Computer Physics Communications 298 (2024) 109122

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Computational Physics

Gradient-enhanced stochastic optimization of high-fidelity simulations [☆]

Alejandro Quirós Rodríguez [a,*], Miguel Fosas de Pando [b], Taraneh Sayadi [a]

[a] Sorbonne Université, Institut Jean Le Rond ∂'Alembert, UMR 7190, 4 Place Jussieu, 75252 Paris Cedex 05, France

[b] Dpto. Ing. Mecánica y Diseño Industrial, Escuela Superior de Ingeniería, Universidad de Cádiz, Av. de la Universidad de Cádiz 10, 11519 Puerto Real, Spain

A R T I C L E I N F O

Keywords:
Stochastic optimization
Surrogate model
Radial basis function
Gradient-enhanced radial basis function
High-fidelity simulation

A B S T R A C T

Optimization and control of complex unsteady flows remains an important challenge due to the large cost of performing a function evaluation, i.e. a full computational fluid dynamics (CFD) simulation. Reducing the number of required function evaluations would help to decrease the computational cost of the overall optimization procedure. In this article, we consider the stochastic derivative-free surrogate-model based Dynamic COordinate search using Response Surfaces (DYCORS) algorithm and propose several enhancements: First, the gradient information is added to the surrogate model to improve its accuracy and enhance the convergence rate of the algorithm. Second, the internal parameters of the radial basis function employed to generate the surrogate model are optimized by minimizing the leave-one-out error in the case of the original algorithm and by using the gradient information in the case of the gradient-enhanced version. We apply the resulting optimization algorithm to the minimization of the total pressure loss through a linear cascade of blades, and we compare the results obtained with the stochastic algorithms at different Reynolds numbers with a gradient-based optimization algorithm. The results show that stochastic optimization outperforms gradient-based optimization even at very low *Re* numbers, and that the proposed gradient-enhanced version improves the convergence rate of the original algorithm. An open-source implementation of the gradient-enhanced version of the algorithm is available.

1. Introduction

Progress in computational capabilities during the past decades have allowed computational fluid dynamics (CFD) to become an ever more present tool in the description and the prediction of complex unsteady flows. However, the computational cost associated with such high-fidelity simulations precludes them from being routinely used in state-of-the-art optimization algorithms, without resorting to reduced order models. Thus, the development of strategies that reduce the number of function evaluations, i.e. CFD simulations, in such optimization algorithms is crucial to achieve an acceptable computational cost.

Optimization algorithms generally fall under two main categories: (i) gradient-based, or (ii) derivative-free methods. Gradient methods rely on the value of local derivatives to identify a descent direction. This derivative is most commonly calculated using analytical expressions or finite differences. Both strategies are inapplicable to high-fidelity simulations: analytical expressions are usually not available and finite difference becomes very expensive in the case of unsteady high-fidelity simulations, and is susceptible to noise. Alternatively, gradient information can be extracted using adjoint-based algorithms, which was pioneered by Pironneau in [1]. Adjoint-based optimization has been widely used in fluid mechanics, from areas dominated by linear dynamics (e.g. acoustics and thermo-acoustics [2,3]), to nonlinear systems (e.g. analysis of high-lift airfoils, mixing enhancement and minimal seeds for transition to turbulence [4–6]). Recently, their application to more complex flow regimes, such as reactive and interfacial flows have also been investigated [7–9]. However, as demonstrated by [7], the objective function encountered in such flows can have multiple minima, rendering the application of gradient methods difficult. In addition, the presence of turbulence makes the gradient-based approach inadmissible in many complex flow scenarios. Derivative-free methods elevate these challenges and have been applied successfully to optimization in fluid mechanics [10,11]. Their main drawback, however, is the requirement for many function evaluations, which proves to be too costly when dealing with cases of practical interest.

[☆] The review of this paper was arranged by Prof. Andrew Hazel.
[*] Corresponding author.
E-mail addresses: alejandro.quiros_rodriguez@etu.sorbonne-universite.fr (A. Quirós Rodríguez), miguel.fosas@uca.es (M. Fosas de Pando), taraneh.sayadi@sorbonne-universite.fr (T. Sayadi).

https://doi.org/10.1016/j.cpc.2024.109122
Received 15 March 2023; Received in revised form 26 October 2023; Accepted 31 January 2024

Available online 6 February 2024

Due to these disadvantages, the application of these methods to optimization problems involving high-fidelity unsteady simulations is not straightforward. A suitable alternative for cases with expensive function evaluations is one that is based on a response surface model (also known as a surrogate model or a meta-model), which is, in essence, an inexpensive approximate model of the underlying expensive function. Performing the optimization procedure on a surrogate surface greatly reduces the number of calls to the expensive high-fidelity model. Surrogate model optimization has been used extensively to identify promising points for function evaluations [12–15] using different interpolation techniques that have been proposed, e.g. Least Squares (LS) [16], Kriging [17], Radial Basis Functions (RBF) [18] and Support Vector Regression (SVR) [19]. The most promising point on the surrogate model can be determined by several techniques, such as the Adaptive Response Surface Method (ARSM) [20], Efficient Global Optimization (EGO) [21] and DYCORS [22], to name a few.

Although surrogate model optimization reduces the number of expensive function calls dramatically, it still suffers from the curse of dimensionality, especially when the number of design variables increases [23]. In addition, typical algorithms still require a large number of function evaluations to be applicable to the problems of practical interest. In order to ameliorate these restrictions, gradient information can be incorporated into the surrogate model. Two main approaches are (i) constructing the surrogate surface using the gradient as well as the local function value [24,25] or (ii) using multiple start algorithms [26,27]. Both approaches show promising results, suggesting that a judicious combination of derivative-free and gradient-based methods can lead to an efficient procedure that converges to the global minimum with a limited number of expensive function evaluations. Gradient-enhanced surrogates have already been used in optimization algorithms in different fields. Gradient-enhanced Kriging is probably the most used gradient-enhanced surrogate model in the computational fluid mechanics community. It has been successfully tested on different problems [28–30]. The benefits of adding the gradient, in this context, have been recently quantified rigorously by [31]. However, as stated in [30], the inversion of the correlation matrix becomes expensive, in terms of memory, as the number of dimensions and sampling points increases and therefore it is not suitable for very high dimensional problems. Gradient-enhanced RBF has also been employed in surrogate-based optimization. Some early examples include the work in [24], where both RBF and gradient-enhanced Kriging are compared to gradient-based algorithms in 6-dimensional problems, showing that gradient-enhanced kriging provides the best results. RBF and gradient-enhanced RBF are also compared in [32] in the context of shape optimization with 24-dimensional control parameters, where Euler equations are used. In [33], a 2-dimensional shape optimization is performed comparing again Kriging and RBF. These studies while informative consider either the optimization of a low-dimensional parameter-space and/or simplified underlying flow equations. Therefore, further analysis on the influence of dimensionality and functions landscape is required to assess the suitability of gradient-enhanced RBFs in the context of surrogate-based optimization of unsteady flow regimes and high-dimensional parameter space. This is the objective of the work presented here.

In this study, the DYCORS algorithm [34] is adopted as the basis of the surrogate model optimization procedure. This algorithm is particularly attractive due to its fast convergence to the global minimum in a high-dimensional parameter space. This characteristic is necessary in applications of interest to CFD, since the control function is most commonly a parametrized/discretized function, distributed in space. To our best knowledge, this work presents one of the first applications of DYCORS to unsteady flows. We aim to provide a measure of its performance at different regimes such as steady, unsteady and non-deterministic flow. In addition, the use of local gradient information is proposed to improve the accuracy of the surrogate model, resulting in a gradient-assisted surrogate model optimization that aims at reducing the number of required function evaluations to reach the global optimum. Moreover, the optimization of the internal parameters of the surrogate model has been included in the optimization procedure to further enhance its accuracy. The resulting optimization algorithm is applied to control the unsteady flow around a linear cascade of compressor rotor blades.

The paper is organized as follows. First, in Section 2 a detailed description of the stochastic optimization algorithm is provided and the enhancements to the original algorithm are highlighted. Then, its performance is assessed in the context of numerical flow simulations. The governing equations and the numerical schemes of the underlying flow solver are briefly presented in Section 3. In Section 4, an application of this algorithm to the reduction of total pressure loss through a linear cascade of blades is presented and the results are discussed. Finally, we provide in Section 5 concluding remarks and suggestions for future work.

2. Optimization framework

The Dynamic Coordinate Search using Response Surfaces (DYCORS) algorithm developed in [34] is first described in this section, and then extended to include derivative information. This algorithm is chosen owing to its performance in a high-dimensional parameter space. Once the algorithm is initialized by evaluating the objective function at selected initial sampling points, it produces a sequence of candidate solutions until a stop criterion is met. At each iteration, the following operations are performed:

- Construction of the surrogate model using information from previously-evaluated points, Fig. 1(a).
- Generation of trial points and evaluation using the surrogate model, Fig. 1(b)
- Selection of best candidate point among the trial points, Fig. 1(c).
- Evaluation of the objective function at the best candidate point, Fig. 1(d).

This procedure is illustrated in Fig. 1, where the one-dimensional Rastrigin function [35], a commonly used function to benchmark algorithms in the presence of a large number of local minima, is considered. These steps will be presented below in more detail.

2.1. *Construction of the surrogate model*

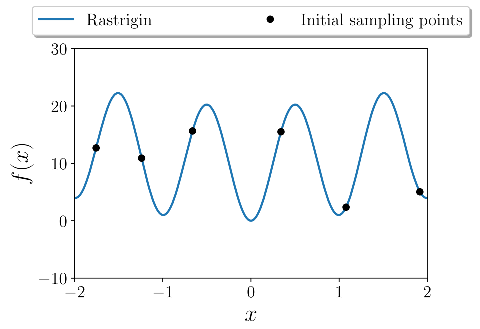
In the following, the interpolation technique underlying the surrogate model in the DYCORS algorithm is briefly discussed. This procedure is then modified to include gradient information and a new criterion is introduced to determine the internal parameters of the interpolant.

2.1.1. *Radial basis function interpolation*

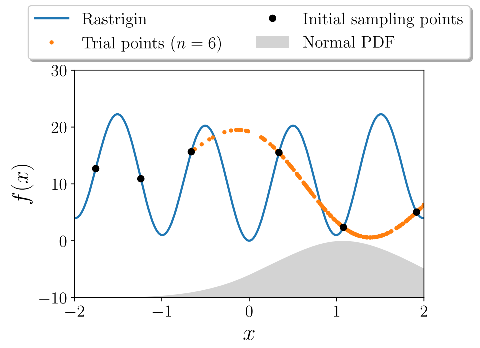
The DYCORS algorithm relies on Radial Basis Functions (RBF) to build the surrogate model. Consider an objective function $f : \mathbb{R}^d \to \mathbb{R}$, where d is the number of parameters. Taking a set of n points in the parameter space $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ and the corresponding values of the objective function $f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)$, the value of the objective function at a point $\mathbf{y}$ can be approximated by the RBF interpolant [18]

$$s_n(\mathbf{y}; \mathbf{x}, \lambda, \mathbf{l}) = \sum_{i=1}^{n} \lambda_i \phi\left(r(\mathbf{y}, \mathbf{x}_i, \mathbf{l})\right), \tag{1}$$

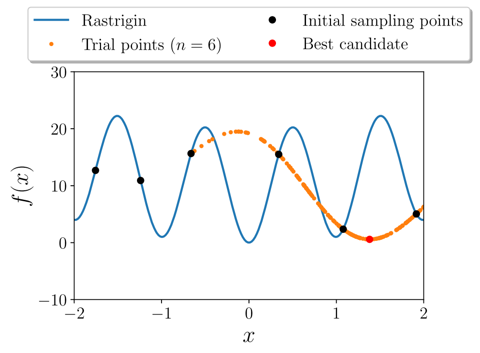
where $\phi(\cdot)$ is a kernel function, λ is a vector containing the coefficients of the interpolant and $r(\mathbf{r}_p, \mathbf{r}_c, \mathbf{l}) = \left\| \operatorname{diag}(\mathbf{l}^{-1}) \left(\mathbf{r}_p - \mathbf{r}_c\right) \right\|$ where $\|\cdot\|$ is the Euclidean norm, $\mathbf{r}_p$ is the point where the radial basis function is going to be evaluated, $\mathbf{r}_c$ is the center of the radial function and $\mathbf{l} \in \mathbb{R}^d$ is a vector of internal parameters corresponding to the spatial length-scale of the kernel function in each parameter direction. A wide variety of kernel functions exist, and some of the most popular choices, e.g. the



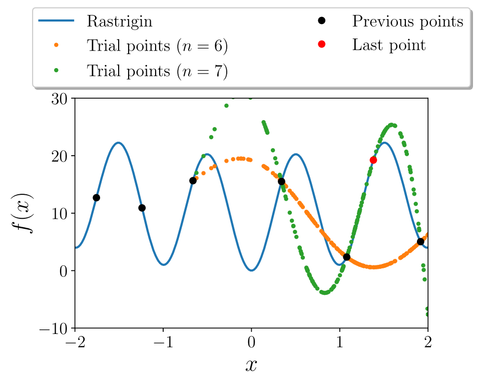
(a) The algorithm is initialized with $m = 6$ initial sampling points distributed through the domain by means of a Latin Hypercube Sampling technique. The objective function is evaluated at these points.



(b) The surrogate model is built using previously evaluated points. This surrogate model is then evaluated at $k = 200$ trial points (shown in orange) drawn from a normal distribution centered at current best point (PDF shown in gray).



(c) The most promising point (shown in red) is selected based on a given criteria.



(d) The objective function is evaluated at the best candidate point and the surrogate model is updated. In this case, the value of the objective function at the best candidate point does not improve the previous best, and the normal distribution used to generate the new trial points is centered at the same location as the previous iteration.

Fig. 1. Illustration of the main steps performed during one iteration of DYCORS algorithm. The one-dimensional Rastrigin function is used as the objective function defined on the domain $x \in [-2, 2]$.

