Computers and Mathematics with Applications 164 (2024) 12–20

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

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

Efficient truncated randomized SVD for mesh-free kernel methods

A. Noorizadegan [a], C.-S. Chen [a,*], R. Cavoretto [b,*,1], A. De Rossi [b,*,1]

[a] Department of Civil Engineering, National Taiwan University, 10617, Taipei, Taiwan
[b] Department of Mathematics "Giuseppe Peano", University of Torino, via Carlo Alberto 10, 10123 Torino, Italy

ARTICLE INFO

Dataset link: https://doi.org/10.24433/CO.2648745.v1

Keywords:
Kernel methods
Radial basis functions
Effective condition number
Shape parameter
Randomized singular value decomposition

ABSTRACT

This paper explores the utilization of randomized SVD (rSVD) in the context of kernel matrices arising from radial basis functions (RBFs) for the purpose of solving interpolation and Poisson problems. We propose a truncated version of rSVD, called trSVD, which yields a stable solution with a reduced condition number in comparison to the non-truncated variant, particularly when manipulating the *scale* or *shape* parameter of RBFs. Notably, trSVD exhibits exceptional proficiency in capturing the most significant singular values, enabling the extraction of critical information from the data. When compared to the conventional truncated SVD (tSVD), trSVD achieves comparable accuracy while demonstrating improved efficiency. Furthermore, we explore the potential of trSVD by employing scale parameter strategies, such as leave-one-out cross-validation and effective condition number. Then, we apply trSVD to solve a 2D Poisson equation, thereby showcasing its efficacy in handling partial differential equations. In summary, this study offers an efficient and accurate solver for RBF problems, demonstrating its practical applicability. The code implementation is provided to the scientific community for their access and reference.

1. Introduction

Radial basis function (RBF) interpolation has garnered significant attention as a powerful technique for approximating functions and interpolating data points across various fields. Its versatility in handling complex geometries, high-dimensional data, and scattered data points has made it a popular choice in applications such as computer graphics, scientific computing, and data analysis [1–5]. However, traditional methods for solving RBF interpolation face significant computational challenges as the problem size increases. These challenges include ill-conditioning of the underlying linear systems and longer computational time required for solving the linear system.

The first element of this study is to increase the efficiency of the RBF-based methods using randomized SVD (rSVD) algorithms, which leverage randomized sampling methods to approximate the SVD of a matrix. By sampling a smaller subset of columns from the matrix, rSVD algorithms construct an approximate basis that captures the dominant features of the original matrix. This low-rank approximation not only significantly reduces the computational complexity but also addresses the ill-conditioning issue by providing a more stable solution with a reduced condition number. This method, particularly used in machine learning field, offers *efficient* alternatives to traditional SVD computations. Notable advancements include the Monte Carlo SVD [7] and robust approaches based on random projections and faster matrix multiplications [8]. In this study, we employ the randomized SVD algorithm proposed by Halko et al. [9]. This algorithm builds upon the foundations laid by previous methods and provides superior error bounds.

The second aspect of this study focuses on mitigating the randomization error in rSVD, specifically in capturing small singular values. To accomplish this, we employ a truncation technique inspired by the conventional truncated SVD (tSVD) [10,11] within the framework of the randomized SVD method. This adaptation gives rise to the truncated randomized SVD (trSVD) method. This technique involves removing singular values below a MATLAB's tolerance, resulting in a more stable solution within the trSVD approach. By discarding smaller singular values that contribute less to the approximation, we mitigate the numerical instability associated with ill-conditioned matrices. This process also en-

The code (and data) in this article has been certified as Reproducible by Code Ocean: https://codeocean.com/. More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding authors.
E-mail addresses: dchen@ntu.edu.tw (C.-S. Chen), roberto.cavoretto@unito.it (R. Cavoretto), alessandra.derossi@unito.it (A. De Rossi).
[1] Member of the INdAM Research group GNCS, Italy.

https://doi.org/10.1016/j.camwa.2024.03.021
Received 12 June 2023; Received in revised form 18 February 2024; Accepted 17 March 2024

Table 1
Some standard RBFs.

Name (Abbrev.)	$\phi(r)$	Smoothness
Gaussian (GA)	$\exp(-r^2)$	Infinite
Multiquadrics (MQ)	$(1+r^2)^{\beta/2},\ \beta>0$	Infinite
Inverse multiquadrics (IMQ)	$(1+r^2)^{\beta/2},\ \beta<0$	Infinite
Matérn/Sobolev (MS)	$(r)^{\nu}K_{\nu}(r),\ \nu>0$	Finite, $m=\nu+d/2$
Wendland C^2 (W2)	$(1-r)_+^4(4r+1)$	Finite, $m=3/2+d/2, d\le 3$
Wendland C^4 (W4)	$(1-r)_+^6\left(35r^2+18r+3\right)$	Finite, $m=5/2+d/2, d\le 3$

sures a full-rank approximation, which is crucial for problems requiring a unique solution [12,13].

In the third aspect of this paper, we establish a connection between the proposed trSVD solver and scale strategies to enhance efficiency in RBF problems. These strategies include leave-one-out cross-validation (LOOCV) [14] and effective condition number (ECN) [15–17]. While the former utilizes a *statistical* approach to determine the optimal scale, the latter leverages *conditioning* evaluation inspired by the *trade-off* or *uncertainty* principle discussed in [18,19].

In Section 2, we introduce the topic of RBF interpolation and scaling, including the extension to a Poisson problem. Section 3 provides an overview of some useful tools for RBF problems, including the SVD, randomized SVD, truncated SVD, and scale strategies. In Section 4, we further evaluate trSVD by comparing it with randomized SVD, least squares, and truncated SVD approaches. These comparisons demonstrate the efficiency and accuracy of our proposed method in the context of RBF interpolation. Additionally, an example regarding the Poisson equation is presented in Section 5. The conclusion and references follow these sections, summarizing our findings and providing additional resources.

2. RBFs

RBFs such as the Gaussian

$$\mathrm{x}\to\Phi(\mathrm{x}):=\exp\left(-\|\mathrm{x}\|_2^2\right),\quad \mathrm{x}\in\mathbb{R}^d,$$

are d-variate functions, but they can be simplified by a scalar function $\phi:\mathbb{R}\to\mathbb{R}$ of the Euclidean norm $\|\mathrm{x}\|_2$, i.e.

$$\Phi(\mathrm{x})=\phi\left(\|\mathrm{x}\|_2\right)=\phi(r),\quad \mathrm{x}\in\mathbb{R}^d,$$

with the "radius" $r=\|\mathrm{x}\|_2$ [20]. This implies rotational invariance. Table 1 starts with four RBFs with non–compact support and adds two of the compactly supported Wendland kernels using the notation $(r)_+=\max(0,r)$. The Matérn family is formulated via the modified Bessel function K_ν of the second kind with an exponential decay towards infinity. This family is strongly connected to Sobolev spaces $W_2^m(\mathbb{R}^d)$ for $m=\nu+d/2$ that arise in many PDE problems. Moreover, Table 1 gives the corresponding m for Wendland kernels.

For more details, we refer the reader to the books [22–24] and references therein.

