# ADAPTIVITY IN LOCAL KERNEL BASED METHODS FOR APPROXIMATING THE ACTION OF LINEAR OPERATORS*

JONAH A. REEGER†

**Abstract.** Building on the successes of local kernel methods for approximating the solutions to partial differential equations (PDE) and the evaluation of definite integrals (quadrature/cubature), a local estimate of the error in such approximations is developed. This estimate is useful for determining locations in the solution domain where increased node density (equivalently, reduction in the spacing between nodes) can decrease the error in the solution. An adaptive procedure for adding nodes to the domain for both the approximation of derivatives and the approximate evaluation of definite integrals is described. This method efficiently computes the error estimate at a set of prescribed points and adds new nodes for approximation where the error is too large. Computational experiments demonstrate close agreement between the error estimate and actual absolute error in the approximation. Such methods are necessary or desirable when approximating solutions to PDE (or in the case of quadrature/cubature), where the initial data and subsequent solution (or integrand) exhibit localized features that require significant refinement to resolve and where uniform increases in the density of nodes across the entire computational domain is not possible or too burdensome.

**Key words.** Kernel Methods, Radial Basis Function, Adaptivity

**MSC codes.** 68Q25, 65R99

## 1. Introduction.

This article concerns the development of an efficiently computable error estimate, and some basic implementations of methods based on the estimate, for adaptively approximating the action of linear operators on a set of sufficiently smooth functions. Kernel methods are employed, whereby a basis that includes shifts of a chosen conditionally-positive definite function–the kernel–and supplemental polynomial terms is used in the approximation. The ideas behind kernel methods allow for efficient function approximation with high orders of accuracy even in the presence of variable node spacing [10]. Such methods are also easily generalized to high dimensions and adaptable to a wide variety of domains.

The work here expands on the successes of kernel methods in interpolation, the approximation of solutions to PDEs and the approximate evaluation of definite integrals [11, 13, 16, 20, 27, 28, 29, 30, 31, 35] to include "$h$-adaptivity"–increasing or decreasing local node density to improve accuracy or efficiency. Adaptivity is necessary or desirable when an integrand, function to be differentiated, or initial data and subsequent solution to a PDE exhibits localized features that require increased node density to resolve, and where uniform refinement across the entire domain is not possible or too burdensome.

Kernel approximations are advantageous because in the settings of interpolation and differentiation a point cloud is all that is necessary to describe the geometry of the domain; that is, the approximations are truly meshless. This is in contrast to other classical methods that require a mesh of the domain, often with restrictions on the quality of the mesh. The absence of requirements to have a mesh of a certain quality opens up opportunities for new, more flexible node refinement strategies–a benefit for $h$-adaptivity. A recent work considered adaptive numerical differentiation

*Submitted to the editors DATE.

**Funding:** This work was funded by the Joint Directed Energy Transition Office project Modeling and Simulation of Laser Propagation in Reactive Media and Air Force Office of Scientific Research project Kernel Methods with Machine Learning and Adaptivity.

†Department of Mathematics and Statistics, Air Force Institute of Technologoy, Wright-Patterson Air Force Base, OH (jonah.reeger@afit.edu).

with a basis including only multivariate polynomial terms, but was fixed to discrete Leja points when constructing the interpolant [6].

Many investigations of adaptive kernel methods focus mainly on selection of the shape parameter in positive-definite kernels, which balances accuracy and numerical stability, or mesh refinement based on global Radial Basis Function (RBF) approximations (see, e.g., [12, 14, 15, 33, 38, 40]). Nearly all of these focus strictly on the problem of interpolation. In contrast, the estimator and methods developed in this work consider local approximations in an effort to reduce overall computational complexity and promote numerical stability. Further, the goal here is to achieve an accurate approximation of the action of a linear operator, which can introduce difficulties beyond those imposed by interpolation alone. The growing body of work considering local $h$-adaptive RBF generated finite differences (RBF-FD) like methods for solving PDEs shares similar goals with this work. However, the indicators or estimators used to provide information on locations in the solution domain that would benefit from a refinement of the discretization differ from what is developed herein. Some existing indicators rely on the size of a computed quantity or the detection of an interface [7, 21], or the indicator depends on differences of computed quantities or low order approximations of these quantities at nearby spatial locations as a measure of local change in a solution [5, 25, 26]. Still, others consider the local residual (how well the current solution satisfies the governing equations) [23, 36] or the difference between solutions computed in fundamentally different ways (e.g., implicit versus explicit computation) [19]. A more generic discussion of error indicators can be found in [34]. This article instead develops an error indicator similar to the one used in, for instance, the adaptive trapezoidal rule (see, e.g., [1]) that is based on the difference of approximations that achieve different orders of accuracy.

Development of error estimates and construction of basic numerical methods that use these estimates is made simpler in this work by considering, for now, only the direct application of linear operators to a known function. For instance, computational demonstrations are provided for the differentiation of and approximate integration of known functions in one and two dimensions. On the other hand, the solution of PDEs requires approximation of derivatives of an unknown function and subsequent inversion of a matrix containing the weights that implement the approximate action of the desired linear operation. These differences alter and complicate the analysis of the error estimates and require changes to algorithms that utilize the estimates. Initial discussions on how to extend this work to PDEs are provided in remarks 3.6 and 4.1.

The following section 2 formulates the problem of local kernel approximations and introduces the interpolants that are used in approximation. Then, section 3 describes the error in approximation using kernel interpolants and an estimate of that error. This is followed by an introduction to an adaptive algorithm that successfully uses the error estimate to achieve a prescribed tolerance in the local approximation of linear operators in section 4. Finally, sections 5 and 6 provide computational experiments and conclusions, respectively.

**2. Local Kernel Approximations Via Interpolation.** Consider the evaluation of $\mathcal{L}f$, where $\mathcal{L}$ is a linear operator and $f : \mathbb{R}^d \mapsto \mathbb{R}$. The domain of $\mathcal{L}$ is $C^\infty(\Omega)$, where $\Omega \subset \mathbb{R}^d$, and the codomain of $\mathcal{L}$ could be, for example, $C^\infty(\Omega)$ or simply $\mathbb{R}$. Suppose that $\mathcal{S} = \{\mathbf{x}_i\}_{i=1}^N$ is a set of $N$ unique points in $\mathbb{R}^d$. Further define a set of points $\mathcal{X}_0 = \{\mathbf{x}_{k,0}\}_{k=1}^K$ and associate to each of these points a set $\mathcal{N}_{k,n} = \{\mathbf{x}_{k,j}\}_{j=1}^n$ of the $n$ points in $\mathcal{S}$ nearest to $\mathbf{x}_{k,0}$. The value of $n$ need not be the same for each $k$. For

instance, the computational experiments in [30] suggest that it would be worthwhile to increase the value of $n$ near domain boundaries to mitigate errors reminiscent of those introduced by the Runge phenomenon. However, for simplicity, here the value of $n$ will be the same for all $k$ with no apparent adverse effects in the computational experiments.

An important property of $\mathcal{N}_{k,n}$ is

$$h_{k,n} = \max_{j=1,2,\ldots,n} \|\mathbf{x}_{k,j} - \mathbf{x}_{k,0}\|_2,$$

which provides a sense of the spacing between points in the set. Estimates of the error in the approximate local action of $\mathcal{L}$ will be presented in terms of this distance. The points in $\mathcal{N}_{k,n}$ do not necessarily need to be those nearest to $\mathbf{x}_{k,0}$; however, other choices may increase the value of $h_{k,n}$, impacting the accuracy of the approximate action of $\mathcal{L}$. The choice of the points to include in $\mathcal{X}_0$ is dependent on the action of $\mathcal{L}$. For instance, in the case of $\mathcal{L}$ being a derivative it is convenient to set $\mathcal{X}_0 = \mathcal{S}$. On the other hand, if $\mathcal{L}$ is a definite integral, then it is useful to construct on the set $\mathcal{S}$ a tessellation $T = \{t_k\}_{k=1}^K$ (via Delaunay tessellation or some other algorithm) of $K$ simplices and to let $\mathcal{X}_0$ be the set of the midpoints of the simplices (for each simplex, the midpoint is the average of its vertices, i.e., its barycenter). For $d = 1$ these simplices are intervals of the real line, and for $d = 2$ and $d = 3$ these are triangles and tetrahedra, respectively.

Approximation of the action of $\mathcal{L}$ on $f$ is accomplished in a manner similar to the concept of RBF-FD where $f(\mathbf{x})$ is first approximated locally by an interpolant, and then $\mathcal{L}$ is applied to the interpolant. For instance, when considering the approximation of a definite integral over a domain $\Omega$, it convenient to let

$$\mathcal{L}f = \int_\Omega f(\mathbf{x})d\mathbf{x} = \sum_{k=1}^K \int_{\omega_k} f(\mathbf{x})d\mathbf{x} = \sum_{k=1}^K \mathcal{L}_k f,$$

where $\omega_k$ is some portion of $\Omega$ associated with $t_k$ and $\mathcal{L}_k f$ is to be understood as the integral of $f$ over $\omega_k$. There is an assumption here that $\Omega = \bigcup_{k=1}^K \omega_k$ and that the intersection of $\omega_k$ and $\omega_{k'}$ is at most a facet shared by the simplices when $k \neq k'$ so that the integral can be decomposed as the sum above. On the other hand, when $\mathcal{L}$ is a derivative operator, $\mathcal{L}_k$ is considered to be the derivative at the point $\mathbf{x}_{k,0}$. In both cases of integration and differentiation, $\mathcal{L}_k$ is applied to a local interpolant of $f$ as the approximation of the action of $\mathcal{L}_k$ on $f$. The following two subsections describe two equivalent approaches to constructing the local interpolant.

**2.1. Approximation on the Basis of Shifts of the Kernel and Polynomials.** Consider a linear combination of (conditionally-) positive definite kernels, $\varphi$, evaluated at a set of center points,

$$\phi_{k,n,j}(\mathbf{x}) := \varphi\left(\|\mathbf{x} - \mathbf{x}_{k,j}\|_2\right), j = 1, 2, \ldots, n$$

and (supplemental) multivariate polynomial (multinomial) terms up to total degree $m$. As is demonstrated here in the definition of $\phi_{k,n,j}$, and throughout the remainder of this work, multiple subscripting using $k$, $n$ and $m$, in particular, explicitly indicates those variables that depend on each of these parameters.

Define $\{\pi_{k,l}(\mathbf{x})\}_{l=1}^{M_{d,m}}$, with

$$M_{d,m} = \begin{pmatrix} m+d \\ d \end{pmatrix},$$

to be the set of all of the multivariate polynomial terms up to total degree $m$. Using multi-index notation these terms have the form $\pi_{k,l}(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_{k,0})^{\boldsymbol{\alpha}_l}$, with $\boldsymbol{\alpha}_l = (\alpha_{l,1}, \alpha_{l,2}, \ldots, \alpha_{l,d})$ ($\alpha_{l,j} \geq 0$, $j = 1, 2, \ldots, d$) a multi-index and $|\boldsymbol{\alpha}_l| \leq m$. As a reminder, with multi-indices (see, e.g., [17] section 11.1)

- $\mathbf{x}^{\boldsymbol{\alpha}_l} = x_1^{\alpha_{l,1}} x_2^{\alpha_{l,2}} \cdots x_d^{\alpha_{l,d}}$,
- $|\boldsymbol{\alpha}_l| = \sum_{j=1}^{d} \alpha_{l,d}$,
- $\partial^{\boldsymbol{\alpha}_l} = \frac{\partial^{\alpha_{l,1}}}{\partial x_1^{\alpha_{l,1}}} \frac{\partial^{\alpha_{l,2}}}{\partial x_2^{\alpha_{l,2}}} \cdots \frac{\partial^{\alpha_{l,d}}}{\partial x_d^{\alpha_{l,d}}}$, and
- $\boldsymbol{\alpha}_l! = (\alpha_{l,1}!)(\alpha_{l,2}!) \cdots (\alpha_{l,d}!)$.

