Parallel Algorithms for RBF-FD Solutions to PDEs on the Sphere

Evan F. Bollig[1] Gordon Erlebacher[1] Natasha Flyer[2]

[1]Dept. of Scientific Computing, Florida State University
[2]Institute for Mathematics Applied to Geosciences (IMAGe), NCAR

Introduction

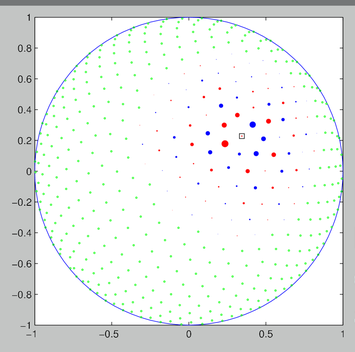


Figure 1: A 75 node RBF-FD stencil with blue (negative) and red (positive) differentiation weights to approximate advective operator at the square.

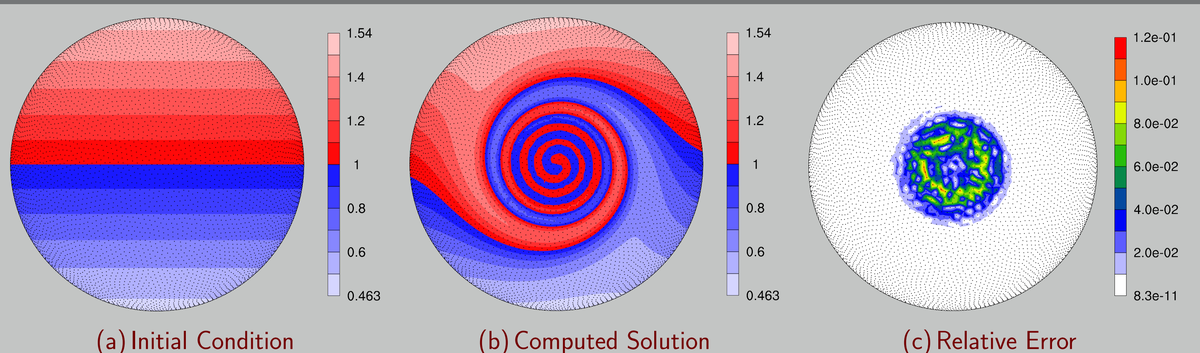
We introduce a multi-CPU/GPU implementation for the solution of hyperbolic PDEs on a sphere using Radial Basis Functions (RBF). This work targets the NSF funded Keeneland GPU cluster, which—like many of the latest HPC systems around the world—offers significantly more GPU accelerators (360 units) than CPU counterparts (240 units). We present our parallelization strategy, algorithms and data-structures used to span computation across the system.

RBF-FD Weights (for one n-node stencil centered at x_j)

$$\begin{pmatrix} \phi(\epsilon\,||\mathbf{x}_1-\mathbf{x}_1||) & \phi(\epsilon\,||\mathbf{x}_1-\mathbf{x}_2||) & \cdots & \phi(\epsilon\,||\mathbf{x}_1-\mathbf{x}_n||) & 1 \\ \phi(\epsilon\,||\mathbf{x}_2-\mathbf{x}_1||) & \phi(\epsilon\,||\mathbf{x}_2-\mathbf{x}_2||) & \cdots & \phi(\epsilon\,||\mathbf{x}_2-\mathbf{x}_n||) & 1 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ \phi(\epsilon\,||\mathbf{x}_n-\mathbf{x}_1||) & \phi(\epsilon\,||\mathbf{x}_n-\mathbf{x}_2||) & \cdots & \phi(\epsilon\,||\mathbf{x}_n-\mathbf{x}_n||) & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{pmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \\ c_{n+1} \end{bmatrix} = \begin{bmatrix} \mathcal{L}\phi(\epsilon\,||\mathbf{x}-\mathbf{x}_1||)|_{\mathbf{x}=\mathbf{x}_j} \\ \mathcal{L}\phi(\epsilon\,||\mathbf{x}-\mathbf{x}_2||)|_{\mathbf{x}=\mathbf{x}_j} \\ \vdots \\ \mathcal{L}\phi(\epsilon\,||\mathbf{x}-\mathbf{x}_n||)|_{\mathbf{x}=\mathbf{x}_j} \\ 0 \end{bmatrix}$$

- ϕ is Gaussian RBF centered at x_k, $k = 1, ..., n$
- $\mathcal{L}$ is some differential operator (i.e., $\frac{\partial}{\partial\lambda}$, $\frac{\partial}{\partial\theta}$, ∇^k, etc.); form multiple RHS system for efficiency
- Repeat this $n \times n$ system solve for all N stencils.

