J Sci Comput (2016) 66:67–90
DOI 10.1007/s10915-015-0013-8

A Reduced Radial Basis Function Method for Partial Differential Equations on Irregular Domains

Yanlai Chen[1] · **Sigal Gottlieb**[1] · **Alfa Heryudono**[1] · **Akil Narayan**[1]

Received: 7 October 2014 / Revised: 28 January 2015 / Accepted: 13 March 2015
Published online: 25 March 2015

Abstract We propose and test the first Reduced Radial Basis Function Method for solving parametric partial differential equations on irregular domains. The two major ingredients are a stable Radial Basis Function (RBF) solver that has an optimized set of centers chosen through a reduced-basis-type greedy algorithm, and a collocation-based model reduction approach that systematically generates a reduced-order approximation whose dimension is orders of magnitude smaller than the total number of RBF centers. The resulting algorithm is efficient and accurate as demonstrated through two- and three-dimensional test problems.

Keywords Reduced basis method · Radial basis function method · Pseudospectral method · Model reduction

1 Introduction

Parameterized systems are common in science and engineering, and a common situation in multi-query contexts is the need to accurately solve these systems for a variety of different parameter values. This requires a large number of repeated and expensive simulations, frequently rendering the total computational cost prohibitive. To overcome this obstacle while maintaining accurate numerical solutions in real time, the *reduced basis method* (RBM) was

Y. Chen: The research of this author was partially supported by National Science Foundation Grant DMS-1216928.
S. Gottlieb, A. Heryudono: The research of this author was partially supported by AFOSR Grant FA9550-09-1-0208.
A. Heryudono, A. Narayan: The research of this author was partially supported by National Science Foundation Grant DMS-1318427.

✉ Yanlai Chen
yanlai.chen@umassd.edu

[1] Department of Mathematics, University of Massachusetts Dartmouth, 285 Old Westport Road, North Dartmouth, MA 02747, USA

developed [4,43,44,49]. RBMs split the solution procedure to two parts: an offline part where a small number of judiciously-chosen solutions are precomputed via a greedy algorithm, and an online part in which the solution for any new parameter value is efficiently approximated by a projection onto the low-dimensional space spanned by the precomputed solutions. This offline-online decomposition strategy is effective when one can afford an initial (offline) investment of significant computational resources so that certifiably accurate solutions for any other parameter value can be obtained in real-time (online).

The classical RBM was originally developed for use with variational methods for approximating solutions to partial differential equations (PDEs) with affine dependence on the parameter. The most popular of these variational approaches is the *Galerkin* method, derived by positing a solution ansatz in a subspace, and subsequently requiring that the projection of the PDE residual onto the same functional subspace is zero. An alternative approach for the solution of PDEs is to require that the PDE residual vanish at some predetermined collocation points. These *collocation* methods are attractive because they are frequently easier to implement compared to Galerkin methods, particularly for time-dependent nonlinear problems [30,51,54]. In [12], two of the authors developed a RBM suitable for collocation methods, and thus introduced a reduced collocation method (RCM). The RCM is extremely efficient and provides a reduced basis strategy for practitioners who prefer a collocation approach for solving PDEs, rather than a Galerkin approach. Indeed, one of the two approaches in [12], the empirical reduced collocation method, eliminates a potentially costly online procedure that is usually necessary for non-affine problems using a Galerkin approach. The RCM's efficiency matches (or, for non-affine problems, exceeds) that of traditional Galerkin-based RBM approaches.

The RCM was developed for use with traditional collocation methods, particularly for pseudospectral methods [30]. However, such collocation methods require a very structured grid which may be inconvenient when the spatial domain associated with the PDE has an irregular shape. When an irregular geometry is present, meshfree methods are a viable choice when compared to traditional meshed methods. Meshfree methods eliminate the need for finite-element-like meshes or adherence to symmetrical grid layouts, and have implementation costs that scale well with the spatial dimension; these properties are advantageous when compared to more standard mesh-based discretization approaches.

One particular mesh-free collocation method that we explore here is based on radial basis functions (RBFs). RBF methods have been widely used for scattered data interpolation and approximation in high dimensions [9,19,58]. RBF collocation methods for elliptic PDEs, based on global, non-polynomial interpolants, have been developed since the early 90s [33–35]. RBF methods are collocation methods, implemented on scattered sets of collocation sites (commonly called *centers*) and, unlike traditional pseudospectral methods, are not tied to a particular geometric structure. We employ a spatially-local variant of more traditional global RBF methods. RBF methods usually approximate differential operators with global stencils, but we employ a local approach which, inspired by its relation to finite-difference (FD) methods, is called RBF-FD methods [22,52,53,59]. The RBF-FD method has the advantage of retaining high-order accuracy while improving computational efficiency by forming sparse operator matrices.

In this paper we extend the RBM strategy to include meshfree collocation methods, in particular RBF and RBF-FD methods. We develop an algorithm that inherits the strengths of model-order reduction and geometric flexibility from RBM/RCM and RBF methods, respectively. This novel Reduced Radial Basis Function Method (R^2BFM) is capable of achieving orders-of-magnitude speedup for solving parameterized problems on irregular

domains. Our numerical results demonstrate exponential convergence with respect to the number of precomputed solutions.

The paper is organized as follows: In Sect. 2 we review both the RBM and RCM order-reduction algorithms. In Sect. 3 we present the particular radial basis function method we adopt (RBF-FD). The novel R^{2}BFM algorithm is proposed in Sect. 4, and we present numerical experiments that illustrate its performance for 2D- and 3D-problems.

2 The Least Squares Reduced Collocation Method

In this section, we briefly review the reduced basis and least-squares reduced collocation methods originally introduced in [12]. RBMs aim to efficiently solve parameterized PDEs with certifiable error bounds. Repeatedly solving the full PDE for several values of the parameter is computationally onerous when the PDE itself is so complicated that a single solve is expensive. The RBM framework mitigates this cost first by carefully choosing a small set of parameter values at which the expensive PDE model is solved and stored. Once this expensive "offline" procedure is completed, then the "online" RBM algorithm computes the PDE solution at any new parameter value as a linear combination of the precomputed and stored solutions. The offline-online decomposition details of the algorithm ensure (1) that this procedure is accurate and (2) that the cost of the new parameter value solve is orders-of-magnitude smaller than a standard PDE solve.

The reduced basis method was invented in the late 1970s for nonlinear structural analysis [1,41,43], and more broadly developed therein [2,4,20,39,44,48]. Recently, it has been systematically analyzed and applied to a wide variety of problems, see e.g. [15,28,29,42,49,55,56,60,61], with [50] containing extensive references. The RBM algorithm is traditionally applied to Galerkin discretizations of PDEs. However, recent work in [12,13] develops a robust framework for applying the RBM algorithm to collocation discretizations of PDEs. Since radial basis function discretization methods are collocative, we will later in the paper apply the RCM developed in [12], in particular the Least Squares RCM. The other RCM, the Empirical RCM [12,13], employs the idea of the Empirical Interpolation Method [3,27] to identify a reduced set of points in the *physical* domain Ω on which to enforce the PDE. We remark that our goal is not to facilitate the offline-online decomposition through a separable form as is typically the case [3,17,27]. Indeed, in our setting we do not address the problem of approximating the parametric operator by a linear combination of high-dimensional basis functions with easily-computed parameter-dependent coefficients. However, EIM methods together with least squares approaches have appeared in other model reduction settings, see e.g. the recent paper [10].

To describe the Least Squares RCM (LSRCM) algorithm, we begin with a linear parametrized PDE of the form

$$\mathbb{L}(\mu)\, u_\mu(\underline{x}) = f(\underline{x};\mu), \qquad \underline{x} \in \Omega \subset \mathbb{R}^d \tag{1}$$

with appropriate boundary conditions. Here, $\underline{x}$ is the spatial variable, μ is the parameter, $\mathbb{L}$ is the differential operator that depends on the parameter, f is a forcing function, and u is the unknown solution. The dimension of the spatial variable obeys $d \leq 3$ for most physical problems of interest, and so we will adopt the notation $\underline{x} = (x, y, z)^T$ for the components of $\underline{x}$. For any parameter $\mu = (\mu^1, \ldots, \mu^p) \in \mathcal{D} \subset \mathbb{R}^p$, a prescribed p-dimensional real parameter domain, we introduce a *discrete* differentiation operator $\mathbb{L}_\mathcal{N}(\mu)$ approximating $\mathbb{L}(\mu)$ such that

$$\mathbb{L}_{\mathcal{N}}(\mu)\, u_{\mu}^{\mathcal{N}}(\underline{x}_j) = f(\underline{x}_j; \mu), \tag{2}$$

is satisfied *exactly* on a given set of $\mathcal{N}$ collocation points $C^{\mathcal{N}} = \{\underline{x}_j\}_{j=1}^{\mathcal{N}}$. We assume that this discretization is such that the resulting approximate solution $u_{\mu}^{\mathcal{N}}$ is highly accurate for *all* parameter values $\mu \in \mathcal{D}$, and refer to $u_{\mu}^{\mathcal{N}}$ as the "truth approximation". With such a robust requirement on accuracy of the truth solution, $\mathcal{N}$ will be sufficiently large so that solving (2) is a relatively expensive operation that we wish to avoid performing too many times.

With this setup, the RCM algorithm in [12] proceeds first with an *offline* stage where the algorithm is sown with a small number of expensive truth solves of (2) along with some preprocessing of reduced operators, followed by an *online* stage where computational savings are reaped whenever the algorithm is queried for the solution at a new parameter value.

2.1 The Offline Stage: Choosing Parameter Values

The first main goal of the offline stage is to choose N parameter values $\mu_1, \ldots, \mu_N$ with $N \ll \mathcal{N}$ such that the corresponding truth solution ensemble $u_{\mu_1}^{\mathcal{N}}, u_{\mu_2}^{\mathcal{N}}, \ldots, u_{\mu_N}^{\mathcal{N}}$ has a span that accurately approximates $u_{\mu}^{\mathcal{N}}$ for any $\mu \in \mathcal{D}$. The truth solutions $u_{\mu_j}^{\mathcal{N}}$ are frequently called "snapshots". That it is even possible to generate such a collection of snapshots has been theoretically verified for several differential operators of interest [7,8,37]. The algorithmic way in which this reduced set of parameters μ is chosen is via a greedy computation that successively chooses parameter values maximizing an error estimate. The definition of this error estimate hinges on the formulation of a reduced approximation: For any $1 \le n \le N$, we seek the reduced solution $u_{\mu}^{(n)}$ defined as

$$u_{\mu}^{(n)}(\cdot) = \sum_{j=1}^{n} c_j(\mu) u_{\mu_j}^{\mathcal{N}}(\cdot), \tag{3}$$

where the coefficients $\mathbf{c} = \left\{c_j(\mu)\right\}_{j=1}^{n}$ are computed by solving a least-squares residual problem on the truth solution nodes $\underline{\mathbf{x}} = C^{\mathcal{N}}$:

$$\mathbf{c}(\mu) = \underset{\mathbf{d}\in\mathbb{R}^n}{\arg\min} R_n(\mu; \mathbf{d}), \quad \text{with} \tag{4a}$$

$$R_n(\mu; \mathbf{d}) = \left\| f(\underline{\mathbf{x}}; \mu) - \sum_{j=1}^{n} d_j(\mu) \mathbb{L}_{\mathcal{N}}(\mu) u_{\mu_j}^{\mathcal{N}}(\underline{\mathbf{x}}) \right\|_{\ell^2}. \tag{4b}$$

The ultimate goal of the error estimate is to approximate the error $u_{\mu}^{(n)} - u_{\mu}^{\mathcal{N}}$ without directly forming the truth solution $u_{\mu}^{\mathcal{N}}$. We denote this error estimate by $\Delta_n(\mu)$ and define it as:

$$\Delta_n(\mu; \mathbf{c}(\mu)) = \frac{R_n(\mu; \mathbf{c}(\mu))}{\sqrt{\beta_{LB}(\mu)}}. \tag{5}$$