As is typical when supplementing a kernel basis set with polynomials, the interpolant is constructed as

$$s_{k,n,m}[f](\mathbf{x}) := \sum_{j=1}^{n} \lambda_{k,n,m,j}[f] \phi_{k,n,j}(\mathbf{x}) + \sum_{l=1}^{M_{d,m}} \gamma_{k,n,m,l}[f] \pi_{k,l}(\mathbf{x}).$$

If, instead, the $n \times 1$ vector $\boldsymbol{\Phi}_{k,n}(\mathbf{x})$ and $M_{d,m} \times 1$ vector $\boldsymbol{\Pi}_{k,m}(\mathbf{x})$ consist of all of the basis functions evaluated at $\mathbf{x}$, i.e.,

$$\boldsymbol{\Phi}_{k,n}(\mathbf{x}) = \begin{bmatrix} \phi_{k,n,1}(\mathbf{x}) & \phi_{k,n,2}(\mathbf{x}) & \cdots & \phi_{k,n,n}(\mathbf{x}) \end{bmatrix}^T$$

and

$$\boldsymbol{\Pi}_{k,m}(\mathbf{x}) = \begin{bmatrix} \pi_{k,1}(\mathbf{x}) & \pi_{k,2}(\mathbf{x}) & \cdots & \pi_{k,M_{d,m}}(\mathbf{x}) \end{bmatrix}^T,$$

then the interpolant can be written

$$s_{k,n,m}[f](\mathbf{x}) := \begin{bmatrix} \boldsymbol{\lambda}_{k,n,m}[f] \\ \boldsymbol{\gamma}_{k,n,m}[f] \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\Phi}_{k,n}(\mathbf{x}) \\ \boldsymbol{\Pi}_{k,m}(\mathbf{x}) \end{bmatrix},$$

with the coefficient vectors

$$\boldsymbol{\lambda}_{k,n,m}[f] = \begin{bmatrix} \lambda_{k,n,m,1}[f] & \lambda_{k,n,m,2}[f] & \cdots & \lambda_{k,n,m,n}[f] \end{bmatrix}^T$$

and

$$\boldsymbol{\gamma}_{k,n,m}[f] = \begin{bmatrix} \gamma_{k,n,m,1}[f] & \gamma_{k,n,m,2}[f] & \cdots & \gamma_{k,n,m,M_{d,m}}[f] \end{bmatrix}^T.$$

To ensure that $s_{k,n,m}[f]$ interpolates $f$ at the set of points in $\mathcal{N}_{k,n}$, the coefficient vectors are chosen to satisfy the interpolation conditions ($j = 1, 2, \ldots, n$),

$$s_{k,n,m}[f](\mathbf{x}_{k,j}) = f(\mathbf{x}_{k,j}).$$

This leads to an underdetermined system of linear equations to solve for the coefficient vectors, so the typical constraints applied to kernel-based interpolants (where the kernel is conditionally positive definite) are also enforced, i.e., ($l = 1, 2, \ldots, M_{d,m}$)

$$\sum_{j=1}^{n} \lambda_{k,n,m,j}[f] \pi_{k,l}(\mathbf{x}_{k,j}) = 0.$$

Consider the $n \times 1$ vector

$$\mathbf{f}_{k,n} = \begin{bmatrix} f(\mathbf{x}_{k,1}) & f(\mathbf{x}_{k,2}) & \cdots & f(\mathbf{x}_{k,n}) \end{bmatrix}^T,$$

consisting of the values of $f$ at the points in $\mathcal{N}_{k,n}$. Likewise, let the $n \times n$ matrix $\Phi_{k,n}$ and $n \times M_{d,m}$ Vandermonde matrix $P_{k,n,m}$ contain the values of the kernel basis elements and polynomial basis elements, respectively, evaluated at each point in $\mathcal{N}_{k,n}$. That is, these matrices have entries

$$[\Phi_{k,n}]_{ij} = \phi_{k,n,j}\left(\mathbf{x}_{k,i}\right), \text{ for } i,j = 1,2,\ldots,n,$$

and

$$[P_{k,n,m}]_{il} = \pi_{k,l}(\mathbf{x}_{k,i}), \text{ for } i = 1,2,\ldots,n \text{ and } l = 1,2,\ldots,M_{d,m}.$$

Then, satisfaction of the interpolation conditions and constraints amounts to solving the system of linear equations

$$S_{k,n,m}\begin{bmatrix} \boldsymbol{\lambda}_{k,n,m}[f] \\ \boldsymbol{\gamma}_{k,n,m}[f] \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{k,n} \\ \mathbf{0}_{M_{d,m}} \end{bmatrix}$$

with the $(n + M_{d,m}) \times (n + M_{d,m})$ matrix

$$S_{k,n,m} = \begin{bmatrix} \Phi_{k,n}^T & P_{k,n,m} \\ P_{k,n,m}^T & 0_{M_{d,m}\times M_{d,m}} \end{bmatrix}$$

and $\mathbf{0}_{M_{d,m}}$ denoting an $M_{d,m} \times 1$ vector of zeros. Assuming that the kernel $\varphi$ is conditionally positive-definite of order $m$ and the set $\mathcal{N}_{k,n}$ is unisolvent on the space, $\mathbb{P}_m^d$, of $d$-variate polynomials up to degree $m$, the matrix $S_{k,n,m}$ is invertible and the interpolation problem has a unique solution [39].

**2.2. The Lagrange Form of the Interpolant.** An alternative formulation of the interpolant, more convenient for the presentation of theoretical results, is written

$$s_{k,n,m}[f](\mathbf{x}) = \sum_{i=1}^{n} \psi_{k,n,m,i}(\mathbf{x}) f(\mathbf{x}_{k,i}),$$

where the new set of basis functions satisfy the cardinal property

$$\psi_{k,n,m,i}(\mathbf{x}_{k,j}) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}.$$

The two sets of basis functions are related by

$$S_{k,n,m}\begin{bmatrix} \boldsymbol{\Psi}_{k,n,m}(\mathbf{x}) \\ \boldsymbol{\Xi}_{k,n,m}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}_{k,n}(\mathbf{x}) \\ \boldsymbol{\Pi}_{k,m}(\mathbf{x}) \end{bmatrix} \tag{2.1}$$

with only the $n \times 1$ vector

$$\boldsymbol{\Psi}_{k,n,m}(\mathbf{x}) = \begin{bmatrix} \psi_{k,n,m,1}(\mathbf{x}) & \psi_{k,n,m,2}(\mathbf{x}) & \cdots & \psi_{k,n,m,n}(\mathbf{x}) \end{bmatrix}.$$

required to form $s_{k,n,m}[f]$.

**3. Error Estimation in Local Kernel Approximations.** An estimate of the error in the approximate application of $\mathcal{L}$ is necessary to determine locations in the domain of interest that require refinement to achieve a desired tolerance. In the following subsections a convenient expression of the pointwise error and an estimate that closely matches that error are discussed. Further, an expression for the interpolation coefficients that correspond to the shifts of the kernel in the basis used for approximation (that is, the coefficient vector $\boldsymbol{\lambda}_{k,n,m}[f]$) is given in terms of linear combinations of the weights of strictly polynomial based approximations of a set of (mixed partial) derivatives.

**3.1. Pointwise Error in the Local Kernel Based Interpolant.** For reasons that will be made clear in the following sections let $\mu \in \mathbb{Z}$ and $\mu \geq 1$. The Taylor formula of a function $f(\mathbf{x})$ about the point $\mathbf{x}_{k,0}$, with $f$ having continuous mixed partial derivatives up to order $m+\mu+1$ in a convex neighborhood of $\mathbf{x}_{k,0}$, can be written as (see, e.g., [4, 37])

$$f(\mathbf{x}) = \sum_{l=1}^{M_{d,m+\mu}} \frac{1}{\boldsymbol{\alpha}_l!} \partial^{\boldsymbol{\alpha}_l} f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_{k,0}} \pi_{k,l}(\mathbf{x}) + R_{m+\mu}[f](\mathbf{x}) \tag{3.1}$$

where the remainder term is expressible as

$$R_{m+\mu}[f](\mathbf{x}) = \sum_{l=M_{d,m+\mu}+1}^{M_{d,m+\mu+1}} \frac{m+\mu+1}{\boldsymbol{\alpha}_l!} \int_0^1 \frac{\partial^{\boldsymbol{\alpha}_l} f(\mathbf{y})|_{\mathbf{y}=\mathbf{x}+t(\mathbf{x}-\mathbf{x}_{k,0})}}{(m+\mu)!} (1-t)^{m+\mu} dt (\mathbf{x}-\mathbf{x}_{k,0})^{\boldsymbol{\alpha}_l}.$$

When evaluated inside the convex hull of $\mathcal{N}_{k,n}$ this remainder term behaves as $O(h_{k,n}^{m+\mu+1})$ as $h_{k,n} \to 0$.

LEMMA 1. *Suppose that $f$ has continuous mixed partial derivatives up to order $m+\mu+1$ in a convex neighborhood of $\mathbf{x}_{k,0}$ containing $\mathcal{N}_{k,n}$. Further assume that the kernel $\varphi$ is conditionally positive-definite of order $m$ and the set $\mathcal{N}_k$ is unisolvent on the space, $\mathbb{P}_m^d$, of $d$-variate polynomials up to degree $m$. The point-wise error in the kernel based interpolant $s_{k,n,m}[f]$ is*

$$\begin{aligned} &s_{k,n,m}[f](\mathbf{x}) - f(\mathbf{x}) \\ &= \sum_{l=M_{d,m}}^{M_{d,m+\mu}} \frac{1}{\boldsymbol{\alpha}_l!} \partial^{\boldsymbol{\alpha}_l} f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_{k,0}} E_{k,n,m,l}(\mathbf{x}) + \left(s_{k,n,m}[R_{m+\mu}[f]](\mathbf{x}) - R_{m+\mu}[f](\mathbf{x})\right). \end{aligned} \tag{3.2}$$

*with*

$$E_{k,n,m,l}(\mathbf{x}) = s_{k,n,m}[\pi_{k,m,l}](\mathbf{x}) - \pi_{k,m,l}(\mathbf{x})$$

*the error in approximating the polynomial term $\pi_{k,m,l}$ with a kernel interpolant when including polynomials up to degree $m$.*

*Proof.* Existence of the unique interpolant $s_{k,n,m}[f]$ follows from the kernel $\varphi$ being conditionally positive-definite of order $m$ and the set $\mathcal{N}_{k,n}$ being unisolvent on the space, $\mathbb{P}_m^d$, of $d$-variate polynomials up to degree $m$. To develop a convenient expression for the error in the kernel based interpolant, it is useful to evaluate the Taylor formula at each point in $\mathcal{N}_{k,n}$ and write

$$\begin{aligned} s_{k,n,m}[f] = &\sum_{i=1}^n \psi_{k,n,m,i}(\mathbf{x}) \left( \sum_{l=1}^{M_{d,m}} \frac{1}{\boldsymbol{\alpha}_l!} \partial^{\boldsymbol{\alpha}_l} f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_{k,0}} \pi_{k,l}(\mathbf{x}_{k,i}) \right) + \cdots \\ &\sum_{i=1}^n \psi_{k,n,m,i}(\mathbf{x}) \left( \sum_{l=M_{d,m}+1}^{M_{d,m+\mu}} \frac{1}{\boldsymbol{\alpha}_l!} \partial^{\boldsymbol{\alpha}_l} f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_{k,0}} \pi_{k,l}(\mathbf{x}_{k,i}) \right) + \cdots \\ &\sum_{i=1}^n \psi_{k,n,m,i}(\mathbf{x}) R_{m+\mu}[f](\mathbf{x}_{k,i}). \end{aligned}$$

Swapping the orders of summation in the first two lines of the expression reveals

$$s_{k,n,m}[f] = \sum_{l=1}^{M_{d,m}} \frac{1}{\boldsymbol{\alpha}_l!} \partial^{\boldsymbol{\alpha}_l} f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_{k,0}} \left(s_{k,n,m}[\pi_{k,l}](\mathbf{x})\right) + \cdots$$

$$\sum_{l=M_{d,m}+1}^{M_{d,m+\mu}} \frac{1}{\boldsymbol{\alpha}_l!} \partial^{\boldsymbol{\alpha}_l} f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_{k,0}} \left(s_{k,n,m}[\pi_{k,l}](\mathbf{x})\right) + \cdots$$

$$\left(s_{k,n,m}[R_{m+\mu}[f]](\mathbf{x})\right). \tag{3.3}$$

Notice that $s_{k,n,m}[\pi_{k,l}](\mathbf{x}) = \pi_{k,l}(\mathbf{x})$ for $l \leq M_{d,m}$ so that the first sum in (3.3) is identical to that in (3.1). Therefore

$$s_{k,n,m}[f](\mathbf{x}) - f(\mathbf{x}) =$$

$$\sum_{l=M_{d,m}+1}^{M_{d,m+\mu}} \frac{1}{\boldsymbol{\alpha}_l!} \partial^{\boldsymbol{\alpha}_l} f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_{k,0}} \left(s_{k,n,m}[\pi_{k,m,l}](\mathbf{x}) - \pi_{k,m,l}(\mathbf{x})\right) + \cdots$$

$$\left(s_{k,n,m}[R_{m+\mu}[f]](\mathbf{x}) - R_{m+\mu}[f](\mathbf{x})\right).$$

Utilizing the definition of $E_{k,n,m,M_{d,m}+l}$ produces the desired result. ∎

*Remark* 3.1. Although in this work $\mu > 0$, if instead $\mu = 0$, then the first term in (3.2) does not appear, and this error formula is analogous to the one presented in [2] but in a different form. Utilizing similar arguments, the first sum in (3.2) behaves as $O(h_{k,n}^{m+1})$ as $h_{k,n} \to 0$, while the second term behaves as $O(h_{k,n}^{m+\mu+1})$ as $h_{k,n} \to 0$. Therefore, the error in the interpolant is dominated by the first term.

*Remark* 3.2. The choices of both $m$ and $\mu$ impose a minimum smoothness requirement on $f$. In particular, $f$ must have continuous mixed partial derivatives up to order $m + \mu + 1$ in a neighborhood of $\mathbf{x}_{k,0}$ containing $\mathcal{N}_{k,n}$. As long as this requirement is satisfied, even in the case of a function with finitely many continuous mixed partial derivatives, the results here still apply. However, in certain cases, especially where solutions of limited smoothness are required, this may be too stringent or prohibitive a requirement. If possible, $m$ and $\mu$ should be chosen to be as small as necessary in these cases.

