Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 373 (2021) 113485
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Multi-fidelity deep neural network surrogate model for aerodynamic shape optimization

Xinshuai Zhang, Fangfang Xie, Tingwei Ji*, Zaoxu Zhu, Yao Zheng

Center for Engineering and Scientific Computation, and School of Aeronautics and Astronautics Zhejiang University, Zhejiang 310027, China

Received 28 April 2020; received in revised form 25 September 2020; accepted 1 October 2020
Available online 12 November 2020

Abstract

In the present study, an effective optimization framework of aerodynamic shape design is established based on the multi-fidelity deep neural network (MFDNN) model. The objective of the current work is to construct a high-accuracy multi-fidelity surrogate model correlating the configuration parameters of an aircraft and its aerodynamic performance by blending different fidelity information and adaptively learning their linear or nonlinear correlation without any prior assumption. In the optimization framework, the high-fidelity model using a CFD evaluation with fine grid and the low-fidelity model using the same CFD model with coarse grid are applied. Moreover, in each optimization iteration, the high-fidelity infilling strategy by adding the current optimal solution of surrogate model into the high-fidelity database is applied to improve the surrogate accuracy. The low-fidelity infilling strategy which can generate the solutions distributed uniformly in the whole design space is used to update the low-fidelity database for avoiding local optimum. Then, the proposed multi-fidelity optimization framework is validated by two standard synthetic benchmarks. Finally, it is applied to the high-dimensional aerodynamic shape optimization of a RAE2822 airfoil parameterized by 10 design variables and a DLR-F4 wing-body configuration parameterized by 30 design variables. The optimization results demonstrate that the proposed multi-fidelity optimization framework can remarkably improve optimization efficiency and outperform the single-fidelity method.

Keywords: Multi-fidelity; Surrogate model; Deep neural network; Aerodynamic shape optimization

1. Introduction

Aerodynamic shape optimization is important in the aeronautical engineering, especially in the modern civil aircraft design. It has been shown that if the drag coefficient of civil aircraft falls down one count (1 count = 0.0001), the payload will increase by 7.56%. With the rapid development of Computational Fluid Dynamics (CFD) technology, aerodynamic shape optimization has attracted numerous scholars' attention and great progresses have been achieved, such as the application of CFD based aerodynamic shape optimization in Boeing 787 and Airbus A380 [1].

The aerodynamic shape optimization methods can be divided into three categories, including the gradient-based method, the gradient-free method and the surrogate-based method. On one hand, the gradient-based optimization

* Corresponding author.
E-mail address: zjjtw@zju.edu.cn (T. Ji).

https://doi.org/10.1016/j.cma.2020.113485

methods, such as Quasi-Newton Method, Interior Point Method, Conjugate Gradient method and Sequential Quadratic Programming [2] are very efficient to search the optimal solution based on the gradient information computed by an adjoint approach [3–6]. However, the optimization results are very sensitive to the initial values and tend to trap in a local optimum. On the other hand, the gradient-free optimization method can find the global optimal solution in the whole design space with thousands or even millions of optimization objective evaluations, such as Simulated Annealing [7], Particle Swarm Optimization [8], Ant Colony Algorithm [9] and Genetic Algorithms [10]. While, if the evaluations of optimization objective function are expensive, the gradient-free optimization method will be extremely time-consuming [11,12]. In order to obtain the global optimal solution within a reasonable amount of time, the surrogate-based optimization method is prominent. Surrogate model can replace a large number of expensive objective evaluations by an approximation model, such as Polynomial Response Surfaces [13], Kriging [14], Support Vector Machines [15], or Artificial Neural Networks [16]. Meanwhile, it provides the search direction towards the global optimum for optimization and it is updated by evaluating new samples by an infilling strategy [17]. Extensive works have been conducted based on the surrogate-based optimization methods [18–20].

Recently, some researchers proposed using the low-fidelity information to enhance the prediction accuracy of surrogate model [21]. Kennedy and O'Hagan [22] proposed a Co-kriging model in terms of Gauss process regressive and an autoregressive scheme by establishing the correlation between the low-fidelity and high-fidelity data. Bonfiglio and Perdikaris [23] applied a multi-fidelity Gauss process regression and Bayesian optimization method [24] in a hydrofoil design optimization problem using a few expensive high-fidelity samples and a large number of cheap low-fidelity samples. Han and Xu [25] proposed a multi-level hierarchical kriging model in the airfoil and wing shape optimization. In their models, the correlation between the low-fidelity and high-fidelity was assumed to be linear. Perdikaris and Raissi [26] extended the linear autoregressive multi-fidelity modeling method to a non-linear modeling by introducing a hypothesis that the correlation parameters obey Gaussian distribution. However, the multi-fidelity approaches based on Gaussian distribution still have some limitations, such as approximations of discontinuous functions, high dimensional problems and strong nonlinearities for optimization. In order to handle the problems mentioned above, Meng and Karniadakis [27] developed a composite neural network in the inverse partial differential equation problems with multi-fidelity data.

To overcome the drawback of the multi-fidelity model based on Gaussian distribution, we adopt the MFDNN model as surrogate model in the aerodynamic shape optimization inspired by Ref. [27], especially in the complex high dimensional optimization problem. In our optimization framework, the Particle Swarm Optimization (PSO) algorithm is applied to find the optimal solution of the surrogate model in each iteration. To update the surrogate model, the high-fidelity infilling strategy is applied to improve the prediction precision by adding the current optimal solution of the surrogate model into the high-fidelity database. And the low-fidelity infilling strategy which can generate the solutions distributed uniformly in the whole design space is used to explore the unknown design space. Then, the multi-fidelity optimization framework is verified through two standard synthetic benchmarks. Finally, the MFDNN-based optimization framework is applied to lift–drag ratio maximization of a RAE2822 airfoil and drag minimization of a DLR-F4 wing-body configuration.

The outline of the current paper is organized as follows: in Section 2, the MFDNN-based optimization framework is proposed. In Section 3, the MFDNN-based optimization framework is validated by two standard synthetic benchmarks. In Section 4, the MFDNN-based optimization framework is applied to optimize the aerodynamic shape of a RAE2822 airfoil and a DLR-F4 wing-body configuration. In Section 5, the conclusions will be drawn.

2. Multi-fidelity deep neural network surrogate-based aerodynamic shape optimization framework

The aerodynamic design optimization consists of the modules of the geometric modeling, the mesh automatic generation, the CFD simulation, the surrogate model and the optimizer. In this section, the surrogate-based aerodynamic shape optimization framework using the MFDNN model is proposed. As shown in Fig. 1, specifically, the multi-fidelity information are blended to construct a surrogate model for the aerodynamic shape optimization and the aerodynamic computing with various fidelities is running in parallel simultaneously on the multi-core computers. Our proposed optimization framework can automatically and efficiently search the optimal aerodynamic shape to satisfy design requirements. Meanwhile, we develop a novel geometry representation method and employ automatically unstructured grid automatic generation method to achieve large-scale geometric deformation optimization.

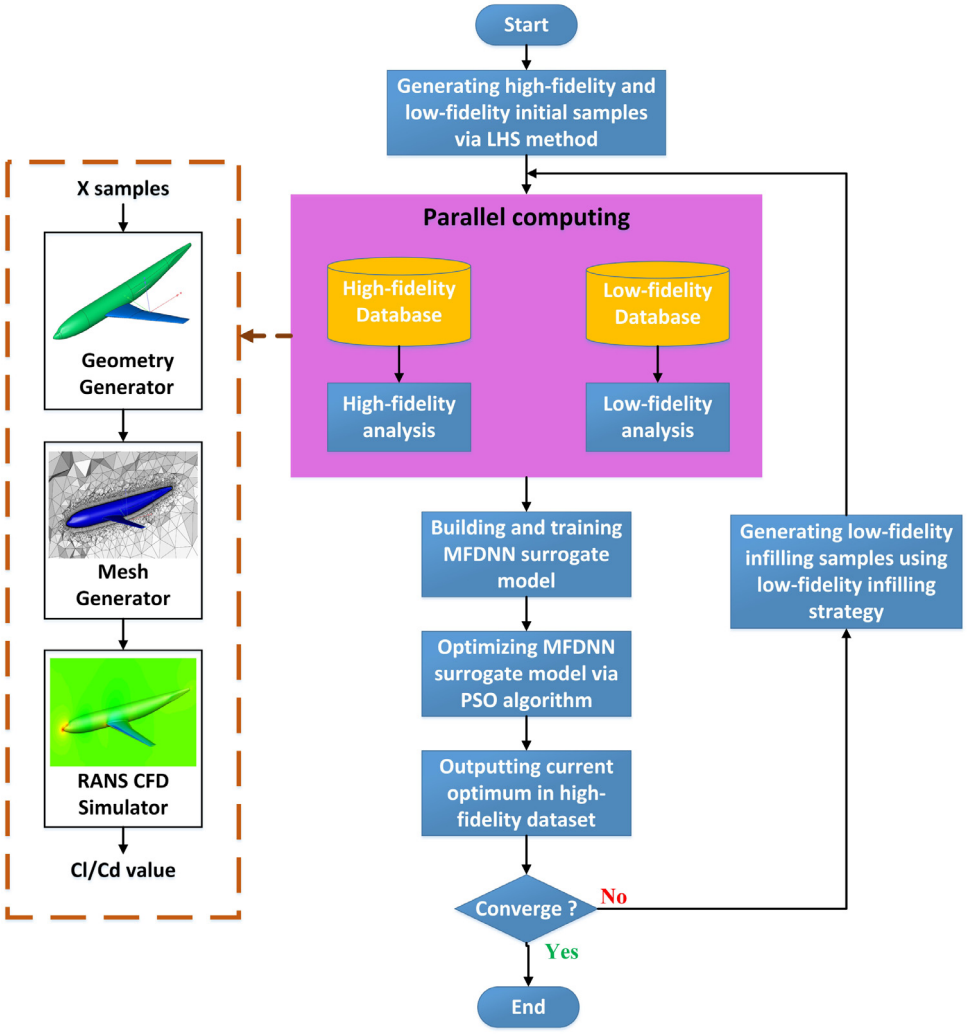


Fig. 1. Flowchart of the multi-fidelity deep neural network surrogate-based optimization framework.

2.1. Multi-fidelity deep neural network surrogate model

In this subsection, the MFDNN model is presented. It is used to construct the accurate responses by blending different fidelity information.

The main issue in the multi-fidelity modeling is how to exploit the correlation between the low-fidelity and high-fidelity information. In the current work, the auto-regressive scheme [22] is adopted and expressed as

$$y_H = \rho(x)\, y_L + \delta(x), \tag{1}$$

where y_L and y_H denote the low-fidelity and high-fidelity data, respectively; $\rho(x)$ is a scaling factor that quantifies the correlation between $\{y_H, y_L\}$, and δ is the corresponding noise. This scheme can obtain a satisfied result for handling the linear correlation between the low-fidelity and high-fidelity data. In order to capture the nonlinear correlation between these two fidelity data, Meng and Karniadakis [27] have put forth a generalized autoregressive scheme:

$$y_H = \alpha F_l(x_H,\ y_L) + (1-\alpha) F_{nl}(x_H, y_L),\ \alpha \in [0,\ 1], \tag{2}$$

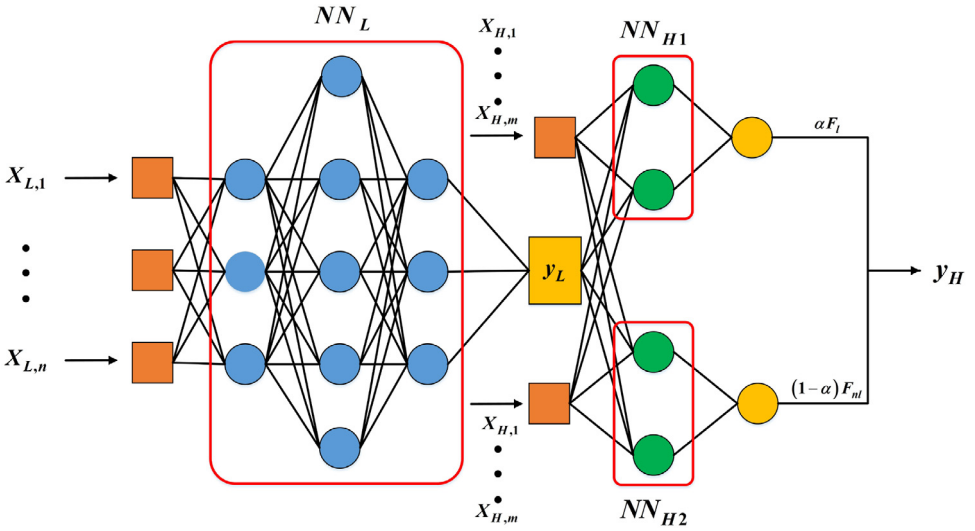


Fig. 2. Architecture of the multi-fidelity deep neural network. The orange squares represent the input units of the model and the yellow squares represent output units of the model. The blue and green circles denote the neurons of the neural network. NN_L is used to approximate the low-fidelity data. The second and third neural networks, NN_{H_1} and NN_{H_2} are used to approximate the correlation between the low-fidelity and high-fidelity data. $X_{L,1}$ and $X_{L,n}$ denote the first and the nth low-fidelity samples respectively. $X_{H,1}$ and $X_{H,m}$ denote the first and the mth high-fidelity samples respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

F represents a mapping from high-fidelity input x_H and low-fidelity neural network prediction y_L to high-fidelity neural network prediction y_H. F is divided into two parts, where F_l denotes the linear term and F_{nl} denotes the nonlinear terms. A new hyper-parameter α is to construct the correlation between the low-fidelity and high-fidelity data, determined by the given data. A higher value of α indicates the low-fidelity and high-fidelity data perform a stronger linear correlation.

