Journal of Applied Mathematics and Physics, 2023, 11, 3867-3886
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

Local Radial Basis Function Methods: Comparison, Improvements, and Implementation

Scott A. Sarra

Department of Mathematics, Marshall University, Huntington, USA
Email: sarra@marshall.edu

How to cite this paper: Sarra, S.A. (2023) Local Radial Basis Function Methods: Comparison, Improvements, and Implementation. *Journal of Applied Mathematics and Physics*, **11**, 3867-3886.
https://doi.org/10.4236/jamp.2023.1112245

Received: November 2, 2023
Accepted: December 22, 2023
Published: December 25, 2023

Abstract

Radial Basis Function methods for scattered data interpolation and for the numerical solution of PDEs were originally implemented in a global manner. Subsequently, it was realized that the methods could be implemented more efficiently in a local manner and that the local approaches could match or even surpass the accuracy of the global implementations. In this work, three localization approaches are compared: a local RBF method, a partition of unity method, and a recently introduced modified partition of unity method. A simple shape parameter selection method is introduced and the application of artificial viscosity to stabilize each of the local methods when approximating time-dependent PDEs is reviewed. Additionally, a new type of quasi-random center is introduced which may be better choices than other quasi-random points that are commonly used with RBF methods. All the results within the manuscript are reproducible as they are included as examples in the freely available Python Radial Basis Function Toolbox.

Keywords

Radial Basis Functions, Shape Parameter Selection, Quasi-Random Centers, Numerical PDEs, Scientific Computing, Open Source Software, Python Programming Language, Reproducible Research

1. Introduction

Radial basis function (RBF) methods are well-established for scattered data interpolation and for solving PDEs in complexly shaped domains. Contributing factors to the popularity of RBF methods are their simplicity, ease of use, and flexibility. The methods are simple and easy to use as their implementation only involves solving linear systems of equations and matrix-vector multiplication.

The extreme flexibility of RBF methods is due to the fact that there is complete freedom as to where centers may be located.

RBF methods were originally implemented in a global manner. The approximation of a derivative at one center used information from all other centers in a domain. In recent years researchers have realized that it is more efficient and that possibly greater accuracy could be obtained by implementing the methods in a local manner in which only a small subset of centers is used to calculate a derivative approximation at any one point. The localization methods include the local RBF method, a partition of unity (PU) method, and a modified PU method that was recently introduced. The modified PU method was introduced and applied in reference [1] but an analysis of the differences between the standard and modified PU methods was not given. The modified PU method has been subsequently applied in [2] and [3].

After briefly reviewing the global RBF method, the three localization methods are presented. All four methods are used to approximate derivatives at scattered center locations and to solve a time-dependent PDE problem. Scripts that reproduce all results in this manuscript are located in the folder */papers/JAMP2023/* of the Python Radial Basis Function Toolbox (PRBFT) distribution. The PRBFT, available from [4], is open source software that uses RBFs for interpolating scattered data and for the numerical solution of partial differential equations (PDEs). Throughout, modules names from the PRBFT are listed in *italic* type that implement the algorithms discussed. Also, listed in *italic* type are the names of scripts that produce the results in the manuscript.

The primary objective of this work is to demonstrate the effectiveness of local RBF methods in comparison to the global RBF method. In particular, a modified PU method is shown to be more efficient than the traditional PU method while still being as accurate as the traditional method. The modified PU method is especially efficient for approximating higher order derivatives. Secondary objectives include reviewing a regularization technique, artificial viscosity techniques, and shape parameter selection methods that can be implemented with both the global and local approaches. Finally, a new quasi-random center distribution will be examined that has superior properties when compared to the current quasi-random center distributions that are currently used in RBF methods.

2. Global RBF Method

A function $\Phi:\mathcal{R}^d \to \mathcal{R}$ is a radial basis function if there exists a function $\phi:[0,\infty) \to \mathcal{R}$ for which $\Phi(x)=\phi(r)$ where $r=\|x\|_2$ for all $x \in \mathcal{R}^d$. RBF interpolation uses a set of N distinct points $X=\{x_1^c,\cdots,x_N^c\}$ in $\mathcal{R}^d$ called centers. No restrictions are placed on the shape of problem domains or on the location of the centers. The RBF interpolant has the form

$$\mathcal{I}_N f(x)=\sum_{k=1}^{N} a_k \phi\left(\left\|x-x_k^c\right\|_2\right) \tag{1}$$

where a is a vector of expansion coefficients. The RBF expansion coefficients are

determined by enforcing the interpolation conditions

$$\mathcal{I}_N f\left(x_k^c\right) = f\left(x_k^c\right), \quad k = 1, 2, \cdots, N \tag{2}$$

which result in a $N \times N$ linear system

$$Ba = f. \tag{3}$$

The symmetric matrix B with entries

$$b_{jk} = \phi\left(\left\|x_j^c - x_k^c\right\|_2\right), \quad j, k = 1, \cdots, N \tag{4}$$

is called the system matrix.

The linear algebra routines in the PRBFT are based on the system matrix being symmetric positive definite (SPD) so that a Cholesky factorization is applicable. Three commonly used RBFs that result in a SPD system matrix which are included in the toolbox are the Gaussian (GA) RBF

$$\phi(r) = \mathrm{e}^{-\varepsilon^2 r^2}, \tag{5}$$

the inverse quadratic (IQ) RBF

$$\phi(r) = \frac{1}{1+\varepsilon^2 r^2}, \tag{6}$$

and the inverse multiquadric (IMQ) RBF

$$\phi(r) = \frac{1}{\sqrt{1+\varepsilon^2 r^2}}. \tag{7}$$

The GA, IQ, and IMQ are representative members of the class of global, infinitely differently RBFs containing a shape parameter, ε, that interpolate with exponential accuracy under suitable circumstances [5].

The evaluation of the interpolant (1) at M points x_j can be accomplished by multiplying the expansion coefficients by the $M \times N$ evaluation matrix H that has entries entries

$$h_{jk} = \phi_k\left(\left\|x_j - x_k^c\right\|_2\right), \quad j = 1, \cdots, M \text{ and } k = 1, \cdots, N. \tag{8}$$

Derivatives are approximated by differentiating the RBF interpolant as

$$\mathcal{L}\left(\mathcal{I}_N f(x)\right) = \sum_{k=1}^{N} a_k \mathcal{L}\phi\left(\left\|x - x_k^c\right\|_2\right) \tag{9}$$

where $\mathcal{L}$ is a linear differential operator. The operator $\mathcal{L}$ may be a single differential operator or a linear differential operator such as the Laplacian. Evaluating (9) at the centers X can be accomplished by multiplying the expansion coefficients by the evaluation matrix $H_{\mathcal{L}}$ with entries

$$h_{jk} = \mathcal{L}\phi\left(\left\|x_j^c - x_k^c\right\|_2\right), \quad j, k = 1, \cdots, N. \tag{10}$$

That is, $\mathcal{L}f \approx H_{\mathcal{L}} a$. Alternatively, derivatives can be approximated by multiplying the function values at the center locations, $\left\{f\left(x_k^c\right)\right\}_{k=1}^{N}$, by the differentiation matrix $D = H_{\mathcal{L}} B^{-1}$ since

$$\mathcal{L}f \approx H_{\mathcal{L}} a = H_{\mathcal{L}}\left(B^{-1} f\right) = \left(H_{\mathcal{L}} B^{-1}\right) f. \tag{11}$$

Recent monographs [5] [6] [7] [8] on RBF methods can be consulted for more information.

