TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Motivation for RBF-FD on the GPU

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 formally introduced in 2003 but 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. We capitalize on the inherent parallelism to develop a collection of multi-GPU test cases that span the compute nodes of a Top 500 supercomputer. Our effort leads the way for application of RBF-FD in the modern field of high performance computing where GPU accelerators will be key to breaching the exa-scale barrier in computing [3].

Rather than solely focus on the optimizations of said algorithms on the GPU, we dedicate significant attention to the practical application of RBF-FD to interesting problems in geophysics. This means we have walked a fine line between research topics to both apply the method to

- Introduction
  - relatively new method: RBF-FD
    - related methods (predecessors)
    - features/benefits
    - limitations: small problems, limited parallelism, no work to target HPC
  - relatively new paradigm: GPU computing
    - new standard in HPC
    - benefits
    - success stories
  - this thesis: application of multiple GPUs to RBF-FD
    - two core concepts: solve and communicate

    * leverage accelerated GPU primitives to achieve high performance (RK4, GMRES, if time permits: SIMPLE?)
    * applications: some seen (hyperbolic), some not seen (elliptic); final goal (if time permits): mantle?

- Chapter 1: RBF-generated Finite Differences
  - G
- Chapter 2: RBF-FD on GPUs
  - G
- Chapter 3: Distributed Computing with MPI
  - G
- Chapter 4: Application of RBF-FD: Stable Advection on the Sphere
  - Distributed SpMV
  - Hyperviscosity
  - Convergence Studies
  - Benchmarks
- Chapter 5: Application of RBF-FD: Implicit Solutions with Preconditioned GMRES
  - Annulus
  - Coupled System on Sphere
- Chapter 6: Approximate Nearest Neighbors, Node Ordering and Conditioning
  - a
  - b

CHAPTER 2

RBF METHODS FOR PDE SOLUTIONS

The process of solving PDEs with RBFs dates back to Kansa's seminary work on global RBF collocation in 1990 [29, **?**]. Beginning with a description of general approximation and interpolation with RBFs, this chapter provides background on the methods that preceded and ultimately led to the introduction of RBF-generated Finite Differences. This is followed by a derivation of Radial Basis Function-generated Finite Differences and examples of operators approximated by the method within this work.

We categorize existing methods for solving PDEs with RBFs as either global or local. Global methods are based on collocation and invert a single large linear system to find an interpolant that satisfies the differential equations at nodes in the domain. Local methods limit the influence of basis functions and seek an interpolant at each node defined in terms of neighboring basis functions (local collocation) or nodal values (RBF-FD).

The selling points of RBF-FD are numerous. While not a panacea for PDEs, the method combines many inherited traits of global RBF methods with lower computational complexity. As demonstrated within this chapter, the method is simple to code, easily extensible for higher accuracy and dimensions, and powerful in its ability to avoid singularities introduced by the coordinate system that might impact other methods.

2.1 History

2.2 Global RBF Method

Consider a PDE expressed in terms of (linear) differential operators, $\mathcal{L}$ and $\mathcal{B}$:

$$\begin{aligned} \mathcal{L}u &= f \quad \text{on } \Omega, \\ \mathcal{B}u &= g \quad \text{on } \Gamma \end{aligned} \tag{2.1}$$

where Ω is the interior of the physical domain, Γ is the boundary of Ω and f, g are known explicitly. In the case of a non-linear differential operator, a Newton's iteration, or some other method, can be used to linearize the problem (see e.g., [**?**]); of course, this increases the complexity of a single time step.