Above, $\beta_{LB}(\mu)$ is a lower bound for the smallest eigenvalue of $\mathbb{L}_{\mathcal{N}}(\mu)^T \mathbb{L}_{\mathcal{N}}(\mu)$, which effectively translates residuals into a bound for the actual errors. The accurate, computationally $\mathcal{N}$-independent computation of β_{LB} (and thus of Δ_n) is in general one of the major difficulties in RBM algorithms, see e.g. [14,31,32]. These ingredients and an $\mathcal{N}-$independent evaluation of $R_n(\mu; \mathbf{c})$ allow us finally to define how the parameter values are chosen through a greedy approach, shown in Algorithm 1. If Δ_n can be computed in a $\mathcal{N}$-independent fashion, then this greedy approach is computationally efficient. Algorithm 1 presumes the ability

Algorithm 1 Outline of the offline RBM/RCM greedy algorithm

1. $W_1 = \text{span}\left\{u^{\mathcal{N}}(\mu_1)\right\}$ (with μ_1 arbitrarily chosen).
2. For $i = 2, \ldots, N$ do:
a). $\mu_i = \arg\max_{\mu \in \mathcal{D}} \Delta_{i-1}(\mu)$. **b).** $W_i = \text{span}\left\{u_{\mathcal{N}}(\mu_j),\ j \in \{1, \ldots, i\}\right\}$.

to extremize $\Delta_n(\mu)$ over the continuous parameter domain $\mathcal{D}$. In practice, one instead discretizes the parameter space, and scans it to generate the "best" reduced solution space. The first parameter value μ_1 is randomly chosen, and the accurate truth solution $u^{\mathcal{N}}_{\mu_1}$ is computed and stored, forming the first iterate of the solution space, W_1. Then for $n \geq 2$, we select the parameter value whose truth solution $u^{\mathcal{N}}_{\mu}$ is worst approximated by the reduced solution $u^{(n)}_{\mu}$; this is accomplished with the error estimator $\Delta_n(\mu)$. The formulation of $\Delta_n(\mu)$ is rigorous, and so one can certify the maximum error over the parameter domain, stopping the iteration whenever this error reaches a desired tolerance level. In practice, a variant of the modified Gram-Schmidt transformation is applied to generate a more stable basis of W_n. An acceptable, certifiable error tolerance is usually reached with only $N \ll \mathcal{N}$ truth snapshots.

2.2 The Offline Stage: Formation of Reduced-Order Operators

Assume that the snapshots $u^{\mathcal{N}}_{\mu_n}$ for $n = 1, \ldots, N$ are precomputed and stored from the previous section. The computation of the reduced-order solution $u^{(N)}_{\mu}$ and the residual $R_N(\mu; \mathbf{d})$ given by (3) and (4) clearly requires $\mathcal{O}(\mathcal{N})$ operations as written because we must compute $\mathbb{L}_{\mathcal{N}}(\mu)u^{\mathcal{N}}_{\mu_j}$ for each new parameter value μ. One condition that breaks this $\mathcal{N}$-dependence is the assumption that the operator $\mathbb{L}(\mu)$ and the forcing function $f(\underline{x}; \mu)$ have *affine* dependence on the parameter, i.e., that

$$\mathbb{L}(\mu) = \sum_{q=1}^{Q_a} a^{\mathbb{L}}_q(\mu)\mathbb{L}_q, \qquad f(\cdot; \mu) = \sum_{q=1}^{Q_f} a^f_q(\mu) f_q(\cdot) \tag{6}$$

where the functions $a^{\mathbb{L}}_q(\mu)$ and $a^f_q(\mu)$ are scalar-valued and $\underline{x}$-independent, and the composite operators $\mathbb{L}_q$ and f_q are μ-independent. Many parameterized operators $\mathbb{L}(\mu)$ of interest do satisfy this assumption and there are effective strategies for approximating non-affine operators and functions by affine ones [3,27]. We assume hereafter that the operator $\mathbb{L}(\mu)$ and the forcing function $f(\cdot; \mu)$ have affine dependence on μ.

The affine dependence assumption allows us to precompute several quantities for use both later in the online stage, and in the offline process of selecting the N snapshots. The discrete truth operator $\mathbb{L}_{\mathcal{N}}$ and forcing function $f\left(\underline{\mathbf{x}}; \mu\right)$ likewise have an affine decomposition

$$\mathbb{L}_{\mathcal{N}}(\mu) = \sum_{q=1}^{Q_a} a^{\mathbb{L}}_q(\mu)\mathbb{L}_{\mathcal{N},q}, \qquad f(\underline{\mathbf{x}}; \mu) = \sum_{q=1}^{Q_f} a^f_q(\mu) f_q(\underline{\mathbf{x}}).$$

Once the parameter values $\mu_1, \ldots, \mu_N$ are chosen, the following quantities may be computed and stored in the offline stage:

$$(M_{r,s})_{j,k} \triangleq \left(\mathbb{L}_{\mathcal{N},r} u^{\mathcal{N}}_{\mu_j}\right)^T \left(\mathbb{L}_{\mathcal{N},s} u^{\mathcal{N}}_{\mu_k}\right), \quad 1 \le j,k \le N, \quad 1 \le r \le s \le Q_a, \tag{7a}$$

$$(g_{q,r})_j \triangleq \left(\mathbb{L}_{\mathcal{N},q} u^{\mathcal{N}}_{\mu_j}\right)^T f_r(\underline{\mathbf{x}}), \qquad j = 1,\ldots,N, \quad q = 1,\ldots,Q_a, \quad r = 1,\ldots,Q_f. \tag{7b}$$

The resulting collection of matrices $\mathbf{M}_{r,s}$ are each $N \times N$, and the vectors $\mathbf{g}_q$ are $N \times 1$. Similar pre-computations are carried out to achieve $\mathcal{N}$-independent evaluation of $R_n(\mu)$; see [12] for details.

2.3 The Online Stage: Computing a Fast Solution at a New Parameter Location

All the operations that have $\mathcal{O}(\mathcal{N})$ complexity were completed in the previous offline sections. During the online stage, all operations are $\mathcal{N}$-independent. Given a new parameter value μ^*, we wish to compute the LSRCM approximation $u^{(N)}_{\mu^*}$ to the truth approximation $u^{\mathcal{N}}_{\mu^*}$. This approximation is given by the coefficients c_j in (4a) with $n = N$. The formuluation (4a) is a standard least-squares problem for the unknown coefficients. The coefficients $\mathbf{c}$ from (4a) can be computed using the normal equations and (7); we need only solve the following square, linear system:

$$\mathbf{K}(\mu)\mathbf{c} = \mathbf{h}(\mu),$$
$$\mathbf{K}(\mu) = \sum_{r,s=1}^{Q_a} a^{\mathbb{L}}_r(\mu^*)\, a^{\mathbb{L}}_s(\mu^*)\, \mathbf{M}_{r,s}, \qquad \mathbf{h}(\mu) = \sum_{q=1}^{Q_a}\sum_{r=1}^{Q_f} a^{\mathbb{L}}_q(\mu^*)\, a^{f}_r(\mu^*)\, \mathbf{g}_{q,r}. \tag{8}$$

The linear system (8) is invertible and $N \times N$ so the coefficients $\mathbf{c}$ may be computed in an efficient and straightforward manner. We emphasize that algorithmic methods to form and solve the above system have computational complexities that are *independent* of the truth discretization parameter $\mathcal{N}$. In this way, for each new μ^*, we can use $\mathcal{N}$-independent operations to compute the LSRCM approximation $u^{(N)}_{\mu^*}$. For several problems of interest [12], the RCM solution $u^{(N)}_{\mu^*}$ converges spectrally to the truth solution $u^{\mathcal{N}}_{\mu^*}$ with respect to N.

3 The Local Radial Basis Function Method

The truth solution for the R^{2}BF method will be a large-stencil finite difference solution that is based on a meshfree radial basis function collocation method. Like finite difference methods, this RBF analogue has the advantages of low computational cost due to relatively sparse matrices. It also features flexibility in distributing collocation points for discretization on an irregular domain. The method is widely known as RBF in finite difference mode (RBF-FD) [53,59], or RBF differential quadrature (RBF-DQ) [52].

The essence of this approach consists in defining differentiation matrices to transform a linear PDE operator $\mathbb{L}$ into a linear algebra problem $\mathbb{L}_{\mathcal{N}}$. What differentiates one method from the other is the methodology of discretization. The local RBF discretization step consists of two ingredients: laying out points in the irregular domain and forming the differentiation matrices via high order local interpolants. In the following subsections we will describe our approach for each of these ingredients.

3.1 Background

3.1.1 Global Approximation with Radial Basis Functions

One of the simplest scenarios in which RBFs are employed is in global approximation methods. The basic idea is to form an interpolant that approximates a function $u(\underline{x})$ based on data values u_k given at scattered nodes $C^{\mathcal{N}} = \{\underline{x}_k\}_{k=1}^{\mathcal{N}}$. The RBF interpolant has the form

$$s(\underline{x}) = \sum_{k=1}^{\mathcal{N}} \lambda_k \phi\left(\varepsilon \|\underline{x} - \underline{x}_k\|\right), \tag{9}$$

where $\underline{x}$ denotes a point in $\mathbb{R}^d$, $\phi(r)$ is a radial basis function, ε is a shape parameter, and $\|\cdot\|$ is Euclidean distance. For the moment, we assume that the nodes $\underline{x}_k$ (sometimes called *centers* or *grid points*) are *a priori* prescribed.

The function $\phi(\cdot)$ depends on the distance between points and not necessarily their orientation; one may generalize this approach to non-radial kernel functions but we stick to radial approximations in this paper. The choice of radial basis function $\phi(r)$ is clearly important since it directly affects the actual reconstruction $u(\underline{x})$. Common choices of $\phi(r)$ are:

- Infinitely smooth functions: Multiquadrics (MQ) $\phi(r) = \sqrt{1+r^2}$, Inverse Multiquadrics (IMQ) $\phi(r) = \frac{1}{\sqrt{1+r^2}}$, and Gaussian (GA) $\phi(r) = e^{-r^2}$;
- Piecewise smooth: Cubic $\phi(r) = r^3$, thin plate splines $\phi(r) = r^2 \ln(r)$;
- Wendland's compactly supported piecewise polynomials [57].

For a more comprehensive list, we refer to [9,19,58]. In this paper we use the inverse multiquadrics (IMQ).

In order to solve for the coefficients $\boldsymbol{\lambda} = \{\lambda_j\}_{j=1}^{\mathcal{N}}$, we collocate the interpolant $s(\underline{x})$ to satisfy the interpolation conditions $s(\underline{x}_k) = u_k$, $k = 1, \ldots, \mathcal{N}$. This results in a system of linear equations

$$\begin{bmatrix} \phi(\varepsilon\|\underline{x}_1 - \underline{x}_1\|) & \cdots & \phi(\varepsilon\|\underline{x}_1 - \underline{x}_{\mathcal{N}}\|) \\ \vdots & \ddots & \vdots \\ \phi(\varepsilon\|\underline{x}_{\mathcal{N}} - \underline{x}_1\|) & \cdots & \phi(\varepsilon\|\underline{x}_{\mathcal{N}} - \underline{x}_{\mathcal{N}}\|) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_{\mathcal{N}} \end{bmatrix} = \begin{bmatrix} u_1 \\ \vdots \\ u_{\mathcal{N}} \end{bmatrix} \Longrightarrow \boldsymbol{\Phi}\boldsymbol{\lambda} = \mathbf{u}, \tag{10}$$

where the matrix $\boldsymbol{\Phi}$ is symmetric with entries $\Phi_{ij} = \phi(\varepsilon\|\underline{x}_i - \underline{x}_j\|)$ and $\mathbf{u} = [u_1, \ldots, u_{\mathcal{N}}]^T$. The interpolation matrix $\boldsymbol{\Phi}$ is guaranteed to be non-singular for many choices of ϕ [40].