In the next sections, three topics are reviewed that are important in the implementation of RBF methods: the location of centers, regularization of ill-conditioned linear algebra operations, and the selection of a good value of the shape parameter.

2.1. Center Locations

While RBF methods work with randomly scattered centers, random centers have a tendency to clump up in certain areas while leaving other areas completely uncovered. For this reason quasi-random centers, such as (p_1, p_2)-Halton and p_1-Hammersley points, have become popular with RBF methods. The numbers p_1 and p_2 are prime numbers that are parameters to the methods. In particular, Halton centers are often used as a result of the points being featured in a popular RBF monograph [6]. Quasi-random sequences are often called low discrepancy sequences. The discrepancy of a sequence is computed by comparing the actual number of sample points in a given volume of multidimensional space with the number of sample points that should be there assuming a uniform distribution. In other words, the discrepancy of a sequence of centers measures how well they cover a domain. It is well-known that the discrepancies of both Halton and Hammersley sequences get larger as the number of space dimensions increase.

A new quasi-random sequence, called a R_d sequence, has been described in [9] (The reference is a blog post as the author does not work in academia and the work was not published in a journal). R_d sequences have superior low discrepancy characteristics when compared with Halton and Hammersely points. Additionally, the method is parameter free and is easy to implement efficiently. Figure 1 compares the coverage of a circular domain with Halton and R_2 points. For each dimension, the generation of R_d points is dependent on a constant ϕ_d that is the unique positive largest root of the equation $x^{d+1} - x - 1 = 0$. In one dimension $\phi_1 = \left(1+\sqrt{5}\right)/2$ which is the well-known golden ratio and in two dimensions $\phi_2 = 1.3247179572$ which has been termed the plastic constant.

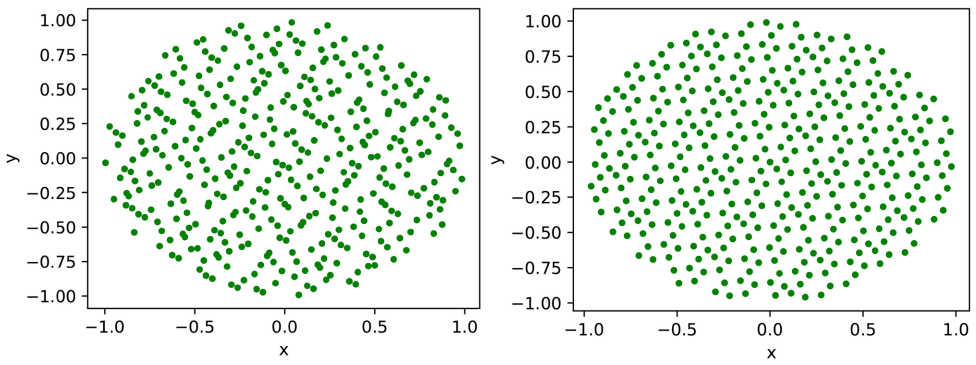

Figure 1. $N = 500$ centers on a circle of radius one (*quasiRandomCenters.py*). Left: quasi-random (2, 3) -Halton points provide a better coverage than do random points. Right: R_2 points provide a superior coverage compared to other quasi-random points.

The method is started by giving it an arbitrary point $\alpha_0 \in \mathcal{R}^d$. The choice of the initial point is arbitrary and does not affect the key characteristics of the distribution of points. After selecting the initial point, the rest of the points are found via the simple iteration

$$x_k = (\alpha_0 + k\alpha)(\text{mod}1), \quad k = 1,2,3,\cdots \tag{12}$$

where

$$\alpha = \left(\frac{1}{\phi_d}, \frac{1}{\phi_d^2}, \cdots, \frac{1}{\phi_d^d}\right).$$

2.2. MDI Regularization

Equations (3) for the expansion coefficients and (11) for the differentiation matrix assume that the system matrix is invertible. The IQ, GA, and IMQ system matrices are SPD and thus invertible. While invertible, the system matrix is typically very poorly conditioned. The eigenvalues of B satisfy $0 < \lambda_{\min} = \lambda_1 \le \lambda_2 \le \cdots \le \lambda_N = \lambda_{\max}$ and the matrix condition number in the 2-norm is $\kappa_2(B) = \lambda_{\max}/\lambda_{\min}$. For a fixed set of centers, the shape parameter affects both the accuracy of the method and the conditioning of the system matrix. Figure 2 shows the condition number of the system matrix and the error over a range of the shape parameter from interpolating the function

$$f(x) = e^{\sin(\pi x)} \tag{13}$$

on the interval $[-1,1]$ using a fixed number of $N = 60$ centers given by

$$x_c = \frac{\sin^{-1}\left(0.99\cos\left(k\pi/(N-1)\right)\right)}{\sin^{-1}(0.99)}, \quad k = 0,1,\cdots,N-1.$$

The centers cluster mildly near the boundaries which is beneficial to the accuracy of RBF methods. The interpolant is evaluated at 200 evenly spaced points. The results are typical of RBF methods in that the method is most accurate for smaller values of the shape parameter where the system matrix is ill-conditioned.

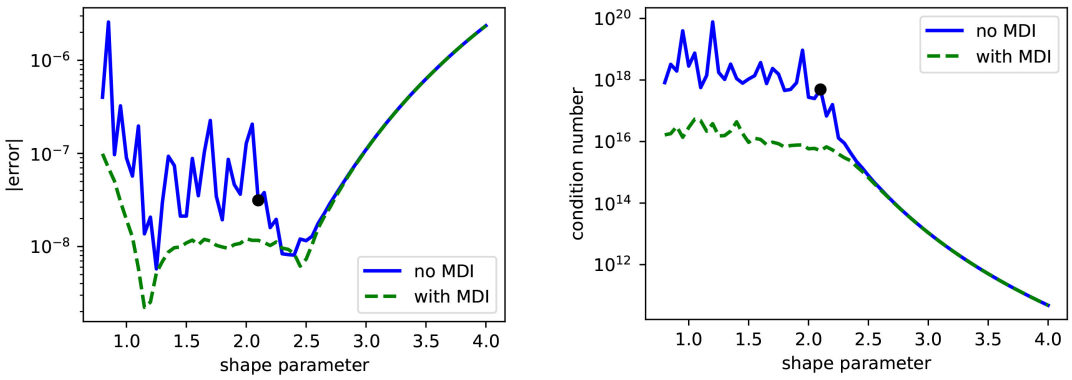

Figure 2. Results from interpolating the function $f(x) = e^{\sin(\pi x)}$ over a range of the shape parameter The script *mdiRegularization.py* produces the results. Left: accuracy versus the shape parameter. Right: system matrix condition number versus the shape parameter.

