ANS RPSD 2014 - 18th Topical Meeting of the Radiation Protection & Shielding Division of ANS
Knoxville, TN, September 14 – 18, 2014, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2014)

Development of CSG-based radiation shielding module for ARCHER: preliminary results for photons

Xining Du, Tianyu Liu, Lin Su, Wei Ji, Peter F. Caracappa, X. George Xu
Nuclear Engineering Program, Rensselaer Polytechnic Institute, Troy, New York, USA; email: xug2@rpi.edu

INTRODUCTION

ARCHER [1] is a parallel Monte Carlo radiation transport code being developed for various hardware platforms including multi-core CPUs, the emerging Nvidia graphics processing units (GPUs) and Intel Xeon Phi coprocessors. Previous studies have demonstrated the accuracy and speed for CT imaging and radiotherapy dosimetry applications. Radiation shielding applications, however, require ARCHER to be more flexible in defining and handling geometries, in addition to the voxel phantom geometries that are commonly used in medical physics calculations.

In this paper, we present the development of a general geometry module used for radiation protection and shielding problems in ARCHER. The methods of geometry design and particle transport for the GPUs are introduced. Preliminary results comparing against the MCNP code for cases involving Cs-137 0.662MeV gammas are reported.

DESCRIPTION OF ACTUAL WORK

For handling common geometries encountered in radiation protection and shielding design, we adopted constructive solid geometry (CSG) definitions used by MCNP/MCNPX codes [2], In CSG, complex 3-dimensional objects are modeled using Boolean operations to combine basic objects. A series of surface primitives are first defined in ARCHER. These include planes, spherical surfaces and cylindrical surfaces. Table 1 lists the surfaces that are supported in the geometry module and the definition parameters. A cell is described as the intersection of a group of half-spaces, which are defined as the region with relative orientations (sense) to corresponding surfaces. Each cell is either filled with certain material, a universe, or a lattice structure. A universe is defined as a space that contains one or more cells.

The new geometry module was integrated with the previously developed photon transport modules in ARCHER by replacing the geometry inquiry functions at every step of the transport process: instead of directly checking the position of each photon in different regions, e.g. a voxel human phantom or bowtie filters of the CT scanner model, the code now calls the geometry function and traverse all the cells to find the one within the particle is located. The ray-tracing algorithm was adopted for transporting photons in the CSG: At each step, the sampled photon collision distance, *s*, is compared with the distance to cell boundary surfaces, *d*. If *s* is larger than *d*, the photon will move to the nearest intersection point and repeat the sampling process. Otherwise the photon will be transported to the new position with step size *s* and the photon interaction outcome is sampled according to the corresponding cross sections. Figure 1 shows a simplified flow chart of photon transport involving the ray-tracing algorithm described above.

Table 1. List of surfaces supported in the geometry module of ARCHER.

Surface	Equation	Parameters
Plane parallel to x-axis	$x - x_0 = 0$	x_0
Plane parallel to y-axis	$y - y_0 = 0$	y_0
Plane parallel to z-axis	$z - z_0 = 0$	z_0
Sphere	$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - R^2 = 0$	x_0, y_0, z_0, R
Cylinder parallel to x-axis	$(y - y_0)^2 + (z - z_0)^2 - R^2 = 0$	y_0, z_0, R
Cylinder parallel to y-axis	$(x - x_0)^2 + (z - z_0)^2 - R^2 = 0$	x_0, z_0, R
Cylinder parallel to z-axis	$(x - x_0)^2 + (y - y_0)^2 - R^2 = 0$	x_0, y_0, R

Three interactions are considered for the MC transport of photons with energy in the range of 1 keV to 1 MeV. These are Compton (incoherent) scattering, photoelectric effects and Rayleigh (coherent) scattering. Form factors are included for Compton and Rayleigh scattering to account for electrons' binding effect in the scattering interaction.

With the radiation shielding module, ARCHER is capable of dealing with more complex user-defined geometries. In input files the users can define a series of surfaces and specify different cells by listing the associated half spaces (surface with sense). The code then reads in all the geometry information, build a list of cells and store them in memory.

We first developed the parallel CPU code and then ported it to the GPU platform using Nvidia's CUDA (Compute Unified Device Architecture) framework [3].