Computing derivatives of the RBF interpolant $s(\underline{x})$ in (9) is a straightforward process, which can be simply done by summing the weighted derivatives of the basis functions. For example, with x the first component of the vector $\underline{x} \in \mathbb{R}^d$, then

$$\frac{\partial}{\partial x} s(\underline{x}) = \sum_{k=1}^{\mathcal{N}} \lambda_k \frac{\partial}{\partial x} \phi\left(\varepsilon \|\underline{x} - \underline{x}_k\|\right), \tag{11}$$

computes the first derivative of $s(\underline{x})$ with respect to x at any location $\underline{x}$. Since $\boldsymbol{\lambda} = \boldsymbol{\Phi}^{-1}\mathbf{u}$, then (11) can be written more compactly as

$$\frac{\partial}{\partial x} s(\underline{x}) = \boldsymbol{\Phi}_x \boldsymbol{\Phi}^{-1}\mathbf{u} \triangleq \boldsymbol{D}_x \mathbf{u}, \tag{12}$$

where $\boldsymbol{\Phi}_x = \left[\frac{\partial}{\partial x}\phi(\varepsilon\|\underline{x} - \underline{x}_1\|), \ldots, \frac{\partial}{\partial x}\phi(\varepsilon\|\underline{x} - \underline{x}_{\mathcal{N}}\|)\right]$. The matrix $\boldsymbol{D}_x = \boldsymbol{\Phi}_x\boldsymbol{\Phi}^{-1}$ has $\mathcal{N}$ columns and is commonly called the RBF "differentiation matrix". The number of rows in $\boldsymbol{D}_x$

depends on where the differentiated interpolant $\frac{\partial}{\partial x}s$ should be evaluated, and we commonly want to evaluate on the same collocation points $C^{\mathcal{N}}$. Thus, the matrix $\boldsymbol{\Phi}_x$ is $\mathcal{N}\times\mathcal{N}$ with entries

$$\boldsymbol{\Phi}_x = \begin{bmatrix} \phi_{x_1}(\varepsilon\|\underline{x}_1 - \underline{x}_1\|) & \cdots & \phi_{x_1}(\varepsilon\|\underline{x}_1 - \underline{x}_{\mathcal{N}}\|) \\ \vdots & \ddots & \vdots \\ \phi_{x_{\mathcal{N}}}(\varepsilon\|\underline{x}_{\mathcal{N}} - \underline{x}_1\|) & \cdots & \phi_{x_{\mathcal{N}}}(\varepsilon\|\underline{x}_{\mathcal{N}} - \underline{x}_{\mathcal{N}}\|) \end{bmatrix}, \tag{13}$$

where, in a slight abuse of notation, $\phi_{x_j}\left(\varepsilon\left\|\underline{x}_j - \underline{x}_k\right\|\right) = \frac{\partial}{\partial x_j}\phi\left(\varepsilon\left\|\underline{x}_j - \underline{x}_k\right\|\right)$ is the partial derivative of the shape function with respect to the first component x_j of $\underline{x}_j = (x_j, y_j, z_j)$. Higher order differentiation matrices or derivatives with respect to different variables (e.g $\boldsymbol{D}_y$, $\boldsymbol{D}_z$, $\boldsymbol{D}_{xx}$, etc) can be computed in the same manner.

Note that in general there are several theoretical and implementation aspects of global RBF approximation that are important to consider in practice. As an example, in cases where data values u_k are sampled from smooth functions, the constructed interpolant $s(\underline{x})$ can be highly accurate if infinitely smooth basis function $\phi(r)$ with small values of shape parameters ε are used. However, as ε becomes smaller, the basis functions becomes "flatter" [16], which leads to an ill-conditioned matrix $\boldsymbol{\Phi}$. Techniques to mitigate these kinds of interpolation instability issues include contour integration [25], QR Decomposition [18,21,24], Hilbert-Schmidt SVD methods [11], and SVD methods utilizing rational interpolants [26]. RBF methods that do not employ any of those techniques are usually called "RBF-Direct". In this work, we only utilize RBF-Direct approximation methods to form the RBM truth approximation for accurately solving problems on irregular geometries. The use of more stable RBF interpolant algorithms is not the central focus of this work and will be left for future study.

The interpolation instability issues result not only from linear algebraic considerations. A judicious placement of nodes plays a crucial role through the classical problem of interpolation stability, as measured by Lebesgue constants and manifested through the Runge phenomenon. In one-dimensional cases, oscillatory behavior near the boundaries do appear when equally-spaced points are used as $\mathcal{N}$ becomes larger. This empirical observation is supported by potential-theoretic analysis [46,47]. A stable approximation scheme for analytic functions on equally-spaced samples cannot converge exponentially [45]. For higher dimensional cases, the precise distribution of scattered points is not yet well-understood although some numerical evidence is shown in [21].

For this paper and for modest size $\mathcal{N}$, we use an algorithm that is based on the power function [38] to select optimal distribution of nodes. For this purpose, we will assume that ϕ is a positive-definite function, meaning that for any collection of $\mathcal{N}$ distinct nodes $\underline{x}_k$, the interpolation matrix $\boldsymbol{\Phi}$ defined in (10) satisfies

$$\mathbf{v}^T\boldsymbol{\Phi}\mathbf{v} > 0, \qquad \forall\ \mathbf{v} \in \mathbb{R}^{\mathcal{N}} \tag{14}$$

We make this assumption for two reasons: it ensures a unique solution to (10), and it allows us to construct a well-defined discrete norm on vectors. The IMQ basis function that we use here satisfies the positive-definite condition.

3.1.2 The Native Space Norm

Given a positive-definite function ϕ, we introduce the collection of functions ϕ^x centered at every location in the physical domain:

$$\mathcal{V} = \left\{\phi^{\underline{x}}(\cdot)\ \middle|\ \underline{x} \in \Omega\right\}, \qquad \phi^{\underline{x}}(\underline{y}) \triangleq \phi\left(\varepsilon\left\|\underline{y} - \underline{x}\right\|\right) \tag{15}$$

and define a proper norm on any function formed from a finite linear combination of elements in $\mathcal{V}$:

$$u(\cdot) = \sum_{k=1}^{\mathcal{N}} \lambda_k \phi^{\underline{x}_k}(\cdot), \qquad \|u\|_{\mathcal{H}}^2 \triangleq \sum_{1 \le j,k \le \mathcal{N}} \lambda_j \lambda_k \phi^{\underline{x}_k}(\underline{x}_j). \tag{16}$$

Above, we have used $\underline{x}_k$ to indicate any selection of distinct points from Ω. The *native space* associated with the function ϕ is the $\|\cdot\|_{\mathcal{H}}$ closure of finite linear combinations of elements from $\mathcal{V}$. The native space is a Hilbert space and we will denote it by $\mathcal{H}$. This construction is relatively abstract, but it is wholly defined by ϕ, and one can characterize $\mathcal{H}$ as being equivalent to more standard L^2 Sobolev spaces by considering the decay rate of the Fourier transform of ϕ [58].

The inner product on $\mathcal{H}$ for $u = \sum_k \lambda_k \phi^{\underline{x}_k}$ and $v = \sum_k \rho_k \phi^{\underline{x}_k}$ is given by

$$\langle u, v \rangle_{\mathcal{H}} \triangleq \sum_{1 \le j,k \le \mathcal{N}} \lambda_j \rho_k \phi^{\underline{x}_k}(\underline{x}_j). \tag{17}$$

Given data $\boldsymbol{u}$ as in (10), we will use the notation $\|\boldsymbol{u}\|_{\mathcal{H}}$ to denote the corresponding $\mathcal{H}$-norm of the global interpolant (9) and from (16) this norm is

$$\|\boldsymbol{u}\|_{\mathcal{H}}^2 \triangleq \boldsymbol{\lambda}^T \boldsymbol{\Phi} \boldsymbol{\lambda} = \boldsymbol{u}^T \boldsymbol{\Phi}^{-1} \boldsymbol{u} = \left\| \boldsymbol{S}^{-1} \boldsymbol{u} \right\|^2, \tag{18}$$

with $\|\boldsymbol{v}\|$ the standard Euclidean norm on vectors $\boldsymbol{v}$, and $\boldsymbol{S}$ is any matrix satisfying $\boldsymbol{S}\boldsymbol{S}^T = \boldsymbol{\Phi}$. For concreteness, we will take $\boldsymbol{S}$ to be the positive-definite square root of $\boldsymbol{\Phi}$, i.e. since from (14) $\boldsymbol{\Phi}$ is diagonalizable with positive spectrum:

$$\boldsymbol{\Phi} = \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{V}^T \qquad \Longrightarrow \qquad \boldsymbol{S} = \boldsymbol{V} \sqrt{\boldsymbol{\Lambda}} \boldsymbol{V}^T$$

However this choice is not necessary and in what follows one can replace $\mathbf{S}$ by, e.g., the Cholesky factor for $\boldsymbol{\Phi}$.

The norm on vectors $\|\cdot\|_{\mathcal{H}}$ defined above will be the RBF-analogue of a continuous norm in our RBM collocation framework.

3.1.3 Choosing Nodes: The Discrete Power Function Method

Traditional collocation methods, such as Fourier or Chebyshev pseudospectral methods, require a particular grid structure. Avoiding this restriction on irregular geometries is one of the major reasons why we turn to radial basis functions, which are mesh-free. However, it is well known that the accuracy of RBF methods is heavily influenced by the location of the centers $\underline{x}_k$. While some RBF nodal arrays are known to produce accurate reconstructions, these are mainly restricted to canonical domains—tensor product or symmetric domains.

We are interested in computations on irregular domains, and therefore require a method for selecting region-specific nodes that will enhance the accuracy of the reconstruction. This is important since the inaccuracy of the RBF truth approximation will lead directly to that of the LSRCM solution. To generate nodes we make use of the Power Function Method applied on a discrete candidate set. We present a short discussion of this method in an effort to keep our presentation self-contained. Simplistic and effective, it relates to the reduced basis method by employing the same type of greedy algorithm. The interested reader may refer to [38] for a thorough discussion of this method; we provide an alternate description below in the context of the reduced basis framework.

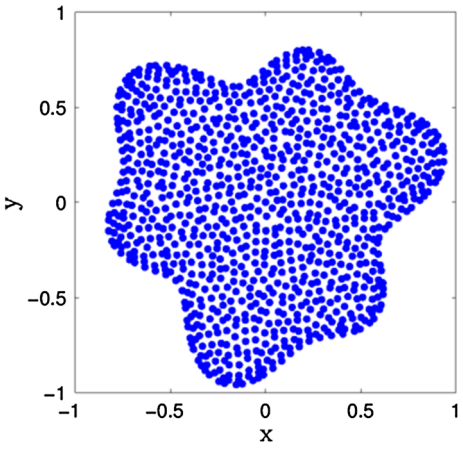


Fig. 1 RBF points resulting from the Power Function method

Recall that Ω represents the physical domain, and that $\mathcal{V}$ from (15) is the collection of RBF shape functions centered at every point in Ω. We consider this space as a collection of parameterized functions; the parameter is the nodal center $\underline{x}$.

The Power Function method selects RBF nodal centers by forming a reduced basis approximation to $\mathcal{V}$. From Algorithm 1, we see that the reduced basis method greedily forms an approximation space by computing parametric values that maximize an error criterion. This error criterion is defined in terms of an error norm. For the space $\mathcal{V}$, it is natural to choose this norm to be $\|\cdot\|_{\mathcal{H}}$, the norm on the ϕ-native space $\mathcal{H}$. Then applying an RBM offline selection of parameter values from $\mathcal{V}$ results in following optimization scheme, which mirrors Algorithm 1:

$$\underline{x}_{n+1} = \underset{\underline{x}\in\Omega}{\operatorname{argmax}}\, \operatorname{dist}_{\mathcal{H}}\left(\phi^{\underline{x}}, \mathcal{V}_n\right) = \underset{\underline{x}\in\Omega}{\operatorname{argmax}}\, \left\|\phi^{\underline{x}} - P_{\mathcal{V}_n}\phi^{\underline{x}}\right\|_{\mathcal{H}}, \tag{19}$$

$$\mathcal{V}_{n+1} = \operatorname{span}\left\{\phi^{\underline{x}_1}, \ldots, \phi^{\underline{x}_{n+1}}\right\}, \tag{20}$$

where $P_{\mathcal{V}_n}$ is the $\mathcal{H}$-orthogonal projector onto $\mathcal{V}_n$. To start the iteration, $\mathcal{V}_0 = \{0\}$ is the trivial subspace. Since the inner product of $\mathcal{H}$ is (17) and ϕ is a radial kernel, then

$$\left\langle \phi^{\underline{x}}(\cdot), \phi^{\underline{y}}(\cdot)\right\rangle_{\mathcal{H}} = \phi^{\underline{x}}(\underline{y}) = \phi^{\underline{y}}(\underline{x}). \tag{21}$$

Thus, all the inner products and norms for the optimization can be computed simply by evaluating the shape function ϕ. The optimization (19) is the Power Function method of [38].