2.1. RBF scaling

All functions of Table 1 are bell-shaped, but the width of the bell can be changed by *scaling*. If one goes over to $\phi_c(r)=\phi(r/c)$ for a positive *scale* or *shape* parameter c, the bell widens when c is increased. An equivalent scaling via $\epsilon=1/c$ is used by many authors, see e.g. [25], and allows an easier access to the *flat limit* $\epsilon\to 0$ that we ignore here.

2.2. Multivariate interpolation

Given a set $\mathcal{X}:=\{\mathrm{x}_i\}_{i=1}^n$ of arbitrary data locations or collocation points, distributed on a domain $\Omega\subset\mathbb{R}^d$, with an associated set $\mathcal{U}:=\{u_i^*=u^*(\mathrm{x}_i)\}_{i=1}^n$ of function or data values that are obtained by sampling some (unknown) function $u^*:\Omega\to\mathbb{R}$ at the data locations x_i, RBFs generate trial functions of the form

$$u(\mathrm{x})=\sum_{k=1}^n\alpha_k\phi_c(\|\mathrm{x}-\mathrm{x}_k\|_2) \tag{1}$$

as superpositions of translates $\phi_c(\|\mathrm{x}-\mathrm{x}_k\|_2)$ using coefficients $\alpha_1,\dots,\alpha_n$. The RBF $\phi:\mathbb{R}\to\mathbb{R}$ in (1) depending on a scale or shape parameter $c>0$ is such that

$$\phi_c(\|\mathrm{x}-\mathrm{x}_k\|_2)=\phi(\|\mathrm{x}-\mathrm{x}_k\|_2/c)=\phi(r/c),\quad \forall \mathrm{x},\mathrm{x}_k\in\Omega.$$

To calculate a trial function u that reproduces the data value set $\mathcal{U}$ of a given function u^* at the data locations, it is needed to solve the $n\times n$ linear system

$$u^*(\mathrm{x}_i)=\sum_{k=1}^n\alpha_k\phi_c(\|\mathrm{x}_i-\mathrm{x}_k\|_2),\quad 1\le i\le n, \tag{2}$$

or, equivalently, expressed in matrix form as follows:

$$\mathbf{A}\alpha=\mathrm{u}^*, \tag{3}$$

where $\mathbf{A}$ is the so-called *kernel matrix* with entries $A_{ik}=\phi_c(\|\mathrm{x}_i-\mathrm{x}_k\|_2)$, $i,k=1,\dots,n$, $\alpha=(\alpha_1,\dots,\alpha_n)^T$, and $\mathrm{u}^*=(u_1^*,\dots,u_n^*)^T$.

Table 1 presents some standard RBFs. GA, IMQ, MS, W2 and W4 are strictly positive definite kernels, i.e. they generate positive definite matrices. MQ is strictly conditionally positive of order 1, but non-singularity of the matrix A is guaranteed (see, e.g., [24]). Moreover, compactly supported RBFs like the Wendland functions lead to sparse kernel matrices when using a small scale.

2.3. Solving Poisson problem

With certain adaptations, the aforementioned observations can be extended to the realm of solving PDEs. In this introductory section, we limit our focus to a specific problem: the Poisson equation on a bounded domain $\Omega\subset\mathbb{R}^2$ with a sufficiently smooth boundary $\partial\Omega$. This particular case serves as a representative example for more general PDEs encountered in scientific and engineering disciplines. Given functions f^Ω defined on the domain Ω and $f^{\partial\Omega}$ defined on the boundary $\partial\Omega$, our objective is to determine a function u defined on $\Omega\cup\partial\Omega$ that satisfies [30]

$$\begin{aligned}-\Delta u&=f^\Omega\\ u&=f^{\partial\Omega}\end{aligned} \tag{4}$$

where Δ represents the Laplace operator

$$\Delta u=\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}$$

in Cartesian coordinate $\mathrm{x}=(x,y)^T\in\mathbb{R}^2$. In the Kansa approach, we make the assumption that the solution $u(\mathrm{x})$ to (4) can be estimated by expressing it as a sum of radial basis functions as shown in (1).

3. Tools for RBFs

3.1. SVD

SVD is a powerful mathematical technique used in various applications. It factorizes a matrix into three components: left singular vectors, singular values, and right singular vectors. For an $n \times n$ matrix $\mathbf{A}$, SVD can be expressed as:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

where $\mathbf{U}$ contains the left singular vectors, $\mathbf{\Sigma}$ is a diagonal matrix with singular values $\sigma_1 > \sigma_2 > \cdots > \sigma_n$ on its diagonal, and $\mathbf{V}^T$ contains the right singular vectors. The SVD decomposition can be easily computed using the MATLAB command `"svd"`.

3.2. Scale parameter strategies

In this section, we introduce two popular approaches, namely LOOCV and ECN, for determining the optimal scale value in RBF methods used for interpolation or solving PDE problems. LOOCV is a technique used in statistical analysis to find the optimal scale RBF kernel methods. It provides a statistical viewpoint for determining the most suitable parameter values in RBF-based models. It estimates model performance by iteratively leaving out one data point at a time, constructing the model on the remaining data, and predicting the omitted point's value. This process is repeated for each data point in the dataset.

To use LOOCV, the following steps are typically followed:

1. For each candidate value of scale parameter c, perform the following steps:
   (a) Iterate through each data point in the dataset, denoted as $(\mathrm{x}_i, \mathrm{u}^*(\mathrm{x}_i))$.
   (b) Temporarily, remove the current data point x_i from the dataset $\mathcal{X}$, resulting in a reduced dataset denoted as $D_{\text{train}} := \mathcal{X} \backslash \{\mathrm{x}_i\}$.
   (c) Construct the RBF interpolation model using the remaining data points in D_{train}, which involves solving a $(n-1) \times (n-1)$ linear system of the form (2) (or (3)) with the reduced dataset.
   (d) For the left-out data point $(\mathrm{x}_i, \mathrm{u}^*(\mathrm{x}_i))$, predict its value using the model, denoted as $\hat{\mathrm{u}}_i = \sum_{k=1,\ k\neq i}^{n} \alpha_k \phi_c(\|\mathrm{x}_i - \mathrm{x}_k\|_2)$.
   (e) Calculate the prediction error for the left-out data point, denoted as $e_i = \mathrm{u}^*(\mathrm{x}_i) - \hat{\mathrm{u}}_i$.
2. Aggregate the prediction errors across all but one data point, resulting in the total prediction error for the given c, denoted as $E(c) = \sum_{i=1}^{n} e_i^2$.
3. Select the candidate value λ that yields the lowest total prediction error across all data points, i.e., $c_{\text{opt}} = \arg\min_{\lambda} E(\lambda)$.

By utilizing LOOCV in the context of RBF, you can determine the scale c that minimizes the approximation error and enhances the accuracy of the RBF-based methods. The LOOCV algorithm can be computationally expensive. However, Rippa [14] introduced a simplification in the computation of error components, which reduces the complexity. The ith error component, denoted as e_i, can be calculated using the following formula:

$$e_i = \frac{\alpha_i}{A_{ii}^{-1}}.$$

Here, α_i corresponds to the coefficient associated with the solution vector $\alpha = \mathbf{A}^{-1}\mathrm{u}^*$ in (3), which is calculated using the *complete* dataset. It specifically represents the coefficient for the ith data point. On the other hand, A_{ii}^{-1} represents the ith diagonal element of the inverse interpolation matrix [24]. More accurate and faster version of cross-validation based approaches can be found in [28,29], while an application to PDEs can be found in [30,31].

