Aerodynamic Shape Optimization via Global Extremum Seeking

Kuan Waey Lee, Student Member, IEEE, William H. Moase, Sei Zhen Khong, Andrew Ooi, and Chris Manzie, Senior Member, IEEE

Abstract—Optimization of aerodynamic shapes using computational fluid dynamics (CFD) approaches has been successfully demonstrated over a number of years; however, the typical optimization approaches employed utilize gradient algorithms that guarantee only the local optimality of the solution. While numerous global optimization techniques exist, they are usually too time consuming in practice. In this brief, a modified global optimization algorithm (DIRECT-L) is introduced and is utilized in the context of sampled-data global extremum seeking. The theoretical framework and conditions under which the convergence to the steady state of the CFD solver can be interpreted as plant dynamics are stated. This method alleviates the computational burden by reducing sampling and requiring only partial convergence of the CFD solver for each iteration of the optimization design process. The approach is demonstrated on a simple example involving drag minimization on a 2-D aerofoil.

Index Terms—Adaptive control, aerodynamics, aerospace engineering, optimization, partial differential equations.

I. INTRODUCTION

There is a growing reliance on using simulation tools to speed up the development life cycle of new high-technology engineering products. This is particularly the case with aerodynamic systems, where computational fluid dynamics (CFD) tools have been used effectively for predicting aerodynamic characteristics such as lift and drag. Coupling CFD approaches with optimization tools has a natural attractiveness, as this potentially reduces the number of hardware prototypes required in the design of a new aerodynamic system.

With this aim in mind, the use of optimization tools within the aerodynamic field was pioneered in [1] and [2]. This adjoint optimization approach used two iterations of a solver to estimate the gradient of the cost function with respect to design parameters for a system of partial differential equations (such as those describing fluid flow). The first iteration is used to solve for the states of the physical system, and the second iteration (in reverse time) solves for the adjoint variables thereby delivering the sensitivity of the cost function to all design variables simultaneously. The gradient may then be used in standard descent techniques to reduce the system cost. The key advantages of this approach are the simultaneous retrieval of all gradients, irrespective of the number of design variables, therefore making it well suited to problems with many degrees of freedom, and the fact that (in the case of the discrete adjoint method) an actual gradient is produced unlike gradient estimation techniques including deterministic finite-difference methods [3] and stochastic algorithms like SPSA [4]. However, it has been reported in [5] that inaccuracies in the gradient calculation exist for the continuous adjoint method and thus reliance on the robustness of the gradient-based optimizer is necessary. Moreover, as with all gradient-based optimization approaches, the presence of local minima may limit the extent to which performance can be optimized.

Naturally, these problems have been previously encountered in the optimization literature, leading to the development of global optimizers that are capable of finding the extremum of unknown static maps given sufficient time and potentially some information regarding the optimization surface. These can be broadly classified into stochastic-based approaches (including evolutionary algorithms [6] and particle swarm methods [7]) along with deterministic approaches (including the Piyasvskii–Shubert algorithm [8] and the dividing rectangles (DIRECT) algorithm [9]).

Within the CFD community, these global optimizers have been deployed in an attempt to circumvent the limitations of local optimization techniques such as the adjoint approach. Such implementations strictly require full convergence of the CFD solver to meet the requirements of the optimization method. There have been some ad hoc implementations that utilize partial convergence of the solver in global optimization [10] and also in one-shot adjoint approaches [11].

There have been a number of developments in the extremum seeking literature since the early 2000s [12] including a number of key theoretical developments [13]–[15], that have spawned numerous applications (see, for example, the references contained in [16]). The key difference between optimizers and extremum-seeking algorithms is the presence of plant dynamics. More recently, the extremum-seeking methodology has been extended to include global optimizers [17]–[19], which use sampled data approaches and consequently broaden the class of problems that can be tackled as the reliance on local gradients may be removed.

