PADOVA UNIVERSITY PRESS

Dolomites Research Notes on Approximation

Special Issue dedicated to Robert Schaback on the occasion of his 75th birthday, Volume 15 · 2022 · Pages 26–36

Solving interpolation problems on surfaces stochastically and greedily

Meng Chen[a,b] · Leevan Ling[c] · Yichen Su[d]

Communicated by Gabriele Santin

Abstract

Choosing suitable shape parameters in the kernel-based interpolation problems is an open question, whose solutions can guarantee accuracy and numerical stability. In this paper, we study various ways to select Kernel's shape parameters for interpolation problems on surfaces. In particular, we use exact and stochastically approximated cross validation approaches to select the shape parameters. When we solve the resultant matrix systems, we also deploy a greedy trial subspace selection algorithm to improve robustness. Numerical experiments are inserted along our discussion to demonstrate the feasibility and robustness of our proposed methods.

1 Introduction

Being one of the effective function approximation tools, radial basis functions (RBFs), also known as the kernel-based methods, are widely used to solve various mathematical and physical problems. In this paper, we study interpolation problems on some smooth and closed surfaces $\mathcal{M} \subset \mathbb{R}^d$ in order to explore various strategies for choosing shape parameters in RBFs. The cross validation (CV) method is used with the stochastic settings as the main method for choosing the shape parameter and the fast block greedy algorithm [8] is introduced to reduce the computational complexity.

Given some set of $N := |X|$ *RBF centers* $X := \{x_1, x_2, ..., x_N\} \subset \mathcal{M}$ and some shape parameter $\epsilon > 0$, we define the translation-invariant surface kernel-based basis function at each center $x_i \in X$ by

$$K_\epsilon(\cdot, x_i) = \phi(\epsilon||\cdot - x_i||) : \mathcal{M} \to \mathbb{R}, \tag{1}$$

for some symmetric positive definite (SPD) RBF ϕ: $\mathbb{R}^d \to \mathbb{R}$ and the Euclidean norm $||\cdot|| := ||\cdot||_{\ell^2(\mathbb{R}^d)}$. In this paper, we focus on the SPD C^∞ Gaussian RBF

$$\phi(r) = \exp\big(-(\epsilon r)^2\big) : \mathbb{R}^+ \to \mathbb{R}. \tag{2}$$

For some smooth real-valued surface function $f : \mathcal{M} \to \mathbb{R}$, we aim to approximate f based on it nodal values $f(X) := [f(x_1), f(x_2), ..., f(x_N)]$ at X. That is, the interpolation problem is stated as

$$\big(X, f(X)\big) \to S_f : \mathcal{M} \to \mathbb{R} \quad \text{such that } S_f(\xi) = f(\xi) \text{ for all } \xi \in X.$$

We seek for approximant S_f in the form of

$$S_f = \sum_{i=1}^{N} \lambda_i K_\epsilon(\cdot, x_i) =: K_\epsilon(\cdot, X)\lambda, \tag{3}$$

where $\lambda := [\lambda_1, ..., \lambda_N]^T \in \mathbb{R}^N$ is the to-be-determined RBF coefficient vector and $K_\epsilon(\cdot, X) := [K_\epsilon(\cdot, x_1), \dots, K_\epsilon(\cdot, x_N)]$ is a row vector function of size N defined on the surface $\mathcal{M}$. Here we consider a special case, we take the same set of points X as the RBF centers and interpolation points. Based on numerical expansion in (3), we approximate $f \in \mathbb{R}^N$ in trial space $U_{\phi,X} := span\{\phi(\epsilon\|\cdot - x_j\|) : x_j \in X\}$ at X and solve a linear system

$$K_\epsilon(X,X)\lambda = f(X), \tag{4}$$

where $K_\epsilon(X,X)$ is the $N \times N$ interpolation matrix of kernel K_ϵ at data points X with entries $[K_\epsilon(X,X)]_{ij} = \phi(\epsilon||x_i - x_j||)$ for x_i, $x_j \in X$. Here, in (4), we overload $f(X) \in \mathbb{R}^N$ to standard for the column vector of nodal values. With the same notation, we can also interpret $K_\epsilon(X,X) = [K_\epsilon(X, x_1), \dots, K_\epsilon(X, x_N)]$.

[a]Department of Mathematics, Nanchang University, Nanchang, China (chenmeng821@sina.com)
[b]Institute of Mathematics and Interdisciplinary Sciences, Nanchang University, Nanchang, China
[c]Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong (lling@hkbu.edu.hk)
[d]Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong (21481210@life.hkbu.edu.hk)

For interpolation problems in bulk domains, experienced researchers can make good guesses on the values of shape parameter based on empirical experience. This guessing game becomes less trivial when we are working on different surfaces. The main objective in this paper is to achieve accurate interpolation of f on surfaces by choosing an appropriate value of ϵ.

Since the RBF shape parameter ϵ determines the shape of RBF, it has a direct impact on the interpolation accuracy. Taking the Gaussian RBF as an example, a large ϵ results in a peakier Gaussian RBF and a small ϵ causes the Gaussian RBF to become flatter. Finding a way to determine/select an appropriate value of ϵ is a long-standing research topic. Many strategies have been proposed to identify good shape parameter [2, 3, 4, 5, 6, 11, 12, 14]. Among them, the cross validation (CV) method is one of the most popular methods in recent years, which shows good performance in finding the proper value of ϵ for function interpolations. In CV methods, the data set X is partitioned into two subsets for interpolation and error validation respectively. Different ways of dividing data set determines the different kinds of cross validation methods. Roughly speaking, the selection of the proper shape parameter is based on some minimization problem on these validation errors. In Section 2, an introduction to CV methods will be presented. We explain how to apply these methods to our function interpolation problems and test the feasibility and accuracy of these methods through numerical experiments. To reduce computational cost, we also consider some stochastic estimation method for the CV solution (i.e., shape parameter). In Section 3, we assume that the value of shape parameter is already determined by some CV methods or its cheap stochastic estimation. Up till this point, there is no guarantee that the resultant matrix system (3) will not suffer from the problem of ill-conditioning. Thus, we propose to solve (3) by a greedy algorithm to circumvent the problem. Numerical results show that the stochastic CV and greedy methods coupling yields more stable algorithms. To demonstrate the generality of our proposed method, we will introduce more surfaces with different shapes to perform function interpolation experiments in Section 4 as well as discuss the corresponding numerical results.

2 Cross validation approaches

Cross validation (CV) methods, also known as the out-of-sample testing, are often used in machine learning to solve prediction problems and test models on unknown data. The main idea of the CV method is to select a subset of data points as the test set, and constitute the remaining data as the training set. It makes predictions and tests on the data in the test set by training the data in the training set. If the training set performs well over the test set, it means that the training set is qualified to make predictions on unknown data. Using CV methods to select the appropriate shape parameter ϵ^* in function interpolation problems was first proposed in [12]. In interpolation problems, this method first extracts a point in the data set as the test set, then treats the remaining points as the training set and estimates the interpolation error of the point in the test set by interpolating the points in the training set.

The interpolation error υ_k at the test set $\{x_k\} \subseteq X$ of function f on the training set $X \setminus \{x_k\}$ can be calculated for each $k = 1, \ldots, N$. This method is also known as *leave-one-out cross validation* (LOOCV) method. Based on the idea, one can get interpolation errors of the *leave-two-out cross validation* (LTOCV) method [1] by using test sets of two data points. In general, the *leave-p-out cross validation* (LpOCV) method is an exhaustive cross validation technique that uses p data point as the test set at a time, and the rest is used to consist the training set. The LOOCV and LTOCV methods are then special cases of the LpOCV method with $p = 1$ and 2. As p increases, the number of possible test sets will also increase and one may only use a subset of all test sets of cardinality p.

