Part I

The RBF-FD Method

The RBF-FD chapter needs: Related papers on RBF-FD specifically (i.e., the complete history; follow Flyer Fornberg book). o Clearly state what every paper in the RBF-FD group accomplished Weight method Similarly to spectral and pseudospectral (only collocation points allow optimizations) modes, RBF-FD and FD are related and share much of the same approach. Use RBF-FD for general node placement and high order accuracy. Use FD for optimized solution, faster solvers (Fourier decomposition, Band solver, etc). Apply weights to single node (figure from Paper 1) or form a DM Solve PDEs with explicit/implicit form (solvers) o Mention GMRES List of weight types o Projection operators Choosing epsilon Hyperviscosity stabilization

Chapter 1

Introduction to RBF-FD

While most of the literature surrounding RBFs for PDEs involves collocation, an alternative method does exist: RBF-generated Finite Differences (RBF-FD). RBF-FD is a hybrid of RBF scattered data interpolation and Finite Difference (FD) stencils.

The idea behind FD stencils is to express various derivative operators as a linear combination of known functional values in the neighborhood of a point where an approximation to the derivative operator is desired. Common approximations such as upwind differencing, center differencing, higher order approximations, and even spectral operators, are of this form.

A common approach to building such discrete operators is to form a local interpolant in a neighborhood of the target point, and simply differentiate it analytically. This is the approach taken in RBF-FD, which allows for stencils with irregular placement and number of nodes, and assigns their weights based on an RBF [57]. Such an approach leads to very simple implementations of time-advancement schemes, whether explicit or implicit. The solution at the new time step is simply some linear—if $\mathcal{L}$ is linear, nonlinear otherwise—combination of the unknown functional values (if implicit scheme) or known functional value (if explicit scheme).

Key challenges lie in the choice of grid, the choice of stencil, whether or not to change the support as a function of the stencil, how to guaranty the stability of the differentiation operator after discretization, etc.

The choice to study RBF-FD within this dissertation is motivated by two factors. First, RBF-FD represents one of the latest developments within the RBF community. The method was first introduced in 2000 [52]. Unfortunately, RBF-FD has yet to obtain the critical-mass following necessary for the method's use in large-scale scientific models. Our goal throughout the dissertation has been to scale RBF-FD to complex problems on high resolution meshes, and to lead the way for its adoption in high performance computational geophysics. Second, RBF-FD inherits many of the positive features from global and local collocation schemes, but sacrifices others for the sake of significantly reduced computational complexity and increased parallelism. Graphics Processing Units (GPUs), introduced in Chapter 2.3, are many-core accelerators capable of general purpose, embarrassingly parallel computations. GPUs represent the latest trend in high performance computing, where compute nodes are commonly supplemented by one or more accessory GPUs. Our effort leads the way for application of RBF-FD in an age when compute nodes with attached accelerator boards will be key to breaching the exa-scale barrier in computing [1].

Prior to considering optimizations of RBF-FD on one or more GPUs, it is prudent to dedicate significant attention to the method definition and related works.

1.1 Background

RBF-generated Finite Differences (RBF-FD) were first introduced by Tolstykh in 2000 [52], but it was the simultaneous, yet independent, efforts in [46], [51], [57] and [6] that gave the method its real start. The RBF-FD method (and the RBF-HFD, "Hermite" equivalent [59]) is similar in concept to classical finite-differences (FD), but differs in that the underlying differentiation weights are exact for RBFs rather than polynomials. The method contrasts with global RBF methods in the sense that it does not interpolate the differential operator of the PDE. Instead, the RBF-FD method applies the differential operator to translates of the RBF in a small stencil/neighborhood of nodes. Solving the resulting system provides a set of generalized FD weights representing the discrete differential operator defined at the stencil center.

RBF-FD share many advantages with global RBF methods, like the ability to function without an underlying mesh, easily extend to higher dimensions and afford large time steps; however spectral accuracy is lost. Other advantages of RBF-FD include lower computational complexity together with high-order accuracy (6th to 10th order accuracy is common). As in FD, increasing the stencil size, n, increases the accuracy of the approximation. While not a panacea for PDEs, the method is simple to code, easily extensible to higher dimensions, and powerful in its ability to avoid singularities introduced by the coordinate system that might impact other methods.

In some ways, RBF-FD and global RBF methods are plagued by the same difficulties. For example, as the number of nodes in the stencil increases, so too does the ill-conditioning of the linear systems to be solved. Similarly, the most accurate weights occur when $\epsilon \to 0$, but values in that regime beget additional ill-conditioning problems—a recurrence of the *Uncertainty Relation* [43]. One key difference in the multiple independent RBF-FD origins was that Wright [57] focused on bypassing ill-conditioning of RBF-FD and investigated its behavior in the limit as $\epsilon \to 0$ by means of the Contour-Padé algorithm.

Given N total nodes in the domain, N linear systems, each of size $n \times n$, are solved to calculate the differentiation weights. Since $n \ll N$, the RBF-FD preprocessing complexity is dominated by $O(N)$; significantly lower than the global RBF or RBF-PS methods ($O(N^3)$). The cost per time step is also $O(N)$.

RBF-FD have been successfully employed for a variety of problems including Hamilton-Jacobi equations [6], convection-diffusion problems [7, 48], incompressible Navier-Stokes equations [46, 8], transport on the sphere [25], and the shallow water equations [18]. Shu et al. [47] compared the RBF-FD method to Least Squares FD (LSFD) in context of 2D incompressible viscous cavity flow, and found that under similar conditions, the RBF-FD method was more accurate than LSFD, but the solution required more iterations of an iterative solver. RBF-FD was applied to Poisson's equation in [56]. Chandhini and Sanyasiraju [7] studied it in context of 1D and 2D, linear and non-linear, convection-diffusion equations, demonstrating solutions that are non-oscillatory for high Reynolds number, with improved accuracy over classical FD. An application to Hamilton-Jacobi problems [6], and 2D linear and non-linear PDEs including Navier-Stokes equations [46] have all been considered.

Author's Note: Expand description of each related work

1.2 The RBF-generated Finite Differences Method

The RBF-FD method is similar to classical Finite Differences in that the RBF-FD method allows derivatives of a function $u(x)$ to be approximated by weighted combinations of n function values in a small neighborhood (i.e., $n \ll N$) around a *center* node, x_c. That is:

$$\mathcal{L}u(x) \mid_{x=x_c} \approx \sum_{j=1}^{n} c_j u(x_j)$$

where $\mathcal{L}u$ again represents a differential quantity over $u(x)$ (e.g., $\mathcal{L} = \frac{d}{dx}$). We refer to the n nodes around x_c as a *stencil*. While not necessary, in practice one typically considers stencils to include the center, x_c, plus the $n-1$ nearest neighboring nodes. The definition of "nearest" depends the choice of distance metric; here, Euclidean distance is preferred.

Figure 1.1 provides two examples of RBF-FD stencils. First, Figure 1.1a illustrates a single stencil of size $n = 13$ in a domain of randomly distributed nodes. The stencil center, x_c, is represented by a green square, with the 12 neighboring nodes connected via red edges. The purple circle, the minimum covering circle for the stencil, demonstrates that the stencil contains only the 12 nearest neighbors of the center node. In Figure 1.1b, a larger RBF-FD stencil of size $n = 75$ on the unit sphere is shown as red and blue disks surrounding the center represented as a square. Green disks are nodes outside of the stencil. The radii and color of the red and blue disks represent the magnitude and alternating sign of coefficients, c_j, determined to calculate a derivative quantity at the stencil center.