We solve PDEs of this form with *collocation*. That is, given points in the domain, we seek the derivative and solution values that best satisfy f, g at node locations. Author's Note: Not clear yet. Review Lynch2005 for a better explanation[**?**]

Global RBF collocation methods pose the problem of interpolating a multivariate function $f : \Omega \to \mathbb{R}$ where $\Omega \subset \mathbb{R}^m$. Given a set of sample values $\{f(x_j)\}_{j=1}^N$ on a discrete set of nodes $X = \{x_j\}_{j=1}^N \subset \Omega$, an approximation $\hat{f}_N$ can be constructed through linear combinations of interpolation functions. Here, we choose univariate, radially symmetric functions based on Euclidean distance ($||\cdot||$), and use translates $\phi(x - x_j)$ of a single continuous real valued function ϕ defined on $\mathbb{R}$ and centered at x_j:

$$\phi(x) := \varphi(||x||).$$

Here, φ is a Radial Basis Function and ϕ the associated kernel. For simplification $\phi_j(x)$ refers to a kernel centered at x_j; i.e., $\varphi(||x - x_j||)$.

The interpolant $\hat{f}_N(x)$ requires a linear combination of translates:

$$\hat{f}_N(x) = \sum_{j=1}^{N} c_j \phi_j(x)$$

with real coefficients $\{c_j\}_{j=1}^N$. Assuming the interpolant passes through known values of f; i.e.,

$$\hat{f}_N(x_i) = f(x_i), \qquad 1 \le i \le N,$$

allows one to solve for coefficients if the following linear system is uniquely solvable:

$$\sum_{j=1}^{N} c_j \phi_j(x_i) = f(x_i), \qquad 1 \le i \le N.$$

This is true if the $N \times N$ matrix Φ produced by the linear system

$$\begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_N(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_N(x_2) \\ \vdots & \ddots & \ddots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) & \cdots & \phi_N(x_N) \end{pmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_N) \end{bmatrix}$$

$$\Phi \vec{c} = \vec{f}$$

is nonsingular.

When choosing an appropriate basis function for interpolation, a subset of RBFs have been shown to produce symmetric positive definite Φ, while others are only conditionally positive definite (for more details see [14]). ***** Evan: reference table with examples of both ***** With the latter set, additional polynomial terms are added to constrain the system and enforce positive definiteness, resulting in this expanded system of equations:

$$\begin{aligned} \sum_{j=1}^{N} c_j \phi_j(x_i) + \sum_{k=1}^{M} d_k p_k(x_i) &= f(x_i), \qquad 1 \le i \le N, \\ \sum_{j=1}^{N} c_j p_k(x_j) &= 0, \qquad 1 \le k \le M. \end{aligned}$$

$$\begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} \vec{f} \\ 0 \end{bmatrix} \tag{2.2}$$

where $\{p_k\}_{k=1}^{M}$ is a basis for Π_p^m (the set of polynomials in m variables of degree $\leq p$) and

$$M = \binom{p+m}{m}.$$

Given the coefficients $\vec{c}$, the function value at a test point x is interpolated by

$$\begin{aligned}\hat{f}_N(x) &= \sum_{j=1}^{N} c_j \Phi_j(x) + \sum_{l=1}^{M} d_l P_l(x) \\ &= \begin{bmatrix} \Phi & \mathbb{P} \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} \\ &= \vec{\Phi}_x^T \vec{c} \\ &= \vec{\Phi}_x^T \Phi^{-1} \vec{f} \end{aligned} \tag{2.3}$$

where $\vec{c}$ is substituted by the solution to Equation 2.2. In Equation 2.3 the term $\Phi_x^T \Phi^{-1}$, dependent only on node positions, can be evaluated prior to knowing f.

2.3 The RBF-generated Finite Differences Method

For the RBF-FD method, derivatives of $u(x)$ are weighted combinations of a small neighborhood of N_s nodes (i.e., a stencil with $N_s \ll N$):

$$\mathcal{L}u(x_1) \approx \sum_{j=1}^{N_s} c_j u(x_j)$$

where $\mathcal{L}u$ denotes a linear differential quantity over u (e.g., $\mathcal{L}u = \frac{du}{dx}$). Here, c_j are unknown, and obtained by enforcing the reconstruction:

$$\mathcal{L}\phi_j(x_1) = \sum_{i=1}^{N_s} c_i \phi_j(x_i) \qquad \text{for } j = 1, 2, ..., N_s$$

with $\mathcal{L}\phi_j$ provided by analytically applying the differential operator to the kernel function. This is equivalent to:

$$\begin{pmatrix} \phi_1(x_1) & \phi_1(x_2) & \cdots & \phi_1(x_{N_s}) \\ \phi_2(x_1) & \phi_2(x_2) & \cdots & \phi_2(x_{N_s}) \\ \vdots & \ddots & \ddots & \vdots \\ \phi_{N_s}(x_1) & \phi_{N_s}(x_2) & \cdots & \phi_{N_s}(x_{N_s}) \end{pmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} \mathcal{L}\phi_1(x_1) \\ \mathcal{L}\phi_2(x_1) \\ \vdots \\ \mathcal{L}\phi_{N_s}(x_1) \end{bmatrix} \tag{2.4}$$

$$\Phi_s \vec{c} = \vec{\Phi_{\mathcal{L}_s}}. \tag{2.5}$$

Solving Equation 2.5 for stencils weights, $\vec{c}$, and substituting into Equation 2.7 reveals the similarity between RBF-FD and the general problem of RBF interpolation (see Equation 2.3 and discussion):

$$\begin{aligned}\mathcal{L}u(x_1) &\approx (\Phi_s^{-1} \vec{\Phi}_{\mathcal{L}_s})^T \vec{u} \\ &\approx \vec{\Phi}_{\mathcal{L}_s}^T \Phi_s^{-T} \vec{u}\end{aligned}$$

Again, based on the choice of RBF, positive definiteness of the system is ensured by adding the same constraints as Equation 2.2, but note differential quantities for $\vec{\mathbb{P}}$ on the right hand side:

$$\begin{pmatrix} \Phi_s & \mathbb{P} \\ \mathbb{P}^T & 0 \end{pmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} \vec{\Phi}_{\mathcal{L}_s} \\ \vec{\mathbb{P}}_{\mathcal{L}} \end{bmatrix}. \tag{2.6}$$

Equation 2.6 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. ***** Evan: Clarify that we add monomials for RBF-FD to reproduce a constant. The weights do not necessarily need to be constrained, but we do this in practice to prevent the systems from resulting in very large weights that could cause truncation errors. We could show a practical set of weight magnitudes with and without the constraint as an example? *****

2.4 Stencil-based Derivative Approximation

The RBF-FD method is similar to classical Finite Differences in that differential quantities at a single node referred to as a *center*, $\mathbf{x}_j$, is approximated by weighted combinations of function values at $n-1$ neighboring nodes. The center node and its neighbors define a *stencil*, $\{\mathbf{x}\}_{i=1}^n$, in a localized (small) region or *neighborhood* of the domain.

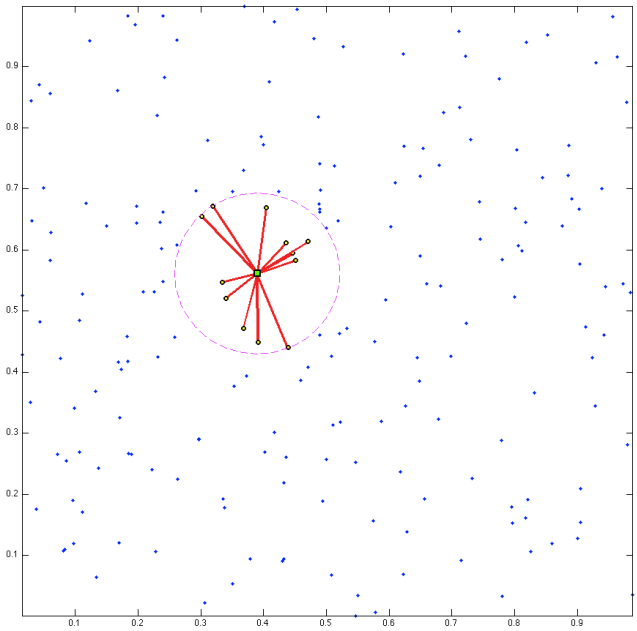
Figure 2.1 provides two examples of RBF-FD stencils. First, Figure 2.1a illustrates a single stencil of size $n = 13$ in a domain of randomly distributed nodes. The stencil center is represented by a green square, with the 12 neighboring nodes connected via red edges. The purple circle, the minimum covering circle for the stencil, demonstrates that the stencil contains only the 12 nearest neighbors of the center node. In Figure 2.1b, 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 the red and blue disks represent the magnitude and sign of the RBF-FD weights determined to calculate a derivative quantity at the stencil center.