Even though the LOOCV and LpOCV methods are simple to implement and reliable to generate good estimation performance, sometimes it is not advisable to use them; say, when the size N of data set is large, the cost of solving N symmetric matrix systems of size $(N-1) \times (N-1)$ by any direct method is $\mathcal{O}(N^4)$. Taking into account the high operations count caused by the use of large data set in function interpolation problems, some research introduce stochastic processes to the CV method to reduce the time cost. For the RBF interpolation, it is possible to reduce the computational cost of LOOCV down to $\mathcal{O}(N^3)$ by a Rippa's algorithm [12]. Generalization to LpOCV can be found in [9]. To further down the overhead cost of selecting shape parameter, one may give up using the exact validation errors and solve the LpOCV optimization problem based on some stochastic estimations. This yields the stochastic-LOOCV (SLOOCV) method [15] and stochastic-LpOCV (SLpOCV) method [10]. Now, we will provide more details about CV methods and how to find the LpOCV optimal shape parameter, denoted by ϵ_p^*.

2.1 LOOCV method

Let us first restate in our notations of the Rippa's algorithm [12] for the LOOCV optimal shape parameter ϵ_1^*, the subscript 1 indicates we leave 1 point out. For any fixed shape parameter $\epsilon > 0$, consider the calculation of LOOCV validation error vector function

$$\begin{aligned}\Upsilon(\epsilon) &:= [\upsilon_1, \ldots, \upsilon_N]^T(\epsilon) \\ &= [f(x_1) - S_{f,1}(x_1;\epsilon), \ldots, f(x_N) - S_{f,N}(x_N;\epsilon)]^T \in \mathbb{R}^N,\end{aligned}$$

where $S_{f,k} := S_{f,k}(\cdot, \epsilon)$ is the approximant of f in the form of (3) - (4) using training data $Y_k := X \setminus \{x_k\}$. Thus, $\upsilon_k := f(x_k) - S_{f,k}(x_k)$ is just the interpolation error of $S_{f,k}$ at the testing set $\{x_k\}$, for $k = 1, \ldots, N$.

Now, the approximant $S_{f,k} : \mathcal{M} \to \mathbb{R}$ of f on Y_k takes the explicit form

$$S_{f,k} = K_\epsilon(\cdot, Y_k)\Lambda_k, \tag{5}$$

where $\Lambda_k := K_\epsilon(Y_k, Y_k)^{-1} f(Y_k) \in \mathbb{R}^{N-1}$, the validation error υ_k can be calculated by

$$\upsilon_k = f(x_k) - S_{f,k}(x_k) =: f(x_k) - K_\epsilon(x_k, Y_k)\Lambda_k \in \mathbb{R}. \tag{6}$$

Next, we define the zero-upsample vector $\widehat{\Lambda}_k \in \mathbb{R}^N$ by inserting an extra zero entry at the kth position in Λ_k. The validation error υ_k in (6) can be equivalently redefined by

$$\upsilon_k = f(x_k) - K_\epsilon(x_k, Y_k)\Lambda_k =: f(x_k) - K_\epsilon(x_k, X)\widehat{\Lambda}_k \in \mathbb{R}. \tag{7}$$

Now, we can play with some linear algebra. Based on kth resultant matrix subsystem $K_\epsilon(Y_k, Y_k)\Lambda_k = f(Y_k)$, we can get

$$K_\epsilon(X, X)\widehat{\Lambda}_k = \widehat{f(Y_k)} \tag{8}$$

by setting that the entries of a nonzero-upsampling $\widehat{f(Y_k)} \in \mathbb{R}^N$ to match with those in $f(Y_k) \in \mathbb{R}^{N-1}$ at all but the kth position. Note that the kth entry $[f(X)]_k = f(x_k)$ can be related to $f(X)$ by correcting the kth entries as

$$\widehat{f(Y_k)} = f(X) - f(x_k)\hat{e}_k + K_\epsilon(x_k, X)\widehat{\Lambda}_k\hat{e}_k = f(X) - \upsilon_k\hat{e}_k, \tag{9}$$

where $\hat{e}_k = [0, ..., 1, ..., 0]^T$ represents the kth standard unit vector. Equating (8) and (9) via $\widehat{f(Y_k)}$, we can work out the followings using (4):

$$\begin{aligned} K_\epsilon(X, X)\widehat{\Lambda}_k &= f(X) - \upsilon_k\hat{e}_k \\ \widehat{\Lambda}_k &= K_\epsilon(X, X)^{-1}(f(X) - \upsilon_k\hat{e}_k) \\ \widehat{\Lambda}_k &= \lambda - \upsilon_k K_\epsilon(X, X)^{-1}\hat{e}_k \\ \hat{e}_k^T\widehat{\Lambda}_k &= \hat{e}_k^T\lambda - \upsilon_k\hat{e}_k^T K_\epsilon(X, X)^{-1}\hat{e}_k. \end{aligned} \tag{10}$$

Note in the last step that $\hat{e}_k^T\widehat{\Lambda}_k$ is the kth entry of the zero-upsample vector $\widehat{\Lambda}_k$. Thus, we rewrite (7) as

$$\upsilon_k = \frac{\lambda_k}{[K_\epsilon(X, X)^{-1}]_{kk}}, \tag{11}$$

where $[K_\epsilon(X, X)^{-1}]_{kk}$ is the (k, k)-entry of the inverse of the kernel interpolation matrix and λ_k is the coefficient for the full problem in (4). For any fixed $\epsilon > 0$, the cost of evaluating the LOOCV validation error vector function $\Upsilon(\epsilon)$ once via (11) is proportional to the cost of finding $K_\epsilon(X, X)^{-1}$, i.e., $\mathcal{O}(N^3)$. The ℓ^2-LOOCV optimal shape parameter is defined to be

$$\epsilon_1^* := \arg\min_{\epsilon > 0} ||\Upsilon(\epsilon)||. \tag{12}$$

If we solve this optimization problem by some iterative method that converges in $\mathcal{O}(1)$, the overall complexity remains at $\mathcal{O}(N^3)$.

We prepare numerical experiments to examine the effectiveness of the LOOCV method for interpolation problems on the unit sphere.