Another approach to determine the optimal scale parameter in RBF-based methods is the ECN introduced in [15,26]. In this method, we maximize the effective condition number to minimize the error. It is inspired by the trade-off or uncertainty principle discussed in [19]. The algorithm to compute the ECN is as follows:

1. For each candidate value of scale parameter c, perform the following steps:
   (a) Compute the RBF interpolation matrix $\mathbf{A}$ in (3).
   (b) Compute the SVD of $\mathbf{A}$ as $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, and $\mathbf{S}$ is a diagonal matrix with singular values.
   (c) Find the minimum value from the diagonal of matrix $\mathbf{S}$ to determine the smallest singular value, σ_n.
   (d) Solve the linear system (3) for coefficients α.
   (e) Compute the ECN:
   $$\kappa_{\text{eff}} = \frac{1}{\sigma_n} \frac{\|\mathrm{u}^*\|_2}{\|\alpha\|_2}.$$
2. Output the optimal scale c_{opt}, which corresponds to the largest ECN in Step 1.

3.3. Truncated SVD

The RBF method often exhibits a *trade-off* or *uncertainty principle* due to the presence of small singular values in the system, leading to poor conditioning [18,19]. To address this, we employ orthogonal transformations to transfer the unknowns and data in (3), resulting in new vectors defined as:

$$\mathrm{g} = \mathbf{U}^T\mathrm{u}^* \in \mathbb{R}^n, \quad \text{and} \quad \mathrm{y} = \mathbf{V}^T\alpha \in \mathbb{R}^n.$$

By minimizing the squared norm, we have:

$$\begin{aligned}\|\mathbf{A}\alpha - \mathrm{u}^*\|_2^2 &= \|\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}\alpha - \mathrm{u}^*\|_2^2 = \|\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}\alpha - \mathbf{U}^{\mathrm{T}}\mathrm{u}^*\|_2^2 \\ &= \|\mathbf{\Sigma}\mathrm{y} - \mathrm{g}\|_2^2 = \sum_{j=1}^{k} \left(\sigma_j y_j - g_j\right)^2 + \sum_{j=k+1}^{n} g_j^2, \end{aligned} \tag{5}$$

where k represents the rank of the kernel matrix. Note that orthogonal transformations are applied using Euclidean norms. The minimum in (5) exists and can be obtained by solving the equations:

$$\sigma_j y_j = g_j, \quad 1 \le j \le k, \tag{6}$$

while $y_{k+1}, \ldots, y_m$ remain undetermined in cases of rank loss. When the rank loss occurs, the small and imprecise σ_j in (6) lead to large and imprecise y_j. To ensure accurate calculations, it is necessary to have a sufficient margin above the numerical pollution for σ_j. It is important to note that implementing the system with rank loss is unreliable and should not be used in critical engineering projects where unique solutions are required.

If a numerical rank $k \le n$ has been determined, an approximate solution to the linear equation system (3) can be obtained by truncating the SVD at dimension k. This truncation process involves calculating $g = \mathbf{U}^T\mathrm{u}^*$, solving (6) for $y_1, \ldots, y_k$, and setting $y_{k+1}, \ldots, y_m$ to *zero*. The resulting solution, denoted as $\hat{\mathrm{y}}$, is used to compute the approximate solution $\hat{\alpha} = \mathbf{V}\hat{\mathrm{y}}$.

The quantity $\|\mathbf{A}\hat{\alpha} - \mathrm{u}^*\|_2^2$ represents the residual norm and can be evaluated to assess the approximation accuracy. Specifically, it is given by $\sum_{j=k+1}^{n} g_j^2$, which ideally should be small.

3.4. Randomized SVD and its truncation

In this section, we introduce the rSVD algorithm, specifically tailored for RBF problems (Fig. 1). rSVD offers an efficient approach to approximate the SVD of a matrix, making it well-suited for large-scale RBF applications. The rSVD algorithm follows the steps outlined below [6]:

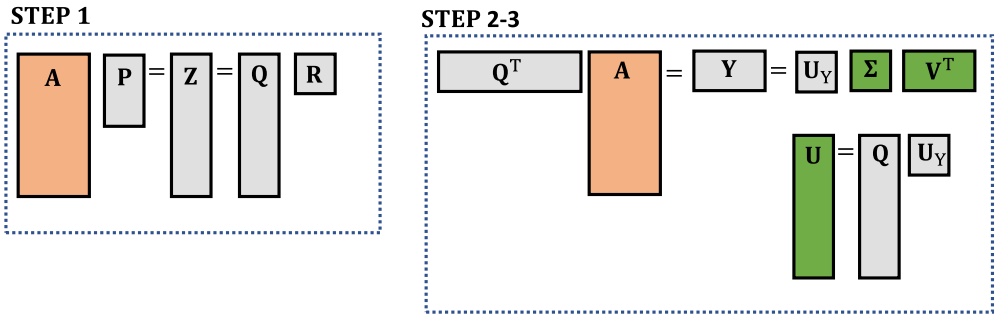


Fig. 1. Schematic of rSVD algorithm.

Step 0: Specify the Target Rank
Determine the target rank r, where $r < n$ is the number of degree of freedom (DOF) in matrix $\mathbf{A}$.

Step 1: Random Projections and Column Space Approximation
We begin by constructing a random projection matrix $\mathbf{P} \in \mathbb{R}^{m\times r}$ to sample the column space of $\mathbf{A} \in \mathbb{R}^{n\times m}$:

$$\mathbf{Z} = \mathbf{AP}.$$

The matrix $\mathbf{Z}$ may be considerably smaller than $\mathbf{A}$, particularly for low-rank matrices with $r \leq m$. The use of a random projection matrix $\mathbf{P}$ is unlikely to eliminate crucial components of $\mathbf{A}$, making $\mathbf{Z}$ a high-probability approximation of the column space of $\mathbf{A}$. Consequently, the low-rank QR decomposition of $\mathbf{Z}$ yields an orthonormal basis for $\mathbf{A}$:

$$\mathbf{Z} = \mathbf{QR}.$$

Step 2: Projection onto Subspace and Matrix Decomposition
With the low-rank basis $\mathbf{Q}$, we project $\mathbf{A}$ into a smaller space:

$$\mathbf{Y} = \mathbf{Q}^T\mathbf{A}.$$

It follows that $\mathbf{A} \approx \mathbf{QY}$, with better agreement when the singular values decay rapidly for $k > r$.