A challenging aspect of working with RBF methods is dealing with the poorly conditioned system matrix. A regularization technique known as the method of diagonal increments (MDI) can be used to mitigate the effects of ill-conditioning. MDI was first used in the 1940's [10] to regularize ill-conditioned linear systems resulting from the use of the normal equations in least squares problems in the field of Statistics. Reference [11] demonstrated that MDI regularization can be used to mitigate the effects of the poor conditioning of RBF system matrices and in most cases ensure that the theoretically SPD RBF system matrix remains numerically SPD (NSPD) so that Cholesky factorization can be used. Instead of solving the system

$$Ba = f \tag{14}$$

the regularized system

$$\left(B + \mu I\right)a = f \tag{15}$$

is considered. The parameter μ is a small positive constant called the regularization parameter and I is the identity matrix. The matrix $B + \mu I$ is better conditioned than B as the condition number κ satisfies

$$\kappa\left(B + \mu I\right) = \frac{\lambda_{\max} + \mu}{\lambda_{\min} + \mu} < \kappa\left(B\right) = \frac{\lambda_{\max}}{\lambda_{\min}}.$$

A good choice of the parameter is $\mu = 5\epsilon_M$ where ϵ_M is machine epsilon in the floating point number system being used.

In both images in **Figure 2** the dot on the solid blue line indicates the point at which the system matrix ceases to be NSPD as the shape parameter is decreased and for which a Cholesky factorization fails. In this example, the failure occurs at $\varepsilon \le 2.1$ and LU factorization is used to produce the remaining solid error curve for $\varepsilon \le 2.1$. With $\varepsilon = 2.1$, $\kappa\left(B\right) \approx 5.0\text{e}17$ and with smaller values of the shape parameter the condition number curve oscillates as the condition number can not be accurately calculated in double precision in this range.

The calculation is repeated using MDI with $\mu = 5\text{e}-15$ and the system matrix remains NSPD over the entire shape parameter range so that a Cholesky factorization is applicable. MDI reduces the condition number and allows better accuracy in the severely ill-conditioned range. With MDI the error curve is smoother and is free of the large oscillations that were present in the unregularized approximation.

2.3. NSPD Shape Parameter Selection

Methods used to select a value of the shape parameter include but are not limited to critical conditioning and leave one out cross validation (LOOCV). Critical conditioning [12] is a trial and error brute force method that selects the value of the shape parameter so that the condition number of the system matrix is in a range, for example $10^{14} \le \kappa\left(B\right) \le 10^{16}$, where the RBF method is most accurate. LOOCV [6] selects a shape parameter by minimizing the least squares error of an interpolant for which one of the centers has been left out.

The approach used here is based on the system matrix being NSPD. In the left image Figure 2 the smallest error occurs when $\varepsilon = 1.15$. Of course in real problems where the exact function is unknown this information is unavailable. Instead, an effective and safe choice is to use the value at which the system matrix first ceases to be NSPD as the shape parameter is decreased. In the example of Figure 2 the shape parameter is selected as $\varepsilon_0 = 2.1$. After ε_0 is selected, the calculation is then repeated using MDI. In the example, the regularized solution is approximately one decimal place more accurate at ε_0.

2.4. An Example Derivative Approximation

For the purposes of introducing the methods, the operator $\mathcal{L}_1 = u_x + u_y$ and the Laplacian operator $\mathcal{L}_2 = u_{xx} + u_{yy}$ are approximated for the function

$$u(x,y) = \mathrm{e}^{(x/2+y/5)} \cos(xy) \tag{16}$$

on a circular domain of radius one that is centered at $(1/2, 1/2)$. The domain is discretized with $N = 1177$ R_2 points. The global RBF, the local RBF method, the RBF PU method, and a modified RBF PU method will be used to approximate the two differential operators. The three local methods will be implemented so that the number of supporting centers is the same. The criteria used evaluate the methods will be their error in the infinity norm.

The script *differentiationGlobal.py* uses the global RBF method to evaluate the derivatives of function (16). The max errors in approximating $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively are 9.42e-04 and 8.64e-02. A shape parameter of $\varepsilon = 1.84$ was selected via the NSPD approach of section 2.3 and a MDI regularization parameter of $\mu = 5\mathrm{e}{-15}$ was used.

3. Why Localize

A comparison of the global RBF method with another well-known global method, the Chebyshev pseudospectral (CPS) method [13], is useful. In 1d the CPS method is implemented on a grid consisting of the Chebyshev-Gauss-Lobatto (CGL) points, $x_k = -\cos\left(k\pi/(N-1)\right)$ where $k = 0,1,\cdots,N-1$, which cluster densely around the endpoints of the interval $[-1,1]$ and are more sparsely located in the interior of the interval. In 2d, the CPS method is implemented on a tensor product grid of the 1d grid.

Consider differentiating a function on a 2d tensor product grid constructed from a 1d grid with $N = 16$ CGL points and let the unknowns be organized in a $N^2 \times 1$ vector. The approximate derivative is calculated via the matrix-vector multiplication $f_d = Df$ where the differentiation matrix (DM) D is $N^2 \times N^2$ with $N^4 = 65536$ elements. However, only 12.1 percent of the elements are non-zero due to the orthogonality of the basis functions. The sparsity structure of the CPS DM is shown in the left image of Figure 3. As N increases to 32, 64, 96, and 128 the percentage of nonzero elements drops respectively to 6.2, 3.1, 2.1, and 1.6 percent.

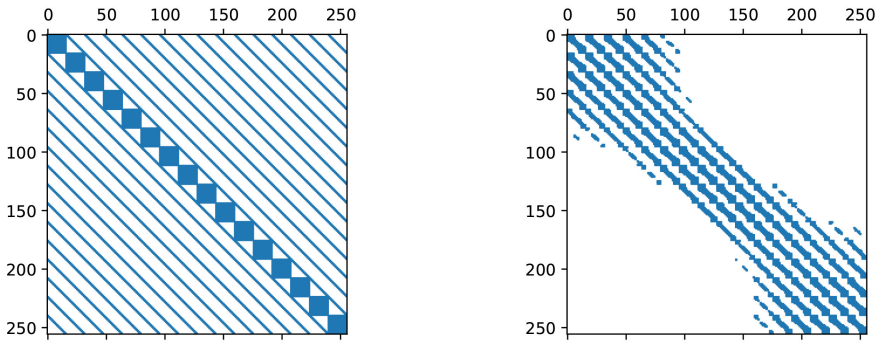

Figure 3. Sparsity of differentiation matrices. Nonzero elements are shaded blue: Left: The global Chebyshev pseudospectral method. Right: The local RBF method.

In contrast, all of the elements are nonzero in the global RBF method DM. As constructed, the global RBF method uses too many supporting centers in order to calculate a derivative at one point which can lead to unreasonably high computational costs and a reduction of accuracy. If the local RBF method were to be implemented on the same tensor product grid as the CPS method with a stencil size $n = 2N - 1$, its DM would have the same number of nonzero elements as does the CPS DM. The sparsity structure of the local RBF DM with $n = 31$ is shown in the right image of **Figure 3**.

The derivative is an inherently local property. Thus only using points in the immediate neighborhood of the point where a derivative is being calculated intuitively makes more sense than using every center in the entire domain. It is easy to construct an example to illustrate the consequences of using the global method with too many centers and to illustrate that the local method can be more accurate than the global method. To do so, the operator $\mathcal{L}_1 = f_x + f_y$ is approximated for the well-known Franke function [14]

$$\begin{aligned} f(x,y) = &\frac{3}{4}\mathrm{e}^{\left[\frac{-1}{4}(9x-2)^2-\frac{1}{4}(9y-2)^2\right]} + \frac{3}{4}\mathrm{e}^{\left[\frac{-1}{49}(9x+1)^2-\frac{1}{10}(9y+1)^2\right]} \\ &+\frac{1}{2}\mathrm{e}^{\left[\frac{-1}{4}(9x-7)^2-\frac{-1}{4}(9y-3)^2\right]} - \frac{1}{5}\mathrm{e}^{\left[-(9x-4)^2-(9y-7)^2\right]} \end{aligned} \tag{17}$$

that is frequently used in conjunction with RBF methods. The calculation is done in the complexly shaped domain in **Figure 4**.

