The Overlapped Radial Basis Function-Finite Difference (RBF-FD) Method for the Numerical Solution of PDEs

Varun Shankar

Department of Mathematics, University of Utah, UT, USA

Abstract

We present a generalization of the RBF-FD method that allows full control of the overlap between RBF-FD stencils. We accomplish this by introducing a continuous overlap parameter $\delta \in [0, 1]$ such that $\delta = 1$ recovers the standard RBF-FD method and $\delta = 0$ results in a full decoupling of the RBF-FD stencils. We show with a simple example that global interpolation with both RBFs and RBFs augmented with polynomials is superior to polynomial least squares in confining Runge oscillations to the boundary, thereby justifying retaining more RBF-FD weights from each RBF-FD stencil than just the center; this also provides greater insight into the success of the recent polyharmonic spline RBF-FD method with high-degree polynomials appended. We demonstrate the efficiency of the overlapped RBF-FD method by solving parabolic differential equations in 2D and 3D. Our results show that the new method can achieve as high as a 20x speedup over the standard RBF-FD method in the task of forming differentiation matrices. Further, the overlapped RBF-FD method allows for stencil sizes up to 6x larger than those usable in the standard RBF-FD method for a lower total computational cost.

Keywords: Radial basis function; high-order method; one-sided stencil; domain decomposition; meshfree method.

Email address: vshankar@math.utah.edu (Varun Shankar)
1. Introduction

Radial basis functions (RBFs) are popular building blocks in the development of numerical methods for PDEs. RBF interpolants can and have been used as replacements for polynomial interpolants in generating pseudospectral and finite-difference methods [1, 2, 3, 4, 5, 6, 7, 8]. Unlike polynomial-based collocation methods, RBF-based methods exhibit geometric flexibility due to their ability to handle irregular collocation node layouts, and their non-singularity in arbitrary dimensions. Further, these methods are easy to implement and achieve high orders of accuracy (algebraic or exponential) on irregular domains. RBF-based methods are also increasingly used for the solution of PDEs in situations where polynomial interpolants are non-unisolvent, like the sphere $S^2$ [9, 10, 11, 12] and other general surfaces [13, 14, 15, 16, 17].

Unfortunately, interpolation matrices formed from standard RBF basis are beset by ill-conditioning as the node spacing $h \rightarrow 0$, regardless of the smoothness of the RBF; error curves for both RBF interpolation and RBF methods for PDEs typically level off as $h$ is decreased. This phenomenon of “stagnation” or “saturation” of the errors is purely due to the use of the standard RBF basis, and not a feature of the approximation space spanned by RBFs. This fact has been exploited to develop many stable algorithms for RBFs with a shape parameter, most applicable for Gaussian RBFs [18, 19, 20, 21, 22]. Unfortunately, these stable algorithms are typically 10-100x more expensive than forming and inverting the standard RBF interpolation matrix. Fortunately, recent work by Flyer et al [23] has overcome this issue: adding high-degree polynomials (upto half the dimension of the RBF space) removes stagnation errors with little to no increase in cost. This technique has been used to solve hyperbolic and elliptic PDEs in 2D and 3D domains [24].

However, the new technique is not without its drawbacks. The primary drawback is that the degree of the appended polynomial now completely dictates the rate of convergence. Since there are only half as many “polynomial points” as RBF points, achieving high orders of accuracy now requires far more points in both 2D and 3D than if one were to simply use RBFs. For instance, consider the case of a 101-point stencil in 2D. Without appended polynomials, this would correspond roughly to a 10th order method [12]; however, when polynomials are appended, only 50 of the points are “used”. This reduces the order of convergence to 8th order. Obviously, the situation is dramatically exacerbated in 3D, with $n = 30$ giving only a second order
method when polynomials are appended. This pattern of lower order for the same stencil size presents problems beyond the order reduction; for instance, adaptive order refinement (akin to $p$-refinement in finite element methods) will now likely be prohibitively expensive. Even GPU-based parallel implementations cannot overcome the cost issues completely, since stencil sizes may now be too large to efficiently fit into shared memory.

We present a generalization of the RBF-FD method designed to overcome the computational cost involved in appending polynomials; our method also can be beneficial even if polynomials are not appended to RBF interpolants. Our new method is based on observations on the nature of error distribution in interpolation with RBFs and RBFs augmented with polynomials. The new overlapped RBF-FD method allows for a dramatic reduction in cost in computing differentiation matrices, especially in 3D problems. Further, the overlapped RBF-FD method allows the generation of higher-order differentiation matrices for \textit{lower total cost} than lower-order ones. To the best of our knowledge, this feature is unique to our new RBF-FD method when compared to all other discretization schemes. Even more interestingly, the overlapped (augmented) RBF-FD method often shows \textit{greater accuracy} than the augmented RBF-FD approach for a significantly lower cost.

The remainder of the paper is organized as follows. In the next section, we briefly review augmented RBF interpolation and the generation of RBF-FD weights from these interpolants. Section 3 contains an exploration of error distributions in global approximations with RBF interpolation, augmented RBF interpolation, and polynomial least-squares. The observations from this section are used in Section 4 to motivate the development of the overlapped RBF-FD method. Then, in Section 5, we present the results of numerical experiments comparing cost and order of accuracy on parabolic problems in 2D and 3D domains and thus demonstrate the unique cost-to-accuracy profile of the overlapped RBF-FD method. We conclude with a short summary of our results and a discussion of future work in Section 6.

2. Augmented RBF-FD

Let $\Omega \subseteq \mathbb{R}^d$, and $\phi : \Omega \times \Omega \to \mathbb{R}$ be a kernel with the property $\phi(x, y) := \phi(\|x - y\|)$ for $x, y \in \Omega$, where $\| \cdot \|$ is the standard Euclidean norm in $\mathbb{R}^d$. We refer to kernels with this property as \textit{radial kernels} or \textit{radial functions}. 