Table 1
Kernel functions, with r being a positive scalar denoting the distance between a point and the center of the radial basis function and ν an internal parameter of the Matérn kernel referring to the order of the modified Bessel function K_ν.

Function	Expression
Exponential	$\phi(r) = \exp\left(-\frac{r^2}{2}\right)$
Matérn	$\phi(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}\lvert r\rvert\right)^{\nu} K_\nu\left(\sqrt{2\nu}\lvert r\rvert\right)$
Cubic	$\phi(r) = r^3$

exponential, the Matérn [36] and the cubic kernels, are presented in Table 1.

The weights λ are determined by setting the value of the interpolant to that of the objective function at every $\mathbf{x}_i$, i.e. $s_n(\mathbf{x}_i; \mathbf{x}, \lambda, \mathbf{l}) = f(\mathbf{x}_i)$. However, depending on the kernel choice, the resulting system of equations can be conditionally positive definite [37]. The interpolant given in Eq. (1) is then modified and polynomials p of degree up to m in d unknowns, i.e. $p \in \Pi_m^d$ are added to the right-hand side; see [37] for further details. We set $m = 1$ for all the kernels, following [34]. The RBF interpolant then reads

$$s_n(\mathbf{y}; \mathbf{x}, \lambda, \mathbf{l}, \mathbf{c}) = \sum_{i=1}^{n} \lambda_i \phi(r(\mathbf{y}, \mathbf{x}_i, \mathbf{l})) + p(\mathbf{y}, \mathbf{c}), \tag{2}$$

where $\mathbf{c} = [c^1, \ldots, c^{d+1}]^T$ is the vector containing the coefficients of the polynomials. To uniquely determine these coefficients, the above system of equations is augmented by enforcing orthogonality between the coefficients of the kernel functions and the polynomial space Π_m^d, i.e.

$$\sum_{i=1}^{n} \lambda_i q_i^j = 0, \quad \text{for } j = 1, \ldots, d+1, \tag{3}$$

where $q_i^1 = 1$ and $q_i^j = x_i^{j-1}$. Finally, the coefficients λ and $\mathbf{c}$ are determined by the following linear system

$$\begin{pmatrix} \mathbf{\Phi} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \lambda \\ \mathbf{c} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{0} \end{pmatrix}, \tag{4}$$

where, $\Phi_{ij} = \phi(r(\mathbf{x}_i, \mathbf{x}_j, \mathbf{l}))$ for $i, j = 1, \ldots, n$ denotes the kernel matrix, $\mathbf{P}_i = [1, x_i^1, \ldots, x_i^d]$ for $i = 1, \ldots, n$ is the polynomial matrix, and $\mathrm{f}_i = f(\mathbf{x}_i)$ for $i = 1, \ldots, n$ is a vector that contains the function values at the evaluated points.

2.1.2. Gradient-enhanced radial basis function interpolation

We now turn the attention to Gradient-enhanced Radial Basis Functions (GRBF). The surrogate model can be improved by including local gradient information such that both the function f and its gradient $\mathbf{g}$ are matched at the evaluated points. With a more accurate surrogate model, the evaluation of the trial points should provide function values closer to the exact values, thereby, improving the convergence rate of the algorithm. In this case, additional basis functions are introduced

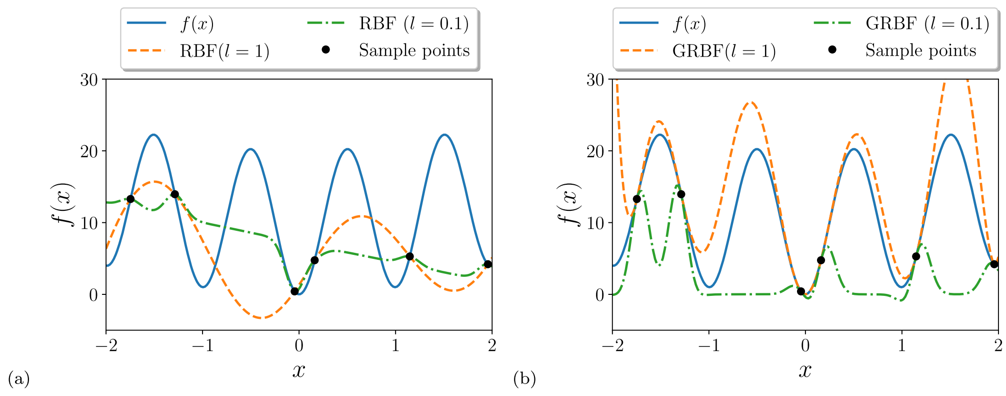


Fig. 2. Interpolation of the one-dimensional Rastrigin function (blue solid line) using 6 sample points (black points) (a) RBF with an exponential kernel and internal parameters $l = 1.0$ (orange dashed line) and $l = 0.1$ (green dash dotted line), (b) GRBF with an exponential kernel and internal parameters $l = 1.0$ (orange dashed line) and $l = 0.1$ (green dash dotted line). (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

to include the local gradient information into the surrogate model. Following [38,39,33], the interpolation now reads

$$s_n(\mathbf{y};\mathbf{x},\lambda,\mathbf{l},\mathbf{c}) = \sum_{i=1}^{n} \lambda_i \phi(r(\mathbf{y},\mathbf{x}_i,\mathbf{l})) + \sum_{j=1}^{d}\sum_{i=1}^{n} c_i^j \frac{\partial \phi}{\partial r_p^j}\Big|_{r(\mathbf{y},\mathbf{x}_i,\mathbf{l})}, \tag{5}$$

where the polynomial term in Eq. (2) has been replaced by a term containing the first derivative of the kernel. Note that the size of the vector of coefficients $\mathbf{c}$ is now dependent on the number of evaluated points with a dimension of nd, thus, an additional set of equations has to be included to uniquely determine the coefficients. To this end, we differentiate Eq. (5)

$$\frac{\partial s_n}{\partial r_c^k}\Big|_{(\mathbf{y};\mathbf{x},\lambda,\mathbf{l},\mathbf{c})} = \sum_{i=1}^{n} \lambda_i \frac{\partial \phi}{\partial r_c^k}\Big|_{r(\mathbf{y},\mathbf{x}_i,\mathbf{l})} + \sum_{j=1}^{d}\sum_{i=1}^{n} c_i^j \frac{\partial^2 \phi}{\partial r_p^j \partial r_c^k}\Big|_{r(\mathbf{y},\mathbf{x}_i,\mathbf{l})}, \tag{6}$$

where both the first and the second derivatives of the kernel function appear. Using Eqs. (5) and (6), the coefficients λ_i and c_i^j are determined by the solution of the following linear system

$$\begin{pmatrix} \mathbf{\Phi} & -\mathbf{\Phi}_d \\ \mathbf{\Phi}_d^T & \mathbf{\Phi}_{dd} \end{pmatrix} \begin{pmatrix} \lambda \\ \mathbf{c} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix}, \tag{7}$$

where $\mathbf{g}_i = [g(\mathbf{x}_i), \dots, g(\mathbf{x}_i)]^T$, $i = 1, \dots, n$, is the vector with the derivatives of f at the evaluated points. This system is analogous to Eq. (4) and is guaranteed to be positive definite [39]. The matrix with the polynomial terms and the zero matrix of the original RBF formulation have been replaced by the first and the second order derivatives of the kernel matrix $\mathbf{\Phi}_d$ and $\mathbf{\Phi}_{dd}$, respectively. The derivatives of the kernel matrix can be computed via chain rule,

$$\Phi_{d_{ij,k}} = \frac{\partial \phi}{\partial r_c^k}\Big|_{r(\mathbf{x}_i,\mathbf{x}_j,\mathbf{l})} = \frac{\partial \phi}{\partial r}\Big|_{r(\mathbf{x}_i,\mathbf{x}_j,\mathbf{l})} \frac{\partial r}{\partial r_c^k}\Big|_{(\mathbf{x}_i,\mathbf{x}_j,\mathbf{l})}, \tag{8}$$

$$\begin{aligned}\Phi_{dd_{ij,kl}} &= \frac{\partial^2 \phi}{\partial r_c^k \partial r_p^l}\Big|_{r(\mathbf{x}_i,\mathbf{x}_j,\mathbf{l})} \\ &= \frac{\partial^2 \phi}{\partial r^2}\Big|_{r(\mathbf{x}_i,\mathbf{x}_j,\mathbf{l})} \frac{\partial r}{\partial r_c^k}\Big|_{(\mathbf{x}_i,\mathbf{x}_j,\mathbf{l})} \frac{\partial r}{\partial r_p^l}\Big|_{(\mathbf{x}_i,\mathbf{x}_j,\mathbf{l})} + \frac{\partial \phi}{\partial r}\Big|_{r(\mathbf{x}_i,\mathbf{x}_j,\mathbf{l})} \frac{\partial^2 r}{\partial r_c^k \partial r_p^l}\Big|_{(\mathbf{x}_i,\mathbf{x}_j,\mathbf{l})}.\end{aligned} \tag{9}$$

The first derivative of the kernel matrix $\mathbf{\Phi}_d$ has a dimension of $nd \times n$ while the second derivative of the kernel matrix $\mathbf{\Phi}_{dd}$ has a dimension of $nd \times nd$. They can be constructed according to

$$\mathbf{\Phi}_d = \begin{pmatrix} \Phi_{d_{11,1}} & \cdots & \Phi_{d_{11,d}} & \cdots & \Phi_{d_{1n,d}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \Phi_{d_{n1,1}} & \cdots & \Phi_{d_{n1,d}} & \cdots & \Phi_{d_{nn,d}} \end{pmatrix}, \tag{10}$$

$$\mathbf{\Phi}_{dd} = \begin{pmatrix} \Phi_{dd_{11,11}} & \cdots & \Phi_{dd_{11,1d}} & \cdots & \Phi_{dd_{1n,1d}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \Phi_{dd_{11,d1}} & \cdots & \Phi_{dd_{11,dd}} & \cdots & \Phi_{dd_{1n,dd}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \Phi_{dd_{n1,d1}} & \cdots & \Phi_{dd_{n1,dd}} & \cdots & \Phi_{dd_{nn,dd}} \end{pmatrix}. \tag{11}$$

2.1.3. *Comparison of the interpolants constructed using RBF and GRBF*

In this section, we provide a comparison between RBF and GRBF models. We consider the Rastrigin function, given by $f(\mathbf{x}) = 10d + \sum_{i=1}^{d}[x_i^2 - 10\cos(2\pi x_i)]$, where d is the number of dimensions of the input vector. To motivate the optimization of the internal parameters, surrogate models with different choices of internal parameters are considered to highlight their effect in the approximation accuracy.

In the one-dimensional case, Fig. 2, interpolants based on the exponential kernel at six sample points and two different values of the internal parameter, $l = 1.0$ and $l = 0.1$, are considered. Fig. 2(a) presents the results using RBF whereas Fig. 2(b) shows the results using GRBF. This figure illustrates the effect of including gradient information in the surrogate model for different values of the internal parameter l. The interpolation achieved using GRBF with $l = 1$ shows a considerable improvement with respect to the one given by RBF with the same value of l. Moreover, in both cases a large difference can be observed between the interpolation obtained with $l = 1$ and that with $l = 0.1$. These results suggest that the accurate construction of the surrogate using GRBF is highly dependent on the value of the internal parameter, otherwise adding the gradient information does not lead to a considerable improvement of the resulting interpolant.

2.1.4. *Optimization of the internal parameters*

As shown in the previous section, the value of the internal parameter of the kernel function must be properly set to reach optimal performance in RBF and GRBF surrogate models. This can be achieved by optimizing the leave-one-out error e_{loo} in the case of RBF as described in [40]. In the case of GRBF, a different approach must be followed as we will show in this section. In this work, we make use of an efficient implementation of the leave-one-out error from [33], where the internal parameter is determined by the solution of the following optimization problem,

$$\min e_{\text{loo}}(\mathbf{l},\nu), \qquad e_{\text{loo}}(\mathbf{l},\nu) = \frac{\mathbf{a}^T \mathbf{H}(\mathbf{l},\nu)^{-2}\mathbf{a}}{n\,\text{diag}(\mathbf{H}(\mathbf{l},\nu)^{-2})} \qquad \text{s.t.} \qquad \kappa(\mathbf{H}) < \frac{1}{10\epsilon}, \tag{12}$$

where e_{loo} is the leave-one-out error, n is the number of evaluated points, vector $\mathbf{a}$ contains the values of the function at the evaluated points in the case of RBF and the values of the function and its gradient in the case of GRBF, κ is the condition number of a matrix, ϵ is the machine precision, and $\mathbf{H}$ can be defined as