At a high level, there are clear parallels between the goal of global extremum seeking on dynamic plants and shape optimization using CFD solvers. In [20], preliminary work for the conditions under which the convergence of a numerical CFD solver to a steady-state solution mimics a traditional dynamic plant were first outlined, and this then allowed sampled data global extremum seeking to be used for shape optimization problems in fluid dynamics.
In this brief, a new modification of the DIRECT algorithm (DIRECT-L) is introduced that makes use of knowledge of the cost function’s Lipschitz constant to reduce the amount of sampling. It is shown that combining this with the partial convergence result in [20] further reduces the total computational effort of the optimization process. The theoretical foundation for this proposed framework is also provided.

This brief is laid out as follows. Section II outlines the proposed approach, where the theoretical framework of the optimizer is described in detail in [9], but is summarized again here for completeness. A minimization problem utilizing the DIRECT algorithm is written as

\[ \min_{u \in \Omega} Q(u) \quad (1) \]

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

\[ \Omega := \{ u \in \mathbb{R}^m | v_i \in [0, 1] \subset \mathbb{R}, \quad i = 1, 2, \ldots, m \} \quad (2) \]

Let \( q \) denote the iteration number of the algorithm and \( k \) the total number of samples.

**Step 1 (Initialization):** Initialize \( q := 1, k := 0 \), and \( u_0 \) as the center of \( \Omega \). Let the output \( y_1 = Q(u_0) \) and \( \hat{y}_0 = y_1 \). Let \( S = \{\Omega\} \).

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

\[ Q(u_j) - \tilde{L}d_j \leq Q(u_i) - \tilde{L}d_i \quad \forall i \]

(3)

\[ Q(u_j) - \tilde{L}d_j \leq \hat{y}_k - \epsilon |\hat{y}_k| \quad \forall i \]

(4)

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

**Step 3 (New Hyper-Rectangles):** For every \( j \in S \), divide the \( j \)th hyper-rectangle with center \( u_j \) according to the following rules:

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

2) Sample \( Q \) at the points \( u_j \pm \delta e_i \) for all \( i \in I \).

3) Divide the box \( j \) containing \( u_j \) into thirds along the dimensions in \( I \) starting with the dimension with the lowest value of \( w_i := \min(Q(u_j + \delta e_i), Q(u_j - \delta e_i)) \) and continuing to the dimension with the highest \( w_i \).

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

\[ \hat{y}_q := \min_{i=1,k} y_i. \quad (5) \]

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

**Step 6 (Termination):** In [18], it is proposed that the termination of the DIRECT algorithm occur when it is within some tolerance of the estimate of the lower bound of \( Q \). Suppose that there is some knowledge of the Lipschitz constant and that a constant \( \bar{L} \) is (conservatively) chosen such that \( \bar{L} > L \). After \( q \) iterations, for some hyper-rectangle \( j \) with center at \( u_j \) and distance to vertices \( d_j \), an estimate of the lower bound of \( Q \) on the hyper-rectangle is given by \( Q(u_j) - \eta_j \), with \( \eta_j := \bar{L}d_j \). As the hyper-rectangles are further divided in successive iterations, the estimate of the lower bound approaches the minimum of \( Q \). DIRECT terminates when \( \eta := \max_j \eta_j \) drops below some specified tolerance, \( \eta^* \), where it is assumed \( \eta^* < \bar{L}d_0 \).

**Remark 1:** If no information about the Lipschitz constant is available, then an alternative termination criterion could be applied, such as an upper limit on the number of samples.

The DIRECT algorithm searches the space in a dense manner, i.e., all hyper-rectangles will eventually be identified as potentially optimal if DIRECT is allowed to run to infinity. This implies that only a subsequence of the sampling points converges to the global optimum. In particular, the following result can be found in [9].

