cdot s^{-2} \)

\[ I_{yy} \] = second mass moment of inertia on pitch axis, \( kg \cdot m^2 \)

\[ J \] = cost function

\[ K_A \] = acceleration loop controller gain

\[ K_{dc} \] = static controller gain

\[ L \] = lift force, N

\[ l_r \] = characteristic length of tail fin, m

\[ M \] = pitching moment, N \cdot m

\[ m \] = mass, kg

\[ N \] = number of design variables

\[ n \] = normal vector of each face

\[ n_S \] = normal surface vector

\[ q \] = pitch rate, rad/s

\[ r(t) \] = reference trajectory

\[ S \] = baseline surface

\[ S_l \] = cell surface area of i-th cell, m²

\[ S' \] = perturbed surface

\[ s \] = surface coordinate

\[ u(t) \] = controller inputs

\[ V \] = missile speed, \( m \cdot s^{-1} \)

\[ V_i \] = cell volume of i-th cell, m³

\[ \bar{V} \] = dimensionless fluid velocity vector

\[ W_f \] = integral loop controller gain

\[ W \] = cost function weighting constant

\[ X \] = shape deformation variable

\[ x_s(t) \] = slow system states

\[ x_f(t) \] = controller states

\[ z(t) \] = fast system states

\[ z_{\text{av}}(t) \] = volume averaged solution of \( z(t) \)

\[ \bar{z}(t) \] = quasi-steady solution of \( z(t) \)

\[ \alpha \] = angle of attack, rad

\[ \gamma \] = flight-path angle, rad

\[ \gamma_s \] = ratio of specific heats

\[ \delta \] = achieved fin deflection, rad

\[ \delta_c \] = commanded fin deflection, rad

\[ \delta f \] = fin deflection rate, \( rad \cdot s^{-1} \)

\[ \delta_{\text{max}} \] = maximum fin deflection rate, \( rad \cdot s^{-1} \)

\[ \epsilon \] = time-scale constant

\[ \varepsilon \] = interpolation mapping error

\[ \zeta_a \] = actuator damping ratio

\[ \zeta_c \] = controller damping ratio

\[ \eta \] = normal acceleration, \( m \cdot s^{-2} \)

\[ \eta_t \] = commanded normal acceleration, \( m \cdot s^{-2} \)

\[ \theta_a \] = design variables using adjoint method

\[ \theta_f \] = design variables of fast states

\[ \theta_{f,U} \] = upper bound for design variables of fast states

\[ \theta_{U} \] = design variables using least-squares finite difference

\[ \theta_{U} \] = design variables of slow states

\[ \theta_{U,L} \] = upper bound for design variables of slow states

\[ \rho \] = dimensionless density of fluid

\[ \Phi \] = additive noise

\[ \omega \] = controller open-loop crossover frequency, Hz

\[ \omega_t \] = controller open-loop crossover frequency, Hz

\[ \Omega_j \] = computed gradient

\[ \Omega J \] = true gradient

Received 27 August 2015; revision received 29 February 2016; accepted for publication 2 March 2016; published online 6 June 2016. Copyright © 2016 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved. Copies of this paper may be made for personal and internal use, on condition that the copier pay the per-copy fee to the Copyright Clearance Center (CCC). All requests for copying and permission to reprint should be submitted to CCC at wwwopyright.com; employ the ISSN 0001-1452 (print) or 1533-385X (online) to initiate your request.

*Student, Department of Mechanical Engineering.

†Postdoctorate, Department of Mechanical Engineering.

‡Professor, Department of Mechanical Engineering.

§Reader in Control Engineering and Optimization, Department of Electrical and Electronic Engineering.

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 [1] 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, use 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 [2]. Even then, the computational 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 [3] 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 methods, 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 [4]. The adjoint method was first applied to single static operating conditions and has since been extended to multidisciplinary and dynamic operating conditions [5].

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 [7] 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 the 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 tradeoff between execution speed and memory consumption [8] and still requires some level of user interaction for the generated code to be computationally efficient [9]. 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 codesign 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 used in a global optimization framework in order to speed up convergence to the optimal solution to a level comparable to gradient based methods [10]. However, global techniques do not scale well with the increase in design variables and cannot be used indiscriminately [11,12].

