LA-UR-12-24621

Approved for public release; distribution is unlimited.

Title:

A NOTABLE COMPARISON OF COMPUTATIONAL GEOMETRIES IN MCNP6 CALCULATIONS

Author(s):

Roger L. Martz and Kevin M. Marshall

Intended for:

Nuclear Science and Engineering, Vol. 184, Nov. 2013.

LA-UR-12-24621

A NOTABLE COMPARISON OF COMPUTATIONAL GEOMETRIES IN MCNP6 CALCULATIONS

Roger L. Martz[1] and Kevin M. Marshall[2]

[1]Los Alamos National Laboratory
PO Box 1663, MS A143, Los Alamos, NM 87545
martz@lanl.gov

[2]Atomic Weapons Establishment
Aldermaston, Reading, England
Kevin.Marshall@awe.co.uk

ABSTRACT

MCNP6 has been extended to include a new capability that permits tracking of neutrons and photons on an unstructured mesh embedded as a mesh universe within its constructive solid geometry capability. Our mesh geometry was created through Abaqus/CAE using its solid modeling capabilities. Monte Carlo transport results are calculated for mesh elements using a path length estimator while particles track from element face to element face on the mesh. This paper presents some performance comparisons for the initialization and calculation phases of two well-known benchmark problems using both the legacy and unstructured mesh tracking capabilities. For detailed geometries, unstructured mesh initialization is always faster. For very detailed geometries where the models are comparable, the unstructured mesh capability is faster than the legacy geometry capability.

Key Words: Monte Carlo, MCNP6, CAE, unstructured mesh, geometry

1. INTRODUCTION

Los Alamos National Laboratory's (LANL) Monte Carlo N-Particle (MCNP) transport code has a more general geometry capability than has been available in most combinatorial geometry codes [1]. In addition to the capability of combining several predefined geometrical bodies, as in a combinatorial geometry scheme, MCNP6 gives the user the added flexibility of defining geometrical regions from all the first and second degree surfaces of analytical geometry plus elliptical tori and then combining them with Boolean operators. This decades-old constructive solid geometry (CSG) capability has been well-tested and verified. However, it has long been recognized that as the model complexity increases, the process of describing the geometry is difficult, tedious, time-consuming, and error-prone [2,3,4].

To address the difficulty of building complex geometry models, we (LANL) have developed a capability that uses an unstructured mesh (UM) representation of a geometry so that particles track directly on that mesh once it is imported into MCNP6 [5-7]. This approach was chosen not only to aid with the complex geometry modeling issue, but also to permit an easier interface for multi-physics calculations where these physics codes use an unstructured mesh. Other approaches [2,3,4,8] based on a computer-aided design (CAD) system to create a geometry model exist and address the issue of building complex geometry models for MCNP calculations, but do not readily support easy interfacing for mesh-based, multi-physics calculations or necessarily support state of the art results visualization. CAD-based tracking [8] has proven to be slower than tracking on CSG.

Computer aided engineering (CAE) tools such as Abaqus/CAE [9] and CUBIT [10] have the ability to generate an unstructured mesh representation of their solid models and are capable of generating mesh geometries for use in MCNP6. The degree of fidelity between the CAE representation of the geometry and the unstructured mesh is generally good and depends to a degree upon the user’s willingness, ability, and need to refine the model.

Previous work [5-7] described the UM capability and how it was implemented in a hybrid geometry approach in MCNP6. The Reference [5] work also presented results using three, geometrically simple benchmark problems and showed that the new UM capability works quite well and yields accurate results for both nuclear criticality calculations and fixed source calculations using the supported finite element types. Since these benchmark problems were relatively simple, geometrically, and have traditionally been modeled with a minimum number of cells and surfaces, the Reference [5] work was not able to compare code performance with consistent geometrical representations; this current work does make that comparison for two of the problems presented in the previous work.

In addition to the previous work [5] with the simple benchmark problems, work with more detailed models at LANL showed some interesting performance results. When CSG and UM models of the same, complex system were run on our high performance computing systems, we observed near identical runtimes even though there was a large difference in the cell and element counts. The CSG model had 2809 cells and the UM model had 170839 first order elements. Other than the geometry and the methods for obtaining results, the input specifications for the two calculations were the same. If the CSG model had the same number of cells as the UM model had elements (i.e., 170839), its runtime would have been much longer. An examination of the findings below should convince the reader of this statement.

Therefore, because of the performance results seen between models that were similar but not equivalent, we decided to undertake as fair a comparison as possible between the two geometry capabilities using the above mentioned benchmarks. Understanding performance with these simpler models should provide insight into using this method with problems requiring more complex geometries.

