A Multi-GPU, Multi-CPU Implementation of RBF-FD for PDE Solutions

Evan F. Bollig, Advisor: Gordon Erlebacher

Department of Scientific Computing, Florida State University

bollig@scs.fsu.edu

Abstract

DURING the last few decades, numerical methods that use collections of radially symmetric, univariate functions for approximation have surfaced for PDE solutions. Such functions are referred to as Radial Basis Functions (RBFs) and have been successfully employed in spectral, pseudo-spectral and localized modes. RBF methods are ideal for unstructured or scattered nodes; e.g., node sets generated with centroidal Voronoi tessellation.

One of the newest areas of research leverages RBFs in the calculation of weights for generalized finite difference stencils in a scheme referred to as RBF-FD. RBF-FD addresses a major concern with other RBF methods—namely, ill-conditioning.

The RBF-FD method exhibits a large amount of parallelism, which we target in a multi-CPU, multi-GPU framework for solving 2D and 3D PDEs. To span multiple CPUs and multiple GPUs, an Additive Schwarz domain decomposition method is used to partition the physical domain and distribute work across processors. Each processor then works with a GPU to offload computationally demanding tasks. Since individual RBF-FD stencils may span multiple subdomains, inter-processor communication is required at each iteration.

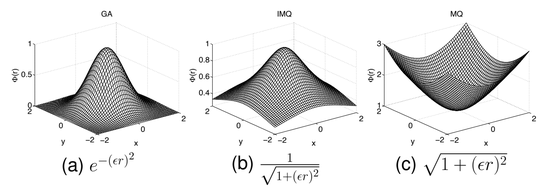
Details of our implementation, along with case studies solving elliptic and parabolic PDEs using implicit and explicit schemes, are provided. Our end goal is to use this code for Tsunami and other geophysical simulations.

1. Radial Basis Functions

Definition 1 *A function $\phi : \mathbb{R}^D \to \mathbb{R}$ is radial provided a univariate function $\varphi : [0, \infty) \to \mathbb{R}$ exists such that*

$$\phi(x) = \varphi(\epsilon \left|\left|x - x_j\right|\right|) = \varphi(\epsilon \, ||r||)$$

where x_j is the center or point of origin *for the function, and $||\cdot||$ is some norm (typ. ℓ_2) on $\mathbb{R}^D$. ϵ is a support scaling parameter [2].*



(a) $e^{-(\epsilon r)^2}$ (b) $\frac{1}{\sqrt{1+(\epsilon r)^2}}$ (c) $\sqrt{1+(\epsilon r)^2}$

Figure 1: *Commonly used Radial Basis Functions*

2. RBF Interpolation

GIVEN a set of sample values $\{u(x_j)\}_{j=1}^N$ on a discrete set of nodes $X = \{x_j\}_{j=1}^N$, weighted combinations of RBFs construct an approximation to the continuous u:

$$u(x) = \sum_{j=1}^{N} c_j \phi_j(x) + \sum_{l=1}^{M} d_l P_l(x), \qquad P_l(x) \in \Pi_m^D$$

$M = \binom{m+D}{D}$. In 2D (m=1): $\mathbb{P} = 1, x, y$. This is the system

$$\Phi c + \mathbb{P} d = u(x)$$

and additional constraints ensure positive definiteness:

$$\mathbb{P}^T c = 0.$$

The resulting system is:

$$\begin{bmatrix} \Phi & \mathbb{P} \\ \mathbb{P}^T & 0 \end{bmatrix} \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} u(x) \\ 0 \end{pmatrix}$$
$$Ac = u.$$

With weights available, linear differential operators, $\mathcal{L}f(x)$ (e.g., $\mathcal{L} = \frac{du}{dx}$), can be approximated:

$$\begin{aligned} \mathcal{L}u &= \sum_{j=1}^{N} c_j \mathcal{L}\Phi_j(x) + \sum_{l=1}^{M} d_l \mathcal{L}P_l(x) = \begin{bmatrix} \mathcal{L}\Phi & \mathcal{L}\mathbb{P} \end{bmatrix} \begin{pmatrix} c \\ d \end{pmatrix} \\ &= A_{\mathcal{L}} c \\ &= A_{\mathcal{L}} A^{-1} u \end{aligned}$$

- RBF Collocation methods use this approach

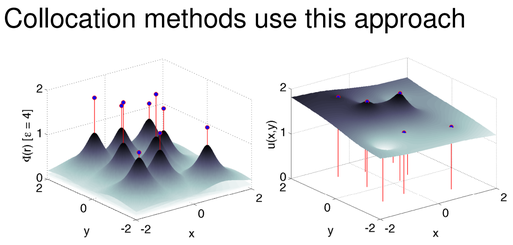


Figure 2: *Linear combinations of ϕ are used for approximation of u and its derivatives.*