Example 2.1. Let $\mathcal{M}$ be the unit sphere. As the sinc function is in the native space of Gaussian function, in which convergence theories apply. We consider test functions

$$f_1(x) = \text{sinc}\,(2\xi_1 + 1/2) \qquad \text{for } x = (\xi_1, \xi_2, \xi_3) \in \mathcal{M} \subset \mathbb{R}^3 \tag{13}$$

and

$$f_2(x) = \arctan(2(\xi_1 + \xi_2 + \xi_3)) \qquad \text{for } x = (\xi_1, \xi_2, \xi_3) \in \mathcal{M} \subset \mathbb{R}^3. \tag{14}$$

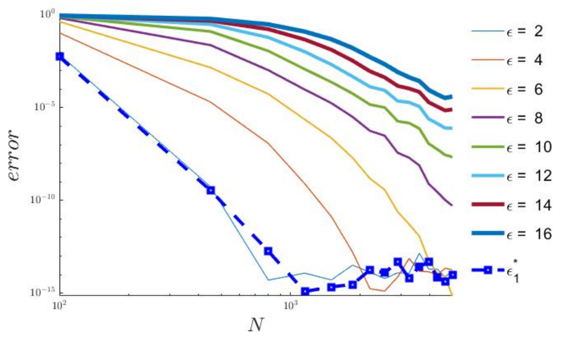
We take different numbers $N \in [2000, 6000]$ of quasi-uniform data point on $\mathcal{M}$ and use the Gaussian RBF to interpolate f_1 and f_2 with different values. To test the efficiency of the LOOCV method and make intuitive comparisons, we respectively take the shape parameter by some integers i.e., $\epsilon = 2, 4, ..., 16$ and take the optimal shape parameter ϵ_1^* chosen by the LOOCV method to solve interpolations. While using the LOOCV method, we search for the solution ϵ_1^* in the search interval $(\epsilon_{min}, \epsilon_{max}] = (0, 10]$ and solve the minimization problem in (12) by calling the MATLAB function **fminbnd**,

$$\epsilon^* = \textbf{fminbnd}(@(\epsilon)\,\Upsilon,\ \epsilon_{min},\ \epsilon_{max}),$$

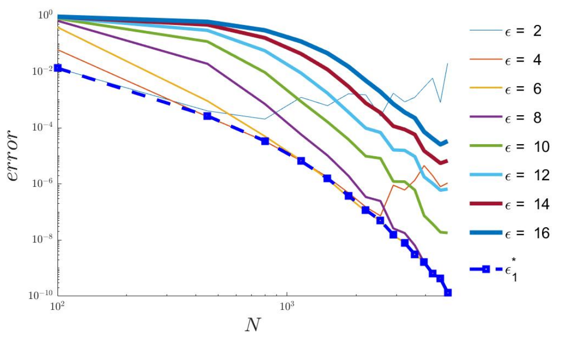
which essentially is a golden section search method.

Figures 1a and 1b compare the interpolation errors of f_1 and f_2 obtained with and without the LOOCV method for ϵ selection, which indicates that ϵ_1^* for f_1 is around 2, while for f_2 the value of ϵ_1^* varies with the value of N. It also shows that although interpolation errors decrease as the number of points N increases, the value of the constant shape parameter affects the accuracy of the interpolation result within our test range of N.

We also find that the LOOCV ϵ_1^* selected for f_1 is slightly smaller than the that for f_2. More specifically, the LOOCV optimal shape parameters for f_1 and f_2 are respectively close to the constant 2 and 6. Moreover, the interpolation error produced by using ϵ_1^* is almost the smallest among all the interpolation errors obtained using constant shape parameters, and error did not go up for large N; see the behaviour of $\epsilon = 2$ and 4 in Figure 1b for large N. In this simple example, the LOOCV strategy did successfully helps to find the ϵ^* with lower interpolation error of f_1 and f_2. □



(a) $f_1(x) = \text{sinc}\,(2\xi_1 + 1/2)$



(b) $f_2(x) = \arctan(2(\xi_1 + \xi_2 + \xi_3))$

Figure 1: Relative root mean squares error of the interpolation problems of tested functions f_1 and f_2 on the unit sphere while using constant ϵ and the LOOCV optimal ϵ_1^*.

2.2 Stochastically estimations of LOOCV shape parameters

To avoid the $\mathcal{O}(N^3)$ cost in computing the inverse of the interpolation matrix $K_\epsilon(X,X)$, we consider to use the SLOOCV method instead of the LOOCV method to find ϵ_1^*. In [15], a stochastic method were used to estimate the diagonal entries in $K_\epsilon(X,X)^{-1}$. The idea is to introduce some small-sized random matrix $W \in \mathbb{R}^{N\times s}$, whose entries $w_{ij} \sim \mathcal{N}(0,1)$ follow standard normal distribution. The number of columns $0 < s \leqslant N$ of W, which should be a small constant in order to balance the computational complexity and interpolation accuracy at the same time. The SLOOCV algorithm can be summarized in a few step. First, we compute a matrix

$$V = W[K_\epsilon(X,X)W]^\dagger \in \mathbb{R}^{N\times N}, \tag{15}$$

where † is the Moore-Penrose inverse operator. Then, we take the diagonal of V to approximate that of $K_\epsilon(X,X)^{-1}$ in (11) and compute its approximated counterpart

$$\tilde{v}_k = \frac{\lambda_k}{V_{kk}}, \tag{16}$$

and, as in (12), we optimize to identify the SLOOCV shape parameter, denoted by ϵ_{S1}^*.

The cost of computing V is dominated by the matrix-multiplication $K_\epsilon(X,X)W$ of sizes $N\times N$ and $N \times s$, that costs $\mathcal{O}(sN^2)$. The cost of a SVD factorization of an $N \times s$ matrix $K_\epsilon(X,X)W$, which is $\mathcal{O}(s^2N)$, is of lower order. In summary, SLOOCV brings the $\mathcal{O}(N^3)$ LOOCV cost down to $\mathcal{O}(s^2N + sN^2)$.

In the next example, we compare the performance of LOOCV and SLOOCV. Here we use the *reduction ratio* instead of the number of columns s to determine the appropriate size of W with respect to the N, the *reduction ratio* is defined as

$$\rho \approx s/N \in (0,1],$$

which is a more meaningful parameter that takes the original problem size into account.

Example 2.2. We interpolate f_1 and f_2 on the unit sphere with $N = 2000$ and 3000 Gaussian basis. For each reduction ratio ρ from 0.05 by 0.05 to 0.5, SLOOCV were run 100 times (to yield 100 different shape parameters that come with 100 different interpolation for each tested function). In Figures 2 and 3, we apply the LOOCV and the SLOOCV methods to select shape parameters and show the interpolation results: optimal shape parameters, computational time (measured in seconds), and relative root mean squares error

$$\text{Rel.RMS} := \frac{\|f(Z) - S_f(Z)\|}{\|f(Z)\|}$$

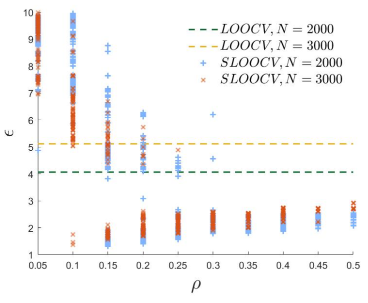
for some sufficiently dense and quasi-uniform set Z of evaluation points to determine the value of ρ when the SLOOCV method has advantages over the LOOCV method in choosing ϵ^*.

We can see in Figure 2a and 3a that, for both interpolations of f_1 and f_2, the SLOOCV selected values of ϵ_{S1}^* gradually approaches the LOOCV's ϵ_1^* from the above (for $\rho \lesssim 0.25$) and then stabilize at some range below ϵ_1^* as ρ increases. From the two subplots (b), $N = 3000$ allows larger value ρ before the computational times of SLOOCV and LOOCV break even.

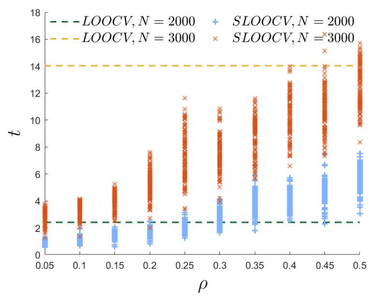
The interpolation errors in Figures 2c and 3c show rather different statistics. The ϵ_1^* selected by the LOOCV method in the interpolation of f_1 is 5.11, and we find that using ϵ smaller than ϵ_1^* does not have a great impact on the interpolation error, while using some ϵ larger than ϵ_1^* leads to inaccuracy interpolation result.

