

Jayenne Implicit Monte Carlo Project: Y2008 Improvements

Todd Urbatsch, CCS-2; Scott Mosher, ORNL; Seth R. Johnson, University of Michigan; Michael Buksas, CCS-2; Aimee Hungerford, X-4-PC; Jeffery Densmore, Chris L. Fryer, Timothy Kelley, Paul Henning, Gabriel Rockefeller, CCS-2

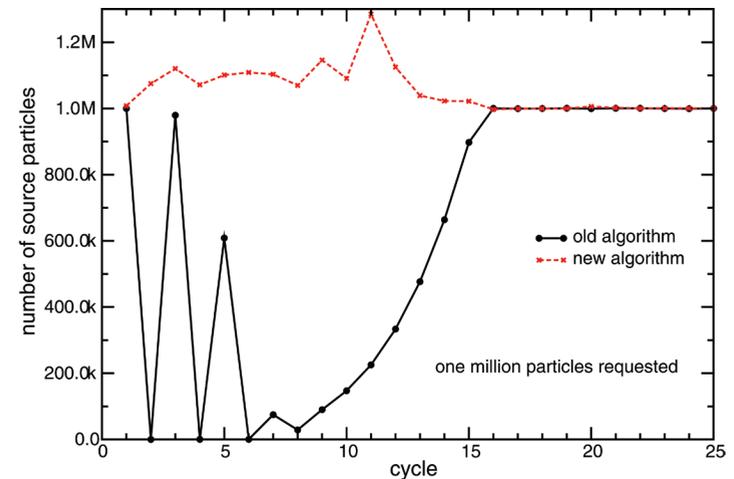
Fig. 1. The new particle sourcing algorithm avoids the undersampling problems with the old algorithm and typically gives at least the number of particles requested.

The Jayenne Implicit Monte Carlo (IMC) Project [1] in CCS-2 is a computational physics software project for simulating thermal X-ray transport using the Fleck and Cummings IMC algorithm [2]. These thermal X-ray simulations often are part of larger radiation-hydrodynamic simulations of such high-energy-density phenomena as supernovae and inertial confinement fusion. We highlight three of the improvements made during the year 2008.

Calculating the Number of Source Particles. In the IMC method, the X-ray energy is represented by Monte Carlo particles. For each timestep, the particles can come from different types of sources (emission, spatial boundary sources, and initial-time step source), and they can be distributed throughout the spatial domain of the simulation problem. Constraints are to keep the total number of particles at the user requested value and to give each particle approximately the same energy-weight. Determining the particle distribution is, roughly, an integer optimization problem. Algorithmically, we are trying to find the root of:

$$F = N(N_{\text{guess}}) - N_{\text{requested}} = 0$$

where $N()$ is the function that determines the full distribution of particles for a given guess of total numbers of particles. The old algorithm was based on a linear residual between iterations and could catastrophically and surreptitiously fail by giving the user nearly zero particles in any given time step, thus propagating large errors thereafter. Our new algorithm is a composite method (similar to that of Brent's [3]) that uses the old linear residual to start, false position, Ridders' method [4], and bisection, and that



uses a new stopping criterion. As shown in Fig. 1, the new algorithm solves the issues with the old algorithm, and, when coupled to a temperature cutoff to avoid sampling particles in unimportant regions, it becomes a very robust algorithm for sourcing IMC particles.

Asynchronous Transport Schemes. When the Jayenne Project started in 1997, one of its goals was to be massively parallel. Large, highly resolved problems cannot fit an entire mesh on a single processor, so hopes of “embarrassingly parallel” IMC via mesh replication were limited. Thus, domain decomposition (DD) parallelism was built into the Jayenne Project at the outset. With DD, a particle transports until it reaches the edge of the spatial domain on that processor, then it is buffered, and the buffers are sent to the processor containing the next spatial domain. The management of communicating these buffers poses a problem for efficiency and robustness. Our original asynchronous algorithm had each processor performing a continuous loop over the following prioritized options: a) transport N source particles, b) look for incoming particles and transport them, c) flush outgoing buffers, d) check for incoming buffers, or e) send number of particles completed to Node0 and check for all-finished flag. Running on LANL’s SGI O2 Bluemountain supercomputer of that era, we could not run with $N > 1$.