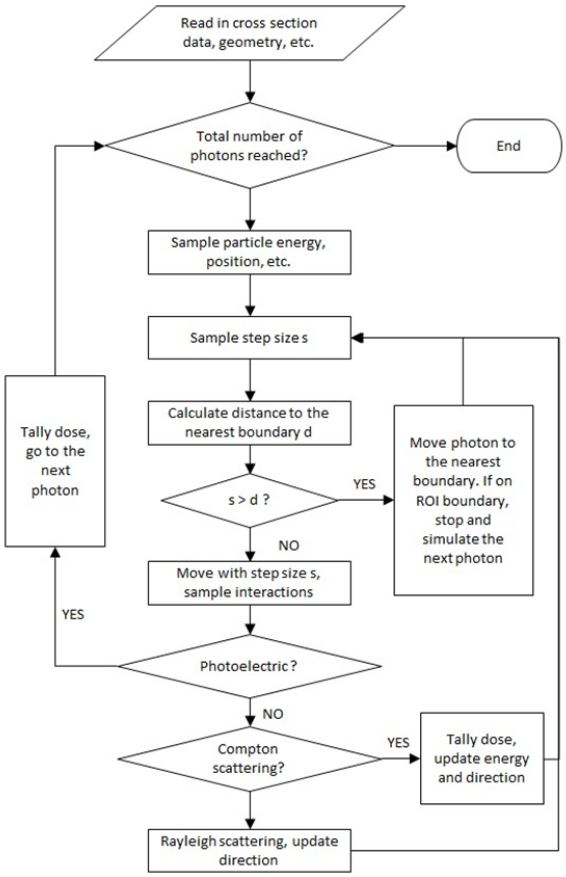

Figure 1. Flow chart of photon transport using the ray tracing algorithm.

In the CUDA framework, the CPU serves the role of the "host" as the main controller of the whole program. The GPU serves the role of an external "device" and performs the computationally intensive, parallelizable part of the work. Thousands of active threads can run concurrently on one GPU. In a typical CUDA program, the data is first transferred from host memory to GPU memory, then a kernel function is called by the CPU and executed on the GPU in parallel, finally the results are transferred back from GPU memory to host memory.

RESULTS

We tested the radiation shielding design module of ARCHER in a problem involving a 1-Ci Cs-137 point source located within a cylindrical lead shield (radius=0.1m, length=0.06m) in a 10m × 10m × 4m room of concrete walls. The geometry setup is depicted in Fig. 2. MCNP and ARCHER were used to calculate the dose to cell 4 using the direct energy deposition tally. Several runs were performed with different thicknesses of the lead container represented by cell 2. In each run a total of 2.56×10^7 photons with energy 0.662MeV are simulated.

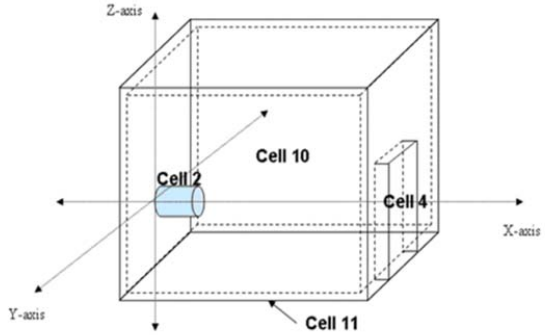

Figure 2. Setup of the radiation shielding design problem used for the test.

We tested ARCHER and MCNP on a workstation equipped with a quad-core Intel Xeon E5507 2.26GHz CPU and an Nvidia Kepler K20 GPU card. For 2.56×10^7 photons, MCNP running on the Intel E5507 with 4 threads took about 118 seconds while $\text{ARCHER}_{\text{GPU}}$ running on the Kepler K20 took 13 seconds. Dose results from both codes are in reasonable agreement with each other. We compared the performance of two methods for photon transport, namely the ray tracing algorithm and the Woodcock delta tracking algorithm [4], and found that the former is much more efficient than the latter for radiation shielding simulations. The reason is that in Woodcock delta tracking, the average step size is small due to the use of fictitious cross sections (the highest one among all materials). Thus significantly more steps are needed to transport the particles in the geometry with the penalty outperforming the time saved for the absence of boundary checking. On-going work is focused on applying variance reduction techniques, and integrating the electron transport model to ARCHER's radiation shielding module.

CONCLUSION

The radiation shielding design module of ARCHER has been developed, which supports general geometry setups in the CSG format. The code was implemented using CUDA on Nvidia GPUs, and the performance was benchmarked against the MCNP code for a simple radiation shielding problem. Results showed that ARCHER worked properly and achieved impressive speedup due to highly parallel computing nature of the Nvidia GPUs compared to the MCNP code running on multi-core CPUs. Further optimization of the geometry module will be carried out using advanced geometry structures and variance reduction techniques to improve the precision and performance of ARCHER.

ACKNOWLEDGEMENT

This work was partially supported by the grant from NIBIB (R01EB015478).

REFERENCES

1. X.G. Xu, T. Liu, L. Su, X. Du, M.J. Riblett, W. Ji, “Update of ARCHER, a Monte Carlo Radiation Transport Software Testbed for Emerging Hardware Such as GPUs”. Transactions of the American Nuclear Society, Vol. 108, Atlanta, Georgia, June 16–20, 2013.
2. X-5 Monte Carlo Team, MCNP-a general Monte Carlo N-Particle Transport code, Version 5. Volume I: overview and theory, 2003.
3. NVIDIA, CUDA C Programming Guide, http://docs.nvidia.com/cuda/cuda-c-programming-guide/
4. E. Woodcock, T. Murphy, P. Hemmings, and T. Longworth, Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry, 1965, Argonne National Laboratories Report ANL-7050.