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. To provide the computational speedup, 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 the 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 Fig. 2. The design problem is posed as a nested optimization problem [13]. The design parameters are the shape of the missile \( \theta_s \) and controller tuning parameters \( \theta_c \). Partitioning of the design variables will be made.

\[
\begin{align*}
\theta_s & \rightarrow \text{Steady} \quad \text{CFD} \\
\theta_c & \rightarrow \tilde{F} \\
\text{Rigid} & \quad \text{body &} \\
\text{controller} & \left( x_s(t), u(t) \right) \\
\text{Cost} & \rightarrow J \\
\end{align*}
\]

**Fig. 2** Dynamic plant model.
apparent in the following sections. Static CFD simulations are used to generate the aerodynamic forces \( F \). Based on some reference trajectory \( r(t) \), the rigid-body and controller model determines the evolution of the system states \( x_s(t) \) and controller inputs \( u(t) \) over some finite time, which is then used to calculate the cost function \( J \).

II. Dynamic Model

Consider a dynamic system model,

\[
\dot{x}_s(t) = f_s[x_s(t), x_b(t), z(t), u(t), e] \quad (1a)
\]

\[
\dot{x}_b(t) = f_b[x_b(t), z(t), \theta_f, e] \quad (1b)
\]

\[
\frac{\partial z(t)}{\partial t} = f_{\partial z}, \frac{\partial z(t)}{\partial z}, \frac{\partial z(t)}{\partial z}, \ldots, u(t), x, y, z, \theta_f, e \quad (1c)
\]

where \( e > 0 \) is constant; \( x_s(t) \) 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 \( x, y, z \). The controller states are represented by \( x_b(t) \) with reference trajectory \( r(t) \).

Property 1: The control input \( u(t) = g[x_s(t), x_b(t), z(t), r(t), \theta_f] \) is designed such that Eqs. (1a) and (1b) are asymptotically stable by a partial differential equation in spatial dimensions \( x, y, z \). The controller states are asymptotically stable and there is time-scale separation between the coupled system \( x_s(t), x_b(t) \) and \( z(t) \), then \( e \) is small, and the coupled system is approximated by

\[
\dot{x}_s(t) = f_s[x_s(t), x_b(t), r(t), \theta_f], u(t), 0 \quad (2a)
\]

\[
\dot{x}_b(t) = f_b[x_b(t), z(t), r(t), \theta_f, 0] \quad (2b)
\]

\[
0 = f_{\partial z}, \frac{\partial z(t)}{\partial z}, \frac{\partial z(t)}{\partial z}, \ldots, u(t), x, y, z, \theta_f, 0 \quad (2c)
\]

Let the solutions of Eqs. (2a) and (2b) be defined as \( \tilde{x}_s(t) \) and \( \tilde{x}_b(t) \), respectively, and let \( \tilde{z}(t) = F(\tilde{x}_s(t), u(t), \theta_f) \) be the quasi-steady solution of \( z(t) \).

Property 2: The time-scale separation parameter \( e \) in Eq. (1c) is sufficiently small such that

\[
x_s(t) - \tilde{x}_s(t) = O(e) \quad (3)
\]

\[
x_b(t) - \tilde{x}_b(t) = O(e) \quad (4)
\]

hold uniformly for all \( t \). Furthermore, given an \( e \), there exists a \( t_0 > 0 \) such that

\[
z(t) - \tilde{z}(t) = O(e) \quad (5)
\]

holds for \( t > t_0 \).

Singular perturbation theory has been previously applied to nonlinear ordinary differential equation (ODE) systems [14] 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 used, a similar time-scale approximation can be made. To this end, the time-scale separation parameter \( e \) 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 \( F(\cdot, \cdot, \cdot) \) is typically not available and 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 \( x_s(t) \) and \( u(t) \).