**3.2. Finite Difference Expressions of Kernel Interpolation Coefficients.** As long as $n > M_{d,m}$, the constraints $\sum_{j=1}^{n} \lambda_{k,n,m,j}[f]\pi_{k,l}(\mathbf{x}_{k,j}) = 0$, $l = 1, 2, \ldots, M_{d,m}$, form an underdetermined system of equations $P_{k,n,m}^T \boldsymbol{\lambda}_{k,n,m} = \mathbf{0}_{M_{d,m}}$. If the set $\mathcal{N}_{k,n}$ is unisolvent with respect to the span of the set $\{\pi_{k,l}(\mathbf{x})\}_{l=1}^{M_{d,m+\mu}}$ the dimension of the null space of $P_{k,n,m}^T$ is $n - M_{d,m}$. Suppose that $\mathbf{d}_{k,n,l}$ satisfies

$$\begin{bmatrix} \pi_{k,1}(\mathbf{x}_{k,1}) & \pi_{k,1}(\mathbf{x}_{k,2}) & \cdots & \pi_{k,1}(\mathbf{x}_{k,n}) \\ \pi_{k,2}(\mathbf{x}_{k,1}) & \pi_{k,2}(\mathbf{x}_{k,2}) & \cdots & \pi_{k,2}(\mathbf{x}_{k,n}) \\ \vdots & \vdots & & \vdots \\ \pi_{k,l-1}(\mathbf{x}_{k,1}) & \pi_{k,l-1}(\mathbf{x}_{k,2}) & \cdots & \pi_{k,l-1}(\mathbf{x}_{k,n}) \\ \pi_{k,l}(\mathbf{x}_{k,1}) & \pi_{k,l}(\mathbf{x}_{k,2}) & \cdots & \pi_{k,l}(\mathbf{x}_{k,n}) \\ \pi_{k,l+1}(\mathbf{x}_{k,1}) & \pi_{k,l+1}(\mathbf{x}_{k,2}) & \cdots & \pi_{k,l+1}(\mathbf{x}_{k,n}) \\ \vdots & \vdots & & \vdots \\ \pi_{k,M_{d,m+\mu}}(\mathbf{x}_{k,1}) & \pi_{k,M_{d,m+\mu}}(\mathbf{x}_{k,2}) & \cdots & \pi_{k,M_{d,m+\mu}}(\mathbf{x}_{k,n}) \end{bmatrix} \mathbf{d}_{k,n,l} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \boldsymbol{\alpha}_l! \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

The set $\{\mathbf{d}_{k,n,l}\}_{l=M_{d,m}+1}^{M_{d,m+\mu}}$ is linearly independent and contained in the null space of $P_{k,n,m}^T$. Therefore, there exists a set of weights $\beta_{k,n,m,l}[f]$, $l = M_{d,m}+1,\ldots,n$, such that

$$\boldsymbol{\lambda}_{k,n,m}[f] = \sum_{l=M_{d,m}+1}^{M_{d,m+\mu}} \beta_{k,n,m,l}[f]\mathbf{d}_{k,n,l} + \sum_{l=M_{d,m+\mu}+1}^{n} \beta_{k,n,m,l}[f]\mathbf{g}_{k,n,l}.$$

where $\{\mathbf{g}_{k,n,l}\}_{l=M_{d,m+\mu}+1}^{n}$ is a linearly independent set of vectors in the null space $P_{k,n,m+\mu}^T$ (and so the set is also in the null space of $P_{k,n,m}^T$).

Notice also that if $f$ has continuous mixed partial derivatives up to order $m+\mu+1$ in a convex neighborhood of $\mathbf{x}_{k,0}$ containing $\mathcal{N}_{k,n}$, then the Taylor formula of $f$ about $\mathbf{x}_{k,0}$ can be evaluated at each point in the set and

$$\mathbf{d}_{k,n,l}^T\mathbf{f}_{k,n} = \partial^{\boldsymbol{\alpha}_l} f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_{k,0}} + \mathbf{d}_{k,n,l}^T \begin{bmatrix} R_{m+\mu}[f](\mathbf{x}_{k,1}) \\ R_{m+\mu}[f](\mathbf{x}_{k,2}) \\ \vdots \\ R_{m+\mu}[f](\mathbf{x}_{k,n}) \end{bmatrix}. \tag{3.4}$$

The elements of the vectors $\mathbf{d}_{k,n,l}$ are $O(h_{k,n}^{-|\boldsymbol{\alpha}_l|})$ as $h_{k,n} \to 0$. Therefore, (3.4) is an $O(h_{k,n}^{m+\mu+1-|\boldsymbol{\alpha}_l|})$ approximation to $\partial^{\boldsymbol{\alpha}_l} f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_{k,0}}$ as $h_{k,n} \to 0$.

Define

$$D_{k,n,m+\mu} = \begin{bmatrix} \mathbf{d}_{k,n,M_{d,m}+1} & \mathbf{d}_{k,n,M_{d,m}+2} & \cdots & \mathbf{d}_{k,n,M_{d,m+\mu}} \end{bmatrix},$$

$$G_{k,n,m+\mu} = \begin{bmatrix} \mathbf{g}_{k,n,M_{d,m+\mu}+1} & \mathbf{g}_{k,n,M_{d,m+\mu}+2} & \cdots & \mathbf{g}_{k,n,n} \end{bmatrix}$$

and

$$\boldsymbol{\beta}_{k,n,m}[f] = \begin{bmatrix} \beta_{k,n,m,M_{d,m}+1}[f] & \beta_{k,n,m,M_{d,m}+2}[f] & \cdots & \beta_{k,n,m,n}[f] \end{bmatrix}^T.$$

Then, in what follows it is more convenient to write

$$\boldsymbol{\lambda}_{k,n,m}[f] = \begin{bmatrix} D_{k,n,m+\mu} & G_{k,n,m+\mu} \end{bmatrix} \boldsymbol{\beta}_{k,n,m}[f].$$

**3.3. An Estimate of the Error in Local Kernel Based Approximations.** To construct a method that locally adapts the spacing of the points to reduce the error,

$$\|\mathcal{L}_k s_{k,n,m}[f] - \mathcal{L}_k f\| = \|\mathcal{L}_k(s_{k,n,m}[f] - f)\|,$$

in the approximation of $\mathcal{L}_k f$ an estimate of the error must be utilized. Here

$$\|\mathcal{L}_k s_{k,n,m}[f] - \mathcal{L}_k s_{k,n,m+\mu}[f]\| = \|\mathcal{L}_k(s_{k,n,m}[f] - s_{k,n,m+\mu}[f])\|,$$

$\mu \in \mathbb{Z}$ and $\mu \geq 1$, is used as the estimate of the error. In the case of approximately evaluating a derivative of $f$ at a point the error and estimate are taken to be evaluated at each point in $\mathbf{x}_{k,0} \in \mathcal{X}_0$. On the other hand, when approximating definite integrals the error and estimate are evaluated once for each subdomain $\omega_k$ (to which there is an associated point $\mathbf{x}_{k,0} \in \mathcal{X}_0$). In either case, when the error estimate is larger than a prescribed tolerance new points are added to the sets $\mathcal{S}$ and $\mathcal{X}_0$. The following lemma and subsequent theorem demonstrate that this error estimate well approximates the error in the approximation for sufficiently smooth $f$.

LEMMA 2. *Suppose that the kernel $\varphi$ is conditionally positive-definite of order $m+\mu$ and the set $\mathcal{N}_{k,n}$ is unisolvent on the space, $\mathbb{P}^d_{m+\mu}$, of $d$-variate polynomials up to degree $m+\mu$. Further, let $\partial^{\boldsymbol{\alpha}_j} f(\mathbf{x})$, $j=1,2,\ldots,n$, be continuous in a convex neighborhood of $\mathbf{x}_{k,0}$ containing $\mathcal{N}_{k,n}$. If $n \geq M_{d,m+\mu}$, then for a set of scalars $C_{k,n,m,(l-M_{d,m})(i-M_{d,m+\mu})}$ independent of $f$, with $i = M_{d,m+\mu}+1, M_{d,m+\mu}+2,\ldots,n$ and $l = M_{d,m}+1, M_{d,m}+2,\ldots,M_{d,m+\mu}$,*

$$s_{k,n,m}[f](x) - s_{k,n,m+\mu}[f](\mathbf{x}) =$$

$$\sum_{l=M_{d,m}+1}^{M_{d,m+\mu}} \frac{1}{\boldsymbol{\alpha}_l!}\mathbf{d}^T_{k,n,m,l}\mathbf{f}_{k,n}E_{k,n,m,l}(\mathbf{x}) + \cdots$$

$$\sum_{i=M_{d,m+\mu}+1}^{n} \mathbf{g}^T_{k,n,m,i}\mathbf{f}_{k,n} \sum_{l=M_{d,m}+1}^{M_{d,m+\mu}} C_{k,n,m,(l-M_{d,m})(i-M_{d,m+\mu})}\frac{1}{\boldsymbol{\alpha}_l!}E_{k,n,m,l}(\mathbf{x}). \tag{3.5}$$

*Proof.* Existence of the unique interpolants $s_{k,n,m}[f]$ and $s_{k,n,m+\mu}[f]$ follows from the kernel $\varphi$ being conditionally positive-definite of order $m+\mu$ and the set $\mathcal{N}_{k,n}$ being unisolvent on the space, $\mathbb{P}^d_{m+\mu}$, of $d$-variate polynomials up to degree $m+\mu$. Let $n \geq M_{d,m+\mu}$ and define

$$V_{k,n,m+\mu} = \begin{bmatrix} \tilde{P}_{k,n,m+\mu} \\ 0_{(n+M_{d,m})\times(M_{d,m+\mu}-M_{d,m})} \end{bmatrix},$$

where $\tilde{P}_{k,n,m+\mu}$ consists of the last $n-M_{d,m}$ columns of $P_{k,n,m+\mu}$. Notice that

$$S_{k,n,m+\mu} = \begin{bmatrix} S_{k,n,m} & V_{k,n,m+\mu} \\ V^T_{k,n,m+\mu} & 0_{M_{d,m+\mu}-M_{d,m}} \end{bmatrix},$$

so that if $S_{k,n,m}$ is invertible and $P_{k,n,m+\mu}$ is full rank (a consequence of the unisolvency of $\mathcal{N}_{k,n}$), block matrix inversion of $S_{k,n,m+\mu}$ yields

$$\Psi_{k,n,m+\mu}(\mathbf{x}) = \Psi_{k,n,m}(\mathbf{x}) - \cdots$$

$$\Lambda_{k,n,m}\left(\tilde{P}^T_{k,n,m+\mu}\Lambda_{k,n,m}\right)^{-1}\left(\tilde{P}^T_{k,n,m+\mu}\Psi_{k,n,m}(\mathbf{x}) - \tilde{\boldsymbol{\Pi}}_{k,n,m+\mu}(\mathbf{x})\right),$$

where

$$\Lambda_{k,n,m} = \begin{bmatrix} \boldsymbol{\lambda}_{k,n,m}[\pi_{k,M_{d,m}+1}] & \boldsymbol{\lambda}_{k,n,m}[\pi_{k,M_{d,m}+2}] & \cdots & \boldsymbol{\lambda}_{k,n,m}[\pi_{k,M_{d,m+\mu}}] \end{bmatrix}$$

is the matrix with columns consisting of the interpolation coefficients corresponding to the kernel basis elements when interpolating the polynomial basis elements of degree $m+1, m+2,\ldots,m+\mu$ using only the kernel basis set supplemented by polynomial terms up to degree $m$. The results of section 3.2 indicate that the matrix $\Lambda_{k,n,m}$ can be written as

$$\Lambda_{k,n,m} = \begin{bmatrix} D_{k,n,m+\mu} & G_{k,n,m+\mu} \end{bmatrix} B_{k,n,m},$$

where

$$B_{k,n,m} = \begin{bmatrix} \boldsymbol{\beta}_{k,n,m}[\pi_{k,M_{d,m}+1}] & \boldsymbol{\beta}_{k,n,m}[\pi_{k,M_{d,m}+2}] & \cdots & \boldsymbol{\beta}_{k,n,m}[\pi_{k,M_{d,m+\mu}}] \end{bmatrix}.$$

Further,

$$\begin{aligned}\tilde{P}^T_{k,n,m+\mu}\Lambda_{k,n,m} =& \tilde{P}^T_{k,n,m+\mu}\left[\begin{array}{cc} D_{k,n,m+\mu} & G_{k,n,m+\mu}\end{array}\right]\hat{B}_{k,n,m}\\ =& A_{m+\mu}\hat{B}_{k,n,m}\end{aligned}$$

with

$$A_{m+\mu} = \left[\begin{array}{cccc} \boldsymbol{\alpha}_{M_{d,m}+1}! & & & \\ & \boldsymbol{\alpha}_{M_{d,m}+2}! & & \\ & & \ddots & \\ & & & \boldsymbol{\alpha}_{M_{d,m+\mu}}!\end{array}\right].$$

and $\hat{B}_{k,n,m}$ the first $M_{d,m+\mu} - M_{d,m}$ rows of $B_{k,n,m}$. Therefore, after substitution

$$\begin{aligned}\Psi_{k,n,m+\mu}(\mathbf{x}) = \Psi_{k,n,m}(\mathbf{x}) - \Bigg(&\left[\begin{array}{cc} D_{k,n,m+\mu} & G_{k,n,m+\mu}\end{array}\right]B_{k,n,m}\left(\hat{B}_{k,n,m}\right)^{-1}\\ &A^{-1}_{m+\mu}\left(\tilde{P}^T_{k,n,m+\mu}\Psi_{k,n,m}(\mathbf{x}) - \tilde{\mathbf{\Pi}}_{k,n,m+\mu}(\mathbf{x})\right)\Bigg).\end{aligned}$$

with $\tilde{\mathbf{\Pi}}_{k,n,m+\mu}(\mathbf{x})$ the last $M_{d,m+\mu} - M_{d,m}$ entries of $\mathbf{\Pi}_{k,n,m+\mu}(\mathbf{x})$.