On the contrary, in the interpolation of f_2, the ϵ_1^* selected by the LOOCV method is about 5.82, and using shape parameter smaller than 5.82 will have a greater impact on the interpolation error than using ϵ that is larger than 5.82. The reason for this difference is that functions f_1 and f_2 are of different types, where f_1 is a band-limited function in the local space of the Gaussian RBF and only applies in convergent estimates, whereas f_2 is of C^∞ but not in the native space of Gaussian RBF.

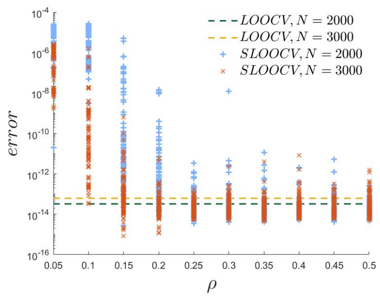
We observed that the larger value of ρ, the more suitable ϵ can be selected. Despite the similarity observed in selected shape parameters for both tested functions, the interpolation errors are rather different. For f_1 in the native space of Gaussian, SLOOCV



(a) ϵ

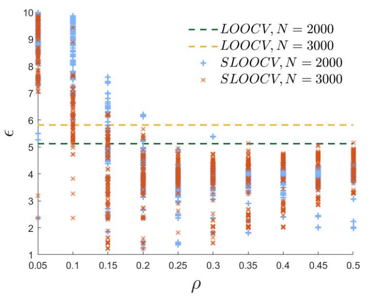


(b) Time

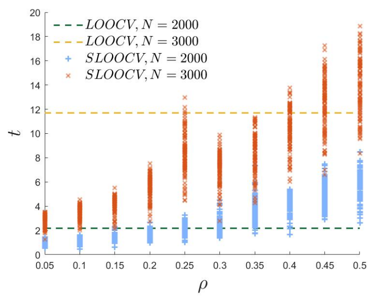


(c) Interpolation error

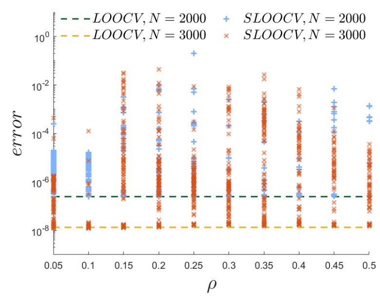
Figure 2: The selected ϵ, running time and interpolation error of f_1 when using the LOOCV method and the SLOOCV method to select ϵ^* with $\rho \in [0.05, 0.5]$.



(a) ϵ



(b) Time



(c) Interpolation error

Figure 3: The corresponding results of Figure 2 when interpolating f_2.

has high probability to yield approximant as accurate as the LOOCV one provided $\rho \gtrsim 0.30$. On the other hand, Figure 3c shows that SLOOCV is unlikely to provide comparable approximations for function f_2. $\square$

2.3 LpOCV and SLpOCV method

We now consider using the LpOCV method and the SLpOCV method to select shape parameters. In the LOOCV method, we extract only one point in X at a time to form the test set in order to calculate the validation error at all of the N points in X separately. Whereas, in the LpOCV method, we divide the data set X into (N/p) folds, i.e., subsets in a partition, of sizes approximately equal to p. Then, we leave one fold out at a time as test set in order to compute the corresponding validation error. The Rippa's algorithm (11) can be extended [9] to the LpOCV method.

We can apply similar ideas in Section (2.1) to get the extended Rippa's algorithm. For simplicity of specifying sizes of matrices and vectors below, let N/p be an integer and all folds are of size p. For $1 \le k \le N/p$, let $X_{\boldsymbol{k}} = \{x_{k_1}, \dots, x_{k_p}\} \subset X \subset \mathbb{R}^d$ be the fold (or subset) to be left-out as the test set. We denote the training set as $Y_{\boldsymbol{k}} := X \setminus X_{\boldsymbol{k}}$. Then the approximant $S_{f,k}$ of this interpolation subproblem on Y_k takes exactly the same form as (5). The validation error vector $\upsilon_k \in \mathbb{R}^p$ is defined as in (6), i.e.,

$$\upsilon_k = f(x_k) - K_\epsilon(x_k, Y_k)\Lambda_k =: f(x_k) - K_\epsilon(x_k, X)\widehat{\Lambda}_k \in \mathbb{R}^p,$$

with coefficient vector

$$\Lambda_k := K_\epsilon(Y_k, Y_k)^{-1} f(Y_k) \in \mathbb{R}^{N-p}.$$

In LpOCV, we zero-upsample $\Lambda_k \in \mathbb{R}^{N-p}$ to vector $\widehat{\Lambda}_k \in \mathbb{R}^N$ by inserting p extra zero entry at the $k_1, \dots k_p$-th positions in Λ_k. The validation error vector now takes the same form as (6)–(7) evaluated on set $X_{\boldsymbol{k}}$ instead of a single point, i.e.,

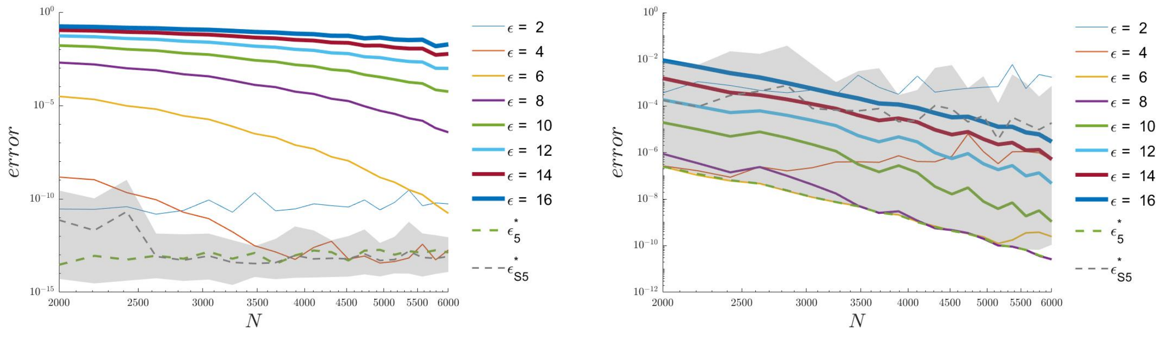
$$\begin{aligned}\upsilon_k :&= f(X_{\boldsymbol{k}}) - S_{f,k}(X_{\boldsymbol{k}}) \\ &= f(X_{\boldsymbol{k}}) - K_\epsilon(X_{\boldsymbol{k}}, Y_{\boldsymbol{k}})\Lambda_{\boldsymbol{k}} \\ &= f(X_{\boldsymbol{k}}) - K_\epsilon(X_{\boldsymbol{k}}, X)\widehat{\Lambda}_{\boldsymbol{k}}.\end{aligned}$$

Let $E_k := [\hat{e}_{k_1}, \cdots, \hat{e}_{k_p}] \in \mathbb{R}^{N\times p}$ be the matrix with standard unit vectors as columns. The correction work in (8)–(9) becomes

$$K_\epsilon(X, X)\widehat{\Lambda}_k = f(X) - E_k \upsilon_k.$$

We will arrive the extended Rippa's algorithm by zero entries in the upsample coefficient vectors

$$E_k^T \widehat{\Lambda}_k = E_k^T \lambda - E_k^T K_\epsilon(X, X)^{-1} E_k \upsilon_k = 0_p \in \mathbb{R}^p.$$



(a) f_1 **(b)** f_2

Figure 4: Relative root mean squares error of the interpolation of f_1 and f_2 on the unit sphere while using various constant ϵ, L5OCV ϵ^*_5, and (mean of error of) SL5OCV ϵ^*_{S5} shape parameters.