**Proposition 1:** Let \( y^* := \min_{u \in \Omega} Q(u) \). The output sequence \( \{y_q\}_{q=1}^{\infty} \) of the DIRECT algorithm contains a subsequence that converges to \( y^* \). It follows that \( \lim_{q \to \infty} \hat{y}_q \to y^* \), where \( \hat{y} \) is as defined in (5).

Along with the termination criteria in Step 6 of the DIRECT algorithm, knowledge of the Lipschitz constant (or a conservative estimate thereof) can also be used to reduce the sampling of the DIRECT algorithm. Conditions (3) and (4) are modified such that \( \bar{L} \leq \tilde{L} \) is enforced when identifying potentially optimal hyper-rectangles. This modification is denoted as DIRECT-L. On the other hand, the restriction imposed by the DIRECT-L modification means that certain hyper-rectangles will be identified as nonoptimal and omitted from the search, which effectively leads to less sampling. Furthermore, the sequence of sampling points converges to the global optimum as detailed below.

**Proposition 2:** The output sequence \( \{y_q\}_{q=1}^{\infty} \) of the DIRECT-L algorithm has the property that \( \lim_{q \to \infty} y_q = y^* \).

**Proof:** The convergence proof follows from that of the unmodified DIRECT algorithm given in [9]. All new hyper-rectangles are generated by first dividing existing potentially optimal hyper-rectangles into thirds along the hyper-rectangle’s longest side lengths. Therefore, a rectangle can only have side lengths of \( 3^{-k} \) for \( k = 1, 2, 3, \ldots \).
After \( r \) divisions, a hyper-rectangle will have \( j := r \mod m \) sides of length \( 3^{-(k+1)} \) and \( m - j \) sides of length \( 3^{-k} \), where \( k := (r - j)/m \). The distance from center to the vertices is given by

\[
d := (j3^{-2(k+1)} + (m - j)3^{-(2k)})^{1/2}.
\]  

(6)

Thus, \( d \) approaches zero as \( k \) increases. Choosing \( \hat{L} > L \) ensures that the global minimum lies inside a hyper-rectangle that is considered to be potentially optimal, and therefore, the sequence will converge to the minimum.

Remark 2: Note that for DIRECT-L, the sequence of sampling points no longer forms a dense subset in the search domain as in the case of DIRECT, but instead concentrates its search around the minimum.

B. DIRECT Algorithm for Global Extremum Seeking

Consider a time-invariant MISO dynamic system subject to bounded input \( u \)

\[
\dot{x} = f(x, u) \\
y = h(x, u).
\]

(7)

Let \( x(t, u, x_0) \) and \( y(t, u, x_0) \) denote the solutions of (7) for \( x \) and \( y \), respectively, at time \( t \), given initial conditions \( x_0 \) and input \( u \in \Omega \).

Assumption 1 (Paraphrased From [18]): The dynamic plant possesses the following qualities for any constant \( u \) in the allowed range:

i) for any \( u \), there exists \( x^* \) such that \( x = x^* \) is a global asymptotic stable equilibrium of (7) uniformly in \( u \);

ii) the static map \( Q(u) := \lim_{t \to \infty} y(t, u, x_0) \) is locally Lipschitz.

The DIRECT and DIRECT-L algorithms with the modified termination constraint have the following convergence property.

Theorem 1: Given \( \Delta > 0 \) and \( \mu > 0 \), and let the termination threshold \( \eta^* \in (0, \mu) \) and \( u^* \) be the minimizing input at termination of DIRECT-L. There exists a critical sample time \( T^*_c > 0 \) such that for any \( \|x_0\| < \Delta \), the DIRECT and DIRECT-L algorithms will terminate with

\[
|y^* - Q(u^*_c)| \leq \mu \quad \text{for all } T > T^*_c.
\]

(8)