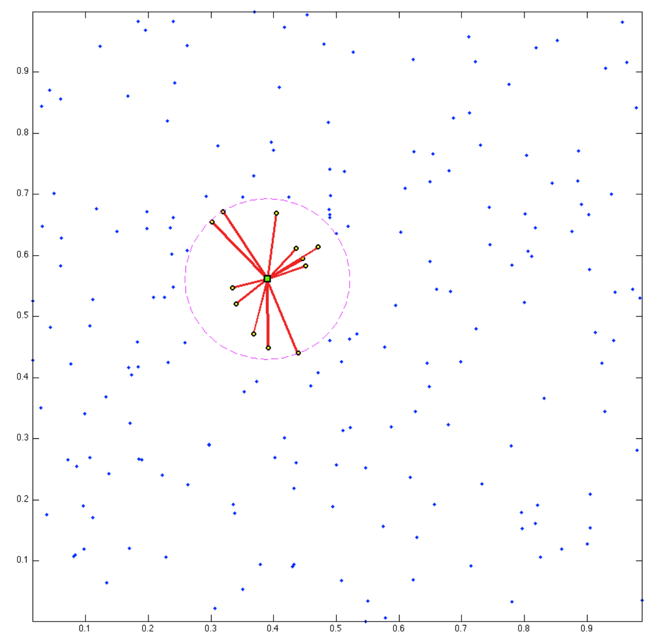
To approximate $\mathcal{L}u(x)$, one requires the *stencil weights* (coefficients), c_j. These are obtained by enforcing that they be exact within the space spanned by the RBFs centered at stencil nodes (i.e., $\phi_j(x)$). Various studies [59, 21, 25, 18] show that better accuracy is achieved when the interpolant can exactly reproduce a constant, p_0, such that

$$\mathcal{L}\phi_i(x) \mid_{x=x_c} = \sum_{j=1}^{n} c_j \phi_j(x_i) + c_{n+1} p_0 \qquad \text{for } i = 1, 2, ..., n$$

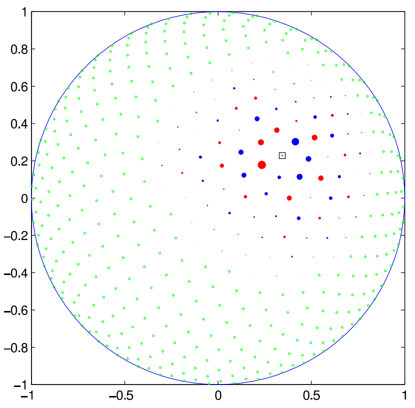
with $\mathcal{L}\phi_i$ provided by analytically applying the differential operator to the kernel function. Assuming $p_0 = 1$, the constraint $\sum_{i=1}^{n} c_i = \mathcal{L}p_0|_{x=x_c} = 0$ completes the system:

$$\begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_n(x_1) & 1 \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_n(x_2) & 1 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ \phi_1(x_n) & \phi_2(x_n) & \cdots & \phi_n(x_n) & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \\ c_{n+1} \end{pmatrix} = \begin{pmatrix} \mathcal{L}\phi_1(x) \mid_{x=x_c} \\ \mathcal{L}\phi_2(x) \mid_{x=x_c} \\ \vdots \\ \mathcal{L}\phi_n(x) \mid_{x=x_c} \\ 0 \end{pmatrix} \tag{1.1}$$

Solving Equation 1.4



(a) A 13 node RBF-FD stencil of randomly distributed nodes. The stencil centered at the green square contains the 12 nearest neighbors contained within the minimum covering circle drawn in purple.



(b) A 75 node RBF-FD stencil with blue (negative) and red (positive) differentiation weights to approximate advective operator at the square. Stencils weights indicated by scale of disk radii. (Image courtesy of Bengt Fornberg and Natasha Flyer)

Figure 1.1: Examples of stencils computable with RBF-FD

$$
\begin{aligned}
\mathcal{L}u(x_1) &\approx (\phi_s^{-1}\phi_{\mathcal{L}_s})^T\,(u)\\
&\approx \vec{\phi}^T_{\mathcal{L}_s}\phi_s^{-T}\,(u)
\end{aligned}
$$

Again, based on the choice of RBF, positive definiteness of the system is ensured by adding the same constraints as Equation **??**, but note differential quantities for P on the right hand side:

$$
\begin{pmatrix} \phi_s & P \\ P^T & 0 \end{pmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} \phi_{\mathcal{L}_s} \\ P_{\mathcal{L}} \end{bmatrix}. \tag{1.2}
$$

Equation **??** is strictly for the stencil centered at node x_1. To obtain weights for all stencils in the domain, a total of N such systems must be solved.

1.3 RBF-FD weights

Given a set of function values, $\{u(\mathbf{x}_j)\}_{j=1}^N$, on a set of N nodes $\{\mathbf{x}_j\}_{j=1}^N$, the operator $\mathcal{L}$ acting on $u(\mathbf{x})$ evaluated at $\mathbf{x}_j$, is approximated by a weighted combination of function values, $\{u(\mathbf{x}_i)\}_{i=1}^n$, in a small neighborhood of $\mathbf{x}_j$, where $n \ll N$ defines the size of the stencil.

$$\mathcal{L}u(\mathbf{x})\mid_{\mathbf{x}=\mathbf{x}_j} \approx \sum_{i=1}^{n} c_i u(\mathbf{x}_i) + w_{n+1}p_0 \tag{1.3}$$

The RBF-FD weights, c_i, are found by enforcing that they are exact within the space spanned by the RBFs $\phi_i(\epsilon r) = \phi(\epsilon\,||\mathbf{x}-\mathbf{x}_i||)$, centered at the nodes $\{\mathbf{x}_i\}_{i=1}^n$, with $r = ||\mathbf{x}-\mathbf{x}_i||$ being the distance between where the RBF is centered and where it is evaluated as measured in the standard Euclidean 2-norm. Various studies show [59, 21, 25, 18] that better accuracy is achieved when the interpolant can exactly reproduce a constant, p_0. Assuming $p_0 = 1$, the constraint $\sum_{i=1}^n c_i = \mathcal{L}1|_{\mathbf{x}=\mathbf{x}_j} = 0$ completes the system:

$$\begin{pmatrix} \phi(\epsilon\,||\mathbf{x}_1-\mathbf{x}_1||) & \phi(\epsilon\,||\mathbf{x}_1-\mathbf{x}_2|| & \cdots & \phi(\epsilon\,||\mathbf{x}_1-\mathbf{x}_n||) & 1 \\ \phi(\epsilon\,||\mathbf{x}_2-\mathbf{x}_1||) & \phi(\epsilon\,||\mathbf{x}_2-\mathbf{x}_2|| & \cdots & \phi(\epsilon\,||\mathbf{x}_2-\mathbf{x}_n||) & 1 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ \phi(\epsilon\,||\mathbf{x}_n-\mathbf{x}_1||) & \phi(\epsilon\,||\mathbf{x}_n-\mathbf{x}_2|| & \cdots & \phi(\epsilon\,||\mathbf{x}_n-\mathbf{x}_n||) & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{pmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ w_{n+1} \end{bmatrix} = \begin{bmatrix} \mathcal{L}\phi(\epsilon\,||\mathbf{x}-\mathbf{x}_1||)|_{\mathbf{x}=\mathbf{x}_j} \\ \mathcal{L}\phi(\epsilon\,||\mathbf{x}-\mathbf{x}_2||)|_{\mathbf{x}=\mathbf{x}_j} \\ \vdots \\ \mathcal{L}\phi(\epsilon\,||\mathbf{x}-\mathbf{x}_n||)|_{\mathbf{x}=\mathbf{x}_j} \\ 0 \end{bmatrix}, \tag{1.4}$$

where w_{n+1} is ignored after the matrix in (1.4) is inverted. This $n\times n$ system solve is repeated for each stencil center $\mathbf{x}_j$, $j = 1...N$, to form the N rows of the DM with n non-zeros ($n \ll N$) per row. As an example, if $\mathcal{L}$ is the identity operator, then the above procedure leads to RBF-FD interpolation. If $\mathcal{L} = \frac{\partial}{\partial x}$, one obtains the DM that approximates the first derivative in x. In the context of time-dependent PDEs, the stencil weights remain constant for all time-steps when the nodes are stationary. Therefore, the calculation of the differentiation weights is performed once in a single preprocessing step of O(n^3N) FLOPs. Improved efficiency is achieved by processing multiple right hand sides in one pass, calculating the weights corresponding to all required derivative quantities (i.e., $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$, ∇^2, etc.).

For each of the N small system solves of Equation (1.4), the n nearest neighbors to $\mathbf{x}_j$ need to be located. This can be done efficiently using neighbor query algorithms or spatial partitioning data-structures such as Locality Sensitive Hashing (LSH) and kD-Tree. Different query algorithms often have a profound impact on the DM structure and memory access patterns. We choose a Raster (ijk) ordering LSH algorithm [**?**] leading to the matrix structure in Figures **??** and **??**. While querying neighbors for each stencil is an embarrassingly parallel operation, the node sets used here are stationary and require stencil generation only once. Efficiency and parallelism for this task has little impact on the overall run-time of tests, which is dominated by the time-stepping. We preprocess node sets and generate stencils serially, then load stencils and nodes from disk at run-time. In contrast to the RBF-FD view of a static grid, Lagrangian/particle based PDE algorithms promote efficient parallel variants of LSH in order to accelerate querying neighbors at each time-step [42, 29].