As seen in Fig. 2, the presented multi-fidelity deep neural network model is composed of three deep fully-connected feed-forward neural networks. The first neural network $NN_L(x_L, \theta)$ is used to approximate the low-fidelity data. The second and third neural networks, $NN_{H_I}(x, y_L, \beta_i, i = 1, 2)$ are used to approximate the correlation between the low-fidelity and high-fidelity data, where NN_{H_1} approximates the linear correlation and NN_{H_2} approximates the nonlinear correlation respectively. $\theta, \alpha, \beta_i, (i = 1, 2)$ are the unknown hyper-parameters of the MFDNN model, where θ and $\beta_i, (i = 1, 2)$ denote weights and biases of neural networks. They can be learned by optimizing the following loss function:

$$MSE = MSE_{y_L} + MSE_{y_H} + \lambda \sum \beta_i^2, \tag{3}$$

with

$$\begin{aligned} MSE_{y_L} &= \frac{1}{N_{y_L}} \sum_{i=1}^{N_{y_L}} \left(\left| y_L^* - y_L \right|^2 \right), \\ MSE_{y_H} &= \frac{1}{N_{y_H}} \sum_{i=1}^{N_{y_H}} \left(\left| y_H^* - y_H \right|^2 \right). \end{aligned} \tag{4}$$

2.2. Infilling strategy

Our proposed MFDNN-based optimization framework aims to search the optimal solution efficiently by the high-fidelity prediction with assistance of low-fidelity data. Specifically, a limited amount of high-fidelity samples and a lot of cheap low-fidelity samples are adopted. During each optimization iteration, the MFDNN surrogate model will be updated by infilling new samples if the convergence condition is not achieved. The aim of infilling strategy

is to efficiently find the minimum of expensive cost functions. So, both exploration (sampling from areas of high uncertainty) and exploitation (sampling areas likely to offer improvement over the current best observation) [28] should be considered. However, the MFDNN model cannot provide the uncertainty quantification. So, the MFDNN model cannot be updated directly according to the uncertainty of the surrogate model. Therefore, we need to adopt a special infilling strategy to improve the accuracy of the MFDNN model during the optimization. Specifically, the high-fidelity infilling strategy is by adding the current optimal solution of the surrogate model into the high-fidelity database. The optimal solution of the surrogate model is obtained by a gradient-free optimization algorithm named PSO algorithm (see the Appendix). For the low-fidelity infilling strategy, we generate the solutions distributed uniformly in the whole design space to explore the unknown landscape of design space by a lot of cheap low-fidelity data, which can capture the complicated function changes, such as, peak, valley and etc. Euclidean Distance is adopted in this study to measure the crowding degree between each other samples, which is expressed as:

$$d\left(x^{(i_1)}, x^{(i_2)}\right) = \sqrt{\sum_{j=1}^{k}\left(x_j^{(i_1)} - x_j^{(i_2)}\right)^2} \tag{5}$$

where $x^{(i_1)}$ and $x^{(i_2)}$ denote two different sample points, and index j denotes jth dimension. The low-fidelity infilling strategy is actually an optimization problem that aims to maximize the minimum Euclidean Distance between the newly added samples and the existed high-fidelity and low-fidelity samples. It can be expressed as:

$$\max \quad \min d\left(x^{new},\ X\right) \qquad X\ is\ [X_{lo},\ X_{hi}]\,. \tag{6}$$

We still adopt the PSO algorithm to handle this optimization problem. This infilling strategy compared to random infilling strategy is shown in Fig. 3.

2.3. Geometry representation method

In this subsection, we focus on the geometry representation method for a airfoil and a wing. The wing shape consists of the planform shape and the sectional shapes. When the control sections are determined, the wing shape can be created by the interpolation of the control sections with the planform parameters. The control sections are the cross-sectional shapes of a wing, which are called airfoils. In the present work, several parametric methods have been put forward. For example, Hicks–Henne bump functions method [29], Parsec method [30], NURBS method [31], Class function/Shape function Transformation (CST) method [32], Bézier method [33], Free Form Deformation method [34]. In this work, the CST method is applied to parameterize the airfoil because of less design parameters and higher accuracy in geometric fitting.

2.3.1. CST method

The upper and lower surfaces of the airfoil can be expressed by following formula.

$$\overline{y} = C(\overline{x}) \cdot S(\overline{x}) + \overline{x} \cdot \Delta y_{te}. \tag{7}$$

The CST method can be described by the mathematical expression as follows:

$$\begin{aligned} &The\ upper\ surface:\ \overline{y}_u = C(\overline{x}) \cdot S(\overline{x}) + \overline{x} \cdot \Delta y_{te,u}, \\ &The\ lower\ surface:\ \overline{y}_l = C(\overline{x}) \cdot S(\overline{x}) + \overline{x} \cdot \Delta y_{te,l}. \end{aligned} \tag{8}$$

The class function is defined as follows:

$$C(\overline{x}) = (\overline{x})^{N_1} \cdot (1-\overline{x})^{N_2}, \quad 0 \leq \overline{x} \leq 1. \tag{9}$$

Different the number of N_1, N_2 can express different geometry. The paper defines $N_1 = 0.5$, $N_2 = 1.0$.
The shape function is defined as follows:

$$(\overline{x}) = \sum_{i=0}^{N_b}[A_t \cdot S_t(\overline{x})] = \sum_{i=0}^{N_b}[A_t \cdot K_{r,N_b} \cdot (\overline{x})^r \cdot (1-\overline{x})^{N_b - r}], \tag{10}$$

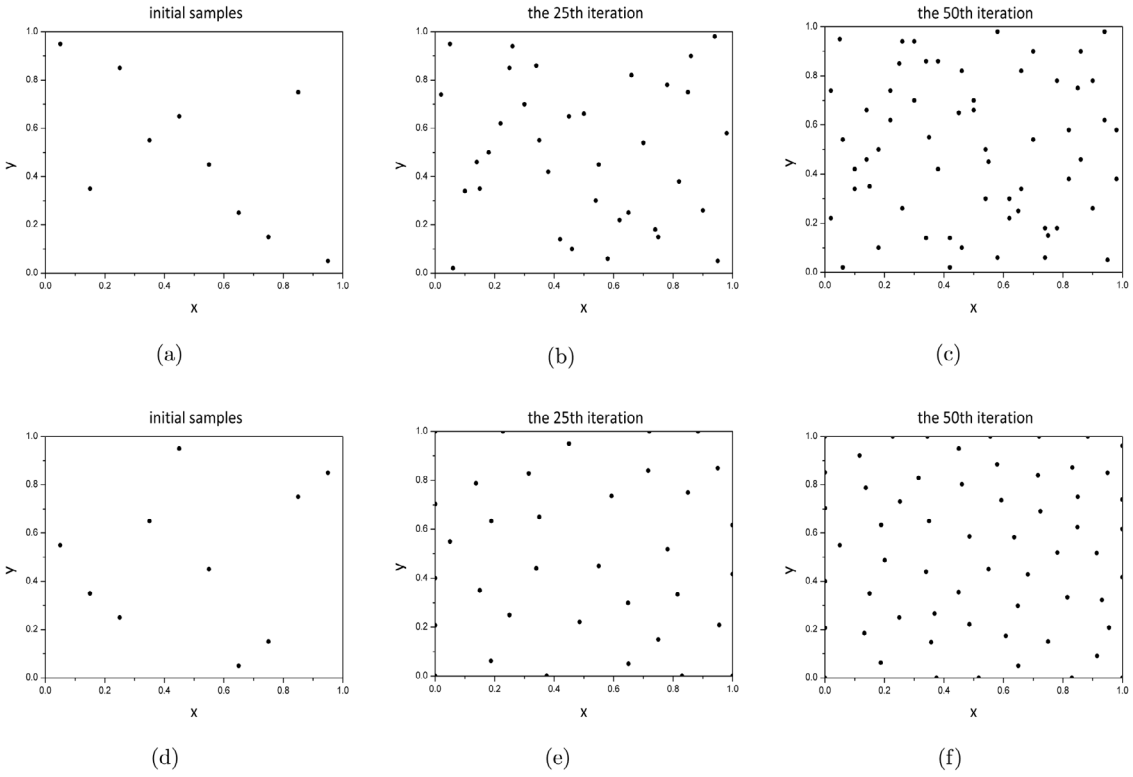


Fig. 3. The comparison of two infilling strategies. (a), (b), (c) are initial samples, infilling samples after 25th iteration and infilling samples after 50th iteration respectively by random infilling strategy. (d), (e), (f) are initial samples, infilling samples after 25th iteration and infilling samples after 50th iteration respectively by the infilling strategy in this study.

where A_t is control coefficients of shape function, N_b is the order of Bernstein polynomial, K_{r,N_b} is the coefficients of Bernstein polynomial, which can be expressed by following equation.

$$K_{r,N_b} = \binom{N_b}{r} = \frac{N_b!}{r!\,(N_b - r)}. \tag{11}$$

So, given an argument A_t, the CST method can output certain airfoil coordinates. On the contrary, given certain airfoil coordinates, the weight coefficient A_t can be determined by optimization algorithms.

$$\begin{aligned} F = &\sum_{j=1}^{n_{up}} \left\{ \left[C\left(\overline{x}\right) \cdot \left(\sum_{i=0}^{N_b} A_{i,u} \cdot S_i\left(\overline{x}\right) \right) + \overline{x} \cdot \Delta y_{te,u} \right] - \overline{y}_{u,j} \right\}^2 \\ &+ \sum_{j=1}^{n_{low}} \left\{ \left[C\left(\overline{x}\right) \cdot \left(\sum_{i=0}^{N_b} A_{i,l} \cdot S_i\left(\overline{x}\right) \right) + \overline{x} \cdot \Delta y_{te,l} \right] - \overline{y}_{l,j} \right\}^2. \end{aligned} \tag{12}$$

For example, we use the CST method to parameterize the RAE2822 airfoil. Here, the L-BFGS [35] optimization algorithm is employed to fit the CST parameters. The fitting precision is summarized in Table 1 with various orders of Bernstein polynomial. It is seen that the order of Bernstein polynomial can increase the fitting precision of CST method (see Fig. 4).

2.3.2. Wing shape parameterized method

For the planform parameters of a wing, there are six independent parameters to describe the shape of a trapezoidal wing: the aspect ratio, the root-to-tip ratio, the leading-edge sweep angle, the dihedral angle, the geometric twist angle and the elastic twist angle.

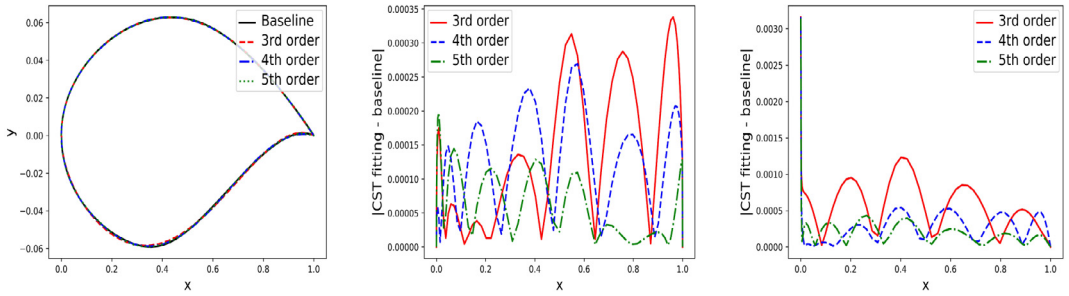


Fig. 4. The comparison of fitting results using various Bernstein polynomial orders. (a) The fitting shapes using 3rd order, 4th order and 5th order respectively and baseline. (b) The fitting error distribution on the upper surface. (c) The fitting error distribution on the lower surface.

Table 1
The effect of various Bernstein polynomial orders.

Bernstein orders	Third order	Fourth order	Fifth order
CST parameter No.	8	10	12
MSE of CST fitting	0.00593558	0.00413384	0.00365861

We focus on the wing shape of the DLR-F4 wing-body configuration(1st AIAA CFD Drag Prediction Workshop, DPW-I). Here, four sections are used to loft the wing, with the first section using the Ha5 airfoil and the other sections using the R4 airfoil. For each airfoil, the CST method with 12 parameters is used. As for the planform shape, only the variable aspect ratios (λ) and root-to-tip ratios (η) are defined, leading to six parameters, as shown in Fig. 5 and Eq. (13).

$$\begin{aligned} \lambda_1 &= 2 \cdot b_1 / (C_1 + C_2), \qquad \eta_1 = C_1/C_2, \\ \lambda_2 &= 2 \cdot b_2 / (C_2 + C_3), \qquad \eta_2 = C_2/C_3, \\ \lambda_3 &= 2 \cdot b_3 / (C_3 + C_4), \qquad \eta_3 = C_3/C_4. \end{aligned} \tag{13}$$

So, there are totally 30 parameters to specify the wing shape in the DLR-F4 wing-body configuration. As seen in Fig. 6, the geometry parameterization is packaged on the FreeCAD platform .

