# Computational Numerical Integration for Spherical Quadratures

## In a highly parallel fashion

Presenter: Huston Rogers
Mentors: Glenn Brook, Greg Peterson

## Introduction

The use of numerical integration techniques is pervasive in several contexts of applied science and engineering. Several approaches to numerical integration allow for the introduction of parallel algorithms. This summer I produced a highly parallel code for approaching numerical integration using the next-generation Beacon supercomputing resource at NICS.

## Equations and approach

The trapezoidal rule was chosen to provide the foundation for numerical integration for its inherent ease of parallel implementation. The final codebase was tested against known values, arising from specific test cases [1], for bulk integrals associated to problems in the Boltzmann regime. Due to an interdependency, the integration was split into two portions, where values from the first were utilized in the second. A grid refinement was then done to verify the program and determine a "sweet-spot" for overall error vs. grid complexity.

$$\iiint_S f(\rho,\theta,\phi)\rho^2 sin(\theta)\mathrm{d}V = Density$$

$$\iiint_S f(\rho,\theta,\phi)V_i\rho^2 sin(\theta)\mathrm{d}V = U_i * Density$$

$$c^2 = (V_x - U_x)^2 + (V_y - U_y)^2 + (V_z - U_z)^2$$

$$\frac{\iiint_S f(\rho,\theta,\phi)c^2\rho^2 sin(\theta)\mathrm{d}V}{3*Density} = Temperature$$

$$Density * Temperature = Pressure$$

$$c^3 = (V_x - U_x)^3 + (V_y - U_y)^3 + (V_z - U_z)^3$$

$$\frac{\iiint_S f(\rho,\theta,\phi)c^3\rho^2 sin(\theta)\mathrm{d}V}{2} = HeatFlux$$

$$\mathrm{d}V = \mathrm{d}\rho\mathrm{d}\theta\mathrm{d}\phi$$

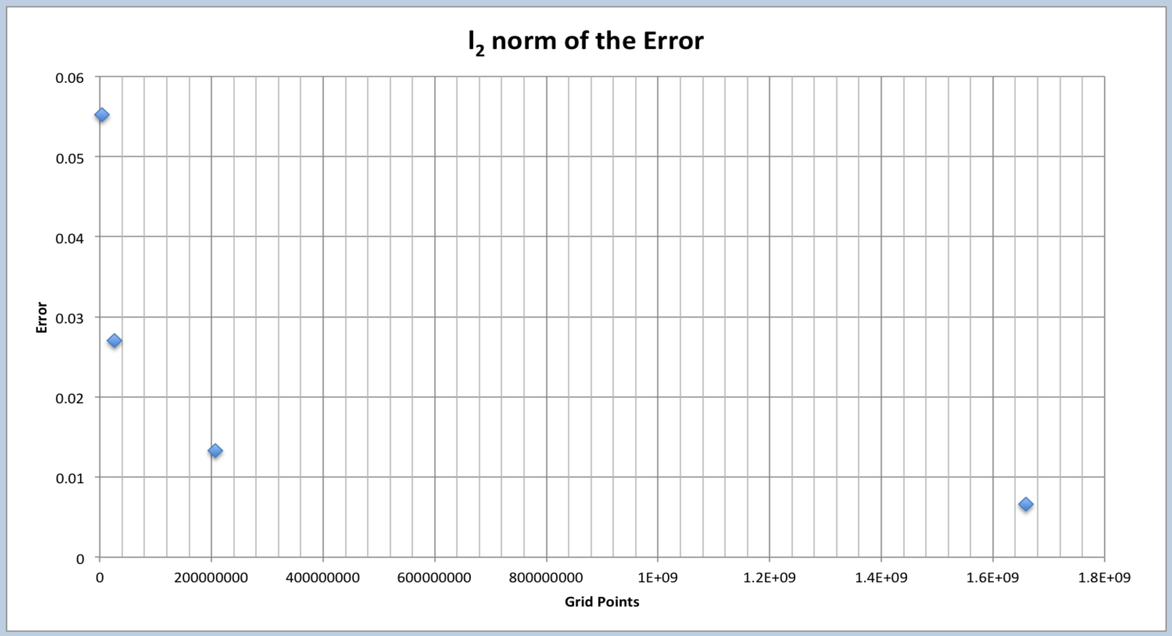
In the above equations U is the average velocity tensor while the tensor V represents discrete velocities.

## Code Characteristics

- Code designed to run on Beacon: A next-generation Green 500 supercomputer based on the Intel®Xeon Phi™coprocessor architecture
- Step 1
  - Hybrid OpenMP and MPI. 8 OMP threads mapped to the 8 cores of the Xeon Processor
  - MPI distributes the problem among multiple processors
  - Sums, reductions, and sharing performed with MPI calls Reduce, Allreduce, and Bcast
- Step 2
  - Migrate code for use with Intel®Many Integrated Core™architectures, specifically the Intel®Xeon Phi™coprocessor
- Pseudocode
  - Initialize function, parameters, and mesh
  - Assign density and moment integrations to MPI ranks; then sum, reduce, and broadcast
  - Using the first stage results, compute, then reduce the temperature

## Verification

The graph at right represents the $l_2$ norm of the error between the known values for each of the quantities ($U_\rho$,$U_\theta$,$U_\phi$) and the values calculated from the parallel quadrature. The initial mesh point distribution was ($N_\rho$,$N_\theta$,$N_\phi$) = (50,180,360) where each of the N values represents the number of grid points in that direction. The refinement was performed in all three directions simultaneously by a factor of 2.




## References

1. R. G. Brook, "A parallel, matrix-free newton method for solving approximate boltzmann equations on unstructured topologies," Ph.D. dissertation, University of Tennessee at Chattanooga, 2008.

## Acknowledgements

Work was done at the National Institute for Computational Sciences utilizing their resources. Supported in part by the University of Tenneessee Knoxville, The Joint Institute for Computational Sciences, Oak Ridge National Lab, and the National Science Foundation.