Author's Note: Use of the 1's constraint is highly recommended. It keeps weights within magnitude 100, whereas not providing the constraint results in much higher magnitudes in the thousands or tens of thousands. Using first order monomials does not add benefit. Include examples of weights in 2D with 5 to 10 nodes.

1.4 Weight Operators

Throughout the development of our parallel code we have verified code correctness through the solution of a variety of PDEs. Here we provide a list of operators we have tested and their corresponding equations for the RHS of Equation 1.4 necessary to compute RBF-FD weights.

The standard first derivatives $\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}$ are produced by the chain rule

$$\frac{\partial \phi}{\partial x} = \frac{dr}{dx}\frac{d\phi}{dr} = \frac{(x - x_j)}{r}\frac{d\phi}{dr} \tag{1.5}$$

$$\frac{\partial \phi}{\partial y} = \frac{dr}{dy}\frac{d\phi}{dr} = \frac{(y - y_j)}{r}\frac{d\phi}{dr} \tag{1.6}$$

$$\frac{\partial \phi}{\partial z} = \frac{dr}{dz}\frac{d\phi}{dr} = \frac{(z - z_j)}{r}\frac{d\phi}{dr} \tag{1.7}$$

where $\frac{\partial \phi}{\partial r}$ for the Gaussian RBFs is given by:

1.4.1 Laplacian (∇^2)

2D:

3D:

1.4.2 Laplace-Beltrami (Δ_S) on the Sphere

The ∇^2 operator can be represented in spherical polar coordinates for $\mathbb{R}^3$ as:

$$\nabla^2 = \underbrace{\frac{1}{r}\frac{\partial}{\partial r}\left(r^2 \frac{\partial}{\partial r}\right)}_{\text{radial}} + \underbrace{\frac{1}{r^2}}_{\text{angular}}\Delta_S, \tag{1.8}$$

where Δ_S is the Laplace-Beltrami operator—i.e., the Laplacian operator constrained to the surface of the sphere. This form nicely illustrates the separation of components into radial and angular terms.

In the case of PDEs solved on the unit sphere, there is no radial term, so we have:

$$\nabla^2 \equiv \Delta_S. \tag{1.9}$$

Although this originated in the spherical coordinate system, [58] introduced the following Laplaci-Beltrami operator for the surface of the sphere:

$$\Delta_S = \frac{1}{4}\left[\left(4 - r^2\right)\frac{\partial^2}{\partial r^2} + \frac{4 - 3r^2}{r}\frac{\partial}{\partial r}\right], \tag{1.10}$$

where r is the Euclidean distance between nodes of an RBF-FD stencil and is independent of our choice of coordinate system.

1.4.3 Constrained Gradient ($P_x \cdot \boldsymbol{\nabla}$) on the Sphere

Additionally following [20, 18], the gradient operator must also be constrained to the sphere with this projection matrix:

$$P = I - \mathbf{x}\mathbf{x}^T = \begin{pmatrix} (1-x_1^2) & -x_1x_2 & -x_1x_3 \\ -x_1x_2 & (1-x_2^2) & -x_2x_3 \\ -x_1x_3 & -x_2x_3 & (1-x_3^2) \end{pmatrix} = \begin{pmatrix} P_{x_1} \\ P_{x_2} \\ P_{x_3} \end{pmatrix} \tag{1.11}$$

where $\mathbf{x}$ is the unit normal at the stencil center.

The direct method of computing RBF-FD weights for the projected gradient for $\mathbf{P} \cdot \nabla$ is presented in [20]. When solving for the weights, we apply the projection on the right hand side of our small linear system. We let $\mathbf{x} = \big(x_1, x_2, x_3\big)$ be the stencil center, and $\mathbf{x}_k = \big(x_{1,k}, x_{2,k}, x_{3,k}\big)$ indicate an RBF-FD stencil node.

Using the chain rule, and assumption that

$$r(\mathbf{x}_k - \mathbf{x}) = ||\mathbf{x}_k - \mathbf{x}|| = \sqrt{(x_{1,k}-x_1)^2 + (x_{2,k}-x_2)^2 + (x_{3,k}-x_3)^2},$$

we obtain the unprojected gradient of ϕ as

$$\nabla\phi(r(\mathbf{x}_k - \mathbf{x})) = \frac{\partial r}{\partial \mathbf{x}} \frac{\partial \phi(r(\mathbf{x}_k - \mathbf{x}))}{\partial r} = -(\mathbf{x}_k - \mathbf{x}) \frac{1}{r(\mathbf{x}_k - \mathbf{x})} \frac{\partial \phi(r(\mathbf{x}_k - \mathbf{x}))}{\partial r}$$

.

Applying the projection matrix gives

$$\begin{aligned} \mathbf{P}\nabla\phi(r(\mathbf{x}_k - \mathbf{x})) &= -(\mathbf{P}\cdot\mathbf{x}_k - \mathbf{P}\cdot\mathbf{x}) \frac{1}{r(\mathbf{x}_k - \mathbf{x})} \frac{\partial \phi(r(\mathbf{x}_k - \mathbf{x}))}{\partial r} \\ &= -(\mathbf{P}\cdot\mathbf{x}_k - 0) \frac{1}{r(\mathbf{x}_k - \mathbf{x})} \frac{\partial \phi(r(\mathbf{x}_k - \mathbf{x}))}{\partial r} \\ &= -(I - \mathbf{x}\mathbf{x}^T)(\mathbf{x}_k) \frac{1}{r(\mathbf{x}_k - \mathbf{x})} \frac{\partial \phi(r(\mathbf{x}_k - \mathbf{x}))}{\partial r} \\ &= \begin{pmatrix} x\mathbf{x}^T\mathbf{x}_k - x_k \\ y\mathbf{x}^T\mathbf{x}_k - y_k \\ z\mathbf{x}^T\mathbf{x}_k - z_k \end{pmatrix} \frac{1}{r(\mathbf{x}_k - \mathbf{x})} \frac{\partial \phi(r(\mathbf{x}_k - \mathbf{x}))}{\partial r} \end{aligned}$$

Thus, we directly compute the weights for $P_x \cdot \boldsymbol{\nabla}$ using these three RHS in Equation 1.4:

$$P\frac{\partial}{\partial x_1} = (x_1\mathbf{x}^T\mathbf{x}_k - x_{1,k}) \frac{1}{r(\mathbf{x}_k - \mathbf{x})} \frac{\partial \phi(r(\mathbf{x}_k - \mathbf{x}))}{\partial r}|_{\mathbf{x}=\mathbf{x}_j} \tag{1.12}$$

$$P\frac{\partial}{\partial x_2} = (x_2\mathbf{x}^T\mathbf{x}_k - x_{2,k}) \frac{1}{r(\mathbf{x}_k - \mathbf{x})} \frac{\partial \phi(r(\mathbf{x}_k - \mathbf{x}))}{\partial r}|_{\mathbf{x}=\mathbf{x}_j} \tag{1.13}$$

$$P\frac{\partial}{\partial x_3} = (x_3\mathbf{x}^T\mathbf{x}_k - x_{3,k}) \frac{1}{r(\mathbf{x}_k - \mathbf{x})} \frac{\partial \phi(r(\mathbf{x}_k - \mathbf{x}))}{\partial r}|_{\mathbf{x}=\mathbf{x}_j} \tag{1.14}$$

Alternatively, assuming weights for operators $\boldsymbol{\nabla} = \frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \frac{\partial}{\partial x_3}$ are already computed, the projected operator could be constructed via weighting the unprojected gradient components. For example:

Author's Note: Show accuracy of derivative approximation when using projected operator vs linear combinations of dx,dy,dz operators

1.5 Stabilization: Hyperviscosity

For RBF-FD, differentiation matrices encode convective operators of the form

$$D = \alpha \frac{\partial}{\partial \lambda} + \beta \frac{\partial}{\partial \theta} \tag{1.15}$$

where α and β are a function of the fluid velocity. The convective operator, discretized through RBF-FD, has eigenvalues in the right half-plane causing the method to be unstable [25, 18]. Stabilization of the RBF-FD method is achieved through the application of a hyperviscosity filter to Equation (1.15) [25]. By using Gaussian RBFs, $\phi(r) = e^{-(\epsilon r)^2}$, the hyperviscosity (a high order Laplacian operator) simplifies to

$$\Delta^k \phi(r) = \epsilon^{2k} p_k(r) \phi(r) \tag{1.16}$$

where k is the order of the Laplacian and $p_k(r)$ are multiples of generalized Laguerre polynomials that are generated recursively (see [25]: Section 3.2). We assume a 2D Laplacian operator when working on the surface of the sphere since a local stencil can be viewed as lying on a plane.