2. BACKGROUND

It should be fairly obvious that with two different geometry systems in MCNP6, the algorithms for problem setup and particle tracking are different. It is this difference that will be important in understanding code performance when different tracking methods are used. A couple important examples are discussed here to illustrate the differences. Before discussing performance results from these examples, we will discuss some crucial differences between these two geometry features.

During problem setup for the CSG geometry, each cell is checked and simplified, if possible, removing any redundant surfaces. Some of the operations in this process are n-squared where n is the number of CSG cells in the entire problem. There is no direct counterpart for this with the UM. Instead, each element's nearest neighbors must be found. This nearest neighbor search is currently done during particle tracking when the first particle enters an element. This could be done during mesh setup, but was initially judged to be more efficient if implemented for only those elements actively seen by the transported particles.

The nearest neighbor search is not an n-squared operation where n is the number of mesh elements in the entire problem, but rather an n-squared operation over the elements in its associated part. In our mesh treatment, nearest neighbors are only meaningful for elements in a part; one can think of the part as the smallest building block in the CAE solid model. The nearest neighbor search is conducted only for the elements in a part and once all of the element's neighbors are found, the search is terminated.

Given a particle's phase space description, the traditional particle tracking methodology in MCNP6 requires that the program select a minimum distance to travel from a distance to a boundary, a collision, a time cutoff, a variance reduction event, etc. If the distance selected is because of a cell boundary in CSG, the particle is advanced to the surface of that cell and the various distances are re-sampled. However, with the UM there are two types of boundaries: element boundaries and part boundaries. In the initial UM implementation, the boundary of interest was the part boundary; hence, many element boundaries could be crossed before a part boundary was encountered. We refer to this type of tracking as multi-tracking. This approach is acceptable as long as the material composition of each part is homogeneous. In the process of this investigation and to make as fair a comparison as possible between CSG and UM for this work, we have added the ability to turn the multi-tracking off. When this is done, one mesh element and one CSG cell are treated as identical in terms of cycling through the history transport loop. However, testing with multi-tracking off yielded little difference in results compared to running with multi-tracking on. For the rest of this paper, all UM calculations use multi-tracking on.

With the MCNP CSG capability, it is possible to model polyhedrons bounded by planes with an arbitrary orientation. Since finite elements of the first order tetrahedral type are guaranteed to have planar faces, it is possible to reproduce these finite elements in MCNP with cells defined by arbitrary planes [1]. That is what we chose to do for our comparisons in this work. From the same Abaqus input file used in our MCNP6 calculations with the UM, we were able to create an equivalent representation of the first order tetrahedra using arbitrary polyhedral cells (APC) with the CSG capability. In order to make it convenient to describe what is known as “the outside world” in MCNP (i.e., the phase-space outside of the geometry of interest), each benchmark geometry was placed in a box of air so that one macrobody surface represented the boundary between the geometry of interest and the outside world. All of the geometry inside this macrobody was described either with a mesh of finite elements or APC’s. These boxes were not fitted tightly around the spheres since space was needed to generate well-formed elements adjacent to the outer spherical surfaces. In the course of this work, the number of finite elements and APC’s were varied in order to look at code performance as a function of elements and cells.

3. THE BENCHMARK MODELS

We chose to revisit the simple Godiva criticality sphere benchmark [11] and Osaka nickel sphere benchmark experiment [12] problems in order to test tracking during a criticality and fixed source calculation, respectively. With both geometries possessing a highly curved outer surface, a large number of first order tetrahedra are required to accurately reproduce the volumes and obtain accurate results [5,6]. More details about the agreement of results between CSG and UM models can be found in the references [5,6], but is generally good, depending upon the type and number of elements/cells.

The Godiva benchmark is a simple, highly enriched uranium sphere of radius 8.7407 cm and density 18.74 g/cm^3. Traditionally, neutrons are tracked in batches using a power iteration method to calculate k-effective for this fissile system.

The Osaka benchmark is a simple nickel shell, density 8.85 g/cm^3, with inner radius 2.5 cm and outer radius 16 cm. Pulsed neutrons were generated at the center of the inner air cavity by the t(d,n)^{4}He reaction using a 245-keV deuteron beam. Neutron leakage is calculated at or beyond the outer nickel surface.

Since our previous experience with the code showed us that increased computational times would be needed as the level of detail in the models increased, the requested history conditions for both problems were altered from the Reference [5] work. These conditions will be discussed in more detail in the following calculation performance section.