The difference of the two interpolants can be written

$$\begin{aligned}& s_{k,n,m}[f](x) - s_{k,n,m+\mu}[f](\mathbf{x})\\ &\quad =(\Psi_{k,n,m}(\mathbf{x}) - \Psi_{k,n,m+\mu}(\mathbf{x}))^T\mathbf{f}_{k,n}\\ &\quad =\Bigg(\left[\begin{array}{cc} D_{k,n,m+\mu} & G_{k,n,m+\mu}\end{array}\right]B_{k,n,m}\left(\hat{B}_{k,n,m}\right)^{-1}\\ &\qquad A^{-1}_{m+\mu}\left(\tilde{P}^T_{k,n,m}\Psi_{k,n,m}(\mathbf{x}) - \tilde{\mathbf{\Pi}}_{k,n,m+\mu}(\mathbf{x})\right)\Bigg)^T\mathbf{f}_{k,n},\end{aligned}$$

so that using the definition of $E_{k,n,m,l}$

$$\begin{aligned}& s_{k,n,m}[f](x) - s_{k,n,m+\mu}[f](\mathbf{x}) =\\ &\quad \sum_{l=M_{d,m}+1}^{M_{d,m+\mu}} \frac{1}{\boldsymbol{\alpha}_l!}\mathbf{d}^T_{k,n,m,l}\mathbf{f}_{k,n}E_{k,n,m,l}(\mathbf{x}) + \cdots\\ &\quad \sum_{i=M_{d,m+\mu}+1}^{n} \mathbf{g}^T_{k,n,m,i}\mathbf{f}_{k,n}\sum_{l=M_{d,m}+1}^{M_{d,m+\mu}} C_{k,n,m,(l-M_{d,m})(i-M_{d,m+\mu})}\frac{1}{\boldsymbol{\alpha}_l!}E_{k,n,m,l}(\mathbf{x})\end{aligned}$$

with $C_{k,n,m} = \tilde{B}_{k,n,m}\left(\hat{B}_{k,n,m}\right)^{-1}$, produces the desired result. ∎

Lemmas 1 and 2, when taken together, reveal that the error estimate provides an $O(h^{m+\mu+1})$ approximation for the interpolation error, particularly when $n = M_{d,m+\mu}$. This is summarized in the following theorem.

THEOREM 3.3. *Suppose that the kernel $\varphi$ is conditionally positive-definite of order $m$ and the set $\mathcal{N}_{k,n}$ is unisolvent on the space, $\mathbb{P}^d_{m+\mu}$, of d-variate polynomials up to degree $m+\mu$. Further, let $\partial^{\boldsymbol{\alpha}_j}f(\mathbf{x})$, $j = 1, 2, \ldots, n$, be continuous in a convex neighborhood of $\mathbf{x}_{k,0}$ containing $\mathcal{N}_{k,n}$. If $n = M_{d,m+\mu}$*

$$s_{k,n,m}[f](x) - s_{k,n,m+\mu}[f](\mathbf{x}) = s_{k,n,m}[f](x) - f(\mathbf{x}) + O(h^{m+\mu+1}_{k,n}).$$

*as $h_{k,n} \to 0$.*

*Proof.* In the case of $n = M_{d,m+\mu}$ the second term on the right hand side of (3.5) does not appear. According to section 3.2

$$\mathbf{d}_{k,n,m,l}^T \mathbf{f}_{k,n} = \partial^{\boldsymbol{\alpha}_l} f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_{k,0}} + \mathbf{d}_{k,n,l}^T \begin{bmatrix} R_{m+\mu}[f](\mathbf{x}_{k,1}) \\ R_{m+\mu}[f](\mathbf{x}_{k,2}) \\ \vdots \\ R_{m+\mu}[f](\mathbf{x}_{k,n}) \end{bmatrix}.$$

Substituting this expression into (3.5) reveals that

$$s_{k,n,m}[f](x) - s_{k,n,m+\mu}[f](\mathbf{x}) =$$

$$s_{k,n,m}[f](x) - f(\mathbf{x}) + \sum_{l=M_{d,m}+1}^{M_{d,m+\mu}} \frac{1}{\boldsymbol{\alpha}_l!} \mathbf{d}_{k,n,l}^T \begin{bmatrix} R_{m+\mu}[f](\mathbf{x}_{k,1}) \\ R_{m+\mu}[f](\mathbf{x}_{k,2}) \\ \vdots \\ R_{m+\mu}[f](\mathbf{x}_{k,n}) \end{bmatrix} E_{k,n,m,l}(\mathbf{x}) - \cdots$$

$$\left(s_{k,n,m}[R_{m+\mu}[f]](\mathbf{x}) - R_{m+\mu}[f](\mathbf{x})\right).$$

The terms [2]

$$E_{k,n,m,l}(\mathbf{x}) = s_{k,n,m}[\pi_{k,m,l}](\mathbf{x}) - \pi_{k,m,l}(\mathbf{x})$$

are at most $O(h_{k,n}^{|\boldsymbol{\alpha}_l|})$. Likewise, section 3.5 reveals that

$$\mathbf{d}_{k,n,l}^T \begin{bmatrix} R_{m+\mu}[f](\mathbf{x}_{k,1}) \\ R_{m+\mu}[f](\mathbf{x}_{k,2}) \\ \vdots \\ R_{m+\mu}[f](\mathbf{x}_{k,n}) \end{bmatrix} = O(h_{k,n}^{m+\mu+1-|\boldsymbol{\alpha}_l|}) \text{ as } h_{k,n} \to 0.$$

Finally, $s_{k,n,m}[R_{m+\mu}[f]](\mathbf{x}) - R_{m+\mu}[f](\mathbf{x}) = O(h_{k,n}^{m+\mu+1})$ as $h_{k,n} \to 0$ [2]. ∎

*Remark* 3.4. For $n > M_{d,m+\mu}$ the second sum on the right hand side of (3.5) does not readily appear to be an approximation for the second term on the right hand side of (3.2). However, the discussion in the proof of theorem 3.3 applies similarly and these terms are also at least one order higher than the size of the dominant term in the error in the kernel interpolant. For simplicity in presenting computational results, unless otherwise stated the computations presented herein use $n = M_{d,m+\mu}$.

*Remark* 3.5. The conclusion of theorem 3.3 highlights that the parameter $\mu$ impacts how closely the error estimate approximates the dominant term in (3.2). That is, larger values of $\mu \geq 1$ translate to more accurate estimates of the error. However, in practice there is little noticeable benefit to choosing $\mu$ much greater than 1, since the difference between the error and the estimate is already one or more orders of magnitude smaller than the actual error. Likewise, larger values of $\mu$ lead to larger systems of linear equations that must be solved to determine $s_{k,n,m+\mu}[f](\mathbf{x})$. This increase in the size of the system of equations can be drastic, particularly for dimension $d > 1$. Further, for functions that exhibit even or odd symmetry about one of the points $x_{k,0} \in \mathcal{X}_0$ it is possible that $s_{k,n,m}[f](x)$ and $s_{k,n,m+1}[f](x)$ will account for exactly the same terms in the Taylor formula so that their difference may be small even when the actual error is large. Additional discussion and computational demonstrations in sections 4.2 and 5 suggest that the choice of $\mu = 2$ is reasonable for balancing accuracy with the computational cost.

*Remark* 3.6. In analogy to what is presented in this work, in the case of solving, e.g., the linear PDE

$$\mathcal{L}u = g$$

with $g$ a given function and $u$ unknown and subject to certain boundary conditions, consider the error

$$\|\hat{u}_{k,n,m} - u_k\|. \tag{3.6}$$

Here $u_k$ is the solution $u$ evaluated at the $k^{\text{th}}$ node in the set $\mathcal{S}$ and $\hat{u}_{k,n,m}$ is an approximate solution at this point. The approximate solution could be obtained by constructing and differentiating a local kernel based interpolant of $u$, including polynomials up to degree $m$, at each point in $\mathcal{S}$ to determine weights for the approximate local action of $\mathcal{L}$ (as in, e.g., section 4.1), assembling a matrix containing these weights, imposing boundary conditions, and solving a system of linear equations. An error estimate in this case would be of the form

$$\|\hat{u}_{k,n,m+\mu} - \hat{u}_{k,n,m}\|, \tag{3.7}$$

with $\hat{u}_{k,n,m+\mu}$ an approximate solution constructed in a similar manner to $\hat{u}_{k,n,m}$ but including polynomial terms up to degree $m + \mu$. The differences in estimating the error when solving a PDE, including the impact of inverting a system of linear equations, from what is presented here for the direct application of a linear operator to a known function are complex enough that they warrant further study. Further differences and complications in implementing algorithms based on (3.7) as an error estimate are discussed in remark 4.1.

## 4. An Implementation of Adaptive Local Kernel Approximations.

Given the error estimate developed in the previous sections, an initial implementation of an $h$-adaptive method for approximating action of a linear operator is now presented. This implementation only adds nodes to the set $\mathcal{S}$ to decrease the spacing between nodes locally (refinement), but it does not remove nodes where the error estimate indicates the error is smaller than the prescribed tolerance (derefinement). Efficient computation of the estimate relies on the similarities between the systems of linear equations used to solve for weights when including in the basis for approximation polynomials up to order $m$ and polynomials up to order $m + \mu$.

### 4.1. Weight Computation.

Application of $\mathcal{L}_k$ to the interpolant written in the cardinal basis reveals

$$\mathcal{L}_k s_{k,n,m}[f] = \sum_{i=1}^{n} \left(\mathcal{L}_k \psi_{k,n,m,i}\right) f_{k,n,i} \tag{4.1}$$

Defining

$$\mathbf{w}_{k,n,m} = \left(\mathcal{L}_k \psi_{k,n,m,i}\right)$$

the approximate operation amounts to an inner product between a vector containing values of the function to which the operation is being applied and a vector of "weights" containing the operator $\mathcal{L}_k$ applied to the cardinal basis elements. This is a standard method for determining approximation weights, e.g., as in the case of finite difference or pseudospectral approximations of derivatives or for Newton-Cotes

quadrature weights for approximating definite integrals. In any case, these weights are independent of the function $f$. However, construction of the cardinal basis to determine these weights is unnecessary, and instead an equivalent process determines the weight set $\mathbf{w}_{k,n,m}$ utilizing (2.1) as

$$S_{k,n,m}\begin{bmatrix} \mathbf{w}_{k,n,m} \\ \mathbf{v}_{k,n,m} \end{bmatrix} = \begin{bmatrix} \mathcal{L}\mathbf{\Phi}_{k,n} \\ \mathcal{L}\mathbf{\Pi}_{k,m} \end{bmatrix}. \tag{4.2}$$

Here $\mathcal{L}_k\mathbf{\Phi}_{k,n}$ and $\mathcal{L}_k\mathbf{\Pi}_{k,m}$ are to be understood as the vectors obtained by entry-wise application of $\mathcal{L}_k$ to the entries of $\mathbf{\Phi}_{k,n}$ and $\mathbf{\Pi}_{k,m}$, respectively.

**4.2. Reduced Computational Cost for the Error Estimate.** Given that $S_{k,n,m}$ is the upper left block of $S_{k,n,m+\mu}$, computation of the error estimate can be completed more efficiently than solving two full systems of linear equations. Block matrix inversion of $S_{k,n,m+\mu}$ yields that if the solution to (4.2) is available, then

$$\mathbf{w}_{k,n,m+\mu} = \mathbf{w}_{k,n,m} - \Lambda_{k,n,m}(\tilde{P}^T_{k,n,m+\mu}\Lambda_{k,n,m})^{-1}\left(\mathcal{L}\tilde{\mathbf{\Pi}}_{k,m+\mu} - \tilde{P}^T_{k,n,m+\mu}\mathbf{w}_{k,n,m}\right). \tag{4.3}$$

The cost of determining $\mathbf{w}_{k,n,m}$ is $O((n + M_{d,m})^3)$, while the dominant part of the cost of determining $\mathbf{w}_{k,n,m+\mu}$ is only a further $O(n(M_{d,M+\mu} - M_{d,m})^2)$ incurred for the multiplication $\tilde{P}^T_{k,n,m+\mu}\Lambda_{k,n,m}$. Now, to guarantee unique values of $\mathbf{w}_{k,n,m+\mu}$ it is necessary that $n \geq M_{d,m+\mu}$.

These costs can be better understood by noting

$$M_{d,m+\mu} = \mathcal{M}_{d,m+\mu}M_{d,m},$$

where

$$\mathcal{M}_{d,m+\mu} = \frac{(m+\mu+d)!}{(m+\mu)!}\frac{m!}{(m+d)!},$$

and considering the case of $n = M_{d,m+\mu}$. For this choice of $n$, the cost of determining $\mathbf{w}_{k,n,m}$ is $\sim 2/3(\mathcal{M}_{d,m+\mu}+1)^3M^3_{d,m}$. Determination of $\mathbf{w}_{k,n,m+\mu}$ through (4.3) is then only an additional $\sim 8/3(\mathcal{M}_{d,m+\mu} - 1)^2(\mathcal{M}_{d,m+\mu} - 1/4)M^3_{d,m}$ operations. If instead, the system of equations

$$S_{k,n,m+\mu}\begin{bmatrix} \mathbf{w}_{k,n,m+\mu} \\ \mathbf{v}_{k,n,m+\mu} \end{bmatrix} = \begin{bmatrix} \mathcal{L}\mathbf{\Phi}_{k,n} \\ \mathcal{L}\mathbf{\Pi}_{k,m+\mu} \end{bmatrix} \tag{4.4}$$

is solved for $\mathbf{w}_{k,n,m+\mu}$, then the cost is an additional $\sim 16/3\mathcal{M}^3_{d,m+\mu}M^3_{d,m}$ operations.