The counterpart (11) is given by

$$v_k = [E_k^T K_\epsilon(X,X)^{-1} E_k]^{-1}[E_k^T \lambda] \in \mathbb{R}^p.$$

In words, we solve a $p \times p$ matrix system using the submatrix of $K_\epsilon(X,X)^{-1}$ formed by extracting its $k_1, \dots k_p$-th rows and $k_1, \dots k_p$-th columns with right-hand vector $E_k^T \lambda$ that is the $k_1, \dots k_p$-th entries of the coefficient vector for the full problem in (4).

After working on all N/p folds, validation error at each data point in X will appear exactly once in one of the v_k for $k = 1, \dots, N/p$. This claim holds because we work on partitions of X and remains true even if the folds are not of the same size. We define the LpOCV validation error vector function by

$$\Upsilon_p(\epsilon) = [v_1^T, \dots, v_{N/p}^T]^T(\epsilon),$$

and the ℓ^2-LpOCV optimal shape parameter is defined to be

$$\epsilon_p^* := \arg\min_{\epsilon>0} ||\Upsilon_p(\epsilon)||_{\ell^2}. \tag{17}$$

To avoid the high cost of computing the inverse of $K_\epsilon(X,X)$ for solving v_k, We can adopt the same strategy as in the SLOOCV method; that is, we schocastically generate an approximated matrix of V as in (15) with some appropriate reduction ratio ρ, i.e., size of random matrix W. In SLpOCV, we extract the $p \times p$ submatrix of rows and columns at indices $k_1, \dots k_p$ of V and extract entries at indices $k_1, \dots k_p$ in λ to get the approximation of v_k. In matrix form, we have

$$\tilde{v}_k = [E_k^T V E_k]^{-1} E_k^T \lambda. \tag{18}$$

After going over all $k = 1, \dots, N/p$ folds, we solve the optimization problem (17) using estimated validation error $\tilde{v}_k$ by (18) and denote the resulting the SLpOCV shape parameter as ϵ^*_{Sp}.

In the following example, we provide some numerical experiments to implement the LpOCV and the SLpOCV method. When no confusion arise, we will simply write **LpOCV** and **SLpOCV** method to stand for Gaussian interpolation with shape parameter selected by the LpOCV and SLpOCV strategy.

Example 2.3. We use the same setup in Example 2.1, but here we focus our attention on larger-scaled interpolation problems, such as $N \in [2000, 6000]$, to easily compare the interpolation results generated by the fast block greedy algorithm later. We use SL5OCV, i.e., $p = 5$ in SLpOCV, and take $\rho = 0.30$.

In Figure 4, we compare interpolation errors corresponding to Gaussian interpolation of f_1 and f_2 on the unit sphere by different constant ϵ, L5OCV ϵ^*_5, and SL5OCV ϵ^*_{S5} shape parameters. Using ϵ^*_5 can achieve almost the most accurate results among all interpolations both of f_1 and f_2. The shaded gray area represents 100 interpolation errors generated by using ϵ^*_{S5}, the grey dash line represents its corresponding mean value.

In Figure 4a, the results yielded with different values of ϵ have already converged to a small error and remain stable in our range of N. The ranges of error in our 100 SL5OCV runs are very small and are within two order of magnitudes. The results of interpolation of f_2 is still unsatisfactory. The range SL5OCV interpolation error is way larger than that of L5OCV and has a large range, i.e., we need a lot of luck to get an accurate approximant using SL5OCV. □

3 Fast block greedy algorithm

From Example 2.2 and 2.3, it is shown that although the SLOOCV or SLpOCV method can reduce the computational complexity to a certain extent, due to its random processes, we cannot guarantee the generality of these methods for solving different kinds of interpolation problems. Just like the results shown in Figure 4, the SL5OCV method works for interpolation of f_1 but not for f_2.

Despite it is hard to avoid getting inappropriate shape parameters from the SLOOCV and SLpOCV method, it is still possible for us to find some strategies to deal with inaccurate results from ill-conditioned linear systems as a remedy.

The fast block greedy algorithm was proposed in [8] to deal with the severe ill-conditioned linear systems. Before this algorithm was proposed, there have been several sequential-greedy algorithms [7, 13] for solving severe ill-conditioned linear systems. The main idea of this series of methods is to use some greedy methods in iterations to select rows and columns to form a well-conditioned system in iterations until the ill-conditioned system appears, which guarantees the generation of the stable solution for the linear system. The common drawback of these methods is that they involve high computational cost, especially when solving a well-conditioned linear system. Take account of this cost issue, the fast block greedy algorithm introduces a more efficient method of selecting submatrices by means of a residual criterion, which helps to reduce the algorithm complexity from $\mathcal{O}(m^4+Nm^2)$ to $\mathcal{O}(Nm^2)$ if $m\,(1\leqslant m\leqslant N)$ columns are selected from the $N\times N$ system.

Now, let us demonstrate this method in detail. Take the linear system (4) in our interpolation problem as an example, the primal and dual residuals are calculated in this algorithm based on a constrained minimization problem

$$\begin{aligned}\min \quad & \frac{1}{2}\lambda^T\lambda\\ \text{s.t.} \quad & K_\epsilon(X,X)\lambda = f(X).\end{aligned}$$

Accordingly, a new kernel matrix is constructed with the rows and columns iteratively selected in $K_\epsilon(X,X)$ until the kernel matrix is ill-conditioned. In specific, in each iteration, this algorithm constructs a new linear system and a corresponding constrained minimization problem based on unselected rows and columns, recalculates the residuals and selects new rows and columns that are expected to have minimal effect on the condition number of the linear system. Besides, it also produces the QR factorization of the selected submatrix that can be used to solve the overdetermined linear subsystems and, if desire, compute the condition numbers of the submatrix cheaply. For more details about this algorithm, see [8].

In our interpolation process, we employ this algorithm to help to improve the interpolation accuracy as well as reduce the computation storage and requirements. We apply this algorithm to select columns in $K_\epsilon(X,X)$ and denote $\boldsymbol{v}=[v_1,\dots,v_m]\in\mathbb{N}^m$ as the indices of the selected columns. Then we use the RBF centers corresponding to the selected columns, denoted by $X_{\boldsymbol{v}}$, to construct the "approximant" of f by solving the overdetermined submatrix system

$$K_\epsilon(X,X_{\boldsymbol{v}})\lambda_{\boldsymbol{v}} = f(X), \tag{19}$$

in the least-squares sense. This is the general way to implement the fast block greedy algorithm in our interpolation problems. In the next example, still on the unit sphere, we combine the SLpOCV method with the fast block greedy algorithm to solve the interpolation/approximation problems of f_1 and f_2 and provide corresponding results. In the following text, the combination of the SLpOCV Gaussian interpolation and the fast block greedy algorithm is named as the **SLpOCV + Greedy** method.

Example 3.1. The fast block greedy algorithm usually has good performance in some large-sized linear systems. In general, it is used to achieve accurate interpolations by selecting a small subset of RBF centers. In this example, we still take $N\in[2000,6000]$, for each N, we generate point set of N points on the unit sphere and respectively use the SLOOCV method and the SL5OCV method to select shape parameters. Then running the block greedy algorithm on the $N\times N$ kernel matrix yield a selected subset $X_{\boldsymbol{v}}$ as RBF centers. Finally we solve least-square approximations of f_1 and f_2 in the form of (19) by Gaussian.

