Doctor Dissertation

Improved methodologies for efficient aerodynamic shape optimization of a realistic vehicle with “noisy” computations

(「ノイズ」を含んだ数値解析による実車両の効率的空力形状最適化手法の改良)

September 2022

Graduate School of Engineering
Faculty of Engineering
Hiroshima University

QINGYU WANG

Abstract

To achieve the maximum possible improvement in expensive optimization problems with “noisy” computations, such as the drag reduction of a realistic vehicle, within a short period of time, an optimization system is proposed. This system combines the regression Kriging with re-interpolation (RKri) surrogate technique and pseudo expected improvement (PEI) criterion-based efficient global optimization (EGO) algorithm (EGO-PEI), thereby called RKri-EGO-PEI. This is then applied to filter out all kinds of noise produced by computational fluid dynamics (CFD) simulations, maintain a smooth functional trend of the surrogate model, and carry out point infills in a parallel manner in a real-world expensive optimization problem. Additionally, the initial samples with high space-filling quality are also beneficial to further reducing a total number of samples required. To employ a limited number of samples to represent a continuous design space, two modified algorithms, namely, the modified enhanced stochastic evolutionary (MESE) and translational propagation MESE (TPMESE) algorithms, are proposed to construct optimal Latin-hypercube designs (OLHDs) efficiently. The proposed algorithms and each of their original algorithms are applied to compare with other state-of-the-art heuristic algorithms using optimization tests with different-size LHDs. The results show that the TPMESE and MESE converge faster not only than other competitive algorithms but also than each of their original algorithms, respectively. To guarantee two critical optimization processes of tunning Kriging hyperparameters and searching for the optimum on the PEI function, the performance advantages of optimizers on the EGO-PEI algorithm are then investigated. Subsequently, the best one in the benchmark tests is chosen as the optimizer in the proposed optimization system. To confirm the performance of the RKri-EGO-PEI system, two existing optimization systems, namely ordinary Kriging-based EGO-PEI (OK-EGO-PEI) and RKri-based EGO (RKri-EGO), are employed to compete with the proposed system in minimizing the drag coefficient (Cd) of a realistic vehicle using CFD simulations. Furthermore, an object-oriented optimization toolbox, namely, vehicular aerodynamic engineering optimization (VAEO), is programmed to guarantee the smooth implementations of all optimization processes. The results of optimizer tests reveal that an optimizer with higher central goal of exploitation-exploration can not only promote better convergence of the EGO-PEI algorithm within a certain number of point infills, but also achieve the same-level convergence of the EGO-PEI algorithm as that using the other optimizers, with fewer iterations. Regarding the real-world application, the results show that the RKri-EGO-PEI converges to a lower Cd of the vehicle model than those of the OK-EGO-PEI and RKri-EGO systems, with a smaller wall-clock time cost.

Acknowledgement

First, I would like to express my sincere gratitude to my supervisor, Assoc. Prof. Takuji Nakashima, not only for his support on my studies, but also for his optimistic attitude to life which always influences me. His enormous knowledge always helps me to progress studies smoothly. I am pretty sure that I cannot achieve so far without his help, and I significantly desire to be the person as he is in the future. Then, I would like to extend my sincere gratitude to Prof. Hidemi Mutsuda and Asst. Prof. Taiga Karehira for their warm encouragement and valuable suggestions. Finally, I also would like to express my sincere gratitude to Prof. Hidetsugu Iwashita and Prof. Kunihiro Hamada for their valuable suggestions and professional comments to improve the quality of the dissertation.

I also would like to appreciate the financial support from China Scholarship Council (CSC) with the grant number of 201908500118, which guarantees that I completed my studies successfully.

Contents

Chapter 1

Introduction

In recent years, global warming has threatened the survival and development of humankind. To achieve the goals of low carbon and environment protection, stricter emission regulations have been proposed by an increasing number of national governments. Consequently, automotive industries have focused on the development of energy efficient vehicles with low emission and environment pollution. Electric vehicle (EV) is an effective alternative of the conventional combustion-engine vehicle because it is low-pollution and no direct carbon emission, and thus it attracts more and more attention. However, the limited driving range still obstructs the wide use of EV due to low energy density of the battery. To enhance the driving range, the reduction of aerodynamic drag is an effective measure. According to investigations, the primary contribution of total resistance for a vehicle in driving originates from aerodynamic drag at high speed [1]. Additionally, the pressure drag produced by flow separation is the main contributor to the total aerodynamic drag, and over 50% power is consumed against the air resistance [2]. Therefore, reducing the aerodynamic drag can significantly improve the driving range of EV. To reduce the aerodynamic drag, the evaluation of flow field around a vehicle body is necessary. Based on the primary properties of the flow filed, the aerodynamic drag of a vehicle can then be optimized.

1.1 Overview of optimization methodologies in vehicle aerodynamics

In early stage, the investigation of vehicle aerodynamics primarily depended on wind tunnel experiments. However, the efficiency of optimization was significantly low because the improvements of vehicle aerodynamics must be carried out through repeated attempts and experiments [3]. With the rapid development of computational technique, the computational fluid dynamics (CFD) have played an important role, and it has been widely applied to replace expensive wind tunnel experiments for the evaluation of the complex flow filed around a vehicle body [4]. Based on CFD, the optimization methodologies of vehicle aerodynamics have become more diverse. d'Apollonia et al. [5] utilized the design of experiments (DOE) to explore the design space. Then, the aerodynamic objective was evaluated using the CFD method. Howell and Gaylard [6] comprehensively investigated all kinds of factors of a body shape, which influence the aerodynamic drag of a sport utility vehicle (SUV). In the following, the aerodynamic performance of a generic SUV was

investigated after installing an upper and lower flat boat tail plates based on CFD method by Krishnani and Zhou [7]. Kang et al. [8] conducted the study on the drag reduction of a passenger vehicle using an actively translating diffuser. However, it can be seen that considerably repeated simulations and attempts are still necessary if the optimum needs to be found. To solve this problem, the surrogate technique was developed to approximate the relationship between the variables and objective, so that a continuous function can be extracted based on discrete samples. According to the continuous approximation, the optimal solution can be searched easily. Therefore, this method was also called as surrogate-based optimization (SBO). Since SBO can explore the design space with high efficiency and effectively reduce the number of required samples, it has become increasingly popular in expensive optimization problems [9, 10]. Typically, Ando et al [11] conceived an adaptive multistage response surface modeling (RSM) to balance the robustness and accuracy in surrogate metamodeling. Based on the proposed surrogate model, the drag coefficient (Cd) of a realistic sedan was minimized. Khondge et al. [12] combined a free-form deformation (FFD) approach and SBO to investigate the sensitivity of output parameter with respect to design variables, and to optimize the drag force of a real-life vehicle. Song et al. [13] searched for the minimum drag coefficient (Cd) on an artificial neural network (ANN) surrogate model through changing the rear end shapes of a sedan. Additionally, Zhang et al. [14] constructed an automatic route to carry out mesh generation, mesh morphing, and CFD simulations, and the Cd of a MIRA notchback model was then optimized using SBO, under a constraint which the lift coefficient (CL) was no larger than zero. In the same year, Sun et al. [15] optimized the two-dimensional shape of a SUV using SBO in the early stage of development. Not only that, Beigmoradi [16] simultaneously considered the aerodynamic and acoustic objectives for a simplified car model, and thus the multi-objective optimization was carried out employing the ANN and multi-objective genetic algorithm (MOGA). Not long ago, the redial basis function (RBF) surrogate model was applied to extrapolate the relationships between the aerodynamic drag of a car and the heat transfer performance of brake disc. Subsequently, the non-dominated sorting genetic algorithm (NSGA) was utilized to search for the Pareto front for multi-objective optimization [17]. In addition, Urquhart et al. [18] optimized the drag of a square-back vehicle based on SBO.

1.2 Development of methodologies in SBO for expensive problem with “noisy” computations

As reviewed previously, SBO becomes increasingly popular in expensive optimization problems. The main reason is that the surrogate model provides an approximation between the variables and objective, thereby replacing the expensive

experiments (e.g., expensive computational code) to cheaply give a prediction at an unknown point. In comparison, evolutionary algorithms (EAs) need a considerable number of evaluations for expensive experiments to search for the global optimum [19, 20]. The huge computational cost is usually unacceptable in expensive optimization problems, particularly for most of industrial cases pressed over time. A typical SBO usually contains the three steps: sampling, surrogate model metamodeling, and searching for the optimum on the surrogate model [21]. Although the searching process for the global optimum on the surrogate model is significantly cheap, it also leads to a problem in which only the optimum on the surrogate model rather than the real global optimum in the design space can be found. This is because the surrogate model is just an approximation, and it always has difference from the real functional relationship between the variables and objective. Fig. 1.1 shows the different optimum and landscape of the surrogate model from the real function. Additionally, the difference between them is probably large under certain occasions. For example, when the spatial properties of the real relationship between the variables and objective are highly nonlinear, and the number of samples is relatively small, the approximate accuracy of the surrogate model may be significantly low. Unfortunately, sudden perturbations between samples always exist in computer experiments, such as CFD simulations. These sudden perturbations lead to high nonlinearities in some local regions, which significantly changes a smooth functional trend. The sudden perturbations from the expected smooth response are what we treat as noise in the study. In this case, an interpolation-based technique is difficult to fit “noisy” discrete samples to a continuously smooth response. To solve this problem, Forrester et al. [22] proposed the regression Kriging with re-interpolation to automatically filter out noise (i.e., does not pass through from sample points) and maintain a smooth functional trend during the process of surrogate metamodeling. Moreover, the approximate accuracy of the surrogate model should also be enhanced, particularly in the region nearby with the global optimal solution, when the quality of the initial surrogate model cannot meet the requirement. In past decades, all kinds of point-infill strategies based on different surrogate techniques have been developed. Kriging interpolation can provide not only prediction at an unknown point but also its uncertainty. Thus, it is significantly suitable for the development of a point-infill criterion [23]. Based on this, Jones et al. [24] proposed the efficient global optimization (EGO) algorithm with the expected improvement (EI) criterion to add points and refine the Kriging model. Compared with other criteria, such as the minimization surrogate prediction (MSP) [25, 26] and probability of improvement (PI) [27], the EI criterion balances the local exploitation and global exploration automatically. Thus, it shows robust and efficient performance during the process of point infills. However, the traditional EGO with EI criterion has

an apparent shortcoming, which is only one point can be added in each cycle because the Kriging model must be updated using the last updating point. Under this situation, the outer unoccupied computational resources cannot be made full use of. Therefore, to match the parallel architecture through making full use of outer computational resources, and significantly reduce the wall-clock time during the entire process of expensive optimization, a multi-point infill strategy is necessary. Ginsbourger et al. [28] proposed the q-point EI (q-EI) criterion. Although the q-EI allows to search for q updating points with a single optimization process, its computation is still time-consuming. To reduce the computational cost, a substitute, namely, constant liar (CL), was further developed by Ginsbourger et al. [28]. The CL strategy uses the current minimum value as a "liar" to replace the evaluation of updating point. Thus, the Kriging model can be updated based on the fake information. Due to only the cheap EI function needs to be evaluated, the computational cost is significantly reduced. Nevertheless, the shortcoming of the CL strategy is also evident. This is because the "liar" without the real evaluation at the updating point easily guides the searching process towards a wrong direction. Consequently, more efforts must be carried out to obtain a real improvement. By observing the behavior during the updates of the EI function, Zhan et al. [29] proposed the pseudo EI (PEI) criterion. The PEI criterion simulates the primary properties in updates of EI function by multiplying the EI function by the influence function (IF). According to this, the later updating point can be updated without the evaluation of early updating points. Because the PEI criterion substantially provides an approximation with the EI function, it shows robust performance during the process of point infills.

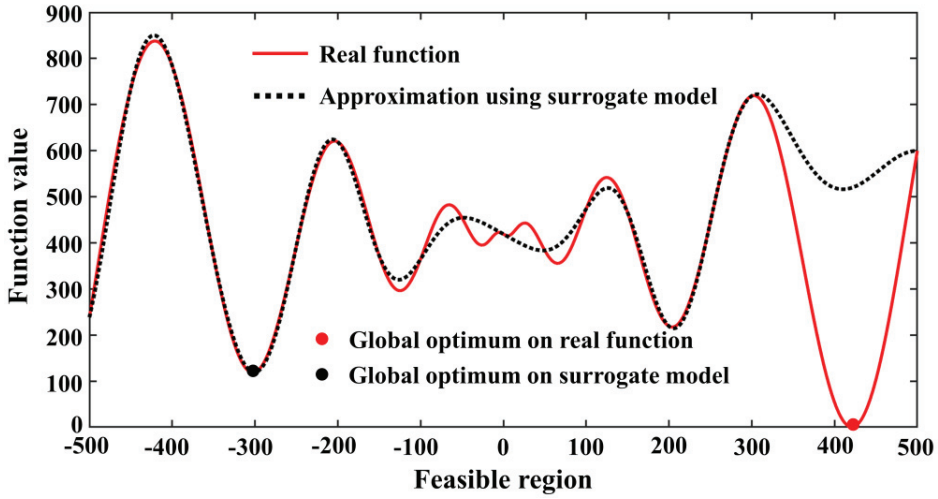


Fig. 1.1 Difference between the real function and approximation using surrogate model

1.3 Objective of research

In fact, the aerodynamic drag reduction of a realistic vehicle is a considerably expensive optimization problem. Although CFD simulations have been widely applied to replace more expensive wind tunnel experiments, the implementations of CFD simulations for a realistic vehicle are still time-consuming. The reasons are simple. On the one hand, the property of a flow field around a realistic vehicle body is usually complicated; on the other hand, relatively fine meshes with a high resolution must be given so that the accuracy of the numerical solution can be acceptable, even for industrial problems. These factors make the higher demands of computational method and burden. However, the optimization cases of vehicle aerodynamics are always pressed over time because of an increasingly intense competition among automotive companies. Apparently, due to the practical limitations of the required time and expensive computational burden, it is usually impossible to search for the real optimal solution without a considerable number of experiments. Under this situation, the primary focus of optimization should be changed to obtain the maximum possible improvement with a reasonable computational cost or a limited time. References [30, 31] are convincing evidence to support this objective. Moreover, the rationality of the objective for this thesis can be further verified by the following points:

- Applicability of an optimization method in expensive industrial optimization cases: Blacha et al. [32] (Audi automotive company) clearly pointed out that the applicability of each optimization method not only depends on its accuracy, but also on its time cost and robustness during the development of vehicle aerodynamics. The point further verifies the importance of optimization method for the efficiency and robustness of expensive engineering optimization. Generally, in the case of expensive optimization, the priority should be changed to achieve high-quality convergence (e.g., a high-quality local solution) within a limited time or reasonable computational cost (e.g., given a reasonable number of point infills) [30, 31, 33].
- Priority in expensive optimization cases: Even though optimization algorithms (e.g., evolutionary computations) and point-infill criteria have been widely studied, and their global convergence performance has also been significantly certified, they are still difficult to be really applied in engineering cases with expensive experiments [14, 17, 34]. The main problem is still an unacceptable computational cost (e.g., a long period of optimization). In fact, if the real global convergence is always the priority, the evolutionary algorithms (EAs), such as the genetic algorithm (GA), exactly

have higher performance than that of SBO or different gradient-based algorithms. However, in the real-world problems, SBO, which pursues high efficiency, has been widely used rather than EAs significantly demonstrating the importance of the balance between convergence quality and computational cost [31]. Similarly, a gradient-based algorithm usually has a higher priority than EA in an expensive optimization mission. The cogent evidence is the application of the stochastic gradient decent (SGD) algorithm [35] in deep neural network (DNN). Apparently, a reasonable computational cost is significantly important in expensive optimization.

Now, the primary objective of the thesis is clear, and its rationality has been adequately verified. In this thesis, we focus more on obtaining a real improvement of aerodynamic objectives within a short period of time. Herein, the improvement indicates that a better objective value (a lower drag coefficient or benchmark fitness in our study) in comparison with the current best objective value is achieved in the process of point infill. To achieve the objective, some improved methodologies in SBO are proposed. Generally, these methodologies aim at achieving following goals:

- Initial evaluation locations with high space-filling quality must be provided, so that a surrogate model with relatively high quality can be constructed using as fewer samples as possible. The step is quite important. Two points can explain its importance. First, well-distributed samples can better extract spatial information from a limited number of discrete samples, which can effectively reduce the computational costs in evaluating the objectives of initial evaluation locations. Second, the property of the initial surrogate model significantly influences the subsequent process of point infills, thereby also affecting the efficiency of convergence with respect to the optimization objective.
- All kinds of noise from the expected smooth response produced by CFD should be robustly filtered out, so that a smooth functional trend, particularly in the local region around the current best solution, can be maintained. Based on it, a surrogate model with a distinct trend for a potential improvement can be constructed, and thus the subsequential point-infill process can employ such information to fast obtain a real improvement.
- To further reduce the conducting time of optimization, all time-consuming steps, such as mesh preparation, waiting gap before simulation, and numerical simulation, can be carried out in a parallel manner. Therefore, the point-infill strategy needs to predict multiple updating points in each cycle of point infills.

- An automatic route must be constructed so that the critical steps, including the construction of surrogate model, predictions of updating points, mesh morphing, mesh generation, and CFD simulation can be carried out automatically. This is necessary to further reduce the labor costs in expensive optimization problems.

1.4 Outline of thesis

Fig. 1.2 provides a graphical overview of this thesis. Subsequently, the primary content of each chapter is introduced in detail.

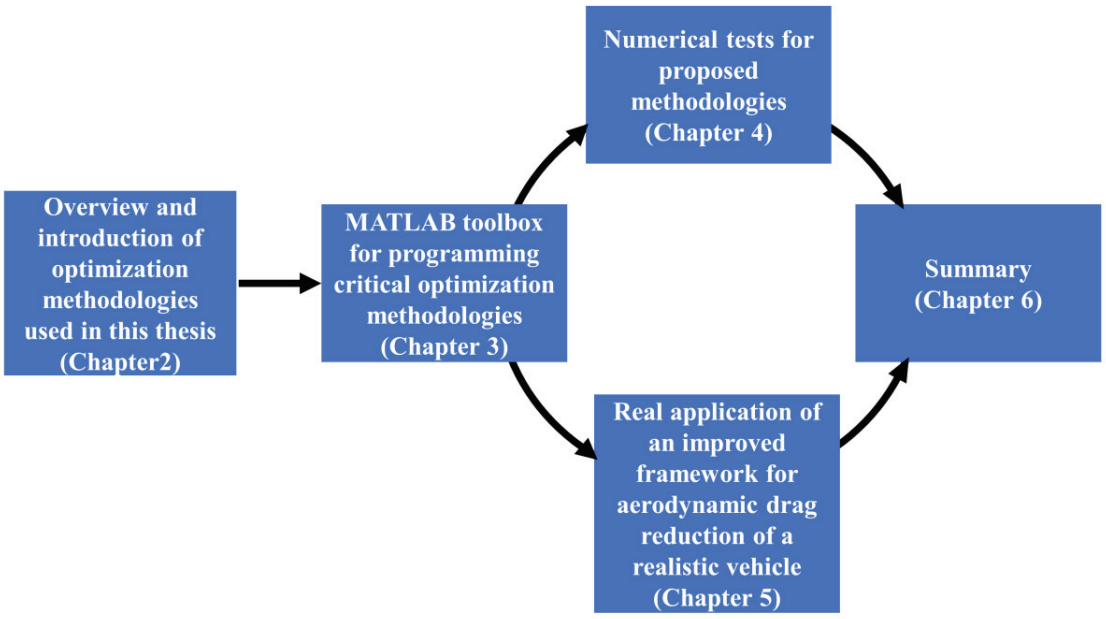


Fig. 1.2 Graphical overview of the thesis

Chapter 2: After a brief review with respect to the state-of-the-art algorithms for constructing optimal Latin-hypercube design (OLHD), two enhanced algorithms are proposed to efficiently construct OLHDs with high space-filling quality in our research. Subsequently, the basic theory of ordinary Kriging (OK) and RKri models are introduced. Because the RKri model can filter out noise produced by CFD simulations and maintain a relatively smooth functional trend, it is suitable to be adopted as a surrogate model in the "noisy" optimization problem. To progress the optimization process towards the global optimization, some widely used point-infill criteria are then reviewed. In these criteria, the PEI criterion

which can robustly and efficiently carry out point infills in a parallel manner is adopted as the point-infill criterion in the research. Moreover, the performance of optimizer significantly influences not only the Kriging hyperparameter tunning but also the search for the optimum on the PEI function. Thus, the influence of an optimizer on the PEI-based EGO (EGO-PEI) algorithm is briefly discussed. Finally, an improved optimization framework, integrated the efficient sampling method, RKri surrogate model, EGO-PEI algorithm, and a suitable optimizer, is proposed.

Chapter 3: A MATLAB toolbox based on object-oriented programming (OOP) are developed so that the construction of a surrogate model, predictions of updating points, error analysis, statistical analysis, sensitivity analysis, and visualizations in the research can be achieved. After a brief introduction of its functionalities, the performance of the toolbox is roughly demonstrated though the comparisons with other toolboxes in Kriging metamodeling and solving a global optimization problem.

Chapter 4: In this chapter, the performance of proposed algorithms for the construction of OLHD are first investigated by comparison with each of their original algorithms and other state-of-the-art algorithms. In the following, the rationality of the combination between the RKri model and EGO-PEI algorithm is investigated. To select a suitable optimizer for the proposed optimization framework, several optimizers with all kinds of performance advantages are adopted as optimizers in OK-based EGO-PEI algorithm first, and the OK-based EGO-PEI algorithm with different optimizers is then tested on benchmark functions. The optimizer performed best in the numerical testes is further tested using the RKri-based EGO-PEI algorithm. Then, the mechanism with respect to the influence of optimizer on promoting a better convergence of the EGO-PEI algorithm is analyzed.

Chapter 5: The proposed optimization system is applied to solve the real-world expensive optimization problem. In this case, the Cd of a realistic vehicle must be minimized as much as possible within a limited number of point infills (an acceptable computational burden). To carry out the real-world optimization, the information of vehicle model, design variables, mesh morphing, mesh strategies, and settings of CFD solver are introduced. After the validation with respect to the accuracy of CFD simulations based on mesh independence and wind tunnel experiment, the proposed optimization system is compared with the other two existing systems in minimizing the Cd of a realistic vehicle. Then, its performance is analyzed based on the results of real-world optimization case.

Chapter 2

Critical methodologies in expensive global optimization

For expensive global optimization problems, SBO can effectively reduce the number of expensive experiments, and it has been widely used. A completive process to carry out SBO usually includes sampling, surrogate metamodeling, and point infills. Moreover, the most time-consuming step in expensive optimization problems is always the evaluation of objective. Therefore, if the number of samples can be reduced, the efficiency of optimization can be improved. In this chapter, the primary goals aim not only to provide brief overview for some widely used techniques in global optimization using SBO method, such as OLHD, Kriging metamodeling, and methodologies of point infills based on Kriging model, but also to introduce some proposed methodologies for reducing the number of samples and progressing the optimization towards the global convergence with high efficiency in the real-word optimization problem with "noisy" computer experiments. To achieve the goals, two new algorithms are developed to efficiently construct OLHD with high space-filling quality in Subsection 2.1. Subsequently, the basic theory of the OK and RKri are introduced in Subsection 2.2. In Subsection 2.3, the point-infill methodologies in serial and parallel manners are reviewed. Due to the optimizer must be applied to tune the Kriging hyperparameters and search for the optimum on point-infill criteria, the influence of optimizer on the parallel point-infill methodology is analyzed in Subsection 2.4. Based on the aforementioned methodologies, an improved optimization system is proposed to efficiently solve the real-world optimization problem with "noisy" computations in Subsection 2.6.

2.1 Optimal Latin-hypercube design for sampling

In most engineering optimization problems, experiments are significantly expensive. Therefore, it is difficult to conduct a lot of experiments so that the properties of design space can be reflected. To reduce experimental costs, an effective method to determine locations where experiments should be conducted is necessary. The techniques of design of experiments (DOE) have been widely employed to achieve this in recent decades. In those techniques, the Latin-hypercube design (LHD) proposed by McKay et al. [36] and Iman and Conover [37] can effectively fill design space. However, due to the random permutation of evaluation points in LHD, a uniform distribution of evaluation points is

difficult to be guaranteed. As a result, a high space-filling quality using finite evaluation points cannot be achieved. To further improve the space-filling quality of a LHD, the technique of OLHD has been widely investigated. Classically, the columnwise-pairwise (CP) algorithm [38] was proposed to optimize the space-filling quality of LHD by exchanging elements in per column of LHD. Fang et al. [39] developed a heuristic algorithm, namely, threshold accepting (TA), in constructing OLHD under the L_p- discrepancy criterion. Bates et al. [40] proposed permutation genetic algorithm (PermGA) with an integer encoded method to search an OLHD. Grosso et al. [41] developed iterated local search (ILS) algorithm for the optimization of LHD. To further increase the capability of global convergence, the enhanced stochastic evolutionary (ESE) algorithm [42] was proposed to better balance the global exploration and local exploitation using a threshold "temperature" in constructing an OLHD. The results of numerical tests showed that the ESE has significant capability to enhance the space-filling quality of LHD. However, it should be notable that the optimization of LHD has huge computational costs, particularly when a design has high dimensions and considerable evaluation points. To balance the computational costs and space-filling quality of design in optimization, many efforts were made to enhance the efficiency of OLHD. Those methods include enhancement of ESE (EESE) algorithm [43], successive local enumeration (SLE) algorithm [44], particle swarm optimization (PSO) algorithm [45], sequencing optimization based on simulated annealing (SOBSA) algorithm [46], a new framework of DOE in accordance with PermGA [47], PermGA with chromosome-length-expansion (CLE) [48], sliced latin-hypercube design (SLHD) [49], maximum projection design [50, 51], sequential-successive local enumeration (S-SLE) algorithm [52], inflate, expand and stack (IES) algorithm [53], an efficient method for constructing space-filling and near-orthogonality Sequential LHD [54], a novel extension algorithm [55], maximin distance Latin squares and related LHD based on Costas arrays and the Welch, Gilbert and Golomb methods [56], and local search-based genetic algorithm(LSGA) [57]. Particularly, a method, namely, the LHD via translational propagation (TPLHD) algorithm, was proposed by Viana et al. [58] to efficiently construct the LHD with near high space-filling quality, which draws our attention. Due to the TPLHD only implements the translational operation without any mathematic evaluation, its computational burden is significantly low. This property makes TPLHD highly efficient and suitable to be as an initial design in an optimization process. Moreover, some important conclusions are worth to be reviewed. In literature [59], the performance of simulated annealing (SA) and ESE algorithms are compared. Husslage et al. [60] also comprehensively compared the performance of SA, ESE, and PermGA algorithms. The results published by those references both showed the ESE has higher performance than comparative algorithms to search an OLHD with high

space-filling quality. Additionally, the ESE was employed to compete with more algorithms including the SOBSA, SLE, GA, SLHD, and LSGA in literatures [46, 54, 55, 57]. The performance of ESE was further verified. Generally, the ESE showed robust and efficient performance in constructing an OLHD with high space-filling quality.

To efficiently generate an OLHD with high space-filling quality, two modified algorithms, namely, modified ESE (MESE) and translational propagation MESE (TPMESE) are proposed, respectively. In the MESE algorithm, a new updating strategy is introduced to adaptively adjust the value of threshold "temperature". The adaptive strategy of threshold "temperature" aims to simultaneously enhance the efficiency of optimization and maintain the sufficient global search capability of the ESE. To further accelerate the convergence of optimization, the MESE is combined with TPLHD algorithm (TPMESE). For the TPMESE, the MESE algorithm optimizes a design with a near high quality generated using the TPLHD rather than beginning the optimization from a random LHD. Evidently, the convergence speed of optimization using the TPMESE can be significantly increased, particularly in the cases with large sizes and many evaluation points.

2.1.1 Optimization criterion

To optimize LHD, the space-filling quality should be evaluated. In those criteria, the φ_p criterion [61] is a popular technique to evaluate the space-filling quality in the optimization of LHD. The φ_p criterion can be calculated as follows:

$$\varphi_p = \left[\sum_{i=1}^{n_p-1} \sum_{j=i+1}^{n_p} d_{ij}^{-p} \right]^{1/p}, \tag{2.1}$$

where d_{ij} and n_p are the distance between two random evaluation points and the number of evaluation points, respectively. The distance of two evaluation points is calculated as:

$$d_{ij} = \left[\sum_{k=1}^{n_v} |x_{ik} - x_{jk}| \right]^{1/t}, \tag{2.2}$$

where x_{ik} and x_{jk} are the k^{th} element of i^{th} and j^{th} points, respectively. For the constants p and t, p = 50 and t = 1 are suggested in reference [61] to evaluate the space-filling quality of design. To achieve a uniform distribution of evaluation

points with high space-filling quality in design space, the maximization of distance between two random points meets above requirement. Apparently, the minimization of φ_p criterion is equal to maximize the distance between evaluation points. Therefore, high space-filling quality of design can be reached by the optimization of φ_p criterion. In our study, the φ_p criterion was finally chosen to be the objective function in the optimization of LHD.

2.1.2 Translational propagation algorithm

TPLHD is an efficient algorithm for the construction of an OLHD. In this algorithm, the translational operation was employed to fill design space without any evaluations of optimization criteria. Here, the process for the construction of a 9 × 2 (9 points and 2 dimensions) TPLHD started from a seed of 1 × 2 is used as an example, so that the construction of TPLHD can be clear. At the beginning, the design space is divided into n_b blocks. The n_b can be calculated by:

$$n_b = n_p/n_s \tag{2.3}$$

Here, n_p is the number of evaluation points, whereas n_s is the number of points in a seed. Fig. 2.1 shows the examples of seed designs. Then, the seed initially placed in the original coordinates translates in the first dimension (horizontal direction) until all divisions in the first dimension are filled by seeds. Additionally, to fill all divisions, the seed should translate per n_d levels each time, where the n_d levels (the number of divisions) are derived as:

$$n_d = {n_b}^{1/n_v}, \tag{2.4}$$

where the n_v indicates the dimension of TPLHD. Subsequently, all seeds in the first dimension are treated as a new seed, and it continues to translate along with the second dimension (vertical direction) until all divisions in the second dimension are filled by new seeds. Fig. 2.2 shows the aforementioned translational operation. From the steps 1 to 3, the seed translates in the first dimension. Then, a new seed is composed of the three points shown in step 3, which continue to translate in the second dimension until the situation shown in step 4 occurs.

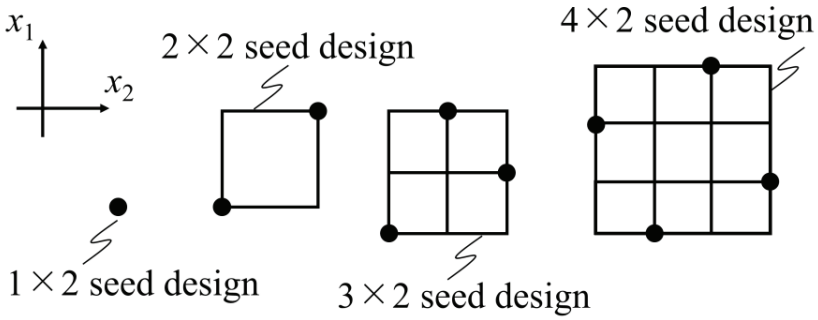


Fig 2.1 Examples of seed designs with two dimensions

Moreover, because of the main property of LHD, one point per level, the aforementioned translational operation also needs to be simultaneously conducted in another dimension. More specifically, the seed located at the original coordinates should simultaneously translate n_d levels and a level in the first and second dimensions, respectively, like the step 2 shows in Fig. 2.2. Meanwhile, a resizing process is prepared to construct a TPLHD with any size, when the number of evaluation points in the initial TPLHD constructed is not equal to the required number. Fig. 2.3 shows the details of resizing process. It is clear that a TPLHD with larger size 9 × 2 is resized by removing an additional point and its corresponding levels, so that a TPLHD with required size of 8 × 2 can be achieved. To maintain space-filling quality as much as possible in the resizing process, the distances between the center of TPLHD and each point are previously evaluated. Then, the point is removed one by one based on the sequence of descending permutation with respect to the distances. Therefore, an additional point which has the largest distance with the center of TPLHD is removed shown in Fig. 2.3.

It should be noted that the TPLHD focus more on efficiently constructing a LHD with near high quality instead of the best quality in comparison with other optimization algorithms of LHD. Due to there is no mathematic evaluation, the computational cost to build a TPLHD becomes significantly low. Thus, it is significantly suitable to be a starting LHD in an expensive optimization process for promoting its convergence. Additionally, the literature [58] shows that different points in a seed have different space-filling properties. Considering the significantly cheap computational cost of a TPLHD, it is possible to choose the best one from several TPLHDs constructed in basis of different seed points. To further accelerate the convergence of global optimization process for modified algorithm, MESE, a TPLHD with the highest space-filling quality is chosen from five TPLHDs constructed by the seed points from one to five. The criterion to evaluate spacing-filling quality is the φ_p criterion. To more clear show the process for the construction of a TPLHD in the study,

Table 2.1 lists the pseudo code.

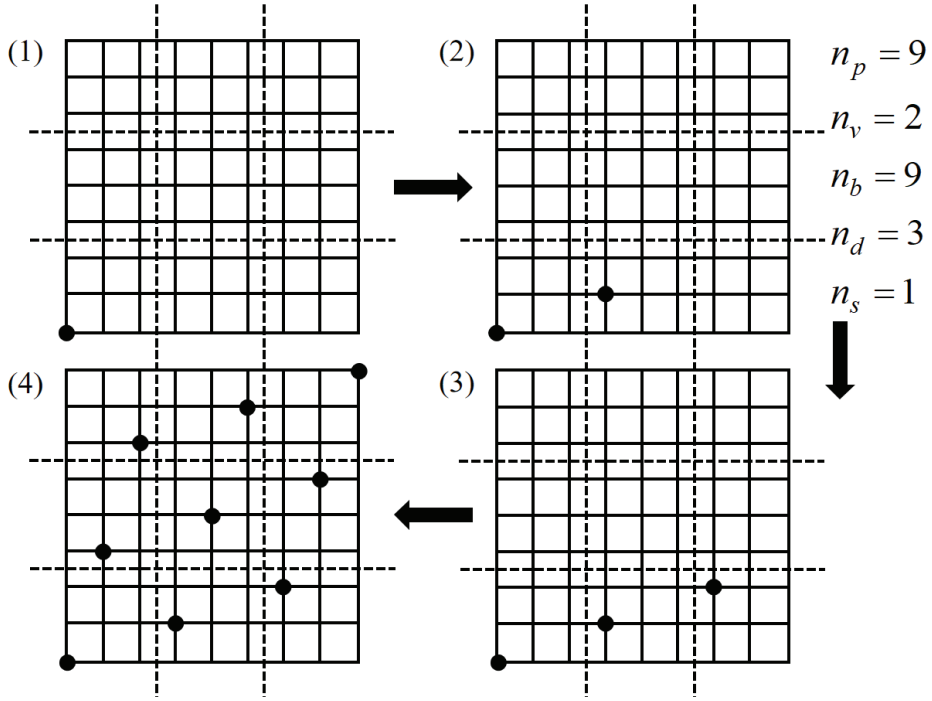


Fig. 2.2 Translational operation of TPLHD

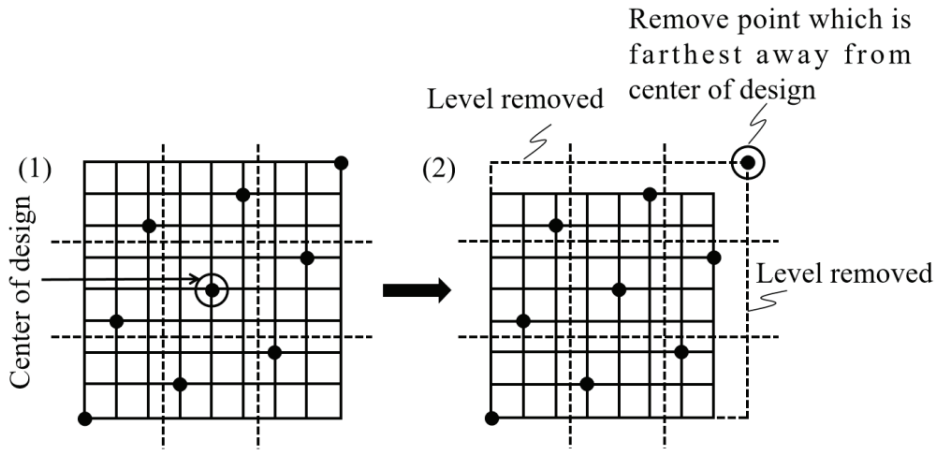


Fig. 2.3 Resizing process of TPLHD

Table 2.1 Pseudo code of TPLHD

Line	Procedure of construction for TPLHD
1	Set n_p, n_v.
2	**For** n_s 1 to 5
3	**If** n_d is an integer
4	$n_{b=}n_p/n_s$.
5	**Else**
6	$n_{b=}\text{ceil}(n_d)^n$.
7	Recalculate points number n_p^* based on the new n_b.
8	**End If**
9	Generate initial TPLHD.
10	**If** $n_p^* > n_p$
11	Resizing process to remove points from initial TPLHD.
12	**End If**
13	**End For**
14	Pick the best TPLHD based on the min value of φ_p.

2.1.3 Enhanced stochastic environment algorithm

The ESE algorithm is modified in terms of the stochastic evolutionary (SE) algorithm [62]. The algorithm mainly contains outer and inner loops to automatically update the "temperature" and implement element-exchange operation in one column, respectively. In the inner loop, *J* distinct elements are initially exchanged in a column based on the design X. Accordingly, a new design is obtained after randomly exchanging two elements each time. Then, the best design X_{try} is picked from *J* candidate designs based on the φ_p criterion. In the following, if the acceptance criterion is met, the current design X will be replaced by the X_{try}. The current design X and the current best design X_{best} are also compared with each other. If X is better than X_{best}, X_{best} will be updated using X. The aforementioned process is iterated *M* times. The acceptance criterion can be written as:

$$f(\mathrm{X}_{try}) - f(\mathrm{X}) \leq T_h \cdot \mathrm{rand}(0,1), \tag{2.5}$$

where $f(\mathrm{X}_{try})$ and $f(\mathrm{X})$ are the values of φ_p criterion with respect to designs X and X_{try}, respectively. T_h is a constant

value of threshold "temperature", whereas the function rand(·) generates a random value from zero to one. In the outer loop, the temperature is updated based on the exploration and improvement processes, respectively. To distinguish different processes, a judgement is calculated. Then, the value of "temperature" is updated using different expressions in terms of the acceptance n_{acpt} / M and improvement n_{imp} / M ratios which are calculated in the inner loop. The expression of judgement is derived as:

$$f(\mathrm{X}_{oldbest}) - f(\mathrm{X}_{best}) > \mathrm{tol}, \tag{2.6}$$

where the $\mathrm{X}_{oldbest}$ indicates the best design before an improvement obtained compared with the current best design X_{best}.

It is easy to find that the most important process to determine the performance of ESE is the update of "temperature". To update the "temperature", the ESE algorithm uses the different updating methods based on the selection of exploration or improvement processes. In the exploration process, T_h will be decreased using the expression $T_h = \alpha_2 T_{h\_old}$, if the acceptance ratio n_{acpt} / M is larger than a large threshold such as 0.8. In contrast, T_h is increased based on the expression $T_h = T_{h\_old}/\alpha_3$, when the improvement ratio n_{imp} / M is lower than a small threshold like 0.1. The exploration process aims to escape from a local solution, so that a new region can be explored. In comparison with the exploration process, the improvement process focuses on the fast convergence to a local optimal solution. To achieve that, the T_h is reduced in terms of $T_h = \alpha_1 T_{h\_old}$, when n_{acpt} / M is larger than a small threshold such as 0.1 and n_{imp} / M. While n_{acpt} / M is larger than a small value like 0.1 but equal to n_{imp} / M, T_h is maintained. In contrary, T_h will be increased based on the equation $T_h = T_{h\_old}/\alpha_1$. For other parameters used in the ESE, the literature [42] suggests that M and J are $M = 2n_e m/J$ and $J = n_e/5$, respectively, where n_e is the number of all element exchanges which can be conducted in one column. Moreover, the J are M should be larger than 50 and 100, respectively. To make the process of ESE clear, Table 2.2 and Fig. 2.4 show the pseudo code and flowchart of ESE algorithm, respectively. The figure was redrawn based on the reference [42].