First, the number of centers is fixed at $N = 5000$ and the stencil size of the local method is varied from 10 to 200. The max error versus stencil size is shown in the left image of **Figure 5**. In this example, the best accuracy occurs with $n = 50$. The optimal stencil size is problem dependent, but in two space dimensions the optimal size is usually between 20 and 100. In the right image of **Figure 5**, the number of centers N is varied from 1000 to 5000 and the local method stencil size is fixed at $n = 40$. For $N \le 1800$, the global method is more accurate than the local method with $n = 40$ but as N is increased the error of the global method does not decrease. Rather it stagnates around 0.001. On the other hand, the error from the local method continues to decrease as N increases and at $N = 5000$ the local method is significantly more accurate than the global method.

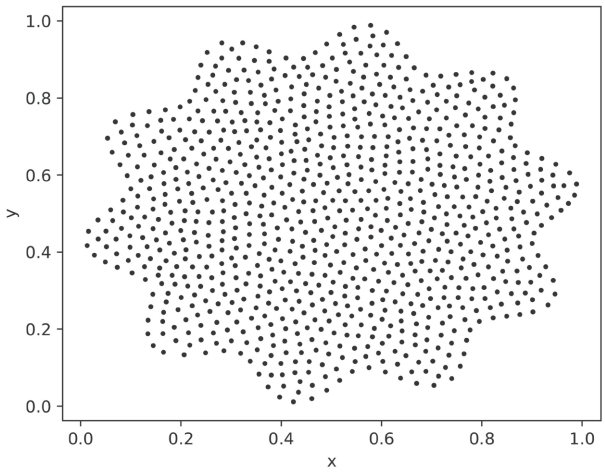

Figure 4. $N = 1100$ scattered centers on a complexly shaped domain. The operator $\mathcal{L}_1$ is approximated for the Franke function (17) on the domain.

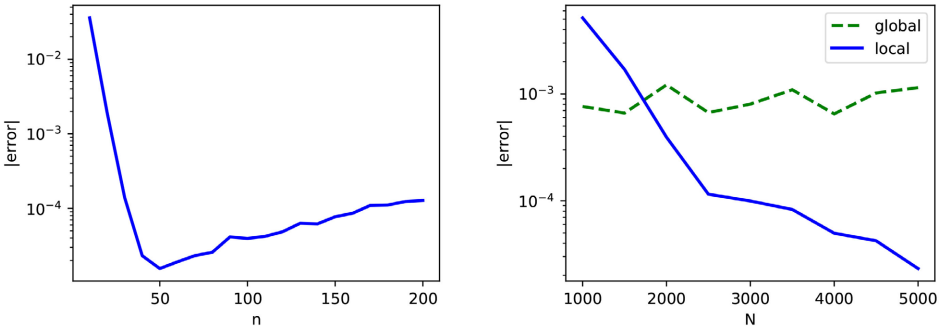

Figure 5. Global versus local derivative approximation. Left: fixed number of centers ($N = 5000$) and varying stencils size n (*globalVsLocalFixedN.py*). Right: fixed stencils size $n = 40$ and varying number of centers (*globalVsLocalVaryN.py*).

4. How to Localize

There are multiple ways that the RBF method can be localized. The ways include: the local RBF (LRBF) method (also called the RBF finite difference method but not used here due to its connection with polynomial based methods), a partition of unity (PU) approach that originated in the field of finite elements, and a modified partition of unity (MPU) method that was first used with a rational RBF method in reference [1]. With each of the three localization approaches, various options exist on how to implement particular elements of the methods such as shape parameter selection, regularization, stencil size, center locations, application of artificial viscosity for stable time-stepping, etc.

4.1. LRBF—The Local RBF Method

Variations of the local RBF method for the solution of steady and time-dependent PDEs were first used in references [15] [16] and [17]. Recent applications of local RBF methods include [18] [19] [20] [21].

At each of the N centers, the local RBF method considers a local interpolant of

the form

$$\mathcal{I}_n f(x) = \sum_{k \in I_i} a_k \phi\left(\left\| x - x_k^c \right\|_2, \varepsilon_i\right) \tag{18}$$

where a is a vector of expansion coefficients and I_i is a vector associated with center i that contains the center number and the indices of the $n-1$ nearest neighboring centers. The shape parameter ε_i may be different at each center. Each center and its $n-1$ neighbors are called a stencil. Two typical stencils, one for a boundary center and one for an interior center, are shown in the left image of Figure 6. The stencils are efficiently constructed using a kd-tree structure [22].

Enforcing the interpolation conditions

$$\mathcal{I}_n f(x_k) = f_k, \quad k \in I_i \tag{19}$$

on each stencil gives N, $n \times n$ linear systems

$$Ba = f \tag{20}$$

to be solved for the expansion coefficients where the matrices B are local system matrices with elements

$$b_{jk} = \phi\left(\left\| x_j^c - x_k^c \right\|_2, \varepsilon_i\right), \quad j,k = I_i(1), \cdots, I_i(n). \tag{21}$$

In a manner analogous to that of the global method, the local method approximates derivatives of a function f at the center locations as

$$\mathcal{L}f(x_i) = \sum_{k \in I_i} a_k \mathcal{L}\phi\left(\left\| x_i^c - x_k^c \right\|_2, \varepsilon_i\right). \tag{22}$$

Equation (22) can be written more concisely as a dot product

$$\mathcal{L}f(x_i) = h \cdot a \tag{23}$$

where a is the $n \times 1$ vector of RBF expansion coefficients and h is a $1 \times n$ vector containing the elements

$$h_i = \mathcal{L}\phi\left(\left\| x_i^c - x_k^c \right\|_2, \varepsilon_i\right), \quad k \in I_i. \tag{24}$$

The dependence on the RBF expansion coefficients can be removed from (23) by noting that

$$\mathcal{L}f(x_i) = hB^{-1} f(I_i) = \left(hB^{-1}\right) f(I_i) = w \cdot f(I_i) \tag{25}$$

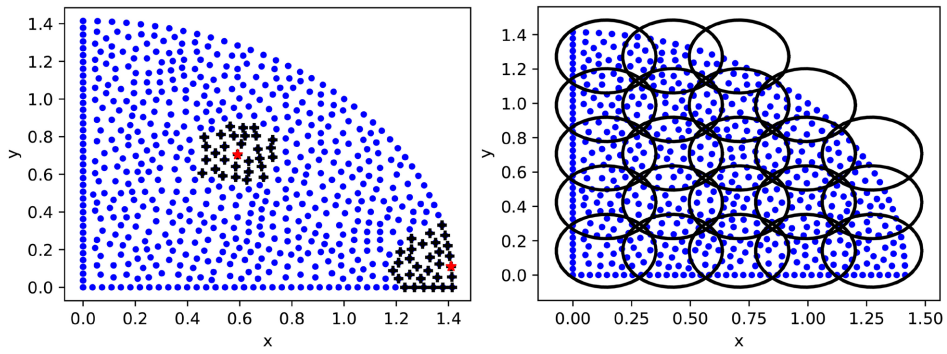

Figure 6. Two ways to localize (*localMethodImages.py*). Left: The local RBF method. Right: A partition of unity approach.

where the stencil weights are given by

$$w = hB^{-1}. \tag{26}$$

That is, the weights are the solution of the linear system

$$wB = h. \tag{27}$$

Then space derivatives are approximated by multiplying the function values at the centers of the stencil by the weights.