Property 3: For any \( \epsilon > 0 \), there is an interpolated mapping \( \tilde{F}(\cdot, \cdot, \cdot) \), where

\[
\|F(x_s(t), u(t), q_i) - \tilde{F}(x_s(t), u(t), q_i)\| < \epsilon, \quad \forall x_s(t), u(t)
\]

(6)

The variable \( \epsilon \) 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 used. If Properties 1 and 2 are satisfied, then the model in the optimization framework can be approximated by Eq. (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 \( \tilde{F}(\cdot, \cdot, \cdot) \) can be used in place of \( F(\cdot, \cdot, \cdot) \). These three properties are important in enabling accurate modeling of the missile system.

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

\[
\dot{x}_b(t) = \begin{bmatrix} \dot{\alpha} \\ \dot{\psi} \\ \dot{\theta}_f \end{bmatrix} = \begin{bmatrix} \frac{-L(u_d) + mg \cos \gamma_f + q}{M \omega_a^2 \delta_z} \\ \frac{L \omega_a \delta_z}{\lambda_f} \\ \omega_a^2 \delta_z - 2 \omega_a \omega_c \delta_c + \omega_c^2 \delta_c \end{bmatrix}
\]

(7)

where \( \alpha \) is the angle of attack, \( \gamma_f \) 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, \( I_{cy} \) 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 \( \delta_z \) is the commanded fin deflection input, \( \delta \) is the achieved fin deflection, \( \delta_c \) is the fin deflection rate, \( \lambda_f \) is the damping ratio, and \( \omega_c \) is the undamped natural frequency of the actuator. The normal acceleration for the missile is

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

(8)

and this is the system output that is to track \( r(t) \). It is required that the commanded input, \( u(t) = \delta_c \), be subject to a control law such that Property 1 is satisfied. To that end, a three-loop autopilot is used as the controller [15], although any other stabilizing controller such as a tuned proportional integral controller could be chosen. The dynamic equations for the controller are

\[
\dot{x}_b(t) = -q + K_A(\eta - \dot{\eta} + K_{dc} \eta_c)
\]

(9)

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

\[
\dot{\delta}_c = -K_{Rf} q + W_{Rf} x_b(t)
\]

(10)

where \( K_{Rf} \) and \( W_{Rf} \) are two more controller gains. Suitable selection of \( K_A, K_{dc}, K_{Rf} \), and \( W_{Rf} \) is required to satisfy Property 1 as described by Zarchan [15].
Now, define
\[ \epsilon := \frac{\delta_{\text{max}} l_f}{V} \]  
(11)

where \( l_f \) represents the characteristic length of the fin and \( \delta_{\text{max}} \) is the maximum fin deflection rate. The CFD model in this example is described by the dimensionless Euler equations, that is,
\[ \frac{\partial \tilde{z}}{\partial t} = -\tilde{V} \cdot \tilde{f} \]  
(12)

where \( \tilde{z} = [\tilde{\rho} \ \tilde{V} \ \tilde{E}]^T \), with \( \tilde{\rho} = \rho/\rho_i \) representing the dimensionless density of the fluid; \( V := 1/V_1 \) is the dimensionless fluid velocity vector; and \( \tilde{E} := E/V^2 \) is the dimensionless specific total energy. The divergence operator \( \tilde{\nabla} := \delta_{\text{max}} V \) and the dimensionless convective flux is \( \tilde{f} := [f_1 \ f_2 \ f_3]^T \) with
\[ f_i = \left( \frac{\tilde{\rho} \tilde{V}_i + \tilde{\rho} \tilde{\delta}_i}{\tilde{\rho}_i \tilde{E} + \frac{1}{2} \tilde{V}^2} \right) \]  
(13)

where \( \tilde{V}_i = v_i/V \), \( \tilde{\delta}_i = [\delta_{1i} \ \delta_{2i} \ \delta_{3i}]^T \) is a vector of Kronecker delta functions, the dimensionless pressure is given by
\[ \tilde{\rho} = \frac{\rho V (1 - \epsilon)}{\rho_i V_i^2} \left( E - \frac{1}{2} \| V \|^2 \right) \]  
(14)

\( \epsilon \) is the ratio of specific heats.