Table 2.2 Pseudo code of ESE algorithm

Line	ESE procedure for inner loop
1	**For** i = 1 to M
2	Pick the best X_{try} from J candidate designs within a column (i mod m).
3	**If** $f(\mathrm{X}_{try}) - f(\mathrm{X}) \leq T_h \cdot \mathrm{rand}(0,1)$
4	$\mathrm{X} = \mathrm{X}_{try}$
5	$n_{acpt} = n_{acpt} + 1$
6	**If** $f(\mathrm{X}) - f(\mathrm{X}_{best})$
7	$\mathrm{X}_{try} = \mathrm{X}$
8	$n_{imp} = n_{imp} + 1$
9	**End If**
10	**End If**
11	**End For**
Line	ESE procedure for outer loop
1	Initialize $\mathrm{X} = \mathrm{X}_0$, $\mathrm{X}_{try} = \mathrm{X}$, $\mathrm{X}_{best} = \mathrm{X}$, $\mathrm{X}_{oldbest} = \mathrm{X}_{best}$ based on initial X_0 design and "temperature" T_0.
2	**While** stopping criterion is not satisfied
3	$\mathrm{X}_{oldbest} = \mathrm{X}_{best}, n_{acpt} = 0, n_{imp} = 0$
4	Implement inner loop.
5	**If** $f(\mathrm{X}_{oldbest}) - f(\mathrm{X}_{best}) > \mathrm{tol}$
6	Implement improvement process to update T_h.
7	**Else**
8	Implement exploration process to update T_h.
9	**End If**
10	**End While**

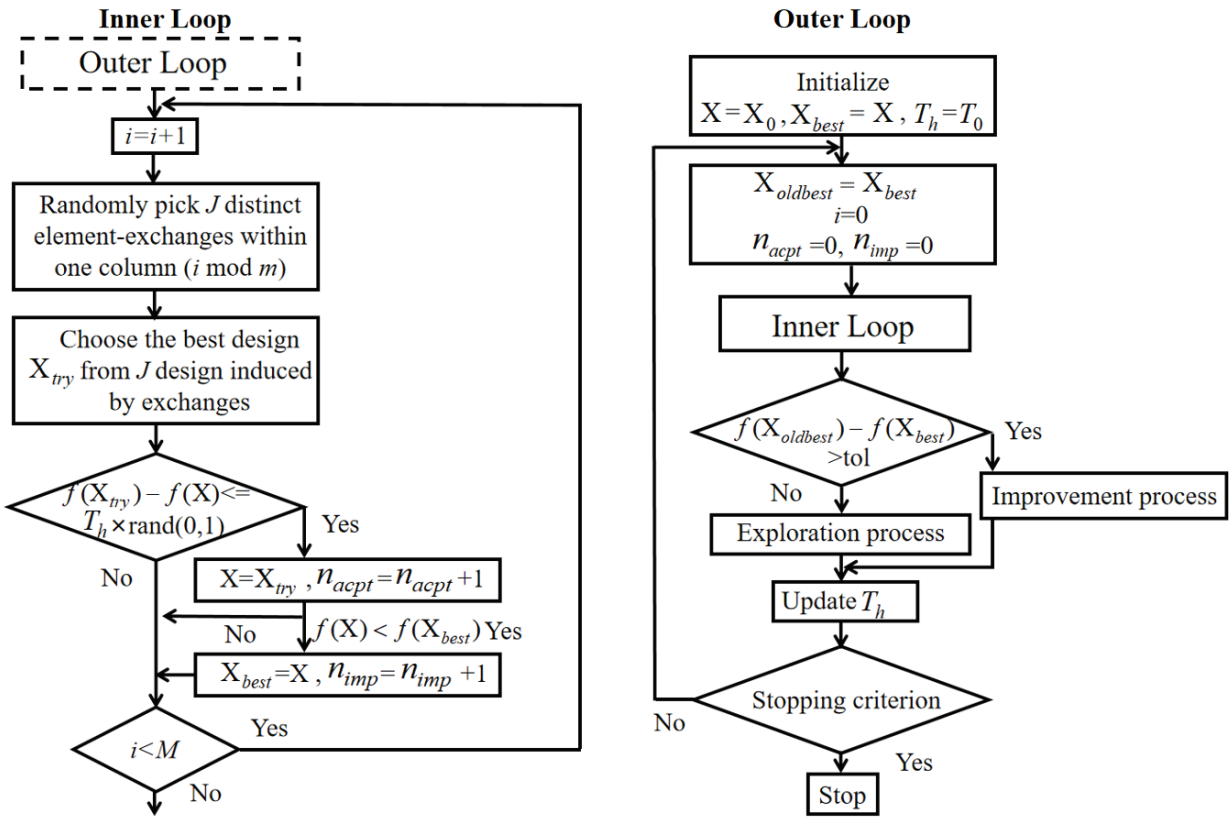


Fig. 2.4 Flowchart of ESE algorithm

2.1.4 Modified algorithms for construction of OLHD

In the ESE algorithm, some intermediate solutions in M iterations can be accepted, even though there is no improvement of X_{try} compared with the current design X. This strategy can appropriately increase diversity and avoid the convergence to a local optimum. To give a threshold for choosing those intermediate solutions, T_h is proposed and adaptively updated in the outer loop of ESE algorithm. However, only the constant scale factors, α_1, α_2, and α_3, are employed to control the update of T_h, which easily leads to an oversize or undersize step in the update of T_h. As a result, some potential intermediate solutions will be lost, which can generate a better design after several iterations. Therefore, the convergence will be slowed down as the algorithm must search those intermediate solutions again in the following implementation. Additionally, the improvement process is usually conducted in the early stage of optimization because

of adequate diversity of individuals. Apparently, T_h is always fast reduced in the process, so that the ESE can quickly converge to a local solution. Then, T_h will vibrate as its update strategy is switched to use the exploration process in the rest of period. To further accelerate the convergence after the summary of main properties with respect to the update of T_h in the ESE algorithm, the scale factors of variable step length in proposed algorithm MESE are applied to replace the previous constant scale factors in the ESE algorithm. In the MESE, there are still two loops, inner and outer loops. The inner loop is the same as the ESE, whereas the outer loop is changed. In the outer loop of MESE, the exploration and improvement processes in the ESE algorithm to update T_h are combined. More specifically, T_h is decreased by multiplying by a factor with continuous variation. The update strategy is expressed as:

$$T_h = T_{h\_old} \times \left(0.9 - \beta_1^{\left((1-C_1)/(n_{acpt}/M-C_1)\right)^{n_1}}\right) \tag{2.7}$$

In Eq. (2.7), the expression can be rewritten as $T_h = T_{h\_old} \times (0.9 - \beta_1)$, when the acceptance ratio n_{acpt}/M approaches to 1. In this situation, T_h can be decreased with a faster speed. Otherwise, T_h is reduced with a slower speed, when n_{acpt}/M is close to a large percentage C_1 (e.g., 0.8). At this time, the expression is gradually changed to $T_h = T_{h\_old} \times 0.9$. Additionally, the power n_1 is applied to further control the variation speed of the factor. For other parameters in Eq. (2.7), β_1 and n_1 are suggested to set as 0.1 and 4, respectively, based on the tested results.

In the same way, T_h can be increased, when the diversity becomes significantly low, and there is no improvement of current design X ($n_{imp} = 0$). The update strategy is derived as:

$$T_h = T_{h\_old} / \left(0.7 + \beta_2^{\left(1+(M/n_{acpt}-1)\times(1-n_{acpt}/M/C_2)\right)^{n_2}}\right) \tag{2.8}$$

It is easy derive that Eq. (2.8) approaches $T_h = T_{h\_old}/0.7$, when n_{acpt} is close to 0. T_h is increased with a fast speed. This can help the search process of MESE fast escaping from a local region. In contrast, Eq. (2.8) will be gradually changed to $T_h = T_{h\_old}/(0.7 + \beta_2)$, if the acceptance ratio n_{acpt}/M approaches a small constant C_2. At this time, the variation speed of T_h can be slightly decreased. Similar to the format of Eq. (2.7), an additional power n_2 is employed to better control the speed of variation for scale factor. Here, the parameters β_2, C_2, and n_2 are set to 0.2, 0.2, and 0.125, respectively, after numerical tests. Moreover, to further accelerate the convergence of MESE, T_h continues to be reduced with a slow speed based on the expression $T_h = \alpha T_{h\_old}$, when the acceptance ratio n_{acpt}/M is within an interval from

C_2 to C_1 and meets one of the following conditions: the improvement ratio n_{imp}/M is larger than zero ($n_{imp} > 0$) or the current design X is worse than the current best design X_{best} over restriction S ($f(X) > S \times f(X_{best})$). The values of relative parameters, α and S could be 0.9 and 1.015, respectively. Regrading to other situations, T_h is maintained. To make the procedure of MESE algorithm clear, Fig. 2.5 shows the flowchart of MESE, whereas Table 2.3 lists the pseudo code of outer loop in the MESE algorithm. Evidently, Eq. (2.7) to (2.8) indicate the scale factors to update T_h can be continuously changed within two separable intervals, and there is no tolerance to divide the update strategy into two processes.

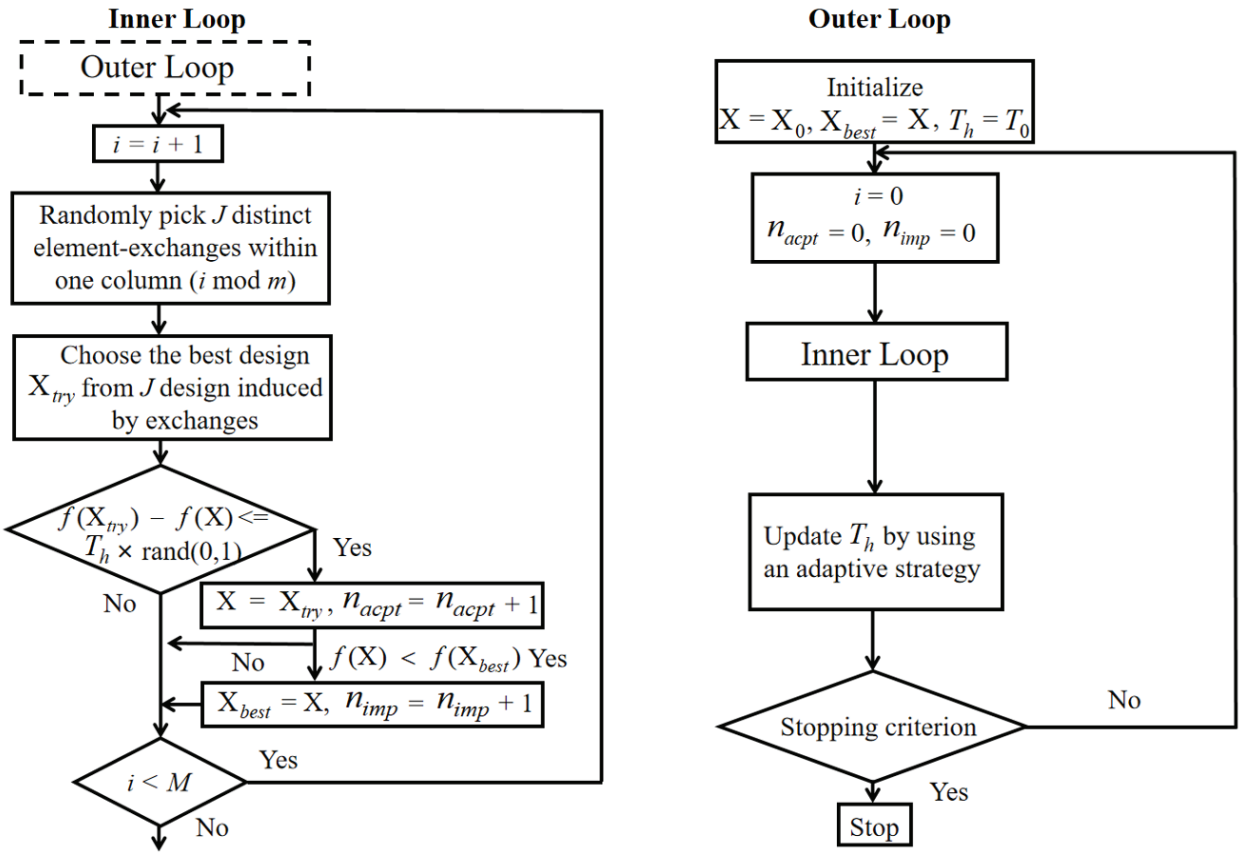


Fig. 2.5 Flowchart of MESE algorithm

Table 2.3 Pseudo code of outer loop in MESE algorithm

Line	Procedure of outer loop in MESE
1	Initialize $\mathrm{X} = \mathrm{X}_0$, $\mathrm{X}_{best} = \mathrm{X}$, $T_h = T_0$ based on initial X_0 design and "temperature" T_0.
2	**While** stopping criterion is not satisfied
3	$n_{acpt} = 0$, $n_{imp} = 0$
4	Perform inner loop.
5	**If** $(n_{acpt}/M) \geq C_1$
6	Reduce T_h.
7	**Else If** $(n_{acpt}/M) \leq C_2$ and $n_{imp} = 0$
8	Increase T_h.
9	**Else If** $C_1 < (n_{acpt}/M) < C_2$ and $((\mathrm{f}(\mathrm{X}) > S \times \mathrm{f}(\mathrm{X}_{best})$ or $n_{imp} > 0))$
10	Continue to reduce T_h.
11	**End If**
12	**End While**

Normally, the process of global optimization for achieving an OLHD starts from a random LHD. Therefore, the convergence speed of optimization significantly depends on the quality of initial design. If its quality is considerably poor, the optimization algorithm must spend long time converging to a near high-quality design. It is easy to derive that there is only one optimal solution, when the size and range of each dimension of a design are determined. Thus, the time cost in the optimization from an initial poor design to the final optimal design can be shortened by shortening the time cost from this initial design to a near high-quality design. In other words, if two preconditions can be met, the whole convergence of optimization can be further speeded up. First, the time cost to obtain a near high-quality design is shorter than that from a poor design to an equal-quality design using global optimization algorithm. Second, there is an algorithm with high global-search capability to escape from the local solution.

In Subsection 2.1.2, a significantly efficient algorithm, TPLHD, using the translational operation without any mathematic evaluations, was reviewed. In practice, TPLHD only spends several seconds generating a near high-quality design with relatively large size (e.g., 100 × 10), while there is almost no time cost for generating a near high-quality design with a smaller size. This time cost is shorter than that spent by an optimization process from a poor design to an equal-quality design using a global optimization algorithm. Therefore, the combination between the TPLHD and MESE or ESE seems to be reasonable for further acceleration of the optimization process to construct an OLHD, particularly in

the beginning of optimization. To consider that, a new combinational algorithm, namely, TPMESE, is proposed. The algorithm aims to further shortening the time cost in the initial stage of optimization through directly starting an optimization process from a near high-quality design generated by the TPLHD algorithm instead of a random LHD with poor quality. It is worth to emphasize that the TPMESE focuses more on achieving a higher central goal of convergence efficiency, particularly when the time is the first concern, or the required size of design is significantly large. Fig. 2.6 shows the procedure of TPMESE algorithm.

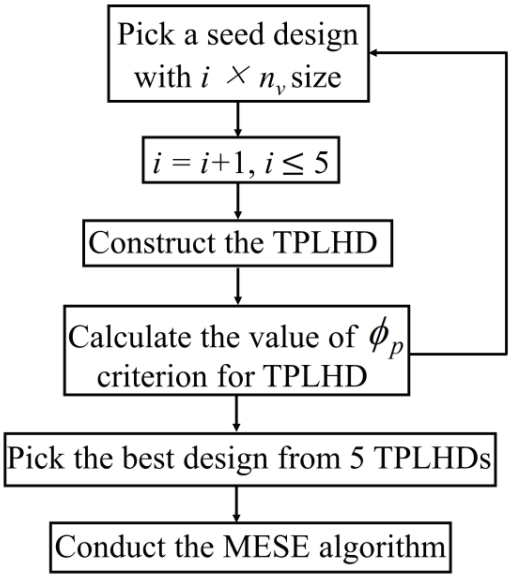


Fig. 2.6 Flowchart of TPMESE algorithm

2.2 Kriging metamodeling

Kriging metamodeling has been widely used to approximate discrete samples. It is useful for the tasks including SBO, exploration in the design space, visualization of the relationship between the variables and objective, sensitivity analysis (SA), and so on. To solve different problems better, some variants have been developed. For instance, the blind [63] and hierarchical [64, 65] Kriging models were proposed to provide a more suitable trend function (mean term) so that the features of discrete samples can be represented using the trend function as much as possible. To give more information in the approximation, the gradient-enhanced Kriging (GEK) model [66, 67] was developed to contain the gradient information of objective with respect to the variables. With the rapid development of computational techniques, computer

simulation becomes increasingly popular. However, most of computer experiments contain all kinds of noise, which may lead to a sudden perturbation from the expected smooth response. To filter out noise and maintain a smooth functional trend, the RKri model was derived [22]. In this subsection, the brief overview of the OK and RKri models are given.

2.2.1 Ordinary Kriging

To generate the OK model, two steps usually need to be implemented. First, a mean term $\hat{\mu}$ can be solved. Second, a Gaussian process $z(\cdot)$ with a mean value zero and standard deviation σ should be implemented. These implementations can help a Kriging model to approximate noise-free data. An expression to construct the mean term and Gaussian process is listed as follows:

$$\hat{y}(\mathbf{x}) = \hat{\mu} + z(\mathbf{x}), \tag{2.9}$$

where the $\hat{y}(\cdot)$ is the prediction of a Kriging model, whereas the $\mathbf{x}$ is an unknown evaluation point. A Gaussian process can be calculated based on a correlation matrix $\mathbf{R}$ ($n \times n$) with respect to n samples and a correlation vector $\boldsymbol{\psi}$ ($n \times 1$) with respect to an unknown point and n samples. Here, the unknown point is $\mathbf{x}_{n+1}$, whereas n samples can be expressed as $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Thus, the Eq. (2.9) can be written as:

$$\hat{y}(\mathbf{x}_{n+1}) = \hat{\mu} + \sum_{i=1}^{n} b_i \, \psi_i(||\mathbf{x}_{n+1} - \mathbf{x}_i \, ||), \tag{2.10}$$

where the element ψ_i in a correlation vector $\boldsymbol{\psi}$ is calculated by:

$$\boldsymbol{\psi} = \begin{pmatrix} \mathrm{Corr}[Y(\mathbf{x}_{n+1}), Y(\mathbf{x}_1)] \\ \vdots \\ \mathrm{Corr}[Y(\mathbf{x}_{n+1}), Y(\mathbf{x}_n)] \end{pmatrix} = \begin{pmatrix} \exp[-\sum_{j=1}^{D} \theta_j (||x_{n+1,j} - x_{1,j}||)^{p_j}] \\ \vdots \\ \exp[-\sum_{j=1}^{D} \theta_j (||x_{n+1,j} - x_{n,j}||)^{p_j} \end{pmatrix} \tag{2.11}$$

In the Eq. (2.11), p_j is a parameter at j^{th} dimension to control the level of smoothness in an approximation. θ_j is the Kriging hyperparameter at j^{th} dimension.

The element b_i from the vector $\mathbf{b}$ ($n \times 1$) is given by:

$$\mathbf{b} = \mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}) \tag{2.12}$$

Here, the $\mathbf{1}$ is the vector of ones ($n \times 1$). To approximate the discrete samples as high quality as possible, the Kriging

hyperparameters should be tuned. In the optimization process, the maximum likelihood estimation (MLE) is usually employed to be as an objective function. Thus, the mean term can be calculated as:

$$\hat{\mu} = \frac{\mathbf{1}^T \mathbf{R}^{-1} \mathbf{y}}{\mathbf{1}^T \mathbf{R}^{-1} \mathbf{1}} \tag{2.13}$$

Here, $\mathbf{y}$ ($n \times 1$ vector) indicates objective values at $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

The predict variance can be calculated by:

$$s^2(\mathbf{x}_{n+1}) = \sigma^2(1 - \boldsymbol{\psi}^T \mathbf{R}^{-1} \boldsymbol{\psi}), \tag{2.14}$$

where the process variance σ^2 is derived as:

$$\sigma^2 = (\mathbf{y} - \mathbf{1}\hat{\mu})^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{1}\hat{\mu})/n \tag{2.15}$$

From the Eq. (2.14), it is easy to see that the predicted variance is equal to zero at a sample point.

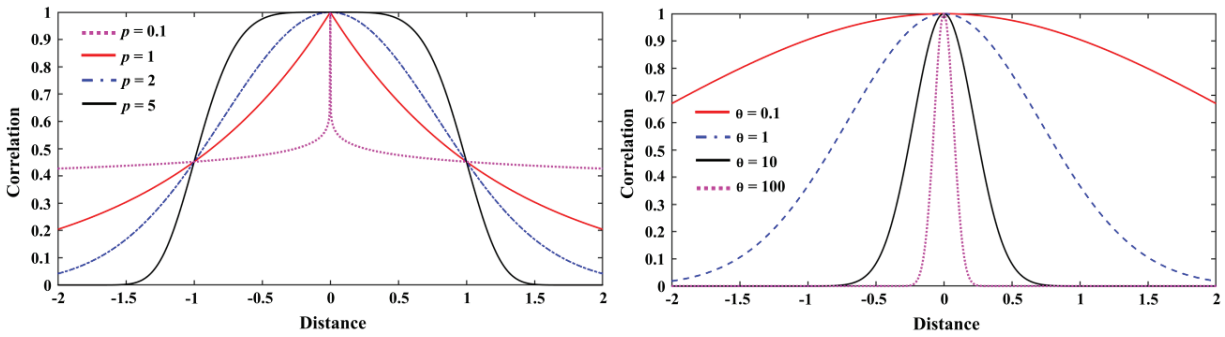
To tune the Kriging hyperparameters, the optimum of MLE function should be searched. In the variants, the marginal likelihood has been widely used [68]. The naturel log of the marginal likelihood can be calculated by:

$$\ln(\mathcal{L}_{marginal}) = \frac{n}{2}\ln(2\pi) + \frac{n}{2}\ln(\sigma^2) + \frac{1}{2}\ln(|\boldsymbol{\psi}|) + \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{1}\hat{\mu})^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{1}\hat{\mu}) \tag{2.16}$$

Thus, Eq. (2.16) can be simplified after removing constants:

$$\ln(\mathcal{L}_{marginal}) = \frac{n}{2}\ln(\sigma^2) + \frac{1}{2}\ln(|\boldsymbol{\psi}|) \tag{2.17}$$

The approximate quality of the Kriging model is also significantly influenced by the format of a correlation function. Eq. (2.11) describes a basic format in correlations between points. It is clear that the correlation depends on the distance between samples in the design space. A higher correlation can be reflected when the distance between two samples is smaller. Thus, the prediction of one point would be more highly influenced by another one. The parameter p_j can control the initial drop of a correlation when the distance between points increases. Apparently, a smaller p_j (e.g., $p = 1$) indicates a faster initial drop in a correlation, which can accommodate a larger difference between two close points. If p_j is relatively large (e.g., $p = 2$), a smaller initial drop in a correlation and a smoother response can be obtained. In the construction of the Kriging model, p_j is normally fixed, and the set of parameters $\boldsymbol{\theta}$ should be optimized based on the MLE function. Here, $\boldsymbol{\theta}$ is $\{\theta_1, \ldots, \theta_d\}$. In the correlation function, $\boldsymbol{\theta}$ indicates an amount of the variation in a correlation.



(a) $\boldsymbol{\theta} = -0.7943$ (b) $p = 2$

Fig. 2.7 Variation of one-dimensional correlation function with different values of p for (a) $\boldsymbol{\theta} = -0.7943$ and θ for (b) $p = 2$.

A smaller value of θ_j indicates that the point is significantly influenced by others. In contrast, a significantly different response can be accommodated when a large value of θ_j is given. Fig. 2.7 shows the variations of the one-dimensional correlation function with the different values of p and $\boldsymbol{\theta}$, respectively. However, even though Eq. (2.11) is most frequent to be used, some variants of correlation functions are probably more useful for a real-world optimization problem. Classically, Matérn class of correlation functions [69] was proposed to better solve engineering problems. Two wide-use formats of Matérn correlation functions are written as:

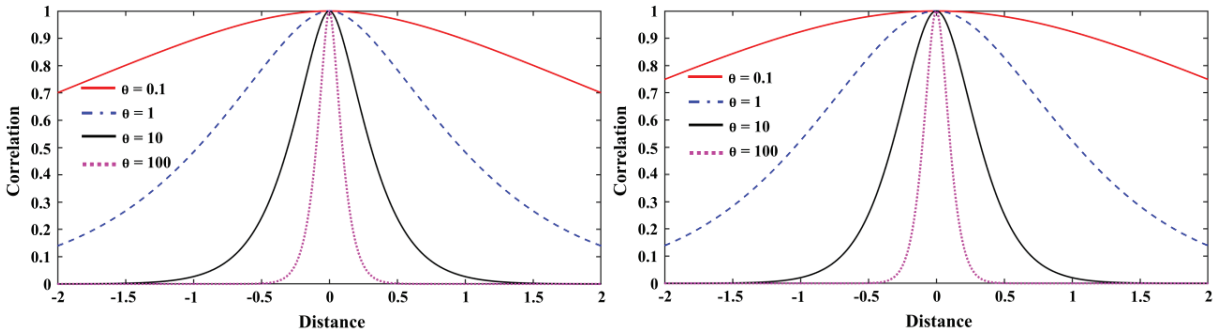
$$\psi(\mathbf{x}, \mathbf{x}')_{\nu=3/2} = \left(1 + \sqrt{3}l\right)\exp\left(-\sqrt{3}l\right), \tag{2.18}$$

$$\psi(\mathbf{x}, \mathbf{x}')_{\nu=5/2} = \left(1 + \sqrt{5}l + \frac{5l^2}{3}\right)\exp\left(-\sqrt{5}l\right), \tag{2.19}$$

where l can be calculated by:

$$l = \sqrt{\sum_{j=1}^{d} \theta_j (x_j - x_j')^2} \tag{2.20}$$

In Eq. (2.18) to (2.19), ν has the similar functionality to p in Eq. (2.11). In the same way, $\boldsymbol{\theta}$ in the Matérn correlation functions indicates the amount of a variation in a correlation. A larger value of θ_j can allow a larger difference of a response. Otherwise, a high correlation exists between points. Fig. 2.8 shows the variations of the Matérn correlation functions with different values of $\boldsymbol{\theta}$.



(a) $\nu = 3/2$ (b) $\nu = 5/2$

Fig. 2.8 Variation of the Matérn correlation functions with different values of $\boldsymbol{\theta}$ for (a) $\nu = 3/2$ and (b) $\nu = 5/2$, respectively.

Besides the Matérn correlation functions, "CUBIC SPLINE" correlation function [66] has been widely used. Generally, "CUBIC SPLINE" is a piecewise function, and its expression can be written as:

$$\psi(\mathbf{x}, \mathbf{x}') = \begin{cases} 1 - 15\xi_j^2 + 30\xi_j^3 & \text{for } 0 \le \xi_j \le 0.2 \\ 1.25\left(1 - \xi_j\right)^3 & \text{for } 0.2 \le \xi_j < 1 \\ 0 & \text{for } \xi_j > 1 \end{cases} \tag{2.21}$$

2.2.2 Regression Kriging with re-interpolation

According to the Eq. (2.10), it is clear to see that an OK model must pass through each sample. However, many factors would produce a sudden perturbation in experiments (e.g., computer experiments like CFD) from the expected smooth response between two close samples. Such perturbation probably changes the main property of a functional trend at a local region in constructing the OK model. It may not cause a problem when the samples are relatively sparse in the design space. However, close samples would be gradually produced when more points are added using a point-infill criterion. It can predict that it is difficult to approximate a smooth function based on those "noisy" samples. In this situation, the interpolation-based method may destroy a smooth functional trend when a point is close with another one. As a result, the Kriging model with poor quality will be generated. Subsequently, the point infill criterion may take more efforts for a wider exploration because effective information to obtain an improvement is significantly changed. Apparently, a regression method (i.e., does not pass through each sample) is more reasonable to be used in this situation.

To achieve a regression process and maintain a smooth functional trend for "noisy" samples, the regression Kriging

(RK) can be employed. To allow that a Kriging model does not pass through each sample, a regression constant λ is imposed on the leading diagonal of a correlation matrix. Thus, Eq. (2.10) can be rewritten as:

$$\hat{y}_r(\mathbf{x}_{n+1}) = \hat{\mu}_r + \boldsymbol{\psi}^T(\mathbf{R} + \lambda\mathbf{I})^{-\mathbf{1}}(\mathbf{y} - \mathbf{1}\hat{\mu}_r) \tag{2.22}$$

Here, the subscript *r* indicates regression. **I** is an identity matrix ($n \times n$).

Correspondingly, the mean term in a RK can be derived as:

$$\hat{\mu}_r = \frac{\mathbf{1}^T(\mathbf{R} + \lambda\mathbf{I})^{-\mathbf{1}}\mathbf{y}}{\mathbf{1}^T(\mathbf{R} + \lambda\mathbf{I})^{-\mathbf{1}}\mathbf{1}} \tag{2.23}$$

The predicted variance for the RK model can be changed to:

$$s_r^2(\mathbf{x}_{n+1}) = \sigma_r^2(1 + \lambda - \boldsymbol{\psi}^T(\mathbf{R} + \lambda\mathbf{I})^{-\mathbf{1}}\boldsymbol{\psi}) \tag{2.24}$$

Here, the process variance of the RK model was:

$$\sigma_r^2 = (\mathbf{y} - \mathbf{1}\hat{\mu}_r)^T(\mathbf{R} + \lambda\mathbf{I})^{-\mathbf{1}}(\mathbf{y} - \mathbf{1}\hat{\mu}_r)/n \tag{2.25}$$

It is easy to derive that the RK no longer forces the predictor to pass through each sample because the correlation of each sample given with itself is not exact when a regression constant is added. Therefore, a sudden perturbation can be filtered out so that a smooth functional trend can be maintained. To obtain as high-quality functional trend as possible, the regression constant λ also needs to be tuned with other Kriging hyperparameters.

However, it can be noticed that the predicted variance cannot be reset to zero at each sample point based on Eq. (2.24). In the previous Eq. (2.14), the predicted variances of the OK model are all zeros at sample points. This property leads to an exact prediction at each sample point. Thus, the value of an expected improvement is also zero. Under this situation, the updating point will not be added at sample points because there is no potential to further improve the quality of the Kriging model and push optimization process towards the global convergence. Nevertheless, a nonzero value of a predict variance produces the exact response of an expected improvement at a sample point. Thus, an updating point can be added into the same location of the previous sample. For a non-deterministic experiment, repeated experiments can reduce the error and obtain a real improvement at the same location. However, for a deterministic computer experiment, such as a CFD simulation, a repeatable error can be produced because the same result will be obtained for the same input given. In other words, the improvement process will be trapped by this point when the process of point infills is implemented at

the sample point. To solve this problem, the re-interpolation technique was proposed by Forrester et al. [22]. Generally, the interpolation-based method was reused to approximate a continuous response surface constructed by the RK. The new surrogate model was named as Rkri, and its expression to predict an unknown point can be rewritten as:

$$\hat{y}(\mathbf{x}_{n+1}) = \hat{\mu} + \boldsymbol{\psi}^T\mathbf{R}^{-1}(\hat{\mathbf{y}}_r - \mathbf{1}\hat{\mu}) \tag{2.26}$$

Therefore, the mean term can be calculated by:

$$\hat{\mu}_r = \frac{\mathbf{1}^T\mathbf{R}^{-1}\hat{\mathbf{y}}_r}{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{1}} \tag{2.27}$$

To verify that the prediction of Eq. (2.26) is equivalent to that of Eq. (2.22). Eq. (2.22) is employed to predict the sample points. Thus, the expression can be given:

$$\hat{\mathbf{y}}_r = \mathbf{1}\hat{\mu}_r + \mathbf{R}(\mathbf{R} + \lambda\mathbf{I})^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}_r), \tag{2.28}$$

where $\hat{\mathbf{y}}_r$ indicates the predictions of Eq. (2.22) at the sample points. It can be expressed by:

$$\hat{\mathbf{y}}_r = \begin{pmatrix} \hat{y}_r(\mathbf{x}_1) \\ \vdots \\ \hat{y}_r(\mathbf{x}_{n+1}) \end{pmatrix} \tag{2.29}$$

Now, Eq. (2.28) is substituted into Eq. (2.27), and $\mathbf{1}^T(\mathbf{R} + \lambda\mathbf{I})^{-1}\mathbf{1}\hat{\mu}_r = \mathbf{1}^T(\mathbf{R} + \lambda\mathbf{I})^{-1}\mathbf{y}$ according to Eq. (2.23). The $\hat{\mu}_r = \hat{\mu}$ can be verified.

Subsequently, Eq. (2.28) is substituted into Eq. (2.26). It is easy to derive that $\hat{y}_r(\mathbf{x}_{n+1}) = \hat{y}(\mathbf{x}_{n+1})$.

Moreover, the predicted variance of the RKri can be rewritten as:

$$s_{ri}^2(\mathbf{x}_{n+1}) = \sigma_{ri}^2(1 - \boldsymbol{\psi}^T\mathbf{R}^{-1}\boldsymbol{\psi}) \tag{2.30}$$

Here, the subscript *ri* denotes the regression with re-interpolation technique. It can be clearly seen that Eq. (2.30) exactly has the same format as Eq. (2.14). Thus, the predicted variance of the RKri can also be reset to zero. Regarding to the process variance. It is easy to previously derive the following expression:

$$\sigma_{ri}^2 = (\hat{\mathbf{y}}_r - \mathbf{1}\hat{\mu}_r)^T\mathbf{R}^{-1}(\hat{\mathbf{y}}_r - \mathbf{1}\hat{\mu}_r)/n \tag{2.31}$$

Now, Eq. (2.28) is substituted into Eq. (2.31). Eq. (2.31) can be changed to:

$$\sigma_{ri}^2 = (\mathbf{y} - \mathbf{1}\hat{\mu}_r)^T(\mathbf{R} + \lambda\mathbf{I})^{-1}\mathbf{R}(\mathbf{R} + \lambda\mathbf{I})^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}_r)/n \tag{2.32}$$

According to the derivations, it can be seen that RKri can simultaneously filter out noise from the expected smooth response and avoid the repeated implementations of point infills at the same sample point. Thus, the optimization process can be progressed towards the global convergence based on deterministic experiments.

2.3 Overview of point-infill criteria for global optimization

When the samples are sparse in the design space, the approximation using the surrogate model has a huge difference from the real functional relationship between the variables and objective. Under this situation, the optimum on the surrogate model is also significantly different from the true optimal solution. Therefore, it is necessary to improve the quality of the approximation, particularly for the region nearby with the optimal solution. To achieve this, the point-infill criteria have been addressed to feed sample points so that the optimization process can be guided towards the global convergence.

2.3.1 Expected improvement criterion

From Subsection 2.2, it can be seen that all kinds of Kriging models can provide not only a prediction at an unknown point but also an estimation of its uncertainty. Thus, it is suitable to employ such stochastic process to develop criteria. Additionally, the Kriging predictor substantially provides a stochastic process in the approximation based on discrete samples. Its estimation at an unknown point is the realization of a stochastic process $Y(\mathbf{x})$. Thus, the prediction subjects to a normal distribution with the mean $\hat{y}(\mathbf{x})$ and standard deviation $s(\mathbf{x})$. Fig. 2.9 shows the probability density function (PDF) at an unknown point $\mathbf{x} = 4.3$.

Accordingly, it is easy to see that there are some probabilities to obtain an improvement which the objective value is better than the current best objective value $y_{\min}$. Generally, the improvement at an unknow point can be expressed as:

$$\mathrm{I}(\mathbf{x}) = \max\left(y_{\min} - Y(\mathbf{x}), 0\right) \tag{2.33}$$

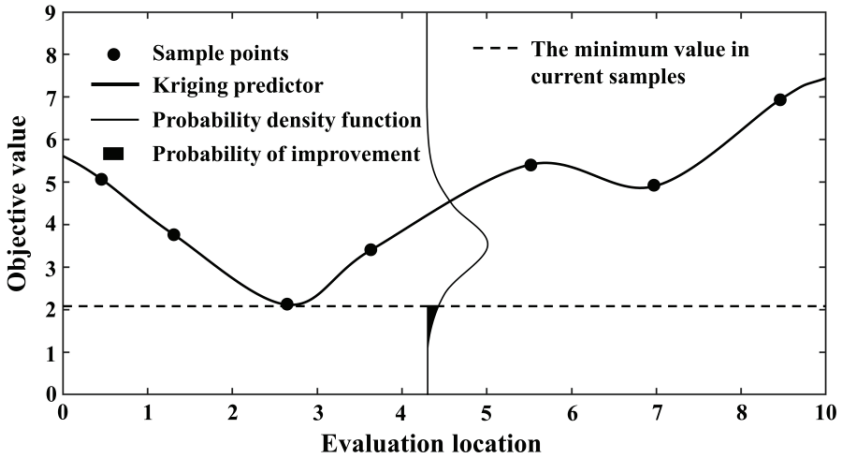


Fig. 2.9 Probability of improvement at an unknown location using Kriging model for approximation

Due to the process $Y(\mathbf{x})$ is a stochastic process, the $\mathrm{I}(\mathbf{x})$ is also a stochastic process. The amount of improvement is determined by the probability which the objective value is better than the current best objective value $y_{\min}$. Thus, the mathematic expectation of improvement, also namely, EI criterion, can be calculated by:

$$\mathrm{EI}(\mathbf{x}) = \int_{-\infty}^{\infty} \max(y_{\min} - Y(\mathbf{x}), 0)\, \phi\big(Y(\mathbf{x})\big)\mathrm{d}Y \tag{2.34}$$

Here, the $\phi(\cdot)$ is the probability density function (PDF) of the normal distribution with mean $\hat{y}(\mathbf{x})$ and standard deviation $s(\mathbf{x})$. Thus, it is equal to:

$$\frac{1}{\sqrt{2\pi}s(\mathbf{x})}\exp\ -\frac{1}{2}\left(\frac{Y(\mathbf{x}) - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right)^2 \tag{2.35}$$

Eq. (2.35) can also be expressed by a PDF $\Phi(\cdot)$ of the standard normal distribution:

$$\phi\big(Y(\mathbf{x})\big) = \frac{1}{s(\mathbf{x})}\Phi\left(\frac{Y(\mathbf{x}) - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) \tag{2.36}$$

When Eq. (2.36) is substituted into Eq. (2.34), Eq. (2.34) can be rewritten as:

$$\begin{cases} \mathrm{EI}(\mathbf{x}) = \big(y_{min} - \hat{y}(\mathbf{x})\big)\,\boldsymbol{\Phi}\left(\dfrac{y_{min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) + s(\mathbf{x})\Phi\left(\dfrac{y_{min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) & \text{if } s > 0 \\ 0 & \text{if } s = 0 \end{cases} \tag{2.37}$$

In Eq. (2.37), $\boldsymbol{\Phi}(\cdot)$ indicates the cumulative distribution function (CDF) of the standard normal distribution. An

updating point can be found through maximizing the EI criterion. Moreover, according to the expression of EI, the value of EI is zero when the predicted variance is zero. This guarantees that there is no probability to implement point infill at the previously sampled locations so that the optimization process can be guided towards the global convergence. Additionally, to efficiently search for the global optimum, the balance between the local exploitation and global exploration is always significantly important. To be clear, a local exploitation is beneficial to obtain a fast convergence in an optimization process, whereas a global exploration can help search process to escape from the local optimum and progress the optimization process towards the global convergence. Eq. (2.37) provides an automatic balance between the local exploitation and global exploration. It can be seen that the value of Eq. (2.37) is simultaneously determined by two terms. Apparently, the search process will focus more on the local exploitation when the prediction is better than the current best objective value. This is because the $\left(y_{min} - \hat{y}(\mathbf{x})\right)$ becomes relatively large and lead to a large response of EI criterion. In contrast, the search process aims more at a global exploration when the predicted variance becomes relatively large as a poor quality of approximation.

