An efficient space-division-based width optimization method for RBF network using fuzzy clustering algorithms

Yunwei Zhang[1], Chunlin Gong[1] ✉, Hai Fang[1], Hua Su[1], Chunna Li[1], Andrea Da Ronch[2]

Abstract The Radial Basis Function (RBF) network is one of the most widely used surrogate models in multidisciplinary design optimization. However, one of the challenges of applying the RBF network to engineering problems is how to efficiently optimize its width parameters. In this work, a novel space-division-based width optimization (SDWO) method is proposed to decompose the complex multivariable width optimization into several small-scale single-variable width optimizations. The SDWO method consists of two main steps. First, a two-stage fuzzy clustering algorithm is carried out to group the samples and divide the input space into several overlapping local regions, and the overlapping degree is controlled by a dimensionless expansion factor. Second, in each local region, one small-scale local RBF network (LRBFN) is constructed by solving a single-variable optimization problem when the LRBFN is needed for prediction. All these LRBFNs are independent of each other and can be constructed in parallel. The proposed method is efficient and particularly suitable for large sample sets. Test results of four sample sets verify that the proposed SDWO method has better performance than the conventional width optimization method in terms of both training efficiency and approximation accuracy. An inter-stage structure optimization is carried out, which demonstrates the efficiency of the proposed method in practical engineering applications.

Keywords: width optimization, Radial Basis Function network, space division, fuzzy clustering, surrogate model

1 Introduction

In the multidisciplinary design optimization (MDO) of complex systems, such as launch vehicles and aircrafts, a large number of computer simulations are involved. High fidelity simulations, like computational structural dynamics (CSD) and computational fluid dynamics (CFD), take a long computing time and are too expensive to be directly carried out for global optimization. Surrogate models can be used as a replacement of high-fidelity simulation models to make a trade-off between accuracy and efficiency.

The Radial Basis Function (RBF) network is one of the most promising surrogate models for global optimization (Haftka et al. 2016; Akhtar and Shoemaker 2016). It approximates an unknown complicated function by a weighted sum of several basis functions directly from the input-output dataset (Sobester et al. 2004; Huang et al. 2005) and has been widely used in a large

✉ Corresponding author: Chunlin Gong
leonwood@nwpu.edu.cn

1 Shaanxi Aerospace Flight Vehicle Design Key Laboratory, School of Astronautics, Northwestern Polytechnical University, Xi'an 710072, China

2 University of Southampton, Southampton, England, SO17 1BJ, United Kingdom

number of practical problems (Alexandridis et al. 2013; Li et al. 2017; Fang et al. 2018; Shi et al. 2018). The training process of the RBF network is to estimate the optimal values of three types of parameters: the RBF centres, the width parameters, and the output weights. The output weights can be analytically calculated by solving the linear regression equations after the RBF centres and the widths being determined. Therefore, the RBF centres and their corresponding widths are key factors affecting the model accuracy. A large amount of work has been carried out to determine these two types of parameters.

First, consider the methods for determining the RBF centres. In the early studies of the RBF network (Broomhead and Lowe 1988; Poggio and Girosi 1990), the RBF centres were allocated at each training sample under the interpolation condition. Such treatment can lead to prohibitive computational costs and overfitting when the number of the training samples is large or the numerical noise exists. A solution (Lowe 1989) was proposed, which selects the RBF centres randomly from the training samples. However, this method does not consider the distribution density of the sample data in the input space, so it is unable to achieve satisfactory generalization performance. A significant improvement has been made by developing a series of algorithms based on a sequential strategy, e.g., the Orthogonal Least Squares (OLS) Algorithm (Chen et al. 1991), Resource Allocating Networks (RAN) (Platt 1991), Growing and Pruning RBF (GAP-RBF) networks (Huang et al. 2004), and other variations (Huang et al. 2005; Bortman and Aladjem 2009; Vuković and Miljković 2013). These methods construct the hidden layer by collecting the RBF centres sequentially and incrementally until a termination criterion is met. Simulation results indicate that these methods can generate compact networks, but their iterative procedures are time-consuming. Different from these sequential methods, some methods based on clustering algorithm (Hansen and Jaumard 1997) can simultaneously determine all the RBF centres. Carvalho (Carvalho and Brizzotti 2001) reviewed different clustering algorithms for determining the RBF centres. Oh (Oh et al. 2012) proposed an advanced architecture of *k*-means clustering-based polynomial RBF networks. Niros (Niros et al. 2015) proposed a new method that combines hierarchical fuzzy clustering and particle swarm optimization to efficiently design RBF networks. These clustering-based methods put the RBF centres at the centroids of the clusters such that the RBF centres spread throughout the entire input space according to the distribution of the sample data.

As far as the determination of width parameters of the RBF network is concerned, the published methods in the literature can be summarized into three categories: 1) all widths are set to the same value; 2) all widths are set to different values; and 3) the widths are partially set to the same value.

In the first category, all RBF centres are assumed to have the same width. Park and Sandberg (Park and Sandberg 1991) assume the single width parameter to be the average Euclidean distance between the RBF centres. Orr (Orr 1998) determines the width by optimization. These methods are easy to implement. However, they are inadequate for most practical problems due to flexibility limitations.

The second category of methods for determining the widths is much more flexible. It assigns a specific width to each RBF centre, and considers all the widths as independent variables. Leonard and Kramer (Leonard and Kramer 1991) proposed a *p*-nearest-neighbour (*p*-NN) heuristic method, which defines the width of each RBF centre as the weighted average Euclidean distance between the RBF centre and its *p* nearest neighbours. Huang (Huang et al. 2006, 2011) proposed the Extreme Learning Machine (ELM), which defines all the widths of the hidden layer as random variables. These methods, based on unsupervised learning, are very efficient but still show difficulty in guaranteeing accuracy (Peng et al. 2006). Other methods based on optimization algorithms for determining the widths are also considered, such as gradient descent method (Neruda and Kudová 2005), genetic algorithms (GA) (Sheta and De Jong 2001; Harpham et al.

2004; Neruda and Kudová 2005), differential evolution (DE) (Liu and Lampinen 2005), etc. However, these methods are extremely computationally expensive, and are slow to converge due to excessive design variables.

The third category of methods for determining the widths makes a compromise between the abovementioned two categories. It divides the RBF centres into different groups, and assumes the RBF centres belonging to the same group have the same width (Verleysenl and Hlavackova 1996; Benoudjit et al. 2002; Benoudjit and Verleysen 2003; Yao et al. 2012). With this assumption, the number of design variables of the width optimization problem is greatly reduced, which is equal to the number of the groups. These methods achieve a trade-off between model flexibility and training efficiency. Despite the reduction of the number of design variables, these methods still lead to multivariable width optimizations, which converge much more slowly in comparison with single-variable optimizations. Hence, these methods are still unsatisfactory in practical engineering problems, especially dealing with large sample sets.

This paper aims to tackle the computationally expensive multivariable width optimization problem of the RBF network. A space-division-based width optimization (SDWO) method using fuzzy clustering technique is proposed to reduce the number of design variables to one without sacrificing model flexibility and approximation accuracy. Different from the hierarchical fuzzy clustering method in Niros's work (Niros et al. 2015), in which one global RBF network is constructed, multiple small-scale local RBF networks (LRBFNs) are created within different local regions of the input space in our work. Recently, Smolik and Skala (Smolik and Skala 2018) proposed a new space-division approach for RBF interpolation of large scattered datasets. They use rectangular grids and a distance parameter to divide the input space into several overlapping cells. However, this space division method based on rectangular grids is only suitable for uniformly distributed samples. In contrast, the space division method based on a two-stage fuzzy clustering algorithm with an expansion factor in this paper is suitable for arbitrary distributed samples.

The remainder of the article is organized as follows. Section 2 briefly describes the RBF network and the fuzzy *k*-means clustering algorithm. In section 3, the space-division-based width optimization method is introduced in detail. Section 4 presents the validation and application of the proposed method. Finally, the conclusions are summarized.

2 The RBF network and the fuzzy *k*-means clustering algorithm

2.1 The RBF network

The RBF network is a single-hidden layer feedforward network. For an arbitrary *n*-dimensional input vector $\mathbf{x} = (x_1, x_2, ..., x_n)^T$, the total response $\hat{f}(\mathbf{x})$ of an RBF network with N_C RBF centres is a linear combination of the basis functions stated as

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{N_C} w_i \phi_i(\mathbf{x}) \tag{1}$$

where $\phi_i(\mathbf{x})$ is the *i*-th basis function with an input vector $\mathbf{x} \in R^n$, and w_i is the corresponding weight. Without loss of generality, the output dimension of the RBF network is assumed to be one throughout this paper. Different types of functions can be used as basis functions. The exponentially decaying Gaussian function is one of the most popular basis functions. Its expression is defined as

$$\phi_i(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{\sigma_i^2}\right) \tag{2}$$

where $\mathbf{c}_i$ and σ_i are the centre and the width of the i-th ($i=1,2,...,N_C$) RBF, respectively. $\|\bullet\|$ denotes the L^2-norm of the underlying vector.

Assume the training sample set is $T=\{(\mathbf{x}_i,y_i)\,|\,y_i=f(\mathbf{x}_i),i=1,2,...,N_T\}$, where $f:R^n\to R$ is the accurate function, and N_T is the number of training samples. The purpose of training an RBF network is to determine the values for the parameters $\mathbf{c}_i$, σ_i, and $w_i(i=1,2,...,N_C)$ in Eq.(1) to best fit the accurate function. The locations of RBF centres $\mathbf{c}_i(i=1,2,...,N_C)$ can be determined by clustering algorithms, which will be discussed in the next subsection. The width parameters $\sigma_i(i=1,2,...,N_C)$ are determined by multi-variable optimization, which is computationally expensive. In order to obtain the output weights $w_i(i=1,2,...,N_C)$, substitute the training samples into Eq.(1), then the resulting equation system can be expressed in the matrix form as

$$\begin{aligned}\mathbf{y}&=\mathbf{\Phi w}\\ \mathbf{y}&=[y_1,y_2,...,y_{N_T}]^T\\ \mathbf{w}&=[w_1,w_2,...,w_{N_C}]^T\\ \mathbf{\Phi}&=\begin{bmatrix}\phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \cdots & \phi_{N_C}(\mathbf{x}_1)\\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \cdots & \phi_{N_C}(\mathbf{x}_2)\\ \vdots & \vdots & \ddots & \vdots\\ \phi_1(\mathbf{x}_{N_T}) & \phi_2(\mathbf{x}_{N_T}) & \cdots & \phi_{N_C}(\mathbf{x}_{N_T})\end{bmatrix}_{N_T\times N_C}\end{aligned} \tag{3}$$

where $\mathbf{\Phi}$ is a $N_T\times N_C$ matrix, and generally, $N_C\ll N_T$. The weight vector $\mathbf{w}$ can be analytically calculated by Least Square Method (LSM):

$$\mathbf{w}=\mathbf{\Phi}^{\dagger}\mathbf{y} \tag{4}$$

where $\mathbf{\Phi}^{\dagger}$ is the pseudo-inverse of $\mathbf{\Phi}$.