Step 3: Reconstruction of High-Dimensional Modes
Now, the singular value decomposition is computed on $\mathbf{Y}$:

$$\mathbf{Y} = \mathbf{U}_Y\Sigma\mathbf{V}^T.$$

Since $\mathbf{Q}$ is orthonormal and approximates the column space of $\mathbf{A}$, the matrices $\mathbf{U}_Y$ and $\mathbf{V}$ are the same for $\mathbf{Y}$ and $\mathbf{A}$.
Finally, the high-dimensional left singular vectors $\mathbf{U}$ are reconstructed using $\mathbf{U}_Y$ and $\mathbf{Q}$:

$$\mathbf{U} = \mathbf{QU}_Y.$$

Choice of random matrix P
There are various options for choosing the random matrix $\mathbf{P}$ including:

- **Gaussian Random Projections:**
  - Elements of matrix $\mathbf{P}$ are independent and identically distributed Gaussian random variables.
  - Preferred for favorable mathematical properties and rich information extraction in matrix $\mathbf{Z}$.
  - Expensive in terms of generation, storage, and computation.
- **Uniform Random Matrices: (preferred in this study)**
  - Frequently used but shares similar limitations to Gaussian projections.
- **Rademacher Matrices [36]:**
  - Entries can be +1 or -1 with equal probability.
- **Sparse Projection Matrices [6]:**
  - Improve storage and computation but include less information in the sketch.

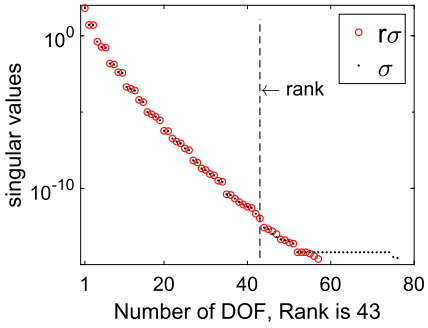


Fig. 2. Comparison of *original* and *randomized* singular values with respect to the DOF. The results are obtained at $c = 1$ of the GA RBF with $n = 76$ and a target rank of $r = 57$. In order to enhance visualization, the results are obtained using small-scale parameters.

To evaluate the efficacy of the rSVD algorithm, we conducted experiments on a dataset of size $n \times n$. Fig. 2 showcases a comparison between the singular values obtained from the rSVD algorithm ($r\sigma$ represented by red circle) and the true singular values of the original SVD algorithm (σ represented by black dots). The vertical line indicates the rank of the randomized singular values. The results demonstrate that the rSVD accurately captures the dominant singular values, although the approximation of smaller singular values exhibits reduced accuracy. To mitigate this error and ensure the RBF linear system computation remains in a full-rank setting, we propose truncating the randomized singular values at the rank from MATLAB, which leads to the modified algorithm referred to as trSVD.

For the purpose of truncating rSVD, in the absence of a user-supplied threshold value, MATLAB calculates a threshold value `tol` using the formula:

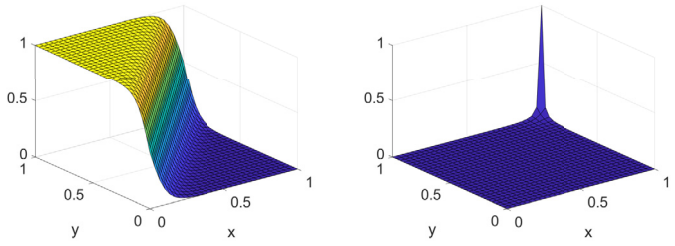
$$\texttt{tol} = \texttt{max(size(}\mathbf{A}\texttt{))} * \texttt{eps(norm(}\mathbf{A}\texttt{))}$$

Here, `eps(x)` represents the positive distance from $|x|$ to the next larger floating-point number with the same precision as x, where $\texttt{max(size(}\mathbf{A}_{n\times n}\texttt{))} = n$.

The trSVD algorithm plays a pivotal role in our research pertaining to RBF approximation. By incorporating the trSVD as a preprocessing step, we can *efficiently* and *accurately* acquire low-rank approximations of matrices involved in RBF computations. This facilitates faster computation in RBF-based applications such as interpolation, regression, and PDE solution.

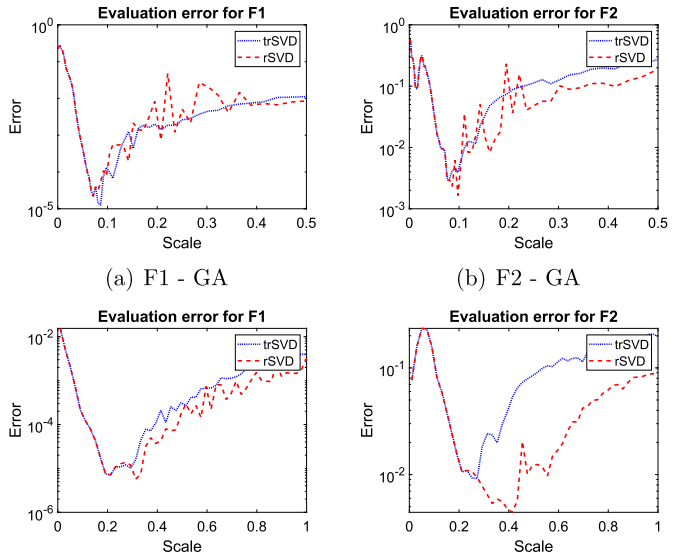
4. Numerical examples

In all subsequent examples in this section, the target rank of the rSVD-based technique is assumed to be $n/2$, unless explicitly mentioned



(a) F1 (b) F2

Fig. 3. The profiles of the test functions.



(a) F1 - GA (b) F2 - GA

(c) F1 - MQ (d) F2 - MQ

Fig. 4. Example 1: The profiles of the rSVD and trSVD.

otherwise. Here, n represents the number of collocation points, and the Halton point distribution is used. To evaluate the errors, we calculate the absolute maximum error, hereinafter simply denoted as *error*, on L interior points uniformly distributed on $\Omega = [0,1]^2$:

$$\text{error} = \max_{i=1,\dots,L} |u(\mathrm{x}_i) - u^*(\mathrm{x}_i)|.$$

The following test functions are used to be interpolated on the domain $\Omega = [0,1]^2$:

$$\mathrm{F1}\,(x,y) = \frac{1}{9}\left[\tanh(9y - 9x) + 1\right],$$

$$\mathrm{F2}\,(x,y) = \frac{0.0025}{(x-1.01)^2 + (y-1.01)^2}.$$

F1 is smooth [27], while F2 has singularity near the boundary [15] (see Fig. 3). The RBFs used in our study are mentioned in Table 1, where both MQ and IMQ are considered with $\beta = 1$. For Matérn and Wendland RBFs, the required parameters are mentioned in the following examples. The numerical experiments were executed on a computer equipped with an Intel(R) Core(TM) i9-9900 CPU operating at 3.10GHz with a total of 64.0 GB of RAM. The codes implemented in MATLAB are available at: https://github.com/CMMAi/Truncated-Randomized-SVD-for-Kernel-Based-Methods. For user's convenience by way of example, we explicitly refer to Example 3, discussed in the following.

Example 1. In the first example, we investigate the error behavior of trSVD in comparison with rSVD with respect to the scale parameter (see Fig. 4). Using GA and MQ RBFs for both smooth and non-smooth test functions, from these plots we can see that:

1. For GA RBF, both smooth and non-smooth test functions show that trSVD produces a much smoother error curve compared to rSVD. The error curve for rSVD exhibits significant fluctuations, indicating instability with respect to the scale. Note that both trSVD and rSVD achieve similar minimum errors.
2. For MQ RBF and the smooth test function, trSVD also exhibits a smoother error curve, but with similar minimum error. However, for the non-smooth test function, rSVD yields a smaller error compared to trSVD. This difference in error is likely due to the removal of small singular values in truncated randomized SVD, which may not be reliable.