Proof: The proof of convergence for DIRECT-L follows directly from the proof for DIRECT found in [18]. The main steps are summarized here. Let \( \nu := (\mu - \eta^*)/2 \). If Assumption 1i) holds for any \( \|x_0\| < \Delta \), then for any sufficiently large \( T_c \)

\[
|y_k - Q(u_{k-1})| \leq \nu \quad \text{for all } k = 1, 2, \ldots
\]

(9)

Proposition 2 states that as the DIRECT-L algorithm iterates it converges to \( y^* \), and then from [18, Lemma 9]

\[
|Q(u_k) - y^*| \leq 2\nu + \eta^* = \mu.
\]

(10)

As discussed in Remark 2, an appropriate choice of \( \hat{L} \) ensures the convergence of DIRECT-L to the global minimum.

C. CFD Solver as a Plant?

Computational modeling of steady fluid flow over a body, e.g., by the Reynolds-averaged Navier–Stokes (RANS) equations or Euler equations which are partial differential equations, is achieved through spatial discretization via the use of finite-element/difference/volume methods. This leads to a very large (although finite) number of discrete cells representing the states of the fluid at different spatial locations in the simulation domain. The states are fluid properties such as density and velocity. An example of this discretization for an aerofoil problem is shown in Fig. 1 consisting of just over 10000 cells. Resulting from the discretization process are coupled nonlinear ordinary differential equations of the form

\[
\frac{dx_h}{dt} = g(x_h, G(u))
\]

(11)

where in the context of this brief, \( x_h \) represents the fluid states in each cell, and the input to the CFD solver is considered to be the mesh, \( \theta \). The mesh is generated from \( u \) which is a set of parameters describing the deformation of the aerodynamic body. This mesh generation process is treated as a mapping \( \theta = G(u) \). The plant is then considered to be the combination of the mesh generation process and the CFD solver. Note that the flow is assumed to be steady so that the states and output measurement approach constant values. These ordinary differential equations (11) can be solved to steady state through iterative time-marching methods.
The residual error is a measure of the error of the states of the numerical solution to the true solution. The convergence of the CFD solver is typically measured by the reduction of the residual error of all (or subset of) the states, where the solver is set to terminate when this error has dropped below some predefined threshold. Let the set of states at the interface between the fluid and the surface of the body be designated $I_B$, i.e., $i \in I_B$ if and only if $x_{h,i}$ is a cell adjacent to the surface of the body. The aerodynamic drag (or lift) on a body is then estimated through the summation involving a function of the states within $I_B$

$$y = \sum_{i \in I_B} C(x_{h,i}, G(u)).$$  \hspace{1cm} (12)

A global extremum seeker utilizing the DIRECT method is proposed in the manner shown in Fig. 2. In this arrangement, there are two effective time scales being utilized. The CFD solver utilizes variable step time-marching iterations to converge toward a steady-state solution, and these iterations are nominally denoted to have a variable time step $T_{CFD}$. It is noted that the wall-clock time associated with any given $T_{CFD}$ may be highly variable. The global extremum seeker samples the numerical solver after a waiting time expressed as a multiple $N$ of the CFD iterations, i.e., $T_{ES} = NT_{CFD}$.

With this description of the combined mesh generator and CFD solver as a plant, the following properties are required.

**Property 1:** For all allowable $u$, the mapping $G(u)$ is locally Lipschitz. Moreover, the sets of indices of the mesh points on the fluid-surface interface ($I_B$) and on each boundary of the numerical domain are independent of $u$.

**Property 2:** The numerical CFD solver satisfies the following,

i) Let $\mathcal{D} := \{G(\sigma) : \sigma \in \Omega\}$. For all $\theta \in \mathcal{D}$, $\exists x_{h}^{\ast}(\cdot)$ such that $x_{h}^{\ast}(\theta)$ is an equilibrium of (11). Moreover, the equilibrium solution is semiglobally asymptotically stable uniformly in $\theta$.

ii) $l(\theta) := \sum_{i \in I_B} C(x_{h,i}^{\ast}(\theta), \theta)$, is locally Lipschitz.

iii) $x_{h}$ is initialized at each extremum seeker sample time, $T_{ES}$, with the final values of the states of the previous CFD solution.