2.4. Numerical simulation framework for transonic flow

For a transonic flow, the inflow Mach number is in the range of 0.72 to 1.0 and the local shock appears in the local supersonic region and subsonic region. First, we consider a compressible Navier–Stokes equation in a Cartesian reference frame to solve the fluid dynamic problem and ignore the body forces and heat sources.

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \left(\mathbf{F} - \mathbf{F}^{v}\right)}{\partial x} + \frac{\partial \left(\mathbf{G} - \mathbf{G}^{v}\right)}{\partial y} + \frac{\partial \left(\mathbf{H} - \mathbf{H}^{v}\right)}{\partial z} = 0, \tag{14}$$

where $\mathbf{Q}$ is the conservative variables, $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{H}$ are the inviscid flux terms in the x-, y- and z-directions respectively, and $\mathbf{F}^{v}$, $\mathbf{G}^{v}$, and $\mathbf{H}^{v}$ are viscid flux terms in the x-, y- and z-directions respectively. To simulate the turbulence flow with the effect of high Reynolds number, the Reynolds-averaged Navier–Stokes method and eddy viscosity model are applied. The Spalart–Allmaras (SA) model [36] is used to evaluate the evolution of the turbulent kinematic viscosity in the Reynolds stresses modeling.

In the present study, a finite-volume method is applied to discretize the governing equations. The implicit Lower-upper symmetric-Gauss–Seidel (LU-SGS) method [37] is employed for the time discretization. The integral form of the flow equations is shown as

$$\frac{d}{dt}\int_{V} \mathbf{Q} dV + \int_{S} \mathbf{F}\left(\mathbf{Q}\right) \cdot d\mathbf{S} = 0, \tag{15}$$

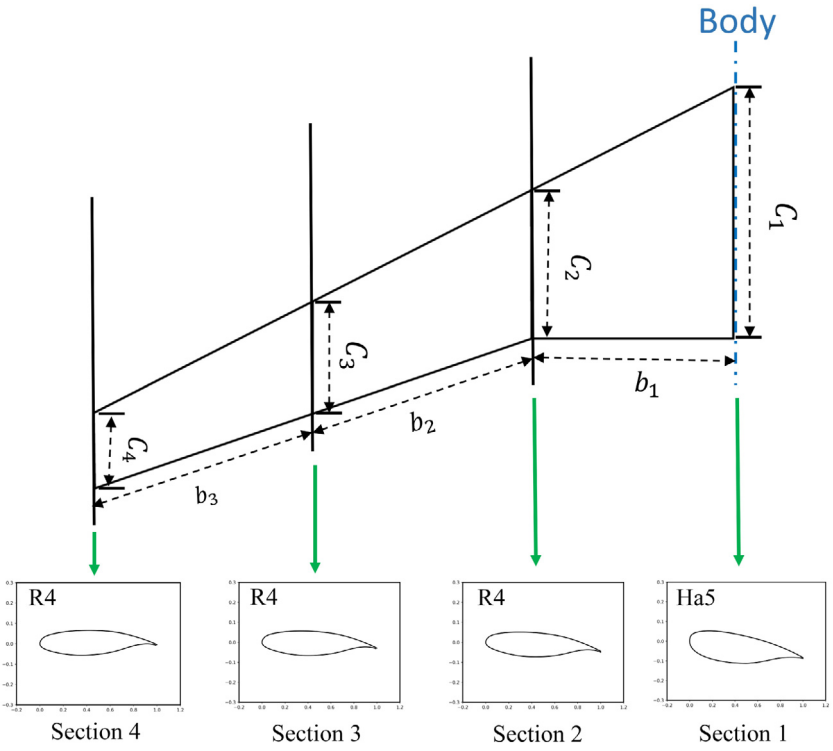


Fig. 5. The DLR-F4 wing geometry. The DLR-F4 wing is originally defined with four stations, and they are linearly lofted between the stations, where the first section is the Ha5 airfoil and the other sections are the R4 airfoil.

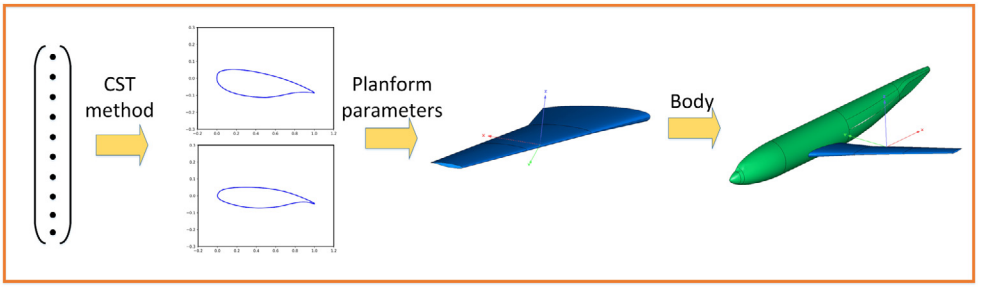


Fig. 6. Parameterized geometric modeling of the wing-body configuration. The CST method is employed to parameterize the wing sections. The wing-body configuration is created on the FreeCAD platform.

for an arbitrary fixed region V with a boundary S. And the integration for a computational cell i can be expressed:

$$V_i \frac{dU_i}{dt} = -\sum_{faces} S\widehat{\mathbf{F}}. \tag{16}$$

The flux term $\mathbf{F}$ on the cell face is

$$\widehat{\mathbf{F}} = n_x\mathbf{F} + n_y\mathbf{G} + n_z\mathbf{H}, \tag{17}$$

where n_x, n_y, and n_z are the components of the unit normal vector in the x-, y- and z-directions respectively. In order to capture the shock and discontinuity with high-resolution, the Roe Flux-Difference Splitting Scheme [38] is used to evaluate the inviscid fluxes. The flux is calculated as

$$\widehat{\mathbf{F}} = T^{-1}\mathbf{F}\left(T\mathbf{Q}\right). \tag{18}$$

Here, matrix T is a rotation matrix which transforms the dependent variables to a local co-ordinate system normal to the cell surface. So, the Cartesian form $\mathbf{F}$ of the flux is only needed.

$$\mathbf{F}\left(\mathbf{Q}^l, \mathbf{Q}^r\right) = \frac{1}{2}\left[\mathbf{F}\left(\mathbf{Q}^l\right) + \mathbf{F}\left(\mathbf{Q}^r\right)\right] - \frac{1}{2}|A|\left(\mathbf{Q}^l - \mathbf{Q}^r\right), \tag{19}$$

where Q^l and Q^r are the solution vectors evaluated on the left and right sides of the cell surface, A is the Jacobian matrix $A = \partial\mathbf{F}/\partial\mathbf{Q}$. We use the Monotone Upstream-centered Schemes for Conservation Laws (MUSCL) method [39] to evaluate Q^l and Q^r. In the interpolation at the region with local shock, the limiter is needed. The following expression is used for a generic element q of the conserved variable vector Q.

$$\begin{aligned} {q^l}_k &= q_i + \phi_i \cdot \nabla q_i \cdot \overrightarrow{r}_{ki}, \\ {q^r}_k &= q_j + \phi_j \cdot \nabla q_j \cdot \overrightarrow{r}_{kj}, \end{aligned} \tag{20}$$

where ∇q_i and ∇q_j are the gradients computed for the ith cell and its neighboring jth cell respectively, ${\mathbf{Q}^l}_k$ and ${\mathbf{Q}^r}_k$ are the distance vectors from the ith and mth cells, respectively, to the kth face, and ϕ_i and ϕ_j denote the limiters in these cells. We employ the Venkatakrishnan limiter [40] to avoid large gradient near the local shock wave. The limiter is expressed as

$$\phi_j = \begin{cases} \min \ \ Lim(\Delta q_i^{\max}, \Delta q_{ji}), & \Delta q_{ji} > 0 \\ \min \ \ Lim(\Delta q_i^{\min}, \Delta q_{ji}), & \Delta q_{ji} < 0 \\ 1, & \Delta q_{ji} = 0 \end{cases}, \tag{21}$$

where

$$\begin{aligned} \Delta q_{ji} &= \nabla q \cdot \overrightarrow{r}_{ji}, \\ \Delta q_i^{\max} &= q_i^{\max} - q_i, \\ \Delta q_i^{\min} &= q_i^{\min} - q_i, \\ q_i^{\max} &= \max\left(q_i,\ q_{neighbors}\right), \\ q_i^{\min} &= \min\left(q_i,\ q_{neighbors}\right), \end{aligned} \tag{22}$$

The Lim function is expressed as

$$Lim\,(a, b) = \frac{a^2 + 2ab + \varepsilon^2}{a^2 + ab + 2b^2 + \varepsilon^2}, \tag{23}$$

where $\varepsilon^2 = \left(K\overline{\Delta}\right)^3$, K is a constant, and $\overline{\Delta}$ represents grid scale.

The numerical solution of the flow governing equations with the numerical method mentioned above is provided using the open source libraries of SU2 based on a finite volume approach. Forces are numerically calculated integrating the pressure distribution on the wing surface.

3. Validation and verification

In this section, two standard synthetic benchmarks are used to validate the regression and optimization effects of the MFDNN-based optimization framework.

3.1. One-dimensional multi-fidelity regression

Here, a one-dimensional test function is used to validate the MFDNN model. The high-fidelity and low-fidelity data are given by following functional expression, respectively.

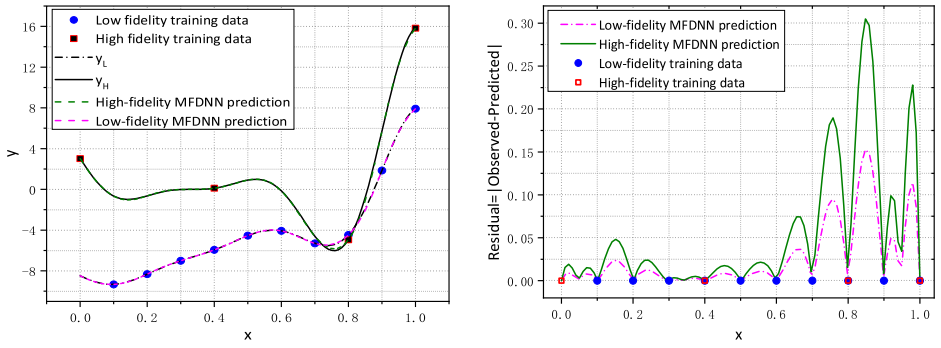
$$\begin{aligned} y_L\,(x) &= A(6x-2)^2 \sin\,(12x-4) + B\,(x-0.5) + C, \quad x \in [0,\ 1], \\ y_H\,(x) &= (6x-2)^2 \sin\,(12x-4), \end{aligned} \tag{24}$$

where the parameters are defined by $A = 0.5, B = 10, C = -5$. The low-fidelity training samples are located at $x_L = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, and the high-fidelity training samples are located at $x_H = \{0, 0.4, 0.8, 1.0\}$.

In this case, as summarized in Table 2, six hidden layers and thirty-two neurons per layer are adopted in NN_L for approximating the low-fidelity data. While, one hidden layer and sixteen neurons per layer are adopted in NN_{H1} for approximating the linear correlation between these two-fidelity data. Moreover, two hidden layers and sixteen

Table 2
Parameters used in the MFDNN model.