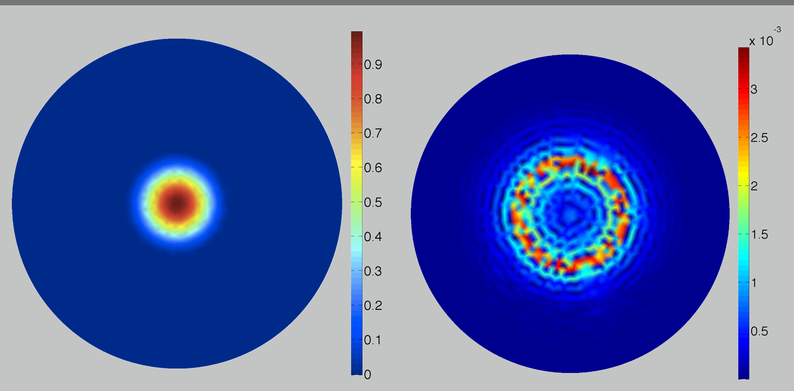
Test Case 1: Vortex Roll-up of a Fluid ($\frac{\partial h}{\partial t} + \frac{\omega(\theta)cos(\theta)}{\cos\theta}\frac{\partial h}{\partial \lambda} = 0$)



(a) Initial Condition (b) Computed Solution (c) Relative Error

Figure 2: Vortex roll-up solution at time $t = 10$ using RBF-FD with $N = 10,201$ and $n = 50$ point stencil. Normalized ℓ_2 error of solution at $t = 10$ is $1.25(10^{-2})$

Test Case 2: Advection of a C^1 Cosine Bell ($\frac{\partial h}{\partial t} + \frac{\sin\theta\cos\lambda}{\cos\theta}\frac{\partial h}{\partial\lambda} - \sin\lambda\frac{\partial h}{\partial\theta} = 0$)



(a) Computed Solution (b) Absolute Error

Figure 3: Cosine bell solution after 10 full revolutions over north and south poles with $N = 10201$ nodes and stencil size $n = 101$. The solid body is intact with the majority of error where the discontinuity in the derivative appears.

Stability

- RBF-FD Differentiation Matrices (DM) contain unstable eigenvalues for both test cases
- Adding a small amount of Hyperviscosity damps high modes and stabilizes DM

$$\frac{du}{dt} = -Du + Hu \quad \text{where} \quad H = \gamma_c N^{-k}\Delta^k$$

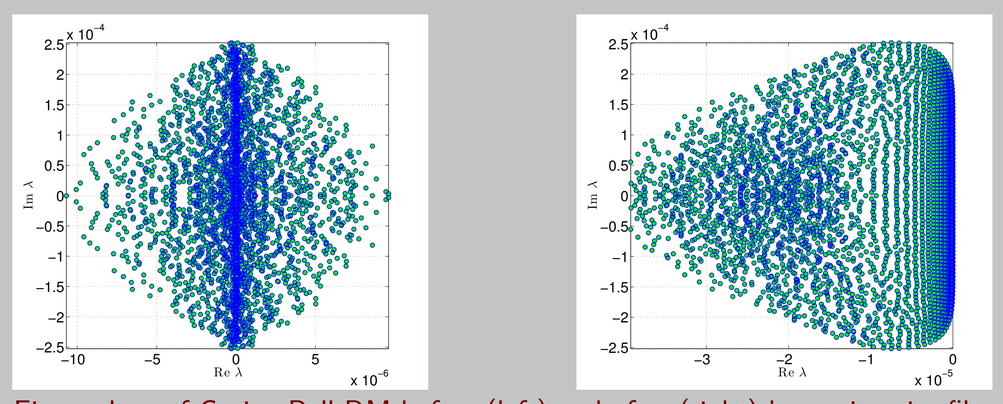


Figure 4: Eigenvalues of Cosine Bell DM before (left) and after (right) hyperviscosity filter is added.

Convergence

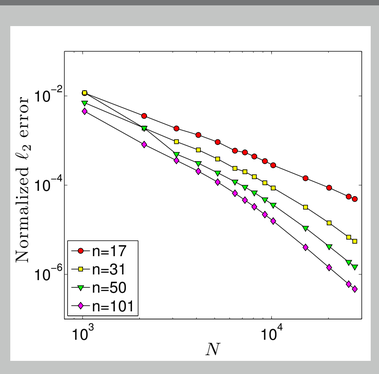


Figure 5: Convergence of C^∞ Vortex Roll-up on Maximum Determinant node sets [2]

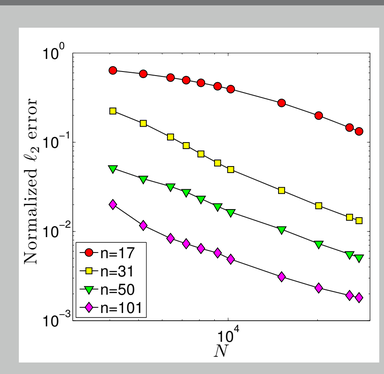


Figure 6: Discontinuity in the derivative of C^1 Cosine Bell limits convergence

Parallelization

- The geometry is partitioned into overlapping subdomains with one partition assigned to each CPU
- One GPU is associated with every CPU
- Stencils may span one or more partitions
- The Message Passing Interface (MPI) enables synchronization of solution/intermediate values between steps of RK4
- When computing on multiple GPUs, synchronization involves GPU→CPU transfer, CPU→CPU communication via MPI, and finally CPU→GPU transfer.