All of the computer runs were performed with the sequential version of the code, optimized to level 1 with version 11.1.072 of the Intel Fortran compiler on a Linux cluster with dual Intel Xeon ES-2670 chips clocked at 2.6 GHz and possessing 2 GB RAM per core. The operating system was 64-bit Chaos. ENDF/B-VII cross sections were used for all nuclides.

4. INITIALIZATION PERFORMANCE

The problem setup time as a function of element or cell count is shown in Figure 1. Setup time is determined by subtracting the calculation time from the total runtime as determine by MCNP and displayed in its output file. This time is independent of the various conditions (number of histories) under which the models were subsequently run as described later in this paper.

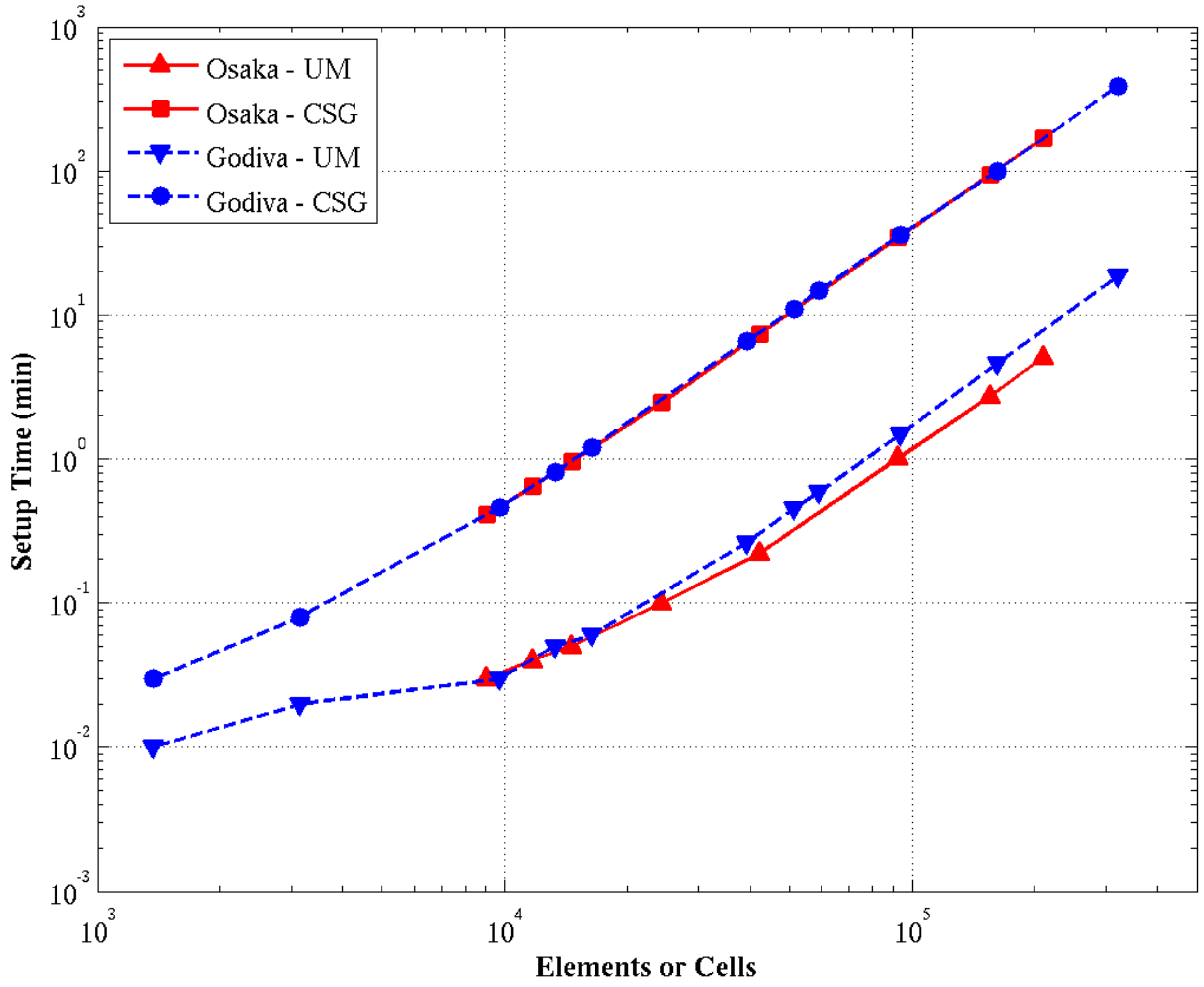

Figure 1. Problem setup time on one processor as a function of geometry detail for the Godiva and Osaka benchmark problems.