Neural network	Layers	Neurons per layer	Activation	Regularization
NN_L	6	32	Tanh	0.0001
NN_{H1}	1	16	None	None
NN_{H2}	2	16	Tanh	0.0001



(a) The MFDNN predictions (b) The corresponding error distribution

Fig. 7. The MFDNN prediction results. (a) The dark green and purple lines denote the high-fidelity and low-fidelity predictions of MFDNN, respectively. The results are compared to the exact low-fidelity and high-fidelity functions y_L and y_H. (b) The corresponding error distribution between the MFDNN predictions and the training data. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

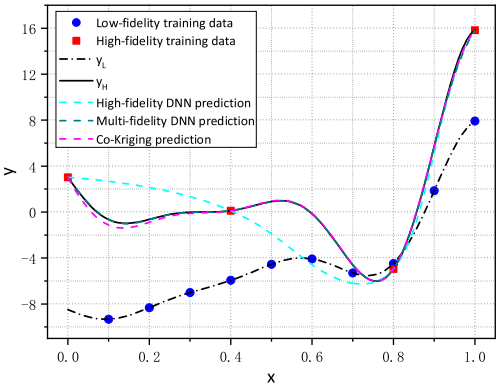


Fig. 8. Comparison of the MFDNN model (dark green dash line), single-fidelity DNN model (light blue dash line) and Co-Kriging model (purple dash line) [41] using the same training data with four high-fidelity data (red squares) and ten low-fidelity data (blue circles). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

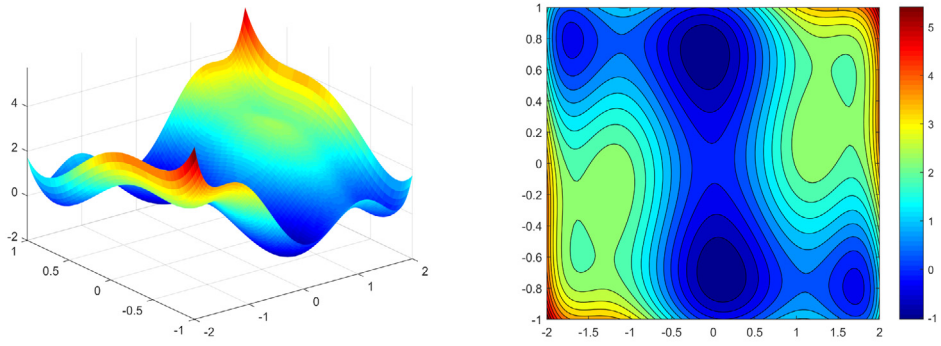
neurons per layer are adopted in NN_{H2} for approximating the nonlinear correlation between the two fidelity data. The hyperbolic tangent function is employed as the activation function in NN_L and NN_{H2}. Especially, there is no activation function used in the neural network NN_{H1}. The regularization is used in the neural network NN_L and NN_{H2}. The learning rate is set as 0.0001, and the regularization rate is set to 0.0001. The loss function is optimized using the Adam method together with Xavier's initialization method.

Fig. 7 shows the low-fidelity training data, the high-fidelity training data and the exact low-fidelity and high-fidelity functions y_L and y_H. The MFDNN prediction through the high-fidelity and low-fidelity training data gives a good approximation to the exact high-fidelity and low-fidelity functions.

Then, we compare the predictions of our MFDNN model with that of the Co-Kriging and only high-fidelity DNN model. As seen in Fig. 8, the MFDNN prediction outperforms the other two algorithms, which means the

Table 3
Comparison of $L2$ error with various models, each algorithm runs 30 times repeatedly.

Model	Hyper-parameters optimizer	$L2$ error mean	$L2$ error std.
High-fidelity DNN	Adam method	0.502982137	0.0306676
Multi-fidelity DNN	Adam method	0.024068781	0.0083567
Co-Kriging	L-BFGS	0.042009379	2.077×10^{-5}



(a) Landscape (b) Contour

Fig. 9. The six-hump camelback test function. (a) The landscape of six-hump camelback test function. (b) The contour of six-hump camelback test function. The six-hump camelback function has two global optimal solution at $x_1^* = (0.0898,\ -0.7126)$ and $x_2^* = (-0.0898,\ 0.7126)$, with the same optimal value $f\left(x_1^*\right) = f\left(x_2^*\right) = -1.0316285$.

MFDNN model have learned the correlation between the low-fidelity and high-fidelity data well. More specifically, the relative $L2$ errors with various models are quantified as:

$$L_2\ error = \sqrt{\frac{\sum_j \left(y_j^* - y_j\right)^2}{\sum y_j^2}}. \tag{25}$$

Each algorithm runs 30 times repeatedly. As seen in Table 3, by including the low-fidelity data, the $L2$ error of both the Co-kriging model and the MFDNN model are reduced, while the MFDNN model is more remarkable.

Although we have only considered combining two fidelity levels of analyses, the MFDNN model can be extended to multiple fidelity levels by using additional neural networks.

3.2. Six-hump camelback test function optimization

In this subsection, a six-hump camelback function is adopted to test the MFDNN-based optimization framework. The optimization problem is defined as:

$$\begin{aligned} &\min\ \ y_H(x) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + \left(-4 + 4x_2^2\right)x_2^2, \\ &w.r.t\ \ x_1 \in [-2,\ 2],\ \ x_2 \in [-1,\ 1]. \end{aligned} \tag{26}$$

The low-fidelity function is used to construct the MFDNN model, and it can be expressed as:

$$y_L(x) = 0.75y_H(x) + 0.375y_H(x_1 + 0.1,\ x_2 - 0.1) - 0.125, \tag{27}$$

As shown in Fig. 9, the exact solutions of this optimization problem are: $x_1^* = (0.0898,\ -0.7126)$ and $x_2^* = (-0.0898,\ 0.7126)$. The corresponding optimal value is $f\left(x_1^*\right) = f\left(x_2^*\right) = -1.0316285$.

Firstly, as shown in Table 4, the single high-fidelity optimization with a feed-forward neural network surrogate model is adopted. Specifically, two hidden layers and sixteen neurons per layer are adopted for approximating the high-fidelity data. The hyperbolic tangent function is employed as the activation function. The learning rate is set

Table 4
Parameters configuration for the DNN and MFDNN model.

Model	Neural network	Layers	Neurons per layer	Activation	Regularization
DNN	NN_{H2}	2	16	Tanh	0.0001
MFDNN	NN_L	6	32	Tanh	0.0001
	NN_{H1}	1	16	None	None
	NN_{H2}	2	16	Tanh	0.0001

Table 5
Comparison of the optimization results by the DNN and MFDNN model, respectively. In the MFDNN-based optimization, The infilling strategy is to add one high-fidelity sample and two low-fidelity samples in each iteration, while no low-fidelity sample is added in the DNN-based optimization.

Model	Number of high -fidelity samples	Number of low -fidelity samples	Optimal value mean	Optimal value std.	Error
DNN	5+25	0	−1.029120	3.160×10^{-4}	2.4313‰
MFDNN	5+25	10+75	−1.031041	5.597×10^{-4}	0.5697‰

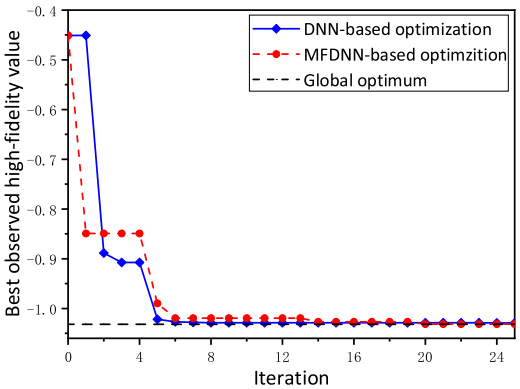


Fig. 10. Comparison of the DNN-based optimization and MFDNN-based optimization. The actual optimal value is −1.0316285. In the DNN optimization, five initial high-fidelity samples are used while in the MFDNN optimization, extra ten low-fidelity samples are included. In both optimization frameworks, the maximum iteration is set as twenty-five.

as 0.001, and the regularization rate is set to 0.0001. The loss function is also optimized by using the Adam method with Xavier's initialization method.

Then, the MFDNN based optimization framework is applied. Specifically, the structure of MFDNN is the same as the front one-dimensional multi-fidelity regression case. In the DNN based optimization, five initial high-fidelity samples are used while in the MFDNN optimization, extra ten low-fidelity samples are included. In the both optimization frameworks, the maximum iteration is set as twenty-five. In each iteration, one high-fidelity sample and two low-fidelity samples are added to update the surrogate model.

As seen in Fig. 10, the DNN-based optimization framework is faster than the MFDNN-based optimization framework in the early-stage optimization, while the MFDNN-based optimization framework converges more accurately in the later-stage optimization. Table 5 summarizes the corresponding details, including the number of initial samples, the number of high-fidelity and low-fidelity evaluations, and the optimal solution. Each algorithm runs 20 times repeatedly. The mean of optimum by MFDNN-based optimization is 5.697×10^{-4}, while the single fidelity DNN-based optimization is 2.431×10^{-3}.

Moreover, the effect of the low-fidelity samples in the optimization is examined. Here, various low-fidelity infilling strategies with 1HF+2LF, 1HF+3LF, 1HF+4LF are compared, where 'LF' stands for the number of low-fidelity samples while 'HF' denotes the number of high-fidelity samples. As seen in Fig. 11 and Table 6, by adding

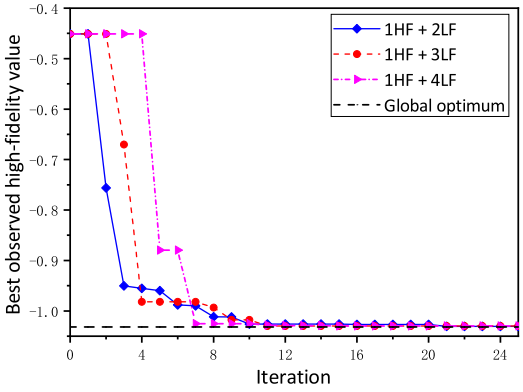


Fig. 11. The effect of the number of the low-fidelity infilling samples.

Table 6
Comparison of optimization results with various number of the low-fidelity infilling samples. Here, various low-fidelity infilling strategy with 1HF + 2LF, 1HF + 3LF, 1HF + 4LF are compared, where 'LF' stands for the number of low-fidelity samples while 'HF' stands for the number of high-fidelity samples.

Infilling strategy	Number of HF samples	Number of LF samples	Optimal value mean	Optimal value std.	Error
1HF + 2LF	5+25	10+50	−1.031041	5.597×10^{-4}	0.5697‰
1HF + 3LF	5+25	10+75	−1.030677	7.712×10^{-4}	0.9228‰
1HF + 4LF	5+25	10+100	−1.029562	2.884×10^{-3}	2.0033‰

1HF+2LF, the MFDNN model provided the best approximation to the exact global optimal value of −1.0316285. But the convergence speed is slowest. It is conjectured that by adding more low-fidelity samples, the predictions of MFDNN model prefer to low-fidelity responses and reduce the surrogate accuracy.

4. Results and discussions

In this section, the MFDNN-based optimization framework is applied to the aerodynamic shape optimization of an airfoil and a wing-body configuration.

4.1. Lift–drag ratio maximization of RAE2822 airfoil in transonic flow

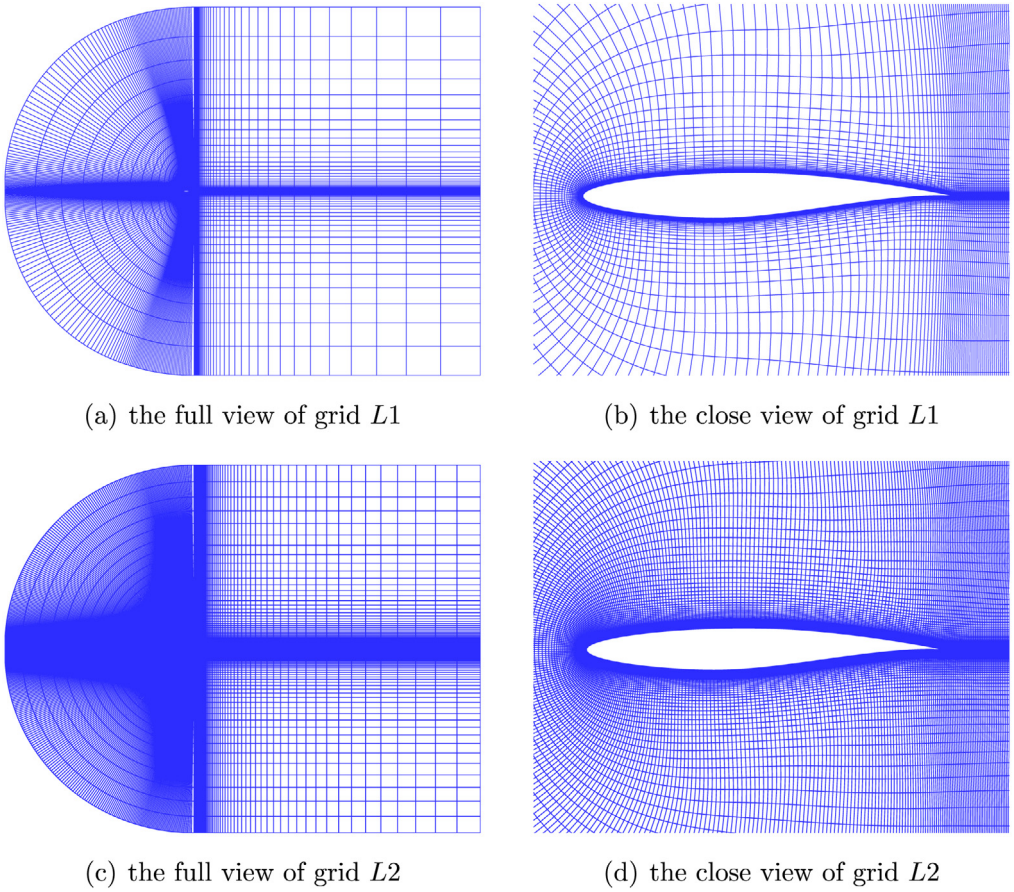
In this subsection, the optimization objective is to maximize lift–drag ratio of a RAE2822 airfoil at the free-stream condition of $Ma = 0.72$, $Re = 6.5 \times 10^6$, and $\alpha = 2.31°$. The optimization problem can be formalized as following:

$$maximize \quad C_l/C_d. \tag{28}$$

4.1.1. Low and high fidelity information sources

In the current study, the MFDNN model relies on two different levels of fidelity data. The high-fidelity data and the low-fidelity data use the same solver but with different resolution meshes. Two different meshes with C-type topology are generated as shown in Fig. 12. We select grid $L1$ with 24 480 cells for low-fidelity CFD model and grid $L2$ with 75 640 cells for high-fidelity CFD model respectively. In order to simulate the boundary layer flow properly, the height of the first layer grid near the wall is 4×10^{-6} m to satisfy the condition of $y^+ < 1$ in both meshes. The simulations are both conducted on a cluster with 28 cores.