In the case of parabolic and hyperbolic PDEs, hyperviscosity is added as a filter to the right hand side of the evaluation. For example, at the continuous level, the equation solved takes the form

$$\frac{\partial u}{\partial t} = -Du + Hu, \tag{1.17}$$

where D is the PDE operator, and H is the hyperviscosity filter operator. Applying hyperviscosity shifts all the eigenvalues of D to the left half of the complex plane. This shift is controlled by k, the order of the Laplacian, and a scaling parameter γ_c, defined by

$$H = \gamma \Delta^k = \gamma_c N^{-k} \Delta^k.$$

Given a choice of ϵ (see Section **??**), it was found experimentally that $\gamma = \gamma_c N^{-k}$ provides stability and good accuracy for all values of N considered here. It also ensures that the viscosity vanishes as $N \to \infty$ [18]. In general, the larger the stencil size, the higher the order of the Laplacian. This is attributed to the fact that, for convective operators, larger stencils treat a wider range of modes accurately. As a result, the hyperviscosity operator should preserve as much of that range as possible. The parameter γ_c must also be chosen with care and its sign depends on k (for k even, γ_c will be negative and for k odd, it will be positive). If γ_c is too large, the eigenvalues move outside the stability domain of our time-stepping scheme and/or eigenvalues corresponding to lower physical modes are not left intact, reducing the accuracy of our approximation. If γ_c is too small, eigenvalues remain in the right half-plane [25, 18].

Chapter 2

TEMP

2.0.1 Parallel RBF Implementations

Author's Note: Related work for start of Parallel/GPU chapter Parallel implementations of RBF methods currently rely on parallel domain decomposition. Depending on the implementation, domain decomposition not only accelerates solution procedures, but can decrease the ill-conditioning that plague all global RBF methods [10]. The ill-conditioning is reduced if each domain is treated as a separate RBF domain, and the boundary update is treated separately. Domain decomposition methods for RBFs were introduced by Beatson et al. [2] in the year 2000 as a way to increase problem sizes into the millions of nodes.

Divo and Kassab [10] used a domain decomposition method with artificial subdomain boundaries for their implementation of a local collocation method [10]. Subdomains are processed independently. The derivative values at artificial boundary points are averaged to maintain global consistency of physical values. Their implementation was designed for a 36 node cluster, but benchmarks and scalability tests are not provided.

Kosec and Šarler [36] used OpenMP to parallelize coupled heat transfer and fluid flow problems on a single workstation. Their test cases used local collocation, explicit time-stepping and Neumann boundary conditions. A speedup factor of 1.85x over serial execution was achieved by executing on two CPU cores; no results from scaling tests were provided.

Stevens et al. [50] mention a parallel implementation under development, but no document is available yet.

Additional RBF implementations are discussed at the end of this chapter in the context of parallel co-processing with the GPU.

In this work, we will add to the above experiences, with the added twist of incorporating an implementation on the GPU (see later chapters).

2.1 Fragments (integrate above)

Stabilization of the RBF-FD method is achieved through the application of a hyperviscosity filter [?]. By assuming the use of Gaussian RBFs, $\phi(r) = e^{-(\epsilon r)^2}$, the hyperviscosity operator simplifies to

$$\Delta^k \phi(r) = \epsilon^{2k} p_k(r) \phi(r). \tag{2.1}$$

The multiples of generalized Laguerre polynomials, $p_k(r)$, are obtained through the following recursive relation:

$$\begin{cases} p_0(r) & = 1, \\ p_1(r) & = 4(\epsilon r)^2 - 2d, \\ p_k(r) & = 4((\epsilon r)^2 - 2(k-1) - \frac{d}{2})p_{k-1}(r) - 8(k-1)(2(k-1) - 2 + d)p_{k-2}(r), \qquad k = 2, 3, ... \end{cases}$$

where d is the dimension of the problem. We assume $d = 2$ below when working on the surface of the sphere.

Many algorithms exist to query the k-nearest neighbors (equivalently all nodes in the minimum/smallest enclosing circle). Some algorithms overlay a grid similar to Locality Sensitive Hashing and query such as... [**?**].

Conditioning

Conditioning is defined as:

With condition number estimates we can choose a proper support parameter for uniformly distributed node sets.

Alternative algorithms exist for solving for RBF-FD weights when the systems are overly ill-conditioned. Currently, we have an unpublished algorithm ContourSVD available to play with (demonstrate accuracy improvements).

Node Placement – Centroidal Voronoi Tessellation

We make the assumption for now that we use regularly distributed nodes. Centroidal Voronoi tessellations provide reasonably good distributions for solutions. Examples (heat ellipse, ellipsoid, sphere, square cavity).

The motivation behind RBF-FD is generality/functionality in the numerical method. Scattered nodes are supported. Distribution requires proper choice of support, and tight nodes result in increased conditioning

2.2 Approximate Nearest Neighbor (ANN) Query

RBF methods are traditionally described as general and meshless in that they apply to unstructured clouds of points in arbitrary dimensions. However, although the term meshless implies a method capable of operating with no node connectivity, all numerical methods—meshless RBF methods included—connect nodes in the domain. For example, the "meshless" global RBF method connects every node in the domain to all other nodes. Compact support or local RBF methods like RBF-FD limit connections to nodes that lie within a predetermined radius.

The connections between nodes form a directed adjacency graph with edges that dictate the paths along which data/phenomena can travel. For example, a plus shaped stencil of five points with a center node and four neighboring nodes allows values to propagate north, south, east and west; not northeast, southeast, etc.

They are robust and function on scattered point clouds. RBF-FD in particular requires stencils to be generated from n nearest neighbors to a stencil center. The cost of these neighbor queries can vary greatly depending on the choice of algorithm or data-structure used to make the query.

For example, in general brute force is inefficient The author of [13] queries n nearest neighbors for a compact-support RBF partition of unity example with a k-D tree. In [18, 25] a k-D Tree is leveraged for all neighbor queries for RBF-FD.

In our work in [3] an alternative to k-D tree was leveraged, based loosely on Locality Sensitive Hashing.

2.2.1 k-D Tree

Most of the RBF community leverages the k-D tree, due to its low computational complexity for querying neighbors and its wide availability as standalone software in the public domain (e.g., matlab central has a few implementations for download, and the MATLAB Statistics Toolbox includes an efficient k-D Tree).

The complexity of assembling he tree is

The Matlab central k-D Tree is MEX compiled and efficient. We integrated the standalone C++ code into our library.

While the k-D Tree functions well for queries, its downfall is a large cost in preprocessing to build the tree. For moving nodes, such as in Lagrangian schemes, this cost is prohibitively high. In an attempt to reduce the cost, lagrangian schemes introduced approximate nearest neighbor queries based on

Approximate nearest neighbors will be nearly balanced. We observe that RBF-FD functions as well on stencils of true nearest neighbors as it does on approximate nearest neighbors.

2.3 Distributed/Parallel RBF Methods

RBFs on GPU work: [44, 45] (global), [62] (compact)

Part II

Appendices

The following appendices are included to illuminate subtleties of the RBF-FD method. The first discusses the method's ability to avoid pole singularities when applied to solid body transport on the sphere. The second considers the difference between directly computing weights for differentiation operators versus leveraging linear combinations of weights to indirectly construct the same operators.

Appendix A

Avoiding Pole Singularities with RBF-FD

This content follows [19, 20].

