Performance of Woodcock delta-tracking in lattice physics applications using the Serpent Monte Carlo reactor physics burnup calculation code

J. Leppänen

This week's article discusses the benefits of using Woodcock delta tracking in the open source Monte Carlo code, Serpent. Delta tracking solves some of the problems that Monte Carlo algorithms experience in regards to having to calculate new track lengths each time a particle transitions from one material region in a geometry to another. To avoid these expensive calculations, delta tracking uses a majorant cross section (the largest cross section in the problem) to calculate path lengths. Collisions are then weighted based on the true cross section of the material (sometimes being rejected as virtual collisions, otherwise counted as a true collision). When materials have generally similar cross section, this procedure saves much of the time spent calculating boundary transtions. Evidence of this enhancement is given in the article, where Leppänen shows an increase in speed for almost all cases tested with what is determined to be the optimal amount of delta tracking for the problem. Some geometries, like BWRs and PWRs see an increase in speed of only about 2 times, while others like HTGRs see a speed increase by a factor of about 13.

I thought that on the whole, the article was well presented and fairly readable. It had a good distribution of background information, motivation, a brief discussion of limitations and results, but was also concise enough that it was not unwieldy. I think it might have been interesting if the author had described what the order of calculations might be for a sample problem ($\mathcal{O}(n^2)$ for traditional MC, but $\mathcal{O}(n)$ for delta tracking). That being said, this calculation would be geometry dependent, so such a calculation might be prohibitively difficult for anything more than one or two simple geometries.