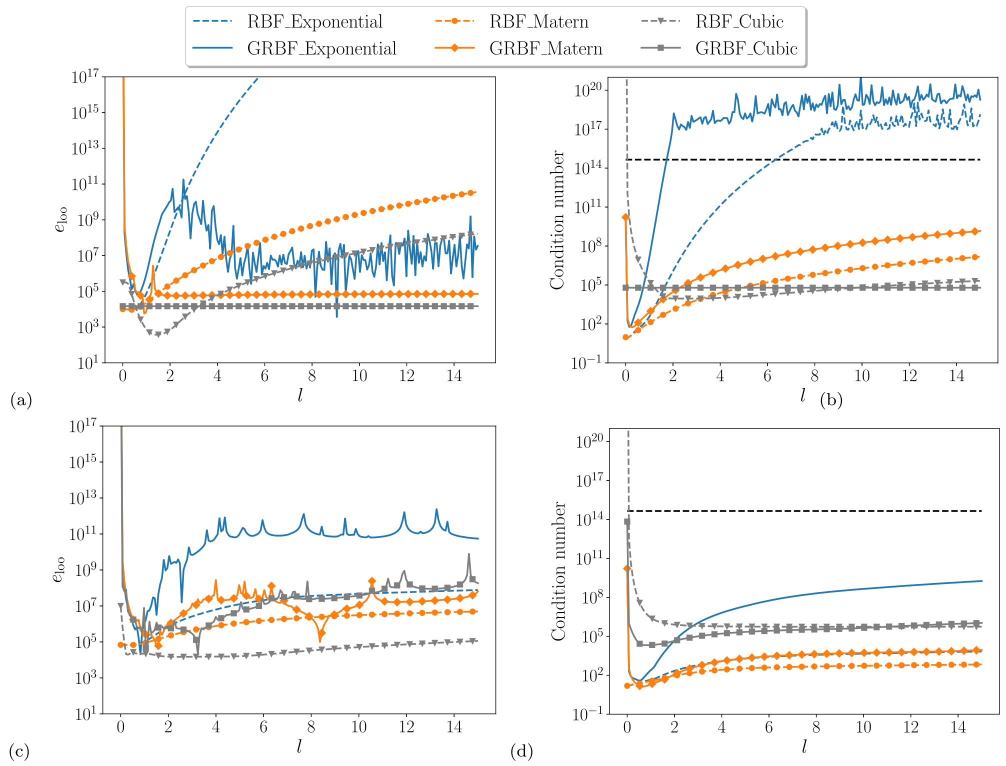


Fig. 3. Leave-one-out error as function of the internal parameter for (a) one-dimensional Rastrigin function and (c) two-dimensional Rastrigin function and condition number of the full kernel matrix for (b) one-dimensional Rastrigin function and (d) two-dimensional Rastrigin function. Results for the exponential kernel are plotted in blue (dashed line RBF and solid line GRFB), for the Matérn kernel are plotted in orange (dashed line with circles RBF and solid line with diamonds GRBF) and for the cubic kernel are plotted in grey (dashed line with triangles RBF and solid line with squares GRBF). The dashed black line represents the constraint on the condition number with a value of $1/(10\epsilon)$.

$$\mathbf{H} = \mathbf{\Phi}, \tag{13}$$

in the case of RBF, and as

$$\mathbf{H} = \begin{pmatrix} \mathbf{\Phi} & -\mathbf{\Phi}_d \\ \mathbf{\Phi}_d^T & \mathbf{\Phi}_{dd} \end{pmatrix}, \tag{14}$$

in the case of GRBF. The constraint on the condition number of the full kernel matrix $\mathbf{H}$ has been added to the optimization to ensure the smoothness of the surrogate model. The maximum value for the condition number is set to $1/10\epsilon$, where ϵ is the machine precision.

Fig. 3 displays the leave-one-out error and the condition number as a function of the internal parameter for two different cases and three kernel functions. Figs. 3(a,b) show the results for the one-dimensional Rastrigin function evaluated at 10 points and Figs. 3(c,d) present the results for the two-dimensional Rastrigin function evaluated at 20 points. In the latter, the internal parameter is kept constant in one direction and varies in the other. As it can be seen, the leave-one-out error presents a smooth behaviour for the RBF kernels when the constraint is satisfied, however when applied to GRBF kernels, the figure shows several peaks even though the condition number is below the constraint. In view of this, an optimal value for the internal parameter cannot be obtained through the optimization of the leave-one-out error in the case of GRBF surrogates. To circumvent this limitation, we propose instead to set the internal parameter l to the inverse of the average absolute value of the derivatives in each direction obtained during the previous iterations of the optimization procedure when a gradient-enhanced kernel is employed. Directions with steeper derivatives are expected to feature smaller spatial scales, and therefore, the widths of the kernel can be reduced accordingly to approximate the objective function more accurately. Eq. (15) gives the expression used to compute the value of the internal parameter in this case,

$$\mathbf{l} = \left\{ \frac{1}{[|\overline{\mathbf{g}}|(x^1)|, \dots, |\overline{\mathbf{g}}(x^d)|]} \right\}. \tag{15}$$

2.2. Stochastic search algorithm

In this section, we provide a description of the steps that are carried out to perform an optimization using the DYCORS algorithm.

DYCORS is a derivative-free stochastic optimization algorithm adapted to the optimization of bound constrained high-dimensional expensive black-box functions. If more complex constraints are needed, they can be added as a penalization term directly into the objective function f. It was developed as a modification of the Local Metric Stochastic Response Surface (LMSRS) method [34] by introducing ideas from the Dynamically Dimensioned Search (DDS) method [41]. In its original form, the algorithm does not rely on the gradient of the objective function to reach a minimum and therefore has no information on the shape of the objective function apart from its value for a given set of control parameters. The algorithm is detailed in Algorithm 1 as well as Algorithms 2-4 given in Appendix C. The main steps of the algorithm are described below:

1-Initialization: The algorithm performs a fixed number of function evaluations $N_{\max}$. It is initialized by evaluating the objective function f defined on the hypercube $\mathcal{D} = [a_d, b_d] \subseteq \mathbb{R}^d$ at a number of m given initial sampling points $\mathcal{I}$. The initial sampling points can be generated by means of Latin Hypercube Sampling techniques. This method creates an optimal distribution through the full hypercube [42]. In this study, an enhanced Latin Hypercube Sampling based on [43] is used to generate the initial sampling points $\mathcal{I}$. This method ensures that the minimum distance between the points is $d_{opt} = m/\sqrt[d]{m}$, and that each region of the hypercube has an equal representation on $\mathcal{I}$.

2-Construction of the surrogate model: At every iteration, a surrogate model is built following the procedure discussed in Section 2.1. The

Algorithm 1: (G)-DYCORS algorithm.

```
Input: Real valued black-box function, $f$ defined on $\mathcal{D} = [a_d, b_d] \subseteq \mathbb{R}^d$
       Real valued black-box function, $g$ defined on $\mathcal{D} = [a_d, b_d] \subseteq \mathbb{R}^d$
       in case of G-DYCORS
       Maximum number of function evaluations, $N_{\max}$
       Initial and minimum standard deviations, $\sigma_0$ and $\sigma_m$
       Number of trial points, $k$
       Response surface model, $\phi$
       Interpolant, $s_n$
       Internal parameter of the kernel, $\mathbf{l}$
       Initial sampling points, $\mathcal{I} = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$
       Weight pattern, $\Upsilon = \{\Upsilon_0, \Upsilon_1, \Upsilon_2, \Upsilon_3\}$
       Limits for number of consecutive failed and successful iterations,
       $\tau_f$ and $\tau_s$
       Number of iterations without optimizing the internal parameter,
       $n_{ip}$

Result: Best point encountered, $x_{best}$

Initialize algorithm: $\mathcal{A}_m = \mathcal{I}, f(\mathbf{x}), (g(\mathbf{x})) : \mathbf{x} \in \mathcal{A}_m$
Select best evaluated point: $f_{best} = f(\mathbf{x}_{best})$
Initialize standard deviation and counters: $\sigma_n = \sigma_0,\ n = m,\ C_f = 0$ and
  $C_s = 0$
while $n < N_{\max}$ do
    Construct the surrogate model: Compute $\lambda$ and $\mathbf{c}$ following
      Sections 2.1.1 and 2.1.2
    Generate and evaluate trial points: Algorithm 2:
      trial_points($n, k, \sigma_n, s_n, \mathcal{A}_n, \lambda, \mathbf{c}$)
    Select best candidate point: Algorithm 3:
      select_next_point($n, \Upsilon, \mathcal{A}_n, \mathbf{y}_{n,j}, s_n(\mathbf{y}_{n,j})$)
    Evaluate function (and gradient): Compute $f(\mathbf{x}_{n+1})$ , $(g(\mathbf{x}_{n+1}))$
    Update information: Algorithm 4:
      update_info($n, \mathbf{x}_{best}, f_{best}, \mathbf{x}_{n+1}, f_{n+1}, C_f, C_s, \tau_f, \tau_s, \sigma_n, \mathcal{A}_n$)
    Optimize internal parameters: following Section 2.1.4
end
```

coefficients of the interpolant λ and $\mathbf{c}$ are given by the solution of the linear systems in Eq. (4) (RBF case) or (7) (GRBF case).

3-Generation of trial points and evaluation using the surrogate model: Following Algorithm 2, the trial points are generated by perturbing the location of the evaluated point with the minimum function value in randomly selected directions. As the optimization procedure advances, the probability of perturbing a direction is reduced according to

$$\varphi(n) = \varphi_0 \left(1 - \frac{\ln(n - m + 1)}{\ln(N_{\max} - m)} \right), \tag{16}$$

where n is the number of function evaluations that have already been performed, m is the size of the initial set of points, N_{max} is the total number of function evaluations to be performed and φ_0 is a constant that will be defined later. Once the perturbed coordinates have been selected, k trial points are generated by means of a normal distribution centered at the current minimum valued point with standard deviation σ_n. Due to the low computational cost of evaluating the trial points using the surrogate model, thousands of evaluations can be performed at a negligible cost. The value of the standard deviation varies depending on the number of consecutive failed or successful iterations, where a failed iteration means that the minimum valued point has not changed in the last iteration and a successful iteration means the algorithm has been able to improve the minimum. The initial value of the standard deviation is set to 0.2 times the distance between boundaries of the hypercube in every direction. If τ_f consecutive failed iterations are performed, the standard deviation is divided by 2. In case τ_s consecutive successful iterations are carried out, the standard deviation is multiplied by 2. If the standard deviation falls below a given threshold σ_m, the algorithm is completely reinitialized to escape from local minima, by keeping just the information of the best evaluated point so far. Once the trial points have been generated, the surrogate model is evaluated at these points using Eq. (2).

4-Selection of best candidate point among the trial points: In order to select the next point that will be evaluated using function f, we have to apply a selection criteria to the trial points. Algorithm 3 provides the steps that are required to select this point. Using this selection criteria two different scores are given to each trial point. On the one hand, the first score (RBF score) takes into account the value of the surrogate model at the trial points, where the lowest value will get the best score. On the other hand, the second score (distance score) takes into account the distance between each trial point and all the already evaluated points, where the higher distances get better scores. The two scores are summed and the trial point with the best overall score is chosen as next point to be evaluated. Depending on the number of the current iteration, one of the scores may be given a greater weight in the overall score. The weight for the first score is rolled through the values $\Upsilon = \{\Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4\}$ whereas the weights for the second score are one minus the value of the first score. By employing these scores, we ensure that different regions of the hypercube are populated, a mandatory criterion to avoid problems with singular matrices when building the surrogate model. This way of proceeding also helps to escape from local minima.

5-Evaluation of the objective function at the best candidate point: After selecting the best candidate point, the objective function (and its gradient in the gradient-enhanced case) is evaluated using the CFD solver. This is the most expensive step in the whole procedure as it requires to perform a full CFD simulation.

6-Update information: After evaluating the objective function, depending on the value obtained after, the counters that keep track of the consecutive failed and successful iterations can either be increased by one or set to zero, C_f and C_s respectively. If they reach the values τ_f or τ_s, respectively, the value of the standard deviation used to generate the trial points σ_n is modified accordingly. Afterwards, the set of evaluated points $\mathcal{A}_n$ and the iteration number n are updated. These steps are indicated in Algorithm 4.

7-Optimization of the internal parameters: Following Section 2.1.4, the internal parameters of the kernel function are optimized to improve the accuracy of the surrogate model. Every n_{ip} iterations of the algorithm, a differential evolution optimization algorithm is employed to optimize the values according to the leave-one-error [40]. This step was not present in the original DYCORS algorithm.

Table 2 presents a summary of all the parameters used in the DYCORS algorithm, defined in [22]. The number of initial points m is fixed to $m = d + 1$ to ensure that singular matrices do not appear when building the RBF, although a higher value may be employed. The value of φ_0 is set such that in the first iteration of low-dimensional optimization problems ($d < 20$) all the coordinates are perturbed, whereas for higher dimensional problems, on average 20 coordinates are perturbed at a time. The justification for this value of φ_0 is that the probability of improving the solution is increased if only a small amount of the variables are perturbed even at the beginning of the optimization procedure. The minimum standard deviation σ_m allows the reduction of the standard deviation up to 6 times before the algorithm is restarted to ensure that local minima are skipped. The weight pattern Υ starts with a value that gives more importance to the distance score and progressively increases the importance of the RBF score in the overall score.

2.3. Influence of dimensionality on the performance of the algorithms

In this section, the performance of the two versions of the optimization algorithm is assessed by applying the DYCORS and G-DYCORS algorithms to the same functions, using the same kernel with the same values of the internal parameters in order to analyze the influence of increasing dimensionality.

Table 2
DYCORS parameters.

Parameter	Description	Value
m	Number of initial sampling points	$d+1$
k	Number of trial points to be generated	$\min(100d, 5000)$
φ_0	Initial probability of perturbing a direction	$\min(20/d, 1)$
σ_0	Initial standard deviation	$0.2(b_d - a_d)$
σ_m	Minimum standard deviation	$0.2/2^6(b_d - a_d)$
τ_s	Maximum number of consecutive successful iterations	3
τ_f	Maximum number of consecutive failed iterations	5
$\boldsymbol{\Upsilon}$	Weight pattern in the score of the trial points	$\{0.3, 0.5, 0.8, 0.95\}$

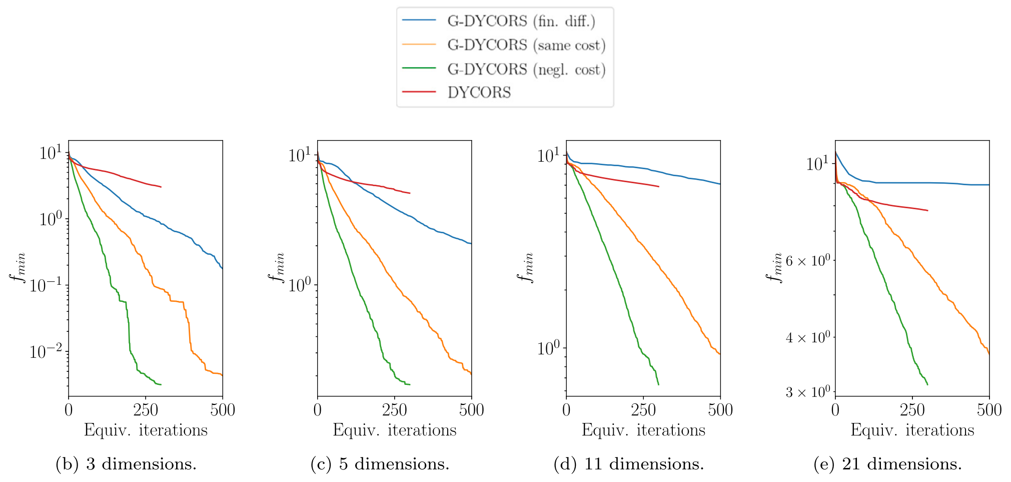


Fig. 4. Performance on the n-dimensional Ackley function. The red line is obtained using the DYCORS algorithm and the rest with the G-DYCORS algorithm, considering negligible cost of the gradient with respect to a function evaluation (green line), same cost (orange line) and much larger cost as if it was computed using finite difference (blue line).