Property 1 simply ensures that the mesh generation process is continuous with respect to the parameters describing the aerodynamic body. Notably, Property 1 prevents meshing techniques that add or remove cells from the numerical domain when the geometry is changed and this is the main reason for the use of mesh deformation rather than remeshing for each new design $u$. Furthermore, for a large $u$, the mesh deformation process can result in cells with large aspect ratios, which can lead to numerical instability in the solver.

There exist a number of preprocessing tools to test the geometric health of a mesh and these can be employed to check that Property 1 is met. Alternatively, one can bound the range of allowable $u$ to conservative values. Property 2 is related to the behavior of the CFD numerics. Specifically, Property 2i) is analogous to requiring the stability of the dynamic plant, and ensures that Assumption 1i) is satisfied. Property 2ii) (in combination with Property 1) is analogous to requiring the steady-state behavior of the dynamic plant to be Lipschitz and ensures that Assumption 1ii) is satisfied. Property 2iii) essentially requires hot starting of each CFD run with the final state of the previous CFD run. This ensures that the CFD numerics behave like a traditional dynamical system by avoiding a periodic resetting of the plant’s states at each sampling instant.

Although it is a comparatively straightforward task to ensure the mesh generation process and CFD solver satisfy Properties 1 and 2ii), it is a less simple task to ensure that Properties 2i) and 2ii) are satisfied. To achieve this, the stability of the solver is considered. Steady flow CFD solvers are built by discretizing the weak form of the steady RANS or Euler equations. Stability of these nonlinear equations can be shown by application of linear methods, such as von Neumann and matrix methods [21] or nonlinear methods, such as the total variation diminishing method [22]. Let $x_{h,0}$ be the state for which the CFD solver is initialized. If $|x_{h,0}(\theta) - x_{h}^{\ast}(\theta)| \leq \Delta$, and if the solver is stable, then there exists a $\beta(\cdot, \cdot) \in KL$ such that

$$|x_{h}(t) - x_{h}^{\ast}| < \beta(|x_{h}(0) - x_{h}^{\ast}|, t).$$ \hspace{1cm} (13)

In other words, (13) states that the solution of the CFD solver is semiglobally asymptotically stable [23, Lemma 4.5]. In addition to the stability, the solver must be consistent, i.e., the discretization error tends to zero, and convergent, the approximate solution tends to the exact solution, as the mesh, $\Delta x$, and the time-step, $\Delta t$, both tend to zero. It is assumed that the mesh and time-step have been sufficiently resolved to ensure that the aforementioned properties are satisfied.

**Proposition 3:** The cascade of mesh generation and CFD solver, i.e., (11), can be modeled as (7). Furthermore, it satisfies Assumption 1.

**Proof:** This is achieved via satisfaction of Properties 1 and 2 by the means discussed above.

The control scheme (i.e., the global extremum seeker) takes a finite amount of time to find an input to the system which drives the steady-state behavior into a neighborhood of a global extremum. The size of this neighborhood can be reduced at the expense of an increase in the waiting period $T_{ES}$ (through an increase in $N$). Coupled with a change in the termination criterion of DIRECT, a further reduction in the size of this neighborhood can be obtained, but both these changes are at an expense of additional CFD computation.
Fig. 3. Residual error of fluid density states. Inset: showing initial divergence of residual error.