For supersonic flight conditions, the pitching and actuator states are an order of magnitude smaller than those in the flowfield. In other cosimulation papers of supersonic systems using CFD and rigid-body dynamics models [16,17], 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 2 for the flow (\( V = 680 \text{ m} \cdot \text{s}^{-1} \)) and the maximum actuator fin rate is \( \delta_{\text{max}} = 100 \text{ rad} \cdot \text{s}^{-1} \) and \( l_f = 0.33 \text{ m} \), 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. To find an approximate solution to the partial differential equations, application of spatial discretization, such as the finite-volume method, to Eq. (12) is necessary. This results in a system of ODEs, where for each discretized cell \( i \)
\[ \frac{\partial z_{i,t}}{\partial t} = -\frac{1}{V_i} \int_{S_i} f \cdot n \, dS \]  
(15)

where \( z_{i,t}(t) \) is the volume-averaged solution of \( z(t) \) for each cell, \( V_i \) is the cell volume, \( n \) 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 Eq. (15) to the steady-state solution.

To populate the maps of the aerodynamic forces, for each missile shape design candidate \( \theta_s \), steady-state CFD simulations for a series of missile poses are evaluated. That is,
\[ \tilde{F}(\alpha, \delta, \theta_s) = [D(\alpha, \delta, \theta_s) L(\alpha, \delta, \theta_s) M_y(\alpha, \delta, \theta_s)]^T \]  
(16)

As stated in Property 3, the chosen interpolation scheme necessarily affects the approximation error. Missile pitch axis aerodynamic maps are typically quite linear, and 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 \( \epsilon \) small.

### III. Framework Overview

The optimization framework is shown in Fig. 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, \( x_s(t) \) and \( u_s(t) \), where the design variables of the fast states \( z(t) \) are held constant. The optimal design variables in the inner loop are given by
\[ \theta^*_s(\theta_s) = \arg \min_{\theta_s} J(\theta_s, \theta_f) \]  
subject to \( g_m(\theta_s) \geq 0 \) and \( h_m(\theta_s) = 0 \)

where \( J \) is a cost function and \( g_m(\cdot) \) and \( h_m(\cdot) \) are, respectively, inequality and equality constraints. The constraints \( g_m(\cdot) \) include bounds on the design variables, and \( h_m(\cdot) \) includes the slow system dynamics (2a) and (2b). The outer optimization problem considers the design parameters associated with the fast states \( z(t) \) and with the restriction \( \theta^*_s(\theta_s) \). The optimal design variables of the outer loop are given by
\[ \theta^*_f(\theta_s) = \arg \min_{\theta_f} J(\theta_s, \theta_f) \]  
subject to \( \theta_{f,\text{ad}} \geq \theta^*_f(\theta_s) \) and \( h_{\text{con}}(\theta_f) = 0 \)

where \( J \) is a cost function; \( \theta_{f,\text{ad}} \) and \( \theta_{f,\text{ad}} \) are, respectively, the upper and lower bound constraints on the design variables, and \( h_{\text{con}}(\cdot) \) 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 \( \theta_s := \{ \theta_s, \theta_m \} \), where \( \theta_s \) represents the variables of which the gradients are calculated via least-squares finite differencing and \( \theta_m \) represents the variables of which the gradients are calculated via the adjoint method. Details of each component of the framework are described in the following subsections.

**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 \( \theta_s \) directly affect the CFD simulations, and the controller parameters \( \theta_m \) 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).
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,

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

(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 the 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 represent all desired performance characteristics and operating conditions, it does implicitly capture the tradeoff 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 Figs. 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 algorithm [18]. The second type of design variables is the control parameters. The three-loop autopilot has been chosen as the controller. This autopilot has four gain variables, but rather than tuning each of these gains, Zarchan’s tuning method [15] is used. Three controller characteristic parameters are specified in the tuning method, namely, the rise time $\tau_c$, damping ratio $\zeta_c$, and open-loop crossover frequency $\omega_{cr}$. Let $\theta_s = [X_1, X_2, X_3, X_{10}, X_{11}]^T$ and $\theta_d = [X_4, X_5, \ldots, X_9]$ be the vectors of shape deformation variables, and let $\theta_c = [\tau_c, \zeta_c, \omega_{cr}]^T$ 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 shows the cost function with respect to two of the controller parameters $\tau_c$ and $\zeta_c$ with $\omega_{cr}$ held constant.