To further illustrate the efficiencies in evaluating (4.3), define $\tau_{d,m}$ and $\tau_{d,m+\mu}$ to be the times it takes to solve (4.2) and (4.4), respectively. Further, let $\tau'_{d,m+\mu}$ be the time it takes to solve (4.3) for $\mathbf{w}_{k,n,m+\mu}$. Figure 1 illustrates $\tau_{d,m+\mu}/\tau_{d,m}$ in the first row and $\tau'_{d,m+\mu}/\tau_{d,m}$ in the second for various choices of $m$ and $\mu$ and for $d = 1, 2, 3$. These frames were generated by averaging the elapsed computation times of 1000 instances of solving each of these systems of equations for each choice of $m$, $\mu$ and $d$. Comparing the contour plots in the first and second rows demonstrates that in every case there is improvement in computational efficiency when solving (4.3) instead of the full system of linear equations and that these improvements become more apparent as $d$ increases. Note that all computations presented here were performed on a workstation with two Intel® Xeon® CPU E5-2697 v3 processors, each running at 2.60GHz, and 256 GB of memory running MATLAB R2022b.

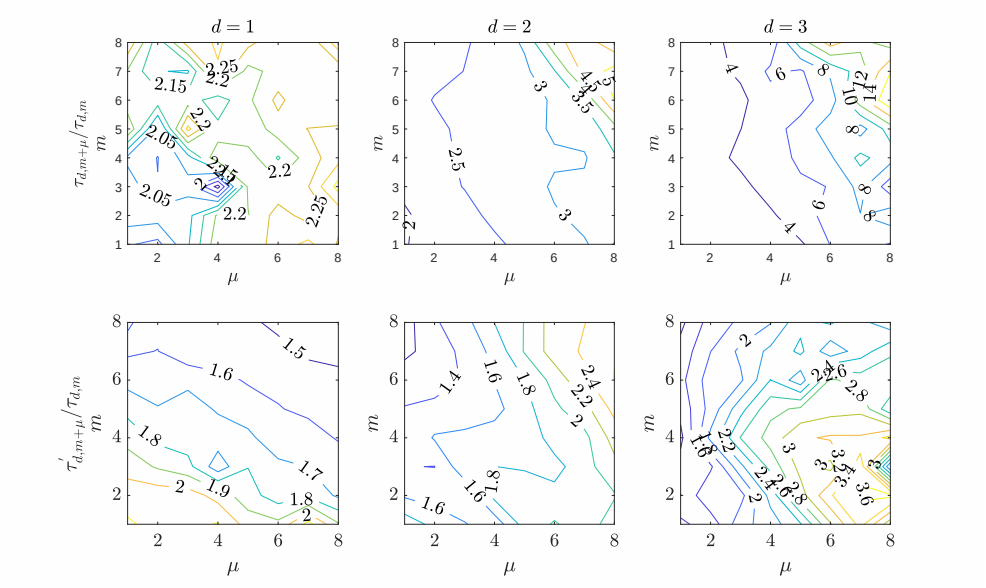


FIG. 1. *Demonstrations of how many times longer it takes to solve* (4.4) *(with average computation time* $\tau_{d,m+\mu}$*) and* (4.3) *(with average computation time* $\tau'_{d,m+\mu}$*) relative to the time it takes to solve* (4.2) *(with average computation time* $\tau_{d,m}$*). These frames were generated by averaging the elapsed computation times of 1000 instances of solving each of these systems of equations for each choice of* $m$*,* $\mu$ *and* $d$*.*

**4.3. A Naive Algorithm for Adaptive Kernel Based Approximation.** The focus of this paper is the development of an efficient error estimate for local "$h$-adaptive" kernel based methods. Such approaches to adaptation add new points to the set $\mathcal{S}$ in a neighborhood of those points in $\mathcal{X}_0$ where the error estimate is found to be too great. This has the effect of locally decreasing the size of $h_{k,n}$. With the estimate available, there are many possibilities for choosing the locations of these new points. The algorithm can be generically described in the following steps:

1: Given a set of $N^{(0)}$ nodes, $\mathcal{S}^{(0)}$, a set of $K^{(0)}$ points, $\mathcal{X}_0^{(0)} = \{\mathbf{x}_{k,0}^{(0)}\}_{k=1}^{K^{(0)}}$, a desired tolerance, $\varepsilon$, and a maximum number of refinement levels, $l_{\max}$. (And, if necessary, a tesselation $T^{(l)}$ of $S^{(l)}$.)
2: Set $l = 0$.
3: **while** $l \leq l_{\max}$ **do**
4: Set $\mathcal{R}^{(l)} = \emptyset$ ▷ *$\mathcal{R}^{(l)}$ stores the indices of the points in $\mathcal{X}_0^{(l)}$ requiring refinement.*
5: **for** $k \in \{1, 2 \ldots, K^{(l)}\}$ **do**
6: Determine the $n$ points in $S^{(l)}$ nearest to $\mathbf{x}_{k,0}^{(l)}$. Call this set of points $\mathcal{N}_{k,n}^{(l)}$.
7: **if** $l = 0$ **then**
8: Set $\mathcal{R}^{(l)} = \mathcal{R}^{(l)} \bigcup k$ ▷ *Initially, all points in $\mathcal{X}_0^{(l)}$ are marked for refinement.*
9: **else**
10: **if** there is an index $i$ such that $\mathbf{x}_{k,0}^{(l)} = \mathbf{x}_{i,0}^{(l-1)}$ **then** ▷ *Check if $\mathbf{x}_{k,0}^{(l)}$ existed at the previous level.*
11: **if** $\mathcal{N}_{k,n}^{(l)} \setminus \mathcal{N}_{i,n}^{(l-1)} \neq \emptyset$ **then** ▷ *Determine if the set of nodes nearest to $\mathbf{x}_{k,0}^{(l)}$ has changed.*

12: Set $\mathcal{R}^{(l)} = \mathcal{R}^{(l)} \bigcup k$ ▷ *If the set of nodes nearest to $\mathbf{x}_{k,0}^{(l)}$ has changed, then mark this point for refinement.*

13: **else**

14: Set $\mathcal{R}^{(l)} = \mathcal{R}^{(l)} \bigcup k$ ▷ *If $\mathbf{x}_{k,0}^{(l)}$ did not exist at the previous level, then mark this point for refinement.*

15: **if** $\mathcal{R}^{(l)} = \emptyset$ **then** ▷ *Determine if there are no elements of $\mathcal{X}_0^{(l)}$ that have been marked for refinement at this level.*

16: Break and return $\mathcal{L}_k^{(l)} s_{k,n,m}$ as the local approximation of $\mathcal{L}_k^{(l)} f$ associated with each $\mathbf{x}_{k,0}^{(l)} \in \mathcal{X}_0^{(l)}$

17: Set $\mathcal{S}^{(l+1)} = \mathcal{S}^{(l)}$ and $\mathcal{X}_0^{(l+1)} = \mathcal{X}_0^{(l)}$. (And, if necessary, set $T^{(l+1)} = T^{(l)}$.) ▷ *Initialize the various sets for the next level of refinement.*

18: **for** $k \notin \mathcal{R}^{(l)}$ **do**

19: Set $\mathcal{L}_k^{(l+1)} s_{k,n,m} = \mathcal{L}_k^{(l)} s_{k,n,m}$ ▷ *Retain the current approximation where refinement is unnecessary.*

20: **for** $k \in \mathcal{R}^{(l)}$ **do**

21: Compute $\mathcal{L}_k^{(l+1)} s_{k,n,m}$ and $\mathcal{L}_k^{(l+1)} s_{k,n,m+\mu}$ utilizing the nodes in $\mathcal{N}_{k,n}^{(l)}$. ▷ *Update the approximations where refinement was necessary at this level.*

22: **if** $\|\mathcal{L}_k^{(l+1)} s_{k,n,m} - \mathcal{L}_k^{(l+1)} s_{k,n,m+\mu}\| > \varepsilon$ **then** ▷ *Determine if the error indicator is too large.*

23: Add new nodes to the set $\mathcal{S}^{(l+1)}$ in a neighborhood of $\mathbf{x}_{k,0}^{(l)}$ and update the set $\mathcal{X}_0^{(l+1)}$ (and, if necessary, update $T^{(l+1)}$).

24: Set $K^{(l+1)} = |\mathcal{X}_0^{(l+1)}|$, then increment $l$ by 1.

Notice that in this implementation, at each level $l$, the approximation $\mathcal{L}_k^{(l+1)} s_{k,n,m}$ is updated for each $\mathbf{x}_{k,0}^{(l)} \in \mathcal{X}_0^{(l)}$ that has a set of nearest neighbors that has changed with the addition of new points to the set $\mathcal{S}^{(l)}$. This is not necessary, and it is possible to update the approximation only for those points $\mathbf{x}_{k,0}^{(l)}$ for which the error estimate in line 22 violates the specified tolerance, reducing the computational cost. Some computational experiments were performed with this method that reduces the cost; however, the best performance is achieved by updating the approximation for all points whose nearest neighbors have changed.

In all of the results presented here, the set $\mathcal{S}^{(0)}$ begins with $10^d$ equally spaced points in $[-1, 1]^d$. Implementation of line 23 of the algorithm then depends upon the operation $\mathcal{L}$. First, for approximating definite integrals at level $l$ of the algorithm note that

$$\int_{\Omega} f(\mathbf{x}) d\mathbf{x} = \sum_{k=1}^{K^{(l)}} \int_{\omega_k^{(l)}} f(\mathbf{x}) d\mathbf{x},$$

where $\omega_k^{(l)}$ is a portion of $\Omega$ associated with a simplex $t_k^{(l)}$, with midpoint $\mathbf{x}_{k,0}^{(l)}$, in a set of simplices constructed on the set $\mathcal{S}^{(l)}$. A set of new nodes, that may contain the point $\mathbf{x}_{k,0}^{(l)}$, is added to $\mathcal{S}^{(l+1)}$. In this implementation $\mathbf{x}_{k,0}^{(l)}$ and the midpoints of the facets of the simplices (e.g., edges of triangles when $d = 2$) are added to $\mathcal{S}^{(l+1)}$. Then $\mathbf{x}_{k,0}^{(l)}$ and $t_k^{(l)}$ are removed from $\mathcal{X}_0^{(l+1)}$ and $T^{(l+1)}$, respectively. To the set $T^{(l+1)}$ are added the simplices in a tesselation of the set of points containing the vertices of $t_k^{(l)}$ and the points that have been added to $\mathcal{S}^{(l+1)}$. The midpoints of these new simplices are then added to the set $\mathcal{X}_0^{(l+1)}$. The left half of figure 2 depicts how the

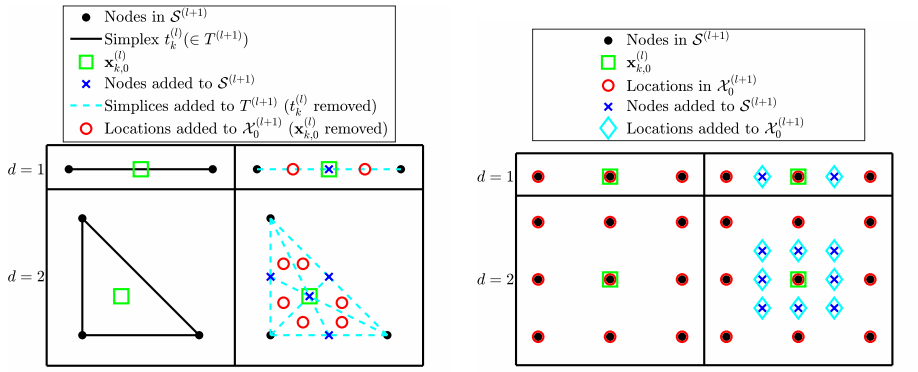


FIG. 2. *Illustration of a naive implementation of line 23 of the algorithm. The left half of this figure illustrates refinement when the operation $\mathcal{L}_k$ is local definite integration. The first column illustrates the state of the sets $S^{(l+1)}$, $\mathcal{X}_0^{(l+1)}$ and $T^{(l+1)}$ defined at step 17 of the algorithm. The second column then illustrates what is added to these sets. Notice that $\mathbf{x}_{k,0}^{(l)}$ and $t_k^{(l)}$ are removed from $\mathcal{X}_0^{(l+1)}$ and $T^{(l+1)}$, respectively, in step 23. The right half of this figure similarly illustrates refinement when the operation $\mathcal{L}_k$ is differentiation at $\mathbf{x}_{k,0}^{(l)}$. Nothing is removed from the sets in the case of differentiation.*

sets $S^{(l+1)}$, $\mathcal{X}_0^{(l+1)}$ and $T^{(l+1)}$ change from their definition in step 17 of the algorithm under refinement in step 23 when approximating a definite integral.

On the other hand, for approximating a derivative when $d = 1$ the two points $\mathbf{x}_{k,0}^{(l)} \pm h^{(0)}/2^{l+1}$ are added to both $\mathcal{S}^{(l+1)}$ and $\mathcal{X}_0^{(l+1)}$, while nothing is removed from either set. Here $h^{(0)}$ is the initial spacing between adjacent nodes. When $d = 2$, the points $\mathbf{x}_{k,0}^{(l)} \pm h^{(0)}/2^{l+1} \begin{bmatrix} 1 & 0 \end{bmatrix}^T$, $\mathbf{x}_{k,0}^{(l)} \pm h^{(0)}/2^{l+1} \begin{bmatrix} 0 & 1 \end{bmatrix}^T$, $\mathbf{x}_{k,0}^{(l)} \pm h^{(0)}/2^{l+1} \begin{bmatrix} 1 & 1 \end{bmatrix}^T$, and $\mathbf{x}_{k,0}^{(l)} \pm h^{(0)}/2^{l+1} \begin{bmatrix} 1 & -1 \end{bmatrix}^T$ are added to the both $\mathcal{S}^{(l+1)}$ and $\mathcal{X}_0^{(l+1)}$, while nothing is removed from either set. The right half of figure 2 depicts how the sets $S^{(l+1)}$ and $\mathcal{X}_0^{(l+1)}$ change from their definition in step 17 of the algorithm under refinement in step 21 when approximating a derivative at a point.