For large numbers of elements the UM models initialize ~40 times faster than the equivalent CSG models. This is primarily because the CSG setup must simplify and check the cell descriptions by removing redundant surfaces. This is an n-squared search operation where “n” is the number of surfaces in the entire problem; this is done for each cell. This operation is not needed in the UM initialization for the mesh elements. Most of the initialization time for the UM models is devoted to finding the element faces that are on the part surfaces. This is an n-squared operation where “n” is the number of elements in the part.

There is a slight difference in the setup time between the two different UM models for large element counts that does not appear in the corresponding CSG models. This effect is because of the different arrangement of mesh elements in the two models. In the Godiva model there is one constraining spherical surface that separates the uranium region from the air region. In the Osaka model there are three constraining spherical surfaces: one between the nickel region and the outside air region, one between the nickel region and the inner air cavity, and one that subdivides the nickel region into two shells. The meshing algorithms in Abaqus/CAE must place nodes on these surfaces in the process of meshing each region. Sometimes during this meshing process, elemental arrangement is more advantageous in some models for certain operations. This will be apparent in some of the tracking comparisons made later in this paper.

Figure 1 shows the results where one processor is used in the problem initialization phase. Input for CSG geometries can only be processed sequentially. Input for UM geometries can be processed in parallel with the message passing interface (MPI) version of the code, provided the geometry consists of multiple parts so that each part may be processed on its own slave node. If the models considered here were constructed from multiple parts, the wall clock time for problem setup would be reduced proportional to the number of parts and the UM curves in Figure 1 would appear lower in the plot and their slopes would not be as steep. If this was to occur, the label “Setup Time” should be more appropriately replaced with “Wall Clock Setup Time”. The obvious consequence of this is that with the UM there is less idle processor time in an MPI run while the slave nodes wait for the problem initialization phase to complete.

5. CALCULATION PERFORMANCE

The calculation times as a function of elements or cells for the Godiva and Osaka benchmarks are shown in Figures 2 and 3, respectively. The calculation time is that determined by MCNP and presented in its output file. The Godiva kcode calculation was run with 5000 histories per cycle for 120 total cycles while only tracking neutrons. The Osaka fixed source neutron calculation was run for 100,000 histories.

Both Figures 2 and 3 show that for large element counts, the calculations with the UM models take less time than the corresponding CSG model. For small element counts, the calculations with the CSG models are faster. Somewhere between 10,000 and 100,000 elements the performance curves cross, but the crossing is different in the two figures. If photons are tracked along with the neutrons in the Godiva kcode calculation, the resulting set of performance curves would be shifted higher than the curves in Figure 2 in such a way that their intersection point is to the left of the intersection point in Figure 2 by ~ 20,000 elements.

Obtaining an understanding of the code's tracking performance from these two benchmark problems is difficult since they are different types of problems; that is, they use different physics. However, except for number and orientation of elements/cells, the underlying geometries are similar. By using MCNP's void card option to replace all problem materials with void and by using an isotropic, fixed point source at the same location relative to the origin in each, both problems can be reduced to one of ray-tracing the geometry from the center outward. This action essentially minimizes any time spent by the code in physics routines and the calculation time is now mostly attributed to tracking through the geometry. This approach will give us a better comparison between the two tracking methods.

Figures 4 and 5 provide the performance curves for the voided geometries described above with 100,000 and 1 million histories, respectively. As expected, the CSG performance curves for the two models are almost identical. The UM performance curves agree, but not quite as well as those for the CSG models, primarily because of the different constraining surfaces used when generating the mesh. Therefore, going forward with this paper, studying one model, the Godiva model, under different conditions should be sufficient to understand the performance issues.

A comparison of the curves between Figures 4 and 5 reveals that as the number of histories increases, the intersection point of the two lines moves from left to right in the plots. That is, as more particles are tracked the UM calculations remain less efficient than the corresponding CSG calculations, except for perhaps very detailed geometries. Running more histories with the CSG allows the code to take advantage of its previously created "other side cell" list so that calculation efficiency increases as it continues to run. More discussion on the “other side cell” lists follows.