Moreover, the Jameson–Schmidt–Turkel (JST) [42] method is employed for the space discretization. Fig. 13 shows the pressure coefficient distributions of the RAE2822 airfoil under two different resolutions. The results are consistent with the experiments. As seen in Table 7, the error of fine grid $L2$ is much smaller than that of coarse grid $L1$, while the computation cost is about 4 times that of coarse grid $L1$.



(a) the full view of grid $L1$ (b) the close view of grid $L1$

(c) the full view of grid $L2$ (d) the close view of grid $L2$

Fig. 12. Sketch of various computational grids. Grid $L1$ has 24 480 cells with $y^+ < 1$, and Grid $L2$ has 75 640 cells with $y^+ < 1$.

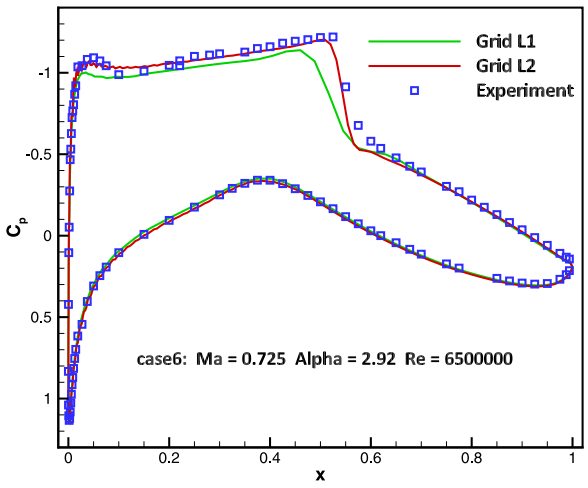


Fig. 13. Comparison of the pressure coefficient Cp distributions by different resolution grids and experiments [43] for the RAE2822 airfoil. The experiment case with $\alpha = 2.92°$, $Ma = 0.725$, $\mathrm{Re} = 6.5 \times 10^6$ in Ref. [43] is selected to verify CFD results of Grid $L1$ (green line) and Grid $L2$ (red line). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 7
Comparison of the aerodynamic coefficients and computation time for the RAE2822 airfoil. The experiment results are: $C_l = 0.743$, $C_d = 0.0127$ [43].

Grid	C_l	Error(C_l)	C_d	Error(C_d)	CPU time
Grid $L1$	0.683038	8.07%	0.013803	8.69%	3 min 52 s
Grid $L2$	0.735923	0.95%	0.013535	6.57%	13 min 19 s

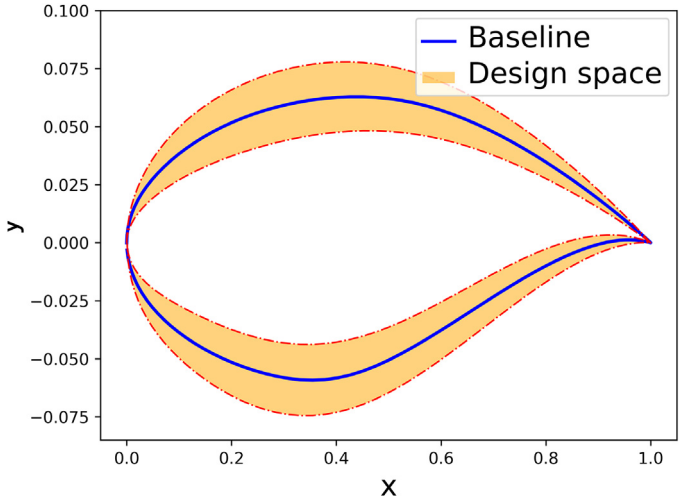


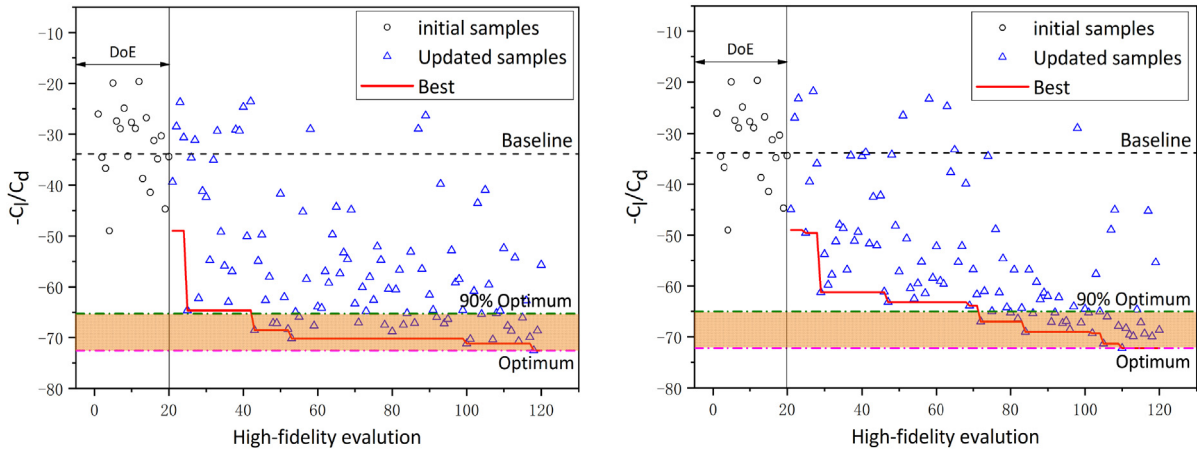
Fig. 14. Determination of design space for the RAE2822 aerodynamic shape optimization. The blue line represents the RAE2822 airfoil and the design space is highlighted in the orange shades.

4.1.2. Multi-fidelity optimization results

For the RAE2822 optimization case, the design space is shown in Fig. 14 where we highlighted in the orange shades. Twenty initial samples by using Latin Hypercube Sampling (LHS) method are selected for the high-fidelity database by using the fine grid $L2$, and forty initial sample points are selected for the low-fidelity dataset by using the coarse grid $L1$. For the single-fidelity optimization, only the high-fidelity database is used.

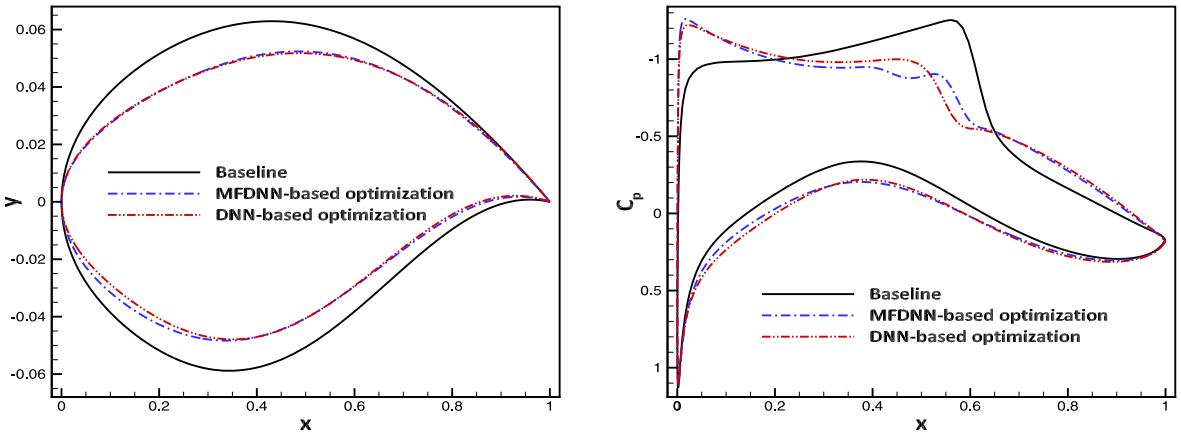
The optimization of the RAE2822 case takes 1 days 10 h 28 min using 56 cores distributed a computer cluster with 28 cores for the high-fidelity model and 28 cores for the low-fidelity model. During the optimization, one high-fidelity and two low-fidelity samples are added in parallel simultaneously in each iteration. Since the evaluation time of low-fidelity samples is much less than high-fidelity samples, the MFDNN-based optimization and DNN-based optimization take the same amount of time in each iteration. The optimization time of the MFDNN-based and DNN-based optimization only depends on the number of high-fidelity evaluations. The convergence histories of the objective function are sketched in Fig. 15. In order to eliminate the influence of initial samples, the high-fidelity initial samples are the same for the MFDNN-based optimization and DNN-based optimization. As Fig. 15 shows, after the 42nd iteration, the best objective value of MFDNN-based optimization is down to 90% of optimum, while the DNN-based optimization needs 72 iterations. It is obvious that the MFDNN-based optimization can achieve 90% of optimum more quickly, compared to the DNN-based optimization. As shown in Table 8, for the single-fidelity DNN-based optimization, the lift–drag ratio is increased by 113.11% with total 120 high-fidelity samples. While, the lift–drag ratio is increased by 114.07% for the MFDNN-based optimization, where totally 120 high-fidelity and 240 low-fidelity samples are evaluated. By comparison, the MFDNN-based optimization method can find a better optimal solution with the same number of high-fidelity evaluations, compared to the single-fidelity DNN-based optimization method.

The optimal results are compared in Fig. 16. It is seen that the optimal airfoils with the MFDNN-based and DNN-based optimization methods are much thinner than the baseline. The corresponding pressure coefficient distributions are shown in Fig. 16(b). It is distinct that the shock wave of the optimum airfoils moves towards the leading edge of the airfoils while with a weaker strength by comparing to that of the baseline.



(a) Multi-fidelity optimization (b) Single-fidelity optimization

Fig. 15. Convergence histories of the optimizing RAE2822 airfoil using the MFDNN-based and DNN-based methods. Black circles denote the initial samples via LHS method and blue triangles denote the updated samples in each iteration. The best response values from the optimization are indicated by red line while the response of baseline (black dash line) is evaluated by CFD results with high-fidelity grid.

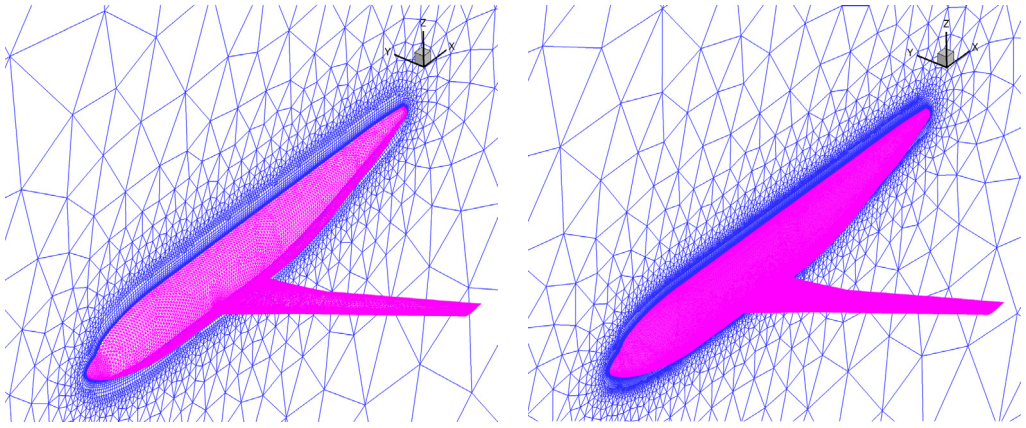


(a) Comparison of the geometic shape of the optimal airfoils and the baseline (b) Comparison of surface pressure distributions of baseline and optimal airfoil

Fig. 16. Comparison of the optimal airfoils and the baseline.

Table 8
Optimization results for the RAE2822 case. The baseline is set as $C_l/C_d = 33.89348$. Here, 20 initial high-fidelity samples for the DNN-based optimization. Extra 40 initial low-fidelity samples are used for the MFDNN-based optimization. The maximum iteration of multi-fidelity optimization is set as 100. In each iteration, one high-fidelity sample and two low-fidelity samples are added.

Optimization method	Initial samples		No. of high-fidelity evaluations	No. of low-fidelity evaluations	Optimum C_l/C_d	Increase of C_l/C_d
	High-fidelity	Low-fidelity				
MFDNN-based optimization	20	40	120	240	72.557	114.07%
DNN-based optimization	20	0	120	0	72.230	113.11%



(a) Grid $L1$ (b) Grid $L2$

Fig. 17. Sketch of various CFD computation grids. Grid $L1$ has 829 914 cells with $y^+ \approx 10$, and Grid $L2$ has 5 451 299 cells with $y^+ < 1$.