These strategies for adding nodes are certainly not the only ones available and may not be optimal. In particular, there is no guarantee that polynomial unisolvency is maintained with this method for adding additional nodes. This could be overcome by locally considering node sets that provide some guarantees of unisolvency, such as modifications of the Leja or Fekete points [22, 32, 9, 6] or through more recent methods specific to approximation by RBF-FD (e.g., as in [24]). However, the refinement strategies discussed above perform well enough to illustrate the performance of even naively implemented algorithms based on the error estimate.

*Remark* 4.1. As discussed in remark 3.6, estimation of the error when solving a PDE requires the construction of two different approximate solutions, both of which require the inversion of a, potentially large, system of linear equations. This alters some of the aspects of the algorithm presented above for approximating the application of $\mathcal{L}$ to a known function. In particular, error estimation and addition of new nodes (similar to lines 22 and 23 of the algorithm) would occur after updating appropriate weights for approximating the local action of $\mathcal{L}$ at all nodes marked for refinement, and then determining the solution of two systems of linear equations, which is an expensive procedure. Algorithms that leverage the relationship (4.3) between the

weights constructed from interpolants that include polynomials of degree $m$ and $m+\mu$ may be able to reduce this cost and deserve investigation.

**5. Computational Experiments.** Demonstrations of the agreement between the error estimate and actual error are provided in the following section when utilizing the algorithm described in section 4.3. These demonstrations explore two common linear operations in both $d = 1$ and $d = 2$ for two test functions with localized features that require significant refinement to be resolved.

**5.1. Test Functions and Kernel Selection for Computational Experiments.** Computational experiments were performed for $d = 1$ and $d = 2$ using the test functions

$$f_1(\mathbf{x}) = \sum_{i=1}^{2d} \frac{1}{1 + a \left\|\mathbf{x} - \mathbf{y}_i\right\|_2^2}$$

and

$$f_2(\mathbf{x}) = \sum_{i=1}^{2d} \mathrm{e}^{-a\|\mathbf{x}-\mathbf{y}_i\|_2^2},$$

both with $\mathbf{y}_i$ a randomly chosen shift in $(-1, 1)^d$. Graphically these functions are similar in character with features that become more localized as $a$ increases. However, the series expressions for the terms in $f_1$ and $f_2$ behave quite differently. That is,

$$\frac{1}{1 + a \left\|\mathbf{x} - \mathbf{y}\right\|_2^2} = \sum_{j=0}^{\infty} (-1)^j a^j \left(\left\|\mathbf{x} - \mathbf{y}\right\|_2^2\right)^j = \sum_{j=0}^{\infty} \sum_{|\boldsymbol{\beta}|=j} \frac{j!(-1)^j a^j}{\boldsymbol{\beta}!} (\mathbf{x} - \mathbf{y})^{2\boldsymbol{\beta}}$$

has radius of convergence $\|\mathbf{x} - \mathbf{y}\|_2 < 1/\sqrt{a}$ with terms that grow for all $j$ when outside the radius of convergence, and

$$\mathrm{e}^{-a\|\mathbf{x}-\mathbf{y}\|_2^2} = \sum_{j=0}^{\infty} (-1)^j \frac{a^j}{j!} \left(\left\|\mathbf{x} - \mathbf{y}\right\|_2^2\right)^j = \sum_{j=0}^{\infty} \sum_{|\boldsymbol{\beta}|=j} \frac{(-1)^j a^j}{\boldsymbol{\beta}!} (\mathbf{x} - \mathbf{y})^{2\boldsymbol{\beta}}$$

has infinite radius of convergence and terms that grow only until finite $j$ (for each fixed $\|\mathbf{x} - \mathbf{y}\|_2$).

While there are many options to choose from for the kernel used in the local approximations, the results here considered the use of $\varphi(r) = r^3$.

**5.2. Results in 1-Dimension.** To illustrate the performance of the algorithm described in section 4.3 for $d = 1$ it was applied to approximate the action of both

$$\mathcal{L}f = \int_{-1}^{1} f(x)dx \left(\text{i.e., } \mathcal{L}_k f = \int_{\omega_k} f(x)dx\right)$$

and

$$\mathcal{L}f = \frac{d}{dx} f(x),\ x \in [-1, 1] \left(\text{i.e., } \mathcal{L}_k f = \frac{d}{dx} f(x)\bigg|_{x=x_{k,0}}\right).$$

Consider the choice of $a = 1000$, $N^{(0)} = 10$, $m = 1$, and $\mu = 2$. Figure 3 illustrates a single example of adaptive quadrature for evaluating the integral of $f_2(x)$ over

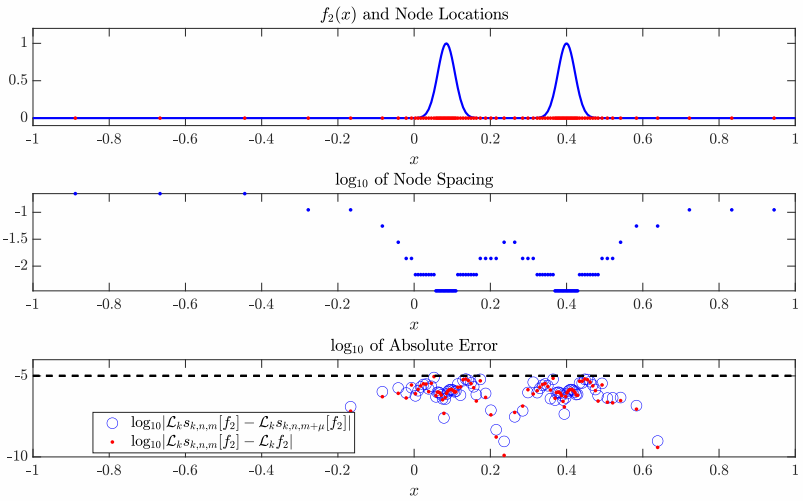


FIG. 3. *An example of adaptive approximate numerical quadrature of $f_2$ for $x \in [-1, 1]$ with $a = 1000$, $m = 1$, $\mu = 2$, $\varepsilon = 10^{-5}$, $y_1 = 0.08$, $y_2 = 0.39$, and $N = 93$. The top frame indicates the function and node locations. The middle frame shows the node spacing, which corresponds to the widths of the intervals being integrated over. The bottom frame illustrates agreement of the absolute error estimate and actual absolute error and satisfaction of the prescribed tolerance. Node locations where no marker for the error estimate or actual error appears indicate estimates and actual errors that fall well below the vertical limits of the plot.*

the interval $[-1, 1]$. The top frame illustrates $f_2$ and the locations of the nodes required to achieve a tolerance of $\varepsilon = 10^{-5}$, while the middle frame shows the node spacing, corresponding to the widths of the subintervals being integrated over. Both the top and middle frames indicate that the algorithm is adaptively placing points with increased density where the function is changing rapidly (i.e., the derivative is larger). This is the expected behavior. The bottom frame then illustrates that the absolute error estimate and actual absolute error are in agreement and both meet the prescribed tolerance with $N = 93$ nodes on $[-1, 1]$.

Similarly, figure 4 shows a single example of adaptive differentiation of $f_2(x)$ on the interval $[-1, 1]$. Since the error estimate is utilized to approximate the *absolute* error in the derivative, more points are required to achieve a specific tolerance in comparison to adaptive quadrature. This is because the derivatives of both $f_1(x)$ and $f_2(x)$ are an order of magnitude larger than the maximum value of their integrals over $[-1, 1]$. In fact, the maximum of the derivative for any choice of $a$ can be as large as $2(3\sqrt{3a}/8)$ for $f_1$ and $2(\sqrt{2ae^{-1}})$ for $f_2$. For this reason, in this example the algorithm is applied to computing the derivative of $f_2(x)$ with $\varepsilon = 10^{-2}$ so that individual nodes are visually distinguishable except where the nodes are adaptively placed with the greatest density. Still, the top and middle frames indicate that the algorithm is again adaptively placing points with increased density where the derivative is large and the bottom frame illustrates that the error estimate and actual error are in agreement and meet the prescribed tolerance, now with $N = 3093$ nodes on $[-1, 1]$.

Second, figure 5 demonstrates the number of nodes, $N$, required to locally reach various prescribed tolerances, $\varepsilon \in [10^{-7}, 10^{-4}]$, for $a = 100$ and both $f = f_1$ (solid curves) and $f = f_2$ (dashed curves) for each value of $m = 1, 2, 3, 4$ and $\mu = 2, 3$. There

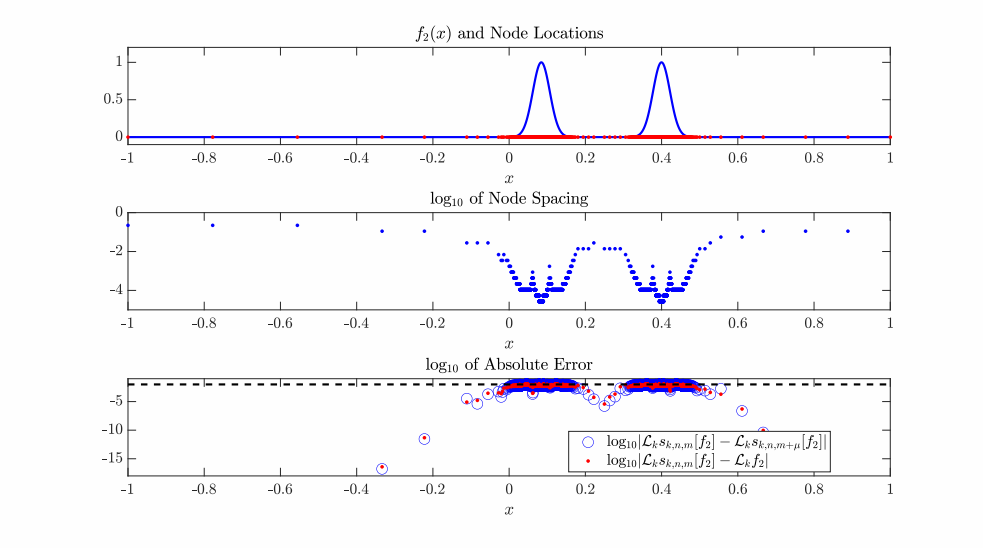


FIG. 4. *An example of approximation differentiation of $f_2$ for $x \in [-1, 1]$ with $a = 1000$, $m = 1$, $\mu = 2$, $\varepsilon = 10^{-2}$, $y_1 = 0.08$, $y_2 = 0.39$, and $N = 3093$. The top frame indicates the function and node locations. The middle frame shows the node spacing. The bottom frame illustrates agreement of the absolute error estimate and actual absolute error and satisfaction of the prescribed tolerance. Node locations where no marker for the error estimate or actual error appears indicate estimates and actual errors that fall well below the vertical limits of the plot.*

is no noticeable improvement, which would be indicated by fewer nodes being required, when increasing $\mu$ from 2 to 3, so that the discussion in section 4.2 indicates that $\mu = 2$ should be preferred for the reduced computational expense. Further computational experiments were performed with $\mu = 1$ and $\mu > 3$, all of which indicated that $\mu = 2$ is a reasonable choice.

Figure 5 also provides a comparison between the adaptive algorithm developed here applied to quadrature and the more familiar adaptive trapezoidal rule. The implementation of the adaptive trapezoidal rule mirrors the presentation in [1, Section 5.5]. The figure illustrates that even when the adaptive kernel method is of the same anticipated order as the adaptive trapezoidal rule (i.e., when $m = 1$) the method here requires fewer nodes to achieve the desired tolerance.