The key bottleneck of training an RBF network is how to efficiently optimize the width parameters $\sigma_i(i=1,2,...,N_C)$. As summarized in section 1, one of the most promising methods in the literature is to divide the RBF centres $\{\mathbf{c}_i|i=1,2,...,N_C\}$ into $N_L(N_L\ll N_C)$ non-overlapping groups $C^{(j)}(j=1,2,...,N_L)$. The RBF centres within the same group have the same width, as stated in Eq.(5).

$$\begin{aligned}&\bigcup_{j=1}^{N_L}C^{(j)}=\{\mathbf{c}_i\,|\,i=1,...,N_C\}\\ &C^{(j)}\cap C^{(k)}=\varnothing\ (\forall j,k=1,2,...,N_L,\ \text{and}\ j\neq k)\\ &\forall\mathbf{c}_i\in C^{(j)},\sigma_i=\sigma^{(j)}\ (j=1,...,N_L)\end{aligned} \tag{5}$$

where $\sigma^{(j)}$ is the width of the RBF centres within the j-th group $C^{(j)}$. Given a validation sample set $V=\{(\mathbf{z}_i,y_i)\,|\,y_i=f(\mathbf{z}_i),\ i=1,2,...,N_V\}$, where N_V is the number of validation samples, the mathematical model of the width optimization problem can be expressed as

$$\begin{aligned}\text{find}\quad & \sigma^{(1)},\sigma^{(2)},...,\sigma^{(N_L)}\\ \min\quad & \frac{1}{2}\sum_{j=1}^{N_V}\left\|y_j-\hat{f}(\mathbf{z}_j)\right\|^2\\ \text{s.t.}\quad & \hat{f}(\mathbf{z})=\sum_{i=1}^{N_C}w_i\exp\left(-\frac{\|\mathbf{z}-\mathbf{c}_i\|^2}{\sigma_i^2}\right)\end{aligned} \tag{6}$$

For the convenience of discussion, this method is referred to as the conventional width optimization (CWO) method in the rest of this paper. The CWO method reduces the number of design variables from N_C to N_L. However, there are two key issues of CWO. The first one is that it still leads to a multi-variable optimization, which may need excessive iterations to converge; and the second one is that it is computationally prohibitive to calculate pseudo-inverse of the

large-scale $\mathbf{\Phi}$, especially when dealing with large sample sets. The SDWO method is proposed mainly for addressing these two issues.

2.2 The fuzzy *k*-means clustering algorithm

Clustering algorithms employ unsupervised learning to find natural data groups in a non-classified dataset (Hansen and Jaumard 1997). The *k*-means clustering algorithm is one of the most widely used clustering techniques in literatures (Harpham et al. 2004; Oh et al. 2012; Yao et al. 2012). The standard *k*-means clustering algorithm is used to find a set of k centroids $\mathbf{c}_j (j = 1, 2, ..., k)$ which minimize the sum of the squared Euclidean distance D between the N sample points $\mathbf{x}_i (i = 1, 2, ..., N)$ and its nearest centroid:

$$\begin{aligned} D &= \sum_{i=1}^{N} \sum_{j=1}^{k} (u_{ij} \bullet \left\| \mathbf{x}_i - \mathbf{c}_j \right\|^2) \\ \mathbf{U} &= \left\{ u_{ij} \right\}_{1 \le i \le N, 1 \le j \le k} \end{aligned} \tag{7}$$

where $\mathbf{U}$ is a $N \times k$ membership matrix. The element u_{ij} of $\mathbf{U}$ can only be 1 or 0, which identifies if the *i*-th sample belongs to the *j*-th cluster or not. u_{ij} is calculated by:

$$u_{ij} = \begin{cases} 1 & j = \arg \min_{m=1}^{k} (\| \mathbf{x}_i - \mathbf{c}_m \|) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Since there can only be one "1" in each row of $\mathbf{U}$, the sum of the elements in each row of $\mathbf{U}$ is 1, as stated in Eq.(9).

$$\sum_{j=1}^{k} u_{ij} = 1 \ (i = 1, 2, ..., N) \tag{9}$$

The following iterative algorithm can be used to minimize Eq.(7):

Step 1: Determine the number of clusters *k*, and randomly generate a set of initial cluster centroids $\mathbf{c}_j^{(0)}$ $(j = 1, 2, ..., k)$, where $\mathbf{c}_j^{(0)} \ne \mathbf{x}_i (\forall i = 1, 2, ..., N)$. Set loop variable $r = 1$.

Step 2: Update the cluster membership matrix $\mathbf{U}^{(r)}$ using Eq.(8).

Step 3: Update cluster centroids $\mathbf{c}_j^{(r)} (j = 1, 2, ..., k)$ by:

$$\mathbf{c}_j^{(r)} = \frac{\sum_{i=1}^{N} (u_{ij}^{(r)} \cdot \mathbf{x}_i)}{n_j^{(r)}} \tag{10}$$

where $n_j^{(r)}$ is the number of points which belong to the *j*-th cluster.

Step 4: Check the termination criterion using Eq.(11).

$$\sum_{j=1}^{k} \left\| \mathbf{c}_j^{(r)} - \mathbf{c}_j^{(r-1)} \right\| < \varepsilon \tag{11}$$

where ε is a positive tolerance. Set $r = r + 1$ and go to step 2 until Eq.(11) is satisfied.

The clustering result of the *k*-means algorithm are also called "hard" clusters, since any sample point either is or is not a member of a particular cluster. The fuzzy *k*-means clustering algorithm, however, allows "soft" or "fuzzy" partition, which means that a sample point has a degree of membership in each cluster (Bezdek 1981). The calculation procedure of the fuzzy *k*-means algorithm is similar to that of the *k*-means algorithm except for the determination of $u_{ij} (1 \le i \le N, 1 \le j \le k)$:

$$u_{ij} = \left(\sum_{m=1}^{k} \left(\frac{\| \mathbf{x}_i - \mathbf{c}_j \|}{\| \mathbf{x}_i - \mathbf{c}_m \|} \right)^{2/(\varphi - 1)} \right)^{-1} \tag{12}$$

where $u_{ij} (0 < u_{ij} < 1)$ represents the degree or possibility that $\mathbf{x}_i$ belongs to the *j*-th cluster, and

$\varphi > 1$ is a constant that determines the degree of fuzziness. As φ increases, the degree of fuzziness increases accordingly, and specifically, when $\varphi \to 1$, the algorithm becomes the standard *k*-means clustering algorithm (Bezdek 1981; Stetco et al. 2015). Generally, $1.5 \le \varphi \le 3.0$ is suitable for most data (Bezdek et al. 1984). In this paper, $\varphi = 2$ is used as suggested in (Chuang et al. 2006).

Eq.(12) can also be written as:

$$u_{ij} = \frac{1}{\sum_{m=1}^{k} \left(\frac{\| \mathbf{x}_i - \mathbf{c}_j \|^{2/(\varphi-1)}}{\| \mathbf{x}_i - \mathbf{c}_m \|^{2/(\varphi-1)}} \right)} = \frac{1}{\| \mathbf{x}_i - \mathbf{c}_j \|^{2/(\varphi-1)} \times \sum_{m=1}^{k} \left(\frac{1}{\| \mathbf{x}_i - \mathbf{c}_m \|^{2/(\varphi-1)}} \right)} \tag{13}$$

Moreover,

$$\begin{aligned} \sum_{j=1}^{k} u_{ij} &= \sum_{j=1}^{k} \left(\frac{1}{\| \mathbf{x}_i - \mathbf{c}_j \|^{2/(\varphi-1)}} \times \frac{1}{\sum_{m=1}^{k} \left(\frac{1}{\| \mathbf{x}_i - \mathbf{c}_m \|^{2/(\varphi-1)}} \right)} \right) \\ &= \sum_{j=1}^{k} \left(\frac{1}{\| \mathbf{x}_i - \mathbf{c}_j \|^{2/(\varphi-1)}} \right) \times \frac{1}{\sum_{m=1}^{k} \left(\frac{1}{\| \mathbf{x}_i - \mathbf{c}_m \|^{2/(\varphi-1)}} \right)} = 1 \end{aligned} \tag{14}$$

Hence, similar to the *k*-means clustering algorithm, the membership matrix $\mathbf{U}$ of the fuzzy *k*-means clustering algorithm still satisfies the condition in Eq.(9). The "soft" partition property of the fuzzy *k*-means clustering algorithm is used in the SDWO method to divide the input space into overlapping local regions.

3 The proposed SDWO method

Based on the divide and conquer (D&C) strategy, the proposed SDWO method uses space division to decompose the complex multivariable width optimization into several small-scale single-variable width optimizations. The detailed process is presented as follows.

3.1 The two-stage fuzzy clustering algorithm with an expansion factor for space division

Clustering algorithms have been widely used for training the RBF network (Harpham et al. 2004; Oh et al. 2012; Yao et al. 2012; Alexandridis et al. 2013). Different from these methods which only perform the clustering algorithm once, SDWO takes a two-stage clustering process to partition the input space. The clustering algorithms used in the two stages and their purposes are different. The schematic diagram of the proposed two-stage fuzzy clustering algorithm with an expansion factor is shown in Fig. 1.

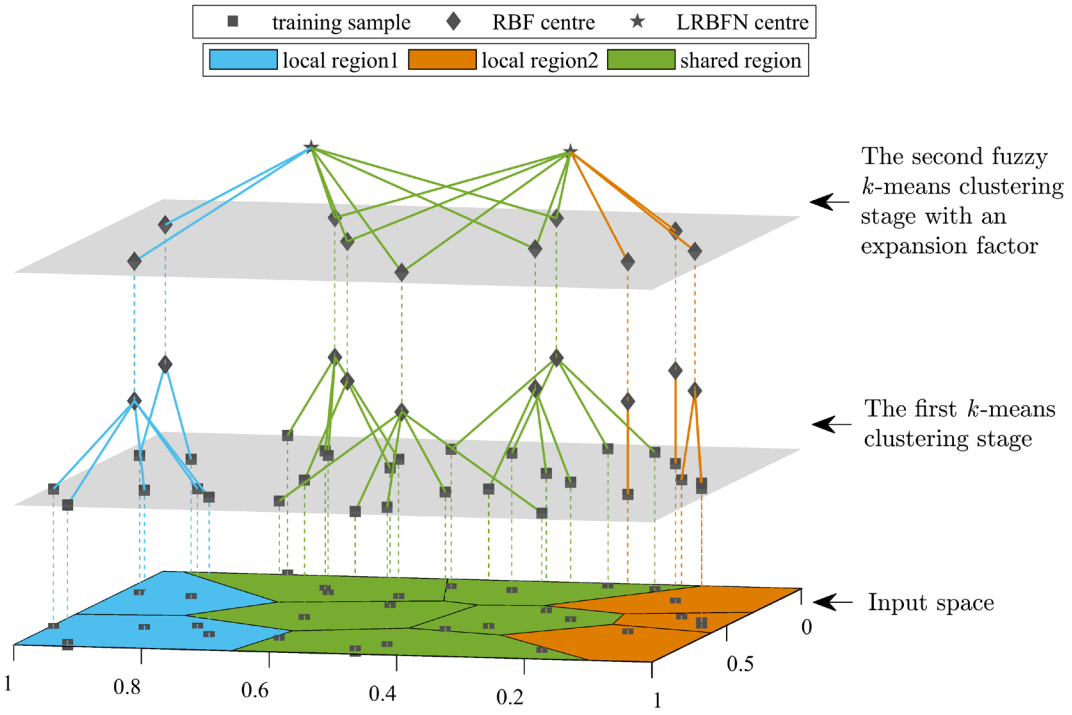


Fig. 1 Schematic diagram of the proposed two-stage fuzzy clustering algorithm with an expansion factor

In the first clustering stage, the input components $\{\mathbf{x}_i | \mathbf{x}_i \in R^n, i = 1,2,...,N_T\}$ of the available training samples $T = \{(\mathbf{x}_i, y_i) \mid y_i = f(\mathbf{x}_i), i = 1,2,...,N_T\}$ are cluster-analysed with the *k*-means clustering algorithm. The obtained cluster centroids $C = \{\mathbf{c}_i \mid \mathbf{c}_i \in R^n, i = 1,2,...,N_C\}$ are used as RBF centres.