### B. Inner Optimization

The rigid-body dynamics model and controller model, respectively, Eqs. (2a) and (2b); the design variables $\theta_s$; 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).

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 used, 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 solver is used to find the optimal tuning parameters.

---

**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.

**Fig. 7** Continuity of cost function with respect to the nose-tip geometry design variable.

---

\footnote{Data available online at www.mathworks.com/products/matlab/ [retrieved 7 April 2016].}
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 Fig. 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 overlain on the cost function surface. The constraints on the inner optimization are then the dynamic equations (7) and (9) and the bounds

\[
\begin{align*}
-\theta_s + \theta_{s, U} & \\
\theta_i - \theta_{i, L} & \\
G_m - G_m & \\
\end{align*}
\]

where \(\theta_{s, L}\) and \(\theta_{s, U}\) 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 \(\theta_i\), 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 an optimizer that can tolerate numerical approximation of solutions can be treated as noise. The missile example uses 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 ten 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 \(\gamma\). In the context of the optimizer, as the optimization 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 discretized 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 [19]. 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 (20). The local model provides gradient information for a line search algorithm. In this paper, the implicit filtering algorithm is modified to neglect sampling in \(\theta_i\). Gradient information is found using the adjoint method instead. The modified algorithm is given in Appendix C.

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 Fig. 1 for sharp-edged geometry variables. In this paper, two candidate gradient methods are used, 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 Fig. 10. Let \( S \) represent the baseline surface and \( S' \) be the deformation of the surface at coordinate \( s \). The surface and its deformation are considered to be smooth if the following property is satisfied.

Property 6: A normal vector \( n_S(s) \) is well defined at \( s \) such that

\[
S' = \{ s + \delta S(s) n_S(s), s \in S \}
\]

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.

Table 1 shows the gradient of the lift coefficient cost function with respect to a selection of tail-fin design variables calculated using the LSFD and the continuous adjoint method and with the central finite difference (FD) method as a benchmark for comparison. The total computational time and number of CFD simulations for each method are 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. The LSFD is applicable for all design variables and is also faster than the central finite-difference method. Note that the speedup for the LSFD method is due to error control of the CFD iterations (i.e., the parameter \( \gamma \)), while the speedup 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 used for design variables \( \theta_{ad} \) defined over smooth surfaces (i.e., tail-fin profile), and the LSFD is used for design variables \( \theta_s \) 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 approximations 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 codesign 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 the previous properties 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 on the noise characteristics of \( \Phi(\gamma) \), on the update law \( f(\cdot) \) of the optimizer, and on 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 optimizer will approach the minimum more quickly. In particular, the optimizer may converge to the solution without necessarily requiring \( \gamma \) 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 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 used for the gradient calculation for a subset of design variables. The computational advantage of using 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 10 g-force positive and negative step change in commanded acceleration over a 3 s period is used, and fin deflection is saturated at 10 deg. The cost function is taken from Eq. (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

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

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 Fig. 6). The aerodynamic data for the missile body are taken directly from the computer program DATCOM since there is no deformation over the body. CFD simulations of the tail fin at \( d = 0, 2.5, 5, 7.5, 10 \) deg are conducted. The control parameters are also optimized by the inner loop. These 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 uses finite differencing) has been included for comparison. In the example, all three runs converge to the same optimal design.

Table 2 shows the optimized control parameters for the initial and final shapes, 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.