3
Given a set of nodes \( X = \{x_k\}_{k=1}^N \subset \Omega \) and a continuous target function \( f : \Omega \to \mathbb{R} \) sampled at the nodes in \( X \), we select subsets (henceforth referred to as *stencils*) \( \{P_k\}_{k=1}^N \) of the set of nodes \( X \), where each stencil \( P_k \) consists of the \( k^{th} \) node \( x_k \) and its \( n-1 \) nearest neighbors, where \( n << N \). Further, with each stencil \( P_k \), we also track an index set \( \mathcal{I}_k = \{I_{k1}, \ldots, I_{kn}\} \) that contains the *global* indices of its nodes in the set \( X \). The nearest neighbors are typically determined in a preprocessing step using a data structure such as a kd-tree. In [23], Flyer et al demonstrate that it is useful to add high-degree polynomials to the RBF interpolant, with the polynomial space possessing a dimension of roughly \( \frac{n^2}{2} \); we adopt this approach, and form *local* RBF interpolants on each stencil \( P_k \) such that

\[
s^k_f(x) = \sum_{j=I_{k1}}^{I_{kn}} c_j \phi(\|x - x_j\|) + \sum_{i=1}^{M} d_i \psi^k_i(x),
\]

so that

\[
\begin{pmatrix}
A_{P_k} & \Psi_k \\
0 & \mathbb{I}
\end{pmatrix}
\begin{pmatrix}
c \\
d^T
\end{pmatrix}
= \begin{pmatrix}
f_0 \\
0
\end{pmatrix},
\]

where \( A_{P_k} \) is the RBF interpolation matrix on \( P_k \) and \( \Psi_k = \psi^k_i(x_j) \). These local interpolants can be used to approximate functions to high-order algebraic accuracy determined by the stencil size \( n \). Such local interpolants when differentiated can be used to generate scattered-node finite difference (FD) formulas, known in the literature as RBF-FD. Note that this requires differentiating both the RBF and the polynomial bases. \( A_{P_k}^H \) is invertible if the node set \( X \) is unisolvent for the polynomials \( \psi^k \); this is a consequence of the fact that the augmented RBF interpolant exactly reproduces the appended polynomial. For more on generating RBF-FD weights from augmented RBF-FD interpolants, we refer the reader to [23].

In their remarkable work [23], Flyer et al compare interpolation with local RBFs to polynomial least squares, and conclude that augmented RBF interpolation is more accurate. Further, they demonstrate that using augmented RBFs can completely remove all stagnation errors typically seen with RBF approximations. More importantly, they note that accuracy is not improved by increasing stencil size when the RBF in question is a polyharmonic spline.
Before we present the overlapped RBF-FD method, we attempt to provide some further insights into the nature of augmented RBF interpolation. In the next section, we tackle the question of why augmented RBF interpolation can be superior to polynomial least squares. We do so by exploring the error distribution when interpolating a 2D analog of the Runge function with global augmented RBF interpolants and polynomial least squares approximants. The results from this section are used to motivate the new overlapped RBF-FD method.

3. Shrinking Runge Zones

In this section, we explore the effect of RBF interpolation, augmented RBF interpolation, and polynomial least squares on the Runge phenomenon in 2D. Consider the simple 2D Runge function given by \( f(x, y) = \frac{1}{1 + \gamma(x^2 + y^2)} \), with \((x, y) \in [-1, 1]^2\); we set \(\gamma = 1\), but our results and observations carry to higher values of \(\gamma\) as well, albeit with higher errors. While such studies have been previously performed for infinitely-smooth RBFs [25], our focus is on augmented RBF interpolation. Let \(X = (x_i, y_j)\) be a simple set of Cartesian nodes. In this setting, polynomial least squares is known to be useful in overcoming the Runge phenomenon (at the cost of sub-geometric convergence). On the other hand, global interpolation with either RBFs or polynomials at Cartesian nodes is expected to result in the Runge phenomenon. For this test, we use the septic RBF given by \(\phi(r) = r^7\). This is a somewhat arbitrary choice; the results here carry over to the other polyharmonic spline RBFs straightforwardly, with larger exponents of \(r\) leading to lower errors.

While it is generally unwise to perform global RBF interpolation in this setting, this test is important for three reasons: first, it establishes the differences between RBFs, augmented RBFs, and polynomial least-squares in a well-understood setting; second, errors in global RBF interpolation at the boundary of the domain can be viewed as a worst-case stand-in for errors at the boundaries of RBF-FD stencils on subdomains of some larger domain; and third, this test shows the differences in the distribution of errors across the domain in the three methods. We evaluate all the approximants at a set of \(10^5\) Cartesian nodes in \([-1, 1]^2\). This large number of evaluation nodes allows us to resolve fine features in the error distributions. The results of this test are shown in Figure 1.
Figure 1: Approximation by polynomial least squares, standard RBF interpolants, and RBF interpolants augmented with polynomials. The left column corresponds to errors in interpolating the Runge function with polynomial least squares, the center column to errors in RBF interpolation, and the right column to errors in augmented RBF interpolation. Darker colors indicate lower errors.
One can draw a trivial conclusion from Figure 1 going down from the top row to the bottom row, the errors decrease in order of magnitude. However, as the figures make clear, the pattern of error distribution is very different across each row. Figures 1a, 1d and 1g show that the error is relatively high at the center of the domain when using polynomial least squares, in addition to being high at the edges and four corners due to Runge oscillations. It is also easy to see darker zones free of this high error in a distinct superellipse pattern radiating from the center. Indeed, increasing the polynomial degree increases the number of such dark low-error zones.