2.3.2 Efficient global optimization algorithm

Based on the EI criterion, the EGO algorithm was proposed to gradually update the Kriging model and add new updating points. Generally, the EGO algorithm can be summarized by two steps. First, an initial Kriging model needs to be constructed based on some discrete samples. Second, the maximum solution of the EI function needs to be found using an optimization algorithm. The maximum solution with the maximum EI value indicates a location which has the largest probability to obtain a better solution in the next cycle of point infills. Thus, an additional experiment should be implemented at this point. Once the objective value of the updating point is evaluated, it will be added into the previous samples and update the Kriging model. In the following, the processes to search the global optimum of the EI function and update the Kriging model will be repeatedly implemented until a stopping criterion is reached. The process of the EGO algorithm is also summarized using a pseudo code listed in Table 2.4.

In past decades, the EGO algorithm has been widely used to solve global optimization problems [70]. Its performance was significantly demonstrated. However, the EGO algorithm has an evident limitation. Apparently, only one point can be added in each cycle. In other words, for computer experiments, even though there are adequate outer computational resources, they cannot be made full use of because only one computational process can be carried out. However, the

parallel computational architecture becomes increasingly popular and mature in industry. Making full use of the parallel computational architecture, such as the implementations of expensive computer experiments in a parallel manner, can significantly reduce wall-clock time. To achieve that, multiple updating points also need to be simultaneously found in each cycle of point infills.

Table 2.4 Pseudo code of EGO algorithm

Line	EGO algorithm
1	Input: Initial samples ($\mathbf{x}$, $\mathbf{y}$), where $\mathbf{x}$ and $\mathbf{y}$ are $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and $\mathbf{y} = \{y_1, \ldots, y_n\}$, respectively.
2	**While** stopping criterion is not reached
3	Construct a Kriging model based on the current samples ($\mathbf{x}$, $\mathbf{y}$).
4	Find the maximum solution $\mathbf{x}_{\text{new}}$ of the EI criterion.
5	Evaluate the objective value y_{new} of the updating point $\mathbf{x}_{\text{new}}$.
6	$\mathbf{x} \leftarrow \mathbf{x} \cup \mathbf{x}_{\text{new}}$
7	$\mathbf{y} \leftarrow \mathbf{y} \cup y_{\text{new}}$
8	Update the current best objective value $y_{\mathbf{min}}$ for a minimization problem.
9	Update the current optimal solution $\mathbf{x}_{\text{best}}$.
10	**End While**

2.3.3 Parallel point-infill criteria

The limitation to simultaneously add multiple points in the traditional EGO algorithm is a necessary update of the Kriging model in each cycle of point infills. Before the search for a new updating point, a new Kriging model must be constructed using the current samples so that the EI function can obtain some necessary information from the last updating point. In other words, the EI function also needs to be updated after adding a new updating point in the last cycle of point infills. If there is no evaluation of an updating point, the EI function cannot be updated and find the next updating point. To find multiple updating points, many efforts have been made. Ginsbourger et al. [28] proposed the q-EI criterion, and it is the first extension of the EI criterion to conduct point infills in a parallel manner. In the q-EI criterion, a joint distribution can be naturally defined when q updating points need to be found.

$$\mathcal{N}\left(\left[\hat{y}_r(\mathbf{x}_{n+1}), \dots, \hat{y}_r(\mathbf{x}_{n+q})\right], \mathbf{C}\right), \tag{2.38}$$

where **C** is the covariance matrix ($q \times q$). It can be expressed as:

$$C_{ij} = C\left(\mathbf{x}_{n+i}, \mathbf{x}_{n+j}; \sigma^2, \boldsymbol{\theta}\right) \tag{2.39}$$

Thus, the improvement function of q updating points related to the current best objective y_{min} is:

$$\mathrm{I}\left(\mathbf{x}_{n+1}, \dots, \mathbf{x}_{n+q}\right) = \max\left(y_{\min} - Y(\mathbf{x}_{n+1}), \dots y_{\min} - Y\left(\mathbf{x}_{n+q}\right), 0\right) \tag{2.40}$$

Accordingly, the q-EI function can be calculated by:

$$\mathrm{EI}(\mathbf{x}_{n+1}, \dots, \mathbf{x}_{n+q}) = \mathrm{E}\left[\mathrm{I}\left(\mathbf{x}_{n+1}, \dots, \mathbf{x}_{n+q}\right) | Y(\mathbf{X}) = \mathbf{y}\right] \tag{2.41}$$

Unfortunately, the calculation of the q-EI criterion is significantly expensive, even though several investigations have been made to reduce the computational complexity [28, 71, 72]. Additionally, the search for the optimum in the q-EI criterion is also considerably time-consuming and challenging [72]. A $d \times q$ hyperspace has to be searched using the optimization algorithm because the dimension of the q-EI is $d \times q$. The expensive computation significantly limits the wide application of the q-EI criterion. Thus, some cheaper strategies have been also widely developed.

a. Constant liar strategy

Constant liar (CL) strategy was proposed by Ginsbourger et al. [28]. The strategy is significantly simple. Since the update of the EI function needs information of the new updating point, the objective value of the updating point should be evaluated. To find multiple points without the evaluations of those points, constant values are employed to replace the expensive computer experiments in the CL strategy. Those constants can be the mean, minimum, or maximum values of the current samples. For example, if the minimum value of the current samples is chosen as a constant, the CL strategy uses the minimum value as the evaluation value of the updating point without a real evaluation. Thus, the Kriging model and EI function can be updated by containing the fake sample. The next updating point can then be obtained through maximizing the updated EI function. The process will be repeated until q requested points are chosen. Subsequently, the q updating points will be reevaluated, and the Kriging model can be updated by containing the q samples into the previous samples.

Table 2.5 Pseudo code of CL strategy

Line	CL strategy
1	Input: Initial samples (**x**, **y**), where **x** and **y** are $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and **y** = $\{y_1, \ldots, y_n\}$, respectively.
2	**While** stopping criterion is not reached
3	Construct a Kriging model based on the current samples (**x**, **y**).
4	**For** i = 1 to q
5	**If** i = 1
6	Find the maximum solution $\mathbf{x}_{\mathrm{new}i}$ of the EI criterion.
7	**Else**
8	**.** $\mathbf{x} \leftarrow \mathbf{x} \cup \mathbf{x}_{\mathrm{new}i}$
9	**.** $\mathbf{y} \leftarrow \mathbf{y} \cup y_{liar}$
10	Update the Kriging model based on the current samples (**x**, **y**).
11	Find the maximum solution $\mathbf{x}_{\mathrm{new}i}$ of the EI criterion.
12	**End if**
13	**End for**
14	Evaluate the objective values $\{y_{\mathrm{new}1}, \ldots, y_{\mathrm{new}q}\}$ of the updating points $\{\mathbf{x}_{\mathrm{new}1}, \ldots, \mathbf{x}_{\mathrm{new}q}\}$.
15	Remove q fake "lairs" from **y**
16	$\mathbf{y} \leftarrow \mathbf{y} \cup y_{\mathrm{new}}$
17	Update the current best objective value $y_{\mathbf{min}}$ for a minimization problem.
18	Update the current optimal solution $\mathbf{x}_{\mathrm{best}}$.
19	**End While**

Table 2.5 lists the procedure of the CL strategy. It can be seen that q updating points can be chosen without the evaluations of their objective values. To achieve that, total q + 1 optimizations need to be conducted in each cycle. Specifically, the Kriging hyperparameters need to be tuned once, and search for the optimum on the EI function needs to be carried out q times. However, it is still significantly cheap in comparison with the calculation of the q-EI criterion because only the optimum on the cheap EI function needs to be found. Even though the CL strategy shows cheap computational burden and performs robustly in the numerical tests, its disadvantage is still obvious. That is because a constant "liar" rather than the real evaluation usually gives some fake information in the updates of a kriging model. Then, the updated EI function received the fake information probably guides the CL strategy to explore some regions without true values of improvement. As a result, more cycles of point infills must be implemented so that a high-quality convergence can be reached.

b. Pseudo expected improvement criterion

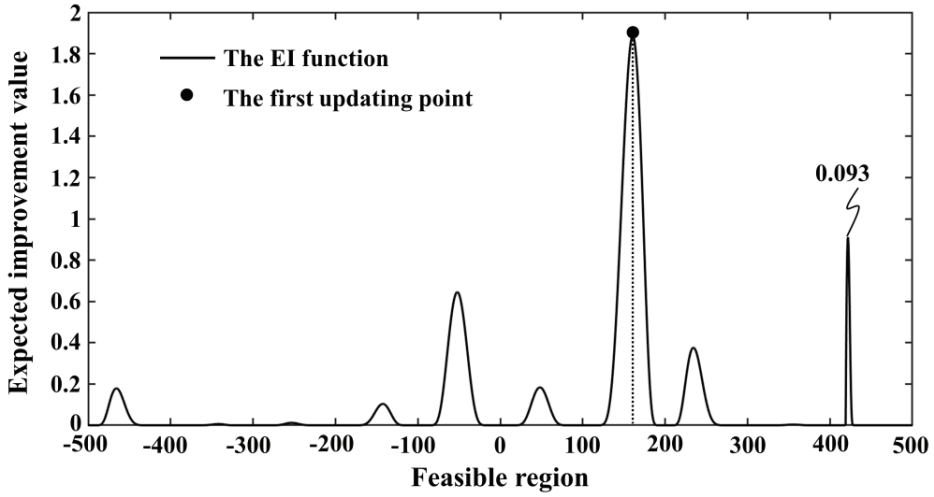
To overcome the aforementioned disadvantage of the CL strategy, a more appropriate method is to simulate the property of an updated EI function after adding an updating point. Based on the idea, a parallel point-infill criterion, namely, PEI, was proposed by Zhan et al. [29]. Fig. 2.10 shows the variation of the EI function before and after adding one point. It can observe that the response of the EI function was significantly reduced in the region nearby with the updating point, whereas there is only a slight variation in the region far away from the updating point. Apparently, the influence of an updating point on the update of the EI function is significantly dependent on the distance. A higher influence of an updating point on the EI function is caused by a smaller distance between the location and the updating point. In contrast, A larger distance corresponds to a smaller influence.

After summarizing the main properties of the EI function after adding an updating point, it is natural to multiply the EI function by an influence function (IF) so that the impact of the updating point on the EI function can be roughly approximated. Based on it, multiple updating points can be simultaneously obtained without the updates of the Kriging model. The IF probably has different formats, but some main features must be contained.

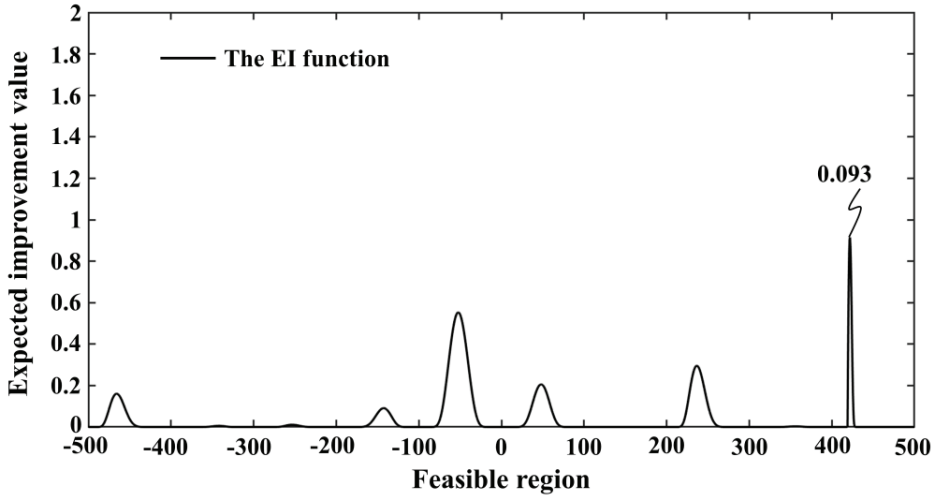
- The IF must be a continuous function in the design space.
- The IF is only relative with the distance between an evaluation location and an updating point. More specifically, the IF needs to be zero at the same location as the updating point but one in the region far away from the updating point.
- Recalling the classical form of the correlation function defined in Eq. (2.11), it can be seen that the function has the value zero at a location far away from the sample points but the value one at the sample points. Thus, a following form exactly contains the main features of the IF.

$$\mathrm{IF}(\mathbf{x}_u, \mathbf{x}_{u+1}) = 1 - \mathrm{Corr}[Y(\mathbf{x}_u), Y(\mathbf{x}_{u+1})] \tag{2.42}$$

Here, $\mathbf{x}_u$ and $\mathbf{x}_{u+1}$ are the updating point added early (sampled location) and the candidate point (unknown location) which needs to be added later, respectively. According to Eq. (2.42), it is easy to find that the IF is only associate with the distance between an unknown location and the sampled location. When the location approaches to the sampled location, the IF has value of zero, whereas the IF is equal to one at the location far away from the sampled location.



(a) Last iteration



(a) Next iteration

Fig. 2.10 Variation of EI function in two iterations

Accordingly, the PEI criterion can be defined in basis of the IF. The process to search q updating points using PEI criterion can be described in turn. When an initial Kriging model was constructed based on the initial samples, or a Kriging model was updated after adding points. The first updating point can be found by searching for the optimum on the EI function. Thus, an equation can be defined as:

$$\mathbf{x}_{n+1} = \text{argmaxEI}(\mathbf{x}), \tag{2.43}$$

where the function $\text{argmaxEI}(\cdot)$ returns the location with the maximum EI value. In the following, the PEI function for searching the second updating point can be defined by multiplying the initial EI function by the IF function. It can be written as:

$$\text{PEI}(\mathbf{x}, \mathbf{x}_{n+1}) = \text{EI}(\mathbf{x}) \cdot \text{IF}(\mathbf{x}, \mathbf{x}_{n+1}) \tag{2.44}$$

Therefore, the second point can be found by maximizing the PEI function.

$$\mathbf{x}_{n+2} = \text{argmaxPEI}(\mathbf{x}, \mathbf{x}_{n+1}) \tag{2.45}$$

In the same way, the PEI function for searching for the qth updating point can be written as:

$$\text{PEI}\left(\mathbf{x}, \mathbf{x}_{n+1}, \mathbf{x}_{n+2} \dots, \mathbf{x}_{n+q-1}\right) = \text{EI}(\mathbf{x}) \cdot \text{IF}(\mathbf{x}, \mathbf{x}_{n+1}) \cdot \text{IF}(\mathbf{x}, \mathbf{x}_{n+2}) \cdot \dots \cdot \text{IF}(\mathbf{x}, \mathbf{x}_{n+q-1}) \tag{2.46}$$

Now, the q^{th} updating point can be searched by:

$$\mathbf{x}_{n+q} = \text{argmaxPEI}\left(\mathbf{x}, \mathbf{x}_{n+1}, \mathbf{x}_{n+2} \dots, \mathbf{x}_{n+q-1}\right) \tag{2.47}$$

2.3.4 Parallel EGO algorithm based on PEI criterion

Based on the PEI criterion, multiple points can be searched in each cycle. Thus, a parallel EGO algorithm using the PEI criterion (EGO-PEI) for point infills was developed by Zhan et al. [29]. Table 2.6 lists the pseudo code of the EGO-PEI criterion. Similar to the CL strategy, when q candidate points need to be found in turn using the PEI criterion, a total of $q + 1$ optimizations should be carried out in each cycle of point infills. Specifically, the Kriging hyperparameters need to be tunned once, whereas q optimizations for searching the optimum on the PEI function should be carried out. Fortunately, the PEI function is just a product with respect to the EI function and IF/IFs. Considering the calculation of the IF is significantly cheap, the computational burden of the PEI criterion is almost no increase in each optimization compared with the EI function. Generally, the computational cost for finding q updating points using the PEI criterion can be ignored compared with the expensive computer experiments and multipoint-infill criterion, such as q-EI.

Table 2.6 Pseudo code of EGO-EPI algorithm

Line	EGO-PEI algorithm
1	Input: Initial samples (**x**, **y**), where **x** and **y** are $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and **y** = $\{y_1, \ldots, y_n\}$, respectively.
2	**While** stopping criterion is not reached
3	Construct a Kriging model based on the current samples (**x**, **y**).
4	**For** i = 1 to q
5	**If** i = 1
6	Find the maximum solution $\mathbf{x}_{\text{new}i}$ of the EI criterion.
7	**Else**
8	Find the maximum solution $\mathbf{x}_{\text{new}i}$ of the PEI criterion.
9	**End if**
10	**End for**
11	Evaluate the objective values $\{y_{\text{new}1}, \ldots, y_{\text{new}q}\}$ of the updating points $\{\mathbf{x}_{\text{new}1}, \ldots, \mathbf{x}_{\text{new}q}\}$.
12	$\mathbf{x} \leftarrow \mathbf{x} \cup \mathbf{x}_{\text{new}}$
13	$\mathbf{y} \leftarrow \mathbf{y} \cup y_{\text{new}}$
14	Update the current best objective value $y_{\mathbf{min}}$ for a minimization problem.
15	Update the current optimal solution $\mathbf{x}_{\text{best}}$.
16	**End While**

2.4 Influence of optimizer on EGO-PEI algorithm

The processes to tune the Kriging hyperparameters and find the optimum of a point-infill criterion are all optimizations. To construct a high-quality Kriging model and search for an updating point with the maximum improvement, an appropriate optimizer should be carefully selected. Before that, the influence of an optimizer on the EGO-PEI algorithm is summarized. According to the procedure of the EGO-PEI algorithm, a total of $N \times (q + 1)$ optimizations should be carried out. Here, N is the number of cycles to implement point infills. It can be seen that the processes to find the updating points using the PEI criterion have to be implemented in a sequential way. The remaining updating points can only be found when the early points were obtained. This is because the search for the next updating point must use the information (the IF function) of the last updating point. Thus, it is clear to see that the early updating points will significantly influence the calculations with respect to the locations of remaining updating points. Unfortunately, the EI and PEI functions show highly multimodal properties [29, 73], which makes the search for the optimum on the EI and PEI functions becomes

significantly challenging. In the literature [29], the influence of the optimizer using the differential evolutionary (DE) algorithm was investigated. The results show that the convergence level of the DE optimizer significantly influences the performance of the EGO-PEI algorithm. In general, the optimizer with the capability for searching a higher-quality solution can promote a better convergence of the EGO-PEI algorithm. Meanwhile, the EGO-PEI also shows some internal robustness, even if the global optimum in the PEI function was missed by the optimizer. However, DE probably performs well in the simple optimization problems (e.g., unimodal or low-dimensional problems), but it is difficult to search a high-quality solution in the complicated problems, such as highly multimodal and high-dimensional problems [74].

Not only that, the Kriging hyperparameters also need to be tuned at the beginning of each cycle for the implementation of point infills. Because the property of the initial PEI function (Essentially, the EI function) is determined by the Kriging model, the remaining predictions of updating points will be influenced in turn. It can predict that a Kriging model with poor quality must take more efforts for improving its quality, which results in more cycles of point infills. Moreover, A poor-quality Kriging model provides considerable ineffective information in the process of point infills, particularly in high-dimensional or highly multimodal problems. As a result, a stochastic process in point infills is implemented. The convergence speed of point infills is then significantly slowed down. Initially, some gradient-based optimization algorithms, such as sequential quadratic programming (SQP) [75], have been widely used to fast tune the Kriging hyperparameters. However, those gradient-based algorithms are easy to be trapped by a local optimal solution. With the fast development of global optimization algorithms and computational techniques, some “heavy” strategies based on global optimization algorithms have been considered to obtain a high-quality Kriging model [75].

Another point needs to be further notable. The difference of the performance between the EGO-PEI and traditional EGO algorithms should be smaller with more their similarities. However, the EGO-PEI significantly performs better than the traditional EGO algorithm in most tested cases [29]. A convincing explanation can be derived. A reasonable balance between the difference and approximation between the EI and PEI functions provides the higher performance of the EGO-PEI algorithm. To further explain the mechanism, the one-dimensional Griewank function with a feasible region [-10, 10] was employed to carry out point infills. In the Griewank function, the optimal function fitness $y = 0$ can be obtained at the location $x = 0$. Fig. 2.11 shows the results with six point infills using the EGO-PEI and EGO algorithms, respectively. In Fig. 2.11, the responses of EI at the fourth and fifth point infills were shown on the left side, whereas the convergence curves of two algorithms were shown on the right side. It can clearly observe that the PEI and EI criteria provide an

approximation between them because some similar responses (similar peaks) exist in the feasible region. Meanwhile, differences between them are also distinctly shown in Fig. 2.11. The factors to produce these differences can be analyzed. On the one hand, the IF can only provide an approximation between the EI and PEI criteria rather than the exact same simulation with respect to the updates of the EI function. On the other hand, the difference will be cumulated with the progress of point infills towards. From Fig. 2.11, the different highest responses of EI provide two updating points in the fourth iteration of point infills. However, the two updating points are still far away from the global solution. In the next iteration, a location with the highest response which is significantly close to the global solution is produced in the PEI function. As a result, the EGO-PEI algorithm can faster achieve the global convergence in the fifth iteration. In contrast, the EGO algorithm must run more iterations to obtain the global convergence with a slower convergence speed. The mechanism of a reasonable balance between the PEI and EI criteria for promoting convergence of the EGO-PEI algorithm can be further analyzed. In comparison with the EGO algorithm, the difference between the EI and PEI criteria brings more diversities to the EGO-PEI algorithm. Thus, a certain randomness is added into the process of the point infills. Now, suppose the EGO algorithm drops into a local region nearby with the current best solution but still far away from the global solution, a quite natural method is to give slight randomness so that the algorithm can escape from the local region. In the same way, the difference between the EI and PEI criteria can be treated to dynamically impose some biases on the EI function, which provides more options for the search process. Once a region nearby with the global solution is explored, the approximation between the EI and PEI criteria can guarantee the EGO-PEI algorithm to converge properly.

Then, our viewpoints can be further summarized as follows:

- The performance of an optimizer to search for the global optimum significantly influences the approximation between the PEI and EI criteria.
- The approximation makes the EGO-PEI algorithm run through a similar mechanism to the traditional EGO algorithm. This guarantees the EGO-PEI algorithm performs properly. However, the difference rather than the same between the EI and PEI criteria brings some randomness and diversities to the EGO-PEI algorithm, which promotes the convergence of the EGO-PEI algorithm on certain occasions.
- A more approximation between the PEI and EI criteria can be provided by a higher global search capability of an optimizer (with a more global exploration for reaching the real global convergence), whereas a solution with lower quality searched by an optimizer (with a more local exploitation for fast obtaining a near-optimal solution) can give

more difference between the PEI and EI criteria. In other words, a reasonable balance with respect to the approximation and difference between the PEI and EI criteria can be produced when an appropriate optimizer is given.

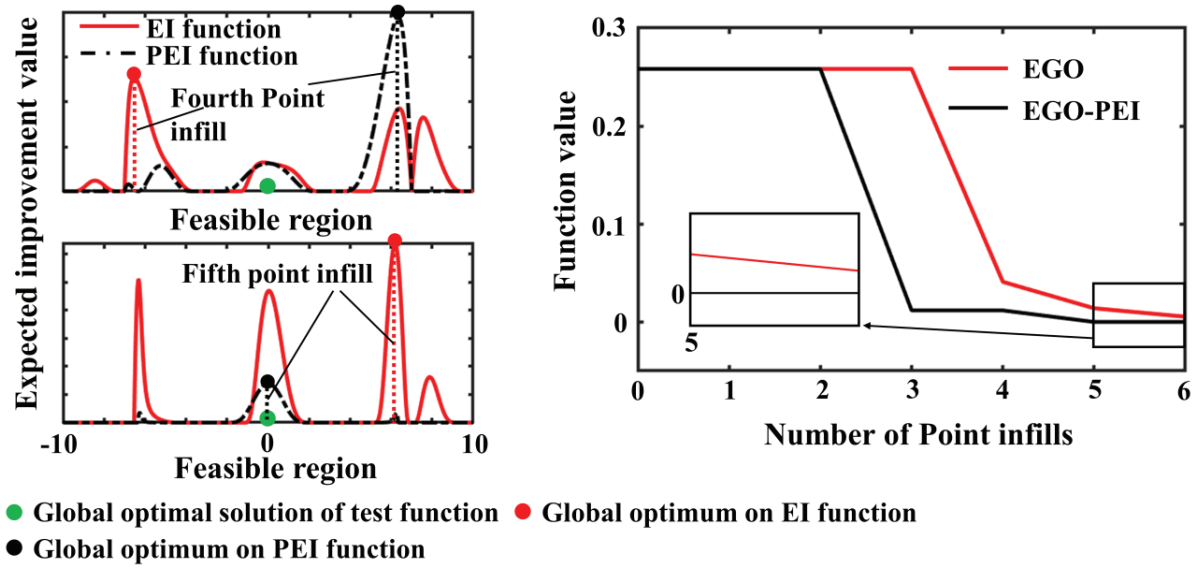


Fig. 2.11 A reasonable balance with respect to difference of PEI from EI for promoting convergence of EGO-PEI algorithm

Therefore, to select an optimizer for guaranteeing the performance of the EGO-PEI algorithm based on the RKri model, the influence with respect to the performance advantages of some state-of-the-art global optimization algorithms on the EGO-PEI algorithm were comprehensively investigated. The investigation can help us not only to choose a suitable optimizer, but also to provide a reference for the selection of an optimizer for other investigators. To implement the investigation, three traditional algorithms which have been widely used in the EGO algorithm, including a real-coded genetic algorithm (RCGA) [76], DE, and basic PSO with a constriction factor [77], were investigated. Additionally, two variants of DE algorithm, namely, success-history based adaptive DE (SHADE) [78] as the winner of CEC 2013 and adaptive guided DE (AGDE) [79], were compared. Not only that, two blending algorithms, namely, bare bones particle swarm optimization integrated DE (BBPSO-DE) [80] and modified Gaussian bare bones DE (MGBDE) [81], were also chosen to take part in the competition. The reason to choose these algorithms can be further explained. First, the three traditional algorithms, RCGA, DE, and PSO, can be treated as standards so that the influence with respect to the

performance advantages of other algorithms on the EGO-PEI algorithm can be clearer. Second, the selected algorithms comprehensively contain different performance advantages. Here, the RCGA, DE, SHADE, and AGDE focus more on the global exploration. The PSO algorithm aims to fast convergence through the personal cognition and social cooperation of particles. With regard to two blending algorithms, BBPSEO-DE and MGBDE, they emphasize to obtain a higher central goal of exploration-exploitation. Accordingly, it is clear to see that the influence of three performance advantages with respect to optimizers, including the global search capability, fast convergence, and better balance between the global exploration and local exploitation, can be comprehensively investigated. Based on the investigation, the performance advantage of an optimizer with the highest impact on the EGO-PEI algorithm can be estimated.

2.5 Efficient system for SBO with "noisy" computations

In SBO, if CFD simulations are employed to evaluate the objective values of evaluation locations, some sudden perturbations from the expected smooth response in fitting a surrogate model is impossible to avoid [22]. Many factors can provide such perturbations: Changes in the variation of a flow structure after the deformation of a geometry, changes in a computational mesh (e.g., a variation with respect to the local quality of a mesh after deformation), and an unavoidable error in the result between a numerical simulation and the corresponding true solution. In other words, a perturbation can still be produced even though a CFD simulation shows the exact same physical property as the physical experiment. This perturbation probably provides a relatively large difference between two close evaluation points and significantly changes the main property of a smooth trend. Here, such perturbation from the expected smooth response is what we treat as noise in the study [22]. Considering the perturbation is essentially produced by CFD simulations, CFD simulations can be treated as "noisy" computations. Now, if an interpolation-based method (must pass through each sample) is employed to fit the "noisy" raw samples, the perturbation must be directly contained. This may not cause problem when the samples are relatively sparse in the design space, whereas the smooth functional trend is possible to be significantly changed when the samples become increasingly dense, particularly in the implementation of point infills. As a result, the implementation of point infills must make more efforts to guide the optimization process towards the global convergence. In comparison with the interpolation-based method, a regression method (i.e., does not pass through from each sample), such as the RKri model, is more appropriate to fit the "noisy" samples and maintain the expected smooth functional trend. Based on it, some important information, such as a specific trend nearby with the current best solution, can be correctly shown. Fig. 2.12 explains the difference between the interpolation and regression methods for fitting the "noisy" samples. It can be

seen that the regression method clearly shows a potential location for the improvement through maintaining a smooth response between samples. In contrast, the functional trend near with some close samples is significantly changed when an interpolation-based method is used. Consequently, a potential location for the improvement cannot be found on the response fitted by the interpolation method.

The shape optimization of vehicle aerodynamics is significantly expensive. To evaluate the aerodynamic performance (e.g., Cd) of different geometries, CFD simulations are more suitable to be used. Unfortunately, some noise is impossible to avoid using a CFD simulation. Therefore, to efficiently solve real-world optimization problems with "noisy" computations, an improved system, namely, RKri-EGO-PEI, was proposed. The system utilizes the RKri to directly filter out noise from the expected smooth response so that a high-quality approximation nearby with the current best solution can be generated. This can reduce the total number of point infills. Moreover, the EGO-PEI algorithm allows the implementation of point infills in a parallel manner, which can make full use of the outer computational resources, thus further reducing the wall-clock time in an optimization process. To guarantee the high-quality processes for tunning Kriging hyperparameters and searching for the optimum on the PEI function, a suitable optimizer should be chosen. Additionally, to initially construct a high-quality surrogate model using as few samples as possible, the OLHD based on TPMESE algorithm is employed to efficiently sample the initial evaluation locations with high space-filling quality. Fig. 2.13 shows the proposed system.

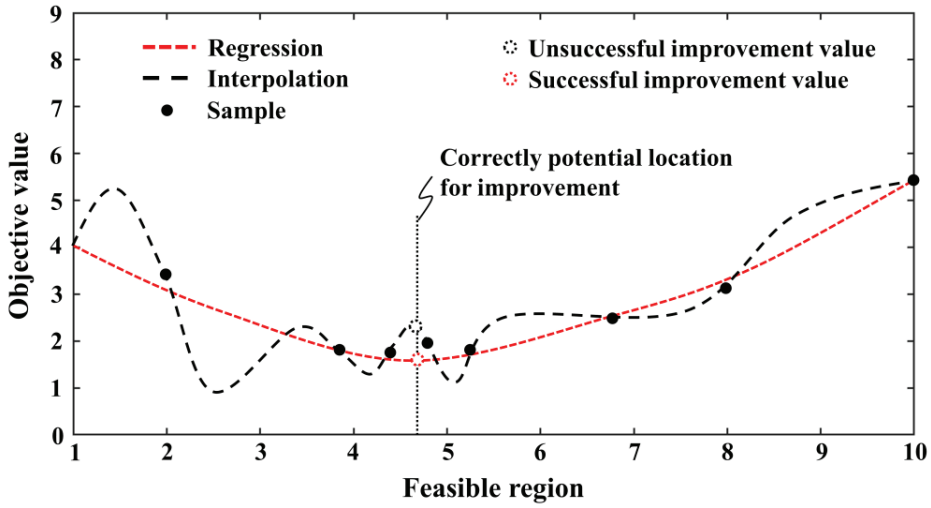


Fig. 2.12 Difference between interpolation and regression methods for fitting "noisy" samples

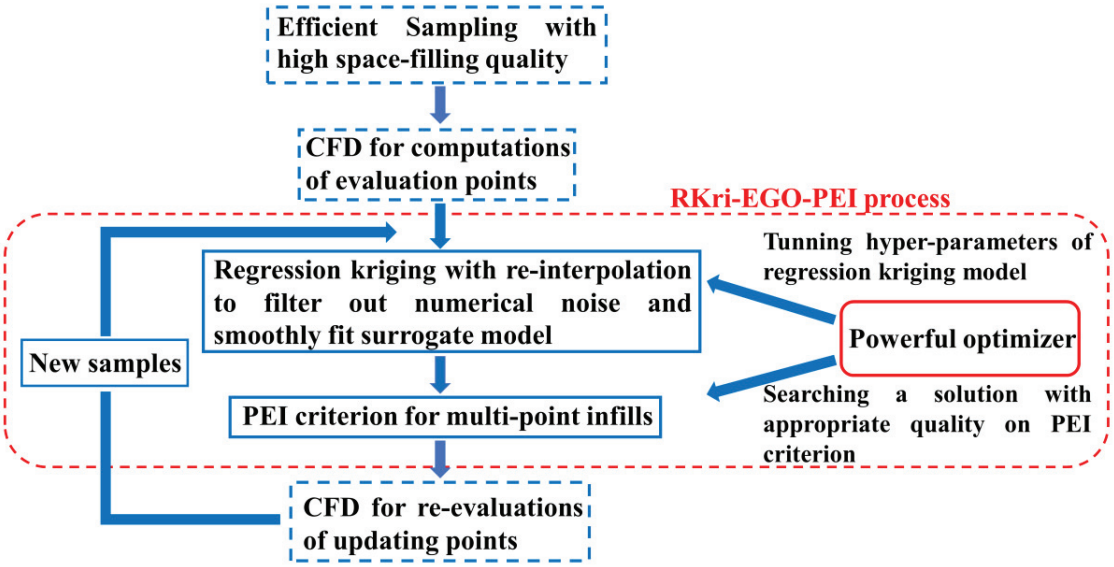


Fig. 2.13 Framework of RKri-EGO-PEI system

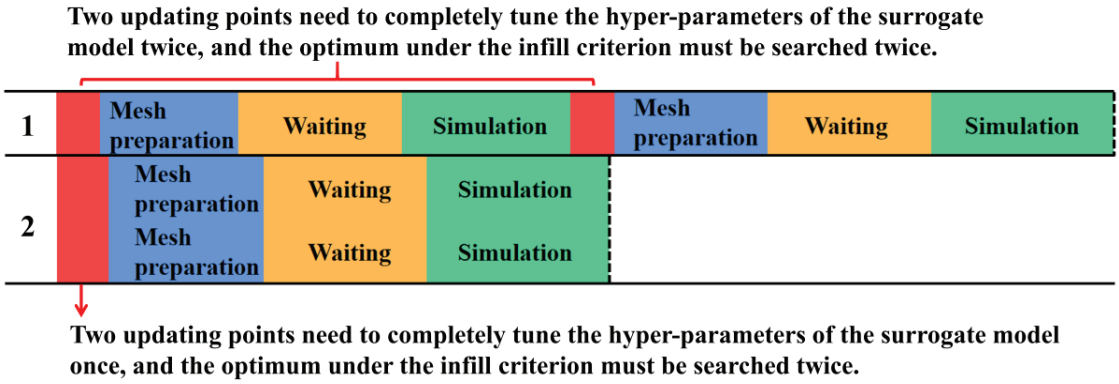


Fig. 2.14 Comparison of wall-clock time costs using different point-infill methods with two point infills

To further demonstrate the potential efficiency of the proposed system, the costs of wall-clock time by different point-infill methods are roughly compared. Normally, the process of point infills based on a CFD simulation contains other time-wasting steps, e.g., mesh preparation, waiting, and numerical simulation. For a serial point-infill method, all time-wasting steps must be implemented in turn even though adequate outer computational resources can be provided. By contrast, all time-wasting steps can be carried out in a parallel manner. Fig. 2.14 roughly evaluates the wall-clock time costs using different point-infill methods with 2 point infills. Here, the method 1 indicates the implementation of point infills using a serial point-infill method, whereas the method 2 denotes a parallel method to carry out point infills. From

Fig. 2.14, it is clear to see that the method 1 must cost double wall-clock time, similar to a parallel way does. Moreover, the total optimizations can be reduced when the parallel point-infill method is used. Fig. 2.14 clearly shows the reason. In the process of two point infills, the Kriging hyperparameters only needs to be tuned once using the parallel point-infill method, whereas it needs to be tuned twice based on a serial point-infill method.

2.6 Conclusion

In this chapter, a new algorithm, namely, MESE, is proposed to construct OLHD with high space-filling quality. The algorithm combines the exploration and improvement processes of ESE algorithm using a new updating strategy of "temperature". The strategy can update the "temperature" in terms of some scale factors with variable step length so that the "temperature" can meet the complicated requirement of update in the optimization of LHD. Additionally, because the TPLHD algorithm just has the translational operation instead of the mathematic evaluation of space-filling criterion, it can rapidly construct a near high-quality design almost without time costs. Therefore, it is natural to employ the TPLHD as the starting design for the MESE algorithm. The combinational algorithm based on the TPLHD and MESE algorithm is named as TPMESE, which aims to further accelerating the convergence speed of optimizing LHD at the beginning.

Subsequently, the OK and RKri surrogate model are reviewed. Compared with the OK model, the RKri model adds a regression constant to the leading diagonal of the correlation matrix, and thus the RKri model can allow the approximation does not pass though from the sample points. Consequently, sudden perturbations from the expected smooth response can be filtered out. Additionally, the re-interpolation is applied to reset the uncertainty to zero at an existing sample point, which can avoid the point infill at the sample points for deterministic computer experiments, such as CFD simulations.

Furthermore, the point infill methodologies, including EGO, CL, and EGO-PEI algorithms, are introduced. As explained in Subsection 2.3, the EGO-PEI algorithm can not only work well through a similar mechanism to the traditional EGO algorithm, but also make full use of outer computational resources through choosing multiple points simultaneously. Thus, it is suitable to be adopted as the point-infill strategy in expensive optimization problems.

Since the approximation rather than exact same between the PEI and EI criteria makes the EGO-PEI algorithm perform better than the traditional EGO algorithm on certain occasions, the influence of optimizers on the approximation and difference between the EI and PEI criteria for promoting the convergence of EGO-PEI algorithm is roughly analyzed. In summary, the global search capability of an optimizer can provide more accurate approximation between the EI and PEI

criteria, thereby ensuring the EGO-PEI carries out trough closer mechanism of action with the traditional EGO algorithm, whereas the local search capability of an optimizer to a near optimum rather than the real global optimum leads to more difference between the EI and PEI criteria, thus bringing more diversities to the EGO-PEI algorithm and promoting it to converge better under certain occasions.

Finally, an improved optimization system is proposed. The system, namely, RKri-EGO-PEI, aims to filtering out noise from the expected smooth response using the RKri model and carrying out the point infills in a parallel manner based on the EGO-PEI algorithm. Therefore, the RKri-EGO-PEI system can extract a smooth functional trend from “noisy” samples, so that a high-quality approximation nearby with the current best solution can be maintained. Moreover, the system can also make full use of the outer computational resources for the further reduction of practical processing time in expensive optimization problems though the parallel implementation of time-consuming steps.

Chapter 3

A toolbox for SBO

A typical SBO usually contains sampling, surrogate metamodeling, optimization process, and result visualization. To achieve these functionalities, some corresponding techniques, such as design of experiments (DOE), optimization algorithms, and point infill strategies, need to be provided. In recent time, some famous programs have been programmed to give users a convenience in SBO. Typically, in MATLAB, the design and analysis of computer experiments (DACE) [82] and the object-oriented DACE (ooDACE) [83] toolboxes were developed to achieve Kriging metamodeling. They provide high extendable programs so that users can define new Kriging predictors, optimizers, point-infill strategies, and so on. To achieve the functionalities of the DOE, MATLAB provides some built-in functions for fast sampling using some mature strategies. To carry out the optimization process, the optimization toolbox with various optimization algorithms has been continuously updated. In Python, many programs for surrogate metamodeling have been widely used to construct surrogate models. Similar to the ooDACE toolbox, the Pykrige [84] was released to generate all kinds of Kriging models. Meanwhile, other surrogate techniques, such as neural networks (NNs), can also be achieved using the open-source program like PyTorch [85]. Additionally, there is a toolbox, namely, surrogate optimization toolbox in Python (pySOT) [86], with more completive functionalities of SBO has been continuously renewed.

However, those toolboxes either only provide some basic functionalities rather than the most advanced techniques in SBO or are difficult to be extended by users. Therefore, to provide a program with highly extendable capability and more advanced techniques in SBO, a toolbox, namely, VAEO toolbox in MATLAB was developed. As the name of the toolbox implies, the VAEO toolbox not only aims to providing the most advanced and completive techniques in SBO, it also focuses on integrating the most efficient and robust schemes for solving complicated engineering optimization problems, such as vehicle aerodynamics. To show the extendable capability and adequate functionalities of the VAEO toolbox, Subsections 3.1 and 3.2 introduce the functionalities and programing design of the VAEO toolbox, respectively. To demonstrate the performance of the VAEO toolbox, the VAEO toolbox was used to compared with the ooDACE toolbox for Kriging metamodeling and the pySOT for solving a global optimization problem in Subsection 3.3.