Two different kinds of test functions are considered, corresponding to two different scenarios that may appear in fluid mechanics. The first function is the n-dimensional Rosenbrok function, which is defined as

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (1-x_i)^2\right], \quad \forall\, \mathbf{x} \in \mathbb{R}^n. \tag{17}$$

The global minimum of this function is located at $\mathbf{x} = [1, \dots, 1]$ and for $n > 4$ some local minima appear. Non-chaotic flows at low Reynolds number are expected to have a small number of local minima, which do not vary significantly as the number of design parameters increase, making the Rosenbrok function a good representative.

Chaotic flows, on the other hand, are expected to present a large number of local minima, which increase as new parameters are added. The Rastrigin and the Ackley function are selected as representative of such flows. The Ackley function is defined as

$$f(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right) + 20 + \exp(1), \quad \forall\, \mathbf{x} \in \mathbb{R}^n. \tag{18}$$

This function has the global minimum at $\mathbf{x} = [0, \dots, 0]$ and a local minimum every length unit in every direction, so the number of local minima increases exponentially with the number of dimensions, similar to the Rastrigin function.

To compare the two algorithms, three different cases are considered. In the first case, the computational cost of computing the gradient is supposed to be negligible with respect to the cost of computing the function value. In the second case, the cost of extracting the gradient is the same as that of the evaluating the function. Finally, in the third case, the cost of computing the gradient is assumed to be n times the cost of computing the function value, as if it was computed using finite differences. These three cases allow us to cover the different kind of scenarios that may arise, when numerically optimising flow problems. Therefore, we will be able to factor in the computational cost of evaluating the gradient.

Fig. 4 shows a comparison of the performance of the DYCORS and G-DYCORS algorithms for the Ackley function. The optimization is performed 200 times for the 3-dimensional function, 100 times for the 5-dimensional function, 50 times for the 11-dimensional function and 25 times for the 21-dimensional function, since less variability is observed between different runs of the algorithm as the number of dimensions are increased. The average function value is represented at each equivalent iteration. Since the behaviour observed in the three functions is very similar, only the results of the Ackley function are shown here. Lack of variability across different functions shows the robustness of the algorithm with respect to different flow regimes. This is a considerable advantage when applied to unsteady problems, since the presence of actuation could easily alter the flow characteristics.

The initial sampling of G-DYCORS, in cases where the cost of extracting the gradient is not negligible, is more costly since the initial number of sampling points is kept the same for all algorithms. However Fig. 4 shows that, even using inefficient methods for the computation of the gradient, leads to better convergence of the algorithm for small number of control parameters. As the dimensions of the problem increase, the size of the initial sampling increases as well and the performance of the gradient-enhanced algorithm deteriorates, and the derivative-free version is the best for small number of iterations. Afterwards, the better interpolation of the gradient-enhanced version improves its convergence rate and it outperforms the derivative-free version. In cases where the cost of extracting the gradient is the same as the function evaluation, the gradient-enhanced version always outperforms the original DYCORS. These results suggest that it is advantageous to use the gradient-enhanced version of the optimization algorithm provided that the gradient can be computed efficiently.

3. Governing equations

The flow solver IBMOS (Immersed Boundary Method for Optimization and Stability analysis) [44] employed in this study implements the projection-based immersed boundary method from [45] for two-dimensional flows. The governing equations in continuous form

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2\mathbf{u} + \int_{C} \mathbf{f}(s,\mathbf{x})\delta(\hat{\mathbf{x}} - \xi(s))\,\mathrm{d}s, \tag{19}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{20}$$

and

$$\mathbf{u}[\xi(s)] = \int_{\mathcal{M}} \mathbf{u}(\hat{\mathbf{x}})\delta(\xi(s) - \hat{\mathbf{x}})\,\mathrm{d}\hat{\mathbf{x}} = \mathbf{u}_B(s), s\in C, \tag{21}$$

are solved on a given domain $\mathcal{M}$ together with suitable initial and boundary conditions. In the above, $\hat{\mathbf{x}}\in\mathcal{M}, \mathbf{u}$, p, $\mathbf{f}(s,\mathbf{x}), \mathbf{x}$ and Re are, respectively, the velocity vector, the pressure, the distributed momentum sources along the boundaries of the solids C, the set of control parameters that define the boundary force when an actuation wants to be applied on the surface, and the Reynolds number. The pressure p and the boundary force $\mathbf{f}(s,\mathbf{x})$ can be regarded as a set of Lagrange multipliers that enforce the incompressibility constraint and the no-slip boundary condition or the actuation on C, respectively. A staggered-mesh finite-volume formulation is used to discretize Eqs. (19)-(21) using the implicit Crank-Nicolson integration method for the viscous terms and the explicit second-order Adams-Bashforth scheme for the advection terms. The integrals that involve the δ function are discretized using the mollified δ function from [46]. The resulting discretized governing equations then are

$$\begin{pmatrix} \mathsf{A} & \mathsf{Q} \\ \mathsf{Q}^T & \mathsf{0} \end{pmatrix}\begin{pmatrix} \mathbf{q}^{k+1} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathsf{B}\mathbf{q}^k - \frac{3}{2}\mathcal{N}(\mathbf{q}^k) + \frac{1}{2}\mathcal{N}(\mathbf{q}^{k-1}) + \mathbf{bc}_1 \\ \mathbf{r}_2 \end{pmatrix}, \tag{22}$$

or in compact form

$$R(\mathbf{q}^{k-2}, \mathbf{q}^{k-1}, \mathbf{q}^k, \mathbf{x}) = 0. \tag{23}$$

In the above, $\mathbf{q}^k$ and λ are the flow field at a given time step and the Lagrange multipliers. The reader is referred to [45] for further details regarding the various definitions of the matrices A, Q and B, the non-linear function $\mathcal{N}(\cdot)$ and the vectors $\mathbf{bc}_1$ and $\mathbf{r}_2$.

3.1. Gradient computation

As mentioned in the introduction, in the context of gradient-based optimization there are three different ways of computing the gradient. The cheapest way of computing the gradient is by means of analytical solutions, which are usually inexpensive to compute. In the context of computational fluid dynamics, analytical solutions are not readily available. The most common numerical alternative is the use of finite differences, which require a function call per design parameter and quickly becomes very expensive as the number of control variables increases. An efficient alternative of evaluating the gradient is the use of adjoint methods, which require solving a system similar to the one solved for the objective function. The computational cost of extracting the gradient using this alternative varies depending on the governing equations, called forward problem in the context of adjoint methods. For steady equations, the cost of solving the adjoint system is usually smaller than the cost of the forward problem, whereas for unsteady problems the cost tends to be roughly the same. The reader is referred to [47] for further details on the adjoint method.

Implementing the adjoint system is usually more complicated than implementing the forward system, and in some cases, it might be virtually impossible. The numerical solver IBMOS (used in this study) allows access to adjoint equations for extraction of the gradients (see Appendix A for a more detailed description of the adjoint system corresponding to the unsteady Navier-Stokes equations).

Most black box algorithms do not give access to adjoint-equations, however, since a surrogate is available, there would be possibilities to compute an approximate gradient in an efficient way (using Neural Networks and back-propagation [48], for example). The accuracy needed in computing the gradient for this method to remain attractive (when applied in the black box sense) is yet to be determined.

Table 3
Grid parameters.

Reynolds	$[x_{\min}, x_{\max}]\times[y_{\min}, y_{\max}]$	$\Delta x_{\min}$	$\Delta x_{\max}$	Δy
800	$[-4.43, 4.62]\times[-0.3, 0.3]$	0.006	0.04	0.006
2000	$[-4.43, 4.62]\times[-0.3, 0.3]$	0.006	0.04	0.006
4000	$[-4.72, 4.69]\times[-0.3, 0.3]$	0.0045	0.03	0.0045

4. Results

In this section, we first provide a description of the test cases that are employed in the optimization problem. Afterwards, the objective function and the control parameters are presented. Finally, the results given by the different optimization algorithms are discussed and compared.

4.1. Problem description

The flow around a linear cascade consisting of five blades is used to assess the effectiveness of the stochastic optimization algorithm described in Section 2. The chosen blade profile was developed in [49] and its aerodynamic characteristics have been extensively investigated experimentally and numerically [50–52]. In the following, the stagger angle of the blades is set to 22.5° and the angle of attack is 32.5°. Periodic boundary conditions are specified along the vertical direction, the velocity components are imposed at the inlet, and a convective outflow boundary condition is used at the outlet boundary. A representative snapshot of this flow at variable Reynolds numbers, depicted by instantaneous levels of vorticity ω_z, is shown in Fig. 5.

A linear stability analysis has been performed to determine the critical Reynolds number. The growth rate of the leading mode for varying Re is shown in Fig. 5, suggesting that the critical Reynolds number for this configuration is $Re_c \approx 750$. To assess the efficiency of the optimization algorithm, representative examples around and far from criticality have been chosen at, respectively, $Re = \{800, 2000, 4000\}$, shown in Fig. 5. At $Re = 800$, which is slightly above the critical Reynolds number, the flow presents an instability developing in the wake of the blades. As the Reynolds number is increased up to $Re = 2000$, an instability develops upstream resulting in pairs of vortices shedding from the trailing edges of the blades. In this case, stronger interaction between the wakes of the different blades is observed, although the wake still displays a regular pattern. Finally, at $Re = 4000$, the figure shows vortex shedding from the suction side close to the leading edge. Vorticity levels are higher in this case in comparison with the previous Reynolds numbers and a stronger interaction between the wakes is displayed, which leads to a chaotic behaviour downstream.

Table 3 gives details on the numerical grids that have been used at each Reynolds number, consisting of a structured rectangular mesh stretched in the horizontal direction in the region around the blades. The vertical grid spacing remains uniform across the full computational domain. Both $Re = \{800, 2000\}$ use the same grid. Numerical grids with larger domain size and finer grid spacing were considered at these Reynolds numbers but no significant differences were observed neither in the spectrum nor in the spatial structures of the modes obtained with the stability analysis and therefore the flow is considered to be well resolved. At $Re = 4000$, a refined grid was considered to avoid numerical instabilities.

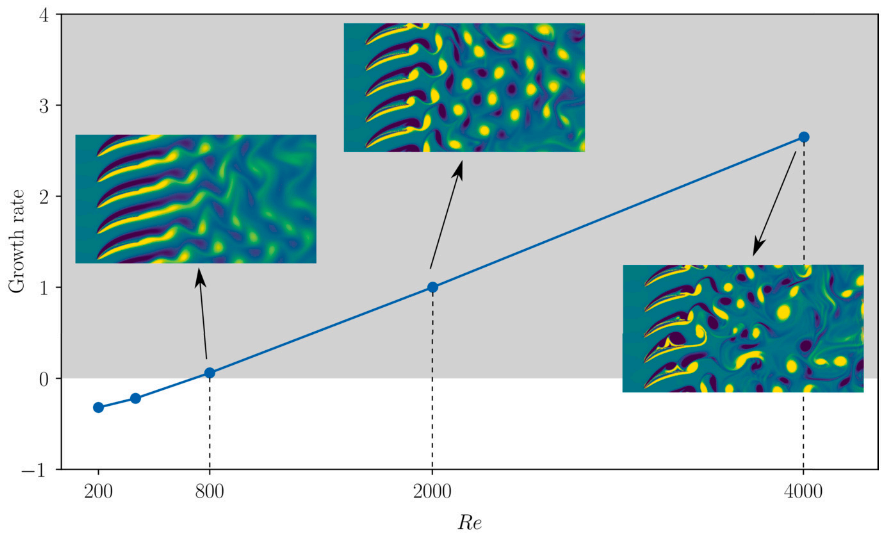


Fig. 5. Growth rates of the leading modes at different Re numbers and instantaneous snapshot showing the vorticity levels at the selected Re numbers.

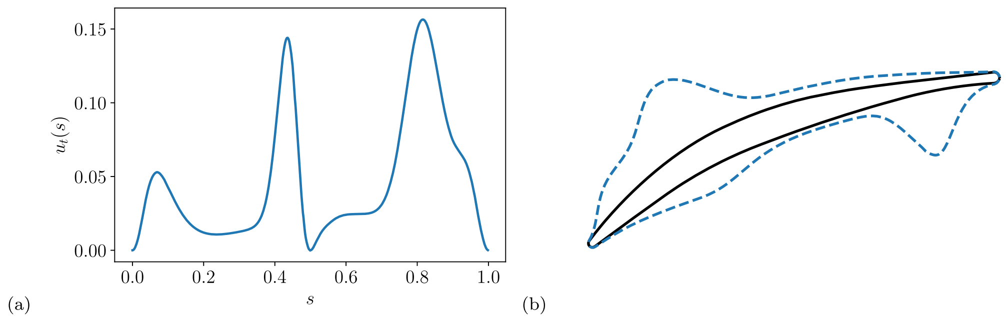


Fig. 6. Example of a representative actuation. (a) presents the tangential velocity as function of the arc-length s while (b) shows the actuation (solid blue line) superposed to the blade profile (black dashed line).