The story is completely different for RBF interpolation. Figures 1b, 1e and 1h show that the errors for the RBF are highest only at the edges and corners of the domain. While Figure 1b shows five higher-error “lobes” (corresponding to oscillations) in the center of the domain, these lobes decrease in relative magnitude as \( n \) is increased. Further, the area occupied by the darker zones increases as \( n \) is increased. Consequently, large-amplitude oscillations are confined more and more to the edges of the domain.

Augmenting the RBFs with the same polynomial degree as in polynomial least squares (selected according to the rule in the previous section) improves this behavior even further. This can be seen in Figures 1c, 1f and 1i. Figure 1c shows the most dramatic improvement. The five higher-error lobes still exist, but are diminished in magnitude (and are therefore darker). Further, the light band around the dark region containing the lobes is now darker as well, indicating that the errors there have also been lowered by adding the polynomial to the RBF. The \( n = 64 \) case is not particularly clear, but can still be interpreted carefully: the lobes in the dark region in Figure 1f are slightly smaller in magnitude, while the outer band surrounding the lobes has lightened. Adding a high degree polynomial to the RBF appears to favorably influence the interior regions of the domain, and unfavorably influence the boundary regions. Figure 1i completes the pattern; at \( n = 100 \) with eighth-degree polynomials appending, the interior lobes vanish almost completely when compared to Figure 1h. The interior region is almost uniformly dark.

Figure 1 makes clear that RBF interpolants localize the errors more towards the boundaries of the domain than polynomial least squares, while augmented RBF interpolants achieve the same effect with lower errors in the interior. In our experiments, the degree of localization and the extent to which the errors decrease depend on the power of the polyharmonic spline. For instance, with
\( \phi(r) = r^3 \), the above patterns persist, albeit with higher errors.

The above patterns suggest very strongly that RBF interpolation (augmented or otherwise) is excellent in a large portion of the domain as the stencil size \( n \) is increased. Viewing the errors in global RBF interpolation as a worst-case stand-in for per-stencil RBF-FD errors, this seems to imply that RBF-FD weights likely possess good approximation properties even beyond the center of the stencil, especially as the stencil size \( n \) is increased. This is contrary to previous observations in the literature that RBF interpolation in the RBF-FD context should only be used towards the center of a stencil \([26, 23]\); however, it is worth noting that in the former of those studies, very small stencil sizes were used. Further, both articles do comment that appending a constant and a linear term to the cubic RBF can be very beneficial in curbing Runge oscillations. It appears that using higher-degree polynomials as suggested in \([23]\) simply continues that trend—as suggested strongly by the experiments in this article. These results also make clear that the approximation properties of augmented RBF interpolation are more akin to RBF interpolation than polynomial least-squares, despite the use of a polynomial least-squares matrix in augmented RBF interpolation, and partially explain the superiority of augmented RBF-FD to polynomial least squares. To further explore

![Figure 2: Low-error regions in augmented RBF interpolation. The figures show regions where the errors are lower than the automatically generated scale of the image by one or more orders of magnitude. These regions are indicated using black.](image)
the quality of both RBF and augmented RBF interpolation in relation to the interpolation domain, we use the same test as above, but visualize the results differently. We revert to the standard Parula colormap in Matlab with one modification: at locations where the errors are one or more orders of magnitude lower than the automatically generated scale of the graph, we color the pixels black. With a sufficient number of evaluation points, this has the effect of showing locations where the error of the RBF interpolants is especially low, enabling us to explore the question of where exactly RBF interpolation is “good”. The results of this experiment are shown in Figure 2.

Figure 2 is not as well suited to showing error transitions as Figure 1. However, the lighter colors allow us to clearly show zones of relatively lower errors in black. Figures 2a, 2b, and 2c clearly show that the lowest errors in augmented RBF interpolation are not necessarily at the center of the domain. In fact, for \( n = 36 \) in Figure 2a, the error is fairly high at the center (recall the lobes from Figure 1b). The errors in the sub-domain containing the five lobes are all roughly on the same order of magnitude, with the exception of the black curves (which are an order of magnitude or more lower in error than the lobes). In the RBF-FD context, it may thus be safe to use weights from more than just the center of the stencil! Figures 2b and 2c show that this is even more true as \( n \) is increased. There are many black curves showing much lower error than the scale of the graph and, more importantly, showing lower errors than the center of the stencil. The “safe zone” appears to grow as \( n \) increases, while the “Runge zone” shrinks.

This trend appears to hold true for RBF interpolation even without augmentation by polynomials. While augmenting with polynomials does remove saturation error [23], it may still be possible to retain RBF-FD weights in some neighborhood around the center of the domain in situations where augmented RBF interpolation is much trickier, e.g. RBF interpolation on general surfaces or other polynomial non-unisolvent scenarios. On the sphere, ongoing work has shown that augmenting RBFs with spherical harmonics has much the same effect as augmentation with polynomials does in \( \mathbb{R}^d \).
Figure 3: Illustration of the effect of the overlap parameter $\delta$ in the overlapped RBF-FD method. The point cloud on the left shows two stencils. In the standard RBF-FD method ($\delta = 1$), RBF-FD weights are only computed at the center of each stencil (circle and cross) and discarded elsewhere. In the overlapped RBF-FD method, depending on $\delta$, the RBF-FD weights for both the circle and cross may be computed by the same thick-line stencil, and the dashed-line stencil may not be used (see right).
4. Overlapped RBF-FD

Armed with the evidence of the previous section that RBF interpolation is valid in large zones outside the center of the domain, we extrapolate this concept to RBF-FD, and generate the following hypothesis: *RBF-FD weights may be of reasonable accuracy well outside the center of each RBF-FD stencil.*

Leaving the testing of this hypothesis to the results section, we can now carefully formulate a generalization of the RBF-FD method which we called the *overlapped RBF-FD method.* The terminology “overlapped” refers to the extent to which stencils are overlapped. In the standard RBF-FD method, neighboring stencils are considered to be *fully overlapped* in that all weights other than those at the center of these stencils are discarded; see Figure 3. Let \( w(P_k) \) be the width of the stencil \( P_k \); in this article, we define the width as the distance from the point at the logical center of the stencil \( P_k \) to its furthest neighbor in that stencil. This effectively defined a *ball* centered at each node \( x_k \) whose radius is \( w(P_k) \), where \( P_k \) is the stencil associated with \( x_k \).