3.1 Functionalities

The VAEO toolbox is programmed for robustly and efficiently solving all kinds of optimization problems using SBO method. Therefore, various functionalities are integrated into the VAEO toolbox. Generally, to provide a high extendable program, different functionalities were achieved by functions, and these functions were then packaged into the classes with their arguments. The main functionalities in VAEO toolbox include:

1) DOE includes: Sobol sequence [87] with 40 maximum dimensions, LHD, and OLHD constructed by ESE, MESE, and TPMESE algorithms.
2) Kriging metamodeling involves: OK, RKri, HK, and GEK models.
3) Optimizer contains: Gradient-based optimizers including SQP, SQP stating a point found by the golden search (GS-SQP) algorithm [88], and SQP starting a point found by the GA (GA-SQP) [88], and "heavy" strategies including RCGA, DE, PSO, AGDE, SHADE, BBPSO-DE, MGBDE, BBPSO, and BBPSO with crossover and mutation operators of DE based on lbest topology (BBPSO-MC-lbest) [80]. All "heavy" strategies were implemented using the vectorization so that their efficiency in operations can be accelerated as much as possible.
4) In EGO, the EI, weighted EI (WEI) [89], PEI criteria were programmed to progress the optimization towards the global convergence by adding a single point or multiple points. To obtain a well distributed Pareto front with high-quality convergence, the expected hypervolume improvement (EHVI) [90], Euclidean distance-based EI matrix (EIMe) [31], and Euclidean distance-based PEI matrix (PEIMe) [91] criteria are available to implement point infills in a series or parallel manner. Not only that, to solve the global optimization problem with inequality and equality constraints, the constrained EI (CEI) [73] and EI versus probability of feasibility function (EIvsPOF) [92] criteria were integrated for single-point infill or multi-point infills.
5) SA contains: Sobol global SA (SGSA) [93] for the calculations of the first- and total-order sensitivity indices of variables in terms of a surrogate model.
6) Other utilities: To test the performance of new techniques, adequate benchmark suites are provided. These suites contain all kinds of benchmark functions with respect to the global optimization, multi-objective optimization, and constrained global optimization. Additionally, VAEO toolbox provides plots with respect to the response of a Kriging model, SA, cross validation, and convergence process. To implement statistical analysis, all kinds of statistical tests, e.g., *t*-test [42], Wilcoxon signed-rank [94], and Friedman tests [94], were repackaged.

Table 3.1 lists the main file structure of the VAEO toolbox, the functionalities are clearly exhibited.

Table 3.1 Main file structure of VAEO toolbox

VAEO toolbox
+VAEO
+VAEO/@GaussianProcess
+VAEO/@Kriging
+VAEO/@RegressionKriging
+VAEO/@HierarchicalKriging
+VAEO/@GradientEnhancedKriging
+VAEO/@Optimizer
+VAEO/@DOE
+VAEO/@EGO
+VAEO/@SensitivityAnalysis
+VAEO/utilities/@Statistical
+VAEO/utilities/@ErrorAnalysis +VAEO/utilities/metric.m +VAEO/basicFunction/correlation.m
+VAEO/basicFunction/doe_basic_func.m

3.2 Programing design

To provide a highly extendable and reusable program for users, the object-oriented programming (OOP) is employed to design the VAEO toolbox. Fig. 3.1 shows the main operations of Kriging metamodeling and the implementations of EGO algorithms. To generate a Kriging model, the corresponding object of a Kriging subclass, e.g., “Kriging” and “HierarchicalKriging”, should be instanced by inheriting some necessary attributes from the superclass “GaussianProcess”. Then, the processes including the construction of the Vandermonde matrix, Kriging hyperparameters tunning, updates with respect to the necessary arguments (e.g., correlation matrix, mean term, and process variance in terms of the tunned hyperparameters) of a Kriging model, and prediction at an unknown point can be achieved by calling the corresponding member functions. Evidently, all objects of different Kriging models can reuse the universal attributes and recall the member functions with the duplicated processes through the inheritance from the superclass

"GaussianProcess". Thus, the program is highly reusable. Meanwhile, if users want to develop a new Kriging model based on the VAEO toolbox, a new subclass can be defined, and it should inherit the constructor and necessary member functions from the superclass "GaussianProcess". To tune the Kriging hyperparameters, an optimizer should be called to search the optimal set of the hyperparameters on the MLE function. The VAEO toolbox provides flexible selections for users to choose a suitable optimizer. The tuning process can be easily achieved by containing the object of a selected optimizer into the attribute of the superclass "GaussianProcess". Then, the optimization process can be implemented by calling the member function "optimize( )". After constructing a Kriging model, an optimization process, such as the search for the optimum on a point-infill criterion or surrogate model, usually needs to be further carried out. To achieve it, the objects of the Kriging model constructed, selected optimizer, and objective function should be simultaneously contained into the attributes of the class "SingleOptim". Subsequently, the member function "optimize( )" in the "SingleOptim" can be called to search the optimum on the selected objective function. In the same way, a new optimizer can be developed by users through the inheritance from the superclass "Optimizer". Moreover, the existing optimizers also provide mature templates for a new definition. Thus, the program can be easily extended by users for the developments of new techniques.

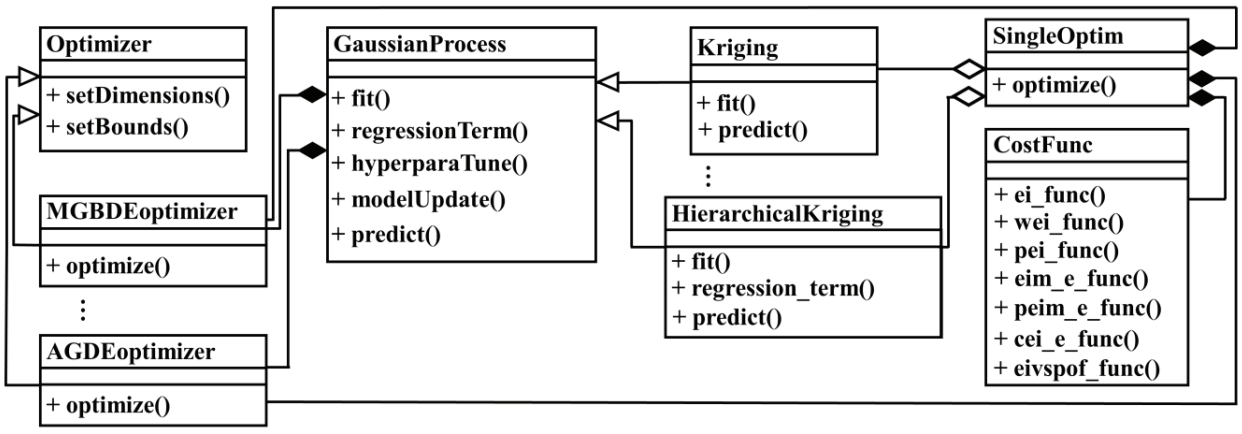


Fig. 3.1 UML class diagram of Kriging metamodeling and optimization in VAEO toolbox

To provide the best solution in programing development, design pattern has been widely used. In the VAEO toolbox, all kinds of design patterns were employed to design different functionalities. As an example, Fig. 3.2 shows the UML class diagram of DOE in VAEO toolbox. Since DOE has different methodologies, the strategy pattern is natural to be

applied for achieving different methodologies. More specially, when the user chooses a concrete methodology of DOE, the corresponding methodology name (string input) should be given. In addition, an object “doe” of “DOE” class needs to be instanced by calling its constructor “DOE( )”. Subsequently, the object “doe” can execute its member function “setStrategy( )” for containing the object of the corresponding methodology through using the string input of the methodology name. When the member function “execution( )” is then executed, the member function “execute( )” of the concrete methodology can be carried out in terms of the object of the corresponding methodology. Apparently, different methodologies can be carried out in a similar manner. Now, if a new methodology of DOE needs to be developed, only a new subclass should be independently defined through inheriting from the abstract superclass “Methodology”. The extendable capability of the VAEO toolbox is further demonstrated.

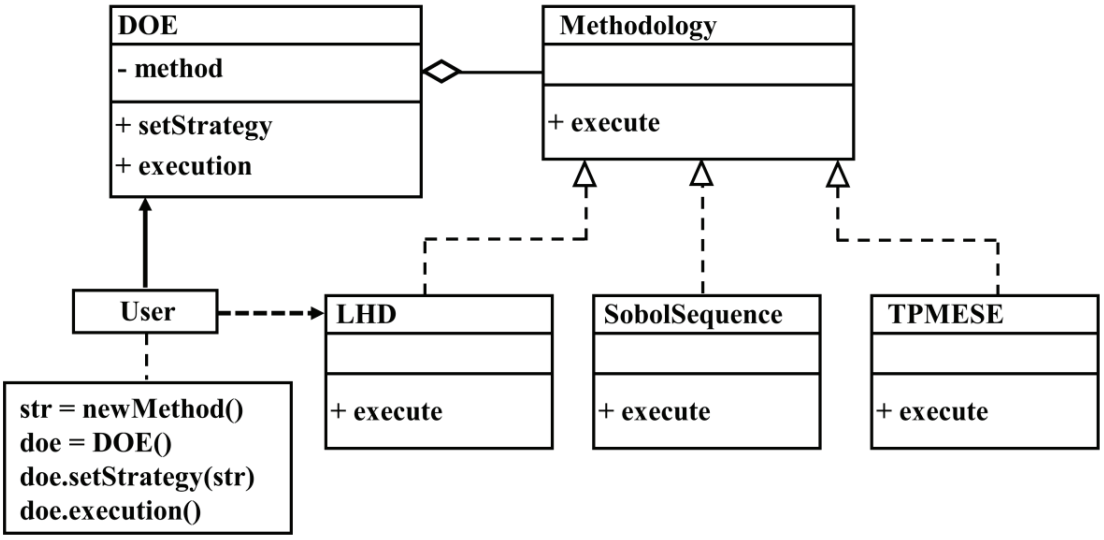


Fig. 3.2 UML class diagram of DOE in VAEO toolbox

3.3 Simple comparisons with other toolboxes

To show the performance of the VAEO toolbox, the VAEO toolbox was used to compare with ooDACE toolbox for Kriging metamodeling and the pySOT for solving a specific global optimization problem. The comparisons were carried out on a computer with 32GB of RAM and eight-core CPUs with a clock speed of 3.6 GHz. The ooDACE and VAEO

toolbox were run in MATLAB R2022b, whereas the pySOT was run in Python 3.9.

3.3.1 Comparison with ooDACE toolbox

To construct a high-quality Kriging model, the Kriging hyperparameters need to be tuned by an optimizer. The VAEO toolbox provides flexible options for users to choose an optimizer. To show the influence of optimizers on tunning the Kriging hyperparameters, the MGBDE, AGDE, and BBPSO-DE optimizers in the VAEO toolbox were chosen to compare with the GA with the highest global searching capability and the SQP with the highest efficiency in the ooDACE toolbox. These optimizers were used to tune the Kriging hyperparameters of the OK model under the different numbers of samples (n = 100, 150, 200, 250). To evaluate the objective values of samples, the two-dimensional Schwefel benchmark function [95] was used in the test. A total of 100 runs were carried out for each optimizer under different n values to tune the Kriging hyperparameters so that the comparison is fair. To evaluate the quality of the OK constructed, the root mean squared error (RMSE) was measured in each run. To terminate the tuning process, the SQP can be stopped when one of stopping conditions, which are 500 maximum iterations and 1e-06 global precision goal, is reached, whereas MGBDE, AGDE, BBPSO-DE, GA all run 2000 function evaluations. To confirm the significant difference of comparative results, the Friedman test [96] with significance level α = 0.05 was used to calculate the mean rankings of difference optimizers over 100 runs under each n value. Table 3.2 lists the results of the Friedman test. Here, a smaller value of the mean ranking indicates the higher performance of an optimizer.

Table 3.2 Comparison of Kriging metamodeling using different optimizers provided by VAEO and ooDACE toolboxes, respectively.

n	p-value		Optimizer				
			MGBDE	BBPSO-DE	AGDE	GA	SQP
100	< 0.0001	Mean ranking	2.38	2.52	2.36	2.74	5
150	< 0.0001	Mean ranking	2.54	2.34	2.42	2.70	5
200	< 0.0001	Mean ranking	2.30	2.48	2.46	2.76	5
250	< 0.0001	Mean ranking	2.20	2.60	2.24	2.96	5

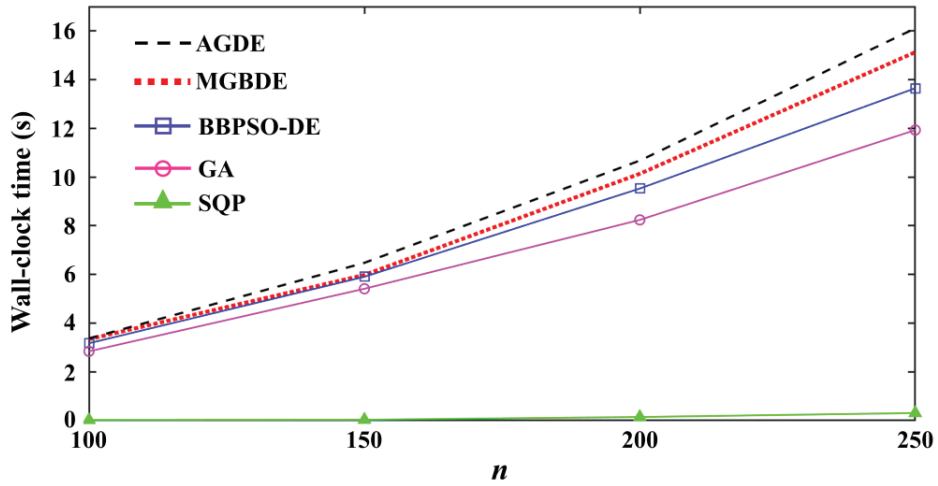


Fig. 3.3 Average wall-clock time costs of optimizers under different n values

According to the results listed in Table 3.2, it can be seen that the three optimizers, including MGBDE, BBPSO-DE, and AGDE, obtain the three smallest mean rankings which are all smaller than the mean rankings of the GA and SQP optimizers in the ooDACE toolbox. Meanwhile, the p-values under all n values are lower than the significance level. Theses reveal that the three optimizers in the VAEO toolbox have higher performance on tunning Kriging hyperparameters than that of the optimizers in the ooDACE toolbox. For the internal comparison of the optimizers in the VAEO toolbox, the MGBDE and AGDE perform better than the BBPSO-DE, except n = 150. In addition, the MGBDE optimizer outperforms the AGDE optimizer when n is relatively large (n = 200 and 250). The lower mean rankings than those of AGDE are obtained by the MGBDE verify this. However, when the n is relatively small (n = 100 and 150), the AGDE shows better robustness than that of the MGBDE to obtain a higher-quality Kriging model. From Fig. 3.3 and Table 3.2, the MGBDE shows the highest balance between the convergence and efficiency in tunning the Kriging hyperparameters. This is because the MGBDE shows a relatively robust performance on tunning Kriging hyperparameters with a relatively low time cost in three heavy strategies in the VAEO toolbox. In comparison, even though the GA and SQP optimizers in the ooDACE toolbox have higher efficiency than that of the optimizers in the VAEO toolbox, the largest two mean rankings obtained by them indicate that a high-quality Kriging is difficult to be constructed using the two optimizers. Evidently, the VAEO toolbox provides more options of optimizers for users to construct the Kriging model with high quality in comparison with the ooDACE toolbox.

3.3.2 Comparison with pySOT toolbox

To show the performance of the VAEO toolbox on global optimization problems. The VAEO toolbox was used to compare with the pySOT in solving a specific global optimization problem. To progress the optimization towards the global convergence, three kinds of EGO algorithms based on the EI, WEI, and PEI criteria (EGO-EI, EGO-WEI, and EGO-PEI) provided by the VAEO toolbox were compared on the six-dimensional Hartmann6 function, whereas two kinds of EGO algorithms based on the EI criterion in a series and asynchronous manners (EGO-EI and EGO-AEI) were adopted as optimization strategies in the pySOT. To make the comparison fair, each strategy ran 50 times starting from 50 groups of OLHDs with 11D-1 initial samples constructed by TPMESE algorithm with 100 iterations. A total of 200 points were added using different strategies. To tune the Kriging hyperparameters and search for the global optimum on the point-infill criteria, the MGBDE optimizers was chosen in the VAEO toolbox. Here, the MGBDE algorithm with 10000 function evaluations was used to search for the global optimum. Additionally, the number of multiple points was set to four for EGO-PEI algorithm. The weight was 0.3 for the WEI criterion. For the pySOT toolbox, the SQP algorithm with a stopping criterion of 500 maximum iterations or 1e-06 precision goal was chosen to tune the Kriging hyperparameters, whereas the GA algorithm with 10000 function evaluations was carried out to search for the global optimum on the point-infill criteria. To confirm the significant difference of comparative results, two non-parametric statistical tests were used: (i) the Wilcoxon signed-rank test with the significance level of $\alpha = 0.05$ [96] was used to confirm the significance difference of the comparative results obtained by two optimization strategies. (ii) the Friedman test with the significance level of $\alpha = 0.05$ [96] was applied to calculate the mean rankings of different optimization strategies. Tables 3.3 and 3.4 show the statistical results. In Table 3.3, R^+ and R^- denote the sum of rankings in which the first optimization strategy performs better and worse than the second one. The larger difference of R^+ from R^- shows the higher discrepancy of the performance between two comparative strategies. Even though the p-value within an interval (0.05, 0.1) is larger than the significance level, it still potentially shows the significant difference between the comparative results. Thus, the results of p-values which are lower than 0.1 are marked in bold in Table 3.3.

From the results listed in Table 3.3, it can be observed that three optimization strategies, the EGO-EI, EGO-WEI, and EGO-PEI, in the VAEO toolbox do not show significant differences between each other. The p-values between two of the strategies in the internal comparison of the VAEO toolbox verify their close performance because the p-values are all significantly higher than the significance level. Similarly, in the internal competition between the EGO-EI and EGO-AEI

provided by the pySOT, they also show close performance in the tested function. The *p*-value which is also higher the significance level between the two strategies confirms the conclusion. The EGO-PEI in the VAEO-toolbox significantly performs better than the EGO-EI and EGO-AEI in the pySOT. The higher values of R^+ than those of R^- obtained by the EGO-PEI and the *p*-values, which are lower than the significance level, confirm the better performance of the EGO-PEI in the tested problem. For the comparison between two EGO-EI algorithms provided by the VAEO toolbox and pySOT, respectively, they do not show significant difference. This is because the *p*-value between them is significantly higher than the significance level. Even though the *p*-values between one of the EGO-WEI and EGO-PEI in the VAEO toolbox, and another one of the EGO-EI and EGO-AEI in the pySOT are slightly higher than the significance level of $\alpha = 0.05$, the values are still lower than $\alpha = 0.1$. This potentially confirms the significant differences between them. Since the values of R^+ obtained by the EGO-PEI and EGO-WEI in the VAEO toolbox are larger than those of R^- obtained by the EGO-EI and EGO-AEI in the pySOT, two strategies in the VAEO toolbox potentially perform better than the EGO-EI and EGO-AEI in the pySOT. Additionally, even though two EGO-EI algorithms in the VAEO toolbox and pySOT, respectively, show close performance, the EGO-EI in the VAEO toolbox still potentially performs better than the EGO-AEI in the pySOT. In the same way, the larger value of R^+ than that of R^- and the *p*-value which is lower than $\alpha = 0.1$ between them confirm the conclusion. The reason why the optimization strategies in the VAEO toolbox generally have better performance than that of the strategies in the pySOT can be demonstrated. On the one hand, the construction of a high-quality Kriging model can be better guaranteed using the MGBDE optimizer; on the other hand, a high-quality solution on the point-infill criteria can also be more easily found by the MGBDE algorithm. Clearly, two critical processes in the EGO algorithm can be carried out with higher quality using the VAEO toolbox. From the results of the Friedman test listed in Table 3.4, it is clear to see that the strategies in the VAEO toolbox obtain the top three ranks, whereas the strategies in the pySOT only obtain the end two ranks. Moreover, the *p*-value of the Friedman test is significantly lower than the significance level, which shows a significant difference of the comparative results. These further demonstrate that the strategies in the VAEO toolbox generally perform better than the strategies in the pySOT. In addition, Fig. 3.4 shows the average convergence curves of different strategies over 50 runs in the VAEO toolbox and pySOT. It can be seen that the EGO-EI, EGO-WEI, and EGO-PEI in the VAEO toolbox converge significantly faster than two strategies of the EGO-EI and EGO-AEI in the pySOT. For the comparison of three strategies in the VAEO toolbox, the EGO-WEI converges to a close-level solution with the fastest speed, whereas the EGO-PEI converge more slowly than other two strategies. For the

comparison of the EGO-EI and EGO-AEI in the pySOT, they both converge to the same level solutions with close speeds. Evidently, the VAEO toolbox can provide more EGO-based strategies than those of the pySOT for more flexibly solving the global optimization problems. Moreover, more optimizers can also be chosen by users so that a high-quality convergence of point infills can be better guaranteed.

Table 3.3 Comparisons of different strategies provided by the VAEO toolbox and pySOT in tested problem using Wilcoxon signed-rank test.

Strategy	p-value	R^+	R^-	$p < 0.05$	$p < 0.1$
EGO-EI in VAEO toolbox versus EGO-PEI in VAEO toolbox	0.3876	727	546	No	No
EGO-EI in VAEO toolbox versus EGO-WEI in VAEO toolbox	0.1463	788	487	No	No
EGO-PEI in VAEO toolbox versus EGO-WEI in VAEO toolbox	0.6397	686	589	No	No
EGO-EI in VAEO toolbox versus EGO-EI in pySOT	0.1190	799	476	No	No
EGO-EI in VAEO toolbox versus EGO-AEI in pySOT	**0.0618**	831	444	No	Yes
EGO-PEI in VAEO toolbox versus EGO-EI in pySOT	**0.0083**	878	347	Yes	Yes
EGO-PEI in VAEO toolbox versus EGO-AEI in pySOT	**0.0041**	935	340	Yes	Yes
EGO-WEI in VAEO toolbox versus EGO-EI in pySOT	**0.0978**	809	466	No	Yes
EGO-WEI in VAEO toolbox versus EGO-AEI in pySOT	**0.0798**	819	456	No	Yes
EGO-EI in pySOT versus EGO-AEI in pySOT	0.5921	693	582	No	No

Table 3.4 Comparisons of different strategies provided by the VAEO toolbox and pySOT in tested problem using Friedman test.

Strategy	Rank	Mean ranking	p-value
EGO-EI in VAEO toolbox	1	2.36	< 0.0001
EGO-PEI in VAEO toolbox	2	2.57	
EGO-WEI in VAEO toolbox	3	2.80	
EGO-EI in pySOT	4	3.65	
EGO-AEI in pySOT	5	3.62	

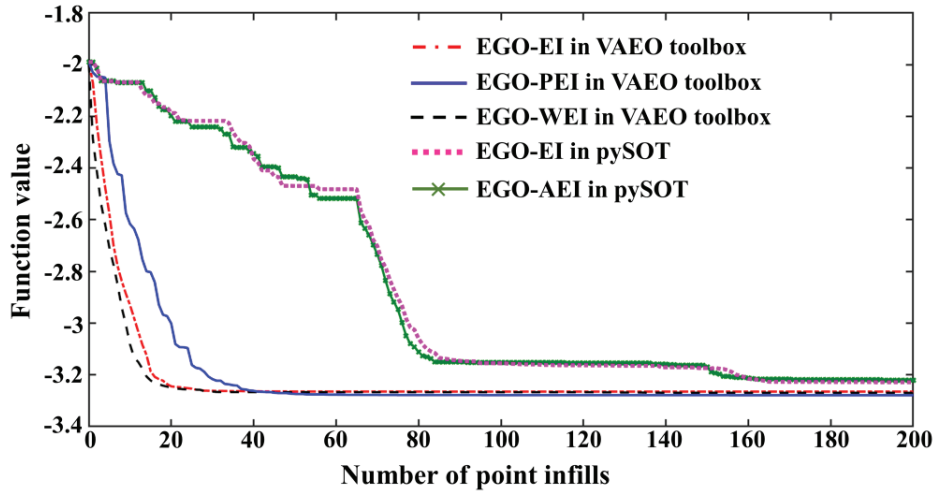


Fig. 3.4 Average convergence curves of different point-infill strategies in VAEO toolbox and pySOT, respectively.

3.4 Conclusion

To implement SBO, a MATLAB toolbox, namely, VAEO toolbox, was programmed using OOP. The VAEO toolbox provides all kinds of functionalities, including various methodologies of DOE, different techniques for Kriging metamodeling, numerous optimization algorithms (optimizers), flexible optimization strategies, SA, and other utilities, for flexibly solving all kinds of complicated optimization problems based on SBO. To make program highly extendable and reusable, the OOP was applied to design the VAEO toolbox. Based on it, a new technique, such as a new methodology of DOE, Kriging model, optimizer, and optimization strategy, can be easily developed. In addition to showing the performance of the VAEO toolbox, the VAEO toolbox was used to compare with the ooDACE toolbox for Kriging metamodeling and the pySOT for solving a specific global optimization problem. On the one hand, the results of the comparisons show that the VAEO toolbox with more powerful optimizers better guarantee the construction of a Kriging model with high quality compared with the ooDACE toolbox; on the other hand, the point infill strategies with the MGBDE optimizer in the VAEO toolbox converge better with faster speeds than those of strategies in the pySOT. Therefore, the VAEO toolbox provides not only the highly extendable program for users to develop their new techniques, but also more flexible and robust options for users to solve global optimization problems.

Chapter 4

Construction of optimization system

In this chapter, the performance of proposed algorithms, namely, TPMESE and MESE, are first investigated in Subsection 4.1. Then, the rationality of the combination between the RKri surrogate model and the EGO-PEI algorithm is discussed in Subsection 4.2. To select a suitable optimizer for the proposed RKri-EGO-PEI system, the seven optimizers are tested on the benchmark functions from five to ten dimensions using the OK-based EGO-PEI algorithm first. Subsequently, the best optimizer in the previous tests is further tested using the RKri-based EGO-PEI algorithm. Then, the mechanism with respect to the influence of an optimizer on the EGO-PEI algorithm is analyzed in Subsection 4.3.

4.1 Performance of proposed algorithms on efficient constructions of OLHDs

To construct a surrogate model with high quality using as fewer discrete samples as possible, an OLHD with high space-filling quality can significantly ensure this goal. Moreover, two key points should be further noticed. First, the optimization is considerably expensive to search for the real optimal design with the highest space-filling quality, particularly for high dimensional problems. Second, the improvement of space-filling quality is significantly small in the later stage of the optimization. Because such small improvement of space-filling quality is hardly helpful to the surrogate modeling, it is usually invaluable to be obtained. Under this situation, the convergence in optimizing an OLHD to a near-global optimum with high efficiency meets the real-life need in SBO better. To efficiently construct OLHDs with high space-filling quality, the TPMESE and MESE were proposed. Subsequently, their performance was compared against each of their original algorithms, namely, Translational propagation ESE (TPESE) and ESE, and other state-of-the-art heuristic algorithms, including PermGA, ILS, and LSGA. The comparison aims to confirming the efficiency of the proposed algorithms. The performance of the different algorithms was tested using three classifications of LHDs. To make comparison fair, all algorithms repeatedly carry out optimizations of LHDs 100 times. The TPMESE and TPESE algorithms started from TPLHDs, whereas other algorithms began the optimizations from random LHDs. The three classifications of LHD sizes contain two small (30 × 3 and 40 × 4), one medium (50 × 5), and two large sizes (60 × 6

and 100 × 10). The φ_p criterion was used to evaluate the space-filling quality of OLHD. To set the parameters of the proposed algorithms, the TPMESE and MESE algorithms were employed to construct OLHDs with sizes of 30 × 3, 50 × 5, and 100 × 10. The optimization using each combination of parameters on each size of LHD was carried out ten times. Moreover, it should be clarified that our current focus is not to search for the optimal set of parameters. Thus, the parameters, including β_1, β_2, n_1, n_2, and α, were roughly set to [0.1, 0.2], [0.1, 0.2], [2, 2.5, 3, 3.5, 4], [0.125, 0.25, 0.5], and [0.8, 0.85, 0.9, 0.95], respectively. Each simulation was evaluated 0.5 million evaluations of φ_p criterion. Then, the best combination of parameters was chosen from above data vectors. As explained in Subsection 2.1.4, the suitable combination of parameters for proposed algorithms were already described, except the tests on the 100 × 10 LHD. In the tests of 100 × 10 LHD, the β_2, n_1, n_2, and α were 0.2, 2.5, 0.5, 0.95, respectively. Moreover, the initial "temperature" was set to 0.005 × φ_p in terms of the initial design, whereas the tolerance for ESE was set to 0.0001 in accordance with reference [43]. Regarding the parameters of other comparative algorithms, the population sizes for small, medium, and large size LHDs were 20 × dimensions of the design, 10 × dimensions of the design, and 10 × dimensions of the design [97], respectively, in the PermGA algorithm. Additionally, the elite size, crossover, and mutation rates were 5, 0.8, and 0.05, respectively [47]. For the LSGA algorithm, the population number P, mutation probability P_m, parameters of P_{max} and P_{min}, and distance ratio c were set as 10, 0.2, 0.3, 0.01, and 0.5, respectively [57]. There were no parameters in the ILS algorithm.

In the tests, the *t*-test was used to confirm the significant difference of comparative results [42]. The significance level of *t*-test was set to 0.025. In other words, when the statistical decision of the *p*-value is lower than the significance level, the null-hypothesis should be rejected. Thus, the comparative results have different means. To make the illustration clear, the symbols "algorithm-algorithm" or "algorithm-algorithm*" were used. The "algorithm-algorithm", such as "TPMESE-MESE", means that the comparison between two algorithms have no significant difference, whereas the symbol "algorithm-algorithm*" shows a significant difference between comparative algorithms. Particularly, the symbol "algorithm-" indicates comparisons between an algorithm and the remaining ones. All tests were carried out on a computer with 32GB RAM and eight-core CPUs with a clock speed of 3.60 GHz using MATLAB R2019b.

To show the performance of the TPMESE and MESE algorithms, Tables 4.1 to 4.3 monitored the mean values and standard deviations of φ_p criterion over 100 runs under the certain numbers of evaluations for φ_p criterion. The best

Table 4.1 Comparison of different heuristic algorithms for optimization of LHDs with small sizes.

Size of LHD	Algorithms	Set1		Set2	
		#Evaluations	Mean (Std)	#Evaluations	Mean (Std)
30×3	MESE	50000	1.9915 (0.0265)	500000	**1.9350 (0.0081)**
	ESE	50000	1.9994 (0.0252)	500000	**1.9353 (0.0081)**
	TPMESE	50000	**1.9834 (0.0211)**	500000	**1.9347 (0.0080)**
	TPESE	50000	1.9912 (0.0239)	500000	**1.9342 (0.0097)**
	ILS	50000	2.0067 (0.0175)	500000	1.9675 (0.0163)
	LSGA	50000	2.0488 (0.0294)	500000	2.0216 (0.0250)
	PermGA	50000	2.1298 (0.0316)	500000	2.0349 (0.0313)
40×4	MESE	120000	1.3519 (0.0095)	1000000	1.3174 (0.0065)
	ESE	120000	1.3585 (0.0110)	1000000	1.3212 (0.0099)
	TPMESE	120000	**1.3474 (0.0076)**	1000000	**1.3153 (0.0063)**
	TPESE	120000	1.3516 (0.0080)	1000000	1.3187 (0.0078)
	ILS	120000	1.3708 (0.0089)	1000000	1.3522 (0.0079)
	LSGA	120000	1.3934 (0.0150)	1000000	1.3810 (0.0147)
	PermGA	120000	1.4496 (0.0135)	1000000	1.3868 (0.0148)

results of mean value and standard deviation, which show the significant difference from others, were marked in bold. It can be seen that the proposed algorithms, TPMESE and MESE, and each of their original algorithms, TPESE and ESE, generally perform better than other comparative algorithms for most of the tested cases except the largest size case of 100 × 10. Two points can support the conclusion. Not only are the smaller mean values of φ_p criterion obtained by the TPMESE, MESE, and each of their original algorithms, but the *p*-values of "one of the TPMESE, MESE, TPESE, and ESE-one of the ILS, LSGA, and PermGA*" show significant difference between the comparisons, except for the tested results of 100 × 10 LHD. The PermGA shows the worst performance in comparison with others. The largest mean values of φ_p criterion over 100 runs and the *p*-values of "PermGA-*" validate this. The ILS algorithm has the closest performance with the proposed algorithms and each of their original algorithms. However, the *p*-values of "ILS-one of TPMESE, MESE, TPESE, and ESE*" still show the significant differences between them, except the largest-size case. For the 100 × 10 tested case, the ILS converges faster than the ESE, but is still worse than the TPMESE and MESE algorithms, at the first monitored evaluation number of φ_p criterion. With the increase of evaluations of φ_p criterion, the

leading position of the ILS algorithm is fast reversed by the ESE algorithm based on the results at the second monitored evaluation number of φ_p criterion. The p-value of "ESE-ILS*" and the mean value of φ_p criterion obtained by the ILS, which changes from a smaller to larger values than that of the ESE algorithm, demonstrate this. Evidently, the ILS, LSGA, and PermGA are significantly time-consuming in comparison with the ESE and its variants, when a sufficient near-optimal or the global optimal design needs to be searched.

For the comparison between the TPMESE, MESE and each of their original algorithms, the TPMESE and MESE generally converge faster than each of their original algorithms and both perform better than the ESE algorithm in the entire measurement interval, except the smallest size case. The smaller mean values obtained by the TPMESE and MESE algorithms, respectively, and the p-values of "TPMESE-TPESE*", "MESE-ESE*", and "TPMESE-ESE*" verify the conclusion. For the smallest tested case, the TPMESE and MESE algorithms only have better performance than each of their original algorithms at the beginning of the optimization for LHD. This is because the p-values of "TPMESE-TPESE*", "MESE-ESE*", and "TPMESE-ESE*" are only obtained at the first monitored evaluation number of φ_p criterion. Moreover, the starting point from the TPLHD can significantly accelerate the convergences of the MESE and ESE algorithms at the beginning of the optimization. From Tables 4.1 to 4.3, it can be observed that the mean values of φ_p criterion obtained by the TPMESE and TPESE algorithms are smaller than those obtained by the MESE and ESE algorithms, respectively, over all tested cases at the first monitored evaluation number of φ_p criterion. In addition, the p-values of "TPMESE-MESE*" and "TPESE-ESE*" confirm the significant differences. The two points verify the acceleration from the TPLHD as an initial design. However, its acceleration becomes inconspicuous with an increase in the number of evaluations of φ_p criterion, except the 40 × 4 and 100 × 10 tested cases. The p-values of "TPMESE-MESE" and "TPESE-ESE" for the 30 × 3, 50 × 5, and 60 × 6 tested cases confirm the close performance between the TPMESE versus MESE and TPESE versus ESE at the second monitored evaluation number of φ_p criterion. For the 100 × 10 tested case, the TPLHD significantly influences the entire measurement interval because the p-values of "TPMESE-MESE*" and "TPESE-ESE*" consistently confirm the significant differences between the TPMESE versus MESE and TPESE versus ESE at two monitored evaluation numbers of φ_p criterion. The reason of the acceleration from the TPLHD can be derived. First, the heuristic algorithms can directly optimize a design with a near high space-filling quality when the initial design is constructed using the TPLHD method almost without time costs. Second, the high diversity of optimization in

Table 4.2 Comparison of different heuristic algorithms for optimization of LHDs with medium size.

50×5	MESE	120000	1.0244 (0.0066)	2000000	**0.9871 (0.0036)**
	ESE	120000	1.0311(0.0067)	2000000	0.9912 (0.0049)
	TPMESE	120000	**1.0209 (0.0059)**	2000000	**0.9881 (0.0043)**
	TPESE	120000	1.0276 (0.0061)	2000000	0.9923 (0.0055)
	ILS	120000	1.0378 (0.0083)	2000000	1.0176 (0.0056)
	LSGA	120000	1.0538 (0.0079)	2000000	1.0387 (0.0081)
	PermGA	120000	1.1061 (0.0090)	2000000	1.0423 (0.0091)

Table 4.3 Comparison of different heuristic algorithms for optimization of LHDs with large sizes.

60×6	MESE	120000	0.8265 (0.0047)	2000000	**0.7936 (0.0029)**
	ESE	120000	0.8284 (0.0049)	2000000	0.7976 (0.0034)
	TPMESE	120000	**0.8240 (0.0044)**	2000000	**0.7931 (0.0031)**
	TPESE	120000	0.8267 (0.0049)	2000000	0.7964 (0.0033)
	ILS	120000	0.8326 (0.0057)	2000000	0.8118 (0.0039)
	LSGA	120000	0.8465 (0.0050)	2000000	0.8222 (0.0054)
	PermGA	120000	0.9088 (0.0056)	2000000	0.8336 (0.0050)
100×10	MESE	1000000	0.4466 (0.0011)	2000000	0.4439 (0.0010)
	ESE	1000000	0.4490 (0.0013)	2000000	0.4450 (0.0009)
	TPMESE	1000000	**0.4459 (0.0008)**	2000000	**0.4435 (0.0008)**
	TPESE	1000000	0.4481 (0.0012)	2000000	0.4446 (0.0010)
	ILS	1000000	0.4484 (0.0010)	2000000	0.4457 (0.0009)
	LSGA	1000000	0.4531 (0.0014)	2000000	0.4494 (0.0012)
	PermGA	1000000	0.5053 (0.0027)	2000000	0.4988 (0.0027)

the initial stage can ensure a fast convergence of the heuristic algorithm. Take 40 × 4 case for example, the optimization can directly start from an initial design whose value of φ_p criterion is 1.6412, whereas the heuristic algorithm must run several thousand evaluations of φ_p criterion to achieve the same-level convergence when the starting point is a random LHD. Thus, the TPLHD as an initial design can significantly accelerate the convergence of optimization for LHD. However, the diversity fast loses along with an increase in the evaluation number of φ_p criterion, particularly for the low-dimensional cases. Thus, a better design is more and more difficult to be searched by the heuristic algorithms in the later stage of optimization. Consequently, the acceleration from the TPLHD as an initial design is no longer apparent. Under

this situation, the heuristic algorithms must spend more time escaping from a local optimal design.

To further illustrate the efficiency of the proposed algorithms, Table 4.4 shows the computational time of different heuristic algorithms to achieve the same convergence level. The convergence level was set to that the value of φ_p criterion was smaller than 104% the final mean value of φ_p criterion over 100 runs when the global convergence was reached. Here, the global convergence was reached, when the TPMESE algorithm ran until no better design was obtained within 1000 generations. The generation means that the number of outer and inner loops completed. To confirm the significant difference, the *t*-test with the significance level of 0.05 was still applied to carry out the statistical analysis for comparative results. The best results which show significant difference from others were marked in bold. Additionally, the thresholds of five, eight, and ten million evaluations of φ_p criterion with small, medium, and large sizes, were set, respectively. If the heuristic algorithms cannot converge to the near-optimal value of φ_p criterion within the thresholds, the optimization can be treated as the divergence. The symbol of "/" indicates that the algorithm cannot search for the near-optimal design over half of 100 runs, as shown in Table 4.4.

From Table 4.4, it can be seen that the proposed algorithm, TPMESE, has the highest efficiency to search for a near-optimal design. The lowest average time costs over all cases and the *p*-values of "TPMESE- *" confirm the conclusion. The PermGA and LSGA significantly lack the global search capability to search for a near-optimal design, particularly for the small and medium cases of LHDs. The symbols of "/" in Table 4.4 verify this. The TPMESE and MESE algorithms generally converge faster than each of their original algorithms. The corresponding lower time costs obtained by the TPMESE and MESE than those of TPESE and ESE, respectively, and the *p*-values of "TPMESE-TPESE*" and "MESE-ESE*" both illustrate the significantly higher efficiency of the TPMESE and MESE than that of each of their original algorithms, respectively.

Table 4.4 Average time costs in optimization of LHDs using different heuristic algorithms.