The script *differentiationLocal.py* uses the local RBF method to evaluate the derivatives of function (16) with a stencil size of $n = 80$. The max errors in approximating $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively are 8.20e-05 and 7.60e-03. NSPD shape parameter selection was used on each stencil and a MDI regularization parameter of $\mu = 5\text{e}-15$ was used in solving the linear systems for the weights. In this example, the local method is more accurate than is the global method.

4.2. PU—RBF Partition of Unity

In the context of scattered data interpolation, it appears that a partition of unity (PU) method was first described in [23] and was later analyzed in more detail in [24]. Reference [25] details the application of the PU method to the numerical solution of PDEs by the finite element method. A recent use of the PU method with RBFs is described in [26].

A partition of unity method is implemented by constructing an overlapping covering $\{\Omega_i\}_{i=1}^{M}$ of the domain Ω. Each Ω_i in the covering is called a patch. For simplicity, it is assumed that the patches are circles with centers $c_i^0 = \left(x_i^0, y_i^0\right)$ and radii R_i. The right image in **Figure 6** displays an example cover of a domain. In general, it is not necessary that the patches be circular, as they may be shaped as squares, ellipses, etc. The PU method is applicable in higher dimensions as well, for example in 3d the patches could be cubes or spherical in shape. Associated with each center in Ω is the index function

$$P_i\left(x_k^c\right) = \left\{i \mid x_k^c \in \Omega_i\right\} \quad i = 1, \cdots, M \tag{28}$$

which is used to keep track of how many patches that each center is located in.

A family of compactly supported, non-negative, continuous functions $\{w_i\}$ which are constructed via Shephard's method [27] as

$$w_i\left(x_k^c\right) = \frac{C_i\left(x_k^c\right)}{\sum_{j \in P_i\left(x_k^c\right)} C_j\left(x_k^c\right)} \quad i = 1, \cdots, M. \tag{29}$$

is used to implement the method. A possible choice for $C_i(x)$ in (29) is a compactly supported Wendland function [28] RBF such as W32

$$C(r) = (1-r)_+^6\left(35r^2 + 18r + 3\right) \tag{30}$$

where

$$(1-r)_+^6 = \begin{cases} (1-r)^6 & 0 \le r < 1 \\ 0 & r > 1 \end{cases} \tag{31}$$

W32 is in C^4 and is positive definite in up to three space dimensions. Compact support on each patch is guaranteed if the Wendland functions (30) used to construct the weight functions (29) are specified as

$$C_i(x) = C\left(\frac{\|x - c_i\|_2}{R_i}\right).$$

As a result, the weight functions satisfy the partition of unity property

$$\sum_{i \in P_i\left(x_k^c\right)} w_i(x) = 1.$$

Additionally, $w_i(x) = 0$ if $i \notin P_i(x)$. A function f is approximated locally by s_i on each patch Ω_i. Then due to properties of the weight function the local approximations are put together as

$$s(x) = \sum_{i \in P_i(x)} s_i(x) w_i(x). \tag{32}$$

The approximation at each x is a linear combination of the local approximations on each patch containing x with the largest weight being placed on patches where x is located near the center of the patch and less weight is placed on patch approximations where x is located near the boundary of a patch.

Derivatives are approximated by applying a linear differential operator $\mathcal{L}$ to (32) as

$$\mathcal{L}s(x) = \sum_{i \in P_i(x)} \mathcal{L}\left[s_i(x) w_i(x)\right]. \tag{33}$$

A drawback of the method is that as the order and complexity of $\mathcal{L}$ increase, the repeated use of the product rule applied to (33) and the quotient rule applied to (29) can quickly result in an unwieldy expression containing a large number of derivatives to evaluate. For example, consider a function of two variables (x_1, x_2) and let $\mathcal{L} = \frac{\partial}{\partial x_1}$. Then

$$\frac{\partial}{\partial x_1} s(x) = \sum_{i \in P_i(x)} s_i(x) \frac{\partial}{\partial x_1} w_i(x) + w_i(x) \frac{\partial}{\partial x_1} s_i(x) \tag{34}$$

where

$$\frac{\partial}{\partial x_1} w_i(x) = \frac{\frac{\partial}{\partial x_1} C_i(x) \sum_{j \in P_i(x)} C_j(x) - C_i(x) \sum_{j \in P_i(x)} \frac{\partial}{\partial x_1} C_j(x)}{\left[\sum_{j \in P_i(x)} C_j(x)\right]^2}. \tag{35}$$

If $\mathcal{L} = \frac{\partial^2}{\partial x_1^2}$, then

$$\frac{\partial}{\partial x_1} s(x) = \sum_{i \in P_i(x)} s_i(x) \frac{\partial^2}{\partial x_1^2} w_i(x) + 2 \frac{\partial}{\partial x_1} w_i(x) \frac{\partial}{\partial x_1} s_i(x) + w_i(x) \frac{\partial^2}{\partial x_1^2} s_i(x) \tag{36}$$

and the second partial derivative of the weight function is becoming unwieldy after Equation (35) is differentiated again. For this reason, the module *rbfPU.py* only implements the PU method for first and second order derivatives.

The script *differentiationPU.py* uses the RBF partition of unity method to evaluate derivatives of function (16). The max errors in approximating $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively are 1.30e-04 and 7.94e-03. The NSPD method was used to select the shape parameter on each patch and a MDI regularization parameter of $\mu = 5\text{e}-15$ was used in forming the local differentiation matrices for each patch. The cover of the domain consists of 25 circles. In this example, the PU method was more accurate than the global method but not as accurate as the local method.

4.3. MPU—Modified RBF Partition of Unity

In order to overcome the major drawback of the PU method the following modification has been proposed [1]. In the modified method, values of the weight function (29) still depend on the location of x within a patch, but then are considered constants rather than functions of x for approximating derivatives which the modified method calculates as

$$\mathcal{L}s(x) = \sum_{i \in P_I(x)} w_i \mathcal{L}s_i(x). \tag{37}$$

Derivative values at a point x are simply linear combinations of their values from each patch the point is contained in. That is, derivative values at a point in the MPU method are put together exactly as interpolants are in the PU method. For first derivatives, the two approaches can be seen to be equivalent. However, for higher order derivatives there is a slight difference in the two methods. It is problem dependent as to which approach is most accurate. In the examples within, the MPU method is slightly more accurate in approximating derivatives above order one and is significantly simpler to implement.