We implement this method on finite candidate sets, e.g. substituting Ω with $Y^M = \left\{\underline{y}_1, \ldots, \underline{y}_M\right\} \subset \Omega$ with large M. From (21), we need only form the interpolation matrix $\boldsymbol{\Psi}$ with entries $(\Psi)_{n,m} = \phi^{\underline{y}_n}(\underline{y}_m)$ for $m, n = 1, \ldots, M$. Then the first $\mathcal{N}$ points produced by the iteration (19) can be computed by standard numerical linear algebra operations on $\boldsymbol{\Psi}$. Either the first $\mathcal{N}$ iterations of a full-pivoting LU decomposition on $\boldsymbol{\Psi}$, or the first $\mathcal{N}$ iterations of a pivoted Choleksy decomposition produce the first $\mathcal{N}$ points of the Power Function method. In practice, we use the Cholesky decomposition method: because we only need the pivoting indices, the required work can be completed in $\mathcal{O}(\mathcal{N}^2 + M\mathcal{N})$ time with $\mathcal{O}(\mathcal{N}^2 + M)$ storage, and we need only perform this operation once as part of setting up the truth approximation. An example of the result of this algorithm for a two-dimensional domain is given in Fig. 1.

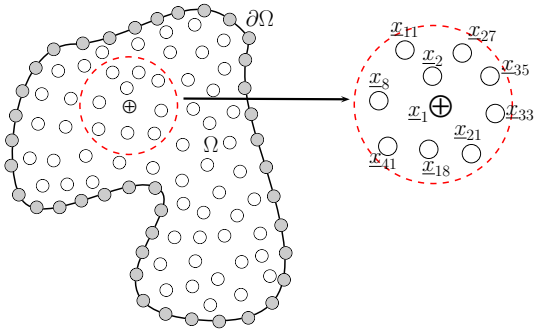


Fig. 2 *Left* Collocation points on a 2D irregular domain. *White nodes* are points inside the domain and *grey nodes* are boundary points. *Right* An example of a 10-point-stencil for computing differentiation weights at $\underline{x}_1$

3.2 RBF-FD Method

3.2.1 Local RBF Finite Difference Differentiation Matrices

The differentiation matrices obtained by using a global RBF interpolant based on infinitely smooth basis functions, such as IMQ, produce dense matrices. This is due to the fact that all $\mathcal{N}$ nodes in the domain are used to generate the interpolant. The cost for inverting the dense matrix $\mathbf{\Phi}$, though done only once, is manageable for modest size $\mathcal{N}$ but can be prohibitively expensive as it grows. One way to avoid dense matrix operations is to borrow ideas from finite-difference approximations, whose differentiation matrices are sparse. However, weights (i.e. entries of differentiation matrices) are difficult to compute when local stencils are scattered and differ in number and distributions. In order to mitigate these issues, local RBF interpolants and derivatives are used to compute stencil weights instead of using Taylor series, as in finite-difference methods. This is a straightforward approach for computing flexible finite-difference-like weights. As mentioned in the previous sections, this approach is known as a generalized finite-difference method or as RBF-FD.

We will illustrate the process of generating differentiation matrices in 2D; the generalization to higher dimensions is straightforward. The $\mathcal{N}$ nodal points in Ω chosen by the Power Function method in Sect. 3.1.3 are denoted as $C^{\mathcal{N}} = \{\underline{x}_1, \ldots, \underline{x}_{\mathcal{N}}\}$. We remark that the Power Function method is usually employed for global approximations and not necessarily for local ones such as the RBF-FD. However, for the large-stencil-size situation in this paper, the RBF approximation is closely related to the global problem. As such, we require a grid that is well-behaved for near-global approximations. The Power Function method provides an extremely simple and effective approach for such a purpose. Numerical tests (not reported here) did demonstrate better numerical stability than other RBF grids, e.g. uniformly distributed ones. Let $C_{(j)} = \{\underline{x}_{j_k} : k = 1, \ldots, n^j_{\text{loc}}\} \subset C^{\mathcal{N}}$ be the (local) set of neighboring points of $\underline{x}_j$ with $\underline{x}_{j_1} = \underline{x}_j$. $C_{(j)}$ will form a stencil for $\underline{x}_j$. We call the point $\underline{x}_j$ the *master* node of the set $C_{(j)}$. All other $n^j_{\text{loc}} - 1$ points in the set $C_{(j)}$ are *slave* nodes. As a simple example, Fig. 2 illustrates $\mathcal{N} = 101$ collocation points on a irregular domain Ω with a local stencil $C_{(1)}$ of 9 slave nodes. While one can vary the size of the local stencil n^j_{loc} with respect to the master location $\underline{x}_j$, we often set n^j_{loc} to be the same in order to guarantee that all local interpolants provide approximately the same accuracy. An approach with different local stencil sizes is useful when there are local accuracy considerations (e.g., boundary layers).

Following the global formulation provided in Sect. 3.1.1, the local interpolant $s_j(\underline{x})$ with master node $\underline{x}_j$ takes the form

$$s_{(j)}(\underline{x}) = \sum_{k=1}^{n_{\text{loc}}^j} \lambda_{j_k} \phi(\varepsilon \|\underline{x} - \underline{x}_{j_k}\|), \tag{22}$$

where $\underline{x}_{j_k} \in C_{(j)}$. As in (12), it follows that the differentiation matrix or derivative weights with respect to x (the first component of $\underline{x}$) evaluated at the master node $\underline{x}_j$ can be easily obtained as

$$\boldsymbol{D}_{x(j)} = \left[\phi_{x_j}(\varepsilon\|\underline{x}_j - \underline{x}_{j_1}\|), \ldots, \phi_{x_j}(\varepsilon\|\underline{x}_j - \underline{x}_{j_{n_{\text{loc}}^j}}\|)\right] \boldsymbol{\Phi}_{(j)}^{-1} = \boldsymbol{\Phi}_{x(j)} \boldsymbol{\Phi}_{(j)}^{-1}, \tag{23}$$

where $D_{x(j)}$ is of size $1 \times n_{\text{loc}}^j$ and $\boldsymbol{\Phi}_{(j)}$ is the local interpolation matrix defined by the local problem (22). Generating the $\mathcal{N} \times \mathcal{N}$ matrix $\boldsymbol{D}_1$ is done by computing $D_{x(j)}$ for $j = 1, \ldots, \mathcal{N}$ and placing these vectors in the corresponding rows and columns of $\boldsymbol{D}_1$. The pseudocode for the process is shown in Algorithm 2. Higher order differentiation matrices or derivatives

Algorithm 2 RBF-FD first derivative matrix with respect to x

Input: $C^{\mathcal{N}} = \{\underline{x}_1, \ldots, \underline{x}_{\mathcal{N}}\}$, $\phi(r)$, ε, n_{loc}.
Output: $\boldsymbol{D}_x$
for $j = 1$ to $\mathcal{N}$ **do**
  Find $C_{(j)} = \{\underline{x}_{j_k} : k = 1, \ldots, n_{\text{loc}}\} \subset C^{\mathcal{N}}$, i.e n_{loc} nearest-neighbors of $\underline{x}_j$.
  Compute $D_{x(j)} = \left[\phi_{x_j}(\varepsilon\|\underline{x}_j - \underline{x}_{j_1}\|), \ldots, \phi_{x_j}(\varepsilon\|\underline{x}_j - \underline{x}_{j_{n_{\text{loc}}}}\|)\right] \boldsymbol{\Phi}_{(j)}^{-1} = \boldsymbol{\Phi}_{x(j)} \boldsymbol{\Phi}_{(j)}^{-1}$
  Store elements of $D_{x(j)}$ as entries of $\boldsymbol{D}_x(j, j_1), \ldots, \boldsymbol{D}_x(j, j_{n_{\text{loc}}})$ accordingly.
end for

with respect to different variables can be computed in the same manner using the appropriate partial derivatives of $\phi(r)$ inside the for-loop of Algorithm 2. This algorithm generates $\mathcal{N} \times \mathcal{N}$ differentiation matrices with only $n_{\text{loc}}\mathcal{N}$ non-zero entries. The computational cost is $O(n_{\text{loc}}^3 \mathcal{N})$ dominated by $\mathcal{N}$ inversion processes of $\boldsymbol{\Phi}_{(j)}$. See [5,6,23,36,59] for stencil weights for RBF-FD for Gaussian and Multiquadric with constant shape parameters.

Once we have computed the RBF-FD differentiation matrices, we can use them to numerically solve boundary value problems or initial boundary value problems. As a motivating example, we consider the following 2D boundary value problem

$$-u_{xx} - \mu^1 u_{yy} - \mu^2 u = f(x, y), \qquad (x, y) \in \Omega, \tag{24a}$$

with boundary conditions

$$u(x, y) = g(x, y), \qquad (x, y) \in \partial\Omega. \tag{24b}$$

The parameters μ^1 and μ^2 are constants.

We then discretize Ω with $\mathcal{N}_i$ nodes and $\partial\Omega$ with $\mathcal{N}_b$ boundary nodes with a total number of nodes $\mathcal{N} = \mathcal{N}_i + \mathcal{N}_b$. We order the indices so that $\mathcal{N}_i$ interior points are followed by $\mathcal{N}_b$ boundary nodes. The discretized version of Eq. (24a) becomes

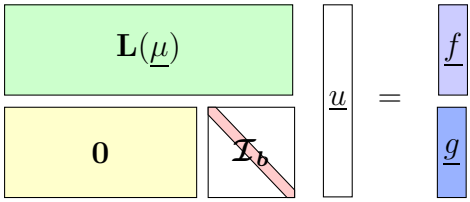
$$-\boldsymbol{D}_{xx}\underline{u} - \mu^1 \boldsymbol{D}_{yy}\underline{u} - \mu^2 \boldsymbol{\mathcal{I}_i}\underline{u} = \underline{f}, \tag{25}$$

or equivalently $\mathbf{L}(\underline{\mu})\underline{u} = \underline{f}$, with $\underline{\mu} = (\mu^1, \mu^2)$. The matrices $\boldsymbol{D}_{xx}$, $\boldsymbol{D}_{yy}$, which can be obtained from Algorithm 2, and consequently $\mathbf{L}(\underline{\mu})$ are of size $\mathcal{N}_i \times \mathcal{N}$. $\boldsymbol{\mathcal{I}_i}$ is an $\mathcal{N}_i \times \mathcal{N}$ matrix with values 1 at entries (j, j) for $j = 1 \ldots \mathcal{N}_i$ and zeros everywhere else.

For Dirichlet boundary conditions, the discretized version of the Eq. (24b) becomes

$$\mathcal{I}_b \underline{u} = \underline{g}, \tag{26}$$

where $\mathcal{I}_b$ is an $\mathcal{N}_b \times \mathcal{N}_b$ matrix with values 1 at entries (j, j) for $j = \mathcal{N}_i + 1, \ldots, \mathcal{N}$ and zeros everywhere else. The systems (25) and (26) are then augmented to form an $\mathcal{N} \times \mathcal{N}$ linear system as shown in (27).