Size of LHD	Algorithms	MESE	ESE	TPMESE	TPESE	ILS	LSGA	PermGA
30×3	Time/s	7	9	**5**	7	24	/	/
40×4		46	59	**37**	47	762	/	/
50×5		52	89	**44**	78	553	/	/
60×6		68	111	**62**	97	291	627	4285
100×10		547	867	**488**	813	921	1946	/

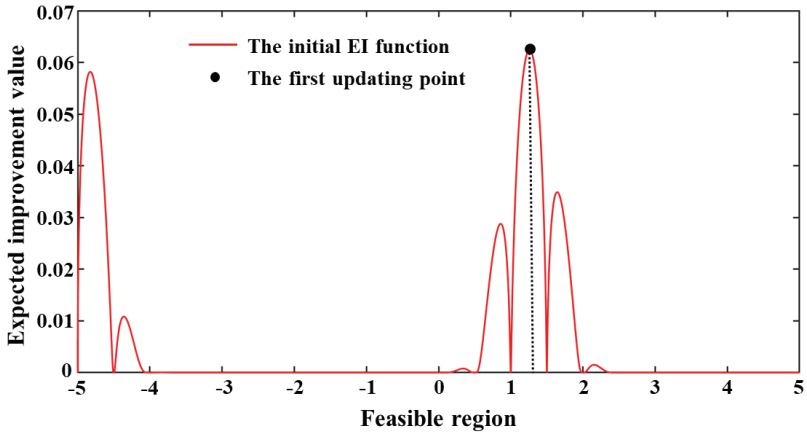
4.2 Rationality of RKri-EGO-PEI

Previously, Forrester et al. [21] employed the RKri and traditional EGO algorithm (RKri-EGO) to minimize the Cd of an airfoil based on a "noisy" CFD simulation. The results showed that the RKri-EGO can robustly filter out noise from the expected smooth response and guide a faster convergence than that of the OK-based EGO algorithm (OK-EGO). To make full use of outer computational resources and further reduce wall-clock time in an optimization process, the combination between the RKri and EGO-PEI is natural to be considered. However, the reason why the EGO-PEI algorithm can work properly is that the PEI criterion substantially provides an approximation with the EI criterion. Therefore, the approximation between the PEI and EI criteria based on the RKri model should be investigated.

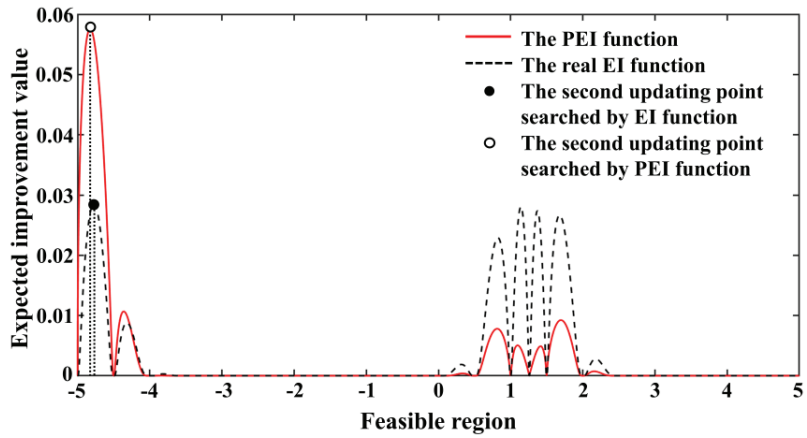
To investigate the rationality of the combination between the RKri and EGO-PEI algorithm, the EGO-PEI and traditional EGO algorithms were used to add updating points based on the RKri. To simulate the expensive deterministic computer experiments, the one-dimensional Griewank function with a feasible region [-5, 5] was adopted. Additionally, a random perturbation from the expected smooth response can be calculated in terms of a normal distribution error ε with the mean $f(\mathbf{x})$ and standard deviation $0.05 \times f(\mathbf{x})$ [98], where the $f(\mathbf{x})$ is the objective value of the Griewank function at the evaluation location $\mathbf{x}$. When the error generated exceeds the bounds $(f_{\min}, f_{\max})$, the error should be reproduced. Here, the $f_{\min}$ and $f_{\max}$ are the minimum and maximum values of objective, respectively. Before the implementation of point infills, the errors were imposed on five samples chosen from 21 initial samples with constant step lengths. Subsequently, the error was added to each updating point. The RKri was used to fit the discrete samples, whereas the traditional EGO and EGO-PEI algorithms were employed to insert the updating points. The number of point infills is four in each cycle for EGO-PEI algorithm.

Fig. 4.1 shows the responses of EI and PEI functions in the process of point infills using the EGO and EGO-PEI algorithms, respectively. It can be seen that the PEI criterion still provides an approximation with the EI criterion using the RKri model. From the first to third iterations, two algorithms provide significantly close predictions of the updating points. Although the different locations for resampling are predicted by the EGO and EGO-PEI algorithms in the fourth iteration based on the different maximum responses of the EI and PEI functions, the EI and PEI functions still show the similar responses. The approximate peaks are clearly exhibited in two functions, which verifies their similarity. Consequently, the location for adding a point searched by the EGO algorithm in the fourth iteration is also the potential location for resampling using the EGO-PEI algorithm. Therefore, an updating point which is significantly close to the

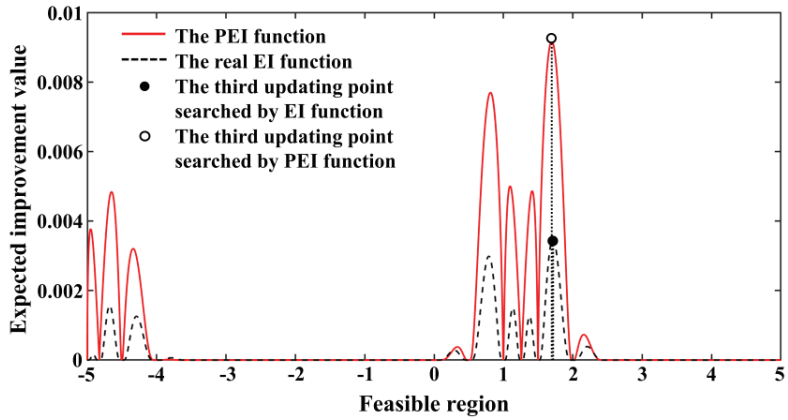
fourth updating point searched by the EGO algorithm in the last iteration is found by the EGO-PEI algorithm at the beginning of the next cycle. Not only that, the similar responses of the EI and PEI functions continue to produce the close predictions for EGO and EGO-PEI algorithms. Thus, more updating points with similar locations are found by two algorithms with the different sequences in the following iterations. In addition, the EI-based criterion only provides the prediction of the location where the maximum probability can obtain any improvement rather than an improvement. This is also the reason why the weighted EI (WEI) criterion can perform better than the EI criterion on certain occasions [89], as it provides a more reasonable balance between global exploration and local exploitation of the point infill criterion. In other words, other locations with a huge response to EI also have potential value for resampling. Therefore, the rationality of the combination between the RKri and EGO-PEI algorithm can be significantly verified because the PEI criterion still provides an approximation with the EI criterion based on the RKri model.



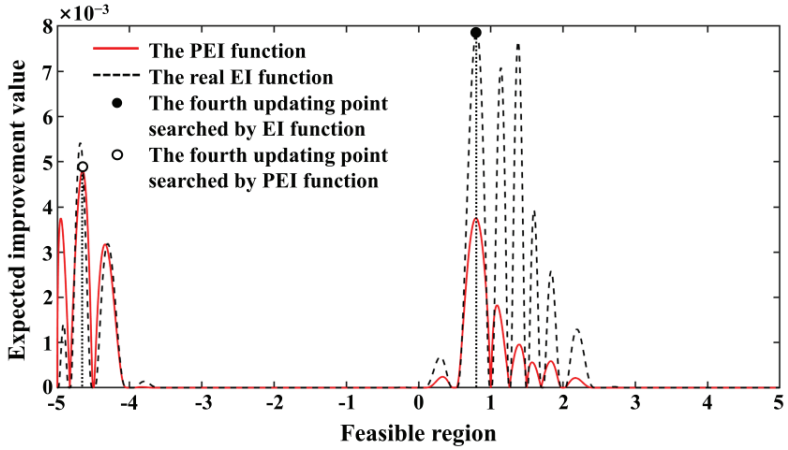
(a) First iteration



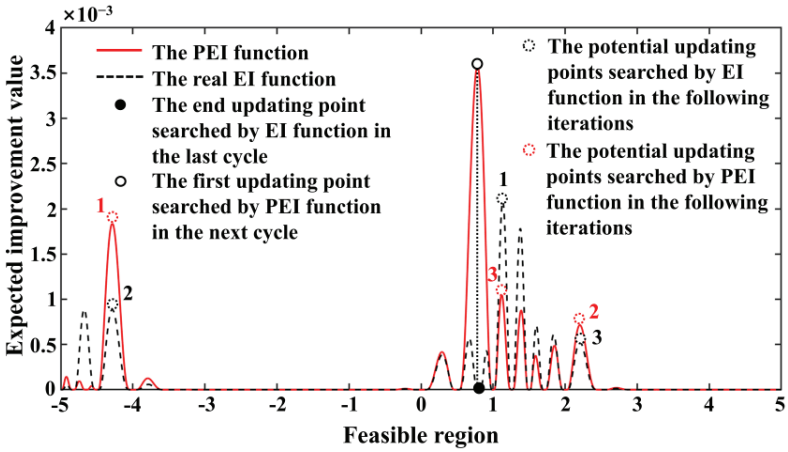
(b) Second iteration



(c) Third iteration



(d) Fourth iteration



(e) First iteration during the next cycle

Fig. 4.1 Comparison with respect to responses of EI and PEI functions during process of point infills based on RKri

4.3 Numerical tests for selection of optimizer

In Subsection 2.4, the importance of an optimizer on the EGO-PEI algorithm was discussed in detail. To select a suitable optimizer for proposed optimization system, RKri-EGO-PEI, and also provide a reference for the selection and development of an optimizer, seven optimizers including DE, RCGA, PSO, BBPSO-DE, MGBDE, AGDE, and SHADE were applied to tune the Kriging hyperparameters and search for the optimum on the PEI function. Additionally, the benchmark functions, $f4$ (5D), $f4$ (10D), $f6$ (5D), $f10$ (5D), $f11$ (5D), $f14$ (5D), and $f22$ (5D), chosen from the CEC 2013 problems [99] were adopted as the substitutes of expensive deterministic computer experiments. Here, D denotes the dimensions of a benchmark function. In the first stage, the OK-based EGO-PEI algorithm (OK-EGO-PEI) was used to add the points. A total of 50 runs were carried out using the EGO-PEI algorithm for each optimizer on each benchmark function so that the comparison is fair. All runs independently started from 50 groups of OLHDs with 11D - 1 samples. These OLHDs were constructed using the TPMESE algorithm with 250 iterations. To provide a reasonable computational cost, the feasible regions of the benchmark functions were rebounded by a smaller hyperspace. The lower and upper bounds of 5D problems are [-50, 0, -60, 30, -50] and [0, 50, -10, 80, 0], respectively, whereas the 10D problem has a lower bound [-50, 0, -60, 30, -50, -70, 20, 0, 30, -50] and an upper bound [0, 50, -10, 80, 0, -20, 70, 50, 80, 0]. To progress the optimization towards the global convergence, 200 points were added in each run [31]. The selection of the number of updating points is based on the following points:

- In real-world optimization problems, the number of point infills is limited by the time and experimental costs. Quickly obtaining a real improvement as large as possible within a short period of time, rather than searching for the real global optimum, much better meets the real-life needs of engineering optimization.
- During the point infill process, the convergence speed of the EGO-PEI algorithm becomes significantly smaller after inserting a certain number of updating points. In other words, if the EGO-PEI algorithm with an optimizer quickly converges to a high-quality solution, it can hold the lead position within a long interval of the following point infills. Considering an acceptable computational cost, the performance of an optimizer to guarantee high-quality convergence of the EGO-PEI algorithm in a relatively short period of time is sufficient for an evaluation within an acceptable number of point infills.

All numerical tests were implemented using the VAEO toolbox in MTALAB 2018b on a computer with 32GB of RAM

and eight-core CPUs (i7 9700K) with a clock speed of 3.6 GHz.

Since the VAEO toolbox keeps the similar style as the ooDACE [83] and DACE [82] toolboxes, the basic settings for the construction of a Kriging model have same meaning. The settings of the OK in the numerical tests are listed as follows:

- Regression function: *repoly*0.
- Correlation function: *correxp*.
- Bounds of Kriging hyperparameters: $[10^{-20}, 10^{20}]$.

Additionally, the parameters of an optimizer usually significantly influence its performance in the search for the global optimum. Fortunately, the parameters of an optimizer are usually strictly tested using the benchmark functions, even real-world problems. Thus, the most robust suite of parameters is normally suggested by authors. In other words, to guarantee that an optimizer works with the best state in the process of point infills, it is necessary to refer to those suggestions and fix the parameters in the numerical tests. Furthermore, to comprehensively investigate the influence of optimizers on the EGO-PEI algorithm, three states of convergence including the incomplete, near-complete, and competed convergences were simulated based on different iterations. To achieve that, each optimizer independently ran 150/250/500 and 250/500/1000 iterations for 5D and 10D problems, respectively. With regard to the specific settings of each optimizer, the details are listed as follows:

- In the DE algorithm, the scale factor was 0.8, whereas the crossover rate was 0.9 [74].
- For the RCGA algorithm, the mutation and crossover rates were set to 0.1 and 0.9, respectively. The tournament selection with a tournament size 2 was adopted in the selection. Additionally, the simulated binary crossover (SBX), non-uniform mutation and elitism were also used [76].
- For the PSO algorithm with a constriction factor, the velocity of the particle can be updated using the following expression:

$$v_{ij}(t+1) = \chi\left[v_{ij}(t) + \mathrm{rand}(0,1) \times \Omega_{1j}(t)\left(p\mathrm{best}_{ij}(t) - x_{ij}(t)\right) + \mathrm{rand}(0,1) \times \Omega_{2j}(t)\left(g\mathrm{best}_{ij}(t) - x_{ij}(t)\right)\right], \tag{4.1}$$

where the $p\mathrm{best}$ and $g\mathrm{best}$ are the personal best solution of a particle in the swarm and the current best solution of the swarm. Here, i and j are ith particle and jth component. The t is the number of iterations. The function rand(0,1) randomly chooses a value from zero to one. With regard to the constriction factor χ, it can be calculated by:

$$\chi = \frac{2\kappa}{|2 - \Omega - \sqrt{\Omega(\Omega - 4)}|} \tag{4.2}$$

In Eq. (4.1), constants Ω_1, Ω_2, and κ satisfy the constraints $\Omega = \Omega_1 + \Omega_2 \geq 4$ and $0 \leq \kappa \leq 1$. A reasonable pair of values for Ω_1, Ω_2, and κ are 2.05, 2.05, and 0.8 [100].

- The BBPSO-DE algorithm had no other parameters.
- The MGBDE algorithm had a scale factor 0.5 for the DE/best/1 mutation strategy [81].
- In the SHADE algorithm, the archive and memory sizes were set to 10 × D, and the *p*best rate was 0.11 [78].
- In the AGDE algorithm, the *p*best rate was 0.1 [79].

To confirm the significant difference of comparative results, the Wilcoxon signed-rank and Friedman tests [96] were used. Here, the Wilcoxon signed-rank test was used to check the significant difference of different optimizers for each tested function, whereas the Friedman test was employed to calculate the mean rankings of different optimizers for all tested functions. Their significance levels are both 0.05. In other words, if the statistical decision *p*-value is lower than the significance level, the null hypothesis should be rejected. Thus, the comparative results show significant difference. By contrast, the null hypothesis which the comparative results have the same mean can be accepted. All *p*-values calculated were accurate to the fourth decimal places. When the *p*-values were lower than 0.0001, the form "<0.0001" was used. In the Wilcoxon signed-rank tests, the R^+ and R^- are the sum of rankings in which the first group of paired samples (comparative results) performs better or worse than the second group. A larger discrepancy between the comparative results can be reflected by the larger difference of the R^+ from R^-. In addition to calculating the sum of rankings, R^+ and R^-, for optimizers over all tested functions, the sums of rankings for each tested function were early calculated. These sums of rankings were then employed to construct the new paired samples over all tested functions. Similarly, to calculate the mean rankings of different optimizers over all tested functions in the Friedman tests, the mean rankings of different optimizers for each tested function were early calculated. Subsequently, these mean rankings were applied to construct the new data matrix for the calculation of mean rankings over all tested functions. Here, a smaller value of mean rankings indicates the better performance of an optimizer for a minimization problem. Correspondingly, a higher rank can be assigned to the optimizer.

Table 4.5 Performance advantages of comparative optimizers

Rank	Global search capability	Convergence speed	Complexity
1	SHADE	PSO	PSO
2	AGDE	MGBDE	DE
3	BBPSO-DE	BBPSO-DE	RCGA
4	MGBDE	AGDE	BBPSO-DE
5	DE	SHADE	MGBDE
6	RCGA	DE	AGDE
7	PSO	RCGA	SHADE

To more clearly analyze the influence with respect to the performance advantages of optimizers on the EGO-PEI algorithm, the performance advantages of seven comparative optimizers was firstly compared. To comprehensively show the performance advantages of the comparative optimizers, the comparisons contained the global search capability, convergence speed, and computational complexity. To make comparison fair, seven comparative optimizers independently ran 50 times on 28 benchmark functions, with five and ten dimensions, chosen form the CEC 2013 benchmark suite [99]. Then, the Friedman test was applied to calculate the mean rankings of the optimizers so that the global search capabilities of the optimizers can be ranked. Additionally, the computational complexity was calculated in terms of the suggestion of the literature [99]. With regard to the comparison of convergence speeds, a relatively subjective results were given based on the main properties in their convergences. Generally, a higher rank indicates a better global search capability, a faster convergence speed, and a smaller complexity. Table 4.5 lists the results in the comparison.

4.3.1 Tests with respect to influence of optimizer on OK-based EGO-PEI algorithm

To clearly show the influence of optimizers on the EGO-PEI algorithm, the results of the EGO-PEI algorithm with seven selected optimizers, under three convergence levels, on the tested functions, after 200 point infills were compared. Table 4.6 lists the comparative results between the MGBDE and other optimizers using the Wilcoxon signed-rank test. In Table 4.6, markers "+" and "-" indicate that the MGBDE optimizer significantly performs better and worse than another comparative optimizer, whereas the marker "=" denotes the comparison between the MGBDE and another optimizer has no significant difference. Additionally, even though the *p*-value which is lower than 0.1 is slightly higher than the significance level (0.05), the relatively small value still can potentially confirm the significant difference of comparative

results. Therefore, as a supplement, the markers "p/<0.1" and "p\<0.1" are additionally listed in Table 4.6 for showing that the MGBDE optimizer potentially has higher or lower performance than that of another one, respectively. To more clearly show the performance of optimizers on the EGO-PEI algorithm, Table 4.7 lists the mean rankings of optimizers under different convergence levels over all tested functions.

Table 4.6 Comparative results between MGBDE and other optimizers under different convergence levels using Wilcoxon signed-rank test.

Convergence	Function	versus BBPSO-DE	versus SHADE	versus AGDE	versus DE	versus RCGA	versus PSO
Incomplete	*f*4 (5D)	=	=	=	=	=	+
	*f*4 (10D)	=	=	=	=	=	=
	*f*6 (5D)	-	+	-	+	+	+
	*f*10 (5D)	=	=	=	+	+	+
	*f*11 (5D)	+	=	+	+	+	+
	*f*14 (5D)	+	+	=	+	+	+
	*f*22 (5D)	=	+	=	+	+	+
Near-complete	*f*4 (5D)	=	=	=	=	=	+
	*f*4 (10D)	=	=(p/<0.1)	=	=	+	=
	*f*6 (5D)	=	=	=	+	+	+
	*f*10 (5D)	=	=	=	=	+	+
	*f*11 (5D)	+	+	+	=	+	+
	*f*14 (5D)	=	=	=	+	+	+
	*f*22 (5D)	= (p/<0.1)	+	=	+	+	+
Completed	*f*4 (5D)	=	=	=	+	+	+
	*f*4 (10D)	=	=	=	+	+	+
	*f*6 (5D)	=	=	=	+	+	+
	*f*10 (5D)	=	=	=	=	+	+
	*f*11 (5D)	= (p/<0.1)	+	+	+	+	+
	*f*14 (5D)	=	=	=	=	+	+
	*f*22 (5D)	=	=	+	+	+	+

Based on the results listed in Tables 4.6 and 4.7, it can observe that the MGBDE performs best in comparison with other optimizers under the near-complete and completed convergences of optimizers. This is because not only are the top

(highest) ranks in the Friedman tests obtained by the MGBDE optimizer in Table 4.7, but all markers are "+" and "=" in Table 4.6. For the comparison under an incomplete convergence of the optimizer, the MGBDE still comprehensively shows the better performance than other optimizers even though the BBPSO-DE and AGDE perform better than the MGBDE on f6. Two points can support the conclusion. First, not only are all markers "+" and "=" except for f6 compared with the AGDE and BBPSO-DE, but the number of markers "+" are more than that of "-" for the comparison between the MGBDE and any other one, except the comparison with AGDE, in Table 4.6. Second, the MGBDE obtains the highest rank in the Friedman test based on Table 4.7. The BBPSO-DE optimizer has the closest performance with the MGBDE optimizer because not only are all second top ranks obtained by the BBPSO-DE optimizer under the near-complete and completed convergences, except for the incomplete convergence of the optimizer in the Friedman test, but there is almost no significant difference between the MGBDE and BBPSO-DE when the optimizer tends to the completed convergence based on the Wilcoxon signed-rank tests in Table 4.6. Two DE variants, SHADE and AGDE, do not show better performance on promoting the convergence of the EGO-PEI algorithm than that using the MGBDE optimizer even though they truly have higher global search capabilities. It is clear to see that the SHADE and AGDE can only guarantee that the EGO-PEI algorithm converges to a same- or worse-level solution in comparison with that using the MGBDE optimizer, except for f6 under an incomplete convergence of the optimizer. The results of the Friedman test in Table 4.7 also confirm that. The SHADE and AGDE just obtain the third and fourth ranks under the near-complete and completed convergences of the optimizer. For three traditional optimizers, DE, RCGA, and PSO, they are all difficult to ensure that the EGO-PEI algorithm converges properly. Clearly, the MGBDE optimizer performs better than them on most of tested functions in terms of the results in Table 4.6. Consequently, three traditional optimizers can only obtain the lower ranks than those of other optimizers in the Friedman test. The influence with respect to the performance advantages of different optimizers can be summarized. The blending optimizers, MGBDE and BBPSO-DE, with the higher central goal of exploitation-exploration, can guarantee a better convergence of the EGO-PEI algorithm within a limited number of point infills in comparison with other optimizers. This is because a more elegant balance of the approximation and difference between the EI and PEI criteria can be obtained using the two optimizers. Three traditional optimizers can only make the EGO-PEI algorithm converge to a low-level solution because they significantly lack the capabilities for tunning the Kriging hyperparameters and searching for a high-quality solution on the PEI function. The SHADE and AGDE have the higher global search capabilities than those of other optimizers. However, the EGO-PEI algorithm with the SHADE and AGDE

just converges better than that with three traditional optimizers but still worse than that with MGBDE and BBPSO-DE optimizers. The reason may be that a high-quality local search from two blending optimizers provides more appropriate randomness, thereby producing an elegant balance of the approximation and difference between the EI and PEI criteria. Such balance brings more diversities to the EGO-PEI algorithm and then promotes a better convergence.

Table 4.7 Mean rankings of different optimizers under incomplete, near-complete and completed convergences of optimizer over all tested functions.

Convergence	Rank	Optimizer	Mean ranking	*p* value
Incomplete	1	MGBDE	1.8571	< 0.0001
	2	AGDE	2.5714	
	3	BBPSO-DE	2.7143	
	4	SHADE	3.5714	
	5	RCGA	5.1429	
	6	PSO	5.7143	
	7	DE	6.4286	
Near-complete	1	MGBDE	2.0714	0.0005
	2	BBPSO-DE	3.0000	
	3	AGDE	3.0714	
	4	SHADE	3.7143	
	5	DE	3.7143	
	6	RCGA	5.8571	
	7	PSO	6.5714	
Completed	1	MGBDE	1.8571	< 0.0001
	2	BBPSO-DE	1.9286	
	3	SHADE	3.3571	
	4	AGDE	3.4286	
	5	DE	4.4286	
	6	RCGA	6.1429	
	7	PSO	6.8571	

To make the influence with respect to the performance advantages of optimizers on the EGO-PEI algorithm clearer, the Wilcoxon signed-rank test was used to calculate the sum of rankings R^+ and R^- in the comparisons with respect to the

Table 4.8 Comparisons with respect to different convergence states of MGBDE and DE using the Wilcoxon sign-rank test, respectively.

Optimizer	Comparison	R^+	R^-	*p* value
MGBDE	Incomplete versus near-complete convergences	3	25	0.0630
	Incomplete versus completed convergences	0	28	0.0180
	Near-complete versus completed convergences	4.5	23.5	0.1077
DE	Incomplete versus near-complete convergences	0	28	0.0180
	Incomplete versus completed convergences	1	27	0.0280
	Near-complete versus completed convergences	21	7	0.2367

different convergence levels of the optimizer using the MEGDE and DE over all tested functions, respectively. Table 4.8 lists the results. From Table 4.8, it can be clear to see that the convergence level of an optimizer significantly influences the performance of the EGO-PEI algorithm. Generally, when the optimizer tends to a completed convergence, the EGO-PEI algorithm gains higher-quality convergence. The conclusion can be verified by the following points. For the results of the MGBDE optimizer, not only are all the values of R^+ smaller than those of R^-, but the *p*-values, 0.0630 and 0.0180, in the comparisons of incomplete versus near-complete convergences and incomplete versus completed convergences are lower than the significance level. These indicate that the convergence of the EGO-PEI algorithm is significantly improved when the completed convergence of the MGBDE optimizer tends to reaching. A close regularity can also be found on the DE optimizer. Since the *p*-values, 0.0180 and 0.0280, in the comparisons of incomplete versus near-complete convergences and incomplete versus completed convergences are lower than the significance level, the significant differences of two comparisons can be confirmed. Moreover, the values of R^+ are also smaller than those of R^- in the comparisons of incomplete versus near-complete convergences and incomplete versus completed convergences. This further demonstrates the EGO-PEI algorithm converges better when convergence state of the optimizer is significantly improved. However, there is an interesting point. In the comparison between the near-complete and completed

convergences of the optimizer, the value of R^+ is larger than that of R^-, and the difference between them is relatively large. These denote that the EGO-PEI algorithm potentially converges better when the state of the DE optimizer changes from the near-complete to completed convergences. However, it is no doubt that the DE under the completed convergence has a larger chance to search for a better solution on the PEI function than that under the near-completed convergence. Therefore, it is cogent evidence to demonstrate that the performance of the EGO-PEI algorithm can be better guaranteed by a higher central goal of exploitation-exploration of the optimizer. Furthermore, Table 4.8 also denotes that the performance of the EGO-PEI algorithm is not significantly improved when the convergence states of the MGBDE and DE change from near-complete to completed convergences. Therefore, a relatively high-quality convergence of the EGO-PEI algorithm can be achieved when a suitable optimizer with the near-completed convergence is adopted. This can achieve a better balance of reducing real conducting time and obtaining a high-quality convergence.

It can be seen that two blending optimizers, particularly for the MGBDE, with a higher central goal of exploitation-exploration are beneficial to promoting the convergence of the EGO-PEI algorithm. To further explain the advantage of the MGBDE optimizer in the EGO-PEI algorithm, the statistical results of the EGO-PEI algorithm using the Wilcoxon sign-rank test is listed in Table 4.9. The comparison was carried out between the MGBDE optimizer under an incomplete convergence and other optimizers under a completed convergence on each tested function. Additionally, Table 4.10 shows the mean rankings of different optimizers in the comparison over all tested functions. From Table 4.9, it is easy to observe that the MGBDE significantly outperforms better than three traditional optimizers, DE, RCGA, and PSO. This is because not only are all markers "+" and "=", but the number of markers "+" is more than that of "-"compared with the RCGA and PSO optimizers. Regarding the comparison with DE optimizer, the MGBDE still shows better performance than that of the DE on *f*6, *f*11, and *f*22. The advantage of the MGBDE is evident. In comparison with other more advanced optimizers, BBPSO-DE, AGDE, and SHADE, the MGBDE generally has a better performance than those of SHADE and AGDE on f11, but worse than those of BBPSO-DE, SHADE, and AGDE on f6. The reason could be a more reasonable balance with respect to the approximation and difference between the EI and PEI criteria can be provided by the MGBDE under such state of convergence on f11, but the quality of a solution searched by the MGBDE is too poor on f6. Additionally, there are no significant differences on other tested functions, except for f22 with AGDE and f4 (10D) with SHADE. However, the *p*-value of the Friedman test in Table 4.10 is lower than the significance level, which denotes the performances of comparative optimizers have significant difference. Moreover, the MGBDE optimizer obtains the second

Table 4.9 Statistical results of comparisons between MGBDE optimizer under incomplete convergence and other optimizers under completed convergence using the Wilcoxon signed-rank test.

Function	versus BBPSO-DE	versus SHADE	versus AGDE	versus DE	versus RCGA	versus PSO
f4 (5D)	=	=	=	=	=	+
f4 (10D)	=	-	=	=	=	=
f6 (5D)	-	-	-	+	+	+
f10 (5D)	=	=	=	=	+	+
f11 (5D)	=	+	+	+	+	+
f14 (5D)	=	=	=	=	+	+
f22 (5D)	=	=	+	+	+	+

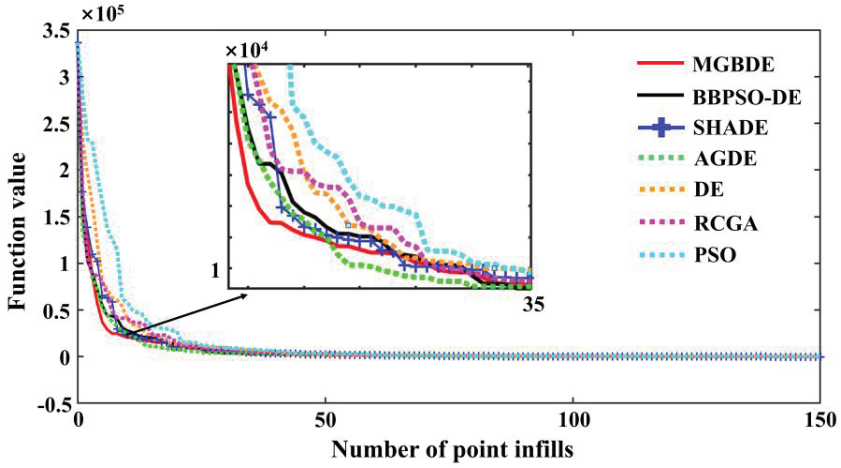
Table 4.10 Mean rankings of MGBDE under incomplete convergence and other optimizers with completed convergence over all tested function.

Ranking	Optimizer	Mean ranking	*p* value
1	BBPSO-DE	1.7143	< 0.0001
2	MGBDE	2.6429	
3	SHADE	3.0714	
4	AGDE	3.2143	
5	DE	4.5000	
6	RCGA	6.1429	
7	PSO	6.7143	

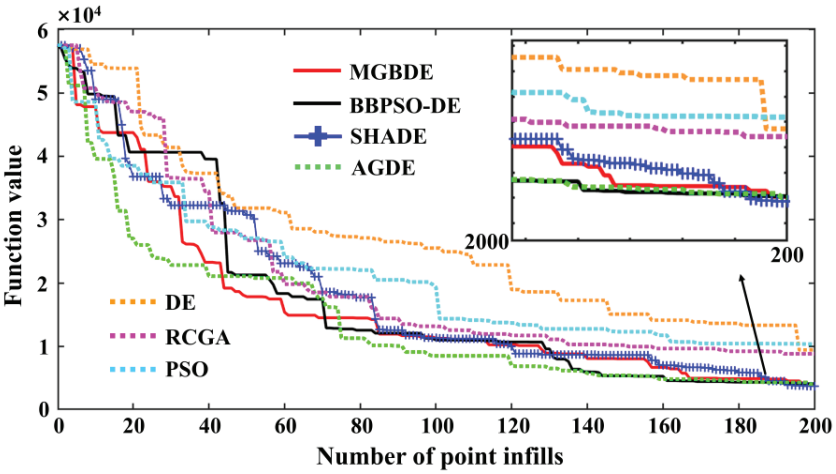
top rank which is only lower than that of another blending optimizer of BBPSO-DE, but higher than other comparative optimizers. The performance of the MGBDE is further demonstrated. First, the MGBDE optimizer can guarantee that the EGO-PEI algorithm converges to the same-level solution as those using the SHADE, AGDE, DE, RCGA, and PSO optimizers, with significantly fewer iterations. Though the MGBDE has a higher computational complexity than those of PSO, DE, RCGA, and PSO algorithms, the significantly fewer iterations still can save the real conducting time in the optimization process. Apparently, the efficiency of the MGBDE can be further improved in comparison with the AGDE and SHADE. Moreover, the MGBDE under the incomplete convergence has no chance to search for a better solution than those of the SHADE and AGDE optimizers under the completed convergence, but it still promises a high-quality

convergence of the EGO-PEI algorithm. Therefore, the importance with respect to an elegant balance between the global exploration and local exploitation provided by an optimizer for the EGO-PEI algorithm is demonstrated again.

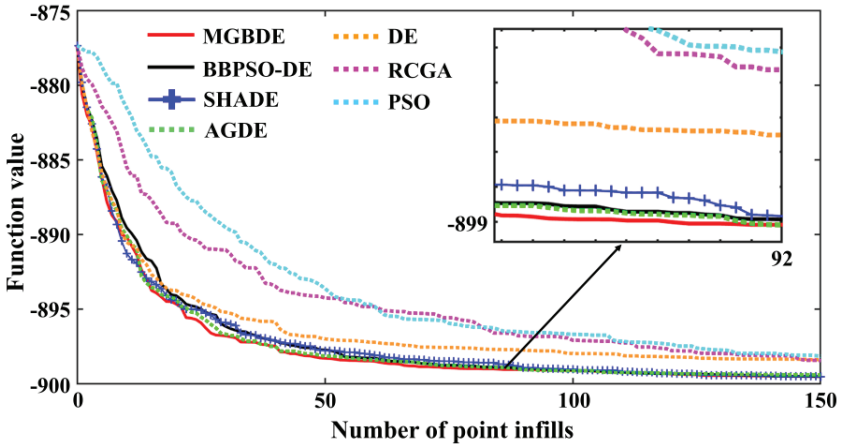
The influence of the optimizers on the convergence level of the EGO-PEI algorithm is clearly demonstrated in previous content. However, it is worth to further investigate the convergence process of the EGO-PEI algorithm using different optimizers. Thus, Fig. 4.2 shows the average convergence curves of the EGO-PEI algorithm with different optimizers under the completed convergence on tested functions over 50 runs. Since the variations of convergence curves on 5D problems are not obvious in end 50 point infills, only the convergence curves within top 150 point infills are shown in Fig. 4.2. With regard to the visualization of 10D problem, the whole process of point infills is visualized. From Fig. 4.2. it can be seen that the more advanced optimizers, MGBDE, BBPSO-DE, SHADE, and AGDE generally promote a faster convergence of the EGO-PEI algorithm than that of the EGO-PEI algorithm using three traditional optimizers, DE, RCGA, and PSO over all tested functions. Clearly, their convergence curves are usually lower than those of DE, RCGA, and PSO optimizers in most of monitoring intervals, which adequately verifies the conclusion. For the comparison of the MGBDE, BBPSO-DE, SHADE, and AGDE optimizers, the MGBDE generally guarantees that the EGO-PEI algorithm converges faster than that with the BBPSO-DE, SHADE, and AGDE on f4 and f6, but the AGDE and BBPSO-DE make the EGO-PEI algorithm gain a larger acceleration than that of the EGO-PEI algorithm with MGBDE and SHADE on f11, f14, and f22. However, a better balance between the convergence speed and diversity of the EGO-PEI algorithm can be obtained when the MGBDE optimizer is adopted. Though the EGO-PEI algorithm with MGBDE optimizer converges more slowly than that with the AGDE and BBPSO-DE optimizers on f11, f14, and f22 in the early stage of point infills, it obtains a larger reduction than that of the EGO-PEI algorithm with the AGDE and BBPSO-DE optimizers in the later stage. Evidently, the advantage of the MGBDE optimizer on the EGO-PEI algorithm can be more clearly demonstrated. In summary, the MGBDE optimizer can promote a better convergence of the EGO-PEI algorithm with a faster speed than that of the EGO-PEI algorithm with three traditional optimizers, whereas the MGBDE optimizer can guarantee that the EGO-PEI algorithm converges to the same- or higher-quality solution with a close speed compared with the BBPSO-DE, AGDE, and SHADE within a limited number of point infills.