A comparison of the end points (left and right) for the Godiva curves in Figures 4 and 5 is rather revealing. Between these two figures, the times for the calculations that mark the left hand ends are about a factor of 10 different and are consistent with the number of histories that are run. This is not the case for the points that mark the right hand ends of the Godiva curves. Here, there is roughly a 20% increase in calculation time as the number of histories is increased by a factor of 10. When the two different CSG calculations are studied with a code profiler, the times in the routines that take the most time scale by roughly the factor of 10, except for the routine that selects the next cell into which the particle tracks. Here we see an increase that is in line with the increase that we see when results from Figures 4 and 5 are compared. This contrast in the differences between the scaling on the left side of the figures versus the right side is attributed to how quickly the “other side cell lists” are built. This is discussed below. A similar discussion exists for the UM calculations when they are studied with a code profiler; the tracking routines scale by a factor of 10 and the nearest neighbor search routines are the largest single time sinks and are approximately the same between the two calculations.

An alternative way to show the performance of the UM models relative to the CSG models is to divide the UM calculation time by the corresponding CSG calculation time, holding the number of histories constant, to generate calculational ratios as a function of element or cell count. If the ratio is greater than 1, the CSG calculation is more efficient and the reverse is true if the ratio is less than 1. Figure 6 shows these ratios for the Godiva benchmark mesh when the ratios are calculated after certain numbers of histories have been accumulated. In this figure, those points with values less than 1 correspond to situations where the UM calculations are faster or more efficient than the corresponding CSG ones. It appears from the asymptotic nature of three of the curves in Figure 6 and the behavior of the fourth one (10 million history run), a point will be reached with the increasing geometric detail that the UM tracking will always be faster or more efficient than the CSG tracking unless an infinite number of histories is run.

In both of the tracking methodologies, the code must determine the closest intersection point in the current element/cell for the direction that the particle is moving. Then, if the particle leaves the element/cell, it must determine the element/cell on the other side of the surface containing the intersection point. Since the elements and cells in these models are tetrahedra, the different intersection routines are similar in computational complexity. What is different between these methods is how the code determines what is on the other side of the surface containing the intersection point. With the CSG capability, the code must do an n-squared search over all cells to find the appropriate one on the other side. Once it has found the new cell, the code adds it to the “other side cell list” for the old cell. When a new particle enters the old cell (or the same particle reenters), the code will first check the “other side cell list” before attempting the n-squared search over all of the cells. Upon problem startup these lists are empty and MCNP must work to populate them. As the lists grow, MCNP tracks more efficiently when using the CSG capability.

When a particle is leaving an element in the unstructured mesh, no n-squared search over elements is needed. Rather, the code checks the element’s nearest neighbor list, where in this case there is at most one element on each face of the tetrahedra (there are no nearest neighbors for surface element faces). If the nearest neighbor list does not exist, the code can quickly build it. However, MCNP currently does not select intelligently from this list; this will be corrected at a later date for a future version of the code provided any additional memory requirements for implementation are not prohibitive.

The points farthest to the left on the curves in Figure 6 correspond to models with 1375 elements or cells. With the large number of histories that have been run to generate these curves, it is fairly safe to assume that the “other side cell lists” have been created quickly and the ratios we see for these points are a good indication of the maximum speed or efficiency differences between these tracking methods. Therefore, based upon these void model calculations, the greatest difference between the UM and CSG tracking methods is a factor of 6.0 to 6.8, meaning that the UM method is slower.

When more meaningful, non-void problems are run using the mesh, such as the original kcode calculation or a fixed source calculation with a 6.1 MeV gamma point source placed at the model’s center, the tracking routines compete for time with the other physics routines and the calculation ratios change. Examples are shown in Figures 7 and 8. Consider the cases with the coarsest mesh – 1375 elements or cells – when viewed with a code profiler. The most dominant routines for the CSG kcode calculation are the ones for tallying results and finding the neutron cross sections. Since there are many banked particles in the photon calculation because of bremsstrahlung, annihilation, and fluorescence the routine that returns particles from the bank dominates this calculation. The most dominant routines for both of the UM calculations are the ones associated with UM tracking. Therefore, different weighting of the various routines influence the overall calculation ratios, as defined in this work, so that for problems with materials the speed difference between the UM capability and the CSG capability is not nearly as pronounced for coarse mesh problems as first indicated by the void problems. As the geometric detail increases by way of more elements/cells, the UM capability surpasses the CSG capability sooner in terms of efficiency.

Overall, a comparison of the curves in Figures 7 and 8 shows that they quickly drop off as a function of elements/cells when fewer histories are run. This is as expected for reasons regarding the “other side cell lists” explained above. The overall slope of the fixed source gamma case curves is not that different between Figures 7 and 8. This is because there is a considerable number of secondary particles produced that help build the “other side cell lists” sooner than what is exhibited for the other cases in these figures.