The original and optimized tail-fin shapes are shown in Fig. 11. It can be seen that the optimal planform shape favors a shorter wing span 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

**Both examples were executed on an Intel Core™ i7-2600 3.4GHz desktop PC with 8 GB of RAM.
vary by much suggests that the optimization was only able to minimize the drag component of the cost function. 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 (X₁ to X₃ in Fig. 5 and X₄ to X₁₁ in Fig. 6). CFD simulations are required for both the missile body and tail fin. For the nose, CFD simulations

<table>
<thead>
<tr>
<th>Shape</th>
<th>τ₀</th>
<th>ζ₀</th>
<th>ω₀₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>0.1165</td>
<td>1.7664</td>
<td>105</td>
</tr>
<tr>
<td>Optimized</td>
<td>0.1158</td>
<td>1.7626</td>
<td>105</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fig. 11</th>
<th>Original and optimized shapes of the tail fin.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Fig. 12</th>
<th>Implicit filtering comparison with and without error control.</th>
</tr>
</thead>
</table>

Tail fin (span and leading edge optimization)

- - - original
--- optimized

<table>
<thead>
<tr>
<th>Fig. 13</th>
<th>Shapes of original and optimized missile (body of missile omitted).</th>
</tr>
</thead>
</table>

Tail fin (span and leading-edge optimization only)

<table>
<thead>
<tr>
<th>Fig. 14</th>
<th>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.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Fig. 15</th>
<th>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.</th>
</tr>
</thead>
</table>

Table 2 Optimal control parameters for original and optimized tail-fin planform shapes

<table>
<thead>
<tr>
<th>Shape</th>
<th>τ₀</th>
<th>ζ₀</th>
<th>ω₀₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>0.1165</td>
<td>1.7664</td>
<td>105</td>
</tr>
<tr>
<td>Optimized</td>
<td>0.1158</td>
<td>1.7626</td>
<td>105</td>
</tr>
</tbody>
</table>
are conducted at \( a = 0, 4, 8, 12, 16 \) deg, and for the tail fin, they are conducted at \( a = 0, 4, 8, 12 \) deg. 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 optimization cases are presented. First, only the nose and the tail-fin planform are optimized; that is, the design variables using the adjoint method are omitted. Second, the nose, tail-fin planform, and profile are optimized. The results of the optimized nose and planform shape are the same for both cases and are shown in Fig. 13. Note that the body of the missile, i.e., between 1.4 and 3.6 m, has been omitted in the diagram so that differences in the nose and tail-fin shapes are 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. To compare the profile shape, cross-sections through the tail fin are shown in Figs. 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 for which only the planform has been optimized. Figure 15 shows the tail fin for which 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 if the optimization surface is nonconvex, a multistart approach 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 speedup in the optimization time. This is particularly useful when using 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 equations with an associated turbulence model. Convexity is a property of the optimization metric, and if the optimization surface is nonconvex, a multistart 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 has similar time-scale separation between the electrical and mechanical states. The optimization of rotor and stator geometry may also rely on electromagnetic field simulations that are computationally intensive.

### Appendix: Algorithms

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

#### Appendix A: Least-Squares Finite-Difference Method

The least-squares finite-difference method [19] calculates the gradient based on a linear least-squares fit of sampled points. Let \( V = \{ v_1, \ldots, v_K \} \) be the stencil search directions. For a central differencing stencil, the search directions are \( V = \{ J_k \} \) and \( K = 2N \). To calculate the gradient, the cost function is evaluated at some point \( \theta_i \) in the design space and points on the stencil \( S(\theta_i, h, V) = \{ Z | Z = \theta_i + hv_k, 1 \leq k \leq K \} \), where \( h \) is the finite-difference step size. This gives the gradient

\[
\frac{dJ}{\partial \theta_i} = \frac{1}{h} \delta J(\theta_i, h, V) V^T \tag{A1}
\]

where

\[
\delta J(\theta_i, h, V) = [J(\theta_i + hv_1) - J(\theta_i), \ldots, J(\theta_i + hv_K) - J(\theta_i)]^T \tag{A2}
\]
and $V' := V'(VV')^{-1}$ is the pseudoinverse of $V$. Note that a central differencing stencil requires $2N + 1$ simulations.