Within the test cases of this dissertation, we solve convective PDEs on the unit sphere with the form:

$$\frac{\partial h}{\partial t} = \mathbf{u} \cdot \nabla h$$

where $\mathbf{u}$ is velocity. For example, the cosine bell advection has this particular form:

$$\frac{\partial h}{\partial t} = \frac{u}{\cos\theta}\frac{\partial h}{\partial \lambda} + v\frac{\partial h}{\partial \theta} \tag{A.1}$$

in the spherical coordinate system defined by

$$\begin{aligned} x &= \cos\theta\cos\lambda \\ y &= \cos\theta\sin\lambda \\ z &= \sin\theta \end{aligned}$$

where $\theta \in (-\frac{\pi}{2}, \frac{\pi}{2})$ is the elevation angle and $\lambda \in (-\pi, \pi)$ is the azimuthal angle. Observe that as $\theta \to \pm\frac{\pi}{2}$, the $\frac{1}{\cos\theta}$ term goes to infinity as a discontinuity.

One of the many selling points for RBF-FD and other RBF methods is their ability analytically avoid pole singularities, which arise from the choice of coordinate system and not from the methods themselves. Since RBFs are inherently based on Euclidean distance between nodes, and not geodesic distance, it is said that they do not "feel" the effects of the geometry or recognize singularities naturally inherent in the coordinate system [19]. Here we demonstrate how pole singularities are analytically avoided with RBF-FD for cosine bell advection.

Let $r = ||\mathbf{x} - \mathbf{x}_j||$ be the Euclidean distance which is invariant of the coordinate system. In Cartesian coordinates, we have

$$r = \sqrt{(x - x_j)^2 + (y - y_j)^2 + (z - z_j)^2}.$$

In spherical coordinates we have:

$$r = \sqrt{2(1 - \cos\theta\cos\theta_j\cos(\lambda - \lambda_j) - \sin\theta\sin\theta_j)}.$$

The RBF-FD operators for $\frac{d}{d\lambda}, \frac{d}{d\theta}$ are discretized with the chain rule:

$$\frac{d\phi_j(r)}{d\lambda} = \frac{dr}{d\lambda}\frac{d\phi_j(r)}{dr} = \frac{\cos\theta\cos\theta_j\sin(\lambda-\lambda_j)}{r}\frac{d\phi_j(r)}{dr}, \tag{A.2}$$

$$\frac{d\phi_j(r)}{d\theta} = \frac{dr}{d\theta}\frac{d\phi_j(r)}{dr} = \frac{\sin\theta\cos\theta_j\cos(\lambda-\lambda_j)-\cos\theta\sin\theta_j}{r}\frac{d\phi}{dr}, \tag{A.3}$$

where $\phi_j(r)$ is the RBF centered at $\mathbf{x}_j$.

Plugging A.2 and A.3 into A.1, produces the following explicit form:

$$\frac{dh}{dt} = u(\cos\theta_j\sin(\lambda-\lambda_j)\frac{1}{r}\frac{d\phi_j}{dr}) + v(\sin\theta\cos\theta_j\cos(\lambda-\lambda_j)-\cos\theta\sin\theta_j\frac{1}{r}\frac{d\phi}{dr})$$

where $\cos\theta$ from A.2 analytically cancels with the $\frac{1}{\cos\theta}$ in A.1.

Then, formally, one would assemble differentiation matrices containing weights for the following operators:

$$\mathbf{D}_\lambda = \cos\theta_j\sin(\lambda-\lambda_j)\frac{1}{r}\frac{d\phi_j}{dr}, \tag{A.4}$$

$$\mathbf{D}_\theta = \sin\theta\cos\theta_j\cos(\lambda-\lambda_j)-\cos\theta\sin\theta_j\frac{1}{r}\frac{d\phi}{dr}, \tag{A.5}$$

and solve the explicit method of lines problem:

$$\frac{dh}{dt} = u\mathbf{D}_\lambda h + v\mathbf{D}_\theta h$$

where now the system is completely free of singularities at the poles [20].

We note that the expression $\cos(\frac{\pi}{2})$ evaluates on some systems to a very small number rather than zero (e.g., $6.1(10^{-17})$ on the Keeneland system with the GNU gcc compiler). The small value in turn allows $\frac{1}{\cos\theta}$ to evaluate to a large value (e.g., $1.6(10^{16})$) rather than "inf" or "NaN". A large value allows the cosine terms to cancel in double precision, whereas an "inf" or "NaN" would corrupt the numerics. Rather than avoid placing nodes at the poles, or assuming the machine will numerically cancel the singularities, it is preferred to use operators A.4, A.5 on the RHS of Equation 1.4 to compute RBF-FD weights.

Appendix B

Projected Weights on the Sphere

It is shown in [20, 18] that a projection operator

$$\mathbf{P} = \mathbf{I} - \mathbf{x}\mathbf{x}^T = \begin{bmatrix} (1-x^2) & -xy & -xz \\ -xy & (1-y^2) & -yz \\ -xz & -yz & (1-z^2) \end{bmatrix} = \begin{bmatrix} \mathbf{p}_x^T \\ \mathbf{p}_y^T \\ \mathbf{p}_z^T \end{bmatrix}$$

where $\mathbf{p}_x^T$ represents the projection operator in the x direction.

From [20], the projected RBF gradient operator is:

$$\begin{aligned} \mathbf{P} \cdot \nabla \phi_k(r(\mathbf{x})) &= \mathbf{P} \cdot \frac{(\mathbf{x} - \mathbf{x}_k)}{r(\mathbf{x})} \frac{d\phi_k(r(\mathbf{x}))}{dr(\mathbf{x})} \\ &= -\mathbf{P} \cdot \mathbf{x}_k \frac{1}{r(\mathbf{x})} \frac{d\phi_k(r(\mathbf{x}))}{dr(\mathbf{x})} \\ &= \begin{bmatrix} x\mathbf{x}^T\mathbf{x}_k - x_k \\ y\mathbf{x}^T\mathbf{x}_k - y_k \\ z\mathbf{x}^T\mathbf{x}_k - z_k \end{bmatrix} \frac{1}{r(\mathbf{x})} \frac{d\phi(r(\mathbf{x}))}{dr}. \end{aligned} \tag{B.1}$$

The operator $\mathbf{I} - \mathbf{x}\mathbf{x}^T$ for $\mathbf{x} = (x, y, z)$ projects a vector onto the plane tangent to the unit sphere at (x, y, z). Therefore, Equation B.1 gives the projection of the gradient operator at $\mathbf{x}_k$ onto the plane tangent to $\mathbf{x}$.

B.1 Direct Weights

Following [18], B.1 takes on the following when adapted to RBF-FD:

$$[\mathbf{p}_x \cdot \nabla f(\mathbf{x})]|_{\mathbf{x}=\mathbf{x}_c} = \sum_{k=1}^{n} c_k \underbrace{\left[x_c\mathbf{x}_c^T\mathbf{x}_k - x_k\right] \frac{1}{r} \frac{d\phi(r(x_c))}{dr}}_{B_{c,k}^{\mathbf{p}_x}}. \tag{B.2}$$

and so forth for the $\mathbf{p}_y \cdot \nabla$, $\mathbf{p}_z \cdot \nabla$ operators, where $\mathbf{x}_c$ is the stencil center and $\mathbf{x}_k$ are stencil nodes. To compute RBF-FD weights for the $\mathbf{p}_x \cdot \nabla$ operator, the RHS of Equation 1.4 is filled with elements $B_{c,k}^{\mathbf{p}_x}$. We will refer to this method of obtaining the weights as the *direct* method due to the ability to directly compute RBF-FD weights for the operators

$\mathbf{P} \cdot \nabla$, and assemble the differentiation matrices $\mathbf{D}_{\mathbf{p_x}\cdot\nabla}, \mathbf{D}_{\mathbf{p_y}\cdot\nabla}, \mathbf{D}_{\mathbf{p_z}\cdot\nabla}$ without the need to compute and/or store other weights.

B.2 Indirect Weights

Alternatively, one is able to compute weights *indirectly* as a weighted combination of existing RBF-FD weights for the unprojected ∇ operator. Here we assume that differentiation matrices to compute the components of ∇ are readily available in memory:

$$\mathbf{D}_{\nabla} = \begin{bmatrix} \mathbf{D}_{\frac{d}{dx}} \\ \mathbf{D}_{\frac{d}{dy}} \\ \mathbf{D}_{\frac{d}{dz}} \end{bmatrix},$$

where each matrix contains weights computed with the operators of Equation **??** applied to the RHS of Equation 1.4.