(27)

3.3 Numerical Validation of the RBF-FD as Truth Solver

Putting all the above pieces together, we have a radial basis function finite difference (RBF-FD) method. This section considers convergence studies to validate the accuracy of the method. To that end, we run two kinds of convergence tests on four Eqs. (28a)–(29b), emulating h-adaptive and p-adaptive refinement convergence studies from classical finite element approaches:

(1) "n_{loc} convergence" – Refinement in the RBF-FD stencil size n_{loc} while holding $\mathcal{N}$ fixed is akin to p-refinement and so we expect exponential convergence in this case.
(2) "$\mathcal{N}$ convergence" – Refinement in the truth parameter $\mathcal{N}$ for a fixed RBF-FD stencil size n_{loc} is akin to h-refinement and so we expect algebraic convergence as a result.

$$\begin{cases} -u_{xx} - \mu^1 u_{yy} - \mu^2 u = f(\underline{x}), & \underline{x} \in \Omega \\ u = g, & \underline{x} \in \partial\Omega \end{cases} \qquad \mu \in \mathcal{D} = [0.1, 4] \times [0, 2] \tag{28a}$$

$$\begin{cases} (1 + \mu^1 x) u_{xx} + (1 + \mu^2 y) u_{yy} = f(\underline{x}), & \underline{x} \in \Omega \\ u = g, & \underline{x} \in \partial\Omega \end{cases} \qquad \mu \in \mathcal{D} = [-0.99, 0.99]^2, \tag{28b}$$

$$\begin{cases} -u_{xx} - \mu^1 u_{yy} - u_{zz} - \mu^2 u = f(\underline{x}), & \underline{x} \in \Omega \\ u = g, & \underline{x} \in \partial\Omega \end{cases} \qquad \mu \in \mathcal{D} = [0.1, 4] \times [0, 2] \tag{29a}$$

$$\begin{cases} (1 + \mu^1 x) u_{xx} + (1 + \mu^2 y) u_{yy} + z u_{zz} = f(\underline{x}), & \underline{x} \in \Omega \\ u = g, & \underline{x} \in \partial\Omega \end{cases} \qquad \mu \in \mathcal{D} = [-0.99, 0.99]^2. \tag{29b}$$

These four equations are elliptic partial differential equations with a two-dimensional parameter $\mu = (\mu^1, \mu^2)$. Among them, Eq. (28) are spatially two-dimensional problems, and Eq. (29) are spatially three-dimensional problems. The problems (28a) and (29a) are examples of the Helmholtz equation, describing frequency-domain solutions to Maxwell's equations of electromagnetics. Examples (28b) and (29b) are steady-state diffusion equations with anisotropic diffusivity coefficients. Finally, we choose the inverse multiquadric (IMQ) shape

Table 1 Parameter-independent exact solutions for the validation of the solver

	2D	3D
Test 1	$u(x, y) = \sin(\pi x)\sin(\pi y)$	$u(x, y, z) = \sin(\pi x)\sin(\pi y)\sin(\pi z)$
Test 2	$u(x, y) = e^{-20(x^2+y^2)} - x^2 + y^3$	$u(x, y, z) = e^{-20(x^2+y^2+z^2)} - x^2 + y^3 - z^2$

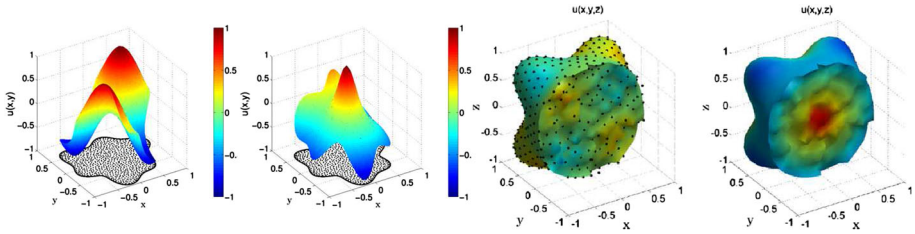


Fig. 3 Plot of the exact solutions: shown from left to right are 2D Test 1, 2D Test 2, 3D Test 1, 3D Test 2

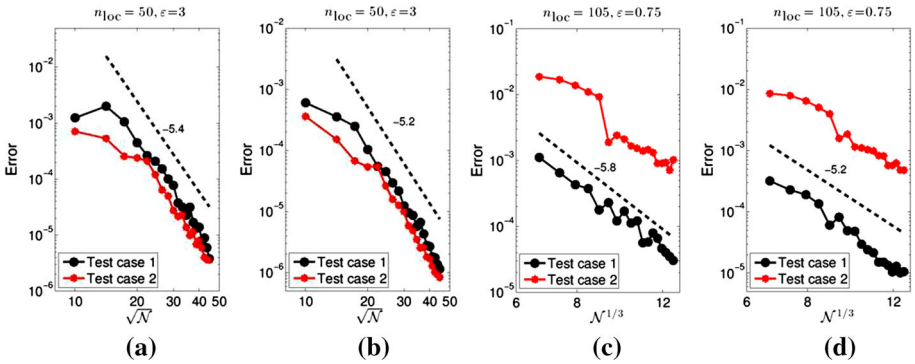


Fig. 4 Convergence of the worst case error $\max_{\mu\in\Xi} \left\| u(\mu) - u^{\mathcal{N}}(\mu) \right\|_{\ell^2}$ for Eqs. (28a)–(29b) (*left* to *right*) as $\mathcal{N}$ increases and n_{loc} is fixed at 50 for 2D and 105 for 3D

function with shape parameter $\varepsilon = 3$ for 2D and 0.75 for 3D. To showcase the functionality of the method, we choose the computational domain Ω to be irregular: for the 2D problems, the domain Ω is centered at the origin with boundary $\partial\Omega$ given by the parametric equation in polar coordinates $r(\theta) = 0.8 + 0.1(\sin(6\theta) + \sin(3\theta)),\ \ 0 \le \theta \le 2\pi$ (see Fig. 1); for 3D ones, Ω is the closed interior of a solid object defined by the following parametric surface:

$$x^2 + y^2 + z^2 - \sin(2x)^2 \sin(2y)^2 \sin(2z)^2 = 1. \tag{30}$$

These same four equations will be used again to test the reduced solver later where we will have a fixed forcing function $f(x)$ and boundary data $g(x)$ (and thus parametric solution). For the purpose of validating the RBF-FD solver in this section, we choose $f(x)$ and $g(x)$ such that the exact solution u is known. They are listed in Table 1 and depicted in Fig. 3.

Accuracy of the truth solver. The $\mathcal{N}-$history of convergence results are shown in Fig. 4, and that for n_{loc} is shown in Fig. 5. The error shown in these figures is the μ-maximum spatial ℓ^2 norm over a candidate set Ξ of 10, 000 equi-spaced parameters in $\mathcal{D}$.

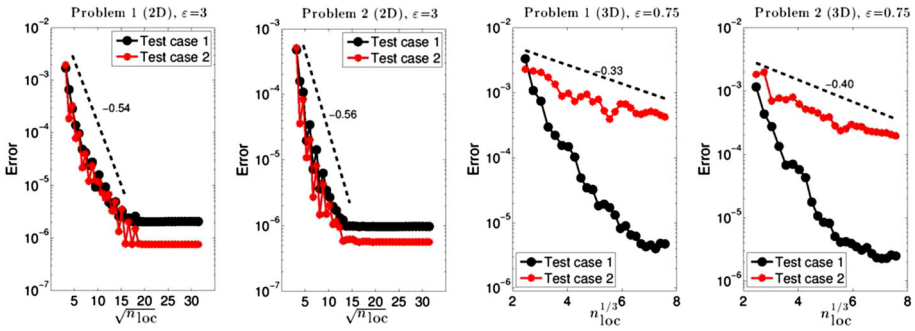


Fig. 5 Convergence of the worst case error $\max_{\mu\in\Xi} \left\| u(\mu) - u^{\mathcal{N}}(\mu) \right\|_{\ell^2}$ for Eqs. (28a)–(29b) (*left* to *right*) as n_{loc} increases and $\mathcal{N}$ is fixed at 1000 for 2D and 2046 for 3D

We observe that these numerical results indicate that the RBF-FD solver is robust and provides solutions that converge to the exact solution algebraically with respect to $\mathcal{N}$, and exponentially with respect to n_{loc}. We note, however, that the error for the n_{loc} convergence does level off with large number of local stencil points. This issue is a result of the ill conditioning observed in traditional RBF methods as the centers become closer. It is partially mitigated by the use of greedily-selected optimal nodes obtained by using the discrete power function method described in Sect. 3.1.3. Without it, the errors may eventually grow instead of flattening out due to ill-conditioning.

Once the *a priori* expectation of $hp-$type of convergence is confirmed, we have a reliable truth solver in RBF-FD. Moreover, these studies provide a reference for the accuracy of the truth approximations underlying the reduced solver in the next section. This accuracy will then provide a rough guideline for selecting the total number of reduced bases.

4 The Reduced Radial Basis Function Method

The novel contribution of this paper is the Reduced Radial Basis Function Method (R^2BFM) that we describe in this section. The R^2BFM algorithm uses the local RBF-FD method described in Sect. 3.2 as the truth solver for a parameterized PDE of the general form (1). We mainly consider irregular geometries, where this truth solver is advantageous compared to other solution methods. The tests run in Sect. 3.3 show that this truth solver is highly accurate, featuring h-adaptivity in the parameter $\mathcal{N}$, and p-adaptivity in the parameter n_{loc}. The R^2BFM algorithm then uses the LSRCM introduced in [12] and described in Sect. 2 to define the offline-online decomposition, defining the low-rank approximation space and the reduced-order operators.

4.1 The R^2BFM Algorithm

In this section we outline the greedy algorithm for the (least squares) Reduced Radial Basis Function Method. Our truth approximation is given by the local RBF method outlined in Sect. 3. The LSRCM method from Sect. 2 is naturally applicable to this solver because the local RBF method is a collocation method. However, we are left to specify the error estimate Δ_n given by (5). To do this, we adapt the Chebyshev pseudospectral arguments from [12] to our local RBF case. In our RBF setting, we have a natural specification for the norm: the native space norm $\|\cdot\|_{\mathcal{H}}$ defined through $\boldsymbol{\Phi} = \boldsymbol{S}\boldsymbol{S}^T$. To state our result, we define

$$\alpha_{UB}^{S} = \max_{\mathbf{w}\in\mathbb{R}^{\mathcal{N}}} \frac{\mathbf{w}^T \mathbf{S}^{-T} \mathbf{S}^{-1} \mathbf{w}}{\mathbf{w}^T \mathbf{w}},$$

which relates the native space norm of a function to the standard Euclidian norm $\|\cdot\|$ of the corresponding vector by

$$\|\boldsymbol{v}\|_{\mathcal{H}}^2 \leq \alpha_{UB}^{S} \|\boldsymbol{v}\|^2, \tag{31}$$

where $\boldsymbol{v}$ is a vector of collocation evaluations of the truth approximation PDE solution. In this context, we can define two types of error estimators, one that works on residuals measured in the native space norm, and the second that is measured in the Euclidean norm:

$$\Delta_n^1(\mu) \triangleq \frac{\sqrt{\alpha_{UB}^{S}} \left\| \boldsymbol{S}^{-1} \left(\mathbf{f}^{\mathcal{N}} - \mathbb{L}_{\mathcal{N}}(\mu) u_\mu^{(n)} \right) \right\|}{\sqrt{\beta_{LB}^{S}(\mu)}}, \tag{32a}$$

$$\Delta_n^2(\mu) \triangleq \frac{\sqrt{\alpha_{UB}^{S}} \left\| \left(\mathbf{f}^{\mathcal{N}} - \mathbb{L}_{\mathcal{N}}(\mu) u_\mu^{(n)} \right) \right\|}{\sqrt{\beta_{LB}(\mu)}}. \tag{32b}$$

The β factors that translate these residuals into (native-space-norm or Euclidean-norm) errors are given by

$$\beta_{LB}^{S}(\mu) \triangleq \min_{\mathbf{w}\in\mathbb{R}^{\mathcal{N}}} \frac{\mathbf{w}^T \mathbb{L}_{\mathcal{N}}^T(\mu) \mathbf{S}^{-T} \mathbf{S}^{-1} \mathbb{L}_{\mathcal{N}}(\mu) \mathbf{w}}{\mathbf{w}^T \mathbf{w}}, \tag{33a}$$

$$\beta_{LB}(\mu) \triangleq \min_{\mathbf{w}\in\mathbb{R}^{\mathcal{N}}} \frac{\mathbf{w}^T \mathbb{L}_{\mathcal{N}}^T(\mu) \mathbb{L}_{\mathcal{N}}(\mu) \mathbf{w}}{\mathbf{w}^T \mathbf{w}}. \tag{33b}$$

The estimators Δ_n^i for $i = 1, 2$ are computable, and they form rigorous error estimators in the native space.