The script *differentiationSarraPU.py* uses the RBF modified partition of unity method to evaluate derivatives of function (16). The max errors in approximating $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively are 1.28e-04 and 6.70e-03. The NSPD method was used to select the shape parameter on each patch and a MDI regularization parameter of $\mu = 5\text{e}-15$ was used in forming the local differentiation matrices for each patch. In this example, the modified PU method is slightly more accurate than is the standard PU method. As was the case in the standard PU example, the cover of the domain consists of 25 circles. This number was selected because it results in 80 being the average number of centers in each patch which is the same as the stencil size used by the local RBF method in this example (**Figure 7**).

4.4. PU and Modified PU Method Differences

For first order derivatives, the term

$$\sum_{i \in P_I(x)} s_i(x) \frac{\partial}{\partial x} w_i(x) \tag{38}$$

in (34) is zero and the PU and modified PU methods are equivalent. For second order derivatives, the term

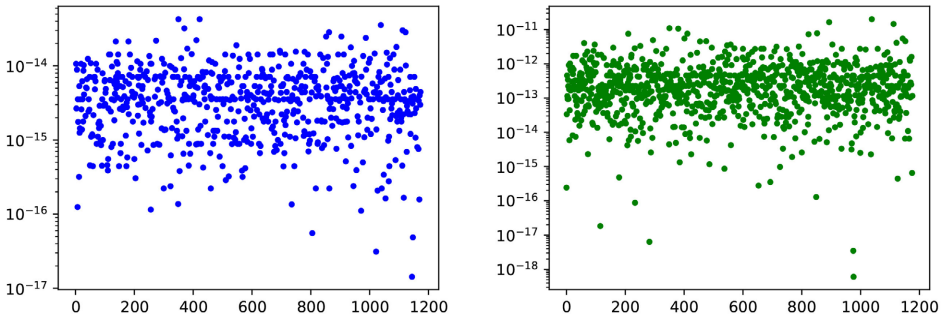

Figure 7. Magnitude of the terms (38) in the left image and (39) in the right image (vertical axis) verus center number (horizontal axis) that are theoretically zero but may not be in floating point arithmetic.

$$\sum_{i\in P_i(x)} s_i(x)\frac{\partial^2}{\partial x_1^2} w_i(x) \tag{39}$$

in (36) is zero. The only difference between the PU and modified PU methods for second order derivatives is the term

$$2\frac{\partial}{\partial x_1} w_i(x)\frac{\partial}{\partial x_1} s_i(x). \tag{40}$$

that appears in Equation (36) of the PU method but is not in Equation (37) of the modified PU method.

While the terms (39) and (40) are theoretically zero, they are often not zero in floating point arithmetic. This is especially true when a large number of centers are in a patch. For this reason they are not included in the function *rbfPU.globalDM* that uses a differentiation matrix based approach while the function *rbfPU.evaluateDerivates* has a logical variable as an argument so that the user may experiment with the effects of including or not including the terms.

Returning to the approximation of $\mathcal{L}_1$ and $\mathcal{L}_2$ from section 4.2, **Figure 8** shows the magnitude of the terms that are theoretically zero in the PU method. The values of (38) are relatively close to machine epsilon while the values of (39) are larger. However, either including the terms or disregarding them are of little consequence as in this calculation the approximate solution has only two decimal places of accuracy.

The maximum value of the term (40) for $\mathcal{L}_2$ is 2.56e-3 while the overall max error of the modified PU method for $\mathcal{L}_2$ is 6.70e-3 which is smaller than the standard PU method error of 7.94e-3. In this example, neglecting the term resulted in the modified PU method being more accurate. In all applications of the method so far by the current author, the order of magnitude of the term neglected by the MPU method is the same as that of the error and thus has little effect on the overall accuracy of the method. In reference [2], where the MPU method was also applied, the authors came to a similar conclusion and they also argued that the term could safely be neglected.

5. Time-Dependent PDEs

The non-homogeneous diffusion equation

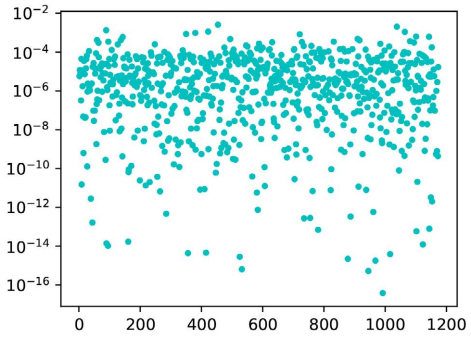

Figure 8. Magnitude of the neglected term (40) (vertical axis) verus center number (horizontal axis) by the modified PU method in approximating $\mathcal{L}_2$.

$$u_t = u_{xx} + u_{yy} + 4k^2\left(x^2+y^2\right)\cos\left(k\left[t-x^2-y^2\right]\right) - 5k\sin\left(k\left[t-x^2-y^2\right]\right) \tag{41}$$

is solved on a circular domain of radius 1 that is centered at (1, 1). The domain is discretized with $N = 4684$ scattered R_2 centers in the interior of the domain and 350 evenly spaced centers on the boundary. The initial condition and Dirichlet boundary conditions are taken from the exact solution $u(x,y,t) = 1+\cos\left(k\left[t-x^2-y^2\right]\right)$. In the following examples $k = 2$ has been used.

The global, local, PU, and MPU methods are used to discretize the space derivatives of the PDE using the GA RBF and then a fourth-order explicit Runge-Kutta method (*rbfMisc.rk4*) with a small constant time-step size of $\Delta t = 4\text{e}-5$ is used to advance the solution to time $t = 0.5$. Each method is regularized via the method of diagonal increments from section 2.2 with a regularization parameter of $\mu = 5\text{e}-15$. In each method the shape parameter is selected via the NSPD technique from section 2.3. The local method uses a stencil size of $n = 25$ and both partition of unity methods use a total of 25 covering circles.

At the final time, the global method (*heatCircleGlobal.py*) relative max error is 1.23e-03, the PU method (*heatCirclePu.py*) relative max error is 3.77e-04, and the modified PU method (*heatCircleSarraPu.py*) relative max error is 3.32e-04.

As it is, the local method can not be advanced in time as its differentiation matrix has eigenvalues with positive real parts as large as 4000 (left image of Figure 9). The next section discusses how the addition of artificial viscosity (dissipation) can remedy the situation.

Artificial Viscosity

Eigenvalue stability for time-dependent PDEs with space derivatives discretized by the RBF method is a well-documented issue, see for example [29] [30]. Frequently, eigenvalues of the discretized space operator have large positive real parts which makes stable time integration by explicit methods impossible. This is especially true for advection-type PDEs that do not have any inherent stabilizing dissipation. Even for purely dissipative PDE such as the diffusion equation (41), eigenvalue stability may still be an issue when local RBF methods are used.

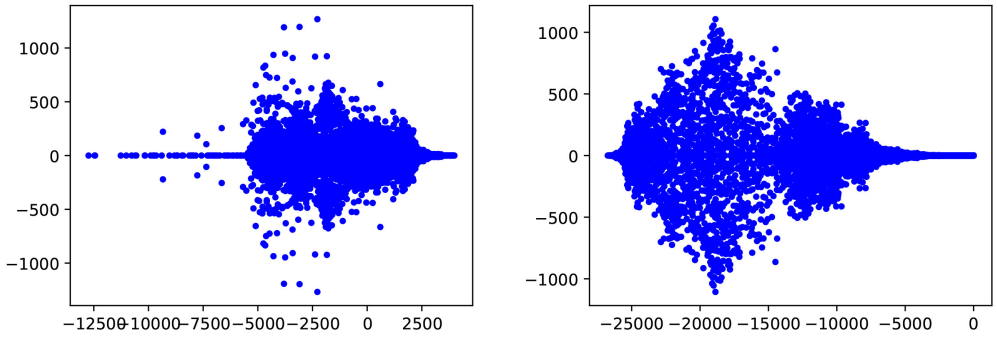

Figure 9. Eigenvalues of the differentiation matrix in the complex plane (real part on the horizontal axis and imaginary part on the vertical axis) that discretizes equation (41) with the local RBF method and stencil size $n=25$. Left: without artificial viscosity. Right: with artificial viscosity of the inverse system matrix filter type with viscosity parameter $\mu=1\mathrm{e}-10$.