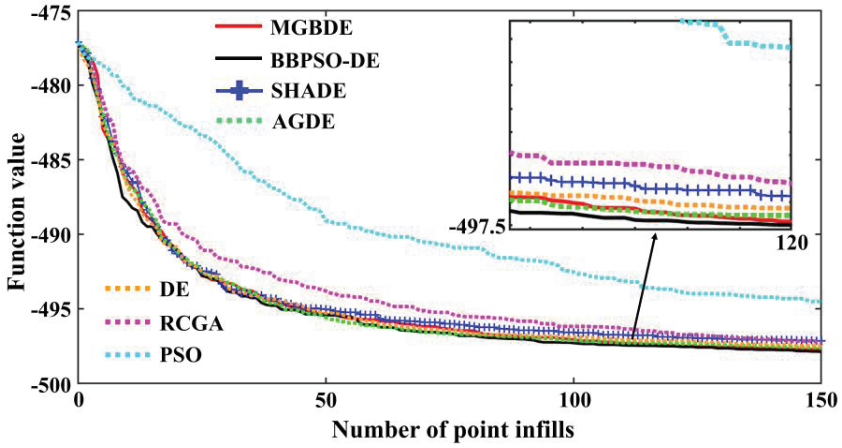
(a) *f*4 with 5D



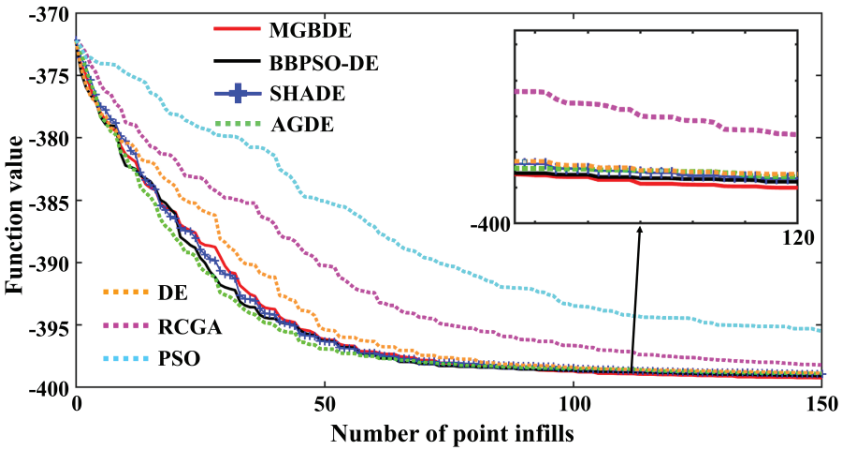
(b) *f*4 with 10D



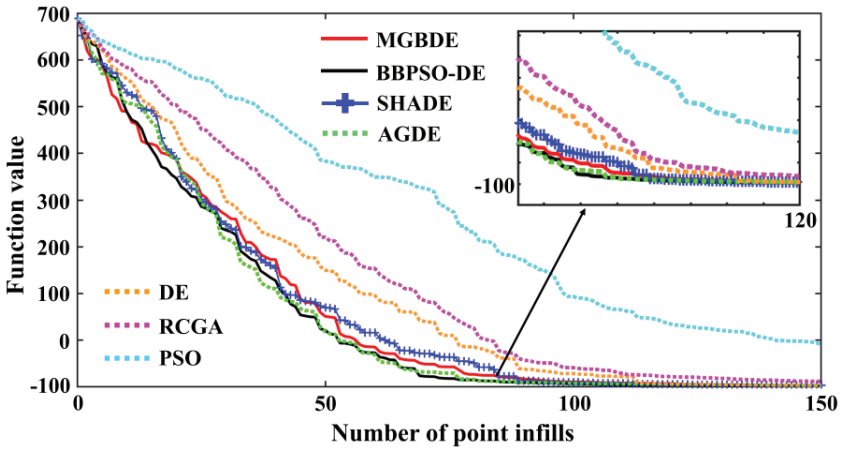
(c) *f*6 with 5D



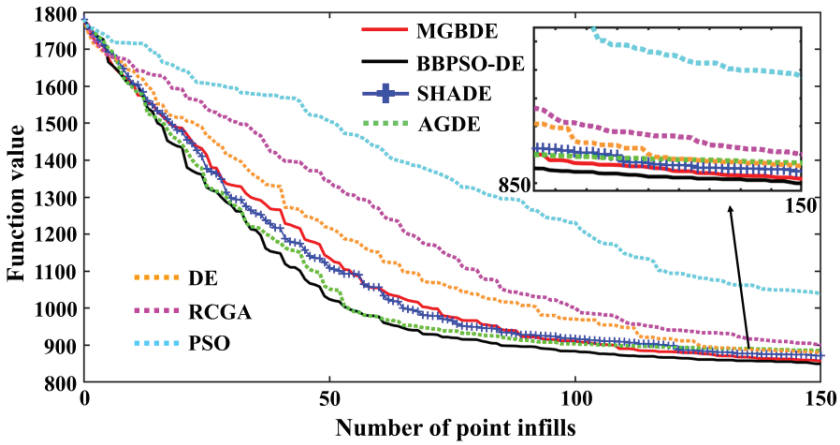
(d) $f10$ with 5D



(e) $f11$ with 5D



(f) $f14$ with 5D



(g) *f*22 with 5D

Fig. 4.2 Average convergence curves of EGO-PEI algorithm with different optimizers under completed convergence on tested functions over 50 runs

4.3.2 Performance of selected optimizer on RKri-EGO-PEI

The performance of the MGBDE optimizer on the OK-based EGO-PEI algorithm was adequately demonstrated in Subsection 4.3.1. It provides an elegant balance with respect to the difference and approximation between the PEI and EI criteria, thereby guaranteeing the EGO-PEI algorithm to converge better than that using other optimizers, within a limited number of point infills. In this sense, it is suitable to choose the MGBDE optimizer in the proposed system of the RKri-EGO-PEI. However, such balance provided by the MGBDE optimizer is probably changed when the Kriging predictor is changed to RKri. Though the rationality of the combination between the RKri and EGO-PEI algorithm was also verified in Subsection 4.2, it cannot ensure that the influence of the MGBDE on the OK-based EGO-PEI algorithm is the same as that on the RKri-based EGO-PEI algorithm. Thus, to verify that the MGBDE optimizer still has an excellent performance on the Rkri-EGO-PEI, the MGBDE optimizer was employed to compare against the SHADE optimizer on *f*4 (5D), *f*6 (5D), *f*11 (5D), and *f*22 (5D) benchmark functions again. Since these functions comprehensively contain the unimodal, multimodal, and composition problems, the performance of the MGBDE on the RKri-EGO-PEI can be adequately analyzed. The reason why the SHADE algorithm rather than the other algorithms was selected as a competitor can be summarized as follows:

- The SHADE optimizer was the top-ranking approach in the Friedman test under the completed convergence of the optimizer, except two blending optimizers, MGBDE and BBPSO-DE, according to the analysis in Subsection 4.3.1.
- Compared with the MGBDE and BBPSO-DE optimizers, SHADE has the highest global search capability for guaranteeing the Kriging hyperparameter tunning and the search for the global optimum on the point infill criterion, and thus the approximation with the maximum level between the EI and PEI criteria can be achieved. Therefore, the influence of a reasonable balance between the approximation and differences in the EI and PEI criteria, rather than only the approximation with the maximum level on the RKri-based EGO-PEI algorithm, can be fully evaluated through a comparison between MGBDE and SHADE. $f_{\min}$

The experimental design is similar to the numerical tests of the OK-based EGO-PEI algorithm in Subsection 4.3.1. The RKri-based EGO-PEI algorithm with the MGBDE and SHADE optimizers independently run 50 times on each benchmark function starting from 50 groups of initial designs with 100 samples constructed by the Sobol sequence [87]. Total of 200 points were added using the RKri-EGO-PEI. To simulate the "noisy" deterministic computer experiments, a random perturbation $\varepsilon = \mathrm{N}(f(\mathbf{x}), 0.1 \times f(\mathbf{x}))$ was imposed on the objective value of each point in the initial samples, whereas a perturbation $\varepsilon = \mathrm{N}(f(\mathbf{x}), 0.01 \times f(\mathbf{x}))$ was added to the objective value of each updating point later. The function $\mathrm{N}(\mu, \sigma)$ was applied to produce a random value based on a normal distribution with a specific mean μ and standard deviation σ. In the same way, when the objective value of a point exceeds the bounds of an interval $(f_{\min}, f_{\max})$, the random perturbation must be reproduced using above expressions. In the tests, four points were be searched in each cycle of point infills in terms of the PEI criteria. All settings of the MGBDE and SHADE were the same as the settings described in Subsection 4.3. According to the results in Subsection 4.3.1, the EGO-PEI algorithm can converge properly when the optimizers tend to a high-level convergence. Thus, the MGBDE and SHADE optimizers both ran 500 iterations to a completed convergence for tunning the Kriging hyperparameters and searching for the optimum on the PEI function.

To confirm the significant difference of the comparative results, the Wilcoxon signed-rank test with a significance level of 0.05 was used. Here, the markers "+" and "-" indicate that the MGBDE optimizer significantly performs better and worse than the SHADE optimizer. The marker "=" denote there is no significant difference between their performance. Meanwhile, the sums of rankings, R^+ and R^-, indicate the MGBDE performs better and worse than the SHADE optimizer in the comparison. Additionally, to more clearly show the influence of the MGBDE optimizer on the RKri-EGO-PEI, the results of the Wilcoxon signed-rank test on the OK-EGO-PEI with the same number of point infills were also shown.

Table 4.11 show the statistical results.

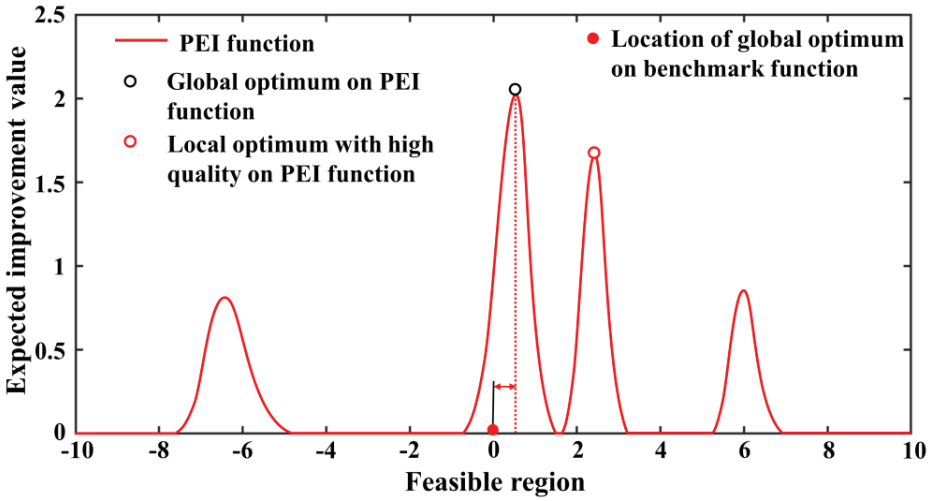
Table 4.11 Wilcoxon signed-rank tests with respect to comparisons of OK-EGO-PEI and RKri-EGO-PEI using MGBDE and SHADE optimizers, respectively, on tested functions.

Optimizer	Kriging model	Function	R^+	R^-	Marker
MGBDE vs SHADE	OK	*f*4	717	558	=
		*f*6	513	762	=
		*f*11	960	315	+
		*f*22	732	543	=
	RKri	*f*4	778	497	=
		*f*6	852	423	+
		*f*11	841	434	+
		*f*22	847	428	+

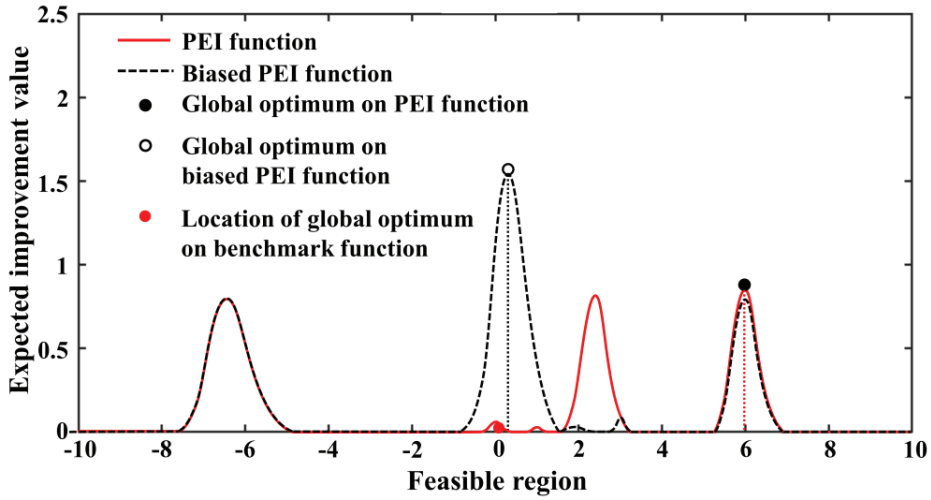
According to the results listed in Table 4.11, the MGBDE optimizer significantly performs better than the SHADE optimizer in the RKri-based EGO-PEI algorithm on *f*6, *f*11, and *f*22. The conclusion is confirmed by the markers "+" on *f*6, *f*11, and *f*22 functions. Therefore, a positive influence of the MGBDE optimizer on the balance between the approximation and difference in the PEI and EI criteria still exists even though the Kriging predictor was changed to the RKri. Moreover, the MGBDE optimizer just outperforms the SHADE on the *f*11 in the OK-EGO-PEI. In other words, the MGBDE seems to provide a better balance between the approximation and difference of the PEI and EI criteria for the RKri-EGO-PEI than that of the OK-EGO-PEI. Since not only are more markers "+" obtained by the MGBDE optimizer in the RKri-EGO-PEI than those in the OK-EGO-PEI, the larger differences between the R^+ and R^- on f4, f6,and f22 in the RKri-EGO-PEI than those in the OK-EGO-PEI also show the larger discrepancies with respect to the performance between the MGBDE and SHADE optimizers. Based on the tested results, it is clear to see that the MGBDE optimizer still performs properly in the RKri-based EGO-PEI algorithm. Thus, the MGBDE is still suitable as an optimizer in the proposed optimization system, RKri-EGO-PEI.

4.3.3 Action mechanism with respect to influence of optimizer on EGO-PEI algorithm

According to the analysis in Subsections 4.3.1 and 4.3.2, it can be seen that the performance of an optimizer significantly influences the convergence quality of the EGO-PEI algorithm. Not only is the performance of the EGO-PEI algorithm relative with the convergence level of an optimizer, it can also be promoted by an elegant balance between the exploitation and exploration of an optimizer. To investigate the mechanism of the influence with respect to an optimizer on the EGO-PEI algorithm, Fig. 4.3 shows a comparison between the PEI and biased PEI functions in two iterations of point infills on a one-dimensional benchmark function. Here, the biased function can be produced in the next iteration when the global optimum in the PEI function is missed in the last iteration. From Fig. 4.3a, it can be observed that an updating point can be searched by an optimizer with a high global search capability, such as the SHADE, at the location with the highest response of the PEI function. However, the updating point still has a relatively large interval with the global optimal solution. Consequently, the EGO-PEI algorithm can only converge to a solution with relatively poor quality in this iteration. However, a high-quality convergence can be accelerated when a high-quality local solution is fast found by an optimizer with a high central goal of exploration-exploitation. Under this situation, a location which is significantly close to the global optimal solution is produced in the biased PEI function. Thus, the EGO-PEI algorithm can fast converge to a high-quality solution in the next iteration. By contrast, the EGO-PEI algorithm must run more iterations for a close or even worse convergence when the real global optimal solution is found in the last iteration. Evidently, a higher central goal of exploration-exploitation in an optimizer can bring more diversities to the EGO-PEI algorithm so that the EGO-PEI algorithm has a larger chance to escape from a poor solution. Moreover, if the EGO-PEI algorithm prematurely converges to a poor local solution in a high dimensional or highly multimodal problem, its convergence speed will be significantly slowed down because the EGO-PEI must take more efforts to explore the design space. In addition to using an over poor optimizer, such as DE, RCGA, or PSO, the search process of the EGO-PEI algorithm is closer to a quasi-stochastic process. Since an optimizer with a poor-quality convergence cannot obtain a high-quality solution in the PEI function, the EGO-PEI algorithm cannot also receive the useful information of a scientific point-infill criterion. Thus, the EGO-PEI algorithm runs unproperly.



(a) Last iteration



(b) Next iteration

Fig. 4.3 Comparison between PEI and biased PEI functions for adding updating points in two iterations

4.4 Conclusion

In this chapter, an optimization system, namely, RKri-EGO-PEI, is constructed to generate initial samples with high space-filling quality, filter out noise, and carry out point infills in a parallel manner.

To fast construct an OLHD with high space-filling quality so that the spatial properties can be extracted using as fewer

discrete samples as possible, the performance of the proposed algorithms, namely, TPMESE and MESE, are compared with other state-of-the-art heuristic algorithms, including the ILS, LSGA, and PermGA, and each of their original algorithms, called TPESE and ESE, on five tested cases with small, medium, and large sizes of LHDs. The performance of the TPMESE and MESE can be summarized as follows:

1) The TPMESE and MESE algorithms and each of their original algorithms show significantly higher efficiency than that of the ILS, LSGA, and PermGA algorithms in the optimizations of LHDs.
2) The TPMESE and MESE algorithms generally converge faster than each of their original algorithms, TPESE and ESE, particularly for relatively large-size cases.
3) The TPLHD can significantly accelerate the optimization process at the beginning, whereas the effect from the TPLHD as an initial design is gradually weaken in the subsequent optimization. However, when the dimension of a design is relatively high, the acceleration can continue for a long period of time. Therefore, the TPMESE algorithm can achieve a sufficient near-optimal design with highest efficiency compared with other completive algorithms, particularly for large-size LHDs.

In the following, the rationality of the combination between the RKri surrogate model and EGO-PEI algorithm is validated. According to the investigation, the PEI criterion still provides an approximation with the EI criterion during the process of point infills. The approximation ensures that the RKri-EGO-PEI works properly based on a similar mechanism to the traditional RKri-EGO.

To select a suitable optimizer for the proposed system, seven optimizers, including the MGBDE, BBPSO-DE, SHADE, AGDE, DE, RCGA, and PSO, are tested on the benchmark functions from five to ten dimensions using the OK-based EGO-PEI algorithm. After the tests, the MGBDE with the best performance on the OK-based EGO-PEI algorithm is chosen to further compare with the SHADE optimizer using the RKri-based EGO-PEI algorithm. Its performance is verified again. Additionally, the mechanism with respect to the influence of optimizer on the EGO-PEI algorithm is analyzed.

1) An optimizer with a higher central goal of exploitation-exploration can give a more element balance of the approximation and difference between the EI and PEI criteria, thereby promoting the better convergence of the EGO-PEI algorithm within a limited number of point infills. Generally, a high global search capability of the optimizer guarantees that the EGO-PEI algorithm can work properly through a similar mechanism to the

traditional EGO algorithm, whereas an excellent local exploitation of the optimizer to search for a high-quality local or near-optimal solution rather than the real optimal solution can bring more diversities to the EGO-PEI algorithm, thereby achieving the convergence with higher quality.

2) The MGBDE optimizer is the winner in the benchmark tests. The MGBDE optimizer can not only ensure that the EGO-PEI algorithm converges to the same level solution as that using other optimizer, with fewer iterations, but also promote a higher-level convergence of the EGO-PEI algorithm than that with other optimizers. Thus, the MGBDE optimizer is chosen as the optimizer in the proposed system.

Chapter 5

Real-world application on aerodynamic shape optimization of a vehicle

In this chapter, the proposed optimization system, namely, RKri-EGO-PEI, is employed to minimize the Cd of a realistic vehicle model based on CFD simulations. Before the optimization, the information of the vehicle model is introduced in Subsection 5.1. Subsequently, the design variables, objective, turbulence model, mesh morphing, mesh scheme, and settings of the CFD solver are introduced in Subsections 5.2, 5.3 and 5.4, respectively. Since the mesh scheme and solver settings significantly influence the accuracy of the CFD simulations, the rationality of the CFD simulation is validated in terms of the analysis of mesh independence and the wind tunnel experiment in Subsection 5.5. To further reduce labor costs in the optimization, an automatic route is generated to carry out the mesh morphing, mesh construction, and CFD simulation in an automatic manner in Subsection 5.6. In Subsection 5.7, the performance of the proposed optimization system on the drag reduction of the vehicle with "noisy" computations is adequately discussed.

5.1 Vehicle model

The DrivAer model [101] was developed by the Technische University Munchen (TUM) in collaboration with the Audi and BMW automotive companies, and it contains three typical rear configurations of vehicles, namely, Fastback, Notchback, and Estateback. Each configuration consists of two underbody types including smooth and detailed underbodies. In comparison with other famous vehicle bodies, such as the Ahmed [102] and Mira bodies [103], the DrivAer model almost maintains all primary characteristics of a realistic vehicle. Thus, the characteristics of the flow field around the vehicle body are worthy of being investigated. To give other engineers and researchers a more valuable reference, the Estateback configuration of DrivAer model was chosen to implement the optimization. In addition, the Estateback model in this study has a smooth underbody, standard mirrors, and non-rotating wheels. The model was scaled down to 20% full model. More detailed sizes can be found in Table 5.1, whereas Fig. 5.1 shows the three rear configurations of the DrivAer model. In Table 5.1, the model scale (MS) was adopted to parameterize other primary sizes [104].

Table 5.1 Primary geometric information of Estateback model.

Parameter	Symbol	Parameterization	Value
Model Scale	MS		0.2
Model length	L_e	4.6126 MS	0.9925 m
Model width	W_e	1.7529 MS	0.3506 m
Model height	H_e	1.4182 MS	0.2836 m
Front projected area	A	2.1605 MS^2	0.0864 m^2

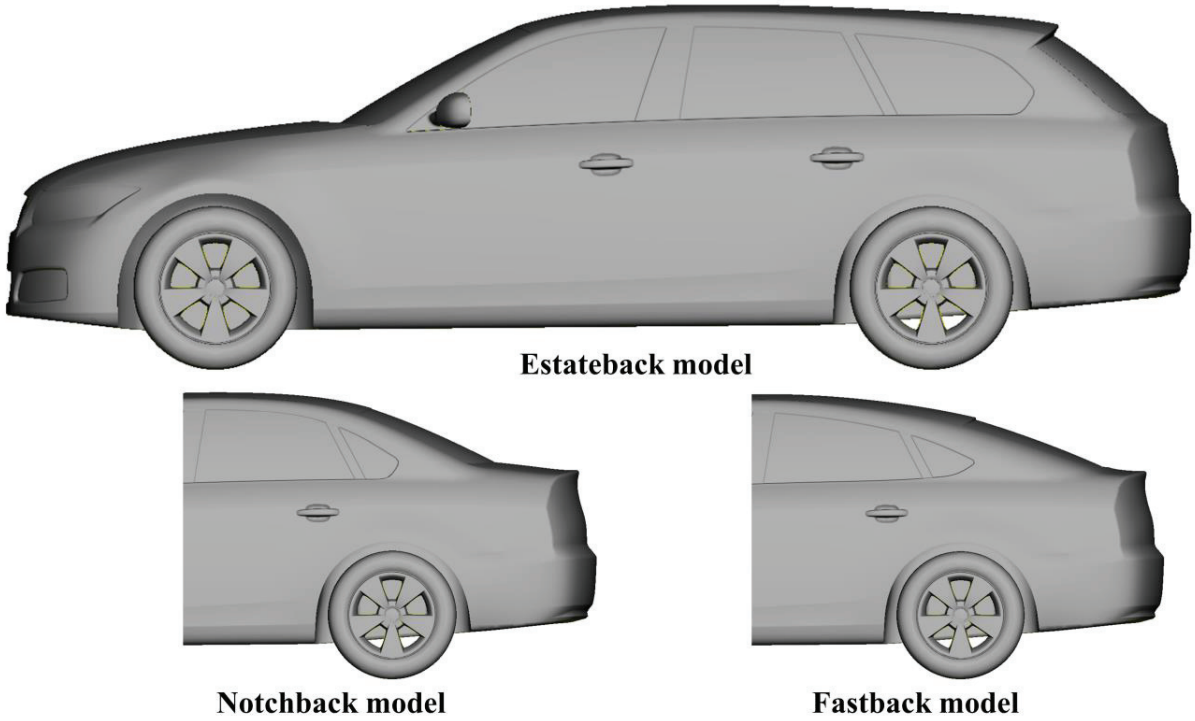


Fig. 5.1 Configurations of DrivAer model

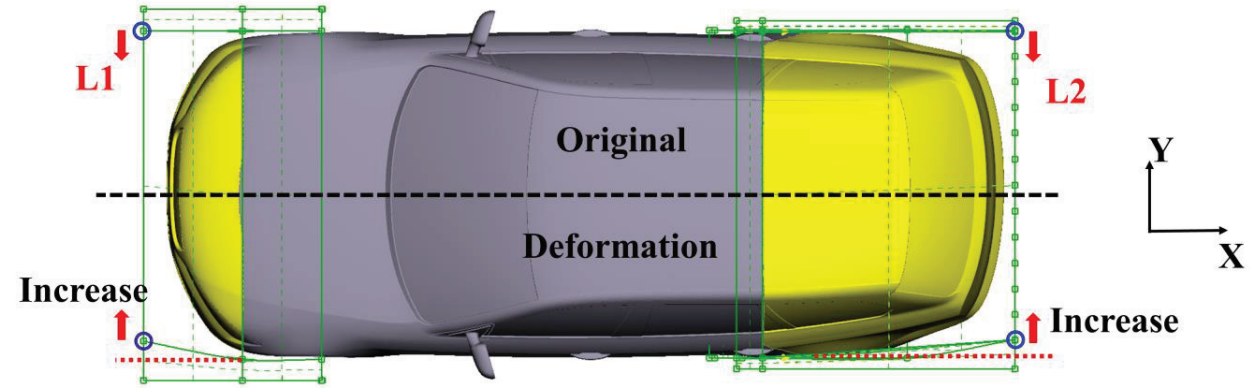
5.2 Objective and variables of optimization

Aerodynamic drag is the primary component of resistances in the movement of a vehicle with high speed [1]. In this study, the Cd minimization was chosen as the objective in the optimization. To provide more valuable reference in the real-world industrial application, the design variables should be carefully defined. In general, the definitions of the design variables are necessary to meet the following requirements. First, the Cd can be reduced as much as possible based on the

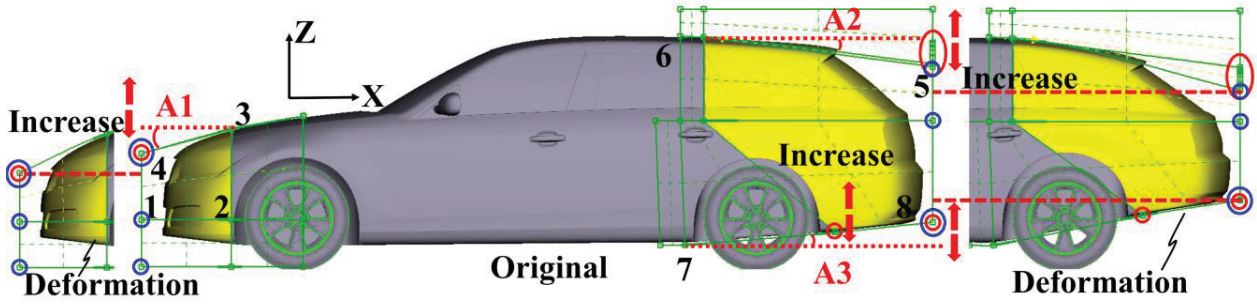
design variables. Second, the computational costs should be reasonable so that the efficiency of the optimization can be maximized. Third, the vehicle configuration cannot be excessively changed when the largest deformation is carried out. To meet the aforementioned requirements, the design variables should have not only huge influences on the Cd, but also the reasonable lower and upper boundaries of the feasible region. According to the reference [105], the characteristics of the wake significantly influence the aerodynamic performance of vehicles. In fact, a steep change of pressure always occurs in the rear of a vehicle, the considerable variation of the pressure distribution on the rear body surface suppresses the pressure recovery of the rear body in comparison with the front body. Consequently, a pressure difference is produced between the front and rear bodies of a vehicle, which leads to a pressure drag. Thus, if the pressure on the rear vehicle body can be recovered better, a grater reduction in drag can be achieved. It is remarkable that a reasonable shape of the rear body is beneficial to reducing the drag [8]. Similarly, since the pressure difference are simultaneously determined by the pressure distributions on the front and rear bodies, the front shape, particularly the stagnant flow aera of the head, also influences the drag. In past decades, many studies have investigated the influences with respect to the different parts of vehicles on the Cd [7, 13, 14]. To achieve a possible maximum improvement under a reasonable computational cost and prevent a difference of deformation model far away from the baseline model, the definitions of the design variables can refer to these references [7, 13, 14]. Accordingly, five design variables for five critical parts including the engine hood, rear spoiler, diffuser, and front and rear sides were chosen in the optimization. The design space surrounded by the lower and upper bounds of the design variables are listed in Table 5.2, whereas the Fig. 5.2 visualizes the design variables and definition of the mesh morphing. Initially, the baseline model has zero values of the design variables.

Table 5.2 Definition of design variables

Deformation part	Variable	Lower bound	Upper bound
Engine hood	A1	- 5°	+ 10°
Rear spoiler	A2	- 3°	+ 7°
diffuser	A3	- 5°	+ 6°
Front side	L1	0 mm	+ 20 mm
Rear side	L2	0 mm	+ 20 mm



(a) Step 1 for deformation on XY plane



(b) Step 2 for deformation on XZ plane

Fig. 5.2 Definitions of design variables and mesh morphing

The morphing technique in ANSA v20 was adopted to implement automatic deformation for the vehicle model. Before the deformation, the morphing regions should be partitioned by the control boxes. Then, the partitioned morphing regions can be activated and deformed along with the movements or deformations with respect to the control points, edges, and surfaces of the control boxes. Fig. 5.2 shows the details of the mesh morphing. The green boxes are the control boxes to define the deformation regions, whereas the red and blue cycles on the control boxes denote the active control points moved in the Y- and Z- directions. In this study, the active parts within the control boxes can be morphed along with the movement of the control points. It can be seen that the vehicle model was generally partitioned into the deformation and stagnation aeras by two control boxes. Additionally, some inner surfaces of the control boxes continue to partition the deformation regions into different control regions. In summary, the regions covered by the yellow color were activated when the control points in blue cycles were moved, whereas the regions covered by the sets of points 1-2-3-4 and 5-6-7-

8 were loaded into the control boxes when the control points in the red cycles were moved. To avoid the deformations of the tyers, the tyers were not loaded into the control boxes. In step 1, the control points in blue cycles translated in the Y-direction, as shown in Fig. 5.2a. Thus, the front and rear sides of the vehicle model, covered by the yellow color, were only changed on the XY plane. In step 2, when the relative control points in red cycles, as shown in Fig. 5.2b, only moved in the Z-direction, the regions covered by the point sets of 1-2-3-4 and 5-6-7-8 were deformed on the XZ plane. Accordingly, the A1, A2, and A3 were changed.

5.3 Turbulence model

The RANS model assumes that the turbulence flow is steady state. Thus, the turbulence velocity can be decomposed into the average and fluctuation components. By substituting the decomposing turbulence velocities into the governing equation of flow, the Navier-Stokes (NS) equation can be rewritten as the Reynolds-average NS equation. Here, the difference between the two equations is only the Reynold stress. In the three-dimensional flow, the Reynold stress can be expressed by a matrix of 3 × 3. The leading diagonal of matrix contains the three terms of the Reynold norm stresses, whereas the other elements in the matrix indicate the rest of terms of the Reynold shear stresses. To close the equation, different methodologies for the approximation of the turbulence viscosity haven been developed [106].

In past decades, the two-equation turbulence models have been widely used to solve the different problems of turbulence flows. The $k-\varepsilon$ model is significantly popular to solve the high Reynolds number flows but lacks enough accuracy for capturing the properties of flows from the near wall layer to separation region. The $k-\omega$ model is more suitable to provide a proper prediction of the flow structure in the boundary layer. However, it fails to solve the problem of pressure induced separation. Moreover, the ω equation is significant sensitivity with the freestream [107] outside the boundary layer. That is the primary reason why the ω equation still cannot substitute the ε equation as the standard scale-equation in turbulence modeling. According to the limitations of the $k-\varepsilon$ and $k-\omega$ models, the $\mathrm{K-omega-SST}$ model was developed to blend their advantages for better solving the problem of pressure induced separation [107].

The $\mathrm{K-omega-SST}$ model can be expressed as:

$$\frac{\partial(\rho k)}{\partial t}+\frac{\partial(\rho U_i k)}{\partial x_i}=\tilde{P}_k-\beta^*\rho k\omega+\frac{\partial}{\partial x_i}\left[(\mu+\sigma_k\mu_t)\frac{\partial k}{\partial x_i}\right] \tag{5.1}$$

$$\frac{\partial(\rho\omega)}{\partial t}+\frac{\partial(\rho U_i\omega)}{\partial x_i}=\alpha\rho S^2-\beta\rho\omega^2+\frac{\partial}{\partial x_i}\left[(\mu+\sigma_\omega\mu_t)\frac{\partial\omega}{\partial x_i}\right]+2(1-F_1)\rho\sigma_{\omega2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial\omega}{\partial x_i}$$

In the Eq. (5.1), the blending function F_1 is defined as follows:

$$\mathrm{F}_1=\tanh\left\{\left\{\min\left[\max\left(\frac{\sqrt{k}}{\beta^*\omega y},\frac{500\nu}{y^2\omega}\right),\frac{4\rho\sigma_{\omega2}k}{CD_{k\omega}y^2}\right]\right\}^4\right\} \tag{5.2}$$

Here, y is the distance to the nearest wall, whereas the CD_{kw} is written as:

$$CD_{k\omega}=\max\left(2\rho\sigma_{\omega2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial\omega}{\partial x_i},10^{-10}\right) \tag{5.3}$$

From Eq. (5.1), it can be seen that the blending function controls the switch between the $k-\varepsilon$ and $k-\omega$ models. When the vertex of mesh cell is far from the wall, the y is large, thus leading to zero value of blending function. Under this situation, the $\mathrm{K-omega-SST}$ is identical to the $k-\varepsilon$ model in the freestream outside the boundary layer. In contrast, the blending function switches the $\mathrm{K-omega-SST}$ to the $k-\omega$ model when the y is close to the wall.

In Eq. (5.1), the turbulent viscosity of μ_t is expressed as:

$$\mu_t=\frac{a_1k}{\max(a_1\omega,S,\mathrm{F}_2)} \tag{5.4}$$

The S and F_2 are the invariant measure of the strain rate and second blending function, respectively, where F_2 is defined as:

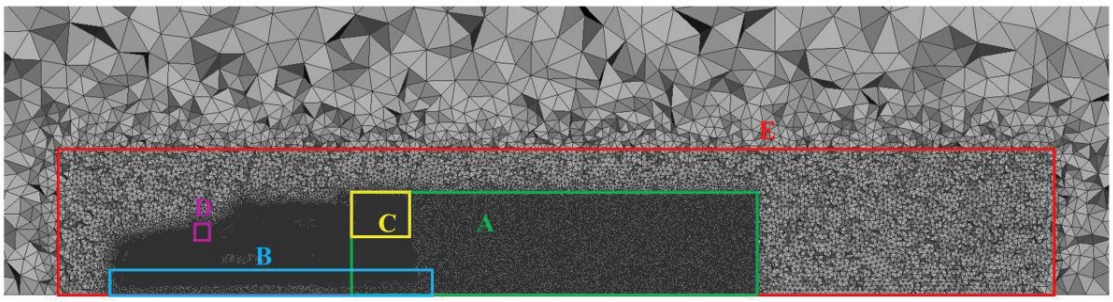
$$\mathrm{F}_2=\tanh\left\{\left[\max\left(\frac{2\sqrt{k}}{\beta^*\omega y},\frac{500\nu}{y^2\omega}\right)\right]^2\right\} \tag{5.5}$$

The constants of β^*, α_1, α_2, β_1, β_2, σ_{k1}, σ_{k2}, $\sigma_{\omega1}$, and $\sigma_{\omega2}$ are 0.09, 5/9, 0.44, 3/40, 0.0828, 0.85, 1, 0.5, 0.856, respectively. All constants are calculated based on a blend from of the $k-\varepsilon$ and $k-\omega$ models, such as $\alpha=\alpha_1F+\alpha_2(1-F)$.

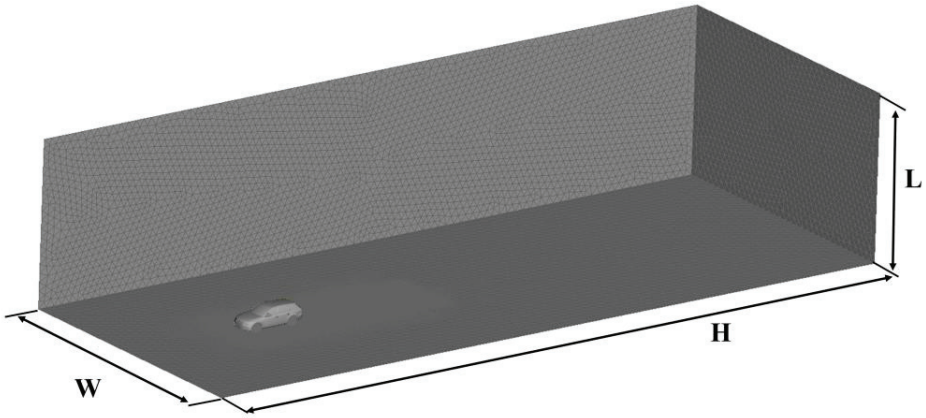
5.4 Mesh scheme and CFD solver

In numerical simulation, the $\mathrm{K-omega-SST}$ turbulence model with the steady state time scheme in OpenFOAM

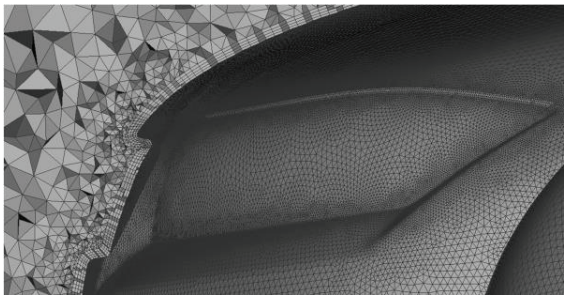
v1706 running on the ITO supercomputer in Kyushu University was employed to solve the flow flied around the vehicle model. A numerical domain with the width W = $11W_e$, length L = $15L_e$, and height H = $7H_e$ was generated, which guarantees that the block ratio was less than 1%. For mesh generation, four prism layers with the first layer thickness of 0.97mm and growth rate of 1.02 were constructed to capture the details of a marginal relative boundary layer. All elements in the first prism layer were adjusted to meet the requirement of y-plus value which is about 30 for allowing the minimum threshold of the log-law section [104]. To make the scheme of mesh generation clear, an expression $s \times h \times$ MS was applied to evaluate the primary sizes of the surface and volume meshes in the numerical domain, where s and h were the constant and threshold, respectively. In this study, h was equal to 10 mm. To provide a high-quality transition from the surface to volume meshes and capture the details of the primary separation zones, such as the wake behind the vehicle rear end, five density boxes (refinement regions) with the different maximum element sizes were utilized to refine the meshes. Fig. 5.3 shows the mesh generation and density boxes. The resolution of the density boxes C, D, and E were set to $h \times$ MS (2mm), $h \times$ MS (2mm), and $6h \times$ MS (12mm), respectively. Additionally, not only do the flow characteristics of the wake significantly influence the pressure distribution on the surfaces of the vehicle rear, but the bottom flow below the underbody also affects the functionality of the diffuser. Thus, the sensitivity of the mesh resolution in density boxes A and B should be investigated [108] before optimization. To achieve this, five resolutions with s = 1.5, 2, 3, 4, and 5 in density boxes A and B were chosen to test the mesh independence. For the surface meshes on the vehicle body, the sizes from $0.25h \times$ MS to $2h \times$ MS (0.5 to 4 mm) were set. All elements in the surface meshes were adjusted to fit fluent skewness which is no higher than 0.5, whereas all cells in the volume mesh should meet a standard of the mesh quality which the non-orthogonality in OpenFOAM is no larger than 70. In the mesh generation, the triangle surface mesh and the tetra volume mesh were both constructed in ANSA v20. The mesh file was then exported by ANSA to OpenFOAM with a format of the fluent mesh file.



(a) Density boxes



(b) Numerical domain



(c) Boundary layer

Fig. 5.3 Visualization of mesh generation

To ensure accurate computations, some reasonable settings and boundary conditions are necessary. In this study, the primary flow variables, such as velocity $\boldsymbol{u}$, pressure p, and turbulent viscosity μ_t, were first prescribed. Table 5.3 shows the boundary conditions. It can be seen that the incoming flow was imposed on the inlet boundary of the velocity with a speed of 22m/s directed towards the vehicle model, whereas the outlet boundary of the pressure was set to 0. The "inletOutlet" condition was used for the outlet boundary of the velocity when the backward flow was unknown. For the turbulent viscosity, the inlet and outlet boundaries were both "calculated" conditions, which indicates the flied value was utilized. Since there is no moving ground device in the wind tunnel of Hiroshima University, the "noSlip" condition was imposed on the ground boundary of the velocity to simulate the effect of the ground. All surfaces of the vehicle model, including primary vehicle body and wheels, and the ground, were set to solid walls with the fixed value of zero, which means the velocity of the air flow was zero on the walls. The "nutUSpaldingWallFunction" condition was used to calculate

the turbulent viscosity using the corresponding wall function on the walls of the vehicle model and ground.

Table 5.3 Boundary condition of velocity, pressure, and turbulent viscosity.

Boundary	$\boldsymbol{u}$	p	μ_t
Inlet	fixedvalue uniform (22 0 0)	zeroGradient	Calculated
Outlet	inletOutlet uniform (22 0 0)	fixedvalue 0	Calculated
Car body	fixedvalue uniform (22 0 0)	zeroGradient	nutUSpaldingWallFunction
Wheel	fixedvalue uniform (22 0 0)	zeroGradient	nutUSpaldingWallFunction
Ground	noSlip	zeroGradient	nutUSpaldingWallFunction
Domain wall	slip	slip	slip

In OpenFOAM, the semi-implicit method of the pressure linked equations (SIMPLE) algorithm with GAMG for pressure and "smoothSolver" for momentum, k, and omega was adopted to calculate the flow filed around the vehicle model. Moreover, the discretization schemes also significantly influence the computational accuracy. To guarantee a high-quality spatial discretization, the second-order scheme is necessary. However, the second-order scheme is less stable, whereas the first-order scheme is significant stable but is less accurate. According to the references [109, 110], the reasonable discretization schemes for each term of the transport equations are listed in Table 5.4. These can better balance the stability and accuracy in simulations using OpenFOAM.

In simulations, the "simpleFoam" solver was used to run the SIMPLE algorithm. The tolerance of 1×10^{-6} for solving the flow variable p was set, whereas the convergence achieved was 7 decades fall for the flow variable of velocity and turbulent quantities of omega and k. To guarantee the convergence, all cases before and after deformation were run for 20000 iterations. Moreover, the moving average with a window for averaging 2000 iterations was employed to check whether a stable trend was reached in the end 10000 iterations. The convergence was reached when the moving average tended to a single value on the convergence curve of the Cd in the end 10000 iterations.

Table 5.4 Discretization scheme

Term	Subitem	Scheme	Order
Time		Steady state	
Gradient		Linear	Second
Convective	div(phi, U)	LinearUpwind	Second
	div(phi, k)	Upwind	First
	div(phi, omega)	Upwind	First
Interpolation		Linear	Second
Laplacian		Linear	Second

Notably, although a stable convergence is reached, the perturbation from the expected smooth response is impossible to be avoided. Some primary factors to cause the noise are explained as follows:

- Discretization error from changes in a mesh: The geometric properties of a realistic car are significantly complicated. The meshes are usually non-regular and contain incomplete prisms and highly skewing elements [32]. These elements have significant effects on the spatial discretization cross over the numerical domain. Unfortunately, some discretization errors are always caused by these distorted elements. When a deformation is implemented, the properties of the meshes in a local region around deformation walls of the vehicle body will be changed. Although the mesh reconstruction should be carried out for improving the overall quality of the meshes after deformation, two meshes before and after deformations are impossible to keep the same. These changes are possible to significantly change the flow characteristics in a sensitive area, such as the boundary layer, which leads to uncontrollable perturbations for different cases.
- Error from turbulence modeling: In past time, the performance of different RANS and DES turbulence models on predicting the aerodynamic properties of realistic vehicles has been widely investigated [108, 111]. The investigations show that none of the RANS and DES models can provide an accurate prediction for all vehicle configurations. In other words, stochastic errors cannot be avoided for different deformation models using a RANS model.
- High non-linearity of fluid mechanics: The characteristics of the flow field around a realistic vehicle are significantly complicated, particularly in the wake. All kinds of vortical structures lead to a steep variation of the pressure at the rear end of the vehicle. A deformation may significantly influence the wake structure, then making a sudden change of the physical property. Accordingly, a high non-linearity occurs in a local region of the design space.