4.2. Objective function and actuation

We now intend to minimize the total pressure loss through the blade by means of an actuation that imposes a tangential velocity on the blade surface. The optimization problem can be stated as follows

$$\min \mathcal{J}(\mathbf{q}^0,\ldots,\mathbf{q}^K,\mathbf{x}) \qquad \text{s.t.} \qquad R(\mathbf{q}^{i-2},\mathbf{q}^{i-1},\mathbf{q}^i,\mathbf{x}) = 0 \;\forall\, i \in \{1,\ldots,K\}, \tag{24}$$

where $\mathbf{q}^i$ is the state vector at the i-th time step, $\mathbf{q}^0$ is the initial condition (by convention, $\mathbf{q}^{-1} = \mathbf{q}^0$), $\mathbf{x}$ is the set of control parameters, R is the residual of the propagator that allows us to determine $\mathbf{q}^i$ as a function of $\mathbf{q}^{i-1}$ and $\mathbf{q}^{i-2}$, K is the total number of iterations of the simulation and $\mathcal{J}$ is the objective function. The objective function is the defined by the sum of two terms: the average total pressure loss through the blade and a penalization term for the actuation. More precisely,

$$\mathcal{J}(\mathbf{q}^0,\ldots,\mathbf{q}^K,\mathbf{x}) = \overline{\Delta p_0(\mathbf{q}^{K_0},\ldots,\mathbf{q}^K)} + \alpha \left\| u_t(\mathbf{x}) \right\|, \tag{25}$$

where K_0 is the index of the first time step that is considered in the temporal average of the total pressure loss and α is a positive constant that penalizes the strength of the actuation. Note that the parameter $K_0 > 1$ is set to remove the contribution of the initial transients from the cost function. The total number of iterations of the simulations K is not fixed. Instead, it is updated dynamically at every simulation by applying the Cauchy criterion [53] to the averaged total pressure loss. The Hann windowing function [54] is employed to speed up convergence. The Cauchy criterion ensures that every simulation has a large enough time window and consequently low frequencies are not bypassed. The prescribed tangential velocity on the blade surface is given by

$$u_t(s,t) = \sum_{i=1}^{n} a_i f(2\pi s, 2\pi s_i, \sigma_i)\cos(\omega_i t + \phi_i), \tag{26}$$

where s is the position on the blade surface measured by the arc-length, t is the time, n is the number of actuators distributed over the surface, a_i is the amplitude of the actuator, s_i is the location of the maximum velocity imposed by the actuator, σ_i sets the width of the actuator, ω_i is the frequency of the actuator and ϕ_i is the phase. The trailing edge corresponds to $s = 0.5$ whereas the leading edge corresponds to $s = 0$ on the pressure side and $s = 1$ on the suction side. Therefore, the pressure side corresponds to values of s in the range $[0, 0.5]$ and the suction side of the blade corresponds to values of s in the range $[0.5, 1]$. Details on function f are given in Appendix B. The set of control parameters for blade j is given by $\mathbf{x}_j = (a_{1,j},\ldots,a_{n,j}, s_{1,j},\ldots,s_{n,j}, \sigma_{1,j},\ldots,\sigma_{n,j}, \omega_{1,j},\ldots,\omega_{n,j}, \phi_{1,j},\ldots,\phi_{n,j})$. An example of a representative actuation with four actuators is shown in Fig. 6, where the maximum amplitude of each actuator, without taking into account the time-dependent term, is considered for the sake of clarity.

The flow around the blades is optimized by means of four actuators on each blade. The location of the actuators is constrained so that two actuators are located on each side. The maximum width of an actuator is fixed to half the arc-length of the blade profile, and the minimum to fifty times the minimum grid size to avoid steep gradients at the surface. The upper bound on the frequency parameters is set to four times the frequency of the leading mode at the

Table 4
Optimization results.

	$Re = 800$			$Re = 2000$			$Re = 4000$		
Methods	$f_{\min}$	$\overline{\Delta p_0}$	$\alpha \left\|u_t(\mathbf{x})\right\|$	$f_{\min}$	$\overline{\Delta p_0}$	$\alpha \left\|u_t(\mathbf{x})\right\|$	$f_{\min}$	$\overline{\Delta p_0}$	$\alpha \left\|u_t(\mathbf{x})\right\|$
No actuation	0.2099	0.2099	0.0	0.1071	0.1071	0.0	0.1277	0.1277	0.0
L-BFGS-B	0.1965	0.1882	0.0083	0.1084	0.1072	0.0012	0.1246	0.1245	0.0001
DYCORS	0.1710	0.1648	0.0062	0.1069	0.1068	0.0001	-	-	-
DYCORS$_{ip}$	0.1691	0.1625	0.0066	0.1059	0.1057	0.0002	0.1109	0.1105	0.0004
G-DYCORS	0.1981	0.1955	0.0026	0.1060	0.1057	0.0003	-	-	-
G-DYCORS$_{ip}$	0.1730	0.1657	0.0073	0.1039	0.1036	0.0003	0.1085	0.1083	0.0002

corresponding Re number whereas the lower bound is set to zero. The amplitude, location, width, and angular frequency of the actuators are taken the same for every blade, and a difference in phase φ_j is allowed. More precisely, the phase of the i-th actuator on the j-th blade is given by $\phi_i + \varphi_j$, and by setting $\varphi_1 = 0$, the first blade is used as reference. The full set of 24 control parameters is then $\mathbf{x} = (a_1, \ldots, a_4, s_1, \ldots, s_4, \sigma_1, \ldots, \sigma_4, \omega_1, \ldots, \omega_4, \phi_1, \ldots, \phi_4, \varphi_2, \ldots, \varphi_5)$.

4.3. Performance of optimization strategies

The effectiveness of the gradient-enhanced DYCORS algorithm is assessed by comparison against the original derivative-free version of DYCORS for simulations at $Re = \{800, 2000, 4000\}$. Cases with and without optimization of the internal parameters of the kernel are presented for $Re = \{800, 2000\}$. The gradient-based alternative L-BFGS-B [55], which uses a limited memory version of the BFGS algorithm [56] to approximate the Hessian matrix is also used at all Reynolds numbers to compare the stochastic-based algorithm with the gradient-based counterpart. All the surrogate model based optimizations for a given Reynolds number are initialized using the same initial sampling points and the gradient-based optimization is initialized using a random point from this initial sample. The optimizations performed using the derivative-free version of the DYCORS algorithm are limited to $N_{\max} = 250$ iterations, while the optimizations carried out using the gradient-enhanced version and the L-BFGS-B algorithm are limited to $N_{\max} = 125$ iterations. Therefore, all the optimizations employ the same CPU time as the cost of computing the gradient of the objective function, using our solver, is roughly the same as the cost of performing a single function evaluation.

The value of the objective function for the optimal set of control parameters, the values of the average total pressure drop, and the penalization term for each optimization case are given in Table 4, where methods with the subscript $_{ip}$ indicate the cases with optimized internal parameters. According to this table, we can see that the gradient-enhanced version of DYCORS obtains the best results at $Re = 2000$ and $Re = 4000$ while the derivative-free version performs the best at $Re = 800$. Moreover, updating the internal parameters of the kernel improves the solution in both versions of the algorithm, and as expected the GRBF surrogates do not perform satisfactorily when internal parameters are not optimized.

Considering the gradient-based algorithm L-BFGS-B, the table shows that it presents a performance comparable to that of the stochastic algorithms only at $Re = 800$. At low Reynolds numbers, the gradient-based algorithm is expected to provide good results since the probability for the presence of multiple local minima is small due to the deterministic nature of the flow. This also implies that the different versions of the stochastic algorithms may not show significant differences, since the computed gradients may not be too steep, resulting in a comparable estimation of the interpolant using either derivative-free or gradient-enhanced version of the algorithm.

At $Re = 2000$, however, the L-BFGS-B algorithm is not even able to improve upon the case without actuation. This behaviour can be explained by the fact that the objective function is expected to have a larger amount of local minima due to the increase in the chaotic nature of flow as the Reynolds number increases, illustrated by comparing the vortical structures shown in Fig. 5. The presence of multiple local minima degrades the performance of gradient-based algorithms which are prone to get stuck in local minima. Also, presence of steeper gradients in the objective function means that derivative-free surrogates should not be able to properly interpolate the objective function and that adding the gradient information should improve the construction of the interpolant, resulting in the superior performance of the gradient-enhanced version of the stochastic algorithm.

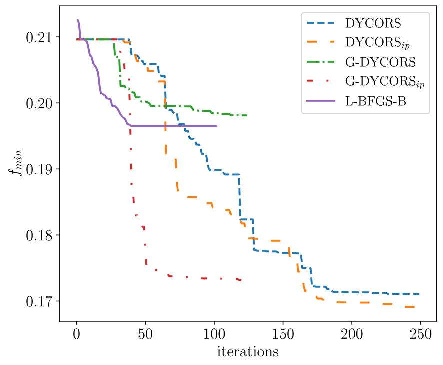
4.3.1. $Re = 800$

Fig. 7 displays the results obtained at $Re = 800$. First, Fig. 7(a) shows the convergence history of the objective function as a function of the number of iterations for each optimization performed at this Reynolds number. It can be seen that introducing the gradient in the surrogate model enhances the convergence rate of the algorithm as expected. This figure also demonstrates that an improvement is obtained when optimizing the internal parameters of the kernel, specially in the gradient enhanced version of the algorithm. Fig. 7(b) shows the same convergence history plot but taking into account the computational cost of the optimization instead of the number of iterations performed. This is accomplished by multiplying the abscissa axis by a factor of 2 in cases where the optimization is performed using the gradient information: G-DYCORS, G-DYCORS$_{ip}$, and L-BFGS-B algorithms.

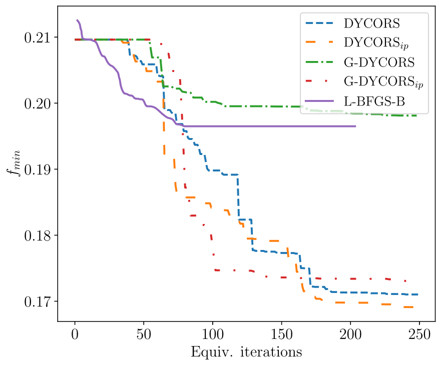
In order to find an explanation as to why the G-DYCORS$_{ip}$ algorithm did not achieve the best result at $Re = 800$ we can examine the optimal actuators that were obtained with the different algorithms, shown in Fig. 7(c). In this figure, the actuators at their maximum amplitude are plotted. It is clear that the DYCORS, DYCORS$_{ip}$ and G-DYCORS$_{ip}$ algorithms converged to a very similar solution in contrast to the G-DYCORS and the L-BFGS-B (not shown here) algorithms, which converged to a very different set of control parameters. This result suggests that the three algorithms arrived at a solution very close to the global minimum leaving little room for improvement. In addition, careful examination of the actuation profile shows that the profile is dominated by one actuator placed at the suction side of the blade between the leading edge and the mid-chord point, and the frequency of this actuator (4.25 rad/s in the case of the DYCORS$_{ip}$ and 4.43 rad/s in the case of the G-DYCORS$_{ip}$) is roughly the same as the frequency of the instability (4.76 rad/s). This observation is also confirmed by looking at Fig. 7(d,e), which displays contours of the average total pressure field, the total pressure profile at the downstream measurement location and contours of the vorticity field at the last time step for the case with and without actuation (the actuation is plotted for the DYCORS$_{ip}$ algorithm). These figures show that a reduction in the size of the low total pressure region around the blades is obtained by decreasing the intensity of the vortical structures that are being generated. Moreover, in the optimized case all the averaged wakes present the same profile whereas this is not the case for the case without actuation. Optimal actuators corresponding to DYCORS and G-DYCORS$_{ip}$ present roughly the same flow fields as that of DYCORS$_{ip}$ (not shown here).

4.3.2. $Re = 2000$

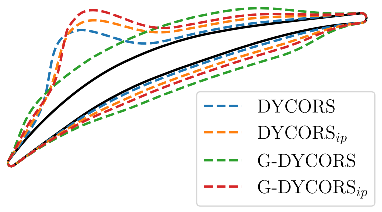
The convergence history at $Re = 2000$ as function of the iterations and as function of the computational cost is shown in Fig. 8(a,b). In



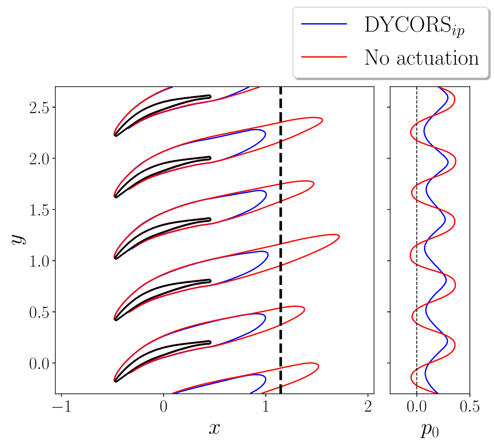
(a) Values of the minimum as function of the optimization iterations.



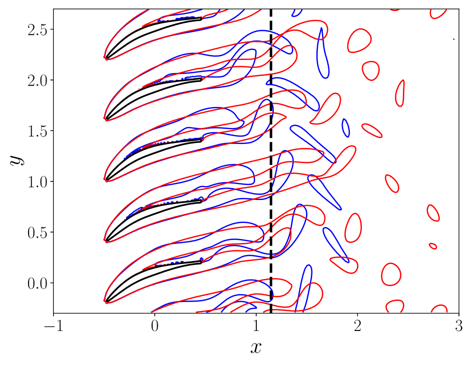
(b) Values of the minimum as function of the computational cost.



(c) Optimal actuation obtained using the DYCORS and the G-DYCORS algorithms with and without optimizing the internal parameters of the kernel.



(d) Average total pressure contour around the blades and total pressure profiles at the downstream measurement location for the case without actuation and the case using DYCORS$_{ip}$.