Theorem 4.1 *For any μ, let $u_\mu^{\mathcal{N}}(\mu)$ be the truth approximation solving (2) and $u_\mu^{(n)}(\mu)$ be the reduced basis solution (3) solving (4). Then we have $\|u_\mu^{\mathcal{N}} - u_\mu^{(n)}\|_{\mathcal{H}} \leq \Delta_n^i(\mu)$ for $i = 1, 2$.*

Proof We have the following error equations on the $\mathcal{N}$-dependent fine domain RBF grid thanks to the equation satisfied by the truth approximation (2):

$$\mathbb{L}_{\mathcal{N}}(\mu) \left(u_\mu^{\mathcal{N}} - u_\mu^{(n)} \right) = f - \mathbb{L}_{\mathcal{N}}(\mu) u_\mu^{(n)}, \quad \boldsymbol{S}^{-1} \mathbb{L}_{\mathcal{N}}(\mu) \left(u_\mu^{\mathcal{N}} - u_\mu^{(n)} \right) = \boldsymbol{S}^{-1} (f - \mathbb{L}_{\mathcal{N}}(\mu) u_\mu^{(n)}).$$

Taking the the standard Euclidian norm and using basic properties of eigenvalues gives

$$\left\| u_\mu^{\mathcal{N}} - u_\mu^{(n)} \right\| \leq \frac{\left\| f - \mathbb{L}_{\mathcal{N}}(\mu) u_\mu^{(n)} \right\|}{\sqrt{\beta_{LB}(\mu)}}, \quad \left\| u_\mu^{\mathcal{N}} - u_\mu^{(n)} \right\| \leq \frac{\left\| \boldsymbol{S}^{-1} (f - \mathbb{L}_{\mathcal{N}}(\mu) u_\mu^{(n)}) \right\|}{\sqrt{\beta_{LB}^{S}(\mu)}}$$

respectively. We then apply the inequality (31) to finish the proof. □

These error bounds can then be used to perform the offline LSRCM computations in Sect. 2.1: determination of the parameter values $\mu^1, \ldots, \mu^N$ and the subsequent snapshots $u_{\mu^1}, \ldots, u_{\mu^N}$. We have used the second estimate Δ_n^2 in our experiments below. The two estimators perform very similarly with the reduced solution defined by (4). However, we expect them to be different if the reduced solution is sought differently, e.g. under the analytically preconditioned setting as in [13]. In that setting, $\boldsymbol{S}$ would have to be replaced by a suitably-defined parameter-dependent pre-conditioner; however, the investigation of the

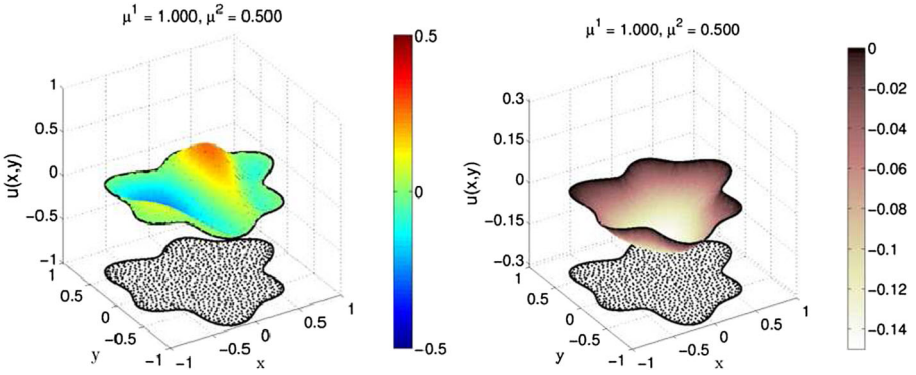


Fig. 6 Surface plots of the sample solutions for the 2D test problems (28a) (*left*) and (28b) (*right*) at $\mu^1 = 1$ and $\mu^2 = 0.5$

effect of different estimators on a reduced solution is outside the scope of this paper. The remaining LSRCM steps in Sects. 2.2 and 2.3 are as described in those sections. We outline the entire offline algorithm in Algorithm 3.

Remark The estimates above work in the "global" native space defined by the shape function ϕ and width parameter ε. However, our RBF approximation is a *local* approximation and so it is perhaps more appropriate to use a "local" native space norm in order to compute these estimates. However, our tests have shown that this distinction does not affect the result much in practice, although it does make a difference for very small local stencil sizes (e.g., $n_{\text{loc}}^{1/d} \leq 3$). In order to keep the method simple, we have therefore used the global native space norm as presented above for the RCM error estimate.

Algorithm 3 Least Squares R^{2}BFM: Offline Procedure

1. Discretize the parameter domain $\mathcal{D}$ by Ξ, and denote the center of $\mathcal{D}$ by μ_c.

2. Randomly select μ_1 and solve the RBF problem $\mathbb{L}_{\mathcal{N}}(\mu_1)\, u_{\mu_1}^{\mathcal{N}}(\underline{x}) = f(\underline{x};\mu_1)$ for $\underline{x} \in C^{\mathcal{N}}$.

3. For $n = 2, \ldots, N$ do

1). Form $\mathbb{A}_{n-1} = \left(\mathbb{L}_{\mathcal{N}}\, u_{\mu_1}^{\mathcal{N}}, \mathbb{L}_{\mathcal{N}}\, u_{\mu_2}^{\mathcal{N}}, \ldots, \mathbb{L}_{\mathcal{N}}\, u_{\mu_{n-1}}^{\mathcal{N}}\right)$.

2). For all $\mu \in \Xi$, solve $\mathbb{A}_{n-1}^T\, \mathbb{A}_{n-1}\, \mathbf{c} = \mathbb{A}_{n-1}^T\, \mathbf{f}^{\mathcal{N}}$ to obtain $u_\mu^{(n-1)} = \sum_{j=1}^{n-1} c_j u_{\mu_j}^{\mathcal{N}}$.

3). For all $\mu \in \Xi$, calculate $\Delta_{n-1}^2(\mu)$. Then set $\mu^n = argmax_\mu\ \Delta_{n-1}(\mu)$.

4). Solve the RBF problem $\mathbb{L}_{\mathcal{N}}(\mu_n)\, u_{\mu_n}^{\mathcal{N}}(\underline{x}) = f(\underline{x};\mu_n)$ for $\underline{x} \in C^{\mathcal{N}}$.

5). Apply a modified Gram-Schmidt transformation, with inner product defined by $(u, v) \equiv \left(\mathbb{L}_{\mathcal{N}}(\mu_c)u, \mathbb{L}_{\mathcal{N}}(\mu_c)v\right)_{L^2(\Omega)}$, on the basis $\left\{u_{\mu_1}^{\mathcal{N}}, u_{\mu_2}^{\mathcal{N}}, \ldots, u_{\mu_n}^{\mathcal{N}}\right\}$ to obtain a more stable basis $\left\{\xi_1^{\mathcal{N}}, \xi_2^{\mathcal{N}}, \ldots, \xi_n^{\mathcal{N}}\right\}$ to use as terms in the expansion (3) for the least squares reduced collocation method.

4.2 Numerical Results

We test our reduced solver on the four Eqs. (28a)–(29b) listed in Sect. 3.3. In all cases, we employ the error estimator Δ_n^2 from (32b) to select snapshots and construct the reduced approximation space.

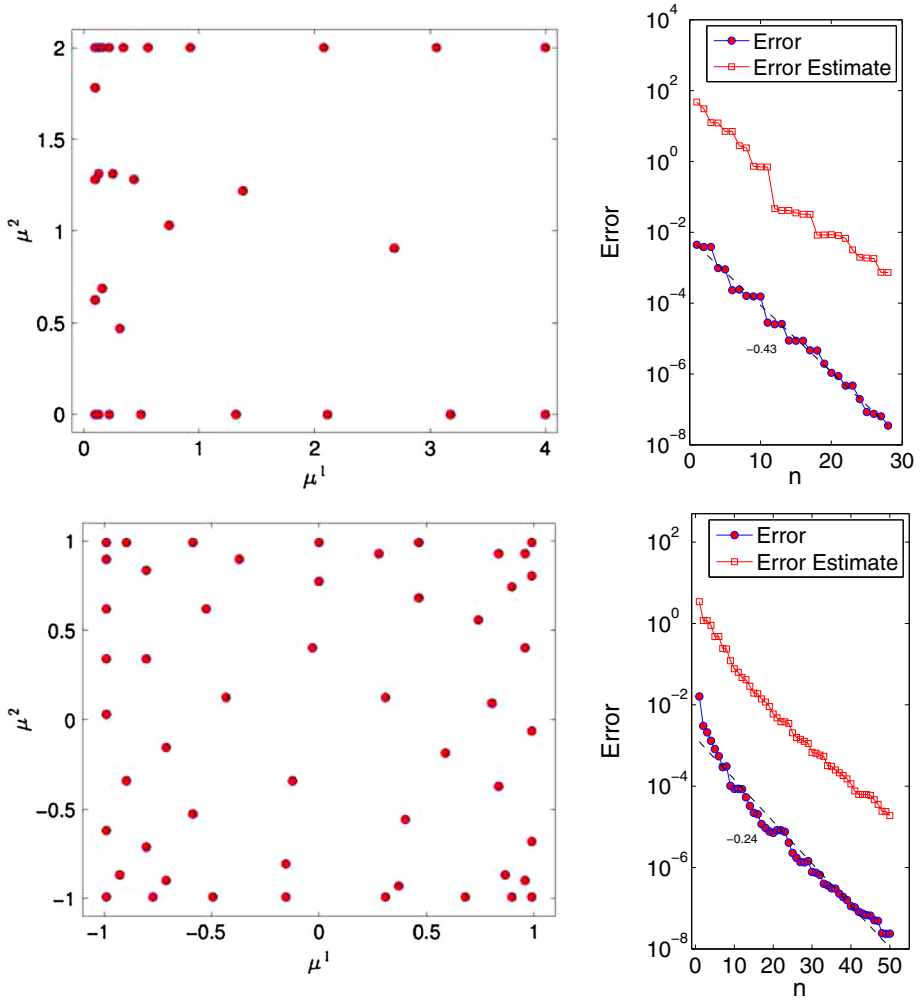


Fig. 7 Selected parameter values to build the reduced solution spaces (*left*), and the history of convergence of the RB approximations (*right*). Shown on top is for 2D problem (28a), and at the bottom is for (28b)

4.2.1 R²BFM: Two Dimensional Cases

In this section, we test our reduced solver R²BFM, taking $f = -10\sin(8x(y-1))$ for (28a) and $f = e^{4xy}$ for (28b), both with homogeneous boundary conditions. We discretize the 2D domain Ω with $\mathcal{N} = 1000$ RBF nodes with stencil of size $n_{\text{loc}} = 50$. These $\mathcal{N}$ RBF nodes (see Fig. 1) are selected by a greedy Cholesky algorithm out of 2984 candidate points that are uniformly distributed on Ω. In the following numerical experiments we use inverse multiquadric RBFs with shape parameter $\varepsilon = 3$.