### Appendix B: Continuous Adjoint Method

The continuous adjoint method [4] is typically applied to calculate the gradient of static steady-flow aerodynamic costs such as the lift or drag. To calculate costs of dynamic systems [6], the chain rule is used, in which

$$
\frac{\partial J}{\partial \theta_{ad}} = \sum_k \frac{\partial J}{\partial \hat{F}(x_{ad}(k), u_k(t))} \frac{d\hat{F}(x_{ad}(k), u_k(t))}{d\theta_{ad}}
$$

(B1)

and in which $[x_{ad}(k), u_k(t)]$ are the $k$ steady-state points over which the interpolated map $\hat{F}(\cdot, \cdot)$ has been handled. On the right-hand side of Eq. (B1), 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 Eq. (B1) is to calculate the gradient via the adjoint method for each value in the map $\hat{F}(\cdot, \cdot)$ for the $k$ points. In the missile example given in paper, the $\hat{F}(\cdot, \cdot)$ has three components [see Eq. (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

$$
\hat{F} = \int_d d \cdot (p n_s) ds
$$

(B2)

where $d$ is a force projection vector, $p$ is the pressure, and $n_s$ is the local normal vector on the surface. Set

$$
\frac{\partial J}{\partial \hat{F}(x_{ad}(k), u_k(t))} = G
$$

(B3)

where the derivatives of $J$ with respect to $\hat{F}(x_{ad}(k), u_k(t))$ are found using finite differences[24] and each row of $G$ in Eq. (B3) corresponds to the forces and moments in the map $\hat{F}(\cdot, \cdot)$. For the tail fin, where the

pose only consists of the fin deflection $\delta$, define

$$
G := C_{\infty} \begin{bmatrix}
\cos \delta & \sin \delta & 0 \\
-\sin \delta & \cos \delta & 0 \\
0 & 0 & \frac{-\delta v_s}{L_{out}}
\end{bmatrix}
$$

(B4)

where $C_{\infty} = (1/2)V^2 \rho_a A_s$. The freestream density is $\rho_a$, the reference length is $L_{out}$, and the reference area is $A_s$. Equations (B3) and (B4) are implemented in Eq. (B1) so that only $k$ adjoint simulations are required.

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

$$
\delta \hat{F} = \frac{\partial \hat{F}}{\partial \delta z} \delta z + \frac{\partial \hat{F}}{\partial \theta_{ad}} \delta \theta_{ad}
$$

(B5)

It can be expensive to compute variations in the fast states $\delta z$; that is, each variation will require an additional CFD simulation. The aim of the adjoint approach is to eliminate this term in Eq. (B5). Suppose that partial differential equations of the form (2c) are introduced as an equality constraint.

### Appendix C: Implicit Filtering Algorithm

The steps of the implicit filtering algorithm [19] are:

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

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 from Property 4 that the selection of successive $\gamma$ values be increasing so that the noise level decreases as the algorithm proceeds.

3) Initialize the current design point $\theta_0 = \theta_{f,0}$, the step size $h = 0.5$, the error control $\gamma = \gamma_1$, the Hessian $H = I$, and the iteration level $k = 1$.

4) Let $\tau > 0$ be the termination constant. The value $\|rh\|$ 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 $J(\theta_{f}, \gamma)$.

b) Compute the gradient $VJ(\theta_{f}, \gamma)$. The gradient can be calculated, for example, by using either least-squares finite differencing (A1) or the adjoint method (B1). If least-squares finite differencing is used, then store the value of the cost function at each stencil point as well.

c) If $\|rh\| > \tau$, go to step (5f), where the projection operator.

set the line search design point $\theta_k = \theta_{f}$.

e) While $\|\theta_{k+1} - \theta_k - VJ(\theta_{k}, \gamma)\| \geq \tau h$:

f) Update the model Hessian $H$ using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton method [21].

\[ R(z, \theta_{ad}) = 0 \]
ii) Determine the search direction,

\[ s = -H^{-1} \nabla J(\theta_+). \]

iii) Perform a backtracking line search

\[ \theta_+ = \mathcal{P}(\theta_+ - \lambda s) \quad \text{(C1)} \]

where \( \lambda \) is initialized 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 \( \theta_{f,\text{min}} \) from the stencil (if least-squares finite differencing is used) or from the successful line search, and set \( \theta_k = \theta_{f,\text{min}}, h = h/2, \gamma = \gamma_{k+1}, \) 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

doi:10.2514/1.J052182


doi:10.1007/BF01061285


doi:10.2514/2.2413


doi:10.2514/1.J052184


K. E. Willcox
Associate Editor