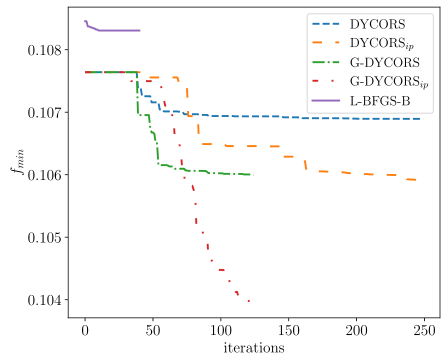
(e) Instantaneous vorticity contour for the case without actuation and the case using DYCORS$_{ip}$.

Fig. 7. Results at $Re = 800$.

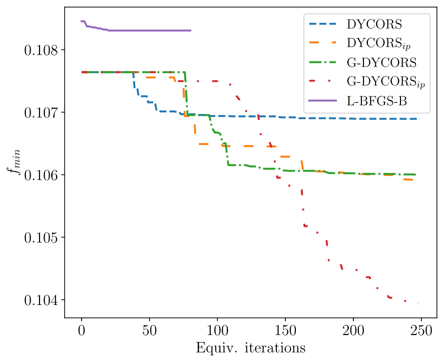
this case, the G-DYCORS$_{ip}$ algorithm is the one that obtains the best result. Again, both versions where the internal parameters are optimized present better results than their counterparts without optimization. Also, both gradient-enhanced versions improve the convergence rate of the derivative-free versions. When taking into account the computational cost, the G-DYCORS$_{ip}$ algorithm is converged after 150 iterations.

The results of Table 4, suggest a considerable difference in the optimal actuators obtained with the G-DYCORS$_{ip}$ algorithms compared to the rest at $Re = 2000$. However, Fig. 8(c) demonstrates that this difference is small. In fact, comparing the reduction in the total pressure loss obtained with the optimal actuation to that of the case without actuation shows a smaller improvement at this Reynolds number than the rest. This result suggests that at this Reynolds number, the flow is not very sensitive to this type of actuation on the blade surface, and that this form of actuation is not the best strategy to reduce the total pressure loss. This can be deduced from the shape of the actuation profile as well, where no dominant actuator is selected, instead all the actuators have similar amplitudes.

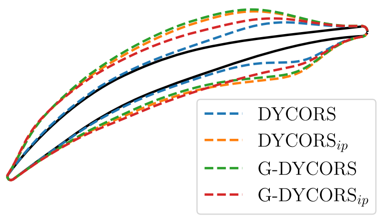
The contours of the average total pressure, the total pressure profile at the measurement location and the contours of the vorticity field are plotted in Fig. 8(d,e) for the case without actuation and for the optimal actuation obtained with the G-DYCORS$_{ip}$ algorithm at $Re = 2000$. At this Re number the instability is developing at the trailing edge of the blades instead of at the wake as in the case of $Re = 800$. In this case there are no significant differences between the case without actuation and the optimized case as we already expected. Only in the wake of the 3 bottom blades the pressure contours show a small reduction in the size of the low total pressure region downstream of the cascade.



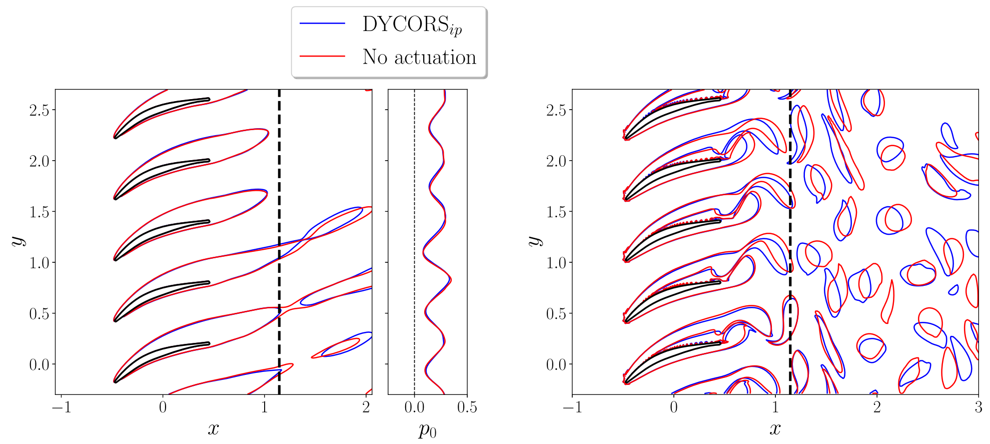
(a) Values of the minimum as function of the optimization iterations.



(b) Values of the minimum as function of the computational cost.



(c) Optimal actuation obtained using the DYCORS and the G-DYCORS algorithms with and without optimizing the internal parameters of the kernel.



(d) Average total pressure contour around the blades and total pressure profiles at the downstream measurement location for the case without actuation and the case using G-DYCORS$_{ip}$.

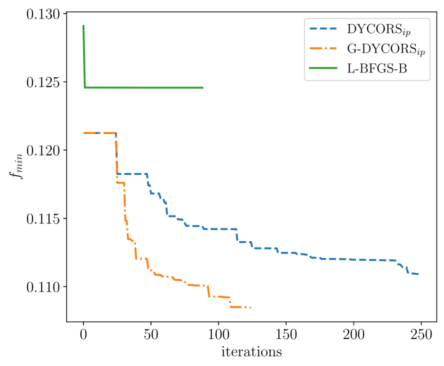
(e) Instantaneous vorticity contour for the case without actuation and the case using G-DYCORS$_{ip}$.

Fig. 8. Results at $Re = 2000$.

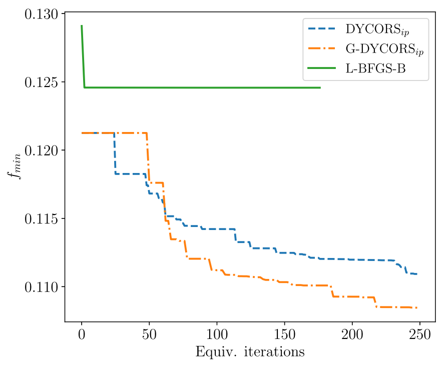
4.3.3. $Re = 4000$

At $Re = 4000$, the optimal actuators obtained with the different algorithms present significant differences as shown by Fig. 9(c), where the results of both stochastic optimizations are depicted. In the case of the DYCORS$_{ip}$ algorithm, the profile shows a dominant actuator in the pressure side close to the mid-chord point, while no dominant actuator exists in the suction side. Nevertheless, the overall contribution to the suction side suggests that the actuators placed on this side also have a greater influence on the minimization of the total pressure loss. In the case of the G-DYCORS$_{ip}$ algorithm, both the suction side and the pressure side are dominated by an actuator placed close to the leading edge, whereas the rest of the profile has smaller values of the tangential velocity when compared to the optimal actuator obtained by the derivative-free version of the algorithm. Although the DYCORS algorithm is ensured to achieve global convergence [34], the large difference between the two results suggests that in this case they followed paths to different local minima. This can be explained also by Fig. 9(a,b), where the convergence curves at this Re number do not end in a plateau shape, suggesting that the algorithms did not reach the global minimum and more iterations of the optimization algorithm are required to reach this point. Nevertheless, the convergence curves show that the G-DYCORS$_{ip}$ is able to reach the same value of the objective function employing half the computational cost of the DYCORS$_{ip}$ algorithm.

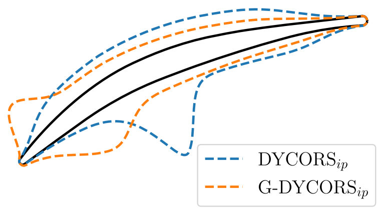
Considering Fig. 9(d,e), where the contours of the average total pressure field, the total pressure profile at the measurement location and the contours of the vorticity field are illustrated for the case without actuation and for the optimal actuation obtained with the G-DYCORS$_{ip}$, it can be observed that the vortical structures are being generated on the suction side of the blades. This explains why the dominant actuators are



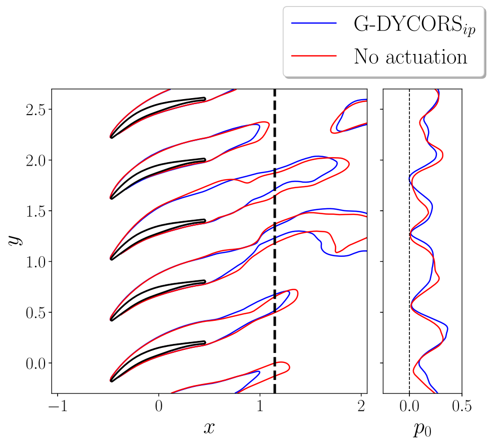
(a) Values of the minimum as function of the optimization iterations.



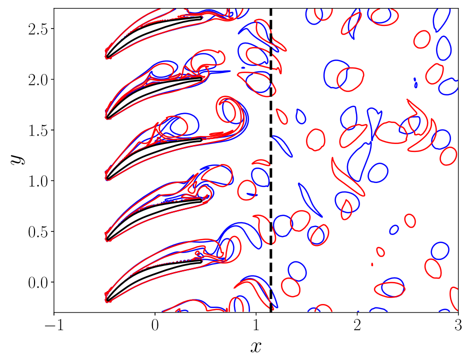
(b) Values of the minimum as function of the computational cost.



(c) Optimal actuation obtained using the DYCORS and the G-DYCORS algorithms with and without optimizing the internal parameters of the kernel.



(d) Average total pressure contour around the blades and total pressure profiles at the downstream measurement location for the case without actuation and the case using G-DYCORS$_{ip}$.



(e) Instantaneous vorticity contour for the case without actuation and the case using G-DYCORS$_{ip}$.

Fig. 9. Results at $Re = 4000$.

place so close to the leading edge in the optimal case. This way the actuation is capable of disturbing the shedding of these vortices and hence modifying the total pressure downstream of the blades. It can be seen that the shed vortices are not noticeably different qualitatively between the two cases, however the vortical structures in the wake show more variations, suggesting that the actuation is not changing the intensity of the vortices but their interaction. Again, the total pressure contours show larger low total pressure regions at the measurement location for the case without actuations although the differences are not as large as in the case at $Re = 800$.

5. Summary and conclusions

In this work we have developed an enhanced version of the derivative-free stochastic DYCORS algorithm by performing the optimization of the internal parameters of the kernel. An alternative version of the algorithm is also proposed by adding gradient information into the surrogate model to create a gradient-enhanced version of the original algorithm, which may be useful whenever the gradient information can be obtained. These two modifications improve the accuracy of the surrogate model and therefore improve the convergence rate of the algorithm. To optimize the internal parameters, the leave-one-out error is used in the case of the derivative-free version. In the case of the gradient-enhanced version, it has been found that the leave-one-out error presented in [33] does not perform satisfactorily and therefore an alternative method is proposed to optimize the values of the internal parameters based on the values of the gradients at the evaluated points. An implementation of both DYCORS and G-DYCORS algorithms together with brief documentation of the code is available at [57].

We have analyzed the performance of the stochastic algorithms and have compared it to the performance of the commonly used gradient-based algorithm L-BFGS-B at different flow regimes, from flows at roughly the critical Reynolds number to flows exhibiting chaotic behaviour. In all the cases, optimizing the internal parameters of the kernel has significantly improved the convergence rate of the algorithm. Moreover, the gradient-enhanced version has clearly outperformed the derivative-free version in two out of the three cases that have been analyzed. The comparison with the gradient-based algorithm L-BFGS-B has demonstrated that stochastic algorithms are able to achieve better results even at low Re numbers where the flow exhibits a purely periodic behaviour and where the objective function is not expected to present many local minima.

The convergence plots show that the gradient-enhanced version of the algorithm always presents a better convergence rate than the derivative-free version, even when the cost of evaluating the gradient has been factored in. However, at the lowest Re number studied, taking into account the computational cost of computing the gradient, both versions of the algorithm perform similarly. In order to compute the gradient information we have made use of the adjoint method, therefore the cost of performing a gradient evaluation is roughly the same as the cost of evaluating the objective function. Nevertheless, there exist methods that can improve the cost of gradient extraction e.g. the parallel-in-time method [58,59] where the linear equations are partitioned by separating the homogeneous and inhomogeneous parts of the equations, resulting in a speeds-up of the computation. Reducing the computational cost of performing a gradient evaluation would improve the performance of the gradient-enhanced version of the algorithm compared to the derivative-free version.

Finally, the gradient-enhanced version can still be improved by updating the function employed to generate the trial points. Since in this case the gradient is evaluated at the best evaluated point, a skew-normal distribution can be used instead of a symmetric normal distribution to randomly generate the trial points by taking into account the direction where the gradient is pointing. This would accelerate convergence to local minima and therefore improve the overall performance of the algorithm and will be investigated further in the future.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

We gratefully acknowledge the computer resources at Finisterrae and the technical support provided by CESGA (Grant No. IM-2020-3-0020 and IM-2021-1-0021).

Appendix A. Adjoint-based gradient computation

By linearizing the system of equations (22) about a steady baseflow, we obtain the following linear system of equations for the advancement of small perturbations $\mathbf{q}$

$$\begin{pmatrix} \mathbf{A} & \mathbf{Q} \\ \mathbf{Q}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{q}^{k+1} \\ \lambda \end{pmatrix} = \begin{pmatrix} (\mathbf{B} - \frac{3}{2}\mathbf{N})\mathbf{q}^k + \frac{1}{2}\mathbf{N}(\mathbf{q}^{k-1}) \\ \mathbf{0} \end{pmatrix}, \tag{A.1}$$

where the matrix $\mathbf{N}$ represents the linearized advection operator. This linearized system of equations is used to perform the gradients computation.