We compare the 100 interpolation/approximation results for each function with and without the fast block greedy algorithm in Figure 5 and 6 and mark the corresponding interpolation errors generated with the fast block greedy algorithm by G-ϵ^*_{Sp}. The orange and grey shaded areas are the 100 approximation errors with and without the fast block greedy algorithm. Dash lines represent their corresponding mean values.

Apparently, adding the fast block greedy algorithm successfully narrows the fluctuation range of the error no matter for the interpolations of f_1 and f_2 while using ϵ^*_{Sp}. Especially for f_1, as N increases, the fast block greedy algorithm can effectively select a suitable set of centers to keep all SLOOCV interpolation/approximation error in a narrower range. Now, Fig. 6b finally shows some improvements in using SLOOCV. With greedy algorithm in place, the worst-case interpolation/approximation error improves by 1 to 2 magnitudes.

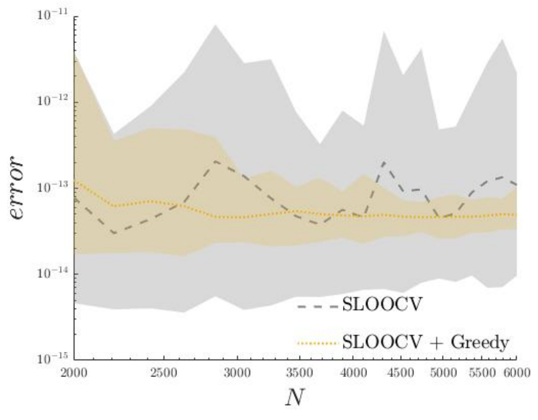
Moreover, the fast block greedy algorithm will not select a very large subspace in general cases, and if lucky, a small group of RBF centers are completely enough for accurate interpolations. Here we denote $X_G\subset X$ as the set of RBF centers selected by the fast block greedy algorithm. In figure 7, we plot the distribution of X_G and select some testing points $x^*\in X_G$ to measure denseness of these centers. The separation distance, defined by -

$$q_{X_G,\mathcal{M}} := \frac{1}{2}\min_{\substack{x_i,x_j\in X_G\\ x_i\neq x_j}} \|x_i-x_j\|_{\ell^2},$$

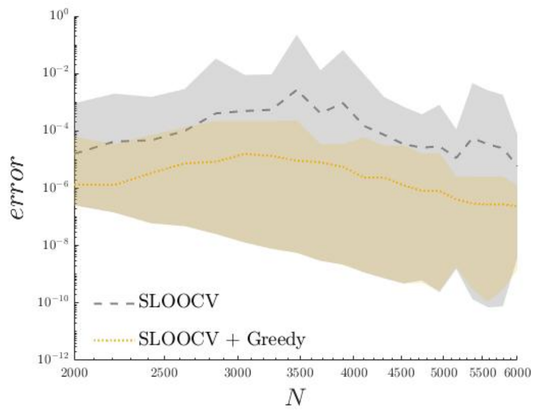
can be used to measure the denseness of the points in X_G on $\mathcal{M}$. Here we use this conception to measure distance to the closest point to x^*, we define the separation distance on point x as

$$q(x)_{X_G,\mathcal{M}} := \frac{1}{2}\min_{x_j\in X_G} \|x-x_j\|_{\ell^2},$$

and choose one particular selection with max error among 100 interpolations to show the distribution and also plot the closest 20 points around testing points in Figure 7. In consequence, there is no big difference in the separation distance among three test

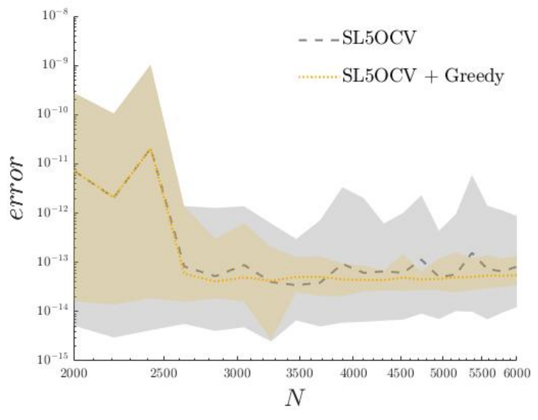


(a) f_1

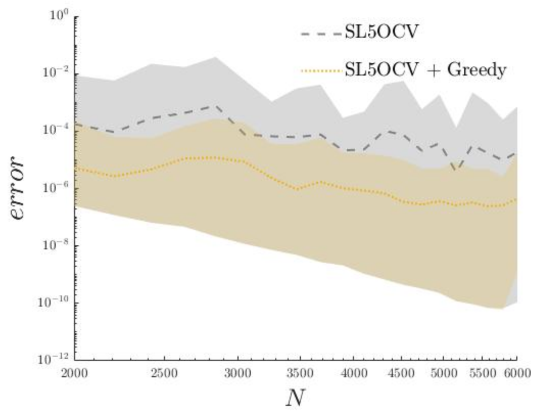


(b) f_2

Figure 5: Greedy algorithm can improve "interpolation" error when shape parameters were chosen by SLOOCV.



(a) f_1



(b) f_2

Figure 6: The corresponding results of Figure 5 when using the SL5OCV method.

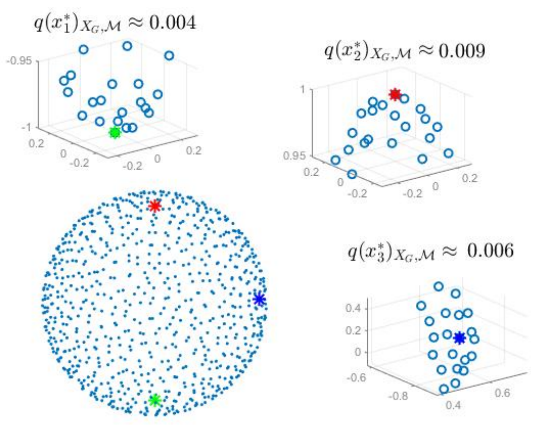
points on the unit sphere, which means X_G roughly follows a uniform distribution. Moreover, in this particular case, only $m = 783$ RBF centers are selected by the fast block greedy algorithm when taking $N = 6000$, the computational cost is successfully reduced to $\mathcal{O}(Nm^2)$.

4 More numerical results

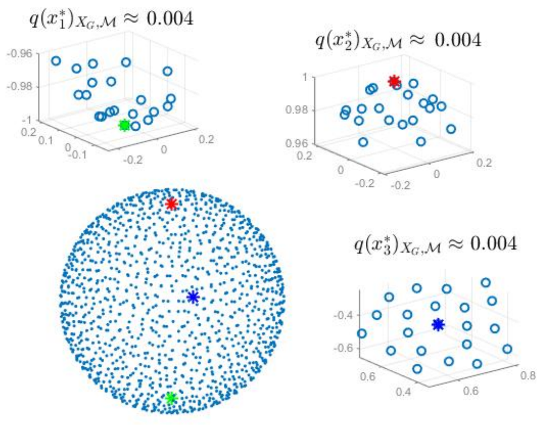
To further verify that the SLpOCV + Greedy method is generally effective, we implement different methods for choosing shape parameters in interpolations on surfaces of different shapes. If no confusion can arise, we stop writing "interpolation/approximation" from now on even when we use greedy algorithm.