In the second clustering stage, the resulting RBF centres from the first stage are cluster-analysed again with the fuzzy *k*-means clustering algorithm, and are divided into N_L clusters $C^{(j)} (1 \le j \le N_L, C^{(j)} \subset C)$.

It should be noted that, in the original fuzzy *k*-means clustering algorithm, one RBF centre $\mathbf{c}_i (i = 1,2,...,N_C)$ is allocated to the cluster which has the maximum membership degree for $\mathbf{c}_i$, as stated in Eq.(15).

$$m = \arg\max_{j=1}^{N_L}(u_{ij}) \Rightarrow \mathbf{c}_i \in C^{(m)} \quad 1 \le i \le N_C \tag{15}$$

However, in this case, one RBF centre can only belong to one cluster. Unlike the CWO method, the second clustering stage of SDWO divides all the RBF centres into multiple overlapping clusters (The reason for overlap is explained in the next subsection). For this purpose, a new rule is used to augment all the clusters, as shown in Eq.(16).

$$\left.\begin{array}{l} m = \arg\max_{j=1}^{N_L}(u_{ij}) \\ \text{or} \quad u_{im} \ge \dfrac{1-\lambda^{1/N_L}}{2} \end{array}\right\} \Rightarrow \mathbf{c}_i \in C^{(m)} \tag{16}$$

where λ is an expansion factor, and $\lambda \in [0,1]$.

This newly introduced rule consists of two conditions. If $u_{im} (i = 1,2,...,N_C,\ m = 1,2,...,N_L)$ satisfies either of the two conditions, $\mathbf{c}_i$ will be allocated to the cluster $C^{(m)}$. The first condition

is the same as Eq.(15). The second condition can add extra RBF centres, whose membership degrees are larger than a threshold, to cluster $C^{(m)}$.

The threshold $(1-\lambda^{1/N_L})/2$ is a function of the expansion factor λ. When λ=0, $(1-\lambda^{1/N_L})/2=0.5$. Since in each row of $\mathbf{U}$, the sum of the elements is equal to 1, as stated in Eq.(9), at most only 1 element in each row of $\mathbf{U}$ can be larger than 0.5. Hence, in this case, one RBF centre only belongs to one cluster, and there is no overlap between different clusters. When λ=1, $(1-\lambda^{1/N_L})/2=0$. Because the membership degree is a positive number, the second condition of Eq.(16) is always satisfied. Therefore, each RBF centre belongs to all RBF centre clusters. In other words, all clusters overlap completely, and each cluster contains all the RBF centres, i.e., $C^{(m)}=\{\mathbf{c}_i \mid \mathbf{c}_i \in R^n, i=1,...,N_C\}$ $(1 \le m \le N_L)$. When λ gradually increases from 0 to 1, each cluster contains more RBF centres, and more RBF centres are shared by different clusters. Fig. 2 depicts the curves of the threshold with respect to λ using different N_L. It can be noticed from Fig. 2 that N_L does not affect the threshold when λ=0 or λ=1, while it only affects the decreasing trend of the threshold. The larger N_L is, the smaller the membership degree that one RBF centre belongs to each cluster will be. As a result, the threshold should decrease rapidly such that the expansion factor λ can smoothly control the overlapping degree of the clusters. Hence, the variable N_L in the threshold $(1-\lambda^{1/N_L})/2$ is employed to make the threshold adaptive to the specific number of clusters of RBF centres.

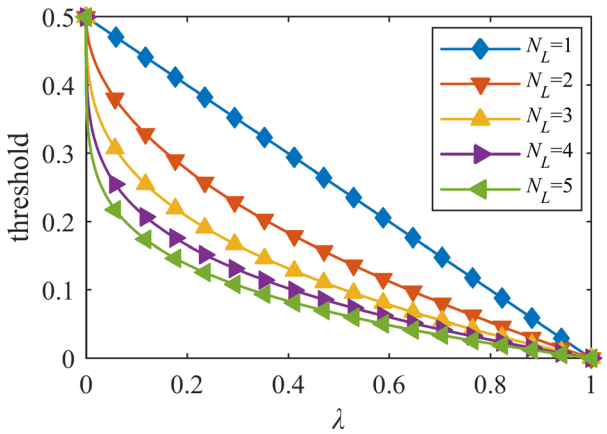


Fig. 2 The curves of the threshold with respect to λ using different N_L

After grouping the RBF centres, the Voronoi diagram (Aurenhammer 1991; Watson 1993) is adopted to divide the input space. Given a set of RBF centres, the Voronoi diagram partitions the input space into multiple local regions based on the nearest neighbor principle. For each RBF centre, there is a corresponding Voronoi cell consisting of all points that are closer to that RBF centre than to any other RBF centres.

Denote the entire input space of the accurate function as *S*. In the Voronoi diagram generated by the RBF centres $C=\{\mathbf{c}_i \mid \mathbf{c}_i \in R^n, i=1,2,...,N_C\}$, the input space *S* are partitioned into N_C Voronoi cells. All the Voronoi cells corresponding to the RBF centres in the *j*-th cluster $C^{(j)}(j=1,2,...,N_L)$ are merged together to form the *j*-th local region $S^{(j)}(j=1,2,...,N_L)$. Due to the overlap between these clusters, adjacent local regions also overlap each other, and the overlapping degree is controlled by the expansion factor λ. Fig. 3 shows an example of different space division schemes when λ takes different values. In this example, 40 RBF centres are divided into 2 clusters, and the input space is divided into 2 overlapping local regions. It can be noticed from Fig. 3 that, as the expansion factor increases, the number of RBF centres shared by the 2 clusters increases, and so the shared area of the two regions also increases.

After the space division, the training samples $T=\{(\mathbf{x}_i, y_i) \mid y_i=f(\mathbf{x}_i), i=1,2,...,N_T\}$ and validation samples $V=\{(\mathbf{z}_i, y_i) \mid y_i=f(\mathbf{z}_i), i=1,2,...,N_V\}$ are also divided into N_L groups with

respect to which local regions they are located in, as stated in Eq.(17).

$$\begin{aligned} T^{(j)} &= \{(\mathbf{x}, y) \,|\, (\mathbf{x}, y) \in T,\ \mathbf{x} \in S^{(j)}\} \ (j = 1, ..., N_L) \\ &\bigcup_{j=1}^{N_L} T^{(j)} = T \\ V^{(j)} &= \{(\mathbf{z}, y) \,|\, (\mathbf{z}, y) \in V,\ \mathbf{z} \in S^{(j)}\} \ (j = 1, ..., N_L) \\ &\bigcup_{j=1}^{N_L} V^{(j)} = V \end{aligned} \tag{17}$$

where $T^{(j)}$ and $V^{(j)}$ are the subsets of the training and the validation samples of the j-th local region, respectively.

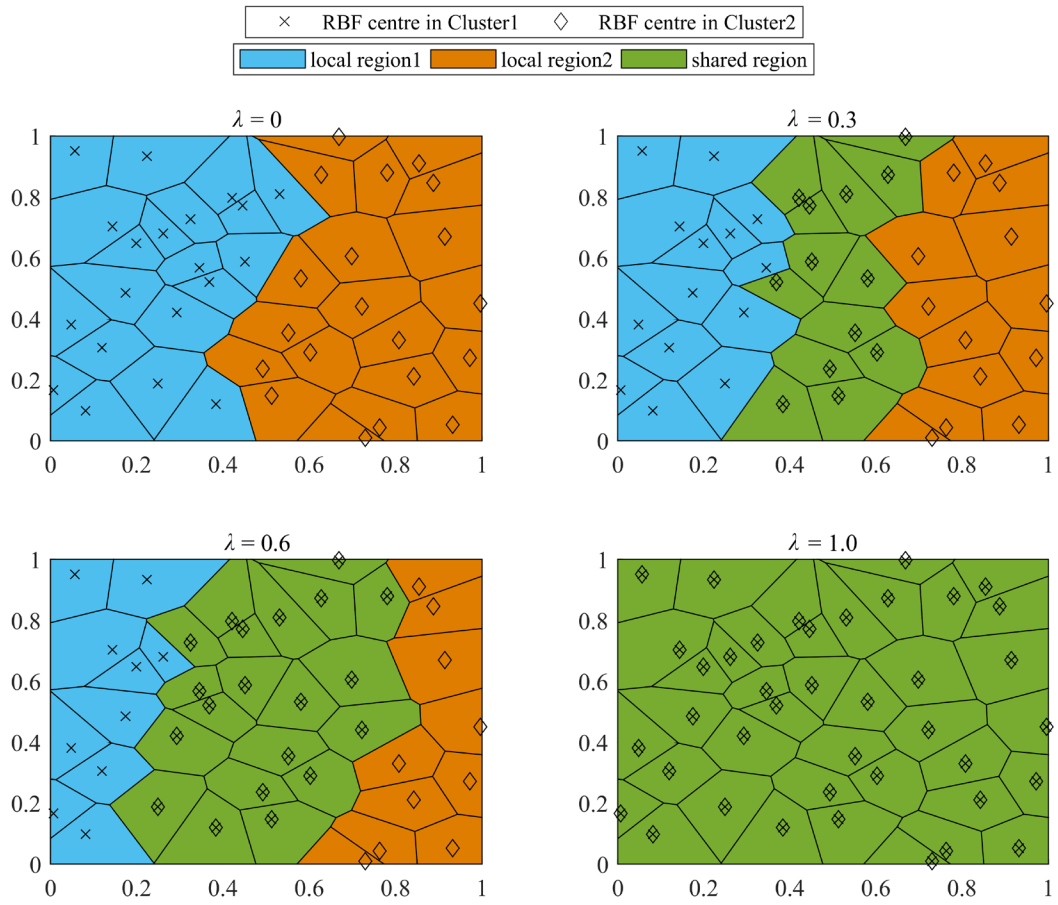


Fig. 3 Different Space division schemes by using different expansion factor λ

3.2 **Train the LRBFNs as needed by solving single-variable optimizations**

After the space division in the previous subsection, the obtained N_L groups of training samples, validation samples, and RBF centres are allocated to N_L width optimizers, and then independent small-scale LRBFNs are constructed as needed in different local regions $S^{(j)}(j = 1, 2, ..., N_L)$. The independence between different LRBFNs is reflected in two aspects: 1) there is no data exchange between the construction processes of different LRBFNs, and all the LRBFNs can be constructed independently and in parallel; 2) after one LRBFN is constructed, it can be independently used for prediction regardless of whether other LRBFNs finish their construction processes or not. The detailed discussions are presented as follows.

Denote the numbers of elements in these subsets $T^{(j)}$, $V^{(j)}$, and $C^{(j)}$ as $N_T{}^{(j)}$, $N_V{}^{(j)}$, and $N_C{}^{(j)}$, respectively, for $j=1,2,...,N_L$. All the RBF centres in $C^{(j)}$ are assumed to have the same width parameter $\sigma^{(j)}$. The function of the *j*-th LRBFN can be expressed as

$$\begin{aligned} \widehat{f}_j(\mathbf{x}) &= \sum_{i=1}^{N_C{}^{(j)}} w_i^{(j)} \exp\left(-\frac{\left\|\mathbf{x}-\mathbf{c}_i\right\|^2}{(\sigma^{(j)})^2}\right) \\ \mathbf{c}_i &\in C^{(j)} \ (i=1,2,...,N_C{}^{(j)}) \end{aligned} \tag{18}$$

The mathematical model of width optimization for constructing the *j*-th LRBFN can be expressed as

$$\begin{aligned} &\text{find} && \sigma^{(j)} \\ &\min && \frac{1}{2}\sum_{i=1}^{N_V{}^{(j)}} \left\|y_i-\widehat{f}_j(\mathbf{z}_i)\right\|^2 \\ &\text{s.t.} && \widehat{f}_j(\mathbf{z}) = \sum_{i=1}^{N_C{}^{(j)}} w_i^{(j)} \exp\left(-\frac{\left\|\mathbf{z}-\mathbf{c}_i\right\|^2}{(\sigma^{(j)})^2}\right) \\ & && \mathbf{c}_i \in C^{(j)} \ (i=1,2,...,N_C{}^{(j)}) \end{aligned} \tag{19}$$

It is obvious that the width optimization in Eq.(19) only involves one design variable. Furthermore, all the N_L width optimizations are independent of each other, and hence they can be carried out in parallel. After the optimal $\sigma^{(j)}(j=1,2,...,N_L)$ are determined, the output weights are calculated with LSM, and finally, the N_L LRBFNs are constructed.