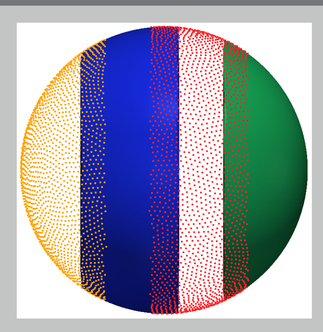
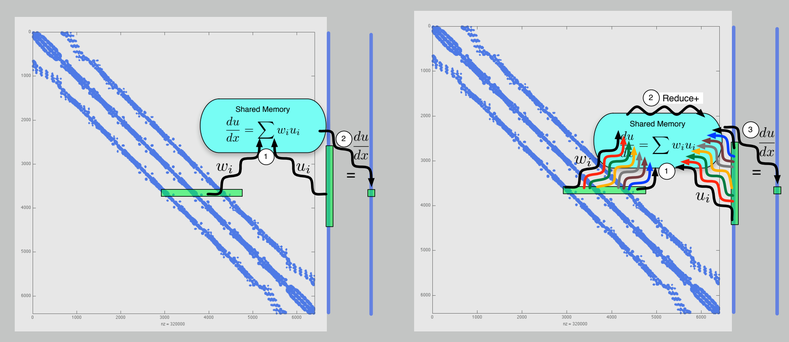


Figure 7: Partitioning of $N = 10,201$ nodes to span four processors with stencil size $n = 31$. Alternating representations (node points and interpolated surfaces) illustrate regions of partitions synchronized via MPI.

GPU Kernels (4th Order Runge Kutta (RK4) in OpenCL)

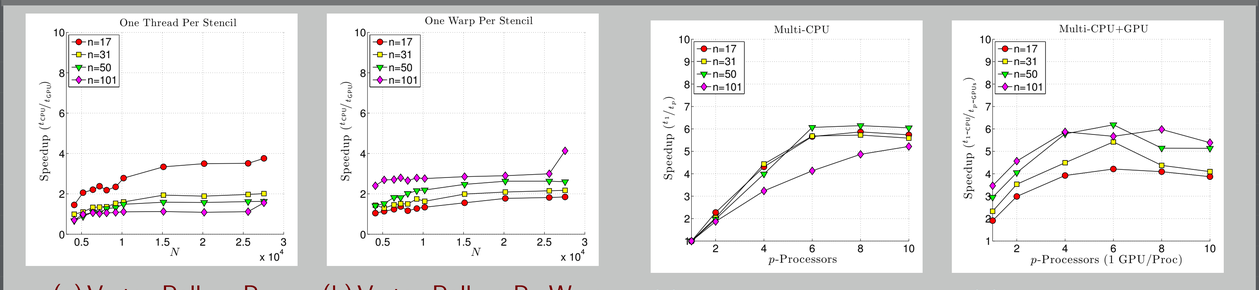
1. Kernel to evaluate derivatives on right hand side of PDE (called 4x per time-step)
2. Kernel for advancing PDE in time (called 1x per time-step)



(a) One Thread Per Stencil (b) One Warp Per Stencil

Figure 8: Two kernels are tested for derivative evaluation. The first dedicates one thread to compute the sparse vector dot product for each stencil. In the second, a full warp (32 threads) collaborate to perform the same task.

Performance: Multi-CPU, Multi-GPU and Various Stencil Sizes



(a) Vortex Roll-up By Thread (b) Vortex Roll-up By Warp (a) Vortex Roll-up with No GPUs (b) Vortex Roll-up with GPUs

(c) Cosine Bell Advection By Thread (d) Cosine Bell Advection By Warp (c) Cosine Bell with No GPUs (d) Cosine Bell with GPUs

Figure 9: Speedup ($t_{serial}/t_{parallel}$) achieved on a single GPU with respect to a single CPU for the two test cases, both GPU kernels, and various stencil sizes (n).

Figure 10: Multi-CPU (a, c) and multi-GPU (b, d) scaling for the fixed problem size $N = 27,556$. Speedup is measured relative to the serial time on a single CPU.

Acknowledgements

This work is supported by NSF awards DMS-#0934331 (FSU), DMS-#0934317 (NCAR) and ATM-#0602100 (NCAR), and additional details on can be found in [1].

[1] Evan F. Bollig, Natasha Flyer, and Gordon Erlebacher. Using Radial Basis Function Finite Difference (RBF-FD) for PDE Solutions on the GPU. submitted to Elsevier J. Comput. Phys., 2011.

[2] Ian H. Sloan and Robert S. Womersley. Extremal systems of points and numerical integration on the sphere. Adv. Comput. Math, 21:107–125, 2003.