Weights for both classical Finite Difference and RBF-FD can be obtained through the solution of linear systems. In the case of Finite Difference, the system is a Vandermonde matrix. For RBF-FD the system is based on a symmetric distance matrix. Author's Note: Need better description and ref: The key difference between FD and RBF-FD systems is the singularity that occurs when two nodes swap.

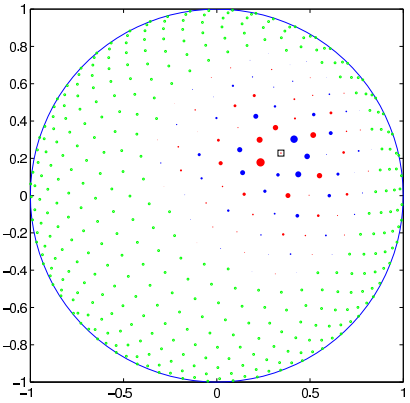
2.5 RBF-FD weights

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.

$$\mathcal{L}u(\mathbf{x}) \mid_{\mathbf{x}=\mathbf{x}_j} \approx \sum_{i=1}^{n} w_i u(\mathbf{x}_i) + w_{n+1} p_0 \tag{2.7}$$



(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 2.1: Examples of stencils computable with RBF-FD Author's Note: Consider: show example FD stencil (5pt with regular grid)

The RBF-FD weights, w_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 [**?**, **?**, 23, 17] that better accuracy is achieved when the interpolant can exactly reproduce a constant, p_0. Assuming $p_0 = 1$, the constraint $\sum_{i=1}^n w_i = \mathcal{L}1|_{\mathbf{x}=\mathbf{x}_j} = 0$ completes the system:

$$\begin{pmatrix} \phi(\epsilon\,||\mathbf{x}_1-\mathbf{x}_1||) & \phi(\epsilon\,||\mathbf{x}_1-\mathbf{x}_2|| & \cdots & \phi(\epsilon\,||\mathbf{x}_1-\mathbf{x}_n||) & 1 \\ \phi(\epsilon\,||\mathbf{x}_2-\mathbf{x}_1||) & \phi(\epsilon\,||\mathbf{x}_2-\mathbf{x}_2|| & \cdots & \phi(\epsilon\,||\mathbf{x}_2-\mathbf{x}_n||) & 1 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ \phi(\epsilon\,||\mathbf{x}_n-\mathbf{x}_1||) & \phi(\epsilon\,||\mathbf{x}_n-\mathbf{x}_2|| & \cdots & \phi(\epsilon\,||\mathbf{x}_n-\mathbf{x}_n||) & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{pmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ w_{n+1} \end{bmatrix} = \begin{bmatrix} \mathcal{L}\phi(\epsilon\,||\mathbf{x}-\mathbf{x}_1||)|_{\mathbf{x}=\mathbf{x}_j} \\ \mathcal{L}\phi(\epsilon\,||\mathbf{x}-\mathbf{x}_2||)|_{\mathbf{x}=\mathbf{x}_j} \\ \vdots \\ \mathcal{L}\phi(\epsilon\,||\mathbf{x}-\mathbf{x}_n||)|_{\mathbf{x}=\mathbf{x}_j} \\ 0 \end{bmatrix}, \tag{2.8}$$

where w_{n+1} is ignored after the matrix in (2.8) 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 $O(n^3 N)$ 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 (2.8), 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 [**?**, **?**].