It should be noticed that, in this paper, the LRBFNs are not blended together to create a whole network covering the entire input space. In contrast, the LRBFNs are independent of each other, and only one LRBFN is used to predict the output to an unknown input vector. In this case, if the local regions do not overlap, i.e. $\lambda=0$, the approximation accuracy loss may occur because each LRBFN uses too few samples for training, and sharp discontinuity may exist near the boundary of one LRBFN and its neighbours. Conversely, if the expansion factor is too large, the computational complexity will increase due to the large-scale matrices involved in the width optimizations. Therefore, the expansion factor, which is introduced to make a trade-off between the approximation accuracy, the function continuity, and the computational efficiency, should be carefully chosen. The sensitivity of SDWO to the expansion factor is analysed by numerical experiments in section 4. N_L can be determined according to the number of available CPU cores. N_L should not be too large, otherwise the accuracy of each LRBFN will decrease due to the too small local region and the too few samples for training.

Because of the independence between the LRBFNs, SDWO has an important special advantage that, once one LRBFN completes its own training process, it can be used for prediction, regardless of whether other LRBFNs finish their training processes or not. Hence, the LRBFNs can be constructed as needed, i.e., one LRBFN can delay its training process until it is needed for prediction. The construction of those LRBFNs which are never needed for prediction can be omitted to save the computational time.

3.3 **The procedure of the proposed SDWO method**

The flowchart of the proposed SDWO method is shown in Fig. 4. It consists of two main steps. First, the two-stage fuzzy clustering algorithm with an expansion factor is carried out to divide the input space into multiple overlapping local regions. Second, in each local region, one LRBFN is trained as needed by solving a small-scale single-variable width optimization problem. The algorithm procedure of the proposed SDWO method is summarized in Table 1.

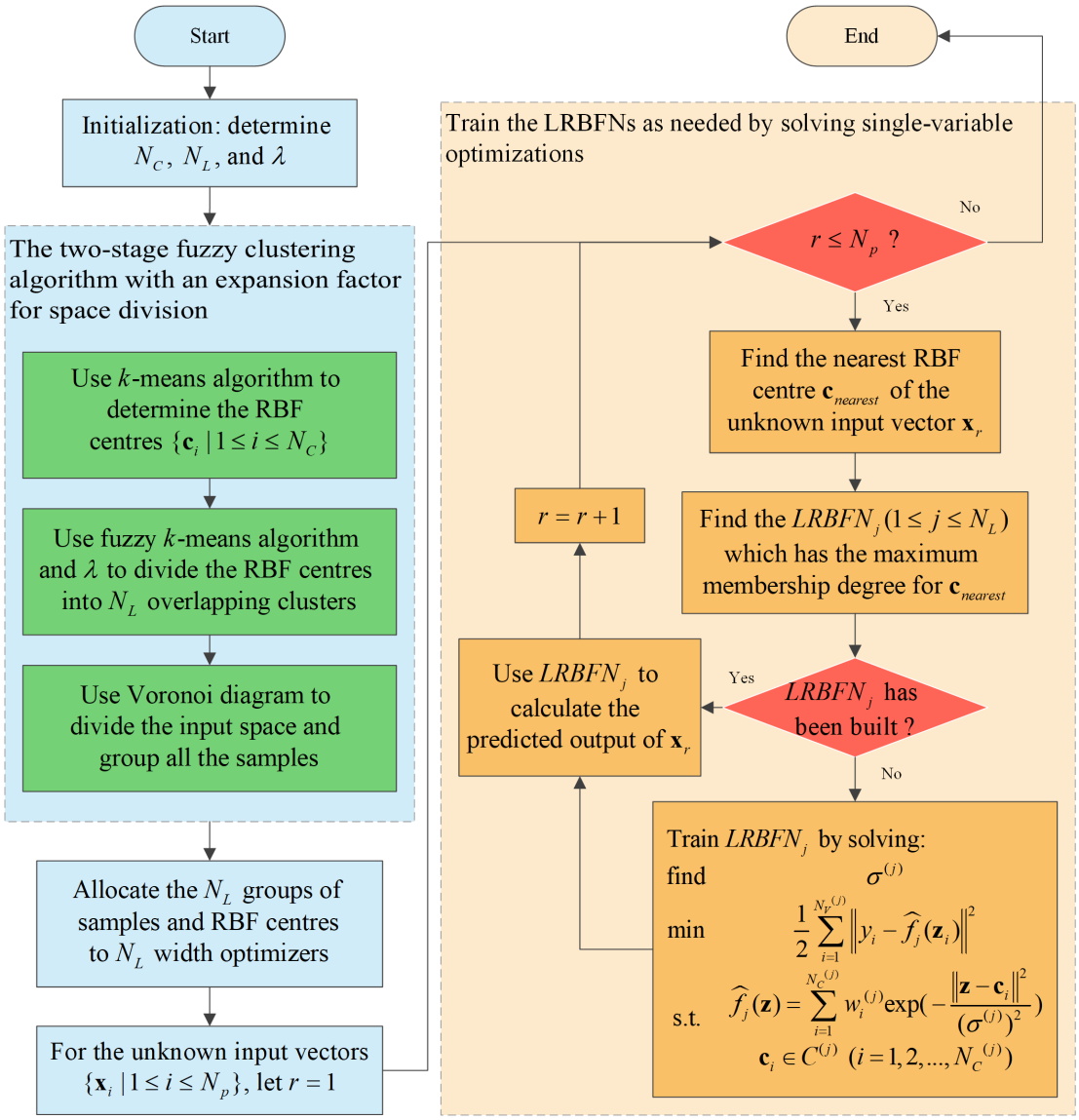


Fig. 4 Flowchart of the proposed SDWO method

Table 1 Algorithm procedure of the proposed SDWO method

Step 0:	Initialization: determine the values of N_C, N_L, and λ.
Step 1:	Carry out the two-stage fuzzy clustering algorithm with an expansion factor for space division.
Step 1.1:	The input components $\{\mathbf{x}_i \mid \mathbf{x}_i \in R^n, i = 1, 2, ..., N_T\}$ of the training samples are cluster-analysed with the k-means clustering algorithm to determine the RBF centres $\{\mathbf{c}_i \mid \mathbf{c}_i \in R^n, i = 1, 2, ..., N_C\}$.
Step 1.2:	According to Eq.(16), use the fuzzy k-means clustering algorithm with the expansion factor λ to divide the obtained RBF centres into N_L overlapping clusters.
Step 1.3:	Based on the Voronoi diagram, partition the input space into N_L overlapping local regions and divide the samples into N_L groups.
Step 2:	Allocate the obtained N_L groups of samples and RBF centres to N_L width optimizers.
Step 3:	Train the LRBFNs as needed by solving single-variable optimizations

Step 3.1: For all the unknown input vectors $\{\mathbf{x}_i | 1 \le i \le N_p\}$, where N_p is the number of input vectors for prediction, let $r = 1$.

Step 3.2: While $r \le N_p$, do:

Step 3.2.1: Find the nearest RBF centre (denoted as $\mathbf{c}_{nearest}$) of $\mathbf{x}_r$ from $\{\mathbf{c}_i \mid \mathbf{c}_i \in R^n, i = 1, 2, ..., N_C\}$.

Step 3.2.2: Find the LRBFN (denoted as $LRBFN_j$, where $1 \le j \le N_L$) which has the maximum membership degree for $\mathbf{c}_{nearest}$.

Step 3.2.3: If $LRBFN_j$ has not been built

build $LRBFN_j$ by solving the single-variable optimization problem in Eq.(19).

end

Step 3.2.4: Use $LRBFN_j$ to calculate the predicted response $\widehat{y_r}$ of $\mathbf{x}_r$.

Step 3.2.5: $r = r + 1$

Step 4: Output all the LRBNs and the predicted responses $\{\widehat{y_r} \mid 1 \le r \le N_p\}$.

3.4 **Computational complexity analysis**

In this subsection, the computational complexity of CWO and the proposed SDWO method is analysed and compared.

For an optimization problem, the number of iterations needed for convergence depends on the specific optimizer, the number of design variables, and the objective function features (unimodal or multimodal, etc.). Generally, if the number of design variables increases, the number of iterations required to obtain the optima will increase significantly for the same objective function with the same optimizer (Yao et al. 2012). Let $h(m)$ denote the number of iterations required to obtain the global optimum of an *m*-variable objective function. The CWO method involves one N_L-variable optimization as stated in Eq.(6), while the proposed SDWO method decomposes it to N_L single-variable optimizations. Hence, the numbers of iterations of CWO and SDWO can be respectively stated as $h(N_L)$ and $h(1)$, and $h(N_L) \gg h(1)$.

For CWO, the computational complexity of one iteration for calculating the pseudo-inverse of $\mathbf{\Phi}$ and the output weight vector $\mathbf{w}$, can be approximated as $O((N_T)^3)$. For SDWO, in the best case, where there is no overlap between all the local regions and the samples are divided into N_L equal groups, the computational complexity of one iteration can be approximated as $O((N_T / N_L)^3)$. Therefore, the ratio of computational complexity of CWO to SDWO can be expressed by

$$\eta_{best} = \frac{O(h(N_L) \times (N_T)^3)}{O(h(1) \times (\frac{N_T}{N_L})^3)} = O(\frac{h(N_L)}{h(1)} \times (N_L)^3) \tag{20}$$

It is obvious that $\eta_{best} \gg 1$, which indicates that the proposed SDWO method has much lower computational complexity than the CWO method. Specifically, the low computational complexity of SDWO is due to two aspects: 1) the much higher convergence rate of the single-variable optimization than the multivariable optimization; and 2) the fewer samples and smaller matrices for constructing the LRBFNs.

In the worst case of SDWO, where each local region extends to the entire input space and each LRBFN makes use of all the available samples for training, the computational complexity of one iteration of the width optimization can be approximated as $O((N_T)^3)$. The ratio of computational complexity of CWO to SDWO can be expressed by

$$\eta_{worst} = \frac{O(h(N_L)\times(N_T)^3)}{O(h(1)\times(N_T)^3)} = O(\frac{h(N_L)}{h(1)}) \tag{21}$$

Hence, as shown in Eq.(21), even in the worst case, SDWO still has much lower computational complexity than CWO due to the decomposition of the N_L-variable optimization into single-variable optimizations.

4 Validation and application

This section presents the validation and application of the proposed SDWO method. First, 4 training sample sets are generated and used to verify the effectiveness and efficiency of the proposed method in comparison with the CWO method. Second, the proposed method is applied in an inter-stage structure optimization problem to show its efficiency in solving practical optimization problems.