In order to obtain the desired derivatives of the objective function $\mathcal{J}$, we make use of adjoint variables to efficiently compute the gradients by solving a system of equations with a similar computational cost than the cost of the forward simulation. By introducing the governing equations R as a constraint in the cost function and using Lagrange multipliers we can transform the minimization problem

$$\min \mathcal{J}(\mathbf{q}^0, \dots, \mathbf{q}^K, \mathbf{x}) \text{ s.t. } R(\mathbf{q}^{i-2}, \mathbf{q}^{i-1}, \mathbf{q}^i, \mathbf{x}) = 0 \ \forall\, i \in \{1, \dots, K\}, \tag{A.2}$$

into an unconstrained problem

$$\min \mathcal{L}(\mathbf{q}^0, \dots, \mathbf{q}^K, \mathbf{x}, \lambda) = \mathcal{J}(\mathbf{q}^0, \dots, \mathbf{q}^K, \mathbf{x}) - \sum_{i=1}^{K} \lambda_i^T R(\mathbf{q}^{i-2}, \mathbf{q}^{i-1}, \mathbf{q}^i, \mathbf{x}), \tag{A.3}$$

where λ_i is the Lagrange multiplier corresponding to the residual of the i-th time step and $\mathcal{L}$ is the new cost function. We are interested in computing the gradients of the objective function with respect to the control parameters

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\mathbf{x}} = \frac{\partial \mathcal{J}}{\partial \mathbf{x}} + \sum_{i=0}^{i=K} \frac{\partial \mathcal{J}}{\partial \mathbf{q}^i} \frac{\partial \mathbf{q}^i}{\partial \mathbf{x}}, \tag{A.4}$$

which by employing the first-order optimality conditions can be rewritten as

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\mathbf{x}} = \frac{\partial \mathcal{J}}{\partial \mathbf{x}} - \sum_{i=0}^{i=K} \lambda_i^T \frac{\partial R}{\partial \mathbf{x}}, \tag{A.5}$$

where the Lagrange multipliers λ_i are obtained by solving the adjoint system backwards in time given by

$$\begin{aligned}
\frac{\partial \mathcal{J}}{\partial \mathbf{q}^0} &= 2\lambda_1^T \frac{\partial R(\mathbf{q}^0, \mathbf{q}^0, \mathbf{q}^1, \mathbf{x})}{\partial \mathbf{q}^0} + \lambda_2^T \frac{\partial R(\mathbf{q}^0, \mathbf{q}^1, \mathbf{q}^2, \mathbf{x})}{\partial \mathbf{q}^0} \\
\frac{\partial \mathcal{J}}{\partial \mathbf{q}^1} &= \lambda_1^T \frac{\partial R(\mathbf{q}^0, \mathbf{q}^0, \mathbf{q}^1, \mathbf{x})}{\partial \mathbf{q}^1} + \lambda_2^T \frac{\partial R(\mathbf{q}^0, \mathbf{q}^1, \mathbf{q}^2, \mathbf{x})}{\partial \mathbf{q}^1} + \lambda_3^T \frac{\partial R(\mathbf{q}^1, \mathbf{q}^2, \mathbf{q}^3, \mathbf{x})}{\partial \mathbf{q}^1} \\
\frac{\partial \mathcal{J}}{\partial \mathbf{q}^i} &= \lambda_i^T \frac{\partial R(\mathbf{q}^{i-2}, \mathbf{q}^{i-1}, \mathbf{q}^i, \mathbf{x})}{\partial \mathbf{q}^i} + \lambda_{i+1}^T \frac{\partial R(\mathbf{q}^{i-1}, \mathbf{q}^i, \mathbf{q}^{i+1}, \mathbf{x})}{\partial \mathbf{q}^i} \\
&\quad + \lambda_{i+2}^T \frac{\partial R(\mathbf{q}^i, \mathbf{q}^{i+1}, \mathbf{q}^{i+2}, \mathbf{x})}{\partial \mathbf{q}^i} \ \forall\, i \in \{2, \dots, K-2\} \\
\frac{\partial \mathcal{J}}{\partial \mathbf{q}^{K-1}} &= \lambda_{K-1}^T \frac{\partial R(\mathbf{q}^{K-3}, \mathbf{q}^{K-2}, \mathbf{q}^{K-1}, \mathbf{x})}{\partial \mathbf{q}^{K-1}} + \lambda_K^T \frac{\partial R(\mathbf{q}^{K-2}, \mathbf{q}^{K-1}, \mathbf{q}^K, \mathbf{x})}{\partial \mathbf{q}^{K-1}} \\
\frac{\partial \mathcal{J}}{\partial \mathbf{q}^K} &= \lambda_K^T \frac{\partial R(\mathbf{q}^{K-2}, \mathbf{q}^{K-1}, \mathbf{q}^K, \mathbf{x})}{\partial \mathbf{q}^K}.
\end{aligned} \tag{A.6}$$

Appendix B. Actuation details

The function f that appears in Eq. (26) is a Gaussian-like function periodic over 2π given by

$$\begin{aligned}
f(\theta, \theta_i, \sigma_i) &= f_1(\theta, 0) f_1(\theta, \pi) f_1(\theta, 2\pi) \sum_{n=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2}\left(\frac{\theta - \theta_i - 2\pi n}{\sigma_i}\right)^2} \\
&= f_1(\theta, 0) f_1(\theta, \pi) f_1(\theta, 2\pi) \frac{1}{2\pi} \vartheta_3 \left(\frac{1}{2}\left(\theta - \theta_i\right), e^{-\frac{\sigma_i^2}{2}} \right),
\end{aligned} \tag{B.1}$$

where ϑ_3 is the Jacobi theta function. By considering this function we ensure that a continuous distribution of tangential velocity will be obtained when applied to the blade surface. The function f_1 damps the tangential velocity to 0 close to the leading and trailing edges in order to avoid problems related to the high curvature of these sections of the blade. It is given by,

$$f_1(\theta, \alpha) = \left(1 - \exp\left(-\frac{(\theta - \alpha)^2}{\sigma_d} \right) \right), \tag{B.2}$$

where $\sigma_d = 0.1$ is chosen so that the damped region does not extend much throughout the blades surface.

Appendix C. Algorithms

This appendix presents the algorithms that have been previously mentioned in the paper. The Algorithm 2 describes the steps that need to be performed in order to generate and evaluate the new trial points. The Algorithm 3 provides the steps to select the next point to be evaluated using the expensive function evaluation given the set of trial points already evaluated at the surrogate surface. Finally, the Algorithm 4 presents the steps that need to be carried out to update several parameters needed in the optimization procedure.

Algorithm 2: Function trial_points().

```
Input: Current iteration, $n$
       Number of trial points, $k$
       Standard deviation, $\sigma_n$
       Interpolant, $s_n$
       Evaluated points, $\mathcal{A}_n$
       Coefficients of the surrogate model, $\lambda$ and $\mathbf{c}$

Result: Set of values of the trial points evaluated at the surrogate
        model, $\mathcal{B}_n$

Compute probability of perturbing a coordinate: $p_{\text{pert}} = \varphi(n)$ using Eq. (16)
Select coordinates to perturb: $\mathcal{I}_{\text{pert}} = \{i : w_i < p_{\text{pert}}\}$, where $w_i$ for
 $i = 1, \dots, d$ are generated randomly
    if $\mathcal{I}_{\text{pert}} = \emptyset$ then
    |  $\mathcal{I}_{\text{pert}} = \{j\}$ where $j$ is selected randomly from $\{1, \dots, d\}$
    end
Generate trial points: $\mathbf{y}_{n,j} = \mathbf{x}_{\text{best}} + \mathbf{z}_j$ for $j = 1, \dots, k$, where
 $\mathbf{z}_j^i = 0 \; \forall \; i \notin \mathcal{I}_{\text{pert}}$ and a random number from the normal distribution
 $\mathcal{N}(0, \sigma_n^2)$
Ensure trial points are in the domain:
    if $\mathbf{y}_{n,j} \notin \mathcal{D}$ then
    |  Replace $\mathbf{y}_{n,j}$ by closest point in the boundary $\partial \mathcal{D}$
    end
Evaluate trial points: Compute $\mathcal{B}_n = s_n(\mathbf{y}_{n,j}, \mathcal{A}_n, \lambda, \mathbf{c})$ for $j = 1, \dots, k$
 where $s_n$ is given by Eqs. (2) and (5)
```

Algorithm 3: Function select_next_point().

```
Input: Current iteration, $n$
       Weight pattern, $\boldsymbol{\Upsilon}$
       Evaluated points, $\mathcal{A}_n$
       Set of trial points, $\mathbf{y}_n$
       Set of values of the trial points, $\mathcal{B}_n$
Result: Best candidate point, $\mathbf{x}_{n+1}$

Compute RBF score: Compute $s_n^{\max} = \max(\mathcal{B}_n)$ and $s_n^{\min} = \min(\mathcal{B}_n)$
    if $s_n^{\max} = s_n^{\min}$ then
    |  $V_{n,j}^{\text{RBF}} = 1$ for $j = 1, \dots, k$
    else
    |  $V_{n,j}^{\text{RBF}} = (\mathcal{B}_{n,j} - s_n^{\min})/(s_n^{\max} - s_n^{\min})$ for $j = 1, \dots, k$
    end
Compute distance score: Compute $\Delta_{n,j} = \min \left\| \mathbf{y}_{n,j} - \mathbf{x} \right\| : \mathbf{x} \in \mathcal{A}_n$,
 $\Delta_n^{\max} = \max(\Delta_{n,j})$ and $\Delta_n^{\min} = \min(\Delta_{n,j})$ for $j = 1, \dots, k$
    if $\Delta_n^{\max} = \Delta_n^{\min}$ then
    |  $V_{n,j}^{\text{dist}} = 1$ for $j = 1, \dots, k$
    else
    |  $V_{n,j}^{\text{dist}} = (\Delta_n^{\max} - \Delta_{n,j})/(\Delta_n^{\max} - \Delta_n^{\min})$ for $j = 1, \dots, k$
    end
Pick weights: $w_n^{\text{RBF}} = \Upsilon_{n\%4}$ and $w_n^{\text{dist}} = 1 - w_n^{\text{RBF}}$
Compute global score: $w_{n,j} = w_n^{\text{RBF}} V_{n,j}^{\text{RBF}} + w_n^{\text{dist}} V_{n,j}^{\text{dist}}$
Select next point to be evaluated: $\mathbf{x}_{n+1} = \min_{\mathbf{y}_{n,j}} w_{n,j}$
```

Algorithm 4: Function update_info().

```
Input: Current iteration, $n$
       Current minimum, $\mathbf{x}_{\text{best}}$ and $f_{\text{best}}$
       Last evaluated point, $\mathbf{x}_{n+1}$ and $f(\mathbf{x}_{n+1})$
       Counters, $C_{\text{f}}$ and $C_{\text{s}}$
       Counter limits, $\tau_{\text{f}}$ and $\tau_{\text{s}}$
       Current standard deviation, $\sigma_n$
       Set of evaluated points, $\mathcal{A}_n$

Output: Updated minimum, $\mathbf{x}_{\text{best}}$ and $f_{\text{best}}$
        Updated counters, $C_{\text{f}}$ and $C_{\text{s}}$
        Updated standard deviation, $\sigma_n$
        Updated set of evaluated points, $\mathcal{A}_{n+1}$
        Updated iteration, $n$

Update counters and current best:
    if $f(\mathbf{x}_{n+1}) < f_{\text{best}}$ then
    |  $C_{\text{s}} = C_{\text{s}} + 1, C_{\text{f}} = 0$
    |  $\mathbf{x}_{\text{best}} = \mathbf{x}_{n+1}, f(\mathbf{x}_{\text{best}}) = f(\mathbf{x}_{n+1})$
    else
    |  $C_{\text{s}} = 0, C_{\text{f}} = C_{\text{f}} + 1$
    end
Update step size:
    if $C_{\text{s}} >= \tau_s$ then
    |  $\sigma_{n+1} = 2\sigma_n, C_{\text{s}} = 0$
    end
    if $C_{\text{f}} >= \tau_f$ then
    |  $\sigma_{n+1} = 0.5\sigma_n, C_{\text{f}} = 0$
    end
Update set of evaluated points: $\mathcal{A}_{n+1} = \mathcal{A}_n \cup \{\mathbf{x}_{n+1}\}$
Update iteration number: $n = n + 1$
```

References

[1] O. Pironneau, On optimum design in fluid mechanics, J. Fluid Mech. 64 (1) (1974) 97–110, https://doi.org/10.1017/S0022112074002023, https://www.cambridge.org/core/product/identifier/S0022112074002023/type/journal_article.

[2] A. Jameson, L. Martinelli, N. Pierce, Optimum aerodynamic design using the Navier-Stokes equations, Theor. Comput. Fluid Dyn. 10 (1–4) (1998) 213–237, https://doi.org/10.1007/s001620050060, http://link.springer.com/10.1007/s001620050060.

[3] M.P. Juniper, Triggering in the horizontal Rijke tube: non-normality, transient growth and bypass transition, J. Fluid Mech. 667 (2011) 272–308, https://doi.org/10.1017/S0022112010004453, https://www.cambridge.org/core/product/identifier/S0022112010004453/type/journal_article.

[4] D. Foures, C. Caulfield, P. Schmid, Optimal mixing in two-dimensional plane Poiseuille flow at finite Péclet number, J. Fluid Mech. 748 (2014) 241–277, https://doi.org/10.1017/jfm.2014.182, https://www.cambridge.org/core/product/identifier/S0022112014001827/type/journal_article.

[5] S. Schmidt, C. Ilic, V. Schulz, N.R. Gauger, Three-dimensional large-scale aerodynamic shape optimization based on shape calculus, AIAA J. 51 (11) (2013) 2615–2627, https://doi.org/10.2514/1.J052245, https://arc.aiaa.org/doi/10.2514/1.J052245.

[6] S.M.E. Rabin, C.P. Caulfield, R.R. Kerswell, Designing a more nonlinearly stable laminar flow via boundary manipulation, J. Fluid Mech. 738 (2014) R1, https://doi.org/10.1017/jfm.2013.601, https://www.cambridge.org/core/product/identifier/S0022112013006010/type/journal_article.