Overall, trSVD demonstrates better stability and smoother error curves, indicating its reliability and suitability for RBF interpolation.

Example 2. In this example, we present a comparative analysis between the proposed trSVD and least squares (lsqr) methods. We evaluate their performance by comparing the results obtained using these methods with an equal number of DOF, specifically set to $n/2$, where n represents the number of collocation points.

To ensure fair comparisons, both methods are truncated at MATLAB's rank, resulting in full-rank implementations. While we can demonstrate the results for various test functions, we focus on showcasing the outcomes obtained by applying different RBFs, as the observations remain consistent across the test functions. The RBFs utilized in our study include GA, MQ, IMQ, MS of order 3 and 5, and W2. From Fig. 5, we observe that the error profiles for both trSVD and lsqr using GA RBF are comparable in accuracy. However, when employing other RBFs, we notice that trSVD outperforms lsqr in terms of accuracy. By capturing the dominant singular values and corresponding singular vectors, trSVD achieves a more compact and accurate representation of the data. In contrast, lsqr does not explicitly utilize these low-rank structures, potentially leading to less compact and accurate representations.

Example 3. In this example, we will provide a detailed comparison between the tSVD and trSVD solvers. Table 2 presents the interpolation results obtained using different RBFs along with two solvers, i.e. tSVD and trSVD for $n = 500, 2000$. Additionally, Fig. 6 plots the CPU time with respect to the number of collocation points. The table and plot provide insights into the error and CPU time, respectively, for each combination, allowing us to analyze the performance of these methods. It is worth noting that the CPU time represents the execution time required for the tSVD or trSVD computations per scale value. Based on these findings, we draw the following observations:

1. Accuracy Comparison:
   Across different RBFs (GA, IMQ, MQ, and MS), both tSVD and trSVD demonstrate similar levels of accuracy, as evident from the error values.
2. Efficiency Comparison:
   Notably, trSVD exhibits a significant advantage in terms of computational efficiency. It consistently outperforms tSVD in terms of CPU time for all RBFs and problem sizes.

Overall, the plot demonstrates that trSVD offers a compelling advantage in terms of computational efficiency compared to tSVD, making it a more desirable choice for large-scale interpolation tasks.

Example 4. In this example, we illustrate the effective integration of the trSVD method with LOOCV and ECN for determining an optimal scale parameter. A large number of studies, such as [21,28,29,32], have demonstrated the relationship between small errors and small LOOCV values. Additionally, the authors in [15,26,34,35] have explored the connection between ill-conditioned linear systems, large ECN values, and small errors. By leveraging these insights, we showcase the effectiveness of trSVD in selecting a optimal scale parameter.

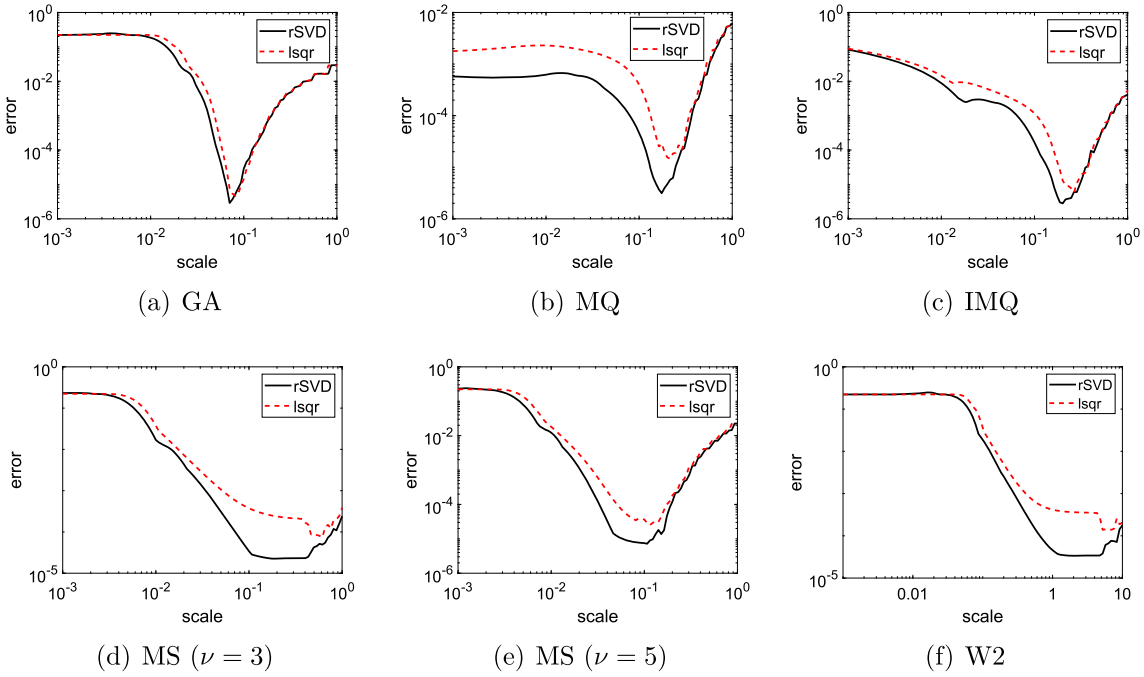


Fig. 5. Example 2: The profiles of the error for randomized SVD and least square solutions.

Table 2
Example 3: Interpolation results using various RBFs with tSVD and trSVD over $L = 900$ evaluation points.

F	n		GA		IMQ		MQ		MS ($\nu = 5$)	
			tSVD	trSVD	tSVD	trSVD	tSVD	trSVD	tSVD	trSVD
F1	500	scale	0.081	0.101	0.241	0.331	0.201	0.291	0.051	0.151
		error	1.15e-3	1.10e-3	2.26e-04	6.36e-04	3.03e-04	6.33e-04	5.37e-04	7.52e-04
	2000	scale	0.071	0.071	0.211	0.191	0.161	0.171	0.061	0.101
		error	1.85e-6	2.99e-6	2.75e-06	2.65e-06	2.96e-06	3.19e-06	4.80e-06	4.79e-06
F2	500	scale	0.191	0.021	0.600	0.460	0.491	0.371	0.270	0.200
		error	3.63e-1	1.58e-1	4.98e-01	4.43e-01	5.25e-01	4.78e-01	6.98e-01	5.54e-01
	2000	scale	0.090	0.080	0.270	0.190	0.190	0.120	0.100	0.060
		error	4.62e-03	3.50e-03	8.76e-03	6.62e-03	1.12e-02	9.57e-03	1.93e-02	1.07e-02