Even though the interpolation results shown in Figure 5b and 6b are not ideal, it is still advisable to employ the SLpOCV + Greedy method to perform function interpolations on surfaces. In this section we try to use this approach for interpolations on torus and Dupin cyclide, and also provide interpolation errors when using different shape parameter selection strategies for comparison. In Figures 8 and 9, we display the errors of interpolations of f_1 and f_2, see that on these surfaces, the SLpOCV method fails to show its efficacy due to its substantial oscillations and extraordinary inaccuracy no matter when $p = 1$ or 5. Same as the interpolations of f_1 on the unit sphere, the SLpOCV + Greedy method finds ϵ^* well which results in smaller errors with small-scaled fluctuations. However, the performance of the SLpOCV + Greedy method in the interpolation of f_2 is still disappointing. Therefore the introduction of the fast block greedy algorithm is indispensable. As we expected, this algorithm effectively sharps the ranges of the interpolation errors and helps to reduce the error to a low level.

The distributions of the RBF centers, selected by the fast block greedy algorithm and scattered in Figures 10 and 11, show that positions with high curvature gather more RBF centers. This phenomenon is particularly evident on the Dupin cyclide, inasmuch as the separation distance of the points located at these places are much smaller. Hence it indicates that high-accurate interpolation may require more RBF centers to distribute at the positions with high curvature. But this phenomenon still needs to be verified experimentally on more surfaces with complex structures.



(a) $m = 783$ out of $N = 6000$ in interpolation of f_1, Rel.RMS $= 1.33e-13$.



(b) $m = 1279$ out of $N = 6000$ in interpolation of f_2, Rel.RMS $= 4.61e-06$.

Figure 7: The distributions of points selected by the fast block greedy algorithm on unit sphere.

5 Conclusions and discussions

In this paper, we discuss how to use the CV approaches to determine the suitable value of the shape parameter in RBF interpolations. Through numerical experiments, we introduce various ways to find the shape parameter and compare the accuracy of function interpolations on surfaces of different shapes. We find and conclude that using the SLpOCV method associated with the fast block greedy algorithm could be a feasible strategy to find suitable shape parameters and save computational consumption for accurate and stable interpolations on surfaces. In addition, more types of functions and interpolations are still needed to test the generality of our strategy. Introducing this method into the RBF collocation method for solving PDEs is also one of the topics worth studying in the future.

Acknowledgement

This work was supported by the Hong Kong Research Grant Council GRF Grants, National Natural Science Foundation (Grant No. 12001261) and Jiangxi Provincial Natural Science Foundation (Grant No. 20212BAB211020).

References

[1] H. R. Azarboni, M. Keyanpour, and M. Yaghouti. Leave-two-out cross validation to optimal shape parameter in radial basis functions. *Engineering Analysis with Boundary Elements*, 100:204–210, 2019.

[2] J. Biazar and M. Hosami. An algorithm for the shape parameter in radial basis functions interpolation. *International Conference on Advances in Applied Mathematics and Mathematical Physics*, August, 2014.

[3] M. Bozzini, L. Lenarduzzi, M. Rossini, and R. Schaback. Interpolation with variably scaled kernels. *IMA Journal of Numerical Analysis*, 35(1):199–219, 2015.

[4] S. N. Chiu, L. Ling, and M. McCourt. On variable and random shape gaussian interpolations. *Applied Mathematics and Computation*, 377:125159, 2020.

[5] M. Esmaeilbeigi and M. Hosseini. A new approach based on the genetic algorithm for finding a good shape parameter in solving partial differential equations by Kansa's method. *Applied Mathematics and Computation*, 429:419–428, 2014.

[6] G. Fasshauer and J. Zhang. On choosing "optimal" shape parameters for RBF approximation. *Numerical Algorithms*, 45:345–368, 2007.

[7] Y. Hon, R. Schaback, and X. Zhou. An adaptive greedy algorithm for solving large RBF collocation problems. *Numerical Algorithms*, 32:13–25, 2003.

[8] L. Ling. A fast block-greedy algorithm for quasi-optimal meshless trial subspace selection. *SIAM Journal on Scientific Computing*, 38:A1224–A1250, 2016.

[9] F. Marchetti. The extension of Rippa's algorithm beyond LOOCV. *Applied Mathematics Letters*, 120:107262, 2021.

[10] F. Marchetti and L. Ling. A stochastic extended Rippa's algorithm for LpOCV. *Applied Mathematics Letters*, 129:107955, 2022.

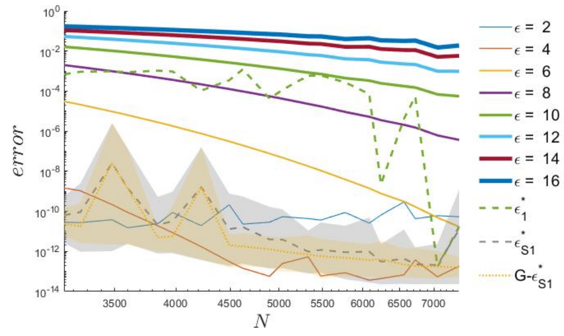
[11] M. Mongillo. Choosing basis functions and shape parameters for radial basis function methods. *SIAM Undergraduate Research Online*, 4:190–209, 2011.

[12] S. Rippa. An algorithm for selecting a good parameter c in radial basis function interpolation. *Advances in Computational Mathematics*, 11:193–210, 1999.

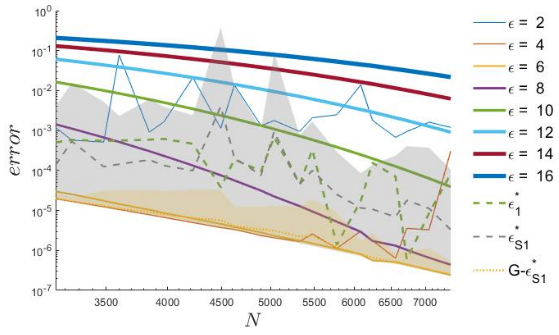
[13] R. Schaback. Adaptive numerical solution of mfs systems. *The Method of Fundamental Solutions - A Meshless Method*, pages 1–27, 2008.

[14] M. Scheuerer. An alternative procedure for selecting a good value for the parameter c in RBF-interpolation. *Advances in Computational Mathematics*, 34:105–126, 2011.

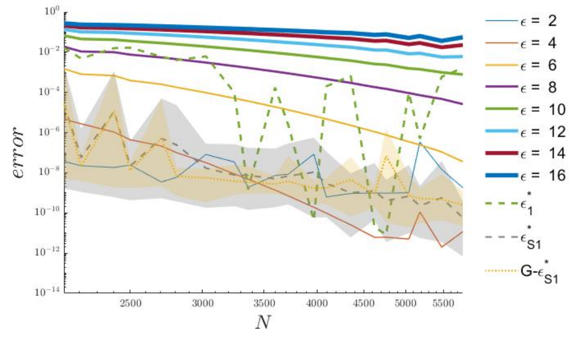
[15] F. Yang, L. Yan, and L. Ling. Doubly stochastic radial basis function methods. *Journal of Computational Physics*, 363:87–97, 2018.