4.2. Drag minimization of DLR-F4 wing-body configuration in the transonic flow

In this subsection, the drag minimization of the DLR-F4 wing-body configuration in the transonic flow is studied. The optimization objective is to minimize drag coefficient of the DLR-F4 wing-body configuration at free-stream condition of $Ma = 0.75$, $Re = 3.0 \times 10^6$, and $\alpha = 0°$. Four constraints are applied, including the lift coefficient, the maximum thickness of the first section and the second section, and the area of the wing. The optimization problem can be summarized as:

$$\begin{aligned} minimize: \quad & Cd, \\ s.t. \quad & C_l \geq C_{l,baseline}, \\ & Thickness_{\text{Section 1}} \geq Thickness_{0,\text{ Section 1}}, \\ & Thickness_{\text{Section 2}} \geq Thickness_{0,\text{ Section 2}}, \\ & Area \geq Area_0. \end{aligned} \tag{29}$$

4.2.1. Low and high fidelity information sources

For multi-fidelity modeling, the CFD evaluations are conducted using the same CFD solver with fine and coarse unstructured grids, respectively. For the low resolution grid, the value of y^+ is 10, and the grid growth rate of viscous layer is set to 1.3. While, for the high resolution grid $L2$, the height of the first layer near the wall is 1.2×10^{-6} m to satisfy the condition of $y^+ < 1$. The grid growth rate of viscous layer is set as 1.2. As shown in Fig. 17, the coarse grid $L1$ with 829 914 cells is applied to the low-fidelity CFD model and the fine grid $L2$ with 5 451 299 cells is applied to the high-fidelity CFD model. The results of the grid convergence study are shown in Figs. 18 and 19, and they are consistent to the experimental results [44]. Here, the CFD computations are conducted on a cluster with 392 cores for the high-fidelity model and 112 cores for the low-fidelity model.

4.2.2. Multi-fidelity optimization results of DLR-F4 wing-body configuration

The design space of various wing sections is present in Fig. 20, which are highlighted in the orange shades, with an amplitude of 0.2 by comparing to the bound of the standard DLR-F4 wing. Initially, 60 high-fidelity sample points and 120 low-fidelity sample points are selected by LHS method to construct the multi-fidelity deep neural network surrogate model. For single-fidelity deep neural network, only the high-fidelity samples are used.

The optimization of the DLR-F4 case takes 4 days 15 h 23 min using 504 cores distributed a computer cluster with 392 cores for the high-fidelity model and 112 cores for the low-fidelity model. In the optimization, the maximum

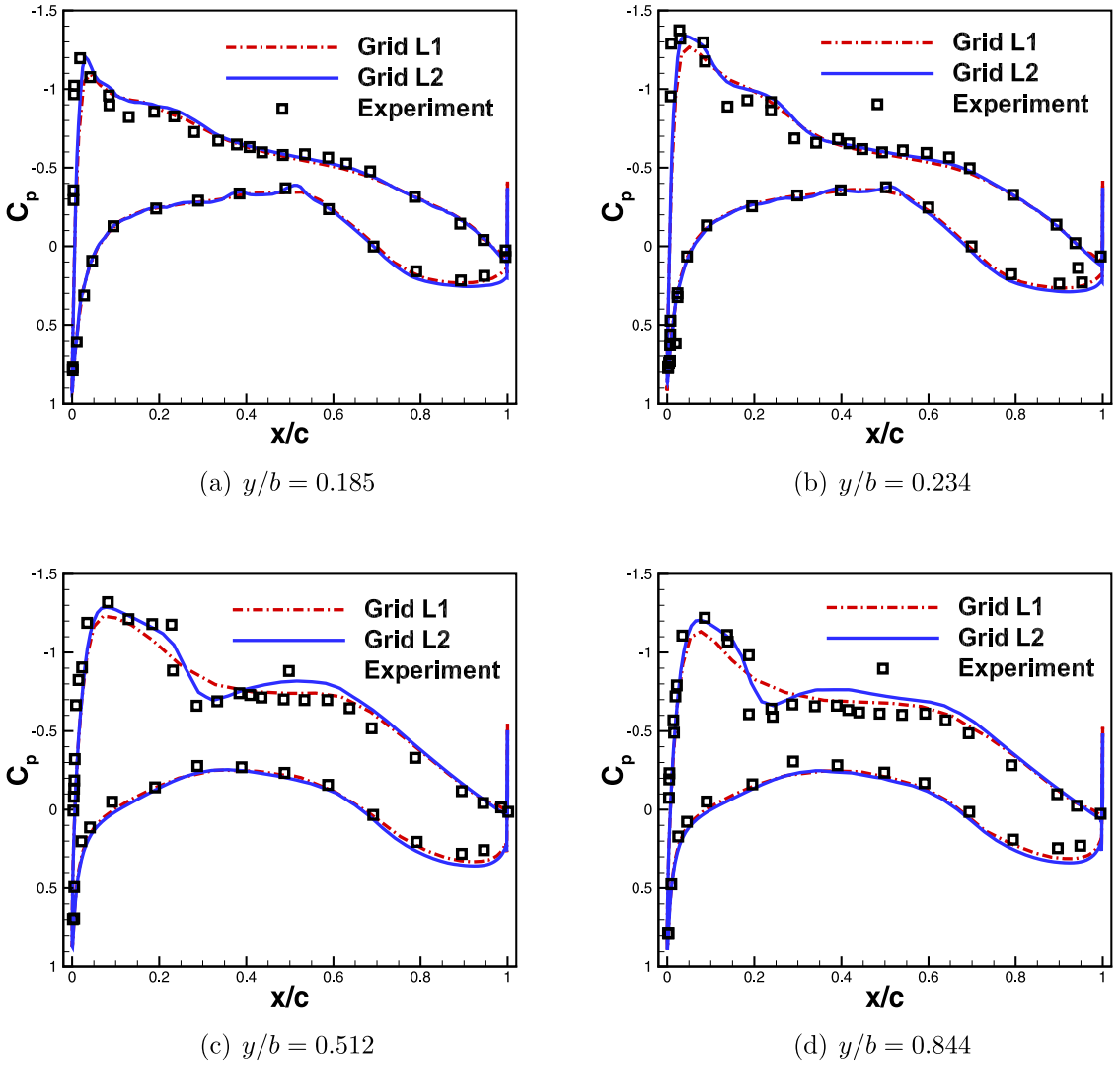
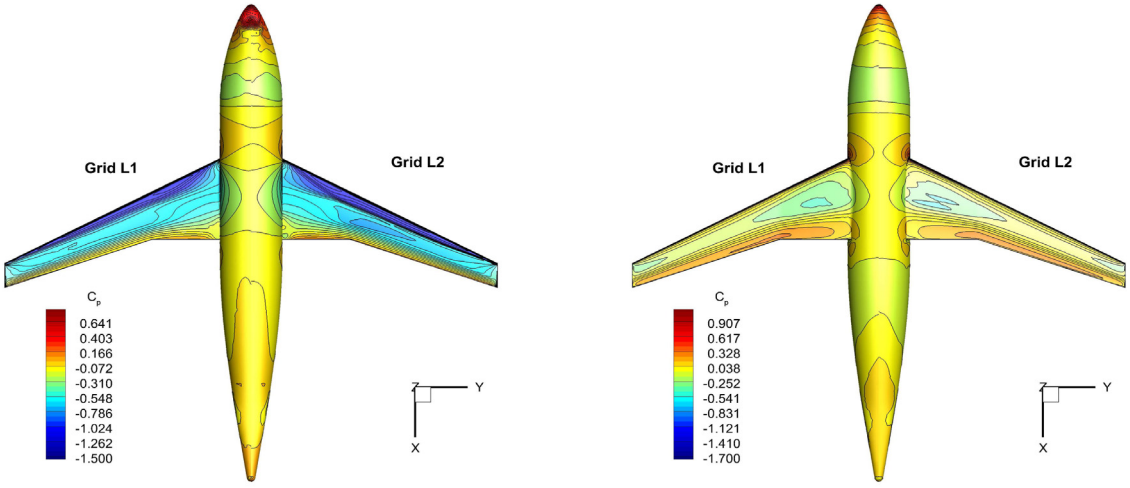


Fig. 18. Comparison of the pressure coefficient distributions of wing sections. The coarse grid $L1$ is applied to low-fidelity CFD model and the fine grid $L2$ is applied to high-fidelity CFD model.

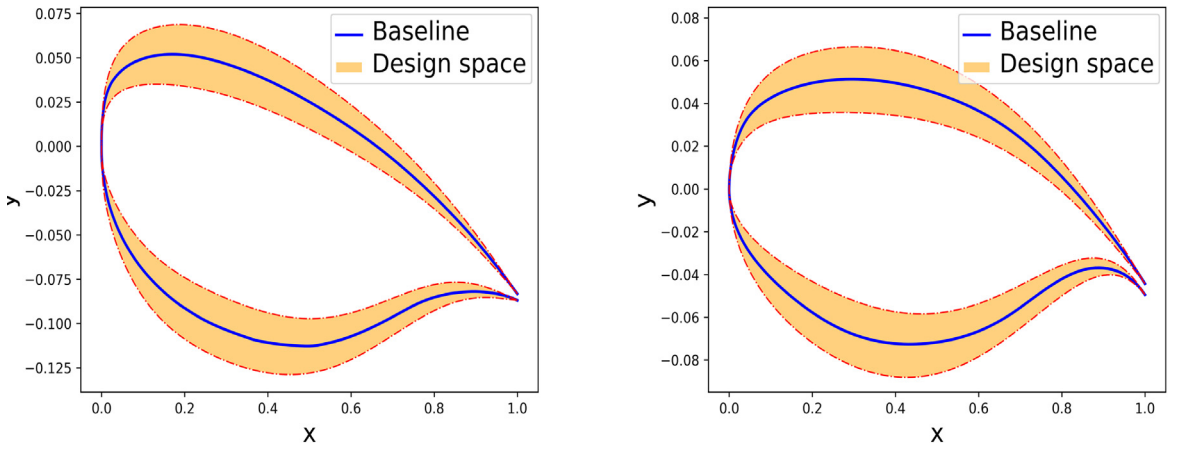
number of iteration is set as 60. In each iteration, one high-fidelity sample and two low-fidelity samples are added to update the multi-fidelity MFDNN surrogate model while only one high-fidelity sample is used to update the single-fidelity DNN model. Fig. 21 shows the optimization objective varying with iterations using MFDNN-based and DNN-based optimization methods. It is found that the MFDNN-based optimization method can produce a better feasible solution, but the single-fidelity optimization method may trap into local optimum due to the strong constraint conditions. It is obvious that our MFDNN surrogate-based optimization method can overcome obstacles of local convergence by including the low-fidelity dataset.

The optimal results of the DLR-F4 wing-body configuration are summarized in Table 9. It can be seen that all the constraints are satisfied in both models. The drag is reduced by 14.66% with the MFDNN-based optimization method, while there is only 3.40% for the single fidelity DNN-based optimization method. As for the lift coefficient, it is increased by 9.26% for single fidelity DNN-based optimization method. While, it is only 0.72% for the MFDNN-based optimization method. However, because of the optimization constraints, the wing area are both enlarged and the maximum thickness of Sections 1 and 2 are slightly increased after the optimization. Moreover, the geometrical shape of wing sections and the corresponding pressure distributions are shown in Figs. 22–25. For



(a) The upper surface (b) The lower surface

Fig. 19. Comparison of the pressure coefficient contours of two grids.

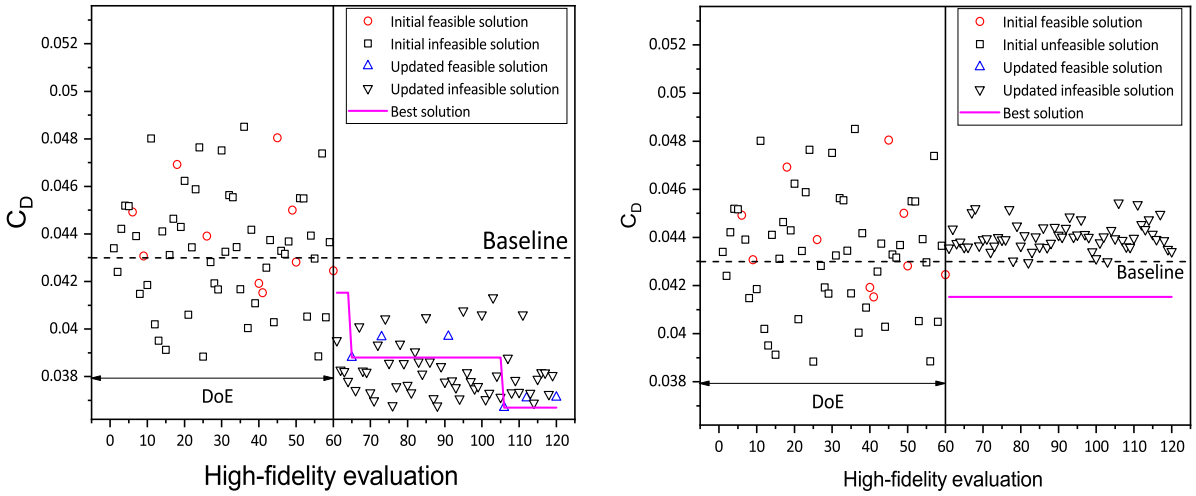


(a) Design space of the 1st section (b) Design space of the 2nd section

Fig. 20. Determination of the decision space for DLR-F4 wing sections.