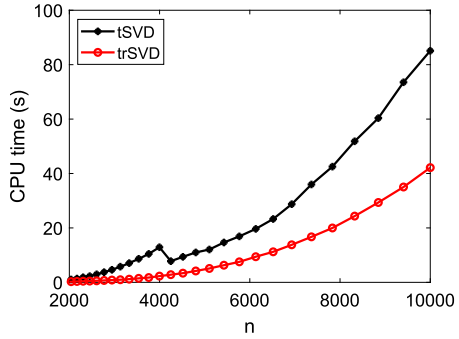
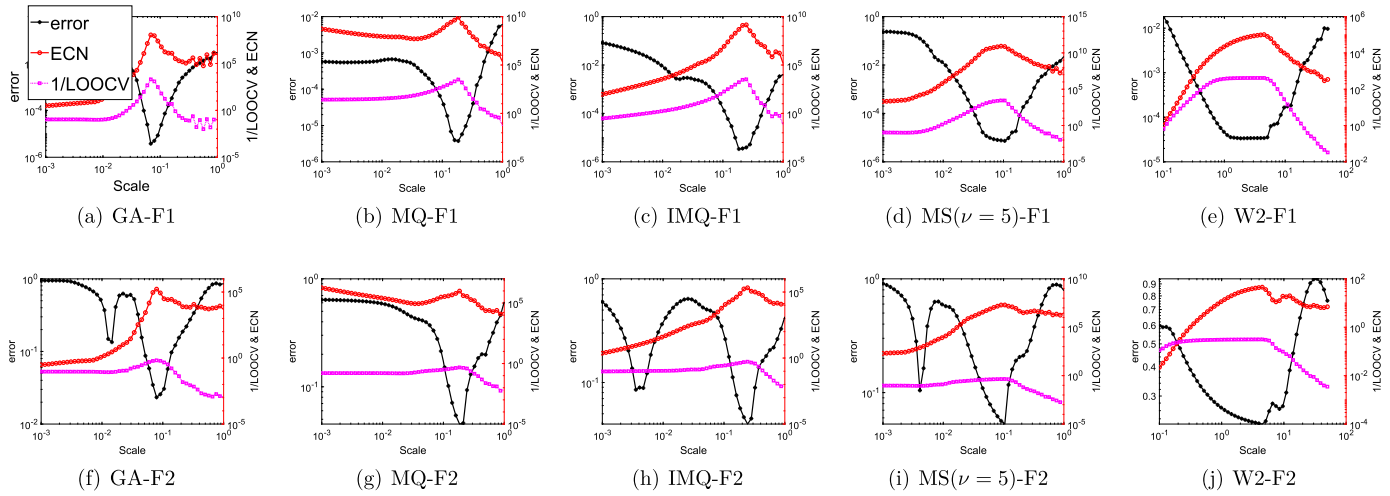


Fig. 6. Example 3: CPU time comparison for tSVD and trSVD.

Fig. 7 displays the interpolation error (ECN) and the reciprocal of LOOCV values (1/LOOCV) as functions of the scale parameter for two test functions, F1 and F2, using various RBFs. It is important to note that the choice of 1/LOOCV instead of LOOCV aims to offer a better scaling adjustment in conjunction with ECN values for the plots. Through our analysis of the experimental results, we have observed that the proposed trSVD method performs well when combined with established scale selection strategies. This integration further enhances the efficiency of the method in searching for an optimal scale parameter. To efficiently determine the scale, the MATLAB minimum finder algorithm `fminbnd` can be employed. It is essential to clarify that, in this context, we utilize the reciprocal of ECN (1/ECN). This choice is made because the goal is to identify the maximum value of ECN.

Example 5. In this example we investigate the effect of the rank target in accuracy and efficiency of the proposed algorithm. The starting and maximum rank targets are set to be 20% up to 80% of total points number. Fig. 8 presents the corresponding results for both GA and MQ RBFs for $n = 500, 2000, 5000$. From these plots, we make the following observations:

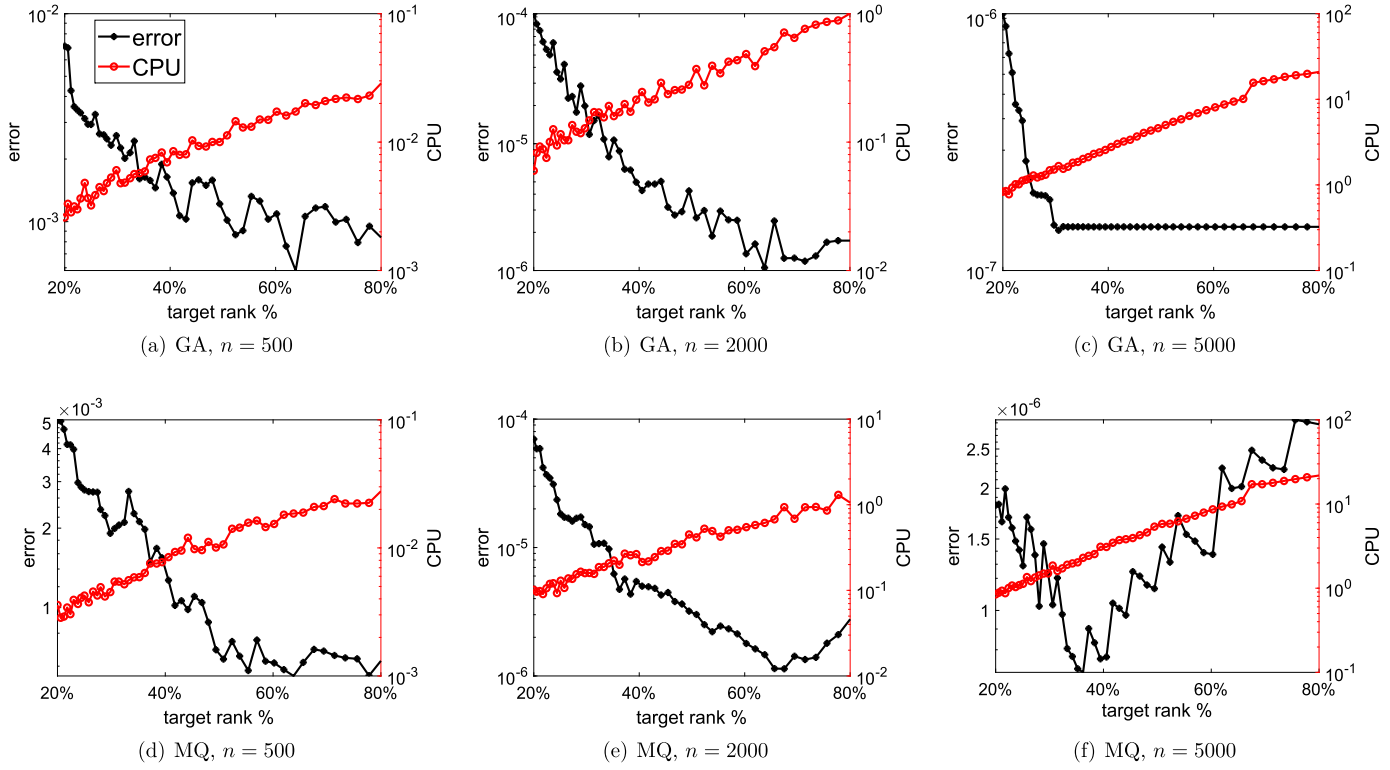
- As the rank target increases, the error decreases for GA RBF, indicating better approximation of the data. However, for $n = 2000$ and 5000, GA RBF reaches a plateau beyond a certain threshold, meaning that further increasing the rank target *does not* significantly improve accuracy.
- For the MQ RBF, increasing the target rank initially reduces the error. However, beyond a certain threshold, adding more singular values and vectors introduces noise or less important information, causing the error to increase.
- The relationship between the target rank and accuracy varies with the number of collocation points. For smaller values of n (e.g., 500, 2000), increasing the target rank can lead to more significant



(a) GA-F1 (b) MQ-F1 (c) IMQ-F1 (d) MS($\nu = 5$)-F1 (e) W2-F1

(f) GA-F2 (g) MQ-F2 (h) IMQ-F2 (i) MS($\nu = 5$)-F2 (j) W2-F2

Fig. 7. Example 4: The profiles of the error with respect to the target LOOCV and ECN over $L = 400$ evaluation points.