5.5 Mesh independence and wind tunnel experiment

Before the optimization, the accuracy of the CFD simulation needs to be validated. As explained in Subsection 5.4, a deformation potentially leads to changes in a numerical mesh, which may produce some discretization errors over the numerical domain. Thus, the simulation should be as mesh-independent as possible. To test the mesh independence, five-level resolutions in the density boxes of A and B were set. The Cd of the baseline model was repeatedly measured when the different resolutions were used. Fig. 5.4 shows the results of the mesh independence. It can be seen that the Cd of the baseline model consistently decreases with the refinement of the mesh. However, the variation rate of the Cd is also significantly decreased. Evidently, when s and the number of mesh elements vary from 5 to 2 and 32 to 52 × 10^6, respectively, the Cd of 0.293 drops down to 0.285 with a total of 8 count declines (0.008). By contrast, the Cd has only one count (0.001) decline when the mesh is further refined from 52 to 79 × 10^6. Thus, the small variation of the Cd indicates that the mesh is relatively independent when s and the number of mesh elements change from 2 to 1.5 and 52 to 79 .52 × 10^6, respectively. To better balance the computational burden and accuracy in the optimization, the mesh strategy with s = 2 (s × h × MS = 4 mm) for density boxes of A and B, and a total of 52 × 10^6 mesh elements were chosen for all simulations.

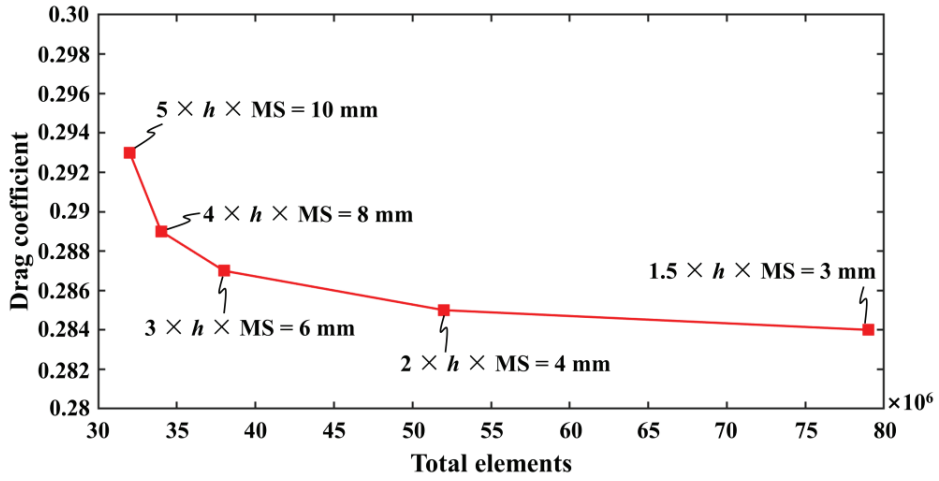


Fig. 5.4 Analysis of mesh independence

The wind tunnel experiment was further conducted to verify the rationality of the CFD simulation. The Cd of the baseline model without the deformation was measured in a wind tunnel at Hiroshima University. The wind tunnel has a square nozzle of 2 m × 2 m and measurement section of 4 m in length [112]. In this wind tunnel, the maximum wind speed was 25 m/s, whereas a wind speed of 22 m/s ($Re = 1.34 \times 10^6$ based on a reference length of 0.9225 m) was set to keep the same condition as the CFD simulation. Consequently, the Cd of 0.292 was obtained in the measurement. In comparison with the Cd of 0.285 in the CFD simulation, the error of the CFD simulation was 2.4%. Additionally, the previous studies [109, 111] comprehensively investigated the performances of different RNAS models on the DrivAer model. The K – omega – SST turbulence model showed relative robustness in the numerical simulation for all configurations of the DrivAer model. The errors 2.7% and 7.1% between the CFD simulations using the K – omega – SST turbulence model on the Estateback form and the wind tunnel experiments were obtained, respectively, in the previous studies [109, 111]. Apparently, the error 2.4% in this study is lower than those in the previous studies, which is acceptable for industrial automotive flow. This further verifies the accuracy of the CFD simulation in this study. In addition, it should be notable that the RANS models are usually applied to predict the magnitude and direction of a trend with a reasonable computational cost rather than an exactly accurate result in the CFD simulation. Since the RANS models can better balance the accuracy and computational cost, they have been widely used in the simulation of a complex geometry in SBO [16, 17, 111, 113].

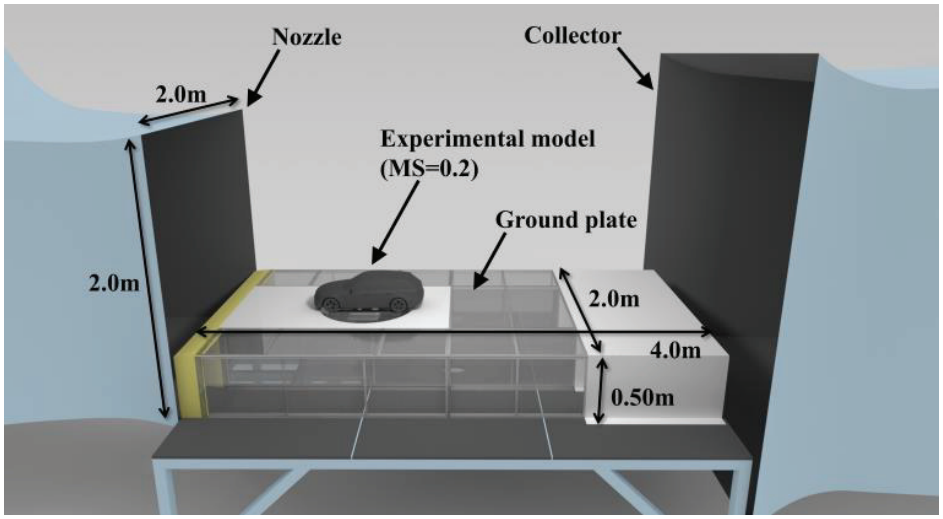


Fig. 5.5 Wind tunnel experiment

5.6 Procedure of optimization and automatic route

In the optimization, some necessary steps including sampling, mesh morphing, mesh generation, CFD simulation, and point infills need to be carried out in turn. To achieve these, some details are clarified as follows:

- Sampling: In the optimization of the vehicle drag reduction, 11D - 1 evaluation locations using the OLHD methodology constructed by the TPMESE algorithm with 250 iterations and the baseline model consisted of the initial evaluation locations. Subsequently, the Cds of these evaluation locations were evaluated based on the CFD simulations. Thus, a total of 55 sample points were obtained and applied to construct the initial surrogate model.
- Mesh: ANSA v20 was applied to carry out the mesh morphing, mesh reconstruction, and volume mesh generation.
- CFD simulation: OpenFOAM v 1706 was used to evaluate the Cd of an evaluation location.
- Stopping criterion: Until a stopping criterion was reached, the RKri-EGO-PEI system was used to repeatedly conduct point infills so that the optimization process can be guided towards the global convergence. However, it should be further clarified that in engineering optimization, it is usually impossible to obtain the optimal value by expensive experiments (such as vehicle aerodynamics). This is because those optimization cases are always simultaneously pressed over time and significantly expensive. Due to the practical limitations of the required time and expensive experimental burden, it is impossible to obtain the real mathematic relationship (mathematic function) and verify it without a considerable number of experiments. Thus, the corresponding optimum cannot also be found and verified. In this case, the applicability of each optimization method depends not only on its accuracy, but also on its time cost and robustness during the development of vehicle aerodynamics [32]. Evidently, achieving the maximum possible improvement within a limited time rather than searching for the real optimal solution with an unacceptable time cost meets a real-life need of expensive optimization problems much better. Take into consideration of computational burden in previous studies [13, 16, 17, 113, 114] and a shortly required time (about one week) to obtain the results of the optimization, 40 points with a total of 95 cases in the optimization were added using the proposed system.

To further reduce the labor costs and maximize the efficiency of the optimization, the morphing, mesh generation, and CFD simulation need to be automatically implemented. Fig. 5.6 shows the automatic route. Specifically, when the surrogate model was constructed or updated, and the updating points were predicted in the VAEO toolbox, the updating points were exported into a txt file. Subsequently, a Python script, namely, “ansa_run.py”, was used to launch the ANSA

and load an ANSA script file for importing the updating points and automatically carrying out the morphing, mesh construction, and mesh generation. After the completion of mesh preparations, the meshes were exported using a format of the Fluent mesh, and all meshes were uploaded on the ITO supercomputer. Based on the uploaded meshes, a shell script was applied to automatically control the file operations and start the CFD simulations.

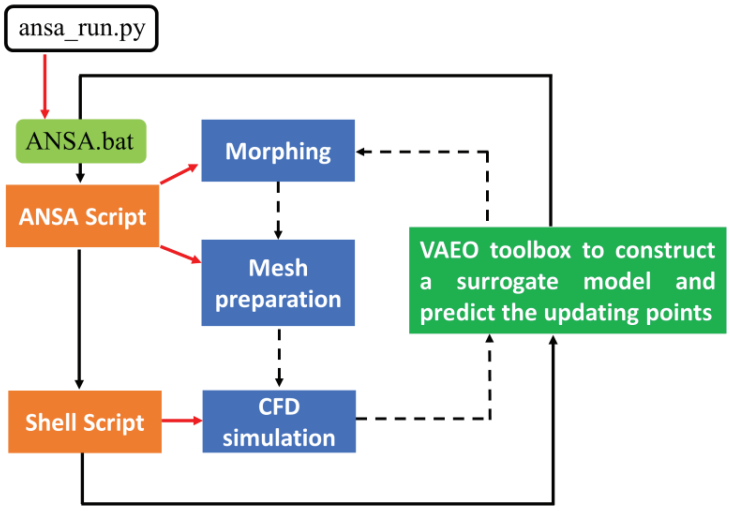


Fig. 5.6 Automatic route to carry out optimization

5.7 Performance of proposed system on real-world application with “noisy” computations

To investigate the performance of the proposed system, RKri-EGO-PEI, the other two existing systems, OK-EGO-PEI and RKri-EGO, were applied to optimize the shape of the Estateback model and guide the optimization process towards the global minimization through adding updating points. A total of 40 points were inserted into the initial samples using the three systems, respectively. In the initial 55 samples, the minimum Cd was 0.267. Thus, the improvement related to the minimum Cd in the initial samples means that a lower value of Cd than 0.267 is found in the point infills. To minimize the waiting time, 4 points were simultaneously added in each cycle of point infills using the RKri-EGO-PEI and OK-EGO-PEI systems, respectively. In the optimization, the necessary arguments for the constructions of the OK and RKri models were listed as follows:

- Regression function: *regpoly*0.
- Correlation function: *correxp.*
- The feasible region of the hyper-parameters: $[10^{-10}, 10^{10}]$.
- The feasible region of regression constant λ : $[10^{-5}, 10^{5}]$.

As discussed in Chapter 4, the MGBDE optimizer was applied to tune the Kriging hyperparameters and search for the optimum on the EI and PEI functions. All parameters of the MGBDE were the same as those set in the numerical tests in Subsection 4.3. To guarantee that the MGBDE optimizer to achieve a completed convergence, the number of iterations was set to 500. In the same way, two non-parametric statistical tests were adopted to confirm the significant difference of comparative results: (i) The Wilcoxon signed-rank test with a significance level of 0.05 was used to check the significant difference of the comparative systems during the process of the point infills. (ii) the Freidman test was used to calculate the mean rankings of the comparative systems during the process of the point infills. The sum of rankings R^+ and R^- calculated by the Wilcoxon signed-rank test indicates the first system performs better and worse than the second system, respectively. In the Freidman test, a smaller value of the mean rankings indicates a better performance of the optimization system, thus a higher rank can be assigned.

To compare the efficiency and convergences of different optimization systems, Fig. 5.7 and 5.8 show the convergence curves of Cd during the process of point infills, and the reduction of Cd with wall-clock time cost using different optimization systems, respectively. In this study, all time-consuming steps including mesh preparation, waiting, and numerical simulation must be carried out in turn when a single point-infill method is used. In contrast, these time-consuming steps can be implemented in a parallel manner. In comparison with the time-consuming steps, the time costs of tunning the Kriging hyperparameters and searching for the optimum based on the point-infill criterion are significantly short. Thus, the two-optimization processes can be ignored. In the optimization, one point infill roughly needs to take 8 h. More specifically, mesh preparation, waiting gap, and CFD simulation need to spend 0.15 h, 2.85 h, and 5 h, respectively. Based on the evaluation, if q points must be added in each cycle of point infills, each point averagely requires 8 / q h using the RKri-EGO-PEI and OK-EGO-PEI systems, whereas the RKri system still takes 8 h to add a point. From Fig. 5.7, it can be seen that the two RKri-based optimization systems can obtain a real improvement more easily compared with the OK-based optimization system. Apparently, the RKri-EGO-PEI system obtains a total of nigh count reductions in Cd with seven point infills, whereas the RKri-EGO converges to a worse solution than that of the RKri-EGO-PEI with 12 point

infills. In comparison with the two RKri-based optimization systems, the convergence of the OK-EGO-PEI system comes to a halt in the top 26 point infills. In the following, a continuous reduction of the Cd is achieved by the OK-EGO-PEI system from 27th to 32th point infills. Although the total reduction of the CD using the OK-EGO-PEI system is slightly larger than that of the RKri-EGO, it still spends considerably more computational costs. However, the parallel implementations for all time-consuming steps significantly reduce the real conducting time of the optimization process. As a result, the OK-EGO-PEI system still spends less wall-clock time converging to a lower Cd than that of RKri-EGO system. The proposed system shows the highest efficiency compared with the other comparative systems. Fig. 5.8 clearly shows that the RKri-EGO-PEI system converges to the lowest Cd with 14 h within a limited number of point infills, whereas the RKri-EGO and OK-EGO-PEI take 96 and 64 h to obtain two lower-level convergences than that of the RKri-EGO-PEI system, respectively. Generally, the RKri-EGO-PEI approximately spends 1/4 and 1/7 wall-clock time converging to a similar solution as that searched by the OK-EGO-PEI system, and searching for a better solution than that of the RKri-EGO system, respectively.

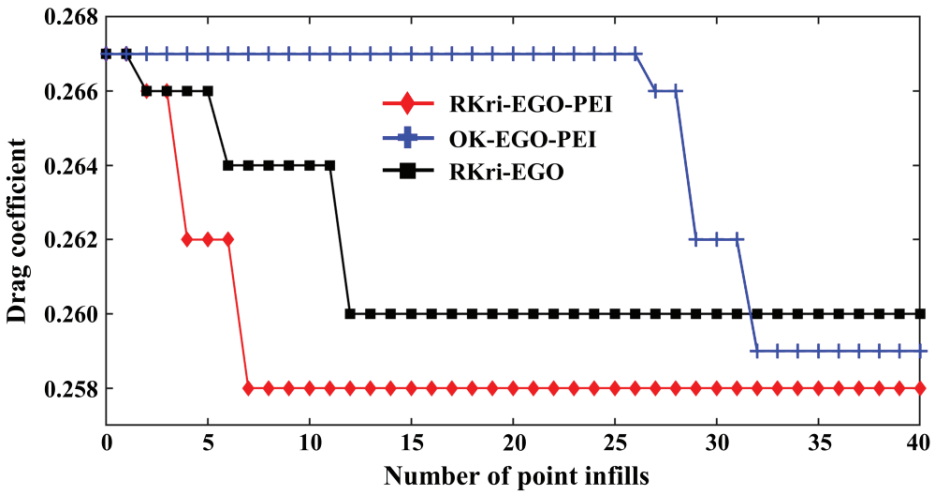


Fig. 5.7 Convergence curves of Cd obtained by different optimization systems

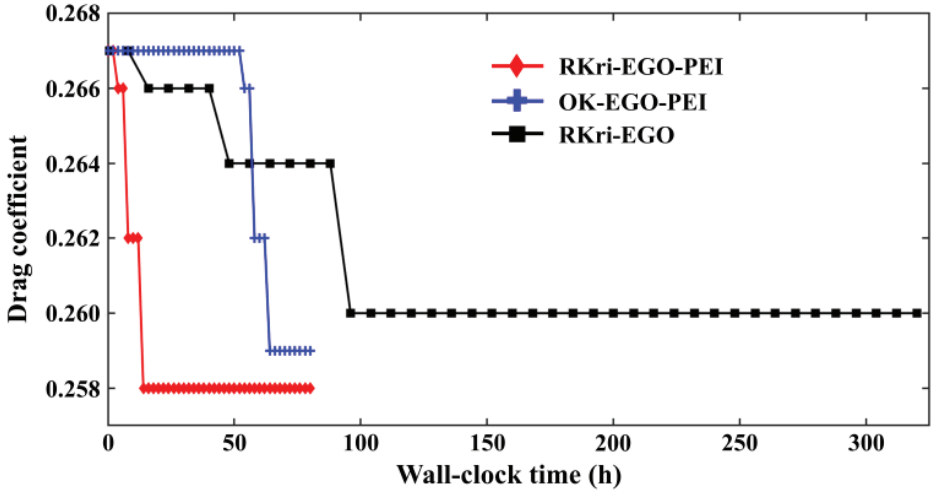


Fig. 5.8 Reduction of Cd with wall-clock time cost

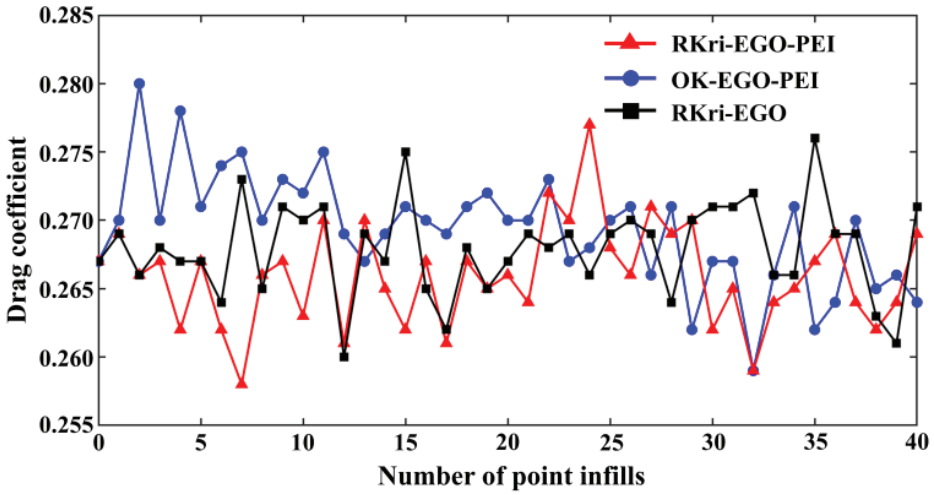


Fig. 5.9 Comparisons of Cds searched by different systems during the process of point infills

Table 5.5 Wilcoxon signed-rank test between different systems

System	R^+	R^-	*p* value
RKri-EGO-PEI versus RKri-EGO	442.5	152.5	0.0126
RKri-EGO-PEI versus OK-EGO-PEI	611.0	169.0	0.0020
RKri-EGO versus OK-EGO-PEI	519.5	260.5	0.0700

Table 5.6 Friedman test between different systems

Rank	System	Mean ranking	*p* value
1	RKri-EGO-PEI	1.5375	< 0.0001
2	RKri-EGO	1.9625	
3	OK-EGO-PEI	2.5000	

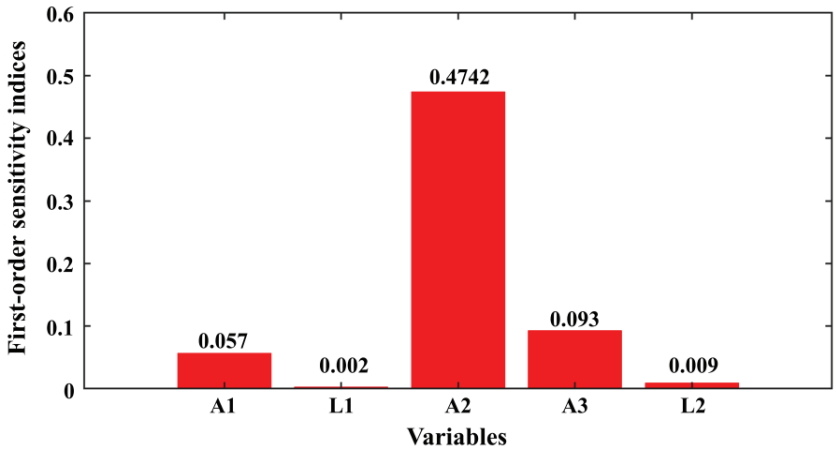
To further investigate the performance of the proposed system on obtaining an actual improvement, the Cds searched by the different optimization systems were recorded during the process of point infills. Fig. 5.9 shows the comparison of Cds searched by the different systems during the entire process of the point infills. When the number of point infills is zero, the Cds of all systems are the minimum Cd (0.267) in the initial samples. To confirm the significant difference of the comparative results, the Cds of the different systems were checked using the Wilcoxon signed-rank and the Friedman tests. Tables 5.5 and 5.6 list the statistical results. It can be observed that the OK-based optimization system cannot obtain a lower Cd than the minimum Cd (0.267) in the initial samples during the top 26 point infills. This is because the noise from the expected smooth response changes the functional property nearby with the minimum solution in the initial samples. Then, the information of the point-infill criterion guides the EGO-PEI algorithm to explore the region far away from the minimum solution in the initial samples, so that the predicted variance of the Kriging model can be reduced. Compared with the OK-EGO-PEI system, the two RKri-based systems more easily achieve an actual improvement compared with the minimum Cd in the initial samples during the process of point infills, particularly in the early stage. The statistical results listed in Table 5.5 also confirm the conclusion. In the comparisons of the RKri-EGO-PEI versus OK-EGO-PEI and RKri-EGO versus OK-EGO-PEI, the larger values of R^+ than those of R^- and the *p*-values of 0.0020 and 0.0700 ($p < 0.1$) show that the two RKri -based systems potentially have a higher opportunity to obtain a lower Cd during the process of point infills than that of the OK-EGO-PEI system. Correspondingly, the RKri-EGO-PEI and RKri-EGO obtained the first and second ranks in the Friedman test, respectively, which also demonstrates the higher performance of the two RKri-based systems. For the comparison between the two RKri-based systems, the larger value of R^+ than that of R^- in the comparison between the RKri-EGO-PEI and RKri-EGO, and the *p*-value of 0.0126 confirm that the RKri-EGO-PEI is easier to search a lower Cd than that of the RKri-EGO system. Although the RKri-EGO potentially has higher opportunity to search for a lower Cd than that of the OK-EGO-PEI, the OK-EGO-PEI converges to a similar solution as that searched by the RKri-EGO-PEI system, which is slightly lower than the lowest Cd searched

by the RKri-EGO system. Therefore, the EGO-PEI algorithm also shows better performance than that of the traditional EGO algorithm in this real-world optimization problem within a reasonable computational cost. Moreover, it can be seen that the RKri-EGO-PEI and RKri-EGO show some similar variation trends during the process of point infills. This further verifies the rationality of the combination between the RKri model and EGO-PEI algorithm. Generally, the RKri-based systems can more easily gain an actual improvement during the process of point infills than that of the OK-based optimization system, in the real-world optimization problem based on CFD simulations, through directly filtering out the noise from the expected smooth response.

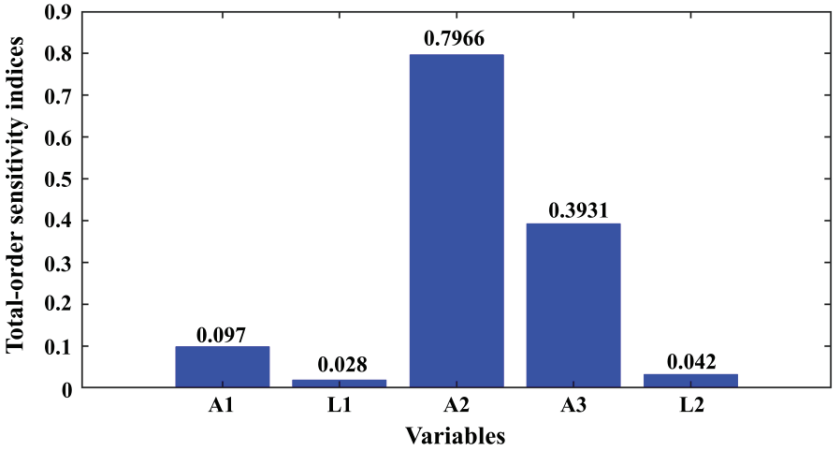
Table 5.7 Optimal results searched by different systems within a limited number of point infills

System	A1	A2	A3	L1 / mm	L2 / mm
RKri-EGO-PEI	+ 3.6°	+ 4.7°	+ 1.6°	+ 4.1	+ 1.7
OK-EGO-PEI	+ 3.6°	+ 4.5°	+ 1.6°	+ 4.1	+ 1.8
RKri-EGO	+ 3.6°	+ 4.2°	+ 1.0°	+ 4.2	+ 5.2

The optimal results searched by the different systems within a limited number of point infills are listed in Table 5.7. According to the results, the RKri-EGO-PEI and OK-EGO-PEI converge to two similar solutions, whereas the RKri-EGO system obtains a solution slightly far away from those solutions searched by the EGO-PEI-based systems. As discussed early, the two solutions obtained by the RKri-EGO-PEI and OK-EGO-PEI systems, respectively, correspond to the lower Cds than that of the solution searched by the RKri-EGO. Thus, it is clear to see that the local exploitation brings a faster convergence to the RKri-EGO in the optimization problem with all kinds of noise, but may lose some diversities, which leads to the RKri-EGO prematurely dropping into a local solution. By contrast, the EGO-PEI can bring more diversities to the RKri model, thereby achieving a more elegant balance of the local exploitation and global exploration. Moreover, the optimal values obtained by the three systems show the same trend for the minimization of the Cd: The relatively larger angles of A1 (engine hood), A2 (spoiler), and A3 (diffuser), a lower Cd. To analyze the effect of each design variable on the Cd, Fig. 5.10a and b show the first- and total-order sensitivity indices of the variables, respectively, based on the SGSA [93]. The first-order sensitivity indices indicate the primary effect of a single variable without interactions with other



(a) First-order sensitivity indices



(b) Total-order sensitivity indices

Fig. 5.10 sensitivity analysis

variables on the output variance, whereas the total-order sensitivity indices denote the effect of a single variable including interactions with the rest of variables on the output variance. For the calculation of the SGSA, the initial samples and all unique samples inserted using the three optimization systems were used to construct an OK model. Then, the OK model is applied to replace the expensive computer experiments for the evaluations of the evaluation locations generated by the Sobol sequence [87]. When the indices are higher than 0.05, the influences of variables on the objective are significant [115]. Although Table 5.7 implies that the larger L1 and L2 guide the optimization towards the convergence, the results of SA, as shown in Fig 5.10, show the two variables have no significant influences on the Cd. The first- and total-order

sensitivity indices of L1 and L2 are both lower than the 0.05 verify this conclusion. In contrast, three angles, particularly the angle of A2, significantly affect the Cd of the Estateback model. The variation trends of the variables for the minimization of the Cd are similar to those in the previous studies [113, 116], which further validates the rationality of the CFD simulation and optimization strategies. This is because the CFD simulations and optimization strategies guide the convergence of the optimization towards a right direction.

To demonstrate the performance of the proposed system on filtering out the noise from the expected smooth response during the process of point infills, the Euclidean distance between the current best solution and a new updating point searched by each optimization system was calculated. It can be predicted that the RKri-based optimization systems can focus more on searching for a region nearby with the current best solution through better accommodating the perturbation between two close points. Correspondingly, a closer approach reflected by a smaller Euclidean distance can be obtained when an optimization system filters out the noise better. Fig. 5.11 shows the results. To calculate the Euclidean distance, all evaluation locations should be first standardized using the Eq. (5.6).

$$\hat{x}_{ij} = (x_{ij} - \mu_j)/\sigma_j \tag{5.6}$$

Here, the x_{ij} is the value of j^{th} variable at i^{th} evaluation location, whereas the μ_j and σ_j are the mean and standard deviation of j^{th} variable, respectively. Evidently, when a solution which is better than the current best solution is searched, the current best solution should be dynamically updated by this solution during the process of point infills. From Fig. 5.11, the two RKri-based optimization systems focus more on searching for a region nearby with the current best solution, whereas the OK-EGO-PEI tends to searching for a region far away from the current best solution, particularly in the early stage of the point infills. The red and black markers are below the blue markers in most intervals, as shown in Fig. 5.11, which demonstrates the conclusion. The reason why such close updating points around the current best solution can be more easily searched by the RKri-based optimization systems can be derived. During the first cycle of point infills, the two RKri-based optimization systems searched for an updating point or some updating points nearby with the current best solution in the initial samples based on each initial RKri model, respectively. Due to the RKri model can filter out the perturbation after adding a close point, a smooth functional trend can be maintained. Thus, an updating point which was much closer to the current best solution could be more easily searched in the following cycles. Consequently, an improvement was also easier to be achieved when more closer updating points nearby with the current best solution were added gradually. In contrast, the OK model could not filter out the noise, which was easier to change a smoothly local

functional trend after inserting some updating points nearby with the current best solution in the first cycle. Therefore, the OK model guided the EGO-PEI algorithm to search for a region far away from the current best solution in the following stage. Only when was the predicted variance of the OK model reduced, the OK-EGO-PEI could search the region nearby with the current best solution again.

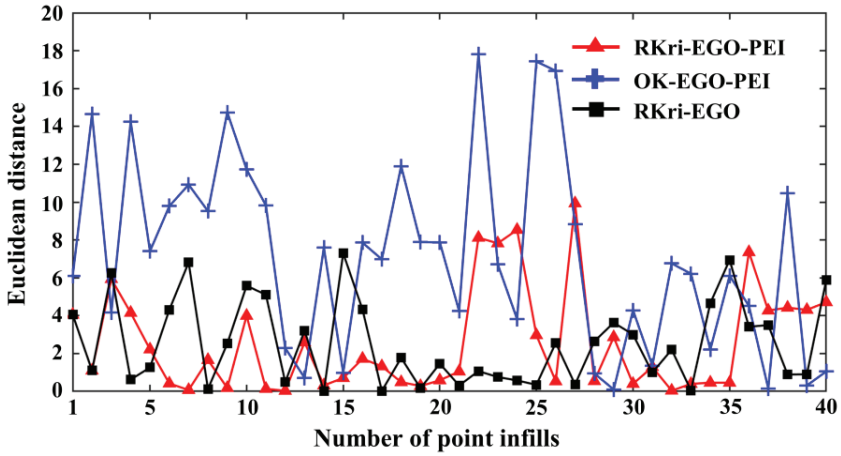


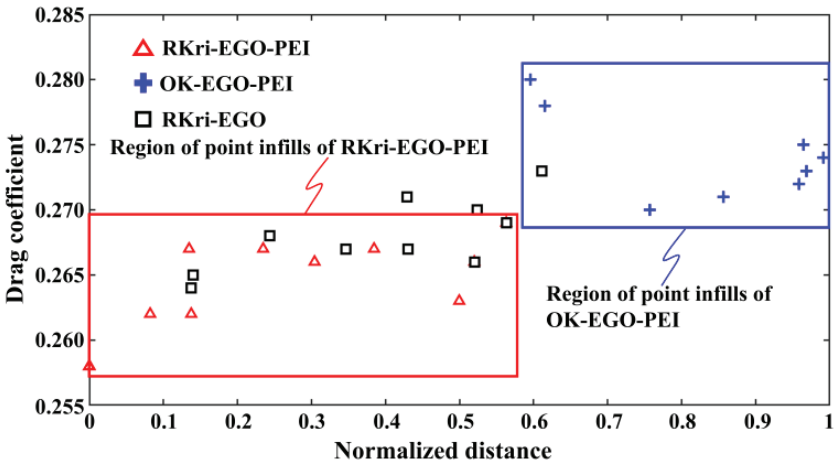
Fig. 5.11 Variation of Euclidean distance between the current best solution and a new updating point during the process of point infills

The variation of the Euclidean distance with an increase in the number of point infills also explains the reason for the high variation of the Cds searched by different systems during the process of point infills. The search process is a dynamic procedure using the EI and PEI criteria because the global exploration and local exploitation can be automatically balanced. According to the expressions of the EI and PEI functions, the search region nearby with the current best solution should be exploited when the relatively large response of the first term in the EI expression exists, whereas the search region far away from the current best solution can be explored when the predicted variance of Kriging model is relatively large. Fig. 5.11 shows the dynamic search process using the different optimization systems. Although the two RKri systems focused more on adding the updating points with a relatively small Euclidean distance, a region far away from the current best solution could also be explored during the process of the point infills. Similarly, the relatively large predicted variance of the OK model guided the OK-EGO-PEI to search for the region far away from the current best solution in the early stage, thus some higher Cds were obtained, as shown in Fig. 5.9. When the predicted variance of the OK model significantly

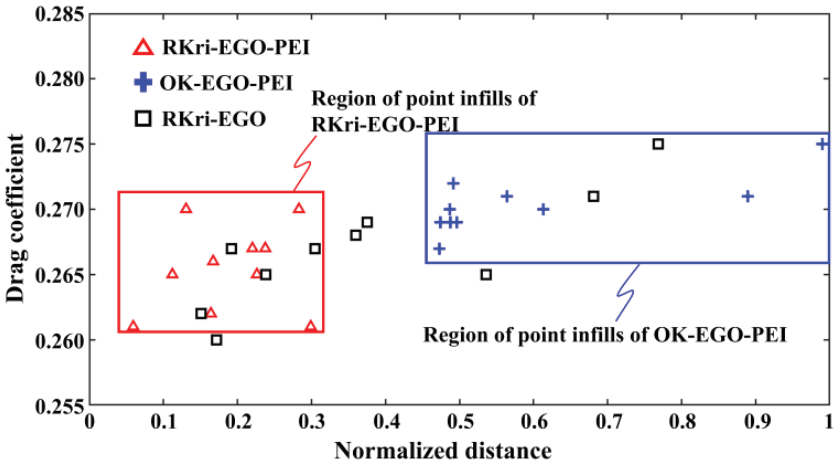
reduced, a region nearby with the current best solution was exploited. Consequently, a lower Cd was gradually obtained by the OK-EGO-PEI. In general, three systems dynamically added the points; thus the Cds searched by the different systems shifted during the process of point infills.

To further investigate the performance of the different optimization systems on the shape optimization of vehicle aerodynamics based on "noisy" CFD simulations, Fig. 5.12 calculates the normalized distances between the best solution searched by all systems and other updating points. As aforementioned, the best solution within a limited number of point infills was searched by the RKri-EGO-PEI system. To calculate the normalized distances, the Euclidean distances between the best solution of all inserted points and other updating points were first calculated. Then, these Euclidean distances were divided by the maximum Euclidean distance. It is clear to see that the two RKri-based solutions focus more on a local exploitation around the best solution searched by the three systems during the process of point infills. Evidently, the red markers are all on the left side of the blue markers, whereas the number of the black markers outside the blue region are more than that inside the blue region, as shown in Fig 5.12a and b. These demonstrate that the search regions of the RKri-based systems are closer to the best solution than that of the OK-EGO-PEI system in the top 20 point infills. The reason can be derived. Initially, the local functional trend around the current best solution was changed when some points around the current best solution were added by three systems. For the two RKri-based systems, the sudden perturbations after adding the close points could be filtered out; thus a smooth functional response was maintained. Based on this, more closer points around the current best solution could be gradually added using the two RKri-based systems. Consequently, the search regions on the lower left side of the region searched by the OK-EGO-PEI system were primarily exploited by the RKri-based systems. In contrast, the varied functional trend guided the OK-EGO-PEI to explore the region far away from the best solution, with a poor fitting quality, during top 20 point infills. During the final 20 point infills, the search regions of the three systems gradually approached with each other, as shown in Fig 5.12c and d. This is because the OK-EGO-PEI changed to search the region nearby with the best solution with the fast reduction of the predicted variance, whereas the RKri-based systems started to explore the region far away from the best solution as the loss of the improvement probability around it. Even if the search regions of the RKri-EGO-PEI and OK-EGO-PEI were significantly close to each other during the end ten point infills, the RKri-EGO-PEI system still aimed more at the search region nearby with the best solution. The conclusion can be verified by the blue region which has an equal lower but larger upper and right-side boundaries compared with the boundaries of the red region, as shown in Fig. 5.12d. Additionally, the RKri-

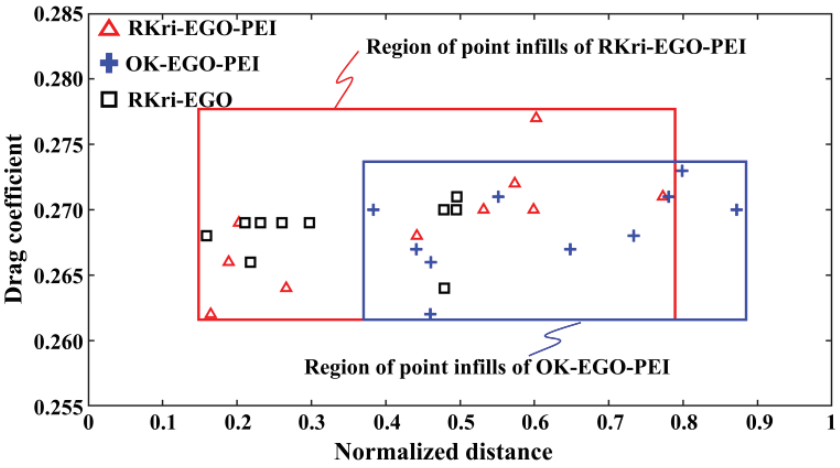
EGO system had more centralized search regions during the process of point infills. It can be seen that the black markers normally located at a region between red and blue regions. This provides a faster exploitation than that of the OK-EGO-PEI system, but may lose an elegant switch between the local exploitation and global exploration. Consequently, the RKri-EGO-PEI and OK-EGO-PEI converged to the similar solutions with lower Cds than that of the solution searched by the RKri-EGO system.



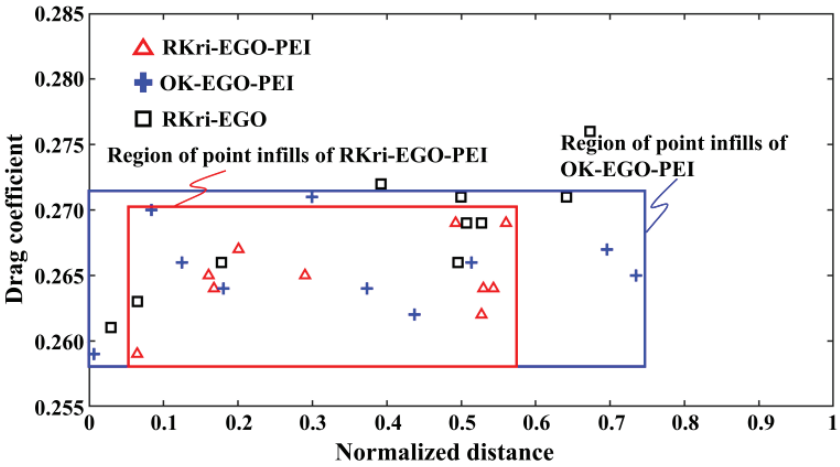
(a) First ten point infills



(b) Second ten point infills



(c) Third ten point infills



(d) End ten point infills

Fig. 5.12 Normalized distances between the best solution searched by all systems and other updating points during the process of point infills.

5.7 Conclusion

The primary goal of this chapter aims at investigating the performance of the proposed system, namely, RKri-EGO-PEI, on the drag reduction of the realistic vehicle with all types of noise produced by CFD simulations. In the optimization, the RKri-EGO-PEI system is easier to obtain a lower Cd than those of the RKri-EGO and OK-EGO-PEI systems.