The impact of the radius of convergence of and the growth of the terms in the Taylor formula on the required number of nodes for each test function was also assessed for both numerical quadrature and differentiation. Both the radius of convergence and growth of the terms were controlled by varying $a$ (here $a \in [1, 1000]$) and the number of required nodes to achieve specified tolerances ($\varepsilon \in [10^{-7}, 10^{-4}]$ for quadrature and $\varepsilon \in [10^{-6}, 10^{-3}]$ for differentiation) was recorded. The contours in each frame of figures 6 and 7 illustrate log base 10 of the number of nodes $N$ required for each choice of $a$ and $\varepsilon$, with each frame corresponding to a different value of $m = 1, 2, 3, 4$. Due to the memory requirements for storing all diagnostic data used in the analyses presented here, the algorithm was forced to terminate when $N > 10^{5.5}$. This limit should not be necessary for more efficient implementations of the algorithm and here only impacted the results for numerical differentiation in the case of $m = 1$. The consequences of this limit are illustrated by the missing curves down and to the right of the curve indicating $N = 10^5$ in the top left frame of figure 7. As expected,

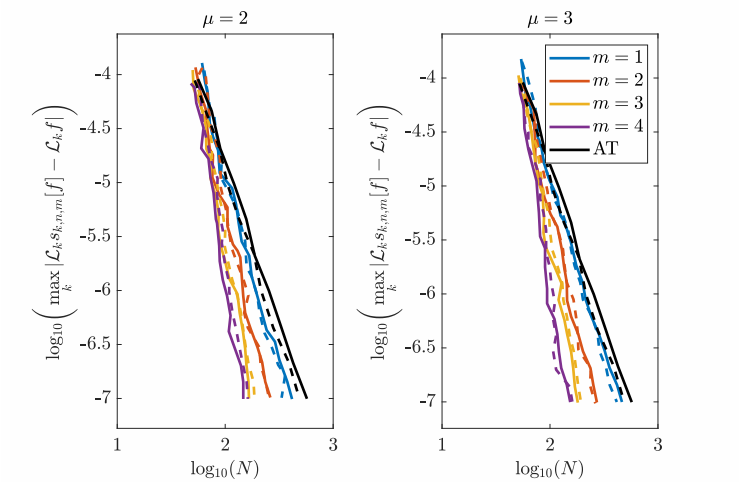


FIG. 5. *Log base 10 of the average maximum absolute error versus log base 10 of the average number of nodes required to achieve that error for approximate definite integration of $f_1$ (solid curves) and $f_2$ (dashed curves) for $x \in [-1, 1]$ with $a = 100$. The average is taken over 10 choices of $y_1$ and $y_2$ for each value of $\varepsilon$, corresponding closely to the error shown. The left frame corresponds to a choice of $\mu = 2$ while in the right frame $\mu = 3$. In both cases curves are shown for $m = 1, 2, 3, 4$ along with analogous results for an adaptive trapezoidal rule (AT in the legend) that utilizes the difference between Simpson's rule and trapezoidal rule as an error estimate.*

an increase in $a$, which leads to more localized features in both $f_1$ and $f_2$ and faster growth of their derivatives, necessitates an increase in $N$ to achieve the same tolerance $\varepsilon$. Consistent with remark 3.1, theorem 3.3 and figure 5 the number of nodes required for each value of $a$ and $\varepsilon$ decreases with increasing $m$. However, despite the finite and infinite radii of convergence of the Taylor formulas of $f_1$ and $f_2$, respectively, the algorithm performs similarly, demonstrating that it gracefully handles the growth in the terms of the series and finite radii of convergence.

**5.3. Examples in 2-Dimensions.** For $d = 2$ computational experiments were performed for both

$$\mathcal{L}f = \int_\Omega f(\mathbf{x})d\mathbf{x},\ \Omega = [-1, 1]^2 \left(\text{i.e., } \mathcal{L}_k f = \int_{\omega_k} f(\mathbf{x})d\mathbf{x}\right)$$

and

$$\mathcal{L}f = \nabla f,\ \mathbf{x} \in [-1, 1]^2 \left(\text{i.e., } \mathcal{L}_k f = \nabla f(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_{k,0}}\right).$$

Consider the choice of $a = 1000$, $N^{(0)} = 100$, $m = 4$, and $\mu = 2$. Figure 8 illustrates a single example of adaptive quadrature for evaluating the integral of $f_2(\mathbf{x})$ over the domain $[-1, 1]^2$ with $\varepsilon = 10^{-6}$. The top left frame illustrates the magnitude of $f_2$, while the top right frame shows the node spacing, represented by the area of the triangles $\omega_k$. The top right frame again indicates that the algorithm is adaptively placing points with increased density where the function is changing rapidly. The bottom frames illustrate that the absolute error estimate and actual absolute error are in agreement with $N = 2366$.

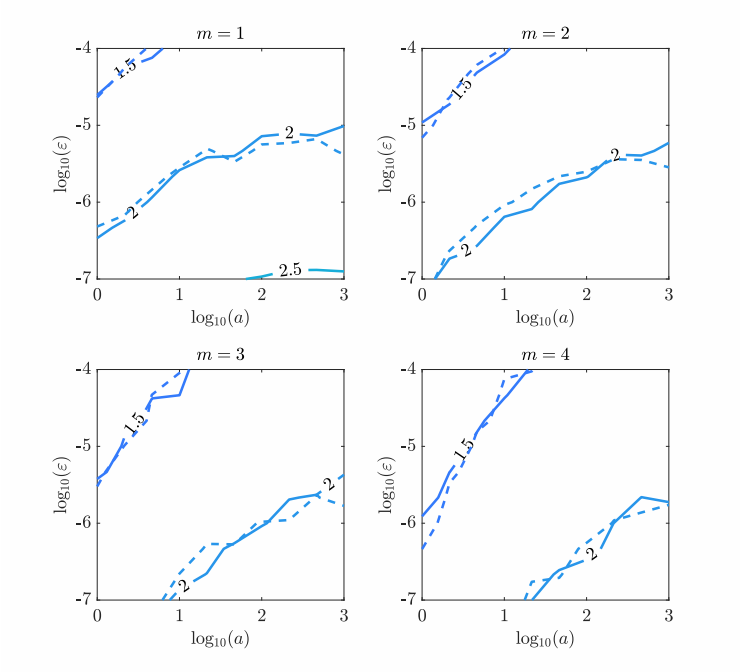


FIG. 6. *An illustration of the number of nodes required to achieve an estimated error tolerance of $\varepsilon$ for approximate definite integration of $f_1$ (solid curves) and $f_2$ (dashed curves) for $x \in [-1,1]$ with $a \in [1,1000]$. The contours indicate the log base 10 of the mean of the number of required nodes for ten random choices of $y_1$ and $y_2$ for each value of $a$ and $\varepsilon$. In all cases $\mu = 2$.*

Similarly, figure 9 shows a single example of adaptive differentiation of $f_2(\mathbf{x})$ on the domain $[-1,1]^2$. Since the derivative can again be large, in this example the algorithm is applied to computing the derivative of $f_2$ with $\varepsilon = 10^{-2}$. Still, the top right frame indicates that the algorithm is again adaptively placing points with increased density where the derivative is large and the bottom frames illustrate that the error estimate and actual error are in agreement and meet the prescribed tolerance, now with $N = 14852$ nodes on $[-1,1]^2$.

Again, the choice of $\mu = 2$ in these examples is motivated by computational experiments. Figure 10 is analogous to the presentation in figure 5, only now for approximate definite integration over $[-1,1]^2$. There is again no noticeable improvement when increasing $\mu$ from 2 to 3, and $\mu = 2$ is likewise preferred for the reduced computational expense. However, it is clear in these examples that the algorithm performs better for a function whose Taylor series has infinite radius of convergence.

*Remark* 5.1. In all of the computational examples presented here, no problems resulting from the conditioning of the problem or systems of linear equations were observed, even up to $m + \mu = 16$ and under refinement. This is in line with the observation in [3] (which follows from a result in [18]) that for the choice of $\varphi(r) = r^\rho$, with $\rho$ an odd positve integer, the high condition numbers of the systems of linear equations do not have the expected impact on the accuracy of the computed weights. This is likely not the case for other choices of the kernel, $\varphi$. In particular, use of several common choices for the kernel (e.g., Gaussian, multiquadric or inverse multiquadric) may necessitate the use of stable algorithms, some of which are detailed in [8, 10].

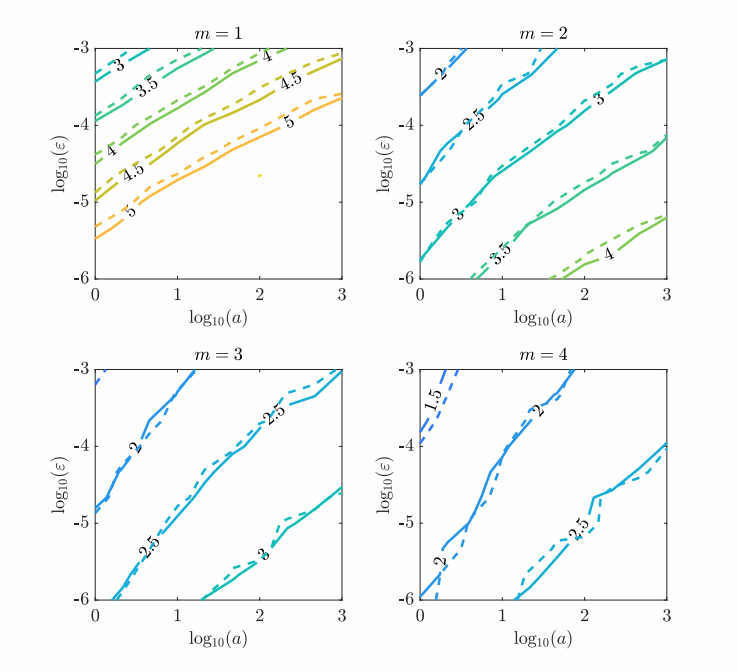


FIG. 7. *An illustration of the number of nodes required to achieve an estimated error tolerance of $\varepsilon$ for approximate differentiation of $f_1$ (solid curves) and $f_2$ (dashed curves) for $x \in [-1, 1]$ with $a \in [1, 1000]$. The contours indicate the log base 10 of the mean of the number of required nodes for ten random choices of $y_1$ and $y_2$ for each value of $a$ and $\varepsilon$. In all cases $\mu = 2$.*

**6. Conclusions.** This article presented a novel approach to constructing approximations to the action of linear operators that locally adapt the spacing of the discrete point set to achieve a prescribed error tolerance. The approach described here utilized kernel methods similar to RBF-FD to overcome the difficulties experienced with an analogous use of polynomial interpolation in the presence of scattered nodes, particularly when $d > 1$. Computational experiments have shown that the estimate and actual absolute error are in agreement and that the estimate can be successfully used to indicate where refinement of the discrete node set is required.

This study is not exhaustive. In particular, there is significant opportunity to explore node placement strategies where the error estimate indicates the need for refinement. Further, the choice to refine at every point where the estimate is computed that has a change in its nearest neighbors from one level to the next is likely unnecessary. Therefore, approaches to refining only where the error indicator is larger than the prescribed tolerance are an important avenue for investigation.

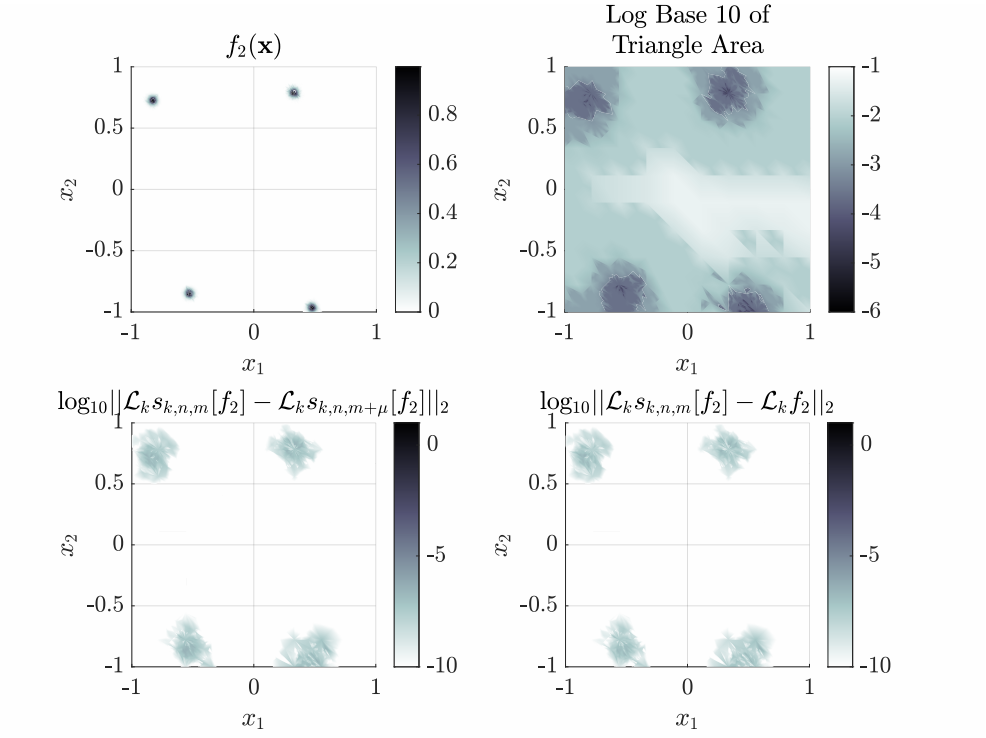


FIG. 8. *An example of adaptive approximate numerical quadrature of $f = f_2$ for $x \in [-1, 1]$ with $a = 1000$, $m = 4$, $\mu = 2$, $n = M_{d,m+\mu} = 28$, $\varepsilon = 10^{-6}$, $\mathbf{y}_1 = \begin{bmatrix} 0.32 & 0.78 \end{bmatrix}^T$, $\mathbf{y}_2 = \begin{bmatrix} 0.47 & -0.96 \end{bmatrix}^T$, $\mathbf{y}_3 = \begin{bmatrix} -0.82 & 0.72 \end{bmatrix}^T$, $\mathbf{y}_4 = \begin{bmatrix} -0.52 & -0.84 \end{bmatrix}^T$ and $N = 2366$. The top left frame illustrates the magnitude of the function. The top right frame shows the node spacing measured by the areas of the triangles $\omega_k$. The bottom frames illustrate agreement of the absolute error estimate and actual absolute error and satisfaction of the prescribed tolerance.*

or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

## REFERENCES

[1] K. E. Atkinson, *An Introduction to Numerical Analysis*, John Wiley & Sons, 2nd ed., 1989.
[2] V. Bayona, *An insight into RBF-FD approximations augmented with polynomials*, Comput. Math. Appl., 77 (2019), pp. 2337–2353, https://doi.org/10.1016/j.camwa.2018.12.029.
[3] V. Bayona, N. Flyer, B. Fornberg, and G. A. Barnett, *On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs*, J. Comput. Phys, 332 (2017), pp. 257–273.
[4] H. Cartan, *Differential Calculus*, Kershaw Publishing Company, London, 1971.
[5] O. Davydov and D. T. Oanh, *Adaptive meshless centres and rbf stencils for poisson equation*, J Comput Phys, 230 (2011), pp. 287–304, https://doi.org/https://doi.org/10.1016/j.jcp.2010.09.005, https://www.sciencedirect.com/science/article/pii/S0021999110004997.
[6] F. Dell'Accio, F. Di Tommaso, N. Siar, and M. Vianello, *Disc: an adaptive numerical differentiator by local polynomial interpolation on multivariate scattered data*, Dolomites Research Notes on Approximation, 15 (2022), pp. 81–91.
[7] T. Dobravec, B. Mavrič, and B. Šarler, *Acceleration of rbf-fd meshless phase-field modelling of dendritic solidification by space-time adaptive approach*, Comput Math Appl, 126