There are multiple factors that contribute to the eigenvalue stability problem. The instability issues may be inherent to the method, caused by poor conditioning of the system matrix, or the result of careless choices in the linear algebra routines that are used. If the condition number of the system matrix is larger than $\mathcal{O}\left(10^{16}\right)$ the likelihood of problems increase. LU factorization ignores symmetry and its use on a SPD system matrix rather than a more appropriate Cholesky factorization can contribute to eigenvalue stability problems. The combination of NSPD shape parameter selection and the use of MDI with Cholesky factorization greatly reduce two of the factors. The remaining factor, that the problem is inherent to the method itself, leads to artificial viscosity being added to the discretization.

Reference [31] suggests two ways to add artificial viscosity to a RBF method via application of a viscosity matrix D^{ν}. The first approach uses the inverse system matrix (ISM) which damps the spurious eigenvalues while leaving the correct eigenvalues intact. In the global and PU methods, D^{ν} is constructed as the solution of the linear system

$$D^{\nu}B=-\nu I$$

where I is an identity matrix and ν is the viscosity coefficient. The viscosity matrix can be combined with the discretization of another linear operator as

$$D_{\mathcal{L}}^{\nu}B=H_{\mathcal{L}}-\nu I$$

where $H_{\mathcal{L}}$ is a derivative evaluation matrix. In the local method, the viscosity matrix is added by modifying Equation (27) to be

$$wB=h-\nu\cdot(1,0,\cdots,0). \tag{42}$$

A second approach is to add a power of the discretized Laplacian (LP) operator $\nu\Delta^{k}$. In this case the viscosity matrix is formed just as is any other differentiation matrix (Equations (9)-(11)) and the viscosity can be combined with the discretization of other linear differential operators. Each RBF class in the PRBFT has a function for the first power of the Laplacian but not higher powers. To get higher powers the first power of the operator can be repeatedly applied or else the Gaussian RBF can be used for which an explicit formula for any power

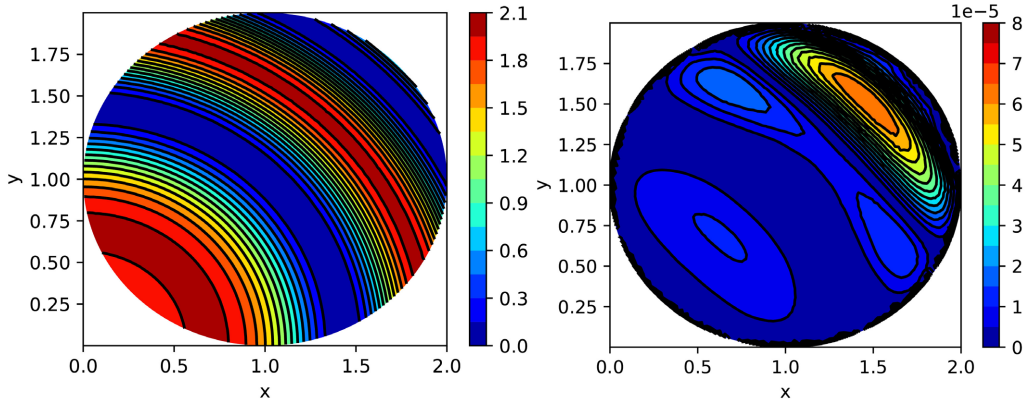

Figure 10. Solution of PDE (41) by the local method with $n=25$. Left: solution at time $t=0.5$. Right: relative point-wise errors at $t=0.5$.

of the operator applied to the GA was given in [31]. The function *gax.hyperViscosity* is programmed with the explicit formula.

In [31] it is recommend that the ISM filter be used with global methods and that the LP viscosity be used with local methods. In our experience, either approach can be used in either the global or local setting but the ISM filter is easier to use as it only contains one user specified parameter, ν, while the LP viscosity contains two, ν and the power k. The function *rbfLocal.weights* can apply either or both of the artificial viscosity types depending on the choice of input parameters.

In some problems, especially when the PDE does not have any natural dissipative terms, it can be difficult (in some cases seemingly impossible) to find values of the parameters ν and/or k that effectively move all the eigenvalues to the left part of the complex plane. Additionally, one must make sure to use the smallest ν possible or else the real parts of the eigenvalues will be moved far into the left part of the complex plane and extremely small time steps will be required for stability with explicit methods. For example in **Figure 9**, $\mu=1\text{e}-10$ was the smallest value to be used with the ISM filter to successfully shift the eigenvalues and the magnitude of the negative real parts of the eigenvalues has approximately doubled.

After adding artificial viscosity, the local method can be used to solve (41). At the final time, the local method (*heatCircleLocal.py*) relative max error is 4.00e-05. The solution and point-wise error are shown in **Figure 10**. The method was stabilized by adding ISM type viscosity with $\mu=1\text{e}-10$. Alternatively, LP type artificial viscosity with $k=2$ and $\nu=1\text{e}-13$ can be used to stabilize the problem and the result has the exact same error as when the ISM filter was applied. In summary, of the four methods applied to the time-dependent PDE problem, the local method with artificial viscosity was most accurate followed by the MPU, PU, and global methods.

6. Conclusions

Four ways to implement RBF methods for the numerical solution of PDEs have

been discussed: global, local, partition of unity, and modified partition of unity. The local method can outperform the global method by selecting a stencil size that results in a sparse differentiation matrix. The standard version of the RBF PU method is an effective localization method for PDE problems with only first-order derivatives or even possibly second-order derivatives. However the complexity of the method quickly rises as higher order derivatives appear in problems. For this reason, a modified PU method has been implemented that calculates derivatives simply as linear combinations of patch results rather than using the product rule and quotient rule. The differences between the standard and modified PU methods have been examined. The modified PU method is more efficient than the standard PU method and on example problems the two methods have comparable accuracy. In our future work, the modified PU method will be used to approximate high-order (greater than two) space derivatives in the numerical solution of PDEs. All three of the local methods demonstrated comparable accuracy in the example problems. Additionally, all three of the local methods were more accurate than the global method in the examples.

A new type of quasi-random center distribution, R_d points, have been have been used as centers with RBF methods. The R_d points have a lower discrepancy than the Halton and Hammersley points which are commonly used with RBF methods. A shape parameter selection algorithm was presented that is based on the system matrix being numerically symmetric positive definite and the method was used to select the shape parameter for all four RBF methods. For time-dependent PDE problems, two ways to add artificial viscosity to ensure eigenvalue stability were discussed and it was detailed how the addition of artificial viscosity can be implemented with each method.

All results in this manuscript are reproducible via example scripts in the Python Radial Basis Function Toolbox [4].