4.1 Validation of the proposed SDWO method

4.1.1 *Training sample sets*

4.1.1.1 "SinC" function with 60 samples

The 1-dimensional "SinC" function (Huang et al. 2006) is a popular test case to verify regression algorithms. The function is expressed by Eq.(22) and its accurate model is shown in Fig. 5. A dataset of 60 samples is generated with the full factorial design method from the input space $[-10,10]$.

$$y(x) = \begin{cases} \sin(x)/x, & x \neq 0 \\ 1, & x = 0 \end{cases} \tag{22}$$
$$x \in [-10,10]$$

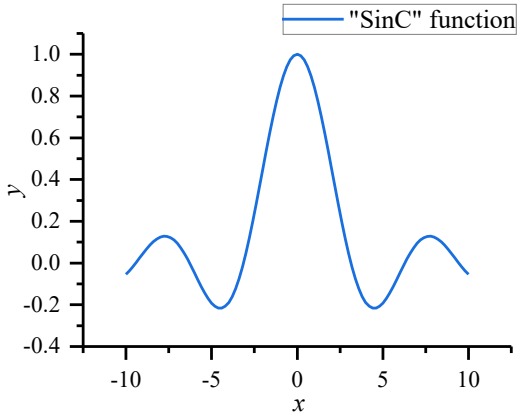


Fig. 5 Accurate model of the "SinC" function

4.1.1.2 2-dimensional function with 300 samples

The 2-dimensional test case (R.L.Haupt and S.E.Haupt 1998; Yao et al. 2012) is expressed by Eq.(23). Its accurate model is shown in Fig. 6. The Optimal Latin Hypercube Design (OLHD)

method with the maximin criterion (van Dam et al. 2009) is used to generate 300 space-filling samples within the input space $[0,3.5]^2$.

$$\begin{aligned} &f(x_1,x_2) = x_1\sin(4x_1) + 1.1x_2\sin(2x_2) \\ &x_1,x_2 \in [0,3.5] \end{aligned} \tag{23}$$

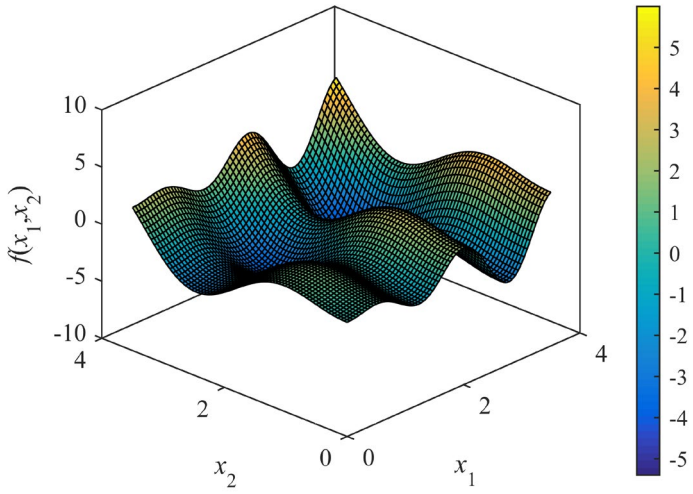


Fig. 6 Accurate model of the second test case

4.1.1.3 7-dimensional function with 2000 samples

This 7-dimensional test function (John John Kocherry, Rahul Rajan 2007) is a standard test case for MDO, which can be expressed by Eq.(24). The OLHD method with the maximin criterion is used to generate 2000 space-filling samples within the input space defined in Eq.(24).

$$\begin{aligned} f &= 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\ &\quad -1.5079x_1(x_6^2 + x_7^2) + 7.477(x_6^3 + x_7^3) \\ &\quad +0.7854(x_4x_6^2 + x_5x_7^2) \\ &x_1 \in [2.6,3.6], x_2 \in [0.7,0.8], x_3 \in [17,28], x_4 \in [7.3,8.3] \\ &x_5 \in [7.3,8.3], x_6 \in [2.9,3.9], x_7 \in [5.0,5.5] \end{aligned} \tag{24}$$

4.1.1.4 Spaceplane model with 10000 samples

This test case considers a spaceplane, which is boosted into space by a launch vehicle, then re-enters the atmosphere at the maximum speed of Mach 20, and lands as a conventional plane. The lift-to-drag ratio L/D is an important parameter to assess the aerodynamic performance of the spaceplane. It is a function of the angle of attack α, angle of sideslip β, flight altitude h, and free stream Mach number Ma, as stated in Eq.(25).

$$L/D = f(\alpha,\beta,h,Ma) \tag{25}$$

In this paper, this test case is used to illustrate the approximation capability of the SDWO method in dealing with practical engineering problems. The surface mesh of the spaceplane is shown in Fig. 7, and its partial geometric parameters are listed in Table 2. The input space of the four input parameters, $\alpha, \beta, h,$ and Ma, is defined in Table 3. Fig. 8 depicts the pressure coefficient distribution of the spaceplane on the condition that $\alpha = 8^\circ$, $\beta = -8^\circ$, $h = 10$ km, and $Ma = 6$. 10000 samples are generated within the input space by the 4-variable 10-level full factorial design method.

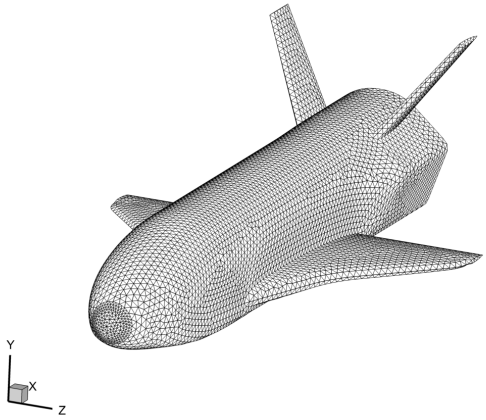


Fig. 7 The surface mesh of the spaceplane in test 4

Table 2 Geometric parameters of the spaceplane

Parameter	Value
Length	9600mm
Height	2860mm
Fuselage height	1650mm
Wingspan	5600mm
Wing area	3844200mm^2

Table 3 The lower and upper bounds for the four input parameters

Parameter	Lower bound	Upper bound
α(deg)	-10	10
β(deg)	-10	10
h(km)	0	40
Ma	0	20

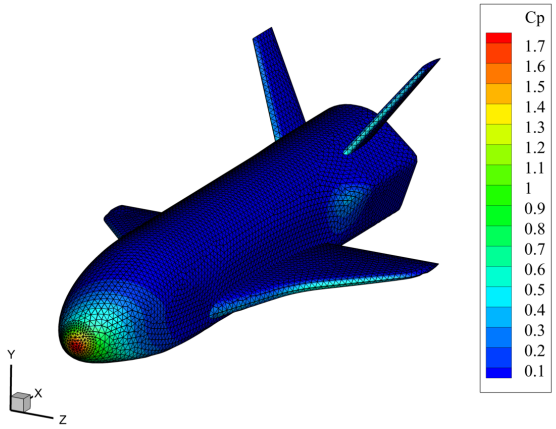


Fig. 8 Pressure coefficient distribution of the spaceplane ($\alpha = 8^{\circ}, \beta = -8^{\circ}, h = 10$ km, and $Ma = 6$)

4.1.2 *Test settings*

All the available samples are randomly divided into 3 subsets, which are used for training, validating, and testing the RBF network, respectively. The numbers of RBF centres used for four test problems are different. The specification is shown in Table 4.

Table 4 Specification of the four test cases

	Test 1	Test 2	Test 3	Test 4
Total samples	60	300	2000	10000
Training samples	36	180	1200	6000
Validation samples	12	60	400	2000
Testing samples	12	60	400	2000
RBF centres	12	60	120	220

In order to accelerate convergence and improve the approximation accuracy, the input and output component of the samples are normalized to the range $[0,1]$. $N_L = 4$ is used for both the CWO and SDWO method and for all the four test cases. The expansion factors, $\lambda = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$, are used to demonstrate its effects on the performance of SDWO.

The Particle Swarm Optimization (PSO) (Kennedy and Eberhart 1995; Alexandridis et al. 2013; Niros et al. 2015) algorithm is used as the optimizer for both CWO and SDWO. The lower and upper bounds of the searching range of the widths are 0.01 and 2, respectively. The swarm size of the PSO algorithm is 20 for both CWO and SDWO. Although the LRBFNs of the SDWO method can be trained in parallel, the training processes of both CWO and SDWO are carried out in serial to fairly compare the training efficiency.

The root mean square error (RMSE) is used as a measurement of approximation accuracy of the surrogate modes. Given *m* samples $\{(\mathbf{x}_i, y_i) \mid y_i = f(\mathbf{x}_i), i = 1, 2, ..., m\}$, the RMSE is defined as

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left\|y_i - \hat{f}(\mathbf{x}_i)\right\|^2} \tag{26}$$

The RMSE of both training and testing samples are calculated to assess the approximation accuracy of the obtained RBF networks of CWO and SDWO.

For each simulation, 30 trials have been carried out to reduce random variation in the numerical results and validate the robustness of the proposed method. The mean values of the achieved training time, training RMSE, and testing RMSE of CWO and SDWO, are calculated and analysed. All these simulations are carried out in a MATLAB 9.2 environment running on a computer with an Intel Core i5, 2.5GHz CPU, and 7.92 GB memory.

4.1.3 *Results and discussion*

Due to space limitation, only partial test results using $\lambda = \{0, 0.3, 0.6, 1\}$ are listed in Table 5. The detailed test results are given in the supplementary material of this paper. The test results of CWO and SDWO are compared in Fig. 9 with respect to training time, the number of objective-function evaluations, training RMSE and testing RMSE. For each subgraph of Fig. 9, the parameter of the horizontal coordinate is the expansion factor λ, which, as explained in section 3, is used to control the overlapping degree of SDWO, and the results of CWO are not affected by λ. Hence, the curves of CWO in Fig. 9 are all horizontal lines. The following discussion focuses on two aspects: 1) the sensitivity of the performance of SDWO to the

expansion factor; 2) the comparison of training efficiency and approximation accuracy between CWO and SDWO.

Table 5 Partial test results of CWO and SDWO using different expansion factors

Test	Method	λ	Training time (s)	Objective-function evaluations	Training RMSE	Testing RMSE
Test 1	SDWO	0.0	0.13	1135.74	0.02275	0.02575
	SDWO	0.3	0.16	1387.53	0.00141	0.00324
	SDWO	0.6	0.19	1377.55	1.38E-4	6.48E-4
	SDWO	1.0	0.31	1430.43	1.38E-4	2.97E-4
	CWO	\	0.90	3375.22	1.52E-4	2.32E-4
Test 2	SDWO	0.0	0.36	1273.57	0.02336	0.03296
	SDWO	0.3	0.73	1425.75	0.00855	0.01144
	SDWO	0.6	1.22	1338.24	0.00576	0.00800
	SDWO	1.0	1.94	1355.72	0.00482	0.00671
	CWO	\	6.45	4710.10	0.00561	0.00764
Test 3	SDWO	0.0	0.80	542.25	0.01825	0.01854
	SDWO	0.3	2.22	702.18	0.00845	0.00856
	SDWO	0.6	4.71	765.55	0.00774	0.00791
	SDWO	1.0	10.87	792.28	0.00737	0.00756
	CWO	\	18.83	1640.91	0.00788	0.00819
Test 4	SDWO	0.0	5.06	910.00	0.02982	0.03024
	SDWO	0.3	24.41	1072.50	0.02730	0.02745
	SDWO	0.6	51.95	905.07	0.02639	0.02645
	SDWO	1.0	141.20	1005.12	0.02612	0.02628
	CWO	\	397.94	3227.19	0.02623	0.02632