[7] A. Fikl, V. Le Chenadec, T. Sayadi, Control and optimization of interfacial flows using adjoint-based techniques, Fluids 5 (3) (2020) 156, https://doi.org/10.3390/fluids5030156, https://www.mdpi.com/2311-5521/5/3/156.

[8] A. Hassan, T. Sayadi, V. Le Chenadec, A. Attili, Sensitivity analysis of an unsteady char particle combustion, Fuel 287 (2021) 119738, https://doi.org/10.1016/j.fuel.2020.119738, https://linkinghub.elsevier.com/retrieve/pii/S0016236120327344.

[9] A. Hassan, T. Sayadi, V. Le Chenadec, H. Pitsch, A. Attili, Adjoint-based sensitivity analysis of steady char burnout, Combust. Theory Model. 25 (1) (2021) 96–120, https://doi.org/10.1080/13647830.2020.1838614, arXiv:2012.00640, http://arxiv.org/abs/2012.00640.

[10] A.L. Marsden, J.A. Feinstein, C.A. Taylor, A computational framework for derivative-free optimization of cardiovascular geometries, Comput. Methods Appl. Mech. Eng. 197 (21–24) (2008) 1890–1905, https://doi.org/10.1016/j.cma.2007.12.009, https://linkinghub.elsevier.com/retrieve/pii/S0045782507004884.

[11] S. Pierret, R. Filomeno Coelho, H. Kato, Multidisciplinary and multiple operating points shape optimization of three-dimensional compressor blades, Struct. Multidiscip. Optim. 33 (1) (2006) 61–70, https://doi.org/10.1007/s00158-006-0033-y, http://link.springer.com/10.1007/s00158-006-0033-y.

[12] D.R. Jones, A taxonomy of global optimization methods based on response surfaces, J. Glob. Optim. 21 (2001) 345–383, https://doi.org/10.1023/A:1012771025575.

[13] H.-M. Gutmann, A radial basis function method for global optimization, J. Glob. Optim. 19 (2001) 201–227, https://doi.org/10.1023/A:1011255519438.

[14] A.W. Moore, J.G. Schneider, Memory-based stochastic optimization, in: Advances of Neural Information Processing Systems, vol. 8, 1995, pp. 1066–1072, https://proceedings.neurips.cc/paper/1995/file/c7635bfd99248a2cdef8249ef7bfbef4-Paper.pdf.

[15] M. Powell, On trust region methods for unconstrained minimization without derivatives, Math. Program. 97 (3) (2003) 605–623, https://doi.org/10.1007/s10107-003-0430-6, http://link.springer.com/10.1007/s10107-003-0430-6.

[16] R.H. Myers, D.C. Montgomery, C.M. Anderson-Cook, Response Surface Methodology: Process and Product Optimization Using Designed Experiments, 3rd edition, Wiley Series in Probability and Statistics, Wiley, 2009.

[17] J.-P. Chilès, N. Desassis, Fifty years of Kriging, in: B. Daya Sagar, Q. Cheng, F. Agterberg (Eds.), Handbook of Mathematical Geosciences, Springer International Publishing, Cham, 2018, pp. 589–612, http://link.springer.com/10.1007/978-3-319-78999-6_29.

[18] M.J.D. Powell, The theory of radial basis function approximation in 1990, in: Advances in Numerical Analysis, Volume 2: Wavelets, Subdivision Algorithms and Radial Basis Functions, W. Light Edition, Oxford Univ. Press, Oxford, UK, 1992, pp. 105–210.

[19] A.J. Smola, B. Schölkopf, A tutorial on support vector regression, Stat. Comput. 14 (3) (2004) 199–222, https://doi.org/10.1023/B:STCO.0000035301.49549.88, http://link.springer.com/10.1023/B:STCO.0000035301.49549.88.

[20] G. Gary Wang, Z. Dong, P. Aitchison, Adaptive response surface method - a global optimization scheme for approximation-based design problems, Eng. Optim. 33 (6) (2001) 707–733, https://doi.org/10.1080/03052150108940940, http://www.tandfonline.com/doi/abs/10.1080/03052150108940940.

[21] D.R. Jones, M. Schonlau, W.J. Welch, Efficient global optimization of expensive black-box functions, J. Glob. Optim. 13 (1998) 455–492, https://doi.org/10.1023/A:1008306431147.

[22] R.G. Regis, C.A. Shoemaker, Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization, Eng. Optim. 45 (5) (2013) 529–555, https://doi.org/10.1080/0305215X.2012.687731, http://www.tandfonline.com/doi/abs/10.1080/0305215X.2012.687731.

[23] Z.-H. Han, Y. Zhang, C.-X. Song, K.-S. Zhang, Weighted gradient-enhanced Kriging for high-dimensional surrogate modeling and design optimization, AIAA J. 55 (12) (2017) 4330–4346, https://doi.org/10.2514/1.J055842, https://arc.aiaa.org/doi/10.2514/1.J055842.

[24] S.J. Leary, A. Bhaskar, A.J. Keane, A derivative based surrogate model for approximating and optimizing the output of an expensive computer simulation, J. Glob. Optim. 30 (1) (2004) 39–58, https://doi.org/10.1023/B:JOGO.0000049094.73665.7e, http://link.springer.com/10.1023/B:JOGO.0000049094.73665.7e.

[25] A. March, K. Willcox, Q. Wang, Gradient-based multifidelity optimisation for aircraft design using Bayesian model calibration, Aeronaut. J. 115 (1174) (2011) 729–738, https://doi.org/10.1017/S0001924000006473, https://www.cambridge.org/core/product/identifier/S0001924000006473/type/journal_article.

[26] D. Peri, F. Tinti, A multistart gradient-based algorithm with surrogate model for global optimization, Commun. Appl. Ind. Math. 3 (2012) 23, https://doi.org/10.1685/JOURNAL.CAIM.393.

[27] Z. Ugray, L. Lasdon, J. Plummer, F. Glover, J. Kelly, R. Martí, Scatter search and local NLP solvers: a multistart framework for global optimization, INFORMS J. Comput. 19 (3) (2007) 328–340, https://doi.org/10.1287/ijoc.1060.0175, http://pubsonline.informs.org/doi/10.1287/ijoc.1060.0175.

[28] H.-S. Chung, J. Alonso, Using gradients to construct cokriging approximation models for high-dimensional design optimization problems, in: 40th AIAA Aerospace Sciences Meeting & Exhibit, American Institute of Aeronautics and Astronautics, Reno, NV, U.S.A., 2002, https://arc.aiaa.org/doi/10.2514/6.2002-317.

[29] J. Wu, M. Poloczek, A.G. Wilson, P. Frazier, Bayesian optimization with gradients, in: Advances in Neural Information Processing Systems, Long Beach, CA, USA, vol. 30, 2017.

[30] M.A. Bouhlel, J.R.R.A. Martins, Gradient-enhanced Kriging for high-dimensional problems, Eng. Comput. 35 (1) (2019) 157–173, https://doi.org/10.1007/s00366-018-0590-x, http://link.springer.com/10.1007/s00366-018-0590-x.

[31] S. Shekhar, T. Javidi, Significance of gradient information in Bayesian optimization, in: Proceedings of the 24th International Conference on Artificial Intelligence and Statistics, vol. 130, PMLR, 2021.

[32] Y.S. Ong, K.Y. Lum, P.B. Nair, Hybrid evolutionary algorithm with Hermite radial basis function interpolants for computationally expensive adjoint solvers, Comput. Optim. Appl. 39 (2008) 97–119, https://doi.org/10.1007/s10589-007-9065-5.

[33] M. Bompard, J. Peter, J.-A. Desideri, Surrogate models based on function and derivative values for aerodynamic global optimization, in: V European Conference on Computational Fluid Dynamics ECCOMAS CFD 2010, ECCOMAS, Lisbonne, Portugal, 2010, p. 18.

[34] R.G. Regis, C.A. Shoemaker, A stochastic radial basis function method for the global optimization of expensive functions, INFORMS J. Comput. 19 (4) (2007) 497–509, https://doi.org/10.1287/ijoc.1060.0182.

[35] G. Rudolph, Globale Optimierung mit parallelen Evolutionsstrategien, Diplomarbeit, Department of Computer Science, University of Dortmund, Jul. 1990.

[36] B. Matérn, Spatial Variation, Lecture Notes in Statistics, vol. 36, Springer, New York, New York, NY, 1986, http://link.springer.com/10.1007/978-1-4615-7892-5.

[37] M.D. Buhmann, Radial basis functions, Acta Numer. 9 (2000) 1–38, https://doi.org/10.1017/S0962492900000015.

[38] K. Giannakoglou, D. Papadimitriou, I. Kampolis, Aerodynamic shape design using evolutionary algorithms and new gradient-assisted metamodels, Comput. Methods Appl. Mech. Eng. 195 (44–47) (2006) 6312–6329, https://doi.org/10.1016/j.cma.2005.12.008, https://linkinghub.elsevier.com/retrieve/pii/S0045782506000338.

[39] L. Laurent, R. Le Riche, B. Soulier, P.-A. Boucard, An overview of gradient-enhanced metamodels with applications, Arch. Comput. Methods Eng. 26 (1) (2019) 61–106, https://doi.org/10.1007/s11831-017-9226-3, http://link.springer.com/10.1007/s11831-017-9226-3.

[40] S. Rippa, An algorithm for selecting a good value for the parameter C in radial basis function interpolation, Adv. Comput. Math. 11 (2) (1999) 193–210, https://doi.org/10.1023/A:1018975909870, http://link.springer.com/10.1023/A:1018975909870.

[41] B.A. Tolson, C.A. Shoemaker, Dynamically dimensioned search algorithm for computationally efficient watershed model calibration: dynamically dimensioned search algorithm, Water Resour. Res. 43 (1) (Jan. 2007), https://doi.org/10.1029/2005WR004723, http://doi.wiley.com/10.1029/2005WR004723.

[42] J.C. Helton, F.J. Davis, Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems, Reliab. Eng. Syst. Saf. 81 (1) (2003) 23–69.

[43] B. Beachkofski, R. Grandhi, Improved distributed hypercube sampling, in: 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, American Institute of Aeronautics and Astronautics, Denver, Colorado, 2002, http://arc.aiaa.org/doi/10.2514/6.2002-1274.

[44] M. Fosas de Pando, IBMOS: immersed boundary method optimization and stability, https://doi.org/10.5281/zenodo.3757783, 2020.

[45] K. Taira, T. Colonius, The immersed boundary method: a projection approach, J. Comput. Phys. 225 (2) (2007) 2118–2137, https://doi.org/10.1016/j.jcp.2007.03.005, https://linkinghub.elsevier.com/retrieve/pii/S0021999107001234.

[46] A.M. Roma, C.S. Peskin, M.J. Berger, An adaptive version of the immersed boundary method, J. Comput. Phys. 153 (2) (1999) 509–534, https://doi.org/10.1006/jcph.1999.6293, https://linkinghub.elsevier.com/retrieve/pii/S0021999199962939.

[47] M.B. Giles, N.A. Pierce, An introduction to the adjoint approach to design, Flow Turbul. Combust. 65 (3) (2000) 393–415, https://doi.org/10.1023/A:1011430410075.

[48] M.P. Deisenroth, A.A. Faisal, C.S. Ong, Mathematics for Machine Learning, 1st edition, Cambridge University Press, 2020.

[49] N.L. Sanger, The use of optimization techniques to design-controlled diffusion compressor blading, J. Eng. Power 105 (2) (1983) 256–264, https://doi.org/10.1115/1.3227410.

[50] H. Yang, L. He, Experimental study on linear compressor cascade with three-dimensional blade oscillation, J. Propuls. Power 20 (1) (2004) 180–188, https://doi.org/10.2514/1.1280, https://arc.aiaa.org/doi/10.2514/1.1280.

[51] L. He, J. Yi, Two-scale methodology for URANS/large eddy simulation solutions of unsteady turbomachinery flows, J. Turbomach. 139 (10) (2017) 101012, https://doi.org/10.1115/1.4036765, https://asmedigitalcollection.asme.org/turbomachinery/article/doi/10.1115/1.4036765/378813/TwoScale-Methodology-for-URANSLarge-Eddy.

[52] H.M. Phan, L. He, Validation studies of linear oscillating compressor cascade and use of influence coefficient method, J. Turbomach. 142 (5) (2020) 051005, https://doi.org/10.1115/1.4045657, https://asmedigitalcollection.asme.org/turbomachinery/article/doi/10.1115/1.4045657/1071624/Validation-Studies-of-Linear-Oscillating.

[53] S. Abbott, Understanding Analysis, Undergraduate Texts in Mathematics, Springer, New York, New York, NY, 2015, http://link.springer.com/10.1007/978-1-4939-2712-8.

[54] P. Kahlig, Some aspects of Julius von Hann's contribution to modern climatology, in: G. McBean, M. Hantel (Eds.), Geophysical Monograph Series, vol. 75, American Geophysical Union, Washington, D.C., 1993, pp. 1–7, http://doi.wiley.com/10.1029/GM075p0001.

[55] R.H. Byrd, P. Lu, J. Nocedal, A limited memory algorithm for bound constrained optimization, SIAM J. Sci. Comput. 16 (5) (1995) 1190–1208, https://doi.org/10.1137/0916069.

[56] R. Fletcher, Practical Methods of Optimization, 2nd edition, John Wiley & Sons, New York, NY, 1987.

[57] A. Quirós Rodríguez, DyCors, https://doi.org/10.5281/zenodo.5907690, 2022.

[58] C.S. Skene, M.F. Eggl, P.J. Schmid, A parallel-in-time approach for accelerating direct-adjoint studies, arXiv:2004.00546, http://arxiv.org/abs/2004.00546, Apr. 2020.

[59] S. Costanzo, T. Sayadi, M. Fosas de Pando, P. Schmid, P. Frey, Parallel-in-time adjoint-based optimization – application to unsteady incompressible flows, J. Comput. Phys. 471 (2022) 111664, https://doi.org/10.1016/j.jcp.2022.111664, https://linkinghub.elsevier.com/retrieve/pii/S0021999122007276.