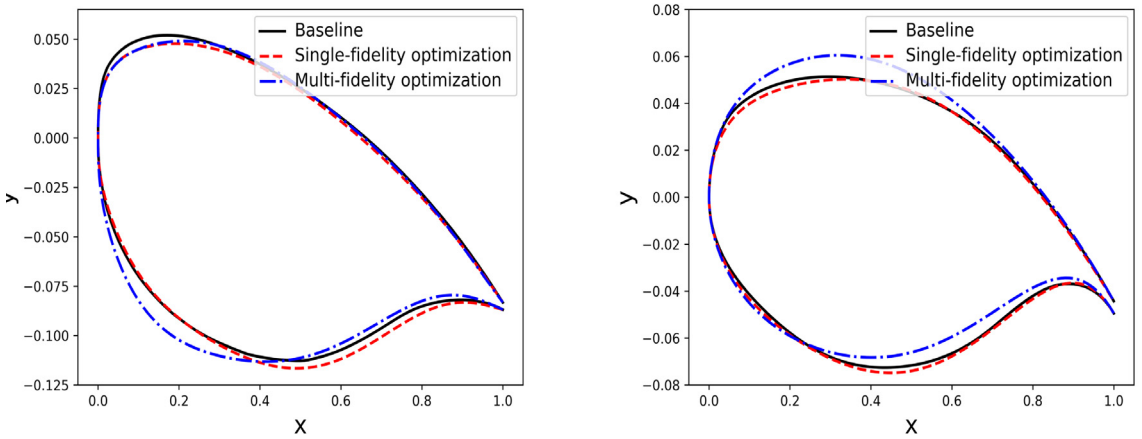
Table 9
Comparison of the aerodynamic force coefficients and the geometry parameters of the baseline and optimized wings.

	Thickness 1	Thickness 2	Area	C_d	C_l
Baseline	0.150446	0.122205	0.072700	0.042994	0.553428
Single-fidelity optimization	0.156878 (+4.28%)	0.127937 (+4.69%)	0.089493 (+23.10%)	0.041531 (−3.40%)	0.604671 (+9.26%)
Multi-fidelity optimization	0.150669 (+0.15%)	0.123719 (+1.24%)	0.100009 (+37.56%)	0.036692 (−14.66%)	0.557386 (+0.72%)



(a) Muti-fidelity optimization (b) Single-fidelity optimization

Fig. 21. Convergence histories of the optimizing DLR-F4 wing-body configuration using the multi-fidelity optimization and single-fidelity optimization. The red circles and black squares indicate the initial feasible and infeasible samples. The blue triangles and black triangles indicate the updated feasible and infeasible samples in each iteration. The purple line represents the best solution in each iteration. The feasible solution means the solution satisfies all the constraints, while the infeasible solution means the solution does not satisfy any constraint.



(a) The first control airfoil (b) The second control airfoil

Fig. 22. Comparison of optimized wing section with the baseline.

both models, the low pressure region at the leading edge is weaken at the spans of $y/b = 0.512, 0.844$, leading to the decrease of drag coefficient. While for the single-fidelity optimization, a shock wave is emerged in the middle section of airfoil, e.g. $0.3 \leq x/c \leq 0.6$, leading to the increase of the lift coefficient.

5. Conclusion

In the present study, an efficient MFDNN-based optimization framework is proposed. By comparing to the Co-kriging method, the deep learning in the current MFDNN model can handle complex mapping problems including linearities, nonlinearities and discontinuities simultaneously. In the optimization framework, to efficiently balance the exploitation and exploration, the high-fidelity infilling strategy by adding the current optimum solution of surrogate

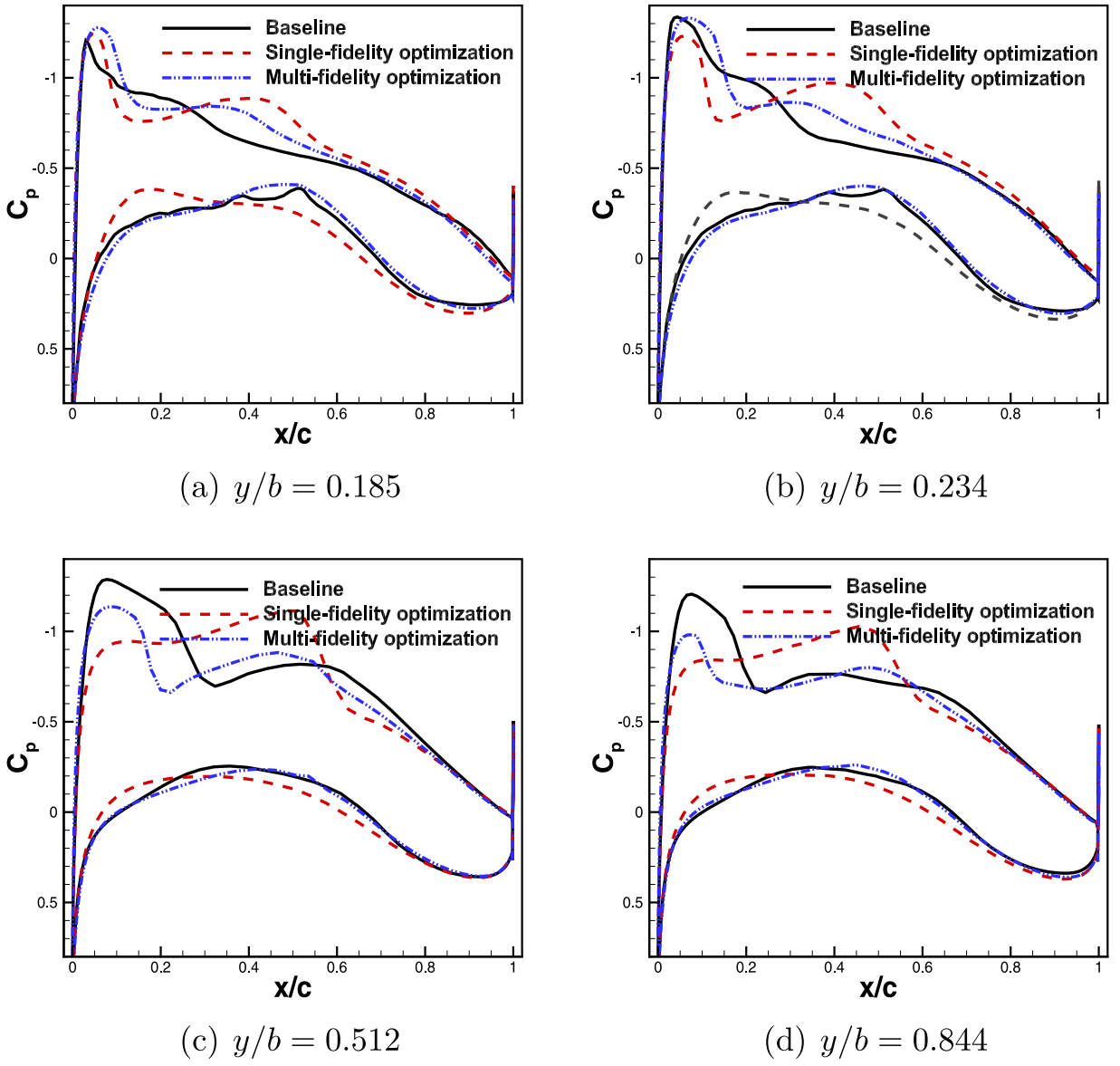
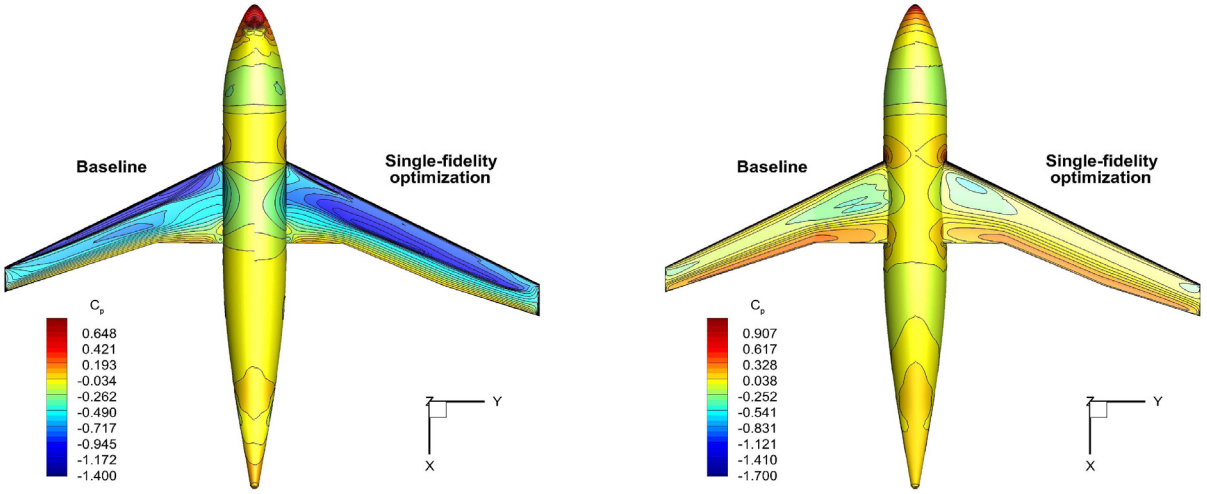


Fig. 23. Comparison of the pressure coefficient distributions of various wing sections.

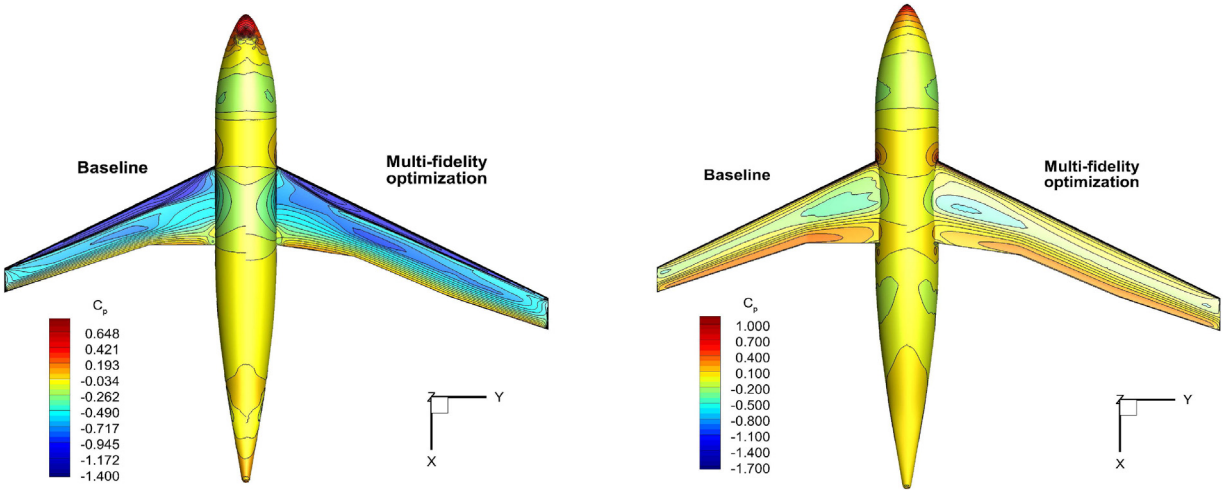
model into the high-fidelity database is employed to find a better solution in the local search. The low-fidelity infilling strategy by adding the solutions distributed uniformly in the whole design space is employed to explore unknown landscape in global search until the convergence criteria is satisfied. Actually, the use of the uncertainty information can more efficiently explore the design space by using acquisition functions. However, the presented MFDNN cannot provide the uncertainty quantification of surrogate model. Therefore, in the future work, we will use Bayesian neural network as surrogate model because it has the properties of safe guard from overfitting and uncertainty quantification.

The proposed MFDNN-based optimization framework is also applied in the aerodynamic shape optimization of a RAE2822 airfoil without any constraint and a DLR-F4 wing-body configuration with multiple strong constraints. The optimization results of the RAE2822 airfoil indicate that the MFDNN-based optimization method performs a faster convergence speed and a higher optimization precision than the single-fidelity DNN-based optimization method. The optimization case of DLR-F4 wing-body configuration is a high-dimensional optimization problem.



(a) The Upper surface (b) The lower surface

Fig. 24. Comparison of the pressure coefficient contour of single-fidelity optimization results and the baseline.



(a) The Upper surface (b) The lower surface

Fig. 25. Comparison of the pressure coefficient contour of multi-fidelity optimization results and the baseline.