(a) GA, $n = 500$ (b) GA, $n = 2000$ (c) GA, $n = 5000$

(d) MQ, $n = 500$ (e) MQ, $n = 2000$ (f) MQ, $n = 5000$

Fig. 8. Example 5: The profiles of the error and CPU time with respect to the target rank for F2.

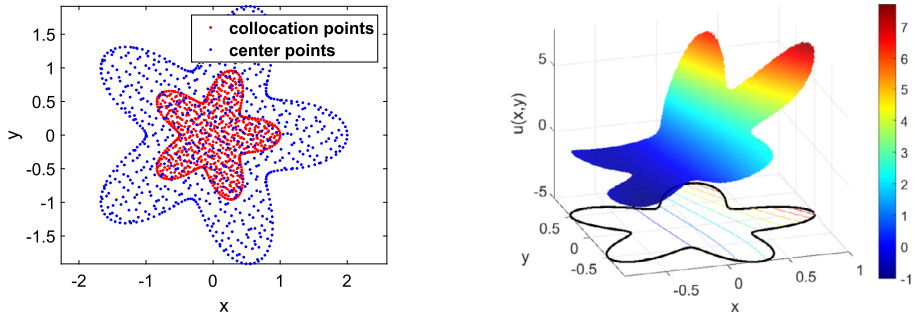
improvements in accuracy compared to the case with $n = 5000$. However, when using a large number of nodes, the computational cost of higher rank approximations may outweigh the marginal accuracy gains. Therefore, it is essential to find a balance between accuracy and computational efficiency based on the specific value of n. For instance, in the GA RBF case with $n = 500$, the error decreases from 0.0069 to 0.00089, resulting in an error difference of 6.00e-3, while for $n = 5000$, the error decreases from 1.00e-6 to 1.44e-7, resulting in an error difference of 8.56e-7. This comparison indicates that as the target rank increases, the absolute improvement in accuracy is more significant for smaller values of n. Therefore, when working with small n values, such as $n = 500$, it is important to exercise caution and carefully select the target rank.

By investigating these trends and differences between GA and MQ RBFs, we observe that the rank target choice significantly impacts the accuracy and efficiency, depending on the RBF type and number of collocation points. Balancing accuracy and computational efficiency is crucial when selecting the rank target, particularly for smaller n values.

5. Applications to PDEs

In this example, we expand the application of the trSVD solver to address the 2D Poisson equation with Dirichlet boundary condition where f^{Ω} and $f^{\partial\Omega}$ of (4) correspond to the given functions associated with the following exact solution:

$$u(x, y) = \sin(\pi x)\cosh(y) + e^{2x+y}, \quad (x, y) \in \overline{\Omega}.$$



(a) $\eta = 2$ (b) exact solution

Fig. 9. Application to PDEs: The profiles of the (a) nodes distribution with $\eta = 2$, and (b) exact solution.

Table 3
Application to PDEs: Details on solution of Poisson equation using the GA RBF.

(n_i, n_b)	η	tSVD		trSVD	
		error	CPU	error	CPU
(600,200)	1	1.28e-04	0.76	4.08e-05	0.32
	2	7.31e-09	0.42	4.79e-08	0.27
	3	1.46e-09	0.64	1.19e-09	0.39
(3000,1000)	1	5.53e-05	81.0	8.31e-05	20.3
	2	4.66e-10	55.5	1.44e-10	32.7
	3	2.26e-10	81.6	1.90e-10	26.6

The mathematical representation of the boundary $\partial\Omega$ is expressed by the following parametric equation:

$$(x, y) = \rho(\theta)(\cos\theta, \sin\theta), \text{ where } \rho(\theta) = \frac{1}{2}(1 + \cos^2 5\theta/2), \quad 0 \le \theta \le 2\pi.$$

This example involves *two* distinct scenarios, differing in terms of the locations of the center points. The first scenario corresponds to the conventional Kansa method [1], where the collocation points and center points are identical. The second scenario involves the fictitious center point method [33], where the collocation and center points are chosen differently. In the latter case, the center nodes are positioned at a magnification factor ($\eta > 1$) of the collocation points. Setting $\eta = 1$ corresponds to the first scenario (conventional Kansa method).

To visualize the distribution of collocation and center points for the star-shaped domain with $\eta = 2$, Fig. 9(a) is provided. The exact solution is illustrated in Fig. 9(b). In order to demonstrate the efficiency and accuracy of the trSVD approach, we compare its results with those of the tSVD method for various numbers of collocation points and magnification factors. In this example, the optimal scale is determined using the LOOCV method linked with MATLAB's search algorithm `fminbnd`, as detailed in Table 3. Note that the CPU time represents the computational duration required to determine the optimal scale value, while n_i and n_b denote the number of interior and boundary points, respectively.

The table reveals that the trSVD and tSVD methods yield comparable accuracy for different numbers of nodes and magnification factors. However, the computational time required by the trSVD solver is significantly lower than that of the tSVD method. By presenting these results, we highlight the effectiveness of the trSVD solver in achieving accurate solutions with reduced computational overhead compared to the conventional tSVD method.

6. Conclusions

This paper investigates the application of randomized SVD (rSVD) for solving interpolation and Poisson problems using kernel matrices derived from radial basis functions (RBFs). Through our experiments, we observed that truncation of rSVD leads to more stable results compared to the non-truncated version. The error curve in relation to the scale also exhibits smoother behavior with truncation, similar to tSVD. Furthermore, our study highlights that trSVD outperforms the least square approach in capturing a more accurate low-rank dimensional representation. To further validate the effectiveness of trSVD, we conducted a comparison between trSVD and tSVD. Our findings demonstrate that trSVD significantly improves efficiency while maintaining comparable accuracy, making it a favorable choice for RBF interpolation tasks. We also conducted an in-depth examination of the trSVD method by exploring its connection to scale or shape parameter strategies, such as LOOCV and ECN, in the search for the optimal scale. Furthermore, we investigated the role of target rank in the trSVD method. Our analysis focused on accuracy and computational time, demonstrating that the selection of the target rank is crucial and sensitive when dealing with a small number of collocation points. However, for a large number of collocation points, the results are more stable. These investigations have been done over various RBFs, as well as over one smooth and one non-smooth test functions. In the final example, we apply the proposed trSVD method to solve a Poisson equation and compare the results with tSVD. The comparison demonstrates similar levels of accuracy between the two methods, with trSVD exhibiting significantly lower CPU time.

In summary, this paper introduces an efficient and accurate algorithm for solving RBF problems, with potential extensions to tackle challenging PDEs. Our approach, based on trSVD, improves efficiency compared to tSVD while maintaining similar levels of accuracy. The incorporation of scale strategies and the investigation of the target rank's role provide valuable insights for practical implementations.

Data availability

I have shared the link to my codes in the manuscript.