The differentiation matrices for $\mathbf{P} \cdot \nabla$ can then be assembled as a weighted combination of the differentiation matrices for the unprojected operator:

$$\mathbf{D}_{\mathbf{P}\cdot\nabla} = \begin{bmatrix} \mathbf{D}_{\mathbf{p_x}\cdot\nabla} \\ \mathbf{D}_{\mathbf{p_y}\cdot\nabla} \\ \mathbf{D}_{\mathbf{p_z}\cdot\nabla} \end{bmatrix} = \begin{bmatrix} diag(1-X^2)\mathbf{D}_{\frac{\partial x}{\partial x}} - diag(XY)\mathbf{D}_{\frac{d}{dy}} - diag(XZ)\mathbf{D}_{\frac{d}{dz}} \\ -diag(XY)\mathbf{D}_{\frac{d}{dx}} + diag(1-Y^2)\mathbf{D}_{\frac{d}{dy}} - diag(YZ)\mathbf{D}_{\frac{d}{dz}} \\ -diag(XZ)\mathbf{D}_{\frac{d}{dx}} - diag(YZ)\mathbf{D}_{\frac{d}{dy}} + diag(1-Y^2)\mathbf{D}_{\frac{d}{dz}} \end{bmatrix} \quad \text{(B.3)}$$

Author's Note: make it "partial x" instead of d/dx where $X = \{x_{c,i}\}_{i=1}^{N}$, $Y = \{y_{c,i}\}_{i=1}^{N}$, $Z = \{z_{c,i}\}_{i=1}^{N}$ are all x-components, y-components and z-components of the stencil centers $\{\mathbf{x}_{c,i}\}_{i=1}^{N}$ respectively. Author's Note: Gordon discussion: B1: explain why we use this manufactured solution. what was it designed to check. Author's Note: Cleanup: This concept equates to classical Finite Differences where for example, the standard 5-point finite difference formula for approximating the Laplacian can be expressed a weighted combination of differences for

Author's Note: Have not found in literature any mention of the fact that RBF-FD weights can be used to compose operators like this. Generally, its easier to directly compute weights and assumed to be more accurate. But how much more accurate?

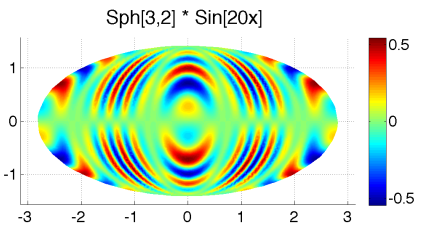
One benefit of indirect weights is conservation of memory. For example, for complex operators, a single DM on N = 1million nodes and $n = 101$ requires roughly 1.6 GB of memory. If the PDE is coupled and requires

This also allows us to compose complex operators with weights loaded from disk

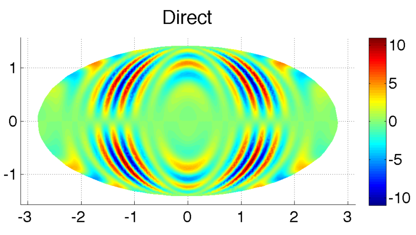
$O(N * n)$ cost to assemble the indirect operators after $O(N * n^3)$ cost of assembling direct operators means this approach has potential to save FLOPs in the long run

But the question is, how accurate is it? In situations where memory is critical and these FLOPs need to be saved (i.e., large N and complicated equations), would it be useful?

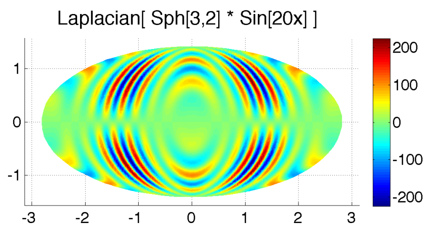
Author's Note: Q: how do direct vs indirect weights compare? Is the lsfc different sparsity with same approximation potentials? Author's Note: :end



(a) Manufactured test function: $f(\mathbf{x}) = Y_3^2(\mathbf{x})\sin(20x)$.



(b) x-component of the projected gradient: $\mathbf{p}_x \cdot \nabla f(\mathbf{x})$.



(c) Surface Laplacian: $\Delta_S f(\mathbf{x})$.

Figure B.1: Test function and its projected derivatives on the surface of the unit sphere.

B.2.1 Comparison of Direct and Indirect Weights

We computed direct and indirect approaches for the MD-node sets with size $N = \{121, 256, 400, 841, 1024, 2500,$

We check the relative error of the approximation:

$$\text{relative } \ell_2 \text{ error} = \frac{||f_{approx} - f_{exact}||_2}{||f_{exact}||_2}$$

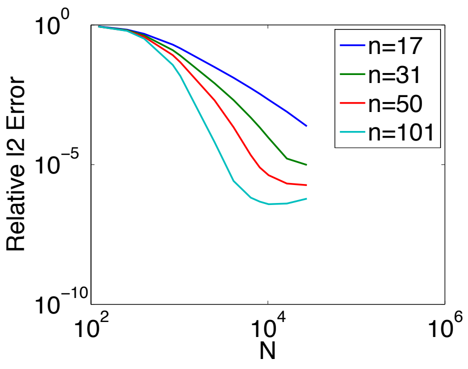
We also look at the difference of relative errors and its absolute value:

$$(\text{relative } \ell_2 \text{ error})_{\text{direct}} - (\text{relative } \ell_2 \text{ error})_{\text{indirect}}$$

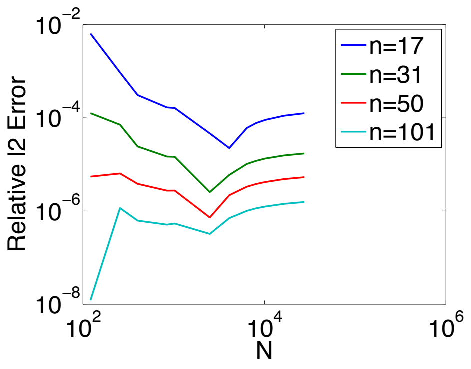
We find that our indirect approach functions well compared to the direct method. For small node sizes ($N < 2500$ nodes) we see that the direct method has the advantage with

B.3 Conclusions

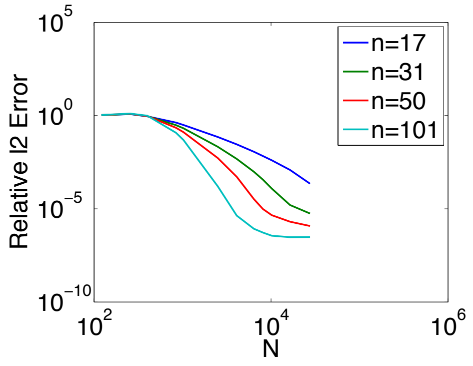
Although it is clear the indirect method functions well compared to the direct method, we must consider its usefulness. Typically, weights are computed only as necessary for the PDE. If the PDE is on the sphere, then directly computing the $\mathbf{P} \cdot \nabla$ operator would be most



(a) Δ_S of $Y_3^2 \sin(20x)$



(b) $\mathbf{p}_x \cdot \nabla(Y_3^2)$

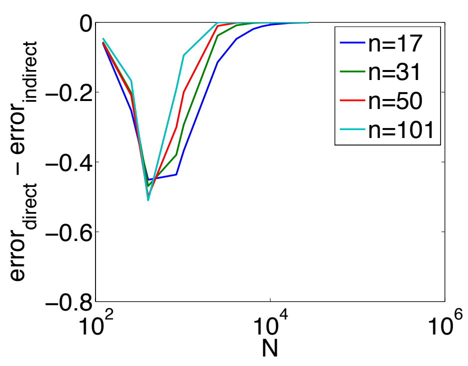


(c) $\mathbf{p}_x \cdot \nabla(Y_3^2 \sin(20x))$ Author's Note: rescale 1e0 down to 1e-8

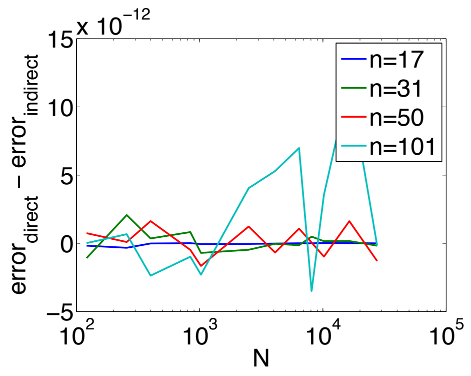
Figure B.2: Relative ℓ_2 error in differentiation.

efficient for both memory and computation. However, one could imagine a scenario such as a 3-D spherical shell domain with physics on the boundaries that must be constrained to the surface, while the interior requires only an unprojected ∇ operator. In such cases, by simply computing for the ∇ operator, we assemble all necessary operators with minimal loss of accuracy and significant savings ($3Nn$ doubles) in storage.