3. RBF-FD

WE solve PDEs using a generalized FD scheme called RBF-FD [3]. Derivatives of $u(x)$ are weighted combinations of a small neighborhood of N_s nodes (i.e., a stencil with $N_s \ll N$):

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

where c_j are unknown, but obtained by solving:

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

If we add constraints for positive definiteness as above we get:

$$\begin{bmatrix} \Phi & \mathbb{P} \\ \mathbb{P}^T & 0 \end{bmatrix} \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} \mathcal{L}\phi(x_1) \\ 0 \end{pmatrix}$$

- Similar to small scale RBF Interpolation; different RHS
- Find stencil weights using only ϕ and analytic derivatives.
- Apply weights to $u(x_j)$ to get derivative quantity at $u(x_1)$ (i.e., stencil center)

4. Implicit PDEs

THE RBF-FD method was used in an implicit scheme to solve Poisson's equation with non-uniform diffusivity:

$$\begin{aligned} \boldsymbol{\nabla} \cdot [D(T, \mathbf{r}) \boldsymbol{\nabla} T(\mathbf{r})] &= F \\ \boldsymbol{\nabla} D \cdot \boldsymbol{\nabla} T + D \boldsymbol{\nabla}^2 T &= \end{aligned}$$

where $\mathbf{r} = (x, y)^T$. In Figure 3 we consider the exact solution:

$$\begin{aligned} T(\mathbf{r}) = 100000 * [&\sin(||\mathbf{r}|| - 1)(||\mathbf{r}|| - 0.5) + \pi) \\ &+ (||\mathbf{r}|| - 1)(||\mathbf{r}|| - 0.5)] \end{aligned}$$

over a 2D annulus with non-uniform diffusivity: $D(T, \mathbf{r}) = y$.

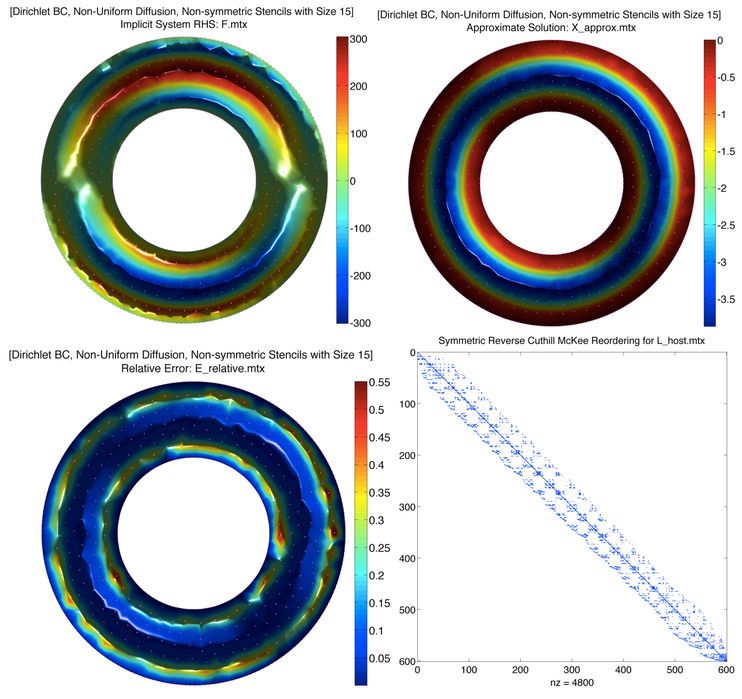


Figure 3: *Example PDE solved using RBFFD. Poisson's equation with non-uniform diffusion.*

- Nodes near boundary (but not on) are an issue — these have lopsided stencils
- Sparse matrix of weights, L (i.e., $Lu = F$), is NOT symmetric
- Uses BiCGStab solver in ViennaCL (a sparse matrix GPU toolkit written in OpenCL) [1]
- Limited to single GPU, single CPU

5. Explicit PDEs

AN explicit PDE solver has been implemented for time dependent problems like the diffusion equation:

$$\frac{\partial T(\mathbf{r}, t)}{\partial t} = \boldsymbol{\nabla} \cdot [D(T, \mathbf{r}) \boldsymbol{\nabla} T(\mathbf{r}, t)]$$

- Stencil weights are the same as implicit scheme, but avoids linear system solve for solution
- OpenCL kernel applies stencils weights to calculate derivatives
- CPU uses derivatives to advance timestep