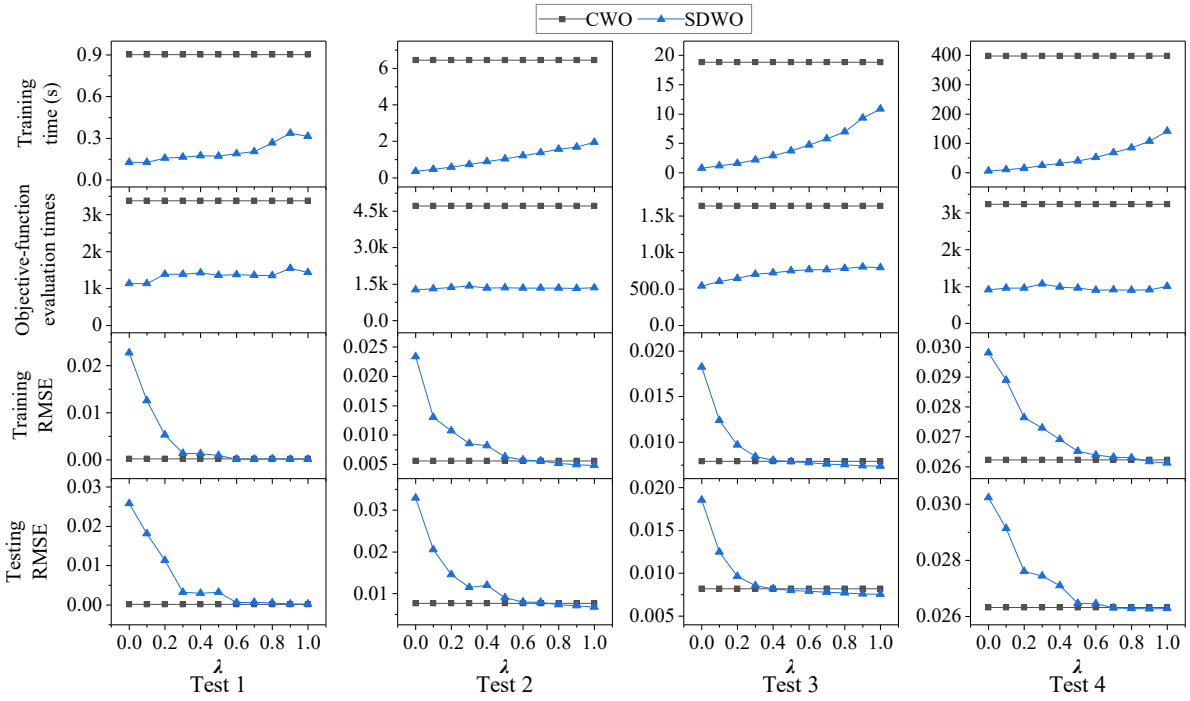


Fig. 9 Comparison of the test results of CWO and SDWO using different expansion factors

4.1.3.1 Sensitivity of SDWO to the expansion factor

First, consider the training time and the numbers of objective-function evaluations of SDWO when using different expansion factors. As it can be seen from Table 5 and Fig. 9, when λ

increases from 0 to 1, the training time of SDWO also increases. For example, in Test 4, the training time increases significantly from 5.06 s to 141.20 s. This is because as λ increases, according to Eq.(16), each local region is expanded, and each LRBFN possesses more samples for training. Hence, larger matrices are involved in the optimization process of the LRBFNs, which lengthens the training time. Additionally, it is also observed that with the increase of λ, the increasing trend of the training time is accelerating. This is because the computational complexity of calculating the pseudo-inverse of matrix $\mathbf{\Phi}$ is superlinear to the scale of the matrix. It can also be noticed that the number of objective-function evaluations changes little as λ increases for all the four test cases. This is because the more samples in each LRBFN do not affect the convergence rate of the width optimization.

Second, consider the approximation accuracy of SDWO when using different expansion factors. It is observed from Table 5 and Fig. 9 that with the increase of λ, the training RMSE and testing RMSE decrease rapidly at first, and then stay all most the same for all the four test cases. This result indicates that using more samples which are near the local region of one LRBFN can greatly reduce its approximation error, and the samples which are too far away from its local region contributes little to accuracy improvement of that LRBFN. Hence, it is not necessary to make use of all samples to construct a huge surrogate model covering the entire input space.

Third, consider the model continuity of the RBF networks obtained from SDWO with different expansion factors. For the convenience of visualization, the results of Test 2 are used to study the effect of λ. Surrogate models of Test 2 using different expansion factors are shown in Fig. 10. It is observed that when λ=0, the surrogate model shows sharp discontinuity near the boundaries of LRBFNs; when λ=0.3, the sharp discontinuity is noticeably reduced; when λ increases to 0.6 and 1, the surrogate models have small approximation error and present very smooth surfaces.

The above results demonstrate the approximation capability of the RBF networks obtained by SDWO and show effectiveness of the expansion factor in making a trade-off between the approximation accuracy, the model continuity, and the computational efficiency.

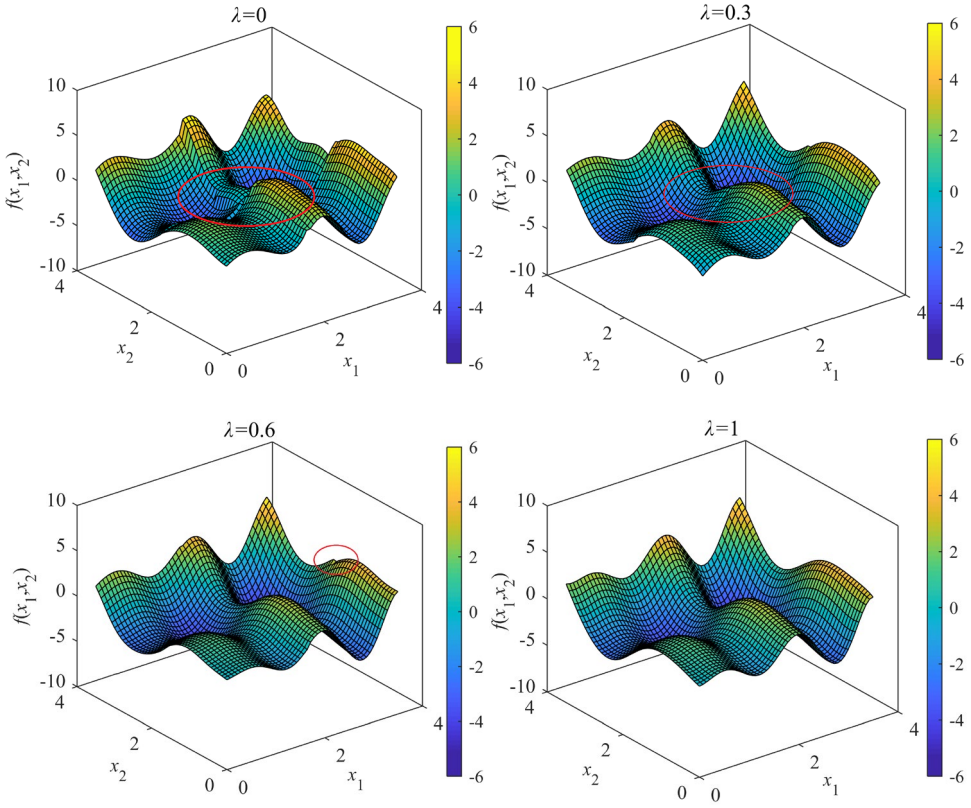


Fig. 10 Surrogate models of Test 2 using SDWO with different expansion factors. The discontinuity of the surrogate models is noticeably reduced with the increase of λ.

4.1.3.2 Comparison of CWO and SDWO

First, consider the training time and the numbers of objective-function evaluations of CWO and SDWO. The percentages of training time and the numbers of objective-function evaluations of SDWO using $\lambda = \{0.6, 1\}$ relative to CWO are listed in Table 6 and Table 7, respectively. It is obvious that SDWO spends much shorter training time than CWO for all the four test cases, even when λ increases to 1. It can also be seen that SDWO needs much fewer objective-function evaluations than CWO, which demonstrates the advantage of the single-variable optimization over the multi-variable optimization.

As it can be observed from Table 5, when $\lambda \geq 0.6$, with much shorter training time, SDWO can obtain comparable approximation accuracy to CWO. In Test 3, SDWO even achieves better approximation accuracy than CWO. This is because CWO involves an N_L-variable optimization problem, and has a much higher dimensional search space than SDWO. Hence, the SDWO method is more likely to find the global optimal or near-optimal solution than CWO with the same swarm size of PSO algorithm.

Table 6 The percentages of training time of SDWO using $\lambda = \{0.6, 1\}$ relative to CWO for the four test cases

	Test 1	Test 2	Test 3	Test 4
λ=0.6	21.1%	18.9%	25.0%	13.1%
λ=1.0	34.4%	30.1%	57.7%	35.5%

Table 7 The percentages of the numbers of objective-function evaluations of SDWO using $\lambda = \{0.6, 1\}$ relative to CWO for the four test cases

	Test 1	Test 2	Test 3	Test 4
λ=0.6	40.8%	28.4%	46.7%	28.0%
λ=1.0	42.4%	28.8%	48.3%	31.1%

The above discussion shows that SDWO can greatly reduce training time whereas maintaining comparable approximation accuracy to CWO. In general, the advantage in efficiency can be transformed to the superiority in accuracy. In Test 4, limited by the computational complexity of CWO, only 220 RBF centres are used, which are far from enough. But for the SDWO method, more RBF centres can be used because of its high efficiency. In order to further improve the approximation accuracy of the LRBFNs obtained by SDWO, additional simulations are carried out. More RBF centres are generated in these extra simulations. The fixed expansion factor $\lambda = 0.6$ is adopted for all these simulations. The simulation results are presented in Table 8, and the plots of training time, training RMSE and testing RMSE with respect to the number of RBF centres are shown in Fig. 11. It is observed that as the number of RBF centres increases, longer training time is needed; meanwhile the training and testing RMSE continue to decrease. When the number of RBF centres reaches 580, the training time of SDWO is 390.5 s, which is close to that of CWO when using 220 RBF centres (397.9 s). However, the training RMSE and the testing RMSE of SDWO are reduced by 37.0% and 33.8%, respectively. This result demonstrates that the proposed SDWO method can achieve higher approximation accuracy than CWO with similar computational cost.

Table 8 Simulation results of Test 4 using SDWO with $\lambda = 0.6$ and different numbers of RBF centres

Number of RBF centres	Training time (s)	Number of objective-function evaluations	Training RMSE	Testing RMSE
220	51.953	905.11	0.02632	0.02645
260	77.407	1021.67	0.02566	0.02585
300	99.730	1105.40	0.02486	0.02532
340	157.871	1505.07	0.02258	0.02328
380	180.738	1445.00	0.01978	0.02038
420	213.117	1450.25	0.01862	0.01938
460	230.960	1373.33	0.01832	0.01916
500	274.612	1415.13	0.01821	0.01899
540	331.552	1495.38	0.01757	0.01851
580	390.526	1580.02	0.01649	0.01740

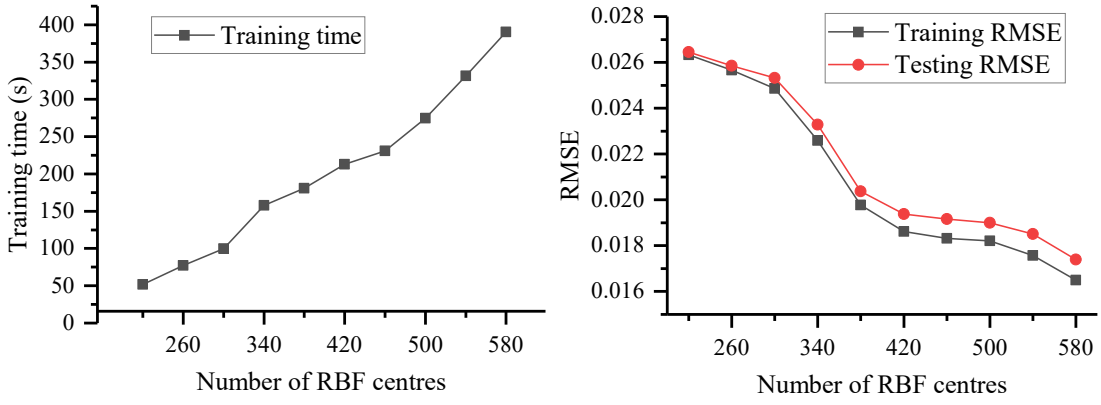


Fig. 11 Plots of the simulation results of Test 4 using SDWO with $\lambda = 0.6$ and different numbers of RBF centres

4.2 Application in structure optimization

4.2.1 *The inter-stage structure optimization*

This structure optimization problem can be referred to Sun's work (Sun et al. 2012), and it has been slightly changed in this paper. The finite element model of the inter-stage structure is shown in Fig. 12, which involves 16526 finite elements. The length of the inter-stage structure is 720 mm, and its diameter is 2300 mm. It contains 72 axial frames and 10 ring frames, and all the frames have a rectangular cross-section as depicted in Fig. 13a. There are 4 elliptical holes in the same shape on the wall of the inter-stage structure, and the schematic diagram of the elliptical holes is shown in Fig. 13b. The upper end of the inter-stage structure is subjected to a uniformly-distributed axial load of 1800 kN, and the lower end is under fixed-supported constraints, as shown in Fig. 12. The material properties are listed in Table 9.