The optimization results of the DLR-F4 wing-body configuration verify that the MFDNN-based optimization method can overcome the problem of trapping into local optimum with the assistance of the low-fidelity data, while the single-fidelity DNN surrogate-based cannot, thus making optimization process fail to continue. By comparing the baseline and the corresponding two optimization results, we conclude that the MFDNN-based optimization framework can efficiently explore the unknown design space.

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.

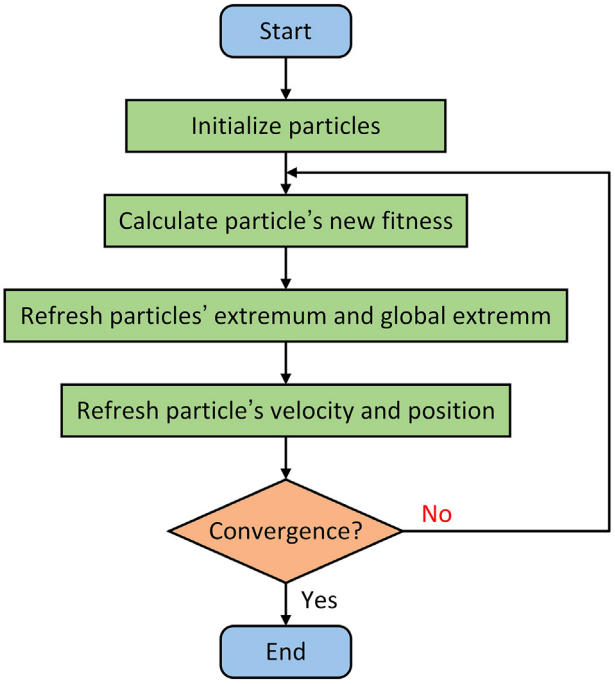


Fig. A.26. Flowchart of PSO algorithm.

Appendix. Particle swarm optimization method

The basic concepts of PSO algorithm are inspired from the behavior study of birds' seeking food [45]. The flowchart of a standard PSO algorithm is shown in Fig. A.26. The PSO algorithm is a random search algorithm based on the group collaboration. Each particle has its own position and velocity. The position of a particle indicates the solution point in the solution domain and the velocity of a particle denotes the flight direction and distance of the particle. Each particle moves towards the direction of the current optimal position g^* and the historical optimal position x_i^* from the previous iterations as the optimization progresses. Meanwhile, it also has a stochastic movement tendency. Therefore, the velocity vectors of particles are updated through the following equation.

$$v_i^{t+1} = v_i^t + \alpha\varepsilon_1 \odot \left|g^* - x_i^*\right| + \beta\varepsilon_2 \odot \left|x_i^* - x_i^t\right|. \tag{A.1}$$

Here, x_i and v_i denote the positions of the tth particles and the velocities of the tth particles, respectively. Both ε_1 and ε_2 are random vectors. Specifically, the Hadamard product $\odot$ represents that each element i, j is the product of elements i, j of the two matrices. The parameters α and β are the learning factors.

Initially, the positions of the particles are generated randomly and ensure decentralized distribution in whole solution domain which is helpful to solve multi-peak optimization problems. Hence the positions of the particles can be updated by following equation in each iteration.

$$x_i^{t+1} = x_i^t + v_i^{t+1}. \tag{A.2}$$

References

[1] Z. Han, Y. Zhang, C. Xu, K. Wang, M. Wu, Z. Zhu, W. Song, Aerodynamic optimization design of large civil aircraft wings using surrogate-based model, Acta Aeronaut. Astronaut. Sin. 40 (1) (2019) 1–16, http://dx.doi.org/10.7527/S1000-6893.2018.22938.

[2] J. Nocedal, S. Wright, Numerical Optimization, Springer Science & Business Media, 2006, http://dx.doi.org/10.1007/978-3-540-35447-5.

[3] A. Jameson, Aerodynamic design via control theory, J. Sci. Comput. 3 (3) (1988) 233–260, http://dx.doi.org/10.1007/978-3-642-83733-3_14.

[4] J. Wang, F. Xie, Y. Zheng, J. Zhang, T. Ji, Virtual stackelberg game coupled with the adjoint method for aerodynamic shape optimization, Eng. Optim. 50 (10) (2018) 1733–1754, http://dx.doi.org/10.2514/2.1441.

[5] J. Wang, Y. Zheng, J. Chen, F. Xie, J. Zhang, J. Périaux, Z. Tang, Single/two-objective aerodynamic shape optimization by a stackelberg/adjoint method, Eng. Optim. 52 (5) (2020) 753–776, http://dx.doi.org/10.1080/0305215X.2019.1618287.
[6] Z. Tang, J. Périaux, J. Dong, Constraints handling in nash/adjoint optimization methods for multi-objective aerodynamic design, Comput. Methods Appl. Mech. Engrg. 271 (2014) 130–143, http://dx.doi.org/10.1016/j.cma.2013.12.006.
[7] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680, http://dx.doi.org/10.1126/science.220.4598.671.
[8] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intell. 1 (1) (2007) 33–57, http://dx.doi.org/10.1109/SIS.2007.368035.
[9] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. B 26 (1) (1996) 29–41, http://dx.doi.org/10.1109/3477.484436.
[10] D.E. Goldberg, J.H. Holland, Genetic algorithms and machine learning, Mach. Learn. 3 (1988) 95–99, http://dx.doi.org/10.1145/168304.168305.
[11] C.C. Da Ronco, R. Ponza, E. Benini, Aerodynamic shape optimization of aircraft components using an advanced multi-objective evolutionary approach, Comput. Methods Appl. Mech. Engrg. 285 (2015) 255–290, http://dx.doi.org/10.1016/j.cma.2014.10.024.
[12] X. Wang, C. Hirsch, S. Kang, C. Lacor, Multi-objective optimization of turbomachinery using improved nsga-ii and approximation model, Comput. Methods Appl. Mech. Engrg. 200 (9–12) (2011) 883–895, http://dx.doi.org/10.1016/j.cma.2010.11.014.
[13] A. Vavalle, N. Qin, Iterative response surface based optimization scheme for transonic airfoil design, J. Aircr. 44 (2) (2007) 365–376, http://dx.doi.org/10.2514/1.19688.
[14] A.I. Forrester, A.J. Keane, Recent advances in surrogate-based optimization, Prog. Aerosp. Sci. 45 (1–3) (2009) 50–79, http://dx.doi.org/10.1016/j.paerosci.2008.11.001.
[15] Y. Yun, M. Yoon, H. Nakayama, Multi-objective optimization based on meta-modeling by using support vector regression, Opt. Eng. 10 (2) (2009) 167–181, http://dx.doi.org/10.1007/s11081-008-9063-1.
[16] S.-S. Shai, S. Ben-David, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, 2014, http://dx.doi.org/10.1017/CBO9781107298019.
[17] D.R. Jones, M. Schonlau, W.J. Welch, Efficient global optimization of expensive black-box functions, J. Global Optim. 13 (4) (1998) 455–492, http://dx.doi.org/10.1023/A:1008306431147.
[18] S. Jeong, M. Murayama, K. Yamamoto, Efficient optimization design method using kriging model, J. Aircr. 42 (2) (2005) 413–420, http://dx.doi.org/10.2514/1.6386.
[19] N.R. Secco, B.S. Mattos, Artificial neural networks applied to airplane design, in: 53rd AIAA Aerospace Sciences Meeting, 2015, http://dx.doi.org/10.2514/6.2015-1013.
[20] L. Chen, H. Qiu, L. Gao, C. Jiang, Z. Yang, Optimization of expensive black-box problems via gradient-enhanced kriging, Comput. Methods Appl. Mech. Engrg. 362 (2020) 112861, http://dx.doi.org/10.1016/j.cma.2020.112861.
[21] I.C. Kampolis, K.C. Giannakoglou, A multilevel approach to single-and multiobjective aerodynamic optimization, Comput. Methods Appl. Mech. Engrg. 197 (33–40) (2008) 2963–2975, http://dx.doi.org/10.1016/j.cma.2008.01.015.
[22] M.C. Kennedy, A. O'Hagan, Predicting the output from a complex computer code when fast approximations are available, Biometrika 87 (1) (2000) 1–13, http://dx.doi.org/10.1093/biomet/87.1.1.
[23] L. Bonfiglio, P. Perdikaris, S. Brizzolara, G. Karniadakis, Multi-fidelity optimization of super-cavitating hydrofoils, Comput. Methods Appl. Mech. Engrg. 332 (2018) 63–85, http://dx.doi.org/10.1016/j.cma.2017.12.009.
[24] H. Zheng, F. Xie, T. Ji, Z. Zhu, Y. Zheng, Multifidelity kinematic parameter optimization of a flapping airfoil, Phys. Rev. E 101 (1) (2020) 013107, http://dx.doi.org/10.1103/PhysRevE.101.013107.
[25] Z. Han, C. Xu, L. Zhang, Y. Zhang, K. Zhang, W. Song, Efficient aerodynamic shape optimization using variable-fidelity surrogate models and multilevel computational grids, Chin. J. Aeronaut. 33 (1) (2019) 31–47, http://dx.doi.org/10.1016/j.cja.2019.05.001.
[26] P. Perdikaris, M. Raissi, A. Damianou, N. Lawrence, G. Karniadakis, Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling, Proc. R. Soc. A 473 (2017) 20160751, http://dx.doi.org/10.1098/rspa.2016.0751.
[27] X. Meng, G.E. Karniadakis, A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems, J. Comput. Phys. 401 (2020) 109020, http://dx.doi.org/10.1016/j.jcp.2019.109020.
[28] E. Brochu, V.M. Cora, N. De Freitas, A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, arXiv preprint arXiv:1012.2599.
[29] R.M. Hicks, P.A. Henne, Wing design by numerical optimization, J. Aircr. 15 (7) (1978) 407–412, http://dx.doi.org/10.2514/3.58379.
[30] P. Della Vecchia, E. Daniele, E. DAmato, An airfoil shape optimization technique coupling parsec parameterization and evolutionary algorithm, Aerosp. Sci. Technol. 32 (1) (2014) 103–110, http://dx.doi.org/10.1016/j.ast.2013.11.006.
[31] S. Painchaud-Ouellet, C. Tribes, J.-Y. Trepanier, D. Pelletier, Airfoil shape optimization using nurbs representation under thickness constraint, in: 42nd AIAA Aerospace Sciences Meeting and Exhibit, 2004, p. 1095, http://dx.doi.org/10.2514/6.2004-1095.
[32] M.H. Straathof, M.J.L. van Tooren, Extension to the class-shape-transformation method based on B-Splines, AIAA J. 49 (2011) 780–790, http://dx.doi.org/10.2514/1.J050706.
[33] H. Prautzsch, W. Boehm, M. Paluszny, Bézier and B-Spline Techniques, Springer Science & Business Media, 2002.
[34] J. Samareh, Aerodynamic shape optimization based on free-form deformation, in: 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2004, p. 4630, http://dx.doi.org/10.2514/6.2004-4630.
[35] D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, 1989, http://dx.doi.org/10.1007/BF01589116.

[36] P. Spalart, S. Allmaras, A one-equation turbulence model for aerodynamic flows, in: 30th Aerospace Sciences Meeting and Exhibit, 1992, p. 439, http://dx.doi.org/10.2514/6.1992-439.

[37] S. Yoon, A. Jameson, Lower-upper symmetric-Gauss–Seidel method for the Euler and Navier–Stokes equations, AIAA J. 26 (9) (1988) 1025–1026, http://dx.doi.org/10.2514/3.10007.

[38] T. Siikonen, An application of roe's flux-difference splitting for k- turbulence model, Internat. J. Numer. Methods Fluids 21 (11) (1995) 1017–1039, http://dx.doi.org/10.1002/fld.1650211102.

[39] B. Van Leer, Towards the ultimate conservative difference scheme. v. a second-order sequel to godunov's method, J. Comput. Phys. 32 (1) (1979) 101–136, http://dx.doi.org/10.1016/0021-9991(79)90145-1.

[40] V. Venkatakrishnan, On the accuracy of limiters and convergence to steady state solutions, in: 31st Aerospace Sciences Meeting, 1993, p. 880, http://dx.doi.org/10.2514/6.1993-880.

[41] A.I. Forrester, A. Sóbester, A.J. Keane, Multi-fidelity optimization via surrogate modelling, Proc. R. Soc. A 463 (2088) (2007) 3251–3269, http://dx.doi.org/10.1098/rspa.2007.1900.

[42] A. Jameson, Origins and further development of the Jameson–Schmidt–Turkel scheme, AIAA J. 55 (5) (2017) 1487–1510, http://dx.doi.org/10.2514/1.J055493.

[43] P. Cook, M. McDonald, M. Firmin, Aerofoil Rae2822-Pressure Distributions, and Boundary Layer and Wake Measurements: AGRD Report AR 138, 1979.

[44] G. Redeker, A Selection of Experimental Test Cases for the Validation of Cfd Codes: AGARD AR 303, 1994.

[45] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, Vol. 4, IEEE, 1995, pp. 1942–1948, http://dx.doi.org/10.1109/ICNN.1995.488968.