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Sarra, S.A. and Bai, Y. (2018) A Rational Radial Basis Function Method for Accurately Resolving Discontinuities and Steep Gradients. *Applied Numerical Mathematics*, **130**, 131-142. https://doi.org/10.1016/j.apnum.2018.04.001

[2] Aiton, K.W. and Driscoll, T.A. (2019) An Adaptive Partition of Unity Method for Multivariate Chebyshev Polynomial Approximations. *SIAM Journal on Scientific Computing*, **41**, A3230-A3245. https://doi.org/10.1137/18M1184904

[3] Mirzaei, D. (2021) The Direct Radial Basis Function Partition of Unity (D-RBF-PU) Method for Solving PDEs. *SIAM Journal on Scientific Computing*, **43**, A54-A83. https://doi.org/10.1137/19M128911X

[4] Sarra, S.A. (2023) The Radial Basis Function Toolkit. http://www.scottsarra.org/rbf/rbf.html

[5] Buhmann, M.D. (2003) Radial Basis Functions. Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9780511543241

[6] Fasshauer, G.E. (2007) Meshfree Approximation Methods with Matlab. World Scientific, Singapore. https://doi.org/10.1142/6437

[7] Sarra, S.A. and Kansa, E.J. (2009) Multiquadric Radial Basis Function Approximation Methods for the Numerical Solution of Partial Differential Equations. *Advances in Computational Mechanics*, **2**, 1-199.

[8] Wendland, H. (2005) Scattered Data Approximation. Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9780511617539

[9] Roberts, M. (2018) The Unreasonable Effectiveness of Quasirandom Sequences. http://extremelearning.com.au/unreasonable-effectiveness-of-quasirandom-sequences/

[10] Piegorsch, W. and Casella, G. (1989) The Early Use of Matrix Diagonal Increments in Statistical Problems. *SIAM Review*, **31**, 428-434. https://doi.org/10.1137/1031089

[11] Sarra, S.A. (2014) Regularized Symmetric Positive Definite Matrix Factorizations for Linear Systems Arising from RBF Interpolation and Differentiation. *Engineering Analysis with Boundary Elements*, **44**, 76-86. https://doi.org/10.1016/j.enganabound.2014.04.019

[12] Chenoweth, M. and Sarra, S.A. (2009) A Numerical Study of Generalized Multiquadric Radial Basis Function Interpolation. *SIAM Undergraduate Research Online*, **2**, 58-70. https://doi.org/10.1137/09S01040X

[13] Canuto, C., Hussaini, M.Y., Quarteroni, A. and Zang, T.A. (1988) Spectral Methods for Fluid Dynamics. Springer-Verlag, New York. https://doi.org/10.1007/978-3-642-84108-8

[14] Franke, R. (1982) Scattered Data Interpolation: Tests of Some Methods. *Mathematics of Computation*, **38**, 181-200. https://doi.org/10.1090/S0025-5718-1982-0637296-4

[15] Tolstykh, A.I., Lipavskii, M.V. and Shirobokov, D.A. (2003) High-Accuracy Discretization Methods for Solid Mechanics. *Archives of Mechanics*, **55**, 531-553.

[16] Tolstykh, A.I. and Shirobokov, D.A. (2003) On Using Radial Basis Functions in Finite Difference Mode with Applications to Elasticity Problems. *Computational Mechanics*, **33**, 68-79. https://doi.org/10.1007/s00466-003-0501-9

[17] Shu, C., Ding, H. and Yeo, K.S. (2003) Local Radial Basis Function-Based Differential Quadrature Method and Its Application to Solve Two-Dimensional Incompressible Navier Stokes Equations. *Computer Methods in Applied Mechanics and Engineering*, **192**, 941-954. https://doi.org/10.1016/S0045-7825(02)00618-7

[18] Hosseini, B. and Hashemi, R. (2011) Solution of Burgers' Equation Using a Local-RBF Meshless Method. *International Journal for Computational Methods in Engineering Science and Mechanics*, **12**, 44-58. https://doi.org/10.1080/15502287.2010.540303

[19] Shan, Y., Shu, C. and Qin, N. (2009) Multiquadric Finite Difference (MQ-FD) Method and Its Application. *Advances in Applied Mathematics and Mechanics*, **5**, 615-638. https://doi.org/10.4208/aamm.09-m0942

[20] Shen, Q. (2011) Numerical Solution of the Sturm-Liouville Problem with Local RBF-Based Differential Quadrature Collocation Method. *International Journal of Computer Mathematics*, **88**, 285-295. https://doi.org/10.1080/00207160903370180

[21] Sanyasiraju, Y. and Chandhini, G. (2009) A RBF Based Local Gridfree Scheme for Unsteady Convection-Diffusion Problems. *CFD Letters*, **1**, 59-67.

[22] Bentley, J.L. (1975) Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, **18**, 509-517. https://doi.org/10.1145/361002.361007

[23] Franke, R. (1977) Locally Determined Smooth Interpolation at Irregularly Spaced Points in Several Variables. *IMA Journal of Applied Mathematics*, **19**, 471-482. https://doi.org/10.1093/imamat/19.4.471

[24] Wendland, H. (2002) Fast Evaluation of Radial Basis Functions: Methods Based on Partition of Unity. In: *Approximation Theory X*: *Wavelets*, *Splines*, *and Applications*, Vanderbilt University Press, Nashville, 473-483.

[25] Babuka, I. and Melenk, J.M. (1997) The Partition of Unity Method. *International Journal for Numerical Methods in Engineering*, **40**, 727-758. https://doi.org/10.1002/(SICI)1097-0207(19970228)40:4<727::AID-NME86>3.0.CO;2-N

[26] Safdari-Vaighani, A., Heryudono, A. and Larsson, E. (2014) A Radial Basis Function Partition of Unity Collocation Method for Convection-Diffusion Equations Arising in Financial Applications. *Journal of Scientific Computing*, **64**, 341-367. https://doi.org/10.1007/s10915-014-9935-9

[27] Shepard, D. (1968) A Two-Dimensional Interpolation Function for Irregularly-Spaced Data. *Proceedings of the* 1968 23*rd ACM National Conference*, New York, 27-29 August 1968, 517-524. https://doi.org/10.1145/800186.810616

[28] Wendland, H. (1995) Piecewise Polynomial, Positive Definite and Compactly Supported Radial Basis Functions of Minimal Degree. *Advances in Computational Mathematics*, **4**, 389-396. https://doi.org/10.1007/BF02123482

[29] Platte, R. and Driscoll, T. (2006) Eigenvalue Stability of Radial Basis Functions Discretizations for Time-Dependent Problems. *Computers and Mathematics with Applications*, **51**, 1251-1268. https://doi.org/10.1016/j.camwa.2006.04.007

[30] Sarra, S.A. (2008) A Numerical Study of the Accuracy and Stability of Symmetric and Asymmetric RBF Collocation Methods for Hyperbolic PDEs. *Numerical Methods for Partial Differential Equations*, **24**, 670-686. https://doi.org/10.1002/num.20290

[31] Fornberg, B. and Lehto, E. (2011) Stabilization of RBF-Generated Finite Difference Methods for Convective PDEs. *Journal of Computational Physics*, **230**, 2270-2285. https://doi.org/10.1016/j.jcp.2010.12.014