In Fig. 6 we show the surface plots of the solutions to (28a) and (28b) at a particular parameter value $\mu^1 = 1$ and $\mu^2 = 0.5$. These images show the irregular domain shape and the complexity of the solution profile resulting from it. In Fig. 7 we see that the R²BFM solution is converging exponentially to the RBF solution. Compared to the full RBF truth

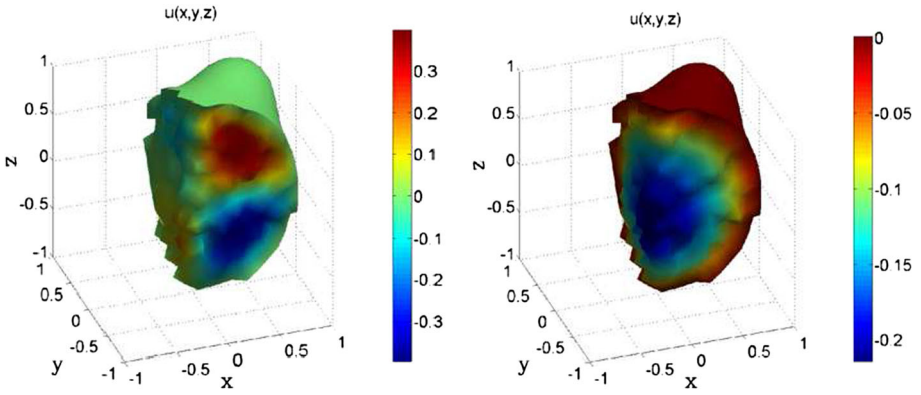


Fig. 8 Sample solutions at $\mu^1 = 1$ and $\mu^2 = 0.5$ for the 3D problems (29a) (*left*) and (29b) (*right*)

simulation with $\mathcal{N} = 1000$ centers and local stencils of size 50, the R^{2}BFM algorithm achieves comparable accuracy with much less than $n = 50$ basis elements. We leave the details of the comparison for Sect. 4.2.3.

4.2.2 R^{2}BFM: Three Dimensional Cases

In this section we test our reduced solver R^{2}BFM on three-dimensional problems by taking $f = -10\sin(8x(y-1)z)$ for (29a) and $f = e^{4xyz}$ for (29b) with homogeneous Dirichlet boundary conditions. Truth approximation simulations were carried out using the inverse multiquadric RBF with $\varepsilon = 0.75$, and local stencils of size $n_{\text{loc}} = 125$ on a mesh comprised of a total of 2046 points from the Power Function method. Figure 8 shows the solutions corresponding to $\mu^1 = 1$ and $\mu^2 = 0.5$. Again, the reduced basis method converges exponentially (Fig. 9) and provides accurate surrogate solutions with a very small number of basis functions. See Sect. 4.2.3 for the details of the comparison.

4.2.3 R^{2}BFM Efficiency: Computational Time

In this section, we report, in Table 2, the computational time for the different stages of the method and the speedup of the R^{2}BFM. All computations are carried out using MATLAB2013.B on a Mac Pro workstation with 12 GB ECC memory and an eight core 2.8 GHz Intel Xeon processor. In Table 2, n is the number of snapshots we use for the comparison between the reduced solver and the full solver (with a particular $\mathcal{N}$ and n_{loc}). They are determined by having the accuracy of the two solvers roughly comparable (around 10^{-4}). The former is located by referring to Figs. 7 and 9, and the latter by Figs. 4 and 5 with the specific values of $\mathcal{N}$ and n_{loc}.

τ_{offline} represents the offline computational time (in seconds) to compute n basis snapshots along with associated reduced-order operators. We point out that τ_{offline} does not include the time spent on the calculation of the stability constant (33). In this paper, we calculate directly these constants which takes time that is comparable (2D) or more than (3D) τ_{offline}. However, we have tested locating a lower bound of the stability constant by the natural-norm successive constraint method [31]. This algorithm cuts the time for the stability constant calculation by at least 75% in all cases.

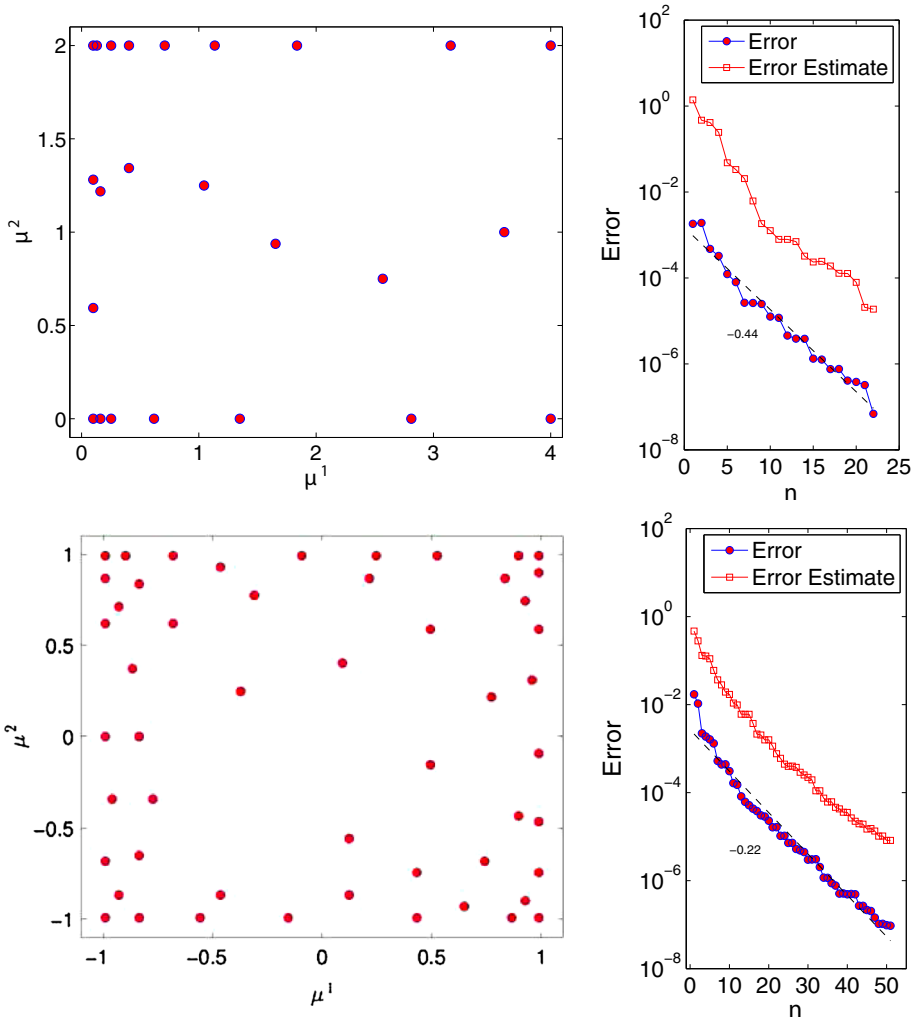


Fig. 9 Selected parameter values for the RB space generation (*left*), and the convergence of the R^{2}BFM solutions (*right*). Plotted on top is for the 3D problem (29a) and at the bottom is for (29b)

$\bar{\tau}_{\text{ts}}$ is the average computational time to solve the PDE with the truth RBF-FD solver. $\bar{\tau}_{\text{rb}}$ is that by using the R^{2}BFM solver with n basis functions. We observe that, in all cases, the R^{2}BFM provides speedup factors of more than 55. We note that the moderate speedup (compared to the usually-reported RBM speedups of $\mathcal{O}(100)$) is due to the specific implementation of our MATLAB code. In particular, the online assembling time is in the order of $Q_a n^2$ which should be negligible in comparison to the time devoted to solving for the reduced solution which is $O(n^3)$.

Unfortunately, this is not the case in our implementation due to the use of the cell data structures and multi-dimentional arrays in MATLAB. In fact, if we exclude the assembling time in our calculation, we *recover* the usual speedup of 2 to 3 orders of magnitude for these type of problems. This speedup calculation (where online computational time does not include operator assembly time) is shown in Fig. 10 for a large number of different n and $\mathcal{N}$.

Table 2 Computational time and the corresponding speedup:

Problem	n	τ_{offline} (s)	$\bar{\tau}_{\text{ts}}$ (s)	$\bar{\tau}_{\text{rb}}$ (s)	Speed up $\left(\frac{\bar{\tau}_{\text{ts}}}{\bar{\tau}_{\text{rb}}}\right)$
1 (2D)	12	190.28	0.0603576	0.00108305	55.7290
2 (2D)	12	111.65	0.0690138	0.00125392	55.0836
1 (3D)	6	145.89	0.1513660	0.00168018	90.0895
2 (3D)	12	221.38	0.1463470	0.00183657	79.6850

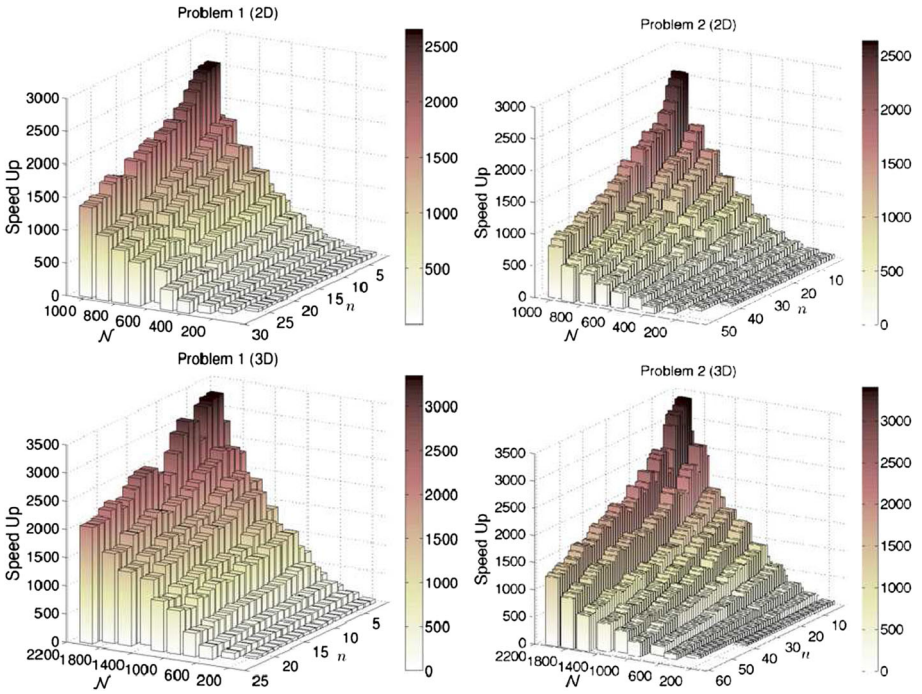


Fig. 10 Computational speedup excluding the online assembling time: shown on top are for the 2D problem (28a) (*left*) and (28b) (*right*), on the bottom are (29a) (*left*) and (29b) (*right*)

5 Conclusion

Partial differential equations that have parametric dependence are challenging problems in scientific computing. The reduced-basis method efficiently handles these problems, even in the collocation setting for pseudospectral approximations. However, when the problem has difficulty compounded by an irregular geometry, standard pseudospectral collocation methods (e.g. Chebyshev, Fourier) cannot be directly applied.

We have addressed this problem by applying a local radial basis function approximation method to this situation. Due to their ability to approximate on irregular geometries, RBF methods provide excellent candidates for a collocation approximation. In particular we have employed a finite-difference version of RBF methods that uses local stencils to form differential operator approximations. The result is an *hp*-adaptive-like method on irregular geometries with local, hence efficient, operators.

We use the RBF-FD method as the truth approximation in a reduced basis collocation method, resulting in the Reduced Radial Basis Function Method. We have shown via extensive tests that this R^2BFM algorithm can efficiently solve parametric problems on irregular geometries, effectively combining the strengths of both RBM and RBF algorithms.