In 2004, Tom Brunner, SNL, started comparing and improving both our method and a method from LLNL. He started with $N > 1$, added a binary tree on the work-completed and all-finished messages, and replaced loops over MPI Test with one MPI_Testsome call [5]. Subsequently, he added true asynchronous Sends and a way to handle a dynamically changing number of particles as happens with Monte Carlo splitting. We analyzed his results and found that, although we would like to make our individual Sends truly asynchronous, the biggest bang for the buck was using $N > 1$. We came up with an empirical formula for automatically setting N . Speedups were on the order of 2 to 4.

Opacity Distribution Functions. The usual way to represent particle-frequency dependence in the material opacity data is to divide the frequency span into groups and, with an assumed weighting function, average the data to get one opacity value per group. Unfortunately, this multigroup (MG) approach cannot practically resolve the detailed structure in the data. The Opacity Distribution Function (ODF) method was first presented in 1935 [6] to represent a fuller range of opacities inside each frequency group. Thus, each group has multiple opacity values. Used mainly in stellar atmosphere simulations, ODFs were not typically used in high-energy-density simulations. We have implemented ODFs using opacity data from the X-1 code TOPS. Our results for a Marshak wave in iron, Fig. 2, show that the 32-group/8-band ODF reproduces 1024 groups in standard MG for a factor of four in memory savings.

Nevertheless, the ODF method loses all frequency dependence within a group, which adversely affects transporting to new materials/cells, and could reduce the accuracy of material motion and Compton algorithms. We initiated research on a modification to the ODF method that attempts to better correlate the band opacity values to the frequency. Our approach was to retain, from the underlying data, the minimum and maximum frequency within each band. Then, allowing for shadowing between bands, we could sample a frequency given a band (and the inverse)

accurately within a group for monotonic or single-peaked opacity-versus-frequency data. Our approach does not have the desired accuracy-to-memory value, but it can pave the way for more sophisticated ways to represent high-order opacity-frequency data in low-order forms.

For further information contact Todd Urbatsch at tmonster@lanl.gov.

- [1] T.J. Urbatsch, T.M. Evans, LANL Report LA-14195-MS (2005).
- [2] J.A. Fleck, Jr., J.D. Cummings, *J. Comp. Phys.* **8**, 313 (1971).
- [3] W.H. Press et al., *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, 2nd Ed., Cambridge University Press (1992).
- [4] C.J.F. Ridders, *IEEE Transactions on Circuits and Systems*, CAS-26, **11**, 979-980 (1979).
- [5] T.A. Brunner et al., *J. Comp. Phys.* **212**, 527-539 (2006).
- [6] J. Van Paradijs, M.S. Vardya, *Astrophys. Space Sci.* **33**, L9-L12 (1975).

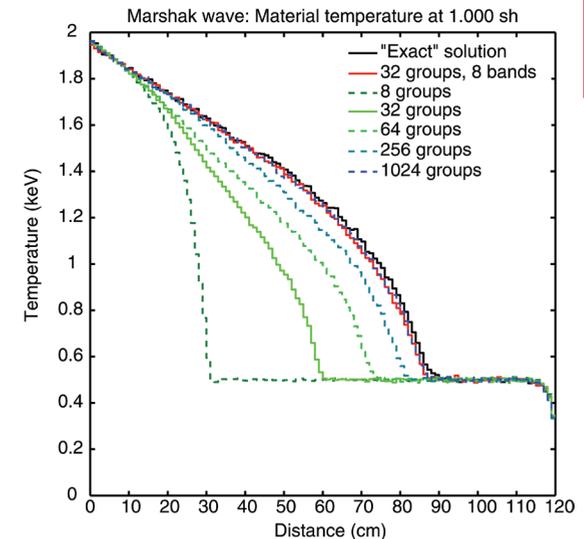


Fig. 2. Opacity distribution functions here require four times less memory and, thus, appear to converge much quicker than the regular multigroup approach.

Funding

Acknowledgments

DOE, NNSA, Advanced Simulation and Computing Program