Link to the Reproducible Capsule

https://doi.org/10.24433/CO.2648745.v1

Acknowledgements

We sincerely thank the reviewers for their invaluable insights and constructive feedback. The first two authors gratefully acknowledge the financial support of the National Science and Technology Council of Taiwan under grant numbers 109-2221-E002-006-MY3, 111-2811-E-002-062, 111-2221-E-002-054-MY3, 112-2221-E-007-028. We also want to acknowledge the NTUCE-NCREE Joint Artificial Intelligence Research Center and the National Center of High-performance Computing (NCHC) in Taiwan for providing computational and storage resources. The work of R.C. and A.D. has been supported by GNCS-INdAM, and by the Spoke "FutureHPC & BigData" of the ICSC–National Research Center in "High-Performance Computing, Big Data and Quantum Computing", funded by European Union – NextGenerationEU. This research

has been accomplished within the RITA "Research ITalian network on Approximation" and the UMI Group TAA "Approximation Theory and Applications".

References

[1] E. Kansa, Multiquadrics scattered data approximation scheme with applications to computational fluid-dynamics solutions to parabolic, hyperbolic and elliptic partial differential equations, Comput. Math. Appl. 19 (8) (1990) 147–161.

[2] A. Karageorghis, C.S. Chen, Training RBF neural networks for the solution of elliptic boundary value problems, Comput. Math. Appl. 126 (2022) 196–211.

[3] R. Cavoretto, A. De Rossi, W. Erb, Partition of unity methods for signal processing on graphs, J. Fourier Anal. Appl. 27: 66 (2021).

[4] R. Cavoretto, A. De Rossi, A. Sommariva, M. Vianello, RBFCUB: a numerical package for near-optimal meshless cubature on general polygons, Appl. Math. Lett. 125 (2022) 107704.

[5] S.-R. Lin, D.L. Young, C.-S. Chen, Ghost-point based radial basis function collocation methods with variable shape parameters, Eng. Anal. Bound. Elem. 130 (2021) 40–48.

[6] S.L. Brunton, J.N. Kutz, Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control, Cambridge University Press, Cambridge, 2019.

[7] A. Frieze, R. Kannan, S. Vempala, Fast Monte-Carlo algorithms for finding low-rank approximations, J. ACM 51 (6) (2004) 1025–1041.

[8] E. Liberty, F. Woolfe, P-G. Martinsson, Vladimir Rokhlin, M. Tygert, Randomized algorithms for the low-rank approximation of matrices, Proc. Natl. Acad. Sci. 104 (2007) 20167–20172.

[9] N. Halko, P.G. Martinsson, J.A. Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2) (2011) 217–288.

[10] E. Rebrova, Y.-H. Tang, Nonlinear matrix approximation with radial basis function components, arXiv:2106.02018, 2021.

[11] E. Larsson, R. Schaback, Scaling of radial basis functions, arXiv:2210.05617, 2022.

[12] R. Wang, Y. Li, E. Darve, On the numerical rank of radial basis function kernels in high dimensions, SIAM J. Matrix Anal. Appl. 39 (4) (2018) 1810–1835.

[13] Z. Majdisova, V. Skala, Radial basis function approximations: comparison and applications, Appl. Math. Model. 51 (2017) 728–743.

[14] S. Rippa, An algorithm for selecting a good value for the parameter c in radial basis function interpolation, Adv. Comput. Math. 11 (1999) 193–210.

[15] C.-S. Chen, A. Noorizadegan, D.L. Young, C.S. Chen, On the selection of a better radial basis function and its shape parameter in interpolation problems, Appl. Math. Comput. 442 (2023) 127713.

[16] S. Haq, M. Hussain, Selection of shape parameter in radial basis functions for solution of time-fractional Black–Scholes models, Appl. Math. Comput. 335 (2018) 248–263.

[17] S. Haq, M. Hussain, The meshless Kansa method for time-fractional higher order partial differential equations with constant and variable coefficients, Rev. R. Acad. Cienc. Exactas Fís. Nat., Ser. A Mat. 113 (3) (2019) 1935–1954.

[18] R. Schaback, Error estimates and condition numbers for radial basis function interpolation, Adv. Comput. Math. 3 (3) (1995) 251–264.

[19] R. Schaback, Small errors imply large evaluation instabilities, Adv. Comput. Math. 49 (2) (2023) 25.

[20] G.E. Fasshauer, M.J. McCourt, Stable evaluation of Gaussian radial basis function interpolants, SIAM J. Sci. Comput. 34 (2) (2012) A737–A762.

[21] R. Cavoretto, G.E. Fasshauer, M. McCourt, An introduction to the Hilbert-Schmidt SVD using iterated Brownian bridge kernels, Numer. Algorithms 68 (2) (2015) 393–422.

[22] H. Wendland, Scattered Data Approximation, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, 2004.

[23] G.E. Fasshauer, M.J. McCourt, Kernel-Based Approximation Methods Using MATLAB, World Scientific Publishing Co., Inc., Singapore, 2015.

[24] G.E. Fasshauer, Meshfree Approximation Methods with MATLAB, World Scientific Publishing Co., Inc., Singapore, 2007.

[25] B. Fornberg, E. Larsson, N. Flyer, Stable computations with Gaussian radial basis functions, SIAM J. Sci. Comput. 33 (2) (2011) 869–892.

[26] A. Noorizadegan, C.-S. Chen, D.L. Young, C.S. Chen, Effective condition number for the selection of the RBF shape parameter with the fictitious point method, Appl. Numer. Math. 178 (2022) 280–295.

[27] R. Franke, Scattered data interpolation: tests of some methods, Math. Comput. 38 (1982) 181–200.

[28] R. Cavoretto, A. De Rossi, M.S. Mukhametzhanov, Y.D. Sergeyev, On the search of the shape parameter in radial basis functions using univariate global optimization methods, J. Glob. Optim. 79 (2) (2021) 305–327.

[29] F. Marchetti, The extension of Rippa's algorithm beyond LOOCV, Appl. Math. Lett. 120 (2021) 107262.

[30] R. Cavoretto, A. De Rossi, A two-stage adaptive scheme based on RBF collocation for solving elliptic PDEs, Comput. Math. Appl. 79 (2020) 3206–3222.

[31] R. Cavoretto, Adaptive LOOCV-based kernel methods for solving time-dependent BVPs, Appl. Math. Comput. 429 (2022) 127228.

[32] L. Ling, F. Marchetti, A stochastic extended Rippa's algorithm for LpOCV, Appl. Math. Lett. 129 (2022) 107955.

[33] C.S. Chen, A. Karageorghis, F. Dou, A novel RBF collocation method using fictitious centres, Appl. Math. Lett. 101 (2020) 106069.

[34] C.S. Chen, A. Noorizadegan, D.L. Young, C.-S. Chen, On the determination of locating the source points of the MFS using effective condition number, Am. J. Comput. Appl. Math. 423 (2023) 114955.

[35] Z.-C. Li, H.-T. Huang, J.-T. Chen, Y. Wei, Effective condition number and its applications, Computing 89 (1) (2010) 87–112.

[36] J.A. Tropp, A. Yurtsever, M. Udell, V. Cevher, Randomized singleview algorithms for low-rank matrix approximation, arXiv preprint, arXiv:1609.00048, 2016.