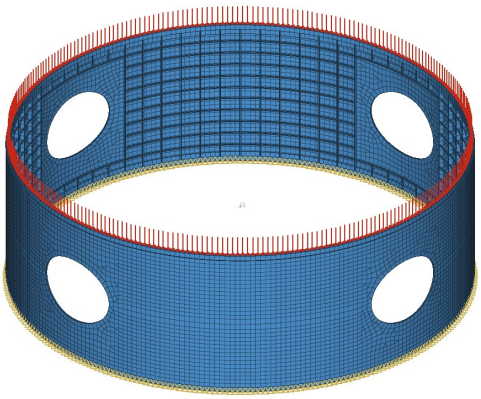
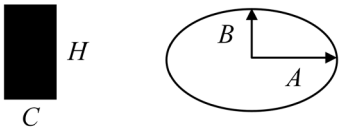


Fig. 12 Finite element model of the inter-stage structure



(a) (b)

Fig. 13 Schematic diagrams of the rectangular frame and the elliptical holes

Table 9 The material properties of the inter-stage structure

Modulus of elasticity	Density	Yield strength	Poisson's ratio
68,646 MPa	2,700 kg/m3	313.8 MPa	0.3

There are 6 design variables in the inter-stage structure optimization problem, including the wall thickness T, the thickness of the ring frames C_1, the thickness of the axial frames C_2, the height of the ring and axial frames H, and the lengths of the semi-major and semi-minor axes of the 4 elliptical holes A and B. The design space is defined by Eq.(27).

$$\begin{cases} 4\text{ mm} \le T \le 10\text{ mm} \\ 4\text{ mm} \le C_1, C_2 \le 15\text{ mm} \\ 15\text{ mm} \le H \le 30\text{ mm} \\ 150\text{ mm} \le A, B \le 220\text{ mm} \end{cases} \tag{27}$$

The objective is to minimize the weight of the inter-stage structure subject to the constraints that: 1) the maximum von Mises stress is less than the yield strength of the material σ_y=313.8 MPa ; 2) the axial displacement of the upper end of the inter-stage structure is less than $s_{\text{limit}} = 0.6$ mm . The mathematical model of the structure optimization problem can be expressed as:

$$\begin{aligned} &\text{find} && T, C_1, C_2, H, A, B \\ &\min && w(T, C_1, C_2, H, A, B) \\ &\text{s.t.} && \sigma_v(T, C_1, C_2, H, A, B) \le \sigma_y \\ & && s(T, C_1, C_2, H, A, B) \le s_{\text{limit}} \end{aligned} \tag{28}$$

where w, σ_v , and s are respectively the weight, the maximum von Mises stress, and the maximum axial displacement of the inter-stage structure.

4.2.2 *Test settings*

The proposed SDWO method is embedded into a surrogate-based optimization (SBO) (Haftka et al. 2016; Liu et al. 2017) framework to solve the structure optimization problem.

First, the OLHD method is used to generate 300 space-filling samples within the design space defined by Eq.(27), and the Finite Element Method (FEM) is carried out at these samples to calculate the maximum von Mises stress and the axial displacement by Nastran. Besides, the weight is calculated by HyperMesh.

Second, with the obtained samples, the proposed SDWO method using $\lambda = 0.6$ is applied to construct three RBF surrogate models $\hat{w}$, $\widehat{\sigma_v}$, and $\hat{s}$ respectively for the weight, the maximum von Mises stress, and the axial displacement of the inter-stage structure with respect to the design variables.

Third, the PSO algorithm is used to solve the optimization problem in Eq.(29).

$$\begin{aligned} \text{find} \quad & T, C_1, C_2, H, A, B \\ \min \quad & \hat{w}(T, C_1, C_2, H, A, B) \\ \text{s.t.} \quad & \widehat{\sigma_v}(T, C_1, C_2, H, A, B) \le \sigma_y \\ & \hat{s}(T, C_1, C_2, H, A, B) \le s_{\text{limit}} \end{aligned} \tag{29}$$

For practical engineering problems, the surrogate model constructed by using the initial samples may not satisfy the required accuracy (Peng et al. 2014; Li et al. 2017). Therefore, it is necessary to refine the surrogate model iteratively by adding new samples. Hence, in this paper, the above process is carried out iteratively. In each iteration, to balance the global exploration and local exploitation (Zhou et al. 2007; Feng et al. 2015), two new points, including the optimum of Eq.(29) and the point at which the samples are most sparse in the design space, are evaluated with the FEM and then added to the training sample set to gradually improve the three surrogate models until the optimization converges.

The structure optimization problem is further solved by the FEM using the PSO algorithm to validate the proposed SDWO-based optimization method, and in the PSO, the swarm size is set to 20. For both optimization methods, the penalty-function method is used to deal with the nonlinear constraints, and the termination criterion is that the optimum objective (weight of the inter-stage structure) remains unchanged over the last 20 consecutive iterations.

4.2.3 *Results and discussions*

A feasible baseline design is used for comparison. The baseline design and the two optimal designs obtained respectively by the proposed SDWO-based optimization method and the direct FEM-based optimization method are all listed in Table 10.

The von Mises stress contour of the baseline design is depicted in Fig. 14. As shown in Table 10, the weight of the baseline design is 180.88 kg, and its maximum von Mises stress and axial displacement are 161.21 MPa and 0.54630 mm, respectively, which are far from the limits of the constraints.

The two optimal designs are very close to each other. The von Mises stress contours of the two optimal designs are depicted in Fig. 15 and Fig. 16, respectively. It can be observed from the two contours that the maximum stress is located around the elliptical holes due to stress concentration. According to Table 10, the maximum von Mises stresses of the two optimal designs are increased compared with the baseline design, but they are still much less than the yield strength of the material. In contrast, their axial displacements almost reach the limit of the axial

displacement constraint. Moreover, the weights of both optimal designs are nearly reduced by 30% in comparison with the baseline design, and the optimal design obtained by the proposed SDWO-based method is slightly better than that obtained by the direct FEM-based method. These results indicate that the proposed SDWO-based method and the direct FEM-based method can achieve similar optimal designs.

The convergence histories of the SDWO-based method and the direct FEM-based method are depicted in Fig. 17. After 106 iterations, the SDWO-based method converges and 212 new samples are analysed by the FEM in addition to 300 initial samples, and therefore, totally 512 FEM analyses are performed. For the direct FEM-based method, it takes 72 iterations for convergence and the total number of the FEM analysis is 1440. In this paper, a single FEM analysis takes 83 seconds on average, and the computational time required for constructing and optimizing the RBF networks is negligible compared to the FEM analyses. Therefore, despite the more iterations, the proposed SDWO-based method is almost three times as efficient as the direct FEM-based method. Hence, the results demonstrate the efficiency of the proposed SDWO method in practical optimization applications.

Table 10 Comparison of the initial baseline and the optimal designs obtained by the two optimization methods

Parameters		Initial baseline	SDWO-based optimization method	Direct FEM-based optimization method
Design variables	T/mm	8.00	5.03	5.26
	C_1/mm	8.00	13.12	11.53
	C_2/mm	8.00	4.00	4.02
	H/mm	20.00	15.00	15.00
	A/mm	220.00	150.00	150.07
	B/mm	170.00	150.23	153.70
Constraints	σ_v/MPa	161.21	197.45	191.66
	s/mm	0.54630	0.59997	0.59985
Objective	w/kg	180.88	**125.77**	126.12
Number of FEM analysis		\	**512**	1440

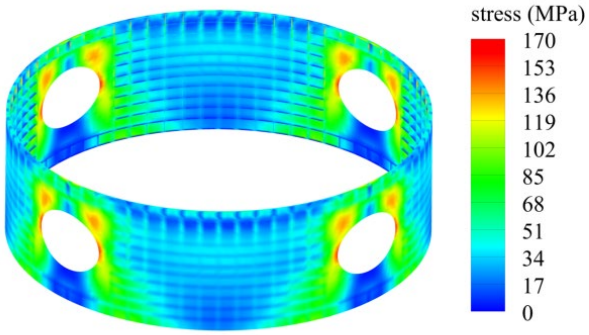


Fig. 14 Von Mises stress contour of the baseline design

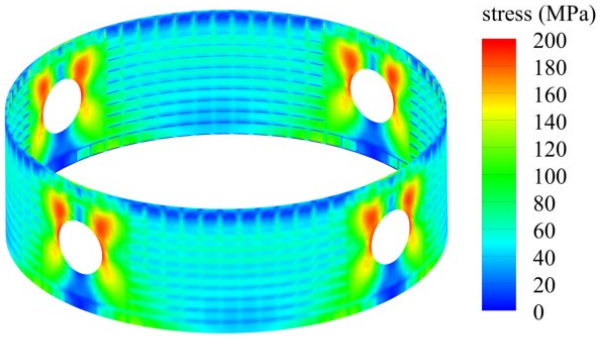


Fig. 15 Von Mises stress contour of the optimal design by the SDWO

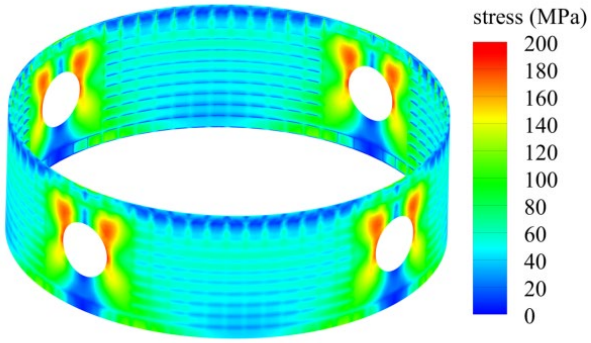


Fig. 16 Von Mises stress contour of the optimal design by the direct FEM-based optimization

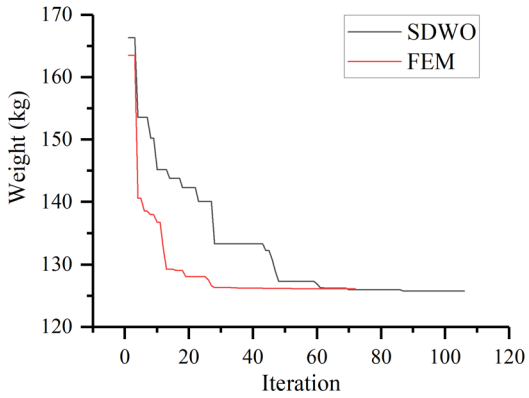


Fig. 17 Comparison of convergence histories of the SDWO and the direct FEM-based optimization

5 Conclusion

In this paper, a novel SDWO method is proposed to overcome the computationally expensive multivariable width optimization problem in training RBF networks. First, a two-stage fuzzy clustering algorithm is carried out to divide the input space into multiple overlapping local regions and to group the available samples, and an expansion factor is introduced to control the overlapping degree. Second, one LRBFN is constructed in each local region by solving a

single-variable optimization problem when the LRBFN is needed for prediction. All the LRBFNs are independent of each other, and can be trained in parallel.