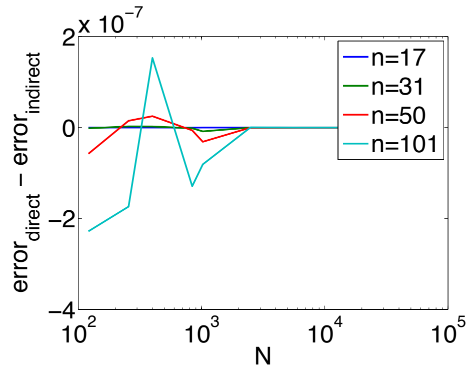
With $N = 1e6$ nodes and stencil size $n = 101$, the matrix market file for weights is approximately 1.6 GB on disk. For a GPU with only 6 GB of global memory space available, it is worthwhile to consider possibilities for memory conservation.



(a) Δ_S of $Y_3^2 \sin(20x)$

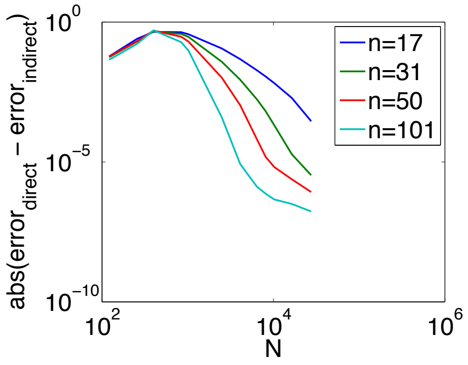


(b) $\mathbf{p}_x \cdot \nabla(Y_3^2)$

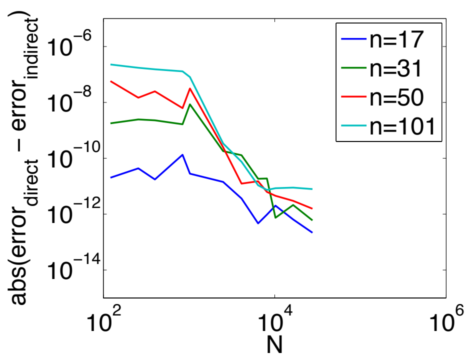


(c) $\mathbf{p}_x \cdot \nabla(Y_3^2 \sin(20x))$

Figure B.3: Signed differences of relative ℓ_2 errors in differentiation between Direct and Indirect weights.



(a) Δ_S of $Y_3^2 \sin(20x)$



(b) $\mathbf{p}_x \cdot \nabla(Y_3^2 \sin(20x))$

Figure B.4: Absolute differences of relative ℓ_2 errors in differentiation between Direct and Indirect weights.

Bibliography

[1] Sylvie Barak. Gpu technology key to exascale says nvidia. http://www.eetimes.com/electronics-news/4230659/GPU-technology-key-to-exascale-says-Nvidia, November 2011. 4

[2] R. K. Beatson, W. A. Light, and S. Billings. Fast Solution of the Radial Basis Function Interpolation Equations: Domain Decomposition Methods. *SIAM J. Sci. Comput.*, 22(5):1717–1740, 2000. 11

[3] Evan F. Bollig, Natasha Flyer, and Gordon Erlebacher. Solution to pdes using radial basis function finite-differences (rbf-fd) on multiple gpus. *Journal of Computational Physics*, (0):–, 2012. 13

[4] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and Representation of 3D Objects with Radial Basis Functions. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76, New York, NY, USA, 2001. ACM.

[5] J. C. Carr, R. K. Beatson, B. C. McCallum, W. R. Fright, T. J. McLennan, and T. J. Mitchell. Smooth Surface Reconstruction from Noisy Range Data. In *GRAPHITE '03: Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 119–ff, New York, NY, USA, 2003. ACM.

[6] Tom Cecil, Jianliang Qian, and Stanley Osher. Numerical Methods for High Dimensional Hamilton-Jacobi Equations Using Radial Basis Functions. *JOURNAL OF COMPUTATIONAL PHYSICS*, 196:327–347, 2004. 4, 5

[7] G Chandhini and Y Sanyasiraju. Local RBF-FD Solutions for Steady Convection-Diffusion Problems. *International Journal for Numerical Methods in Engineering*, 72(3), 2007. 4

[8] P P Chinchapatnam, K Djidjeli, P B Nair, and M Tan. A compact RBF-FD based meshless method for the incompressible Navier–Stokes equations. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 223(3):275–290, March 2009. 4

[9] CD Correa, D Silver, and M Chen. Volume Deformation via Scattered Data Interpolation. *Proceedings of Eurographics/IEEE VGTC Workshop on Volume Graphics*, pages 91–98, 2007.

[10] E Divo and AJ Kassab. An Efficient Localized Radial Basis Function Meshless Method for Fluid Flow and Conjugate Heat Transfer. *Journal of Heat Transfer*, 129:124, 2007. 11

[11] G. E. Fasshauer. RBF Collocation Methods and Pseudospectral Methods. Technical report, 2006.

[12] Gregory E. Fasshauer. Solving Partial Differential Equations by Collocation with Radial Basis Functions. In *In: Surface Fitting and Multiresolution Methods A. Le M'ehaut'e, C. Rabut and L.L. Schumaker (eds.), Vanderbilt*, pages 131–138. University Press, 1997.

[13] Gregory E. Fasshauer. *Meshfree Approximation Methods with MATLAB*, volume 6 of *Interdisciplinary Mathematical Sciences*. World Scientific Publishing Co. Pte. Ltd., 5 Toh Tuck Link, Singapore 596224, 2007. 13

[14] Gregory E. Fasshauer and Jack G. Zhang. On choosing "optimal" shape parameters for rbf approximation. *Numerical Algorithms*, 45(1-4):345–368, 2007.

[15] A. I. Fedoseyev, M. J. Friedman, and E. J. Kansa. Improved Multiquadric Method for Elliptic Partial Differential Equations via PDE Collocation on the Boundary. *Computers & Mathematics with Applications*, 43(3-5):439 – 455, 2002.

[16] Natasha Flyer and Bengt Fornberg. Radial basis functions: Developments and applications to planetary scale flows. *Computers & Fluids*, 46(1):23–32, July 2011.

[17] Natasha Flyer and Erik Lehto. Rotational transport on a sphere: Local node refinement with radial basis functions. *Journal of Computational Physics*, 229(6):1954–1969, March 2010.

[18] Natasha Flyer, Erik Lehto, Sebastien Blaise, Grady B. Wright, and Amik St-Cyr. Rbf-generated finite differences for nonlinear transport on a sphere: shallow water simulations. *Submitted to Elsevier*, pages 1–29, 2011. 4, 5, 7, 9, 10, 13, 18

[19] Natasha Flyer and Grady B. Wright. Transport schemes on a sphere using radial basis functions. *Journal of Computational Physics*, 226(1):1059 – 1084, 2007. 16

[20] Natasha Flyer and Grady B. Wright. A Radial Basis Function Method for the Shallow Water Equations on a Sphere. In *Proc. R. Soc. A*, volume 465, pages 1949–1976, December 2009. 9, 16, 17, 18

[21] B Fornberg, T Driscoll, G Wright, and R Charles. Observations on the behavior of radial basis function approximations near boundaries. *Computers & Mathematics with Applications*, 43(3-5):473–490, February 2002. 5, 7

[22] B. Fornberg and G. Wright. Stable computation of multiquadric interpolants for all values of the shape parameter. *Computers & Mathematics with Applications*, 48(5-6):853 – 867, 2004.

[23] Bengt Fornberg, Elisabeth Larsson, and Natasha Flyer. Stable Computations with Gaussian Radial Basis Functions in 2-D. Technical Report 2009-020, Uppsala University, August 2009.

[24] Bengt Fornberg, Elisabeth Larsson, and Natasha Flyer. Stable Computations with Gaussian Radial Basis Functions. *SIAM J. on Scientific Computing*, 33(2):869—-892, 2011.