**Proposition 4:** Let \( u_{\text{min}} \) be the geometrical parameters that minimize \( Q \), and \( u_{\text{ES}} \) be the optimal geometrical parameters resulting from the DIRECT-L optimization. Then

\[
\forall \varepsilon > 0, \; \exists T^*_e \; \text{s.t.} \; |Q(u_{\text{min}}) - Q(u_{\text{ES}})| < \varepsilon \quad \forall T_{\text{ES}} > T^*_e.
\]

**Proof:** In view of Proposition 3, the claim follows directly from Theorem 1.

Fig. 3 shows the residual error plotted against the time-marching iterations of the CFD solver. The dynamics of the CFD code during stabilization may lead to some initial divergence in the residual error. The choice of the number of CFD iterations may need to be conservative, as it is required not only for the convergence of the initial nominal system, but also for every subsequent shape perturbation within the optimization process.

Fig. 3 is also useful in clearly highlighting the motivation for early termination of the CFD solver (at around 10–50 iterations). Given the number of states involved in each iteration of the CFD code, there is a large computational burden associated with each iteration. Waiting for full convergence (i.e., approximately 300 iterations) of the CFD at each iteration of the extremum seeker therefore induces a significant additional computational expense, particularly when the global search method necessarily will explore wide regions of the input space that may later be dismissed as nonviable.

### III. Results

To demonstrate the proposed approach, the drag minimization of an aerofoil at an angle of attack of \( 1.25^\circ \) and a Mach number of 0.8 is considered. The CFD solver package used here is the SU2 open-source CFD code [24]. In addition to the CFD solver, the package incorporates mesh deformation and adjoint subroutines. The initial mesh, as shown in Fig. 1, was also provided with the package and is assumed to be of good quality. Consider the 2-D Euler equations, which are nonlinear partial differential equations used to describe fluid flow

\[
\frac{\partial x}{\partial t} + \nabla \cdot F = 0
\]  

where \( x = [\rho, \rho u, \rho v, \rho E]^T \) are the states of the system, with \( \rho \) representing the density of the fluid, \( u \) and \( v \) are the fluid velocities, and \( E \) is the enthalpy. For a 2-D flow problem, \( F = [F_x, F_y] \) is the convective flux, with

\[
F_x = \begin{pmatrix} \rho u \\ \rho u v + p \\ \rho u (E + \frac{p}{\rho}) \end{pmatrix}, \quad F_y = \begin{pmatrix} \rho v \\ \rho u v + p \\ \rho v (E + \frac{p}{\rho}) \end{pmatrix}
\]  

where the pressure is given by

\[
p = (\gamma - 1) \rho \left( E - \frac{1}{2} (u^2 + v^2) \right)
\]

and \( \gamma \) is the ratio of specific heats. Application of spatial discretization, such as the finite-volume method, to (15) results in a spatially coupled system of ordinary differentiation equations of the form

\[
\frac{dx_h}{dt} + \frac{1}{\nu_h} \int_{S_h} F \cdot ndS = 0
\]

where \( x_h \) is the volume-averaged solution of \( x \) for each cell, \( \nu_h \) is the cell volume, \( n \) is the surface normal vector for each face, and \( S_h \) is the total cell surface area. Performance measures of interest to an aerodynamicist include the aerodynamic drag coefficient, which is defined as

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

with

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

where \( p_\infty \) is the free stream density of the fluid, and \( v_\infty \) is the free stream velocity, \( A \) is the surface area of the aerofoil, \( p_\infty \) is the free stream pressure, \( \alpha \) is angle of attack, and \( \phi \) is the angle of the surface normal relative to the vertical axis. Note that the discretized Euler equations (18) is of the same form as (11) and the drag coefficient (19) is estimated by (12) defined previously. The drag coefficient is the cost to be optimized.

The initial geometry was taken to be the NACA0012 aerofoil. In the simulation results presented here, Hicks–Henne functions [25] are utilized to deform the aerofoil in the normal direction. The function consists of deformation parameters \( a_i \), which defines the magnitude of the deformation, \( b_i \) is the location of the center of the bump (control point), \( c := 3.0 \) controls the width of the bump, and \( \chi \) is the horizontal coordinate of the aerofoil. The function is defined as

\[
z(\chi) = a_i \left( \sin \left( \pi \frac{\chi}{b_i} \right) \right)^c, \quad 0 \leq \chi \leq 1.
\]

Two control points are located on the upper surface centered at one-third and two-thirds along the chord length and spanning two-thirds of the chord. The other two control points are located on the lower surface at the same chord locations. The input is defined by \( u := [a_1, a_2, a_3, a_4] \).