Although the RKri-EGO system can also achieve an improvement with fewer point infills than that of the OK-EGO-PEI system during the process of point infills, the parallel implementations for all time-consuming steps, including mesh preparation, waiting gap, and CFD simulation, significantly shorten the real conducting time in the optimization. Consequently, the OK-EGO-PEI spends less wall-clock time achieving a larger reduction of the Cd than that of the RKri-EGO system. In summary, the RKri-based optimization systems can filter out noise from the expected smooth response, so that the smooth functional trend can be maintained after adding some close points. Thus, a lower Cd can be continuously obtained with the more insertions of close points around the current minimum Cd. In contrast, the OK-based system focuses more on an exploration for a region far away from the current minimum Cd, with a poor fitting quality. Only the predicted variance significantly reduces, the OK-based system can search for a region nearby with the current minimum solution. Additionally, the EGO-PEI algorithm brings more diversities to the optimization. Thus, the RKri-EGO-PEI and OK-EGO-PEI systems converge to two similar solutions which are slightly better than the solution searched by the RKri-EGO system within a limited number of point infills. Evidently, the proposed RKri-EGO-PEI system shows more robustness and efficiency than those of the OK-EGO-PEI and RKri-EGO systems in the real-world optimization problem with “noisy” computations. Not only is a faster convergence obtained by the RKri-EGO-PEI through directly filtering out noise using the RKri model, but also a higher-level convergence with high efficiency is achieved because the EGO-PEI algorithm with a higher diversity than that of the traditional EGO algorithm carries out the optimization in a parallel manner.

Chapter 6

Summary

The primary goal of this thesis is to provide an efficient framework of SBO for expensive optimization problems with "noisy" computations, such as the aerodynamic shape optimization of a vehicle. In real-world expensive optimization problems, it is usually impossible to obtain the optimal result by expensive experiments. This is because those optimization cases are always simultaneously pressed over time and significantly expensive. Due to the practical limitations of the required time and expensive experimental burden, it is impossible to obtain the real mathematic relationship (mathematic function) and verify it without a considerable number of experiments. Thus, the corresponding optimum cannot also be found and verified. In this case, the surrogate model is only a rough approximation of the functional trend between variables and objective. The primary focus of optimization is to quickly obtain the improvement of experimental objectives. Additionally, computer experiments become increasingly popular, and they have been widely applied to replace the practical physical experiments. However, the computer experiments always produce all kinds of noise, which leads to sudden perturbations from the expected smooth response and gives wrong information to the point-infill criterion. If the noise cannot be filtered out during the process of surrogate model modeling, more efforts must be carried out to find a better solution than the current best solution, thereby significantly slowing down the convergence speed of optimization. In this thesis, some efficient methodologies were improved to minimize the aerodynamic drag of a realistic vehicle with complicated geometric features using the CFD simulations with all kinds of noise. Generally, these methodologies aim at achieving the maximum possible improvement of objective, such as Cd, using as fewer samples as possible within a limited time. The primary contributions and conclusions of the investigation are further summarized in Subsection 6.1 in detail.

6.1 Primary contributions and conclusions

1) Methodologies of SBO for expensive problems with "noisy" computations

To achieve the goal of solving expensive optimization problem with "noisy" computations efficiently, some critical methodologies, including sampling using OLHD, Kriging metamodeling, and point-infill criteria, in SBO, were reviewed

in Chapter 2. By referencing the traditional ESE algorithm, two algorithms, namely, TPMESE and MESE, were developed to efficiently optimize LHDs with high space-filling quality so that a high-quality surrogate model can be constructed using as fewer initial samples as possible. Then, a new optimization system, namely, RKri-EGO-PEI, was proposed. The system aims at filtering out the noise from the expected smooth response using the RKri surrogate model and carrying out the optimization in a parallel manner based on the EGO-PEI algorithm. Since the Kriging hyperparameter tunning and searching for the optimum on the point-infill criterion are both optimizations, the performance of an optimizer significantly influences the proposed system. Therefore, the influence of an optimizer on the EGO-PEI algorithm was briefly analyzed. In summary, the global search capability of an optimizer to accurately search for the optimum ensures a more accurate approximation between the PEI and EI criteria, thus guaranteeing the EGO-PEI works properly though an approximate mechanism of action with the traditional EGO algorithm. In contrast, the local search capability for fast exploitation in a local region provides more differences between EI and PEI criteria, which brings more diversities to the EGO-PEI algorithm. Consequently, an elegant balance between the difference and approximation of EI and PEI criteria promotes that the EGO-PEI algorithm performs better than the traditional EGO algorithm on certain occasions.

2) A MATLAB toolbox for SBO

To provide an automatic route for the implementation of SBO and highly extendable program in future development, a toolbox, namely, VAEO toolbox, using the OOP, was developed. The VAEO toolbox provides a wide selection of Kriging surrogate models, optimizers, point-infill strategies, statistical analysis, and data analysis. Based on these techniques, the complicated global, multi-objective, and constrained optimization problems can be efficiently solved. For the programming design, the design pattern makes the VAEO toolbox highly extendable for users. Thus, a new SBO technique, such as a Kriging model, optimizer, and point-infill strategy, can be programmed easily. The performance of the VAEO toolbox was tested by the comparison with other state-of-the-art programs. First, the VAEO toolbox was compared against the ooDACE toolbox for Kriging metamodeling using different optimizers. The results showed that the VAEO toolbox provided more heavy strategies to tune the Kriging hyperparameters than those in the ooDACE toolbox. Although the heavy strategies, such as MGBDE, AGDE, and BBPSO-DE, in the VAEO toolbox are more time-consuming than the optimizers of GA and SQP in the ooDACE toolbox, a higher-quality Kriging can be constructed using these strategies. Subsequently, the VAEO toolbox was used to compare with the pySOT for solving a specific global optimization problem. The result showed that the point-infill strategies in the VAEO toolbox converged better and faster than those in the pySOT

because the critical processes of tunning Kriging hyperparameters and searching for the optimum on the point-infill criterion could be better guaranteed by the MGBDE optimizer. Therefore, it is clear to see that the VAEO toolbox can provide not only an extendable environment for users to develop new techniques of SBO, but also more efficient strategies to solve the global optimization problem flexibly. In fact, the VAEO toolbox made huge contribution to the whole investigation.

3) Efficient construction of OLHD with high-space filling quality

The MESE and TPMESE algorithms were proposed to efficiently construct OLHDs with high space-filling quality. To confirm their performance, the proposed algorithm and each of their original algorithms were compared with several state-of-the-art heuristic algorithms, including ILS, LSGA, and PermGA, on five sizes of LHDs. The tested LHDs included two small (30 × 3 and 40 × 4), one medium (50 × 5), and two large (60 × 6 and 100 × 10) sizes. In the numerical tests, The TPMESE, MESE, and each of their original algorithms, showed considerably high efficiency than that of ILS, LSGA, and PermGA algorithms. Additionally, the TPMESE and MESE algorithms generally converged faster than each of their original algorithms, including TPESE and ESE, respectively, except the optimization of LHD with smallest size of 30 × 3. The TPLHD as an initial design significantly accelerated the convergence speed at the beginning of optimization in comparison with a random LHD as an initial design. The reason can be analyzed. On the one hand, the TPLHD can construct a near high-quality design almost without time costs, whereas heuristic algorithms, such as MESE and ESE, stating from a random LHD must converge to the same-level design running thousands of iterations, which has a higher time cost than that of generating a TPLHD; on the other hand, the MESE and ESE have enough diversities to fast escape from a local design, thereby maintaining the leading position from the TPLHD as an initial design.

4) Rationality of combination between RKri and EGO-PEI

As explained in Subsections 2.3 and 2.4, the reason why the EGO-PEI algorithm can work well without the updates of Kriging model in each cycle of point infills is that the PEI criterion substantially provides an approximation with EI criterion. Thus, the rationality of the combination between the RKri model and EGO-PEI algorithm was investigated before the application of RKri-EGO-PEI in a real-world optimization problem. The variations of the PEI and EI functions using the RKri model was tested on a one-dimensional benchmark function. The results showed that the PEI criterion still provided an approximation with the EI criterion when the RKri model was used. Therefore, the rationality of the

combination between the RKri model and EGO-PEI algorithm was illustrated.

5) Selection of optimizer

The optimizer significantly influences the performance of the EGO-PEI algorithm, no matter what type of Kriging model is. This is because two important processes, including the Kriging hyperparameter tunning and searching for the optimum on the PEI function, are influenced by the optimizer. To select a suitable optimizer, seven optimizers, including DE, RCGA, PSO, SHADE, AGDE, MGBDE, and BBPSO-DE, with different performance advantages, were tested on six benchmark functions chosen from the CEC 2013 benchmark suites using the OK-based EGO-PEI algorithm first. Then, the best optimizer of MGBDE in the previous tests was further tested on four benchmark functions through the comparison with the SHADE optimizer with the highest global search capability. The results of numerical testes showed that the MGBDE with a higher central goal of exploitation-exploration had the highest performance on the EGO-PEI algorithm. In general, the MGBDE could not only ensure that the EGO-PEI algorithm converged to the same-level solution as that of the EGO-PEI algorithm using other optimizers, with fewer iterations, it also had a higher chance to promote the convergence of the EGO-PEI algorithm than those of the other competitive optimizers within a limited number of point infills. The mechanism of the higher central goal of exploitation-exploration for promoting the convergence of the EGO-PEI algorithm was discussed. A higher global search capability from an optimizer substantially provides more approximation between the EI and PEI criterion, thereby ensuring that the EGO-PEI works properly through a similar mechanism to the traditional EGO algorithm, whereas a higher local search capability from an optimizer to a high-quality solution rather than the real global optimum brings more differences between the EI and PEI functions, thus giving more diversities to the EGO-PEI algorithm. In summary, the higher central goal of exploitation-exploration from the MGBDE produced a more elegant balance between the difference and approximation of EI and PEI criteria, which made the EGO-PEI perform better. Therefore, the MGBDE was adopted as the optimizer in the proposed RKri-EGO-PEI system.

6) Aerodynamic shape optimization of a realistic vehicle using improved methodologies

The proposed optimization system, called RKri-EGO-PEI, was then applied to minimize the Cd of DrivAer Estateback model with five variables based on CFD simulations with all kinds of noise. To further reduce the labor during the process of optimization, an automatic route was developed. Based on this route, the mesh morphing, mesh preparation, and CFD simulations can be automatically carried out using the ANSA and shell scripts. Subsequently, the performance of the RKri-

EGO-PEI was confirmed by the real-world optimization problem started from a 55 initial samples with 40 point infills. To compare with the RKri-EGO-PEI system, the two existing systems, including RKri-EGO and OK-EGO-PEI, were also applied to carry out the same number of point infills in the minimization of Cd of the Estateback model. The results revealed that the RKri-EGO-PEI system is the winner in the real-world optimization problem with "noisy" computations. Not only was the RKri-EGO-PEI easier to search for a lower Cd than that of the RKri-EGO and OK-EGO-PEI systems, but it finally obtained a lower Cd which is lower than the Cds searched by other systems within a limited number of point infills. Although the RKri-EGO could also search for a lower Cd than the current minimum Cd with less computational burden (fewer point infills) than that of the OK-EGO-PEI system, the implementations of all time-consuming steps in a parallel manner reduced the real conducting time of the OK-EGO-PEI system. Moreover, the EGO-PEI algorithm brings more diversities to the optimization, thereby achieving a convergence which is close to the RKri-EGO-PEI but is slightly better than the RKri-EGO, for OK-EGO-PEI system. The optimal results searched by the different systems and the SGSA of the variables confirmed the rationality of the optimization process. This is because the effects of variables on the Cd and the trend of the convergence were consistent with previous efforts. In summary, the RKri model can filter out the noise from the expected smooth response so that a smooth functional trend can be maintained, particularly in the local region nearby with the current best solution, whereas the EGO-PEI algorithm allows implementations of time-consuming steps in a parallel manner. Thus, the RKri-EGO-PEI showed highest robustness and efficiency in the expensive optimization problem with "noisy" computations.

6.2 Future research

In industrial optimization, the balance between the computational cost and global convergence must be carefully considered because optimization cases are usually pressed over time and considerably expensive. Under this situation, the time becomes the primary limitation to carry out the optimization in industry, particularly for high dimensional problems. Therefore, an efficient optimization framework should effectively reduce the required number of samples and achieve the maximum possible improvement with a reasonable computational cost. To achieve this goal, two points need to be further investigated in the future.

1) Surrogate model with gradient information

To construct a high-quality surrogate model, a certain number of initial samples must be given. Unfortunately, there is

not a specific standard for the number of initial samples because many factors influence its selection. Generally, the required number of initial samples should be increased with increases in the complexity of the relationship between variables and objective, and the dimensions of problems. It is easy to predict that a lot of samples must be given when the spatial property is complex (e.g., high nonlinearity), or the dimension of a case is high. Take the OK model for example, the literatures [31] suggest that the number of initial samples can be set to $11 \times D - 1$, which can provide a relatively robust performance for fitting the OK model with an acceptable approximation quality. However, it is still heavy burden for expensive optimization problems, particularly the situation in which additional samples also need to be added in the subsequent optimization process. To achieve the goal of obtaining the maximum possible improvement using as few samples as possible, a surrogate model with additional information, such as the first-order gradient of the objective with respect to variables, is an effective measure to reduce the required number of initial samples as well as promote faster convergence during the process of point infills. The reasons can be derived. On the one hand, a surrogate model with gradient information, such as GEK surrogate model [66, 67], is usually beneficial to extracting and capturing more special information from a limited number of samples, thereby achieving the same-level approximation as that using a normal model, with fewer samples. On the other hand, a more accurate approximation can also provide more effective information for a point-infill criterion, thus ensuring a more efficient process of point infills.

2) Point infill strategy with higher efficiency in a parallel manner

Another way to further reduce the number of samples and real conducting time in optimization is to carry out point infills in a highly efficient manner. Although the PEI criterion achieves high-quality parallel processes of point infills in this study, its efficiency (average improvement of each point) is significantly weakened with an increase in the number of point infills in each cycle, particularly in the early stage of point infills. Consequently, the convergence speed of the PEI criterion is reduced within a certain total number of point infills. Thus, more computational costs with more point infills possibly need to be taken so that the same convergence level as that using a single point infill method can be reached. This may weaken or invert the advantage of saving time through the implementations of point infills in a parallel manner. The reference [29] clearly illustrated the reasons why the efficiency of the PEI criterion is weakened with an increase in the number of point infills. First, the PEI criterion must be multiplied by more IFs when more points need to be added in each cycle. Therefore, the PEI function becomes increasingly unreliable in comparison with the original EI function. Second, the number of Kriging updates is reduced when a certain total number of point infills is given. As a result, the

effective information from the Kriging updates, received by each updating point, is also decreased. Therefore, the average improvement of each point is weakened. To minimize the time cost in optimization, a point-infill criterion needs to ensure a process of point infills not only in a parallel manner but also with high efficiency. Thus, it is worth to further investigate and develop a more powerful point-infill criterion in the future.

Reference

[1] Huluka A W, Kim C H (2019) Numerical study on aerodynamic drag reduction and energy harvest for electric vehicle: a concept to extend driving range. In: IOP Conf. Ser.: Mater. Sci. Eng. pp 012009. IOP Publishing.

[2] Mukut A M I, Abedin M Z (2019) Review on aerodynamic drag reduction of vehicles. Int. J. Eng. Mater. Manuf. 4(1), 1-14. https://doi.org/10.26776/ijemm.04.01

[3] Hucho W H, Janssen L J, Emmelmann H J (1976) The optimization of body details—a method for reducing the areodynamic drag of road vehicles. SAE Transactions 85(2), 865-882.

[4] Hucho W, Sovran G (1993) Aerodynamics of road vehicles. Annual review of fluid mechanics 25(1), 485-537.

[5] d'Apollonia A S, Granier B, Debaty P (2004) Design of experiments methods applied to CFD simulations in vehicle aerodynamics. SAE transactions 113(6), 187-196.

[6] Howell J, Gaylard A (2006) Improving SUV aerodynamics. In: 6th MIRA International Vehicle Aerodynamics Conference, pp 1-17. Gaydon, United Kingdom.

[7] Krishnani P N, Zhou D (2009) CFD analysis of drag reduction for a generic SUV. In: ASME International Mechanical Engineering Congress and Exposition, 43864, pp 589-598.

[8] Kang S O, Jun S O, Park H I, Song K S, Kee J D, Kim K H, Lee D H (2012) Actively translating a rear diffuser device for the aerodynamic drag reduction of a passenger car. Int J Automot Technol 13(4): 583-592. https://doi.org/10.1007/s12239-012-0056-x

[9] Li Z, Dong Z, Liang Z, Ding Z (2021) Surrogate-based distributed optimisation for expensive black-box functions. Automatica 125, 109407. https://doi.org/10.1016/j.automatica.2020.109407

[10] Przysowa K, Łaniewski-Wołłk Ł, Rokicki J (2018) Shape optimisation method based on the surrogate models in the parallel asynchronous environment. Appl Soft Comput. 71: 1189-1203. https://doi.org/10.1016/j.asoc.2018.04.028

[11] Ando K, Takamura A, Saito I (2010) Automotive aerodynamic design exploration employing new optimization methodology based on CFD. SAE Int. J. Passeng. Cars - Mech. Syst. 3(2010-01-0513), 398-406.

[12] Khondge A D, Sovani S, Verma G (2011) Automation of vehicle aerodynamic shape exploration and optimization using integrated mesh morphing and CFD. SAE International 2011-01, 0170.

[13] Song K S, Kang S O, Jun S O, Park H I, Kee J D, Kim K H, Lee D H (2012) Aerodynamic design optimization of

rear body shapes of a sedan for drag reduction. Int. J. Automot. Technol. 13(6): 905–914. https://doi.org/10.1007/s12239-012-0091-7

[14] Zhang Y, Ding W, Zhang Y (2014) Aerodynamic shape optimization based on the MIRA reference car model. SAE Technical Paper 2014-01-0603. https://doi.org/10.4271/2014-01-0603

[15] Sun S, Chang Y P, Fu Q, Zhao J, Ma L, Fan S, Friz H (2014) Aerodynamic shape optimization of an SUV in early development stage using a response surface method. SAE Int J Passeng. Cars – Mech. Syst. 7(4):1252-1263. https://doi.org/10.4271/2014-01-2445

[16] Beigmoradi S, Hajabdollahi H, Ramezani A (2014) Multi-objective aero acoustic optimization of rear end in a simplified car model by using hybrid robust parameter design, artificial neural networks and genetic algorithm methods. Comput Fluids 90: 123-132. https://doi.org/10.1016/j.compfluid.2013.11.026

[17] Wang D, Zhang S, Zhang S, Wang, Y (2019) Analysis and multi-objective optimization design of wheel based on aerodynamic performance. Adv Mech Eng 11(5): 1687814019849733. https://doi.org/10.1177/1687814019849733

[18] Urquhart M, Varney M, Sebben S, Passmore M (2020) Aerodynamic drag improvements on a square-back vehicle at yaw using a tapered cavity and asymmetric flaps. Int J Heat Fluid Flow 86: 108737. https://doi.org/10.1016/j.ijheatfluidflow.2020.108737

[19] Akhtar T, Shoemaker C A (2016) Multi objective optimization of computationally expensive multi-modal functions with RBF surrogates and multi-rule selection. J Glob. Optim. 64(1), 17-32. https://doi.org/10.1007/s10898-015-0270-y

[20] Chen Z, Zhou Y, He X (2019) Handling expensive multi-objective optimization problems with a cluster-based neighborhood regression model Appl. Soft Comput. 80, 211-225. https://doi.org/10.1016/j.asoc.2019.03.049

[21] Forrester A I, Keane A J (2009) Recent advances in surrogate-based optimization. Prog. Aerosp. Sci. 45(1-3), 50-79. https://doi.org/10.1016/j.paerosci.2008.11.001

[22] Forrester A I, Keane A J, Bressloff N W (2006) Design and analysis of "noisy" computer experiments. AIAA J 44(10): 2331-2339. https://doi.org/10.2514/1.20068

[23] Kleijnen J P (2017) Regression and Kriging metamodels with their experimental designs in simulation: a review. Eur. J. Oper. Res. 256(1), 1-16. https://doi.org/10.1016/j.ejor.2016.06.041

[24] Jones D R, Schonlau M, Welch W J (1998) Efficient global optimization of expensive black-box functions. J Glob

Optim 13(4): 455-492. https://doi.org/10.1023/A:1008306431147

[25] Jones D R (2001) A taxonomy of global optimization methods based on response surfaces. J. glob. optim. 21(4), 345-383. https://doi.org/10.1023/A:1012771025575

[26] Sasena M J, Papalambros P, Goovaerts P (2002) Exploration of metamodeling sampling criteria for constrained global optimization. Eng. Optim, 34(3), 263-278. https://doi.org/10.1080/03052150211751

[27] Booker A J, Dennis J E, Frank P D, Serafini D B, Torczon V, Trosset M W (1999) A rigorous framework for optimization of expensive functions by surrogates. Struct. Optim. 17(1), 1-13. https://doi.org/10.1007/BF01197708

[28] Ginsbourger D, Le Riche R, Carraro L (2010) Kriging is well-suited to parallelize optimization. Computational Intelligence in Expensive Optimization Problems. Adaptation Learning and Optimization 2: 131–162. https://doi.org/10.1007/978-3-642-10701-6_6

[29] Zhan D, Qian J, Cheng Y (2017) Pseudo expected improvement criterion for parallel EGO algorithm. J. Glob. Optim. 68(3): 641-662. https://doi.org/10.1007/s10898-016-0484-7

[30] Qian J, Cheng Y, Zhang J, Liu J, Zhan D (2021). A parallel constrained efficient global optimization algorithm for expensive constrained optimization problems. Eng. Optim. 53(2): 300-320. https://doi.org/10.1080/0305215X.2020.1722118

[31] Zhan D, Cheng Y, Liu J (2017) Expected improvement matrix-based infill criteria for expensive multiobjective optimization. IEEE Trans. Evol. Comput. 21(6): 956-975. https://doi.org/10.1109/TEVC.2017.2697503

[32] Blacha T, Gregersen M, Islam M, Bensler H (2016) Application of the adjoint method for vehicle aerodynamic optimization. SAE Technical Paper 2016-01-1615. https://doi.org/10.4271/2016-01-1615

[33] Briffoteaux G, Gobert M, Ragonnet R, Gmys J, Mezmaz M, Melab N, Tuyttens D (2020). Parallel surrogate-assisted optimization: Batched bayesian neural network-assisted ga versus q-ego. Swarm. Evol. Comput. 57: 100717. https://doi.org/10.1016/j.swevo.2020.100717

[34] Qiu S, Xue Z, He H, Yang Z, Xia E, Xu C, Li L (2021). Multi-objective optimization study on the power cooling performance and the cooling drag of a full-scale vehicle. Struct. Multidiscipl. Optim. 64(6): 4129-4145. https://doi.org/10.1007/s00158-021-03035-6

[35] Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010 pp. 177-186. Physica-Verlag HD. https://doi.org/10.1007/978-3-7908-2604-3_16

[36] McKay M D, Beckman R J, Conover W J (1979) Acomparisonof three methodsforselecting valuesofinputvariablesinthe analysisofoutputfrom acomputercode. Technometrics 21(2), 239-245.

[37] Iman R L, Conover W J (1980) Small sample sensitivity analysis techniques for computer models. with an application to risk assessment. Commun. Stat. Theory Methods 9(17), 1749-1842.

[38] Li W W, Jeff Wu C F (1997) Columnwise-pairwise algorithms with applications to the construction of supersaturated designs. Technometrics 39(2), 171-179.

[39] Fang K T, Ma C X, Winker P (2002) Centered L_2-discrepancy of random sampling and Latin hypercube design, and construction of uniform designs. Math. Comput. 71(237), 275-296.

[40] Bates S, Sienz J, Toropov V (2004) Formulation of the optimal Latin hypercube design of experiments using a permutation genetic algorithm. In: 45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference pp. 2011. Palm Springs.

[41] Grosso A, Jamali A R M J U, Locatelli M (2009) Finding maximin latin hypercube designs by iterated local search heuristics. Eur. J. Oper. Res. 197(2), 541-547. https://doi.org/10.1016/j.ejor.2008.07.028

[42] Jin R, Chen W, Sudjianto A (2005) An efficient algorithm for constructing optimal design of computer experiments. J. Stat. Planning Inference, 134(1), 268-287.

[43] Chantarawong T, Rungrattanaubol J, Na-udom A (2010) Enhancement of enhanced stochastic evolutionary algorithm for computer simulated experiment. Inf. Technol. J. 6(2), 65-69.

[44] Zhu H, Liu L, Long T, Peng L (2012) A novel algorithm of maximin Latin hypercube design using successive local enumeration. Eng. Optim. 44(5), 551-564. https://doi.org/10.1080/0305215X.2011.591790

[45] Chen R B, Hsieh D N, Hung Y, Wang W (2013) Optimizing Latin hypercube designs by particle swarm. Statist. Comput. 23(5), 663-676. https://doi.org/10.1007/s11222-012-9363-3

[46] Pholdee N, Bureerat S (2015) An efficient optimum Latin hypercube sampling technique based on sequencing optimisation using simulated annealing. Int. J. Syst. Sci. 46(10), 1780-1789. https://doi.org/10.1080/00207721.2013.835003

[47] Kianifar M R, Campean F, Wood A (2014) PermGA algorithm for a sequential optimal space filling DoE framework. In: 2014 14th UK Workshop on Computational Intelligence (UKCI) pp. 1-8. IEEE.

[48] Mahmoudi H, Zimmermann H (2015) On optimal latin hypercube design for yield analysis of analog circuits. In:

2015 Austrian Workshop on Microelectronics pp. 46-49. IEEE.

[49] Ba S, Myers W R, Brenneman W A (2015) Optimal sliced Latin hypercube designs. Technometrics 57(4), 479-487. https://doi.org/10.1080/00401706.2014.957867

[50] Joseph V R, Gul E, Ba S (2015) Maximum projection designs for computer experiments. Biometrika 102(2), 371-380. https://doi.org/10.1093/biomet/asv002

[51] Joseph V R (2016) Space-filling designs for computer experiments: A review. Qual. Eng. 28(1), 28-35. https://doi.org/10.1080/08982112.2015.1100447

[52] Long T, Wu D, Chen X, Guo X, Liu L (2016) A deterministic sequential maximin Latin hypercube design method using successive local enumeration for metamodel-based optimization. Eng. Optim. 48(6), 1019-1036. https://doi.org/10.1080/0305215X.2015.1081518

[53] Le Guiban K, Rimmel A, Weisser M A, Tomasik J (2018) The first approximation algorithm for the maximin Latin hypercube design problem. Oper. Res. 66(1), 253-266. https://doi.org/10.1287/opre.2017.1665

[54] Wu Z, Wang D, Okolo P N, Zhao K, Zhang W (2017) Efficient space-filling and near-orthogonality sequential Latin hypercube for computer experiments. Comput. Methods Appl. Mech. Eng. 324, 348-365. https://doi.org/10.1016/j.cma.2017.05.020

[55] Li W, Lu L, Xie X, Yang M (2017) A novel extension algorithm for optimized Latin hypercube sampling. J. Stat. Comput. Simul. 87(13), 2549-2559. https://doi.org/10.1080/00949655.2017.1340475

[56] Xiao Q, Xu H (2017) Construction of maximin distance Latin squares and related Latin hypercube designs. Biometrika 104(2), 455-464. https://doi.org/10.1093/biomet/asx006

[57] Shang X, Chao T, Ma P, Yang M (2020) An efficient local search-based genetic algorithm for constructing optimal Latin hypercube design. Eng. Optim. (52)2, 271-287. https://doi.org/10.1080/0305215X.2019.1584618

[58] Viana F A, Venter G, Balabanov V (2010) An algorithm for fast optimal Latin hypercube design of experiments. Int. J. Numer. Methods Eng. 82(2), 135-156. https://doi.org/10.1002/nme.2750

[59] Damblin G, Couplet M, Iooss B (2013) Numerical studies of space-filling designs: optimization of Latin Hypercube Samples and subprojection properties. J. Simul. 7(4), 276-289. https://doi.org/10.1057/jos.2013.16

[60] Husslage B G, Rennen G, Van Dam E R, Den Hertog D (2011) Space-filling Latin hypercube designs for computer experiments. Optim. Eng. 12(4), 611-630. https://doi.org/10.1007/s11081-010-9129-8

[61] Morris M D, Mitchell T J (1995) Exploratory designs for computational experiments. J. Stat. Planning Inference 43(3), 381-402.

[62] Saab Y G, Rao V B (1991) Combinatorial optimization by stochastic evolution. IEEE Trans. Comput.-Aided Design Integr. 10(4), 525-535.

[63] Couckuyt I, Forrester A, Gorissen D, De Turck F, Dhaene T (2012) Blind Kriging: Implementation and performance analysis. Adv. Eng. Softw. 49, 1-13. https://doi.org/10.1016/j.advengsoft.2012.03.002

[64] Du D, He E, Li F, Huang D (2020) Using the hierarchical Kriging model to optimize the structural dynamics of rocket engines. Aerosp. Sci. Technol. 107, 106248. https://doi.org/10.1016/j.ast.2020.106248

[65] Bu Y, Song W, Han Z, Zhng Y, Zhang L (2020). Aerodynamic/aeroacoustic variable-fidelity optimization of helicopter rotor based on hierarchical kriging model. Chinese J. Aeronaut. 33(2), 476-492. https://doi.org/10.1016/j.cja.2019.09.019

[66] Han Z, Görtz S, Zimmermann R (2013) Improving variable-fidelity surrogate modeling via gradient-enhanced kriging and a generalized hybrid bridge function. Aerosp. Sci. Technol. 25(1), 177-189.

[67] Ulaganathan S, Couckuyt I, Dhaene T, Degroote J, Laermans E (2016) Performance study of gradient-enhanced Kriging. Eng. Comput. 32(1), 15-34. https://doi.org/10.1007/s00366-015-0397-y

[68] Mukhopadhyay T, Chakraborty S, Dey S, Adhikari S, Chowdhury R (2017) A critical assessment of Kriging model variants for high-fidelity uncertainty quantification in dynamics of composite shells. Arch. Comput. Methods Eng. 24(3), 495-518. https://doi.org/10.1007/s11831-016-9178-z

[69] Stein M L (1999) Interpolation of spatial data: some theory for kriging. Springer Science & Business Media.

[70] Xu S, Chen H, Zhang J (2019) A study of Nash-EGO algorithm for aerodynamic shape design optimizations. Struct. Multidiscipl. Optim. 59(4), 1241-1254. https://doi.org/10.1007/s00158-018-2126-9

[71] Janusevskis J, Riche R L, Ginsbourger D, Girdziusas R (2012) Expected improvements for the asynchronous parallel global optimization of expensive functions: Potentials and challenges. In： International Conference on Learning and Intelligent Optimization pp. 413-418. Springer, Berlin, Heidelberg.

[72] Chaudhuri A, Haftka R T (2013) A stopping criterion for surrogate based optimization using ego. In： 10th world congress on structural and multidisciplinary optimization pp. 20-24. Springer.

[73] Zhan D, Xing H (2020) Expected improvement for expensive optimization: a review. J. Glob. Optim. 78(3), 507-

544. https://doi.org/10.1007/s10898-020-00923-x

[74] Islam S M, Das S, Ghosh S, Roy S, Suganthan P N (2011) An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. IEEE Trans. Syst., Man, Cybern. B. Cybern. 42(2): 482-500. https://doi.org/10.1109/TSMCB.2011.2167966

[75] Toal D J, Bressloff N W, Keane A J, Holden C M E (2011) The development of a hybridized particle swarm for kriging hyperparameter tuning. Eng. Optim. 43(6): 675-699. https://doi.org/10.1080/0305215X.2010.508524

[76] Kakde M R O (2004) Survey on multiobjective evolutionary and real coded genetic algorithms. In: Proceedings of the 8th Asia Pacific Symposium on Intelligent and Evolutionary Systems, pp. 150-161.

[77] Van den Bergh F, Engelbrecht A P (2006) A study of particle swarm optimization particle trajectories. Inf. Sci. 176(8): 937-971. https://doi.org/10.1016/j.ins.2005.02.003

[78] Tanabe R, Fukunaga A (2013) Evaluating the performance of SHADE on CEC 2013 benchmark problems. In: 2013 IEEE Congress on Evolutionary Computation (CEC), pp 1952-1959. IEEE.

[79] Mohamed A W, Mohamed A K (2019) Adaptive guided differential evolution algorithm with novel mutation for numerical optimization. Int. J. Mach. Learn Cybern. 10(2): 253-277. https://doi.org/10.1007/s13042-017-0711-7

[80] Yang S, Sato Y (2016) Modified bare bones particle swarm optimization with differential evolution for large scale problem. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp 2760-2767. IEEE.

[81] Wang H, Rahnamayan S, Sun H, Omran M G (2013) Gaussian bare-bones differential evolution. IEEE Trans. Cybern. 43(2): 634-647. https://doi.org/10.1109/TSMCB.2012.2213808

[82] Nielsen H B (2002) Design and analysis of computer experiments (DACE)-a matlab kriging toolbox v2.0 (Version 2.0).

[83] Couckuyt I, Dhaene T, Demeester P (2014) ooDACE toolbox: A flexible object-oriented Kriging implementation. J. Mach. Learn. Res. 15: 3183-3186.

[84] Murphy B (2014) PyKrige: development of a kriging toolkit for Python. In: AGU fall meeting abstracts, San Francisco, CA: AGU press.

[85] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L. et al. (2019) “Pytorch: An imperative style, high-performance deep learning library,” Adv. Neural Inf. Process. Syst. pp. 8024–8035.

[86] Eriksson D, Bindel D, Shoemaker C (2019) pySOT and POAP: An event-driven asynchronous framework for surrogate optimization. arXiv, Preprint 1908.00420, [Online]. Available: https://arxiv.org/pdf/1908.00420

[87] Joe S, Kuo F Y (2008) Constructing Sobol sequences with better two-dimensional projections. SIAM J on Sci Comput. 30(5), 2635-2654. https://doi.org/10.1137/070709359

[88] Ollar J, Mortished C, Jones R, Sienz J, Toropov V (2017) Gradient based hyper-parameter optimisation for well conditioned kriging metamodels. Struct. Multidiscipl. Optim. 55(6), 2029-2044.

[89] Xiao S, Rotaru M, Sykulski J K (2013) Adaptive weighted expected improvement with rewards approach in kriging assisted electromagnetic design. IEEE Trans Magn 49(5): 2057-2060. https://doi.org/10.1109/TMAG.2013.2240662

[90] Yang K, Emmerich M, Deutz A, Bäck T (2019) Efficient computation of expected hypervolume improvement using box decomposition algorithms. J. Glob. Optim. 75(1), 3-34. https://doi.org/10.1007/s10898-019-00798-7

[91] Zhan D, Qian J, Liu J, Cheng Y (2017) Pseudo expected improvement matrix criteria for parallel expensive multi-objective optimization, In: World Congress of Structural and Multidisciplinary Optimization, pp. 175-190. Springer, Cham. https://doi.org/10.1007/978-3-319-67988-4_12

[92] Parr J M, Keane A J, Forrester A I, Holden C M (2012) Infill sampling criteria for surrogate-based optimization with constraint handling. Eng. Optim. 44(10), 1147-1166. https://doi.org/10.1080/0305215X.2011.637556

[93] Saltelli A, Annoni P, Azzini I, Campolongo F, Ratto M, Tarantola S (2010) Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index. Comput Phys Commun 181(2): 259-270. https://doi.org/10.1016/j.cpc.2009.09.018

[94] Hollander M, Wolfe D A (1973) Nonparametric Statistics. J. Wiley, New York.

[95] Laguna M, Marti R (2005) Experimental testing of advanced scatter search designs for global optimization of multimodal functions. J. of Glob. Optim. 33(2), 235-255. https://doi.org/10.1007/s10898-004-1936-z

[96] Mohamed A W, Suganthan P N (2018) Real-parameter unconstrained optimization based on enhanced fitness-adaptive differential evolution algorithm with novel mutation. Soft Comput 22(10): 3215-3235. https://doi.org/10.1007/s00500-017-2777-2

[97] Mohamed A W, Sabry H Z (2012) Constrained optimization based on modified differential evolution algorithm, Inf. Sci. 194, 171–208. https://doi.org/10.1016/j.ins.2012.01.008

[98] Huang D, Allen T T, Notz W I, Zeng N (2006) Global optimization of stochastic black-box systems via sequential

kriging meta-models. J Glob Optim 34(3): 441-466. https://doi.org/10.1007/s10898-005-2454-3

[99] Liang J J, Qu B Y, Suganthan P N, Hernández-Díaz A G (2013) Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report 201212(34): 281-295

[100] Sengupta S, Basak S, Peters R A (2019) Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives. Mach. Learn. Knowl. Extr. 1(1): 157-191. https://doi.org/10.3390/make1010010

[101] Heft A I, Indinger T, Adams N A (2012) Introduction of a new realistic generic car model for aerodynamic investigations. SAE Technical Paper 2012-01-0168. https://doi.org/10.4271/2012-01-0168

[102] Ahmed S R, Ramm G, Faltin G (1984) Some salient features of the time-averaged ground vehicle wake. SAE Technical Paper 840300. https://doi.org/10.4271/840300

[103] Wickern G (2001) On the application of classical wind tunnel corrections for automotive bodies. SAE Technical Paper 2001-01-0633. https://doi.org/10.4271/2001-01-0633

[104] Soares R F, Garry K F, Holt J (2017) Comparison of the far-field aerodynamic wake development for three driver model configurations using a cost-effective RANS simulation. SAE Technical Paper 2017-01-1514. https://doi.org/10.4271/2017-01-1514

[105] Strangfeld C, Wieser D, Schmidt H J, Woszidlo R, Nayeri C, Paschereit C (2013). Experimental study of baseline flow characteristics for the realistic car model driver. SAE Technical Paper 2013-01-1251. https://doi.org/10.4271/2013-01-1251

[106] Yusuf S N A, Asako Y, Sidik N A C, Mohamed S B, Japar W M A A (2020) A short review on rans turbulence models. CFD Letters 12(11), 83-96. https://doi.org/10.37934/cfdl.12.11.8396

[107] Menter F R, Kuntz M, Langtry R (2003) Ten years of industrial experience with the SST turbulence model. Turbulence, heat and mass transfer, 4(1), 625-632.

[108] Ashton N, West A, Lardeau S, Revell A (2016) Assessment of RANS and DES methods for realistic automotive models. Comput. Fluids 128: 1-15. https://doi.org/10.1016/j.compfluid.2016.01.008

[109] Shinde G, Joshi A, Nikam K (2013) Numerical investigation of the drivAer car model using opensource CFD

solver openFOAM. In: Tata Consultancy Services, pp. 1-12. Pune, India

[110] Shaharuddin N H, Ali M S M, Mansor S, Muhamad S, Salim, S A Z S, Usman M (2017) Flow simulations of generic vehicle model SAE type 4 and DrivAer Fastback using OpenFOAM. J Adv Res Fluid Mech Therm Sci 37(1): 18-31.

[111] Ashton N, Revell A (2015) Comparison of RANS and DES methods for the DrivAer automotive body. SAE Technical paper 2015-01-1538. https://doi.org/10.4271/2015-01-1538

[112] Shimizu K, Nakashima T, Hiraoka T, Nakamura Y, Nouzawa T (2018) Investigation of increase in aerodynamic drag caused by a passing vehicle. SAE Technical Paper 2018-01-0719. https://doi.org/10.4271/2018-01-0719

[113] Beigmoradi S, Vahdati M (2021) Multi-objective optimization of a hatchback rear end utilizing fractional factorial design algorithm. Eng Comput 37(1): 139-153. https://doi.org/10.1007/s00366-019-00813-1

[114] Munoz-Paniagua J, García J (2020). Aerodynamic drag optimization of a high-speed train. J. Wind Eng. Ind. Aerodyn. 204: 104215. https://doi.org/10.1016/j.jweia.2020.104215

[115] Zhang X, Trame M N, Lesko L J, Schmidt S (2015) Sobol sensitivity analysis: a tool to guide the development and evaluation of systems pharmacology models. CPT: pharmacometrics & systems pharmacology 4(2): 69-79. https://doi.org/10.1002/psp4.6

[116] Wu L L, Fu Y, Bu X B, Li X R, Ma X L (2020) Research on automatic aerodynamic optimization for a SUV based on RBF model. J. Phys. Conf. Ser. 1550(4), 042025.