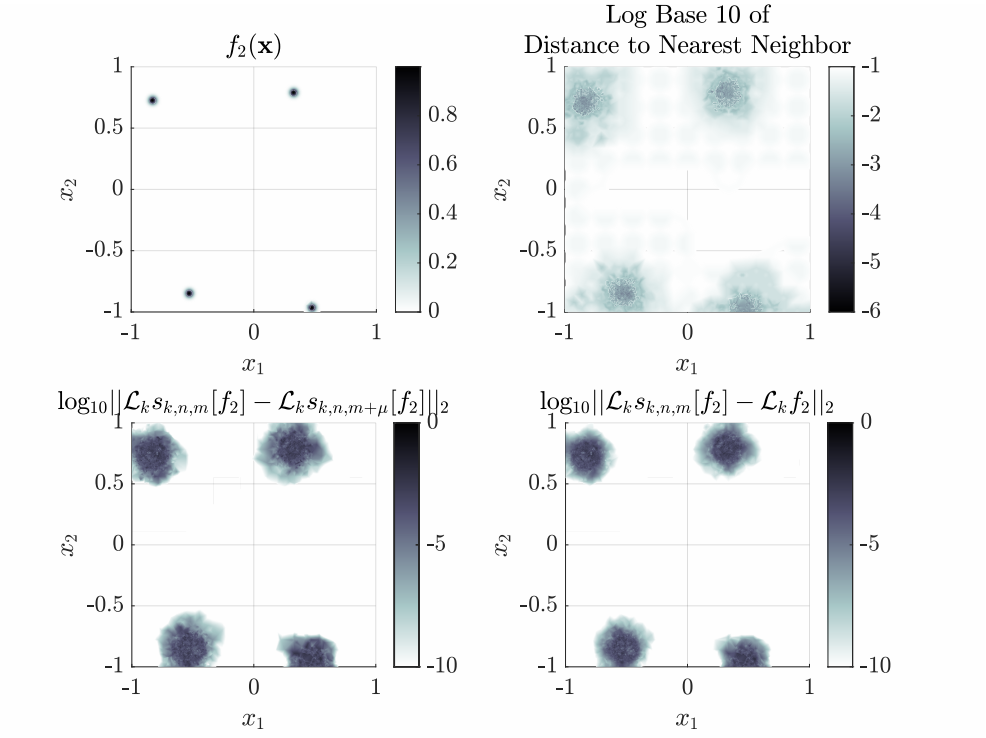


FIG. 9. *An example of approximation differentiation of $f = f_2$ for $\mathbf{x} \in [-1,1]^2$ with $a = 1000$, $m = 4$, $\mu = 2$, $n = M_{d,m+\mu} = 28$ $\varepsilon = 10^{-2}$, $\mathbf{y}_1 = \begin{bmatrix} 0.32 & 0.78 \end{bmatrix}^T$, $\mathbf{y}_2 = \begin{bmatrix} 0.47 & -0.96 \end{bmatrix}^T$, $\mathbf{y}_3 = \begin{bmatrix} -0.82 & 0.72 \end{bmatrix}^T$, $\mathbf{y}_4 = \begin{bmatrix} -0.52 & -0.84 \end{bmatrix}^T$, and $N = 14852$. The top left frame illustrates the magnitude of the function. The top right frame shows the node spacing measured by the distance of each node to its nearest neighbor. The bottom frames illustrate agreement of the absolute error estimate and actual absolute error and satisfaction of the prescribed tolerance.*

(2022), pp. 77–99, https://doi.org/https://doi.org/10.1016/j.camwa.2022.09.008, https://www.sciencedirect.com/science/article/pii/S0898122122003881.

[8] G. E. Fasshauer and M. J. McCourt, *Stable evaluation of gaussian radial basis function interpolants*, SIAM Journal on Scientific Computing, 34 (2012), pp. A737–A762, https://doi.org/10.1137/110824784.

[9] M. Fekete, *Über die Verteilung der Wurzeln bei gewissen algebraischen gleichungen mit ganzzahligen Koeffizienten*, Math. Zeit., 17 (1923), pp. 228–249.

[10] B. Fornberg and N. Flyer, *A primer on radial basis functions with applications to the geosciences*, SIAM, Philadelphia, U.S., 2015.

[11] E. Fuselier, T. Hangelbroek, F. J. Narcowich, J. D. Ward, and G. B. Wright, *Kernel based quadrature on spheres and other homogeneous spaces*, Numer. Math., 127 (2014), pp. 57–92.

[12] K. Gao, G. Mei, S. Cuomo, F. Piccialli, and N. Xu, *Arbf: Adaptive radial basis function interpolation algorithm for irregularly scattered point sets*, Soft Computing, (2020), https://doi.org/10.1007/s00500-020-05211-0.

[13] J. Glaubitz and J. A. Reeger, *Towards stability results for global radial basis function based cubature formulas*, BIT Numerical Mathematics, 63 (2023), https://doi.org/10.1007/s10543-023-00956-0.

[14] J. Gu and J. Jung, *Adaptive radial basis function methods for initial value problems*, J. Sci. Comput., 82 (2020).

[15] J. Gu and J. Jung, *Adaptive gaussian radial basis function methods for initial value problems: Construction and comparison with adaptive multiquadric radial basis function methods*, Journal of Computational and Applied Mathematics, 381 (2021), p. 113036, https://doi.

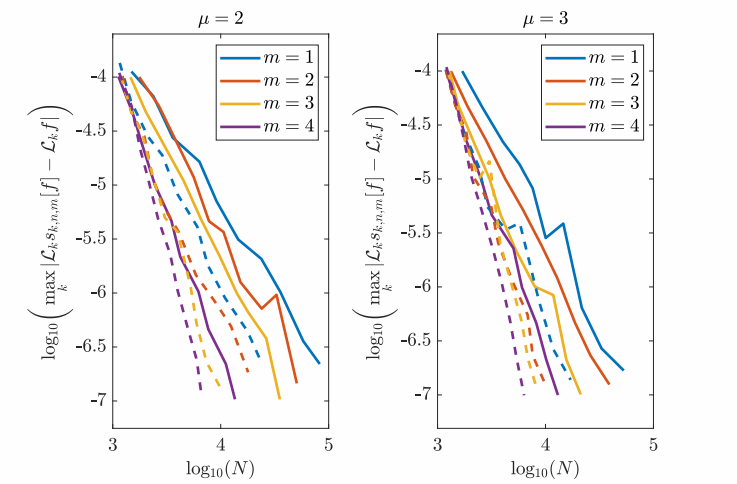


FIG. 10. *Log base 10 of the average maximum absolute error versus log base 10 of the average number of nodes required to achieve that error for approximate definite integration of $f_1$ (solid curves) and $f_2$ (dashed curves) for $\mathbf{x} \in [-1,1]^2$ with $a = 100$. The average is taken over 10 choices of $\mathbf{y}_1$, $\mathbf{y}_2$, $\mathbf{y}_3$ and $\mathbf{y}_4$ for each value of $\varepsilon$. Here $\varepsilon$ corresponds closely to the error shown. The left frame corresponds to a choice of $\mu = 2$ while in the right frame $\mu = 3$. In both cases curves are shown for $m = 1, 2, 3, 4$.*

org/https://doi.org/10.1016/j.cam.2020.113036, https://www.sciencedirect.com/science/article/pii/S0377042720303277.

[16] R. L. HARDY, *Multiquadric Equations of Topography and Other Irregular Surfaces*, J. Geophys. Res., 76 (1971), pp. 1905–1915.

[17] J. HUNTER AND B. NACHTERGAELE, *Applied Analysis*, World Scientific, 2001, https://books.google.com/books?id=oOYQVeHmNk4C.

[18] A. ISKE, *On the approximation order and numerical stability of local lagrange interpolation by polyharmonic splines*, in Modern Developments in Multivariate Approximation, W. Haussmann, K. Jetter, M. Reimer, and J. Stöckler, eds., Basel, 2003, Birkhäuser Basel, pp. 153–165.

[19] M. JANČIČ, F. STRNIŠA, AND G. KOSEC, *Implicit-explicit error indicator based on approximation order*, in 2022 7th International Conference on Smart and Sustainable Technologies (SpliTech), 2022, pp. 01–04, https://doi.org/10.23919/SpliTech55088.2022.9854342.

[20] E. J. KANSA, *Multiquadrics - A Scattered Data Approximation Scheme with Applications to Computational Fluid-Dynamics - I: Surface Appoximations and Partial Derivative Estimates*, Comput. Math. Appl., 19 (1990), pp. 127–145.

[21] G. KOSEC AND B. ŠARLER, *H-adaptive local radial basis function collocation meshless method*, CMC-Comput Mater Con, 26 (2011), pp. 227–254, https://doi.org/10.3970/cmc.2011.026.227, http://www.techscience.com/cmc/v26n3/27844.

[22] F. LEJA, *Sur certaines suites liées aux ensembles plans et leur application à la représentation conforme*, Ann. Polon. Math., 4 (1957), pp. 8–13.

[23] J. LI, S. ZHAI, Z. WENG, AND X. FENG, *H -adaptive rbf-fd method for the high-dimensional convection-diffusion equation*, Int Commun Heat Mass, 89 (2017), pp. 139–146, https://doi.org/https://doi.org/10.1016/j.icheatmasstransfer.2017.06.001, https://www.sciencedirect.com/science/article/pii/S0735193317301215.

[24] T. LIU AND R. PLATTE, *Node generation for RBF-FD methods by QR factorization*, Mathematics, 9 (2021), p. 1845.

[25] D. T. OANH, O. DAVYDOV, AND H. X. PHU, *Adaptive rbf-fd method for elliptic problems with point singularities in 2d*, Appl Math Comput, 313 (2017), pp. 474–497, https://doi.org/https://doi.org/10.1016/j.amc.2017.06.006, https://www.sciencedirect.com/science/article/pii/S0096300317304186.

[26] D. T. OANH AND N. M. TUONG, *An approach to adaptive refinement for the rbf-fd method for 2d elliptic equations*, Appl Numer Math, 178 (2022), pp. 123–154, https://doi.org/https://doi.org/10.1016/j.apnum.2022.03.015, https://www.sciencedirect.com/science/article/pii/

S0168927422000770.
[27] J. A. Reeger, *Approximate integrals over the volume of the ball*, J. Sci Comput, 83 (2020), https://doi.org/https://doi.org/10.1007/s10915-020-01231-y.
[28] J. A. Reeger, *Approximate integrals over bounded volumes with smooth boundaries*, J. Comput. Phys., 488 (2023), https://doi.org/10.1016/j.jcp.2023.112235.
[29] J. A. Reeger and B. Fornberg, *Numerical quadrature over the surface of a sphere*, Stud. Appl. Math., 137 (2016), pp. 174–188.
[30] J. A. Reeger and B. Fornberg, *Numerical quadrature over smooth surfaces with boundaries*, J. Comput. Phys., 355 (2018), pp. 176–190.
[31] J. A. Reeger, B. Fornberg, and M. L. Watts, *Numerical quadrature over smooth, closed surfaces*, P. Roy. Soc. Lon. A Mat., 472 (2016). doi: 10.1098/rspa.2016.0401.
[32] L. Reichel, *Newton interpolation at leja points*, BIT, 30 (1990), pp. 332–346.
[33] S. Sarra, *Adaptive radial basis function methods for time dependent partial differential equations*, Applied Numerical Mathematics, 54 (2005), pp. 79–94, https://doi.org/https://doi.org/10.1016/j.apnum.2004.07.004, https://www.sciencedirect.com/science/article/pii/S0168927404001199.
[34] J. Slak, *Adaptive RBF-FD method*, PhD thesis, University of Ljubljana, Slovenia, 2020.
[35] A. Sommariva and M. Vianello, *Meshless cubature by Green's formula*, Appl. Math. Comput., 183 (2006), pp. 1098–1107.
[36] B. Tóth and A. Düster, *h-Adaptive radial basis function finite difference method for linear elasticity problems*, Comput Mech, (2023), pp. 433–452.
[37] W. F. Trench, *Introduction to Real Analysis*, Pearson Education, New Jersey, 2003.
[38] A. Webb and S. Shannon, *Shape-adaptive radial basis functions*, IEEE Transactions on Neural Networks, 9 (1998), pp. 1155–1166, https://doi.org/10.1109/72.728359.
[39] H. Wendland, *Scattered Data Approximation*, vol. 17, Cambridge University Press, Cambridge, United Kingdom, 2005.
[40] Q. Zhang, Y. Zhao, and J. Levesley, *Adaptive radial basis function interpolation using an error indicator*, Numerical Algorithms, 76 (2017), https://doi.org/10.1007/s11075-017-0265-5.