The proposed SDWO method decomposes the complex multivariable width optimization into multiple small-scale single-variable optimizations, which accelerates the convergence rate. Fewer samples and smaller-scale matrices are involved for training each LRBFN without sacrificing the model flexibility and the approximation accuracy. Therefore, the SDWO method achieves much lower computational complexity than the CWO method.

Numerical experiments of four training tests verify the effectiveness of the introduced expansion factor in making a trade-off between the approximation accuracy, the model continuity, and the computational complexity, and show that the proposed SDWO method has better performance than the CWO method in terms of both training efficiency and approximation accuracy. Results of the inter-stage structure optimization demonstrate the proposed method is efficient in practical engineering applications.

The proposed SDWO method has the potential of being used to approximate expensive blackbox functions based on discrete input-output data in engineering applications, such as SBO, reliability-based optimization (RBO), and MDO of aircraft and spacecraft systems. In future studies, gradient information of the accurate function can be considered to be employed so as to further improve the accuracy and reduce discontinuity at the boundaries of LRBFNs with smaller expansion factors.

Acknowledgments

The present work was partially supported by the National Natural Science Foundation of China (Grant No. 51505385, 11502209), the National Defense Fundamental Research Funds of China (Grant No. JCKY2016204B102, JCKY2016208C001), and the China Civil Aerospace Program (Grant No. D010403, D010402).

Conflict of interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

Replication of results

The total sample datasets and the detailed results of the four validation examples are given in the supplementary material of this paper. The source code of the proposed method (written in MATLAB) and the source files of the inter-stage structural optimization examples are available at https://github.com/AlanZhangNpu/SDWO.

References

Akhtar T, Shoemaker CA (2016) Multi objective optimization of computationally expensive multi-modal functions with RBF surrogates and multi-rule selection. Journal of Global Optimization 64:17–32. doi: 10.1007/s10898-015-0270-y

Alexandridis A, Chondrodima E, Sarimveis H (2013) Radial Basis Function Network Training Using a Nonsymmetric Partition of the Input Space and Particle Swarm Optimization. IEEE Transactions on Neural Networks and Learning Systems 24:219–230. doi: 10.1109/TNNLS.2012.2227794

Aurenhammer F (1991) Voronoi diagrams. ACM Computing Surveys. doi: 10.1111/j.1447-0756.2010.01436.x

Benoudjit N, Archambeau C, Lendasse A, et al (2002) Width optimization of the Gaussian kernels in Radial Basis Function Networks. In: European Symposium on Artificial Neural Networks. Bruges, pp 425–432

Benoudjit N, Verleysen M (2003) On the Kernel Widths in Radial-Basis Function Networks. Neural Processing Letters 18:139–154. doi: 10.1023/A

Bezdek JC (1981) Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York, London

Bezdek JC, Ehrlich R, Full W (1984) FCM: The fuzzy c-means clustering algorithm. Computers & Geosciences 10:191–203. doi: 10.1016/0098-3004(84)90020-7

Bortman M, Aladjem M (2009) A Growing and pruning method for radial basis function networks. IEEE Transactions on Neural Networks 20:1039–1045. doi: 10.1006/brcg.1996.0066

Broomhead DS, Lowe D (1988) Multivariable Functional Interpolation and Adaptive Networks. Complex Systems 2:321–355

Carvalho AD, Brizzotti MM (2001) Combining RBF Networks Trained by Different Clustering Techniques. Neural Processing Letters 14:227–240. doi: 10.1023/A:1012703414861

Chen S, Cowan CFN, Grant PM (1991) Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks. IEEE Transactions on Neural Networks 2:302–309

Chuang K-S, Tzeng H-L, Chen S, et al (2006) Fuzzy c-means clustering with spatial information for image segmentation. Computerized Medical Imaging and Graphics 30:9–15. doi: 10.1016/j.compmedimag.2005.10.001

Fang H, Gong C, Li C, et al (2018) A surrogate model based nested optimization framework for inverse problem considering interval uncertainty. Structural and Multidisciplinary Optimization. doi: 10.1007/s00158-018-1931-5

Feng Z, Zhang Q, Zhang Q, et al (2015) A multiobjective optimization based framework to balance the global exploration and local exploitation in expensive optimization. Journal of Global Optimization 61:677–694. doi: 10.1007/s10898-014-0210-2

Haftka RT, Villanueva D, Chaudhuri A (2016) Parallel surrogate-assisted global optimization with expensive functions – a survey. Structural and Multidisciplinary Optimization 54:3–13. doi: 10.1007/s00158-016-1432-3

Hansen P, Jaumard B (1997) Cluster analysis and mathematical programming. Mathematical Programming 79:191–215. doi: 10.1007/BF02614317

Harpham C, Dawson CW, Brown MR (2004) A review of genetic algorithms applied to training radial basis function networks. Neural Computing and Applications 13:193–201. doi: 10.1007/s00521-004-0404-5

Huang G Bin, Wang DH, Lan Y (2011) Extreme learning machines: A survey. International Journal of Machine Learning and Cybernetics 2:107–122. doi: 10.1007/s13042-011-0019-y

Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: Theory and applications. Neurocomputing 70:489–501. doi: 10.1016/j.neucom.2005.12.126

Huang G, Saratchandran P, Sundararajan N (2005) A Generalized Growing and Pruning RBF (GGAP-RBF) Neural Network for Function Approximation. IEEE Transactions on Neural Networks 16:57–67. doi: 10.1109/TNN.2004.836241

Huang G, Saratchandran P, Sundararajan N (2004) An Efficient Sequential Learning Algorithm for Growing and Pruning RBF(GAP-RBF) Networks. IEEE TRANSACTION ON SYSTEM,MAN,AND CYBERNETICS 34:2284–2292. doi: 1083-4419/04

John John Kocherry, Rahul Rajan AV (2007) MDO Test Suite. http://www.eng.buffalo.edu/Research/MODEL/mdo.test.orig/class2prob4.html

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks. IEEE, Piscataway,NJ, pp 1942–1948

Leonard JA, Kramer MA (1991) Radial Basis Function Networks for Classifying Process Faults. IEEE Control Systems 11:31–38

Li X, Gao W, Gu L, et al (2017) A cooperative radial basis function method for variable-fidelity surrogate modeling. Structural and Multidisciplinary Optimization 56:1077–1092. doi: 10.1007/s00158-017-1704-6

Liu H, Ong YS, Cai J (2017) A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design. Structural and Multidisciplinary Optimization 1–24. doi: 10.1007/s00158-017-1739-8

Liu J, Lampinen J (2005) A Differential Evolution Based Incremental Training Method for RBF Networks. In: Proc. 2005 conference on Genetic and evolutionary computation. Washington DC, pp 881–888

Lowe D (1989) Adaptive radial basis function nonlinearities, and the problem of generalisation. In:

Proceedings of first IEE international conference on artificial neural networks. pp 171–176

Neruda R, Kudová P (2005) Learning methods for radial basis function networks. Future Generation Computer Systems 21:1131–1142. doi: 10.1016/j.future.2004.03.013

Niros AD, Tsekouras GE, Tsolakis D, et al (2015) Hierarchical Fuzzy Clustering in Conjunction with Particle Swarm Optimization to Efficiently Design RBF Neural Networks. Journal of Intelligent & Robotic Systems 78:105–125. doi: 10.1007/s10846-014-0152-4

Oh SK, Kim WD, Pedrycz W, Joo SC (2012) Design of K-means clustering-based polynomial radial basis function neural networks (pRBF NNs) realized with the aid of particle swarm optimization and differential evolution. Neurocomputing 78:121–132. doi: 10.1016/j.neucom.2011.06.031

Orr M (1998) Optimising the widths of radial basis functions. In: Proceedings 5th Brazilian Symposium on Neural Networks (Cat. No.98EX209). IEEE Comput. Soc, pp 26–29

Park J, Sandberg I (1991) Universal Approximation using Radial-Basis-Function Networks. Neural Computation 3:246–257

Peng J, Li K, Huang D (2006) A Hybrid Forward Algorithm for RBF Neural Network Construction. IEEE Transactions on Neural Networks 17:1439–1451

Peng L, Liu L, Long T, Guo X (2014) Sequential RBF surrogate-based efficient optimization method for engineering design problems with expensive black-box functions. Chinese Journal of Mechanical Engineering 27:1099–1111. doi: 10.3901/CJME.2014.0820.138

Platt JC (1991) A Resource-Allocating Network for Function Interpolation. Neural Computation 3:213–225. doi: 10.1162/neco.1991.3.2.213

Poggio T, Girosi F (1990) Networks for approximation and learning. Proceedings of the IEEE 78:1481–1497. doi: 10.1109/5.58326

R.L.Haupt, S.E.Haupt (1998) Practical Genetic Algorithms. John Wiley&Sons

Sheta AF, De Jong K (2001) Time-series forecasting using GA-tuned radial basis functions. Information Sciences 133:221–228. doi: 10.1016/S0020-0255(01)00086-X

Shi R, Liu L, Long T, et al (2018) Multidisciplinary modeling and surrogate assisted optimization for satellite constellation systems. Structural and Multidisciplinary Optimization. doi: 10.1007/s00158-018-2032-1

Smolik M, Skala V (2018) Large scattered data interpolation with radial basis functions and space subdivision. Integrated Computer-Aided Engineering 25:49–62. doi: 10.3233/ICA-170556

Sobester A, Leary SJ, Keane AJ (2004) A parallel updating scheme for approximating and optimizing high fidelity computer simulations. Structural and Multidisciplinary Optimization 27:371–383. doi: 10.1007/s00158-004-0397-9

Stetco A, Zeng XJ, Keane J (2015) Fuzzy C-means++: Fuzzy C-means with effective seeding initialization. Expert Systems with Applications 42:7541–7548. doi: 10.1016/j.eswa.2015.05.014

Sun Y, Zhu X, Zhang L, Zhao Z (2012) Structure Optimization Design of Interstage Section. MISSILES AND SPACE VEHICLES 7182:5–6. doi: 1004-7182(2012)05-0006-05

van Dam ER, Rennen G, Husslage B (2009) Bounds for maximin Latin hypercube designs. Operations Research 57:595–608

Verleysenl M, Hlavackova K (1996) Learning in RBF networks. In: International Conference on Neural Networks (ICNN). Washington, DC, pp 199–204

Vuković N, Miljković Z (2013) A growing and pruning sequential learning algorithm of hyper basis function neural network for function approximation. Neural Networks 46:210–226. doi: 10.1016/j.neunet.2013.06.004

Watson D (1993) Spatial tessellations: concepts and applications of voronoi diagrams. Computers & Geosciences. doi: 10.1016/0098-3004(93)90024-Y

Yao W, Chen X, Zhao Y, van Tooren M (2012) Concurrent Subspace Width Optimization Method for RBF Neural Network Modeling. IEEE Transactions on Neural Networks and Learning Systems 23:247–259. doi: 10.1109/TNNLS.2011.2178560

Zhou Z, Ong YS, Nair PB, et al (2007) Combining global and local surrogate models to accelerate evolutionary optimization. IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews 37:66–76. doi: 10.1109/TSMCC.2005.855506