[25] Bengt Fornberg and Erik Lehto. Stabilization of RBF-generated finite difference methods for convective PDEs. *Journal of Computational Physics*, 230(6):2270–2285, March 2011. 4, 5, 7, 10, 13

[26] Bengt Fornberg and Cécile Piret. A Stable Algorithm for Flat Radial Basis Functions on a Sphere. *SIAM Journal on Scientific Computing*, 30(1):60–80, 2007.

[27] Bengt Fornberg and Cécile Piret. On Choosing a Radial Basis Function and a Shape Parameter when Solving a Convective PDE on a Sphere. *Journal of Computational Physics*, 227(5):2758 – 2780, 2008.

[28] Richard Franke. Scattered Data Interpolation: Tests of Some Method. *Mathematics of Computation*, 38(157):181–200, 1982.

[29] Prashant Goswami, Philipp Schlegel, Barbara Solenthaler, and Renato Pajarola. Interactive sph simulation and rendering on the gpu. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 55–64, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association. 7

[30] R Hardy. Multiquadratic Equations of Topography and Other Irregular Surfaces. *J. Geophysical Research*, (76):1–905, 1971.

[31] Y. C. Hon and R. Schaback. On unsymmetric collocation by radial basis functions. *Appl. Math. Comput.*, 119(2-3):177–186, 2001.

[32] Yiu-Chung Hon, Kwok Fai Cheung, Xian-Zhong Mao, and Edward J. Kansa. A Multiquadric Solution for the Shallow Water Equations. *ASCE J. Hydraulic Engineering*, 125:524–533, 1999.

[33] A. Iske. *Multiresolution Methods in Scattered Data Modeling.* Springer, 2004.

[34] E J Kansa. Multiquadrics–A scattered data approximation scheme with applications to computational fluid-dynamics. I. Surface approximations and partial derivative estimates. *Computers Math. Applic*, (19):127–145, 1990.

[35] E J Kansa. Multiquadrics–A scattered data approximation scheme with applications to computational fluid-dynamics. II. Solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers Math. Applic*, (19):147–161, 1990.

[36] G Kosec and B Šarler. Solution of thermo-fluid problems by collocation with local pressure correction. *International Journal of Numerical Methods for Heat & Fluid Flow*, 18, 2008. 11

[37] Elisabeth Larsson and Bengt Fornberg. A Numerical Study of some Radial Basis Function based Solution Methods for Elliptic PDEs. *Comput. Math. Appl*, 46:891–902, 2003.

[38] Yuxu Lin, Chun Chen, Mingli Song, and Zicheng Liu. Dual-RBF based surface reconstruction. *Vis Comput*, 25(5-7):599–607, May 2009.

[39] X. Liu, G.R. Liu, K. Tai, and K.Y. Lam. Radial point interpolation collocation method (RPICM) for partial differential equations. *Computers & Mathematics with Applications*, 50(8-9):1425 – 1442, 2005.

[40] Bryan S. Morse, Terry S. Yoo, Penny Rheingans, David T. Chen, and K. R. Subramanian. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[41] C.T. Mouat and R.K. Beatson. RBF Collocation. Technical Report UCDMS2002/3, Department of Mathematics & Statistics, University of Canterbury, New Zealand, February 2002.

[42] Jia Pan and Dinesh Manocha. Fast GPU-based Locality Sensitive Hashing for K-Nearest Neighbor Computation. *Proceedings of the 19th ACM SIGSPATIAL GIS '11*, 2011. 7

[43] Robert Schaback. Multivariate Interpolation and Approximation by Translates of a Basis Function. In C.K. Chui and L.L. Schumaker, editors, *Approximaton Theory VIII–Vol. 1: Approximation and Interpolation*, pages 491–514. World Scientific Publishing Co., Inc, 1995. 4

[44] J. Schmidt, C. Piret, B.J. Kadlec, D.A. Yuen, E. Sevre, N. Zhang, and Y. Liu. Simulating Tsunami Shallow-Water Equations with Graphics Accelerated Hardware (GPU) and Radial Basis Functions (RBF). In *South China Sea Tsunami Workshop*, 2008. 13

[45] J. Schmidt, C. Piret, N. Zhang, B.J. Kadlec, D.A. Yuen, Y. Liu, G.B. Wright, and E. Sevre. Modeling of Tsunami Waves and Atmospheric Swirling Flows with Graphics Processing Unit (GPU) and Radial Basis Functions (RBF). *Concurrency and Computat.: Pract. Exper.*, 2009. 13

[46] C. Shu, H. Ding, and K. S. 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(7-8):941 – 954, 2003. 4, 5

[47] C Shu, H Ding, and N Zhao. Numerical Comparison of Least Square-Based Finite-Difference (LSFD) and Radial Basis Function-Based Finite-Difference (RBFFD) Methods. *Computers and Mathematics with Applications*, 51(8):1297–1310, 2006. 4

[48] D Stevens, H Power, M Lees, and H Morvan. The use of PDE centres in the local RBF Hermitian method for 3D convective-diffusion problems. *Journal of Computational Physics*, 2009. 4

[49] David Stevens, Henry Power, Michael Lees, and Herve Morvan. A Meshless Solution Technique for the Solution of 3D Unsaturated Zone Problems, Based on Local Hermitian Interpolation with Radial Basis Functions. *Transp Porous Med*, 79(2):149–169, Sep 2008.

[50] David Stevens, Henry Power, and Herve Morvan. An order-N complexity meshless algorithm for transport-type PDEs, based on local Hermitian interpolation. *Engineering Analysis with Boundary Elements*, 33(4):425 – 441, 2008. 11

[51] A. I. Tolstykh and D. A. Shirobokov. On using radial basis functions in a "finite difference mode" with applications to elasticity problems. In *Computational Mechanics*, volume 33, pages 68 – 79. Springer, December 2003. 4

[52] A.I. Tolstykh. On using RBF-based differencing formulas for unstructured and mixed structured-unstructured grid calculations. In *Proceedings of the 16 IMACS World Congress, Lausanne*, pages 1–6, 2000. 3, 4

[53] Robert Vertnik and Božidar Šarler. Meshless local radial basis function collocation method for convective-diffusive solid-liquid phase change problems. *International Journal of Numerical Methods for Heat & Fluid Flow*, 16(5):617–640, 2006.

[54] B. Šarler and R. Vertnik. Meshfree Explicit Local Radial Basis Function Collocation Method for Diffusion Problems. *Computers and Mathematics with Applications*, 51(8):1269–1282, 2006.

[55] J. G. Wang and G. R. Liu. A point interpolation meshless method based on radial basis functions. *Int. J. Numer. Methods Eng.*, 54, 2002.

[56] Grady Wright and Bengt Fornberg. Scattered node mehrstellenverfahren-type formulas generated from radial basis functions. In *The International Conference on Computational Methods*, December 15-17 2004. 4

[57] Grady B. Wright. *Radial Basis Function Interpolation: Numerical and Analytical Developments*. PhD thesis, University of Colorado, 2003. 3, 4

[58] Grady B. Wright, Natasha Flyer, and David A. Yuen. A hybrid radial basis function–pseudospectral method for thermal convection in a 3-d spherical shell. *Geochem. Geophys. Geosyst.*, 11(Q07003):18 pp., 2010. 8

[59] Grady B. Wright and Bengt Fornberg. Scattered node compact finite difference-type formulas generated from radial basis functions. *J. Comput. Phys.*, 212(1):99–123, 2006. 4, 5, 7

[60] Z M Wu. Hermite-Birkhoff interpolation of scattered data by radial basis functions. *Approx. Theory Appl*, (8):1–10, 1992.

[61] Xuan Yang, Zhixiong Zhang, and Ping Zhou. Local Elastic Registration of Multimodal Medical Image Using Robust Point Matching and Compact Support RBF. In *BMEI '08: Proceedings of the 2008 International Conference on BioMedical Engineering and Informatics*, pages 113–117, Washington, DC, USA, 2008. IEEE Computer Society.

[62] Rio Yokota, L.A. Barba, and Matthew G. Knepley. PetRBF — A parallel O(N) algorithm for radial basis function interpolation with Gaussians. *Computer Methods in Applied Mechanics and Engineering*, 199(25-28):1793–1804, May 2010. 13

[63] Hao Zhang and Marc G. Genton. Compactly supported radial basis function kernels, 2004.