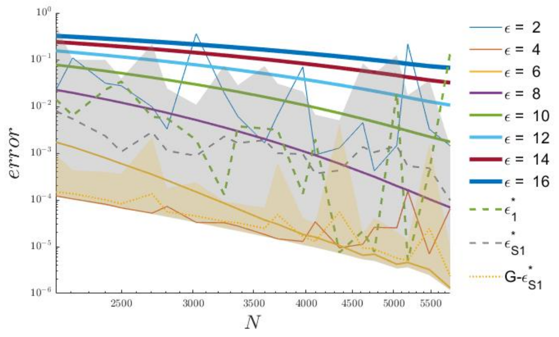
(a) Interpolation of f_1 on torus.



(b) Interpolation of f_2 on torus.

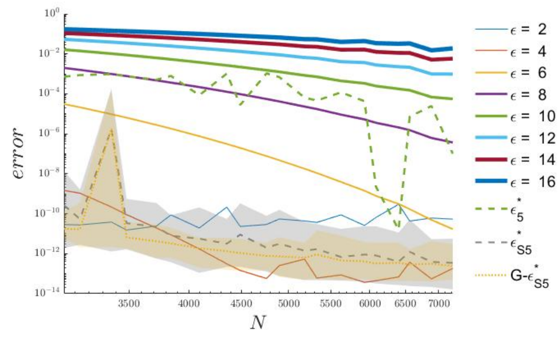


(c) Interpolation of f_1 on Dupin cyclide.

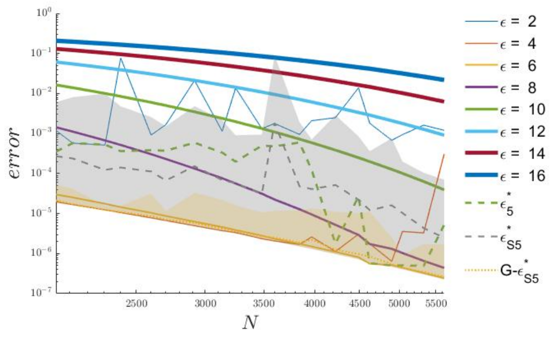


(d) Interpolation of f_2 on Dupin cyclide.

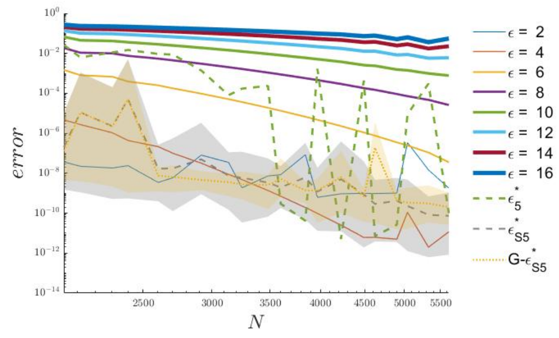
Figure 8: Relative root mean square norm of the interpolation error of f_1 and f_2 on the torus and Dupin cyclide while using constant ϵ, ϵ_1^*, ϵ_{S1}^* and G-ϵ_{S1}^*.



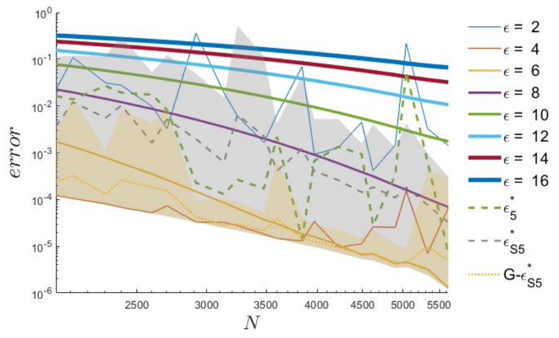
(a) Interpolation of f_1 on torus.



(b) Interpolation of f_2 on torus.

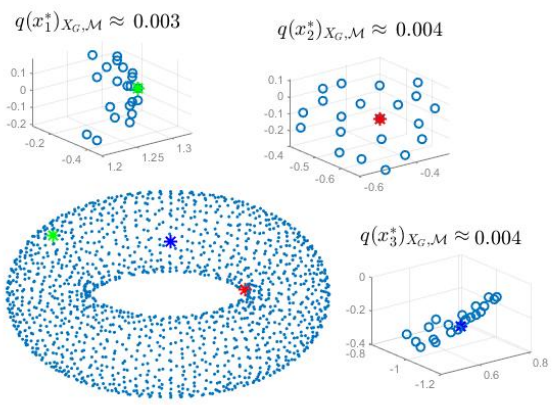


(c) Interpolation of f_1 on Dupin cyclide

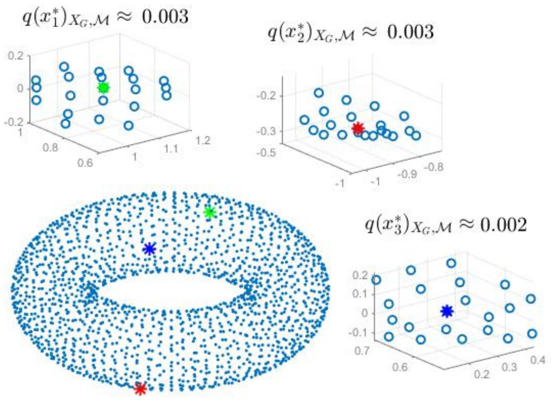


(d) Interpolation of f_2 on Dupin cyclide

Figure 9: The corresponding results of Figure 8 when using constant ϵ, ϵ_5^*, ϵ_{S5}^* and G-ϵ_{S5}^*.

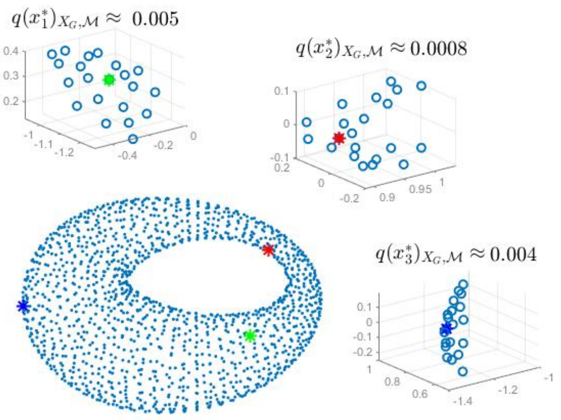


(a) $m = 1597$ out of $N = 7225$ in interpolation of f_1, Rel.RMS = 1.32e-12.

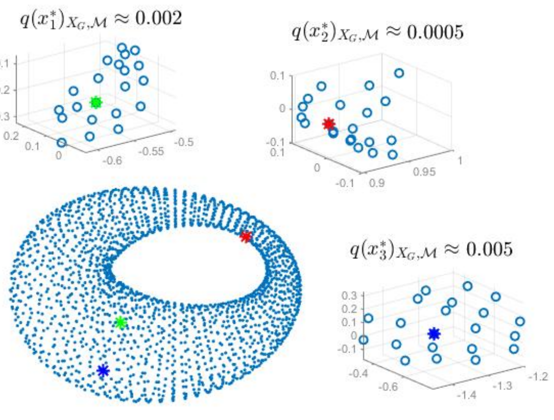


(b) $m = 1767$ out of $N = 7225$ in interpolation of f_2, Rel.RMS = 9.53e-07.

Figure 10: The distributions of points selected by the fast block greedy algorithm on torus.



(a) $m = 1443$ out of $N = 5625$ in interpolation of f_1, Rel.RMS = 1.54e-09.



(b) $m = 2045$ out of $N = 5625$ in interpolation of f_2, Rel.RMS = 8.03e-05.

Figure 11: The distributions of points selected by the fast block greedy algorithm on Dupin Cyclide.