2.6 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 2.8 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

$$\frac{\partial \phi}{\partial x} = \frac{dr}{dx}\frac{d\phi}{dr} = \frac{(x - x_j)}{r}\frac{d\phi}{dr} \tag{2.9}$$

$$\frac{\partial \phi}{\partial y} = \frac{dr}{dy}\frac{d\phi}{dr} = \frac{(y - y_j)}{r}\frac{d\phi}{dr} \tag{2.10}$$

$$\frac{\partial \phi}{\partial z} = \frac{dr}{dz}\frac{d\phi}{dr} = \frac{(z - z_j)}{r}\frac{d\phi}{dr} \tag{2.11}$$

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

2.6.1 Laplacian (∇^2)

2D:

3D:

2.6.2 Laplace-Beltrami (Δ_S) on the Sphere

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

$$\nabla^2 = \underbrace{\frac{1}{r}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\right)}_{\text{radial}} + \underbrace{\frac{1}{r^2}\Delta_S}_{\text{angular}}, \tag{2.12}$$

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:

$$\nabla^2 \equiv \Delta_S. \tag{2.13}$$

Although this originated in the spherical coordinate system, [47] introduced the following Laplaci-Beltrami operator for the surface of the sphere:

$$\Delta_S = \frac{1}{4}\left[\left(4 - r^2\right)\frac{\partial^2}{\partial r^2} + \frac{4 - 3r^2}{r}\frac{\partial}{\partial r}\right], \tag{2.14}$$

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

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

Additionally following [19, 17], the gradient operator must also be constrained to the sphere with this projection matrix:

$$P = I - \mathbf{x}\mathbf{x}^T = \begin{pmatrix} (1-x_1^2) & -x_1x_2 & -x_1x_3 \\ -x_1x_2 & (1-x_2^2) & -x_2x_3 \\ -x_1x_3 & -x_2x_3 & (1-x_3^2) \end{pmatrix} = \begin{pmatrix} P_{x_1} \\ P_{x_2} \\ P_{x_3} \end{pmatrix} \tag{2.15}$$

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 [19]. When solving for the weights, we apply the projection on the right hand side of our small linear system. We let $\mathbf{x} = \left(x_1, x_2, x_3\right)$ be the stencil center, and $\mathbf{x}_k = \left(x_{1,k}, x_{2,k}, x_{3,k}\right)$ 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 2.8:

$$P\frac{\partial}{\partial x_1} = (x_1\mathbf{x}^T\mathbf{x}_k-x_{1,k})\frac{1}{r(\mathbf{x}_k-\mathbf{x})}\frac{\partial\phi(r(\mathbf{x}_k-\mathbf{x}))}{\partial r}|_{\mathbf{x}=\mathbf{x}_j} \tag{2.16}$$

$$P\frac{\partial}{\partial x_2} = (x_2\mathbf{x}^T\mathbf{x}_k-x_{2,k})\frac{1}{r(\mathbf{x}_k-\mathbf{x})}\frac{\partial\phi(r(\mathbf{x}_k-\mathbf{x}))}{\partial r}|_{\mathbf{x}=\mathbf{x}_j} \tag{2.17}$$

$$P\frac{\partial}{\partial x_3} = (x_3\mathbf{x}^T\mathbf{x}_k-x_{3,k})\frac{1}{r(\mathbf{x}_k-\mathbf{x})}\frac{\partial\phi(r(\mathbf{x}_k-\mathbf{x}))}{\partial r}|_{\mathbf{x}=\mathbf{x}_j} \tag{2.18}$$

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

2.7 Stabilization: Hyperviscosity

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

$$D = \alpha\frac{\partial}{\partial\lambda}+\beta\frac{\partial}{\partial\theta} \tag{2.19}$$

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 [23, 17]. Stabilization of the RBF-FD method is achieved through the application of a hyperviscosity filter to Equation (2.19) [23]. By using Gaussian RBFs, $\phi(r)=e^{-(\epsilon r)^2}$, the hyperviscosity (a high order Laplacian operator) simplifies to

$$\Delta^k\phi(r) = \epsilon^{2k}p_k(r)\phi(r) \tag{2.20}$$

where k is the order of the Laplacian and $p_k(r)$ are multiples of generalized Laguerre polynomials that are generated recursively (see [23]: 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

$$\frac{\partial u}{\partial t} = -Du + Hu, \tag{2.21}$$

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 $N \to \infty$ [17]. 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 [23, 17].