References

1. Almroth, B.O., Stern, P., Brogan, F.A.: Automatic choice of global shape functions in structural analysis. AIAA J. **16**, 525–528 (1978)
2. Balmes, E.: Parametric families of reduced finite element models: theory and applications. Mach. Syst. Signal Process. **10**(4), 381–394 (1996)
3. Barrault, M., Nguyen, N.C., Maday, Y., Patera, A.T.: An "empirical interpolation" method: application to efficient reduced-basis discretization of partial differential equations. C. R. Acad. Sci. Paris Sér I **339**, 667–672 (2004)
4. Barrett, A., Reddien, G.: On the reduced basis method. Z. Angew. Math. Mech. **75**(7), 543–549 (1995)
5. Bayona, V., Moscoso, M., Kindelan, M.: Optimal constant shape parameter for multiquadric based RBF-FD method. J. Comput. Phys. **230**(19), 7384–7399 (2011)
6. Bayona, V., Moscoso, M., Kindelan, M.: Gaussian RBF-FD weights and its corresponding local truncation errors. Eng. Anal. Bound. Elem. **36**(9), 1361–1369 (2012)
7. Binev, P., Cohen, A., Dahmen, W., Devore, R., Petrova, G., Wojtaszczyk, P.: Convergence rates for greedy algorithms in reduced basis methods. SIAM J. Math. Anal. **43**, 1457–1472 (2011)
8. Buffa, A., Maday, Y., Patera, A.T., Prud'homme, C., Turinici, G.: A priori convergence of the greedy algorithm for the parametrized reduced basis. ESAIM Math. Model. Numer. Anal. **46**, 595–603 (2011). (Special Issue in honor of David Gottlieb)
9. Buhmann, M.D.: Radial Basis Functions: Theory and Implementations, Cambridge Monographs on Applied and Computational Mathematics, vol. 12. Cambridge University Press, Cambridge (2003)
10. Carlberg, K., Farhat, C., Cortial, J., Amsallem, D.: The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. J. Comput. Phys. **242**, 623–647 (2013)
11. Cavoretto, R., Fasshauer, G.E., McCourt, M.: An introduction to the Hilbert-Schmidt SVD using iterated Brownian bridge kernels. Numer. Algorithms **68**(2), 393–422 (2015)
12. Chen, Y., Gottlieb, S.: Reduced collocation methods: reduced basis methods in the collocation framework. J. Sci. Comput. **55**(3), 718–737 (2013)
13. Chen, Y., Gottlieb, S., Maday, Y.: Parametric analytical preconditioning and its applications to the reduced collocation methods. C. R. Math. **352**(7–8), 661–666 (2014)
14. Chen, Y., Hesthaven, J.S., Maday, Y., Rodríguez, J.: Improved successive constraint method based a posteriori error estimate for reduced basis approximation of 2d Maxwell's problem. Math. Modell. Numer. Anal. **43**, 1099–1116 (2009)
15. Chen, Y., Hesthaven, J.S., Maday, Y., Rodríguez, J.: Certified reduced basis methods and output bounds for the harmonic Maxwell's equations. SIAM J. Sci. Comput. **32**(2), 970–996 (2010)
16. Driscoll, T., Fornberg, B.: Interpolation in the limit of increasingly flat radial basis functions. Comput. Math. Appl. **43**, 413–422 (2002)
17. Drohmann, M., Haasdonk, B., Ohlberger, M.: Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation. SIAM J. Sci. Comput. **34**(2), A937–A969 (2012)
18. Fasshauer, G.E., McCourt, M.J.: Stable evaluation of Gaussian radial basis function interpolants. SIAM J. Sci. Comput. **34**(2), A737–A762 (2012)
19. Fasshauer, G.E.: Meshfree approximation methods with MATLAB, Interdisciplinary Mathematical Sciences, vol. 6. World Scientific Publishing Co., Pte. Ltd., Hackensack, NJ: With 1 CD-ROM. (Windows, Macintosh and UNIX) (2007)
20. Fink, J.P., Rheinboldt, W.C.: On the error behavior of the reduced basis technique for nonlinear finite element approximations. Z. Angew. Math. Mech. **63**(1), 21–28 (1983)
21. Fornberg, B., Larsson, E., Flyer, N.: Stable computations with Gaussian radial basis functions. SIAM J. Sci. Comput. **33**(2), 869–892 (2011)
22. Fornberg, B., Lehto, E.: Stabilization of RBF-generated finite difference methods for convective PDEs. J. Comput. Phys. **230**(6), 2270–2285 (2011)

23. Fornberg, B., Lehto, E., Powell, C.: Stable calculation of Gaussian-based RBF-FD stencils. Comput. Math. Appl. **65**(4), 627–637 (2013)
24. Fornberg, B., Piret, C.: A stable algorithm for flat radial basis functions on a sphere. SIAM J. Sci. Comput. **30**(1), 60–80 (2008)
25. Fornberg, B., Wright, G.: Stable computation of multiquadric interpolants for all values of the shape parameter. Comput. Math. Appl. **48**(5–6), 853–867 (2004)
26. Gonnet, P., Pachón, R., Trefethen, L.N.: Robust rational interpolation and least-squares. Electron. Trans. Numer. Anal. **38**, 146–167 (2011)
27. Grepl, M.A., Maday, Y., Nguyen, N.C., Patera, A.T.: Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. Math. Modell. Numer. Anal. **41**(3), 575–605 (2007)
28. Grepl, M.A., Patera, A.T.: A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. Math. Model. Numer. Anal. **39**(1), 157–181 (2005)
29. Haasdonk, B., Ohlberger, M.: Reduced basis method for finite volume approximations of parametrized linear evolution equations. Math. Model. Numer. Anal. **42**(2), 277–302 (2008)
30. Hesthaven, J.S., Gottlieb, S., Gottlieb, D.: Spectral Methods for Time-Dependent Problems, Cambridge Monographs on Applied and Computational Mathematics, vol. 21. Cambridge University Press, Cambridge (2007)
31. Huynh, D.B.P., Knezevic, D.J., Chen, Y., Hesthaven, J.S., Patera, A.T.: A natural-norm successive constraint method for inf-sup lower bounds. Comput. Methods Appl. Mech. Eng. **199**, 1963–1975 (2010)
32. Huynh, D.B.P., Rozza, G., Sen, S., Patera, A.T.: A successive constraint linear optimization method for lower bounds of parametric coercivity and inf-sup stability constants. C. R. Acad. Sci. Paris Sér I. **345**, 473–478 (2007)
33. Kansa, E.J.: Multiquadrics—a scattered data approximation scheme with applications to computational fluid dynamics I: surface approximations and partial derivative estimates. Comput. Math. Appl. **19**(8/9), 127–145 (1990)
34. Kansa, E.J.: Multiquadrics—a scattered data approximation scheme with applications to computational fluid dynamics II: solutions to parabolic, hyperbolic, and elliptic partial differential equations. Comput. Math. Appl. **19**(8/9), 147–161 (1990)
35. Larsson, E., Fornberg, B.: A numerical study of some radial basis function based solution methods for elliptic PDEs. Comput. Math. Appl. **46**(5–6), 891–902 (2003)
36. Larsson, E., Lehto, E., Heryudono, A., Fornberg, B.: Stable computation of differentiation matrices and scattered node stencils based on Gaussian radial basis functions. SIAM J. Sci. Comput. **35**(4), A2096–A2119 (2013)
37. Maday, Y., Patera, A.T., Turinici, G.: A priori convergence theory for reduced-basis approximations of single-parameter elliptic partial differential equations. J. Sci. Comput. **17**, 437–446 (2002)
38. Marchi, S.D., Schaback, R., Wendland, H.: Near-optimal data-independent point locations for radial basis function interpolation. Adv. Comput. Math. **23**(3), 317–330 (2005)
39. Matache, A.M., Babuška, I., Schwab, C.: Generalized p-FEM in homogenization. Numer. Math. **86**(2), 319–375 (2000)
40. Micchelli, C.: Interpolation of scattered data: distance matrices and conditionally positive definite functions. Constr. Approx. **2**, 11–22 (1986)
41. Nagy, D.A.: Modal representation of geometrically nonlinear behaviour by the finite element method. Comput. Struct. **10**, 683–688 (1979)
42. Nguyen, N.C., Rozza, G., Huynh, D.B.P., Patera, A.T.: Reduced basis approximation and a posteriori error estimation for parametrized parabolic pdes; application to real-time bayesian parameter estimation. In: Biegler, L., Biros, G., Ghattas, O., Heinkenschloss, M., Keyes, D., Mallick, B., Marzouk, Y., Tenorio, L., van Bloemen Waanders, B., Willcox, K. (eds.) Large-Scale Inverse Problems and Quantification of Uncertainty. Wiley, New York (2010)
43. Noor, A.K., Peters, J.M.: Reduced basis technique for nonlinear analysis of structures. AIAA J. **18**(4), 455–462 (1980)
44. Peterson, J.S.: The reduced basis method for incompressible viscous flow calculations. SIAM J. Sci. Stat. Comput. **10**(4), 777–786 (1989)
45. Platte, R.B., Trefethen, L.N., Kuijlaars, A.B.J.: Impossibility of fast stable approximation of analytic functions from equispaced samples. SIAM Rev. **53**(2), 308–318 (2011)
46. Platte, R.B.: How fast do radial basis function interpolants of analytic functions converge? IMA J. Numer. Anal. **31**(4), 1578–1597 (2011)
47. Platte, R.B., Driscoll, T.A.: Polynomials and potential theory for Gaussian radial basis function interpolation. SIAM J. Numer. Anal. **43**(2), 750–766 (2005). (electronic)
48. Porsching, T.A.: Estimation of the error in the reduced basis method solution of nonlinear equations. Math. Comput. **45**(172), 487–496 (1985)

49. Prud'homme, C., Rovas, D., Veroy, K., Maday, Y., Patera, A.T., Turinici, G.: Reliable real-time solution of parametrized partial differential equations: reduced-basis output bound methods. J. Fluids Eng. **124**(1), 70–80 (2002)
50. Rozza, G., Huynh, D.B.P., Patera, A.T.: Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations: application to transport and continuum mechanics. Arch. Comput. Methods Eng. **15**(3), 229–275 (2008)
51. Shen, J., Tang, T.: Spectral and High-Order Methods with Applications, Mathematics Monograph Series, vol. 3. Science Press Beijing, Beijing (2006)
52. Shu, C., Ding, H., Yeo, K.S.: Computation of incompressible Navier–Stokes equations by local RBF-based differential quadrature method. CMES Comput. Model. Eng. Sci. **7**(2), 195–205 (2005)
53. Tolstykh, A.I., Shirobokov, D.A.: On using radial basis functions in a "finite difference mode" with applications to elasticity problems. Comput. Mech. **33**(1), 68–79 (2003)
54. Trefethen, L.N.: Spectral Methods in MATLAB, Software, Environments, and Tools, vol. 10. Society for Industrial and Applied Mathematics (SIAM), Philadelphia (2000)
55. Urban, K., Patera, A.T.: An improved error bound for reduced basis approximation of linear parabolic problems. Math. Comput. **83**(288), 1599–1615 (2014)
56. Veroy, K., Prud'homme, C., Patera, A.T.: Reduced-basis approximation of the viscous Burgers equation: rigorous a posteriori error bounds. C. R. Acad. Sci. Paris Sér I **337**(9), 619–624 (2003)
57. Wendland, H.: Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. Adv. Comput. Math. **4**(4), 389–396 (1995)
58. Wendland, H.: Scattered Data Approximation, Cambridge Monographs on Applied and Computational Mathematics, vol. 17. Cambridge University Press, Cambridge (2005)
59. Wright, G.B., Fornberg, B.: Scattered node compact finite difference-type formulas generated from radial basis functions. J. Comput. Phys. **212**(1), 99–123 (2006)
60. Yano, M., Patera, A.T.: A space–time variational approach to hydrodynamic stability theory. Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. **469**(2155), 20130036 (2013)
61. Yano, M., Patera, Anthony T., Urban, K.: A space–time *hp*-interpolation-based certified reduced basis method for Burgers' equation. Math. Models Methods Appl. Sci. **24**(9), 1903–1935 (2014)