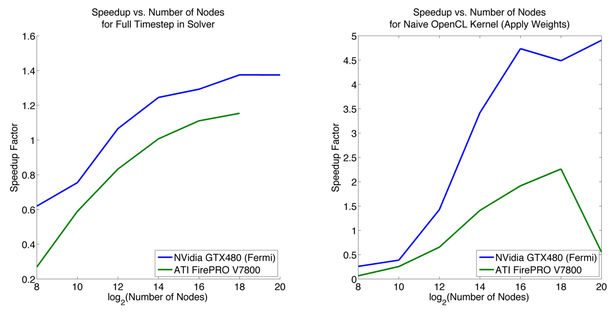
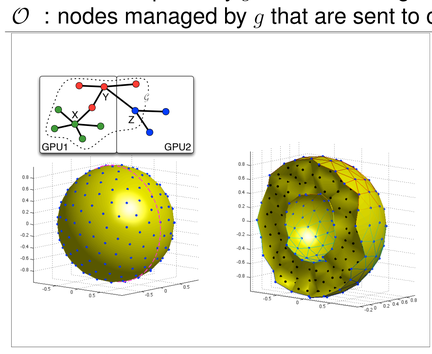


Figure 4: *Speedup factors achieved with unoptimized OpenCL on NVidia and ATI hardware. NVidia GTX 480 (Fermi) with 2x 4core Xeon @ 2.93 GHz, vs. ATI FirePRO V7800 with 2core Duo @ 3.0 GHz. (Left) Full timestep with CPU and GPU kernel. (Right) Apply weights kernel only (Subset of timestep).*

6. Domain Decomposition

WE implement a similar Additive Schwarz physical domain decomposition as [4] to parallelize computation across multiple processes (each associated with a GPU). Stencil sets are structured in memory to enable overlapping computation and communication for increased efficiency. A stencil centered on one CPU/GPU may contain nodes stored on other CPU/GPUs. Intermediate updates for ghost nodes are passed across subdomains using MPI.

$\mathcal{G}$	: all nodes received and contained on the GPU g
$\mathcal{Q}$	: stencil centers managed by g (equivalently, stencils computed by g)
$\mathcal{B}$	: stencil centers managed by g that require nodes on another GPU
$\mathcal{R}$	: nodes required by g that are managed by another GPU
$\mathcal{O}$	: nodes managed by g that are sent to other GPUs



$x, y, z \in \mathcal{G}$	$x, y \in \mathcal{Q}$
$y \in \mathcal{B}$	$z \in \mathcal{R}$
$y \in \mathcal{O}$	$\mathcal{G} = \mathcal{R} \cup \mathcal{Q}$
$x \in \mathcal{Q} \backslash \mathcal{B}$	$\mathcal{B} \subset \mathcal{Q}$

Set order in memory:
$\mathcal{G} = \{\mathcal{Q} \backslash \mathcal{O} \;\; \mathcal{O} \;\; \mathcal{R}\}$

Table 1: *Stencil set partitioning by compute node (GPU1).*

Multi-stage derivative calculation:

$$u'_k = (A_D A^{-1})_k u_{k,\mathcal{Q}\backslash\mathcal{O}} + (A_D A^{-1})_k u_{k,\mathcal{O}} + (A_D A^{-1})_k u_{k,\mathcal{R}}$$

- First and second products do not require communication and can be processed immediately
- Last product can block until communication finishes
- Benchmarks from acm.sc.fsu.edu (GPU cluster) and HPC's new GPU nodes are forthcoming

7. Conclusions & Future Work

Introduced the RBF-FD method and two implementations:

- an implicit scheme implementation using ViennaCL
- a parallel, multi-GPU explicit implementation using OpenCL
  - current speedup factor is small per time-step
  - consider both NVidia and ATI specific optimizations in the near future
  - offload more computation to the GPU
  - need to amortize cost of MPI communication by overlapping GPU computation
  - will be one of first test codes to benchmark on new HPC GPU nodes

8. Acknowledgements

This work is supported by NSF award No. #0934331: "CMG Collaborative Research: Fast and Efficient Radial Basis Function Algorithms for Geophysical Modeling on Arbitrary Geometries".

References

[1] ViennaCL. http://viennacl.sourceforge.net/.

[2] FASSHAUER, G. E. *Meshfree Approximation Methods with MATLAB*, vol. 6 of *Interdisciplinary Mathematical Sciences*. World Scientific Publishing Co. Pte. Ltd., 5 Toh Tuck Link, Singapore 596224, 2007.

[3] TOLSTYKH, A. I., AND SHIROBOKOV, D. A. On using radial basis functions in a "finite difference mode" with applications to elasticity problems. In *Computational Mechanics*, vol. 33. Springer, December 2003, pp. 68 – 79.

[4] YOKOTA, R., BARBA, L., AND KNEPLEY, M. G. PetRBF — A parallel O(N) algorithm for radial basis function interpolation with Gaussians. *Computer Methods in Applied Mechanics and Engineering 199*, 25-28 (May 2010), 1793–1804.