2.8 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

$$\Delta^k \phi(r) = \epsilon^{2k} p_k(r) \phi(r). \tag{2.22}$$

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

2.9 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 [14] queries n nearest neighbors for a compact-support RBF partition of unity example with a k-D tree. In [17, 23] a k-D Tree is leveraged for all neighbor queries for RBF-FD.

In our work in [4] an alternative to k-D tree was leveraged, based loosely on Locality Sensitive Hashing.

2.9.1 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.

CHAPTER 3

RBF-FD ON THE GPU

BIBLIOGRAPHY

[1] HMPP Data Sheet. http://www.caps-entreprise.com/upload/article/fichier/64fichier1.pdf, 2009.

[2] AccelerEyes. *Jacket User Guide - The GPU Engine for MATLAB*, 1.2.1 edition, November 2009.

[3] 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. 1

[4] 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. 12

[5] Andreas Brandstetter and Alessandro Artusi. Radial Basis Function Networks GPU Based Implementation. *IEEE Transaction on Neural Network*, 19(12):2150–2161, December 2008.

[6] 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.

[7] 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.

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

[9] 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.

[10] A Corrigan and HQ Dinh. Computing and Rendering Implicit Surfaces Composed of Radial Basis Functions on the GPU. *International Workshop on Volume Graphics*, 2005.

[11] N Cuntz, M Leidl, TU Darmstadt, GA Kolb, CR Salama, M Böttinger, D Klimarechenzentrum, and G Hamburg. GPU-based Dynamic Flow Visualization for Climate Research Applications. *Proc. SimVis*, pages 371–384, 2007.

[12] 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.

[13] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal Voronoi Tessellations: Applications and Algorithms. *SIAM Rev.*, 41(4):637–676, 1999.

[14] 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. 4, 12

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

[16] 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.

[17] 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. 7, 9, 10, 11, 12

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

[19] 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

[20] 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.

[21] Bengt Fornberg and Natasha Flyer. Accuracy of Radial Basis Function Interpolation and Derivative Approximations on 1-D Infinite Grids. *Adv. Comput. Math*, 23:5–20, 2005.

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

[23] 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. 7, 10, 11, 12

[24] 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.

[25] 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.

[26] E. J. Fuselier and G. B. Wright. A High-Order Kernel Method for Diffusion and Reaction-Diffusion Equations on Surfaces. *ArXiv e-prints*, May 2012.

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

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

[29] 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. 3

[30] Khronos OpenCL Working Group. *The OpenCL Specification (Version: 1.0.48)*, October 2009.

[31] 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.

[32] 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.

[33] Shaofan Li and Wing K. Liu. *Meshfree Particle Methods*. Springer Publishing Company, Incorporated, 2007.

[34] NVidia. *NVIDIA CUDA - NVIDIA CUDA C - Programming Guide version 4.0*, March 2011.

[35] Portland Group Inc. *CUDA Fortran Programming Guide and Reference*, 1.0 edition, November 2009.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] J.S. Vetter, R. Glassbrook, J. Dongarra, K. Schwan, B. Loftis, S. McNally, J. Meredith, J. Rogers, P. Roth, K. Spafford, and S. Yalamanchili. Keeneland: Bringing heterogeneous GPU computing to the computational science community. *IEEE Computing in Science and Engineering*, 13(5):90–95, 2011.

[44] M Weiler, R Botchen, S Stegmaier, T Ertl, J Huang, Y Jang, DS Ebert, and KP Gaither. Hardware-Assisted Feature Analysis and Visualization of Procedurally Encoded Multifield Volumetric Data. *IEEE Computer Graphics and Applications*, 25(5):72–81, 2005.

[45] Holger Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4:389–396, 1995. 10.1007/BF02123482.

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

[47] 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. 9

[48] 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.