To illustrate the presence of multiple local minima for these types of aerodynamic problems, the drag coefficient surface, where only two control points on the bottom surface...
of the aerofoil are allowed to vary, is plotted in Fig. 4. The figure shows a Lipschitz continuous surface with a main basin in the middle where the drag is minimal and also a local minimum toward the top right. While the region of attraction of this local minimum is small, one cannot eliminate the possibility of becoming entrapped in one with gradient-based optimizers.

The plot of the pressure coefficient against the chord length (where \( \chi = 0 \) is the leading edge and \( \chi = 1 \) is the trailing edge) for the NACA0012 aerofoil is shown in Fig. 5. The drag coefficient was calculated to be \( C_d = 0.021568 \) and from the plot it can be seen that the main contribution to the pressure drag is due to the stronger shock that developed over the top surface of the aerofoil indicated by the \( C_p \) values along the leading two-thirds of the chord.

To identify the best possible solution and to benchmark the performance of the different approaches, full time scale separation between the plant and optimizer was introduced by waiting for the CFD to fully converge to a residual error tolerance of \( 1 \times 10^{-11} \) before updating the optimizer via the DIRECT algorithm. In addition, the adjoint method was also considered as a best-in-class gradient-based local optimization method in place of the DIRECT method. The aerofoil was initialized to the original shape with the next input parameters determined by using a sequential least-squares optimizer and the gradient at each optimization step provided by the adjoint method. In implementing the global extremum-seeking approach with partial convergence of the CFD, three different waiting times, \( T_{ES} \), were considered. These were 10, 25, and 50 iterations of the CFD numerical solver. Finally, the DIRECT-L optimizer is also tested with 10, 25, and 50 iterations of the CFD numerical solver and utilizing a conservative value of 0.025 for the Lipschitz constant. The input domain of the DIRECT and DIRECT-L algorithms is the set \( u = \{ a \in \mathbb{R}^4 \mid a_i \in [-0.05, 0.05], \ i = 1, 2, \ldots, 4 \} \).

Fig. 6 shows the resulting \( C_p \) values of all the optimization strategies. In all cases, the shock on the upper surface is weakened; however, as there are only four control points, there are structural limits to the shape deformation, and so the \( C_p \) values (and thus the drag) can be altered only to a certain extent. The figure also shows the variation in \( C_p \) values along the lower surface of the aerofoil for each optimization strategy. Note that for the partially converged optimization results, the values in the figure are produced by rerunning the CFD solver until it is fully converged for the optimized shape that was found by the optimizer. This shows the true value of the pressure (and drag) coefficient as opposed to that observed by the optimizer due to early termination.

Fig. 7 shows the shape of all the optimized aerofoils, including the original NACA0012 aerofoil. There is partial asymmetry in the optimized aerofoil shape, primarily a consequence of the nonzero angle of attack. It can be seen that the upper surface of all the optimized aerofoil shapes are practically the same, and the variation in shape is from the lower surface. This is indicative that the cost function with respect to the lower control points is similar to that shown earlier in Fig. 4.
TABLE I
COMPARISON OF OPTIMIZATION RESULTS