We now define an *overlap parameter* \( \delta \) such that the limit \( \delta \rightarrow 1 \) gives the standard RBF-FD method (full overlap) with weights computed only at the centers of stencils; the limit \( \delta \rightarrow 0 \) forces weights to be computed and retained at *all* points in a stencil. Now, for each stencil \( P_k \), define a quantity \( r^m \) such that

\[
    r^m = (1 - \delta)w(P_k). \tag{3}
\]

\( r^m \) can be thought of as the “retention distance”– the distance from the center of the stencil up to which RBF-FD weights must be calculated and retained. In this work, we assume that \( \delta \in [0, 1] \) is a single real number across all stencils, though an obvious generalization would be to allow \( \delta \) to vary from stencil to stencil. We now define a sequential algorithm for the overlapped RBF-FD method.

1. Maintain a global list of \( N \) nodes.
2. For the first node in the list \( x_1 \), compute \( r_1^m \) based on \( \delta \) and the \( w(P_1) \) for its stencil.
3. Let \( I_1^1 \ldots I_p^1 \) be the *global* indices of the nodes in \( P_1 \) that satisfy \( \| x_1 - x_{I_j} \| \leq r_j^m \), \( j = 1, \ldots, p \).
4. Compute the RBF-FD weights for $x_{I_j^1}$, $j = 1, \ldots, p$.

5. Delete the nodes with the global indices $I_j^1$, $j = 1, \ldots, p$ from the global list.

6. Move to the next node in the global list and repeat the above steps.

The immediate consequence of this approach is that the order in which the nodes are traversed determines the RBF-FD weights assigned to a node and its stencil constituents. While this may seem a cause for concern, our experiments indicate that this is not at all detrimental to the solution of PDEs. We also note that there are some trivial ways to bypass the issue: for instance, one could use a partition of unity function to combine the different RBF-FD weights obtained for a node that belongs to multiple stencils. However, for the sake of simplicity, we do not explore this approach here.

Another consequence of this approach is that the length of the global list of nodes shortens by more than one as we proceed from one entry in the list to the next; specifically, the list only contains $N^e$ elements, where $N_\delta < N$. If $\delta$ is sufficiently small, it is possible that $N = 40N_\delta$. In the results section, we explore the computational cost versus accuracy characteristics for different choices of $\delta$. If the interpolation nodes never change, the overlapped RBF-FD method results in faster computations of discrete differential operators, despite the increased number of derivative evaluations and weight calculations per stencil. In scenarios where PDEs are solved on time-varying domains or where the nodes in a domain are adaptively modified to track some solution feature, we anticipate the overlapped RBF-FD method as being extremely useful in rapidly and accurately recomputing differentiation matrices.

A third consequence of this approach is that by increasing $\delta$ as $n$ is increased, one could potentially fix the cost of RBF-FD despite the increase in accuracy. In fact, in the results section, we demonstrate that it is possible to use larger stencil sizes for a lower total computational cost than for the smaller stencil sizes with somewhat straightforward choices of $\delta$. We note that this paves the way for adaptive node refinement and adaptive order refinement with little to no increase in cost. This is to be explored in future work.

Finally, before proceeding, we note that while $\delta$ is a continuous parameter in $[0, 1]$, the node spacing of a given node set will effectively enforce a discrete constraint on how many nodes are chosen from a given stencil to be within the retention distance $r^m$. We do not analyze this aspect of the overlapped
RBF-FD method in this work.

5. Results

We focus all our experiments on a single test problem: solution of the forced diffusion equation. The forcing term is selected to maintain a prescribed solution for all time, and the prescribed solution is used to test spatial convergence rates. We solve this test problem on the closed unit disk in $\mathbb{R}^2$ and the closed unit ball in $\mathbb{R}^3$. These domains were chosen due to the presence of curved boundaries, usually a source of difficulty for high-order finite difference methods.

5.1. Forced Diffusion on the disk

Figure 4: Quasi-uniform nodes on the unit disk. The figure shows the nodes used for the RBF-FD discretization on the interior (solid) and boundary (circles) of the unit disk.

We solve the forced heat equation on the closed disk:

$$
\Omega(x) = \{x = (x, y) : ||x||_2 \leq 1\}.
$$

(4)
This equation is given as

\[ \frac{\partial c}{\partial t} = \nu \Delta c + f, \]  

(5)

where \( c(x, t) = c(x, y, t) \) is some scalar concentration or density, \( \nu \) is the coefficient of diffusion, and \( f \) is a forcing term. For the following convergence tests, our prescribed exact solution is

\[ c(x, y, t) = 1 + \sin(\pi x) \cos(\pi y) e^{-\pi t}. \]  

(6)

The forcing term that makes this solution hold is given by

\[ f(x, y, t) = \pi (2\pi \nu - 1) \sin(\pi x) \cos(\pi y) e^{-\pi t}. \]  

(7)

The heat equation is subject to a time-dependent Dirichlet boundary condition, given by evaluating \( c(x, y, t) \) on the boundary \( \|x\|_2 = 1 \). We use a quasi-uniform node distribution obtained from the Distmesh program [27] as our node set on the disk (see Figure 4). The resulting nodes have a spacing of approximately \( h \propto \frac{1}{\sqrt{N}} \).