| Method      | N (number of CFD iterations) | k (number of samples) | Total time | $C_d$ measured at $T_{ES}$ | $|| u - u^* ||_2$ | True $C_d$ |
|-------------|-----------------------------|-----------------------|------------|-----------------------------|--------------------|-------------|
| Adjoint     | $\rightarrow \infty$       | 60                    | 34m        | n.a.                        | 0.0023             | 0.006697    |
| DIRECT      | $\rightarrow \infty$       | 1809                  | 1 hr 53m   | n.a.                        | 0                  | 0.00511     |
| DIRECT      | 50                          | 463                   | 1 hr 2m    | 0.006672                    | 0.0061             | 0.006750    |
| DIRECT      | 25                          | 369                   | 41m        | 0.006672                    | 0.0083             | 0.006928    |
| DIRECT      | 10                          | 285                   | 27m        | 0.007500                    | 0.0112             | 0.007875    |
| DIRECT-L    | 50                          | 281                   | 37m        | 0.006782                    | 0.0081             | 0.006832    |
| DIRECT-L    | 25                          | 203                   | 23m        | 0.006742                    | 0.0100             | 0.006992    |
| DIRECT-L    | 10                          | 275                   | 26m        | 0.007440                    | 0.0114             | 0.007762    |

Fig. 7. Aerofoil profiles.

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

1) the value of $C_d$ at termination of the optimization, when the CFD has only run for $N$ iterations at the optimal geometry;
2) the norm of the difference between the input parameters of each method compared with the optimal input parameters determined by the fully converged DIRECT optimization;
3) the true value of $C_d$ after running the CFD to convergence for the resulting optimal geometry of each method.

It is clear that the results predicted by Proposition 4 are maintained, i.e., the closeness of the extremum-seeking solution to the global optimum can be controlled through the waiting time. It is also apparent that the proposed global extremum-seeking approach does not converge particularly close to the global optimum when the waiting time is too short, as for $T_{ES} = 10T_{CFD}$.

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

Comparing the results of DIRECT with 10, 25, and 50 iterations and DIRECT-L with the respective iterations, it can be seen that DIRECT-L is able to converge to the same solution; in some cases with over 40% reduction in the number of samples and time taken. While DIRECT-L does reduce the number of samples taken, the main drawback is that knowledge of the Lipschitz constant of the cost function is necessary. In the case presented here, the Lipschitz constant was set arbitrarily but it could also be found or estimated, for example, via methods such as response surface modeling [26]. Also, as the number of control points increases, the difference between the computational effort of the adjoint method and DIRECT will be exacerbated as the regions the DIRECT algorithm is required to explore increase exponentially with input dimension. A major limiting factor is the time taken to reinitialize the mesh following the perturbation of the control input.

IV. CONCLUSION

The framework for the DIRECT-L optimization algorithm was introduced and the conditions under which it could be applied to a CFD solver were stated. The algorithm was then employed to optimize the shape of a 2-D aerofoil to minimize the aerodynamic drag in steady flow, and the results were contrasted against other global extremum-seeking methods and the gradient-based adjoint method. It was found that by utilizing DIRECT-L with a suitably chosen Lipschitz constant the number of samples and time taken could be reduced by up to 40% when compared against the DIRECT algorithm and that the solutions found were still in the vicinity of the true global optimum. The time difference between utilizing global optimization techniques (even with partially converged CFD results and exploiting knowledge of the Lipschitz constant) and local gradient-based approaches remains problematic, at least in the face of high-dimensional optimization requirements.

Potential future directions to explore include the development of hybrid global–local approaches, where a coarse
global search using fewer control points is initially conducted followed by a finer resolution local method, and/or the use of multiagent approaches to global extremum seeking. Methods of estimating the Lipschitz constant of the cost function globally or within each hyper-rectangle could also be used to speed up the DIRECT-L algorithm. There is also a need to consider state constraints in the global optimizer. From a practical standpoint, clear motivation exists in developing faster mesh initialization routines.

ACKNOWLEDGMENT

The authors would like to thank BAE Systems Australia, the Australian Research Council, and the Swedish Research Council through the LCCC Linnaeus Center. They would also like to acknowledge P. Riseborough and J. Vethecan of BAE Systems Australia for their insight and discussion regarding this research.

REFERENCES