After discretizing the Laplacian with the RBF-FD method, we time-step the resulting set of ODEs with the BDF4 scheme [28] with the forcing term evaluated implicitly; this is a typical method of lines approach. Further, to invert the time-stepping matrix, we use the GMRES iterative method preconditioned with an incomplete LU factorization with zero fill-in, \( i.e., \) ILU(0); at each time level, the GMRES method is fed the solution from the previous step as a starting guess. With this combination of choices, we have observed that the GMRES scheme typically converges in 1-2 iterations to a relative residual of \( O(10^{-14}) \) with no restarts required. For the following convergence tests, we set \( \nu = 0.1 \) and the final time to \( T = 0.2 \). The time-step is set to \( \Delta t = 10^{-3} \) to ensure that the errors are purely spatial. The results of this experiment are shown in Figure 5.

Figure 5 shows different orders of convergence for the RBF-FD method based on the \( n \) values used. As the stencil size \( n \) is increased, the degree of the appended polynomial increases as well. This increases the order of convergence of the method, giving us third, fourth, sixth, seventh and eighth order methods in both the \( \ell_2 \) and \( \ell_\infty \) norms. Interestingly, despite differentiating the interpolants twice, we appear to retain the order of convergence corresponding to the degree of the polynomial, despite the short final time \( T \) and the
Figure 5: Convergence rates of the standard augmented RBF-FD method ($\delta = 1$) for the forced diffusion equation on the disk. The figures show the $\ell_2$ (a) and $\ell_\infty$ (b) errors in the numerical solution to the forced diffusion equation on the disk with time-dependent Dirichlet boundary conditions. The dashed lines are lines of best fit to the data, showing the approximate rates of convergence for different values of $n$. The polynomial degree increases as $n$ is increased and is mainly responsible for the convergence rates seen in the figure.
small diffusion coefficient $\nu$ being used. Of equal interest, it is clear that appending polynomials completely removes the saturation or stagnation error seen in RBF interpolation; the error curves go down all the way to $O(10^{-12})$ as $N$ and $n$ are increased.

5.1.1. Cost versus Accuracy as a function of $\delta$

We now test the effect of $\delta$ on the accuracy of the solution. To highlight the power of the overlapped RBF-FD method, we plot the accuracy as a function of the computational cost in computing the discrete Laplacian operator for different $\delta$ values. The results are shown in Figure 6. Analyzing the subplots in Figure 6, it becomes clear that using smaller $\delta$ values becomes more and more beneficial as $n$ increases. For $n = 30$ (Figure 6a), it is clear that using $\delta < 0.7$ results in a loss in accuracy with no great decrease in computational cost. $\delta = 0.7$ results in a lower error for a slightly lower computational cost, possibly because the method selects weights from regions of higher accuracy like those seen in Figure 2.

On the other hand, for $n = 50$, all values of $\delta < 1$ are of significantly lower cost than $\delta = 1$. Further, barring the value of $\delta = 0.2$, all the tested $\delta$ values appear to result in errors of comparable or lower magnitude than the $\delta = 1$ case. The greatest accuracy is obtained for $\delta = 0.7$, with an almost 4x maximum speedup over the $\delta = 1$ case. The greatest speedup is obtained for $\delta = 0.3$ with a comparable error to $\delta = 1$; in this case, the maximum speedup is approximately 8x for a similar (slightly lower) accuracy. For $n = 70$, the lowest error is for $\delta = 1$, but all values of $\delta \in (0.2, 1)$ show errors of approximately the same order of magnitude. The best cost–accuracy balance appears to be for $\delta = 0.4$, with a maximum speedup of approximately 10.3x. Finally, for $n = 101$, $\delta = 0.4$ once again offers the best cost–accuracy balance; for $\delta = 0.4$, we obtain a similar accuracy to $\delta = 1$ with a maximum speedup of approximately 16x. It is safe to conclude that using small values of $\delta$ is highly beneficial as $n$ is increased. Further, using $\delta = 0.7$ appears safe even for $n = 30$, giving a slight speedup.

5.1.2. Higher order for a lower total cost

We now explore how to keep the cost fixed as the stencil size $n$ is increased. In other words, we wish to determine how to increase the order of accuracy of
Figure 6: Cost versus accuracy as a function of $\delta$ on a semilog scale. The figures show the $\ell_2$ errors as a function of computational cost in computing the Laplacian for different values of $\delta$, the overlap parameter. Each subfigure corresponds to a specific $n$ value. The computational cost scales as $O(n^3 N_\delta)$, where $N_\delta$ is the effective number of global nodes used in the computation.
Figure 7: Higher orders at lower cost. The figures show the $\ell_2$ errors as a function of computational cost in computing the Laplacian for different values of $\delta$, $n$, and $N$. The results are shown for the standard RBF-FD method ($\delta = 1$) (top left); the overlapped RBF-FD method with decreasing $\delta$ (top right); and the overlapped RBF-FD method with optimal delta values (bottom).
the RBF-FD method without an increase in computational cost. To explore this possibility, we run two types of tests. First, we allow $n$ for a range of $N$ values to increase while gradually decreasing $\delta$, with the hope that the decreasing overlap will reduce the cost associated with forming higher-order RBF-FD operators. For the second test, we again allow $n$ to increase, but this time pick the “optimal” values of $\delta$, with “optimal” defined as the values of $\delta$ that result in lowest computational cost for a given order of magnitude of error. This gives us the following $(n, \delta)$ pairs: $(30, 0.7)$, $(50, 0.3)$, $(70, 0.4)$ and $(101, 0.4)$. With these values, we now compare the overlapped RBF-FD method for different values of $n$ in terms of accuracy versus computational cost. The results of this experiment are shown in Figure 7.

Figure 7a shows the reference results for the standard RBF-FD method (corresponding to $\delta = 1$). Clearly, increasing $n$ results in higher-order methods for higher total cost, but a greater efficiency measured as cost for a given accuracy; the orders correspond to those seen in the earlier subsections and are determined by the degree of the appended polynomial. Figure 7b shows some surprising results: by gradually decreasing $\delta$ as $n$ is increased, we can in fact obtain higher-order methods at a much lower overall total cost, even without factoring in the higher accuracy! This shows the power of the overlapped RBF-FD method very clearly. Finally, Figure 7c shows that if we pick the optimum $\delta$ values that balance cost and accuracy for each $n$ value, we obtain similar results to those seen in Figure 7a with the exception that the higher-order methods are still of a lower total cost than the lowest order one corresponding to $n = 30$; indeed, while the cost does increase as $n$ increases, a high-order overlapped RBF-FD method will almost always be cheaper than the lower-order ones, since the highest order one here is already approaching machine precision.

It is also worth noting that the optimum $\delta$ values deliver not only lower cost, but similar or higher accuracy to $\delta = 1$.

5.2. Forced Diffusion in the unit ball

In this test, we solve the forced heat equation in the closed unit ball in $\mathbb{R}^3$:

$$\Omega = \{ x = (x, y, z) : \|x\|_2 \leq 1 \}.$$  (8)
Figure 8: Quasi-uniform nodes in the unit ball. The figure shows the nodes used for the RBF-FD discretization in the interior (solid) and boundary (circles) of the unit ball.

Once again, we use the method of manufactured solutions. Our prescribed solution is

\[ c(x, y, z, t) = 1 + \sin(\pi x) \cos(\pi y) \sin(\pi z)e^{-\pi t}, \]

and the corresponding forcing term is

\[ f(x, y, z, t) = \pi(3\pi \nu - 1) \sin(\pi x) \cos(\pi y) \sin(\pi z)e^{-\pi t}. \]

The time-dependent boundary condition is obtained by evaluation \( c(x, t) = c(x, y, z, t) \) on the boundary of the unit ball, \( i.e. \), the sphere \( \|x\|_2 = 1 \). We obtained quasi-uniform node distributions in the unit ball by using the beautiful interactive meshing program Gmsh [29] to generate a mesh, and then retaining the resulting mesh node distribution. The resulting nodes have a spacing of approximately \( h \propto \frac{1}{\sqrt{N}} \), and are shown in Figure 8. To the best of our knowledge, this is the first application of RBF-FD to a 3D parabolic problem.

As in the 2D test problem, we employ the method of lines approach: we first discretize the Laplacian with RBF-FD; we then discretize the resulting
system of ODEs using the BDF4 scheme, evaluating both the Laplacian and forcing term implicitly in time. To invert the time-stepping matrix, we now use the BiCGStab method; this is to avoid the potential memory requirements of the GMRES method in 3D. We precondition the BiCGStab solver with the ILU(0) factorization of the time-stepping matrix. Further, for each iteration, we feed the solution at the previous time level as a guess. Much like in the case of GMRES, we find that BiCGStab converges in most cases in 1 or 2 iterations per time-step to a relative residual of $O(10^{-14})$. Once again, we set $\nu = 0.1$ and the final time to $T = 0.2$. The time-step is once again set to $\Delta t = 10^{-3}$. The results are shown in Figure 9.

![Figure 9: Spatial Convergence of Numerical Solution](image)

(a) $\ell_2$ errors
(b) $\ell_\infty$ errors

Figure 9: Convergence rates of the standard augmented RBF-FD method ($\delta = 1$) for the forced diffusion equation in the ball. The figures show the $\ell_2$ (a) and $\ell_\infty$ (b) errors in the numerical solution to the forced diffusion equation in the ball with time-dependent Dirichlet boundary conditions. The dashed lines are lines of best fit to the data, showing the approximate rates of convergence for different values of $n$. The polynomial degree increases as $n$ is increased and is mainly responsible for the convergence rates seen in the figure.

Figure 9 again shows different orders of convergence based on $n$. Unfortu-
nately, in a full 3D problem, the stencil sizes are much larger than those seen in a 2D problem for the same order. This is due to the fact that we augment the local RBFs with polynomials: if the polynomial degree is to be equal to about half the dimension of the space defined by \( n \), \( n \) must be very large in 3D to support high-degree polynomials. Regardless, we see that it is possible to obtain second, fourth, and sixth order methods in both the \( \ell_2 \) and \( \ell_\infty \) norms. Note that the \( n \) values used here are somewhat arbitrary. One could certainly find the smallest \( n \) value for a given order. However, our goal here is to simply test the overlapped RBF-FD method on very large values of \( n \) and \( N \). In this case, the largest values correspond to \( n = 301 \) and \( N = 275,544 \). The lowest errors are \( O(10^{-10}) \) in the \( \ell_2 \) norm.

5.2.1. Cost versus Accuracy as a function of \( \delta \)

As in 2D, we test the effect of \( \delta \) on the accuracy of the solution by plotting the accuracy as a function of the computational cost in computing the discrete Laplacian operator for different \( \delta \) values. The results are shown in Figure 10. The improvement for small \( \delta \) values is even higher than in the 2D case. For instance, in the 3D problem, it is now genuinely beneficial to use a small \( \delta \) value even for \( n = 50 \), a second order method. All values of \( \delta \in [0.4, 1) \) appear to be useful in both decreasing cost and increasing accuracy (Figure 10a). If one wants the best possible accuracy, \( \delta = 0.5 \) is the clear winner; on the other hand, the best balance between computational cost and accuracy is found at \( \delta = 0.4 \). The former gives a 2.3x speedup, while the latter gives a 3.41x speedup. \( \delta = 0.3 \) was found to be unstable, likely because it reaches into the Runge zone associated with each stencil.

The speedup improves even further as \( n \) is increased in 3D. When \( n = 150 \), it is clear from Figure 10b that all tested values of \( \delta < 1 \) result in greater accuracy for lower computational cost. The best accuracy is obtained for \( \delta = 0.6 \), though \( \delta = 0.5 \) gives a similar accuracy for a lower cost. However, the best balance between cost and accuracy is found at \( \delta = 0.3 \). The error is half that for \( \delta = 1 \), but the speedup obtained with \( \delta = 0.3 \) is 9.2x. Finally, Figure 10c shows the best results of all. All tested values of \( \delta < 1 \) appear to be more accurate for a significantly lowered computational cost. The lowest error is found for \( \delta = 0.7 \), with the error being an order of magnitude lower than \( \delta = 1 \) for a 4.4x speedup. However, the lowest cost is obtained for \( \delta = 0.3 \). Here, the error is about half that for \( \delta = 1 \), but the speedup
is approximately 20x, despite the complete lack of parallelization. Clearly, the overlapped RBF-FD method performs better as the number of spatial dimensions is increased from two to three, helping partially ameliorate the cost associated with sampling higher dimensional spaces.

5.2.2. Higher order for a lower total cost

As in 2D, we again explore the relationship between cost and stencil size for different $\delta$ values. We do this again with two types of tests: increasing $n$ while gradually decreasing $\delta$; and increasing $n$ while using the optimal $\delta$ values, with optimal again implying the best balance between cost and accuracy—these are the $\delta$ values reported in the previous subsection. The results of this experiment are shown in Figure 11. Figure 11a shows the reference results for the standard RBF-FD method ($\delta = 1$). As in 2D, increasing $n$ increases total cost, but improves efficiency (accuracy for a given cost). Figure 11b again shows that the 2D trends hold: decreasing $\delta$ as $n$ is increased actually allows higher-order methods for a lower total cost, and therefore a vastly improved efficiency. Finally, in Figure 11c, we see that using the optimal $\delta$ values dramatically reduces cost for all the $n$ values, with $n = 150$ and $n = 301$ still resulting in lower total cost than $n = 50$; in the latter case, this is a 6x increase in stencil size for a small decrease in cost. Unlike in the 2D case, this trend will likely not continue as $n$ increases if we continue to pick optimal $\delta$ values, due to the fact that even low-order methods are efficient with overlapped RBF-FD in 3D. Regardless, all the optimal $\delta$ values deliver both lower cost and higher accuracy than $\delta = 1$ in 3D.

It is important to note an almost consistent trend in both the 2D and 3D results: decreasing $\delta$ actually appears to improve the accuracy, sometimes by an order of magnitude. A possible explanation for this is that the center of an RBF-FD stencil is not the best place to pick RBF-FD weights from. Indeed, this hypothesis is partially supported by the black contours seen in Figure 2. In fact, if one could determine these contours with some testing, it may be possible to further generalize the overlapped RBF-FD method by picking weights only along contour lines rather than in a ball centered on the center of the stencil. We leave this line of research for future work.
6. Summary and Future Work

The overlapped RBF-FD method helps reduce the cost of the standard RBF-FD method when using large stencil sizes. It paves the way for fixed-cost adaptive order refinement algorithms due to its unique feature of generating high-order methods at a lower total cost than low-order methods. Interestingly, due to the nature of error distributions in augmented RBF interpolation, the overlapped RBF-FD method is sometimes more accurate than the standard RBF-FD method, especially at larger stencil sizes.

A minor issue with the method is its non-uniqueness. The RBF-FD weight for a node is currently determined solely by the order of traversal of the node set. A natural extension would be to find a way to select between different possible weights assigned to a node, or to combine them in some systematic fashion (possibly with the RBF Partition of Unity approach). Also, while the current work only tests the method on parabolic problems, preliminary tests indicate that it is quite effective on hyperbolic problems as well. This needs to be explored in greater detail.

The current article only focuses on a serial implementation of the overlapped RBF-FD method. In practical applications, GPU implementations will be necessary. A GPU implementation of the method is currently being developed and compared against a GPU implementation of the standard RBF-FD method. We expect the speedups of the overlapped RBF-FD method to “stack” with the speedup offered by a GPU over a serial implementation.

A natural application of the overlapped RBF-FD method would be the solution of PDEs on time-varying domains and surfaces, where the cost of computing differential operators can no longer be considered a preprocessing step. We also plan to explore the use of this method for adaptive node refinement and adaptive order refinement, akin to $h-p$ refinement in finite elements.

Acknowledgments

This work was supported by NSF grants DMS-1521748 and DMS-1160432. The author also acknowledges useful discussions with Professors Grady Wright and Edward Fuselier.
References

[1] C. Shu, H. Ding, K. Yeo, 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 (7) (2003) 941–954.

[2] V. Bayona, M. Moscoso, M. Carretero, M. Kindelan, Rbf-fd formulas and convergence properties, Journal of Computational Physics 229 (22) (2010) 8281–8295.

[3] O. Davydov, D. Oanh, Adaptive meshless centres and rbf stencils for poisson equation, Journal of Computational Physics 230 (2) (2011) 287–304.

[4] A. Tolstykh, D. Shirobokov, On using radial basis functions in a finite difference mode with applications to elasticity problems, Computational Mechanics 33 (1) (2003) 68–79.

[5] T. Cecil, J. Qian, S. Osher, Numerical methods for high dimensional Hamilton-Jacobi equations using radial basis functions, J. Comput. Phys. 196 (2004) 327–347.

[6] G. B. Wright, B. Fornberg, Scattered node compact finite difference-type formulas generated from radial basis functions, Journal of Computational Physics 212 (1) (2006) 99 – 123. doi:10.1016/j.jcp.2005.05.030 URL http://www.sciencedirect.com/science/article/pii/S0021999105003116

[7] G. Chandhini, Y. Sanyasiraju, Local RBF-FD solutions for steady convection–diffusion problems, International Journal for Numerical Methods in Engineering 72 (3) (2007) 352–378.

[8] D. Stevens, H. Power, M. Lees, H. Morvan, The use of PDE centers in the local RBF Hermitean method for 3D convective-diffusion problems, J. Comput. Phys. 228 (2009) 4606–4624.
[9] N. Flyer, G. B. Wright, Transport schemes on a sphere using radial basis functions, J. Comp. Phys. 226 (2007) 1059–1084.

[10] N. Flyer, G. B. Wright, A radial basis function method for the shallow water equations on a sphere, Proc. Roy. Soc. A 465 (2009) 1949–1976.

[11] B. Fornberg, E. Lehto, Stabilization of RBF-generated finite difference methods for convective PDEs, Journal of Computational Physics 230 (2011) 2270–2285.

[12] N. Flyer, E. Lehto, S. Blaise, G. Wright, A. St-Cyr, A guide to RBF-generated finite differences for nonlinear transport: shallow water simulations on a sphere, Journal of Computational Physics 231 (2012) 4078–4095.

[13] C. Piret, The orthogonal gradients method: A radial basis functions method for solving partial differential equations on arbitrary surfaces, J. Comput. Phys. 231 (20) (2012) 4662–4675.

[14] C. Piret, The fast radial basis functions orthogonal gradients method for solving pdes on arbitrary surfaces, to be submitted.

[15] E. J. Fuselier, G. B. Wright, A high-order kernel method for diffusion and reaction-diffusion equations on surfaces, Journal of Scientific Computing (2013) 1–31 doi:10.1007/s10915-013-9688-x

URL http://dx.doi.org/10.1007/s10915-013-9688-x

[16] V. Shankar, G. B. Wright, A. L. Fogelson, R. M. Kirby, A radial basis function (RBF) finite difference method for the simulation of reaction-diffusion equations on stationary platelets within the augmented forcing method, International Journal for Numerical Methods in Fluids 75 (1) (2014) 1–22. doi:10.1002/fld.3880.

URL http://dx.doi.org/10.1002/fld.3880

[17] V. Shankar, G. B. Wright, R. M. Kirby, A. L. Fogelson, A radial basis function (RBF)-finite difference (FD) method for diffusion and reaction-diffusion equations on surfaces, J. Sci. Comput. In Press. doi:10.1007/s10915-014-9914-1
[18] B. Fornberg, G. Wright, Stable computation of multiquadric interpolants for all values of the shape parameter, Comput. Math. Appl. 48 (2004) 853–867.

[19] B. Fornberg, C. Piret, A stable algorithm for flat radial basis functions on a sphere, SIAM Journal on Scientific Computing 30 (2007) 60–80.

[20] G. E. Fasshauer, M. J. McCourt, Stable evaluation of Gaussian radial basis function interpolants, SIAM Journal on Scientific Computing 34 (2012) A737–A762.

[21] B. Fornberg, E. Larsson, N. Flyer, Stable computations with Gaussian radial basis functions, SIAM Journal on Scientific Computing 33(2) (2011) 869–892.

[22] B. Fornberg, E. Lehto, C. Powell, Stable calculation of Gaussian-based RBF-FD stencils, Comp. Math. Appl. 65 (2013) 627–637.

[23] N. Flyer, B. Fornberg, V. Bayona, G. A. Barnett, On the role of polynomials in rbf-fd approximations: 1. interpolation and accuracy, Journal of Computational Physics 321 (2016) 21 – 38. doi:http://dx.doi.org/10.1016/j.jcp.2016.05.026 URL http://www.sciencedirect.com/science/article/pii/S0021999116301632

[24] N. Flyer, G. A. Barnett, L. J. Wicker, Enhancing finite differences with radial basis functions: Experiments on the navierstokes equations, Journal of Computational Physics 316 (2016) 39 – 62. doi:http://dx.doi.org/10.1016/j.jcp.2016.02.078 URL http://www.sciencedirect.com/science/article/pii/S0021999116300195

[25] B. Fornberg, J. Zuev, The Runge phenomenon and spatially variable shape parameters in RBF interpolation, Comput. Math. Appl. 54 (2007) 379–398.

[26] B. Fornberg, T. Driscoll, G. Wright, R. Charles, Observations on the behavior of radial basis function approximations near boundaries, Computers & Mathematics with Applications 43 (3) (2002) 473 – 490. doi:http://dx.doi.org/10.1016/S0898-1221(01)00299-1.
[27] P. O. Persson, Distmesh: a simple Matlab mesher (2005).

[28] U. M. Ascher, S. J. Ruuth, B. T. R. Wetton, Implicit-explicit methods for time-dependent pdes, SIAM J. Numer. Anal 32 (1997) 797–823.

[29] C. Geuzaine, J.-F. Remacle, Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering 79 (11) (2009) 1309–1331. doi: \url{10.1002/nme.2579}

URL \url{http://dx.doi.org/10.1002/nme.2579}
Figure 10: Cost versus accuracy as a function of $\delta$ on a semilog scale. The figures show the $\ell_2$ errors as a function of computational cost in computing the Laplacian for different values of $\delta$, the overlap parameter. Each subfigure corresponds to a specific $n$ value. The computational cost scales as $O(n^3 N_\delta)$, where $N_\delta$ is the effective number of global nodes used in the computation.
Figure 11: Higher orders at lower cost. The figures show the $\ell_2$ errors as a function of computational cost in computing the Laplacian for different values of $\delta$, $n$ and $N$. The results are shown for the standard RBF-FD method ($\delta = 1$) (top left), the overlapped RBF-FD method with decreasing $\delta$ (top right), and the overlapped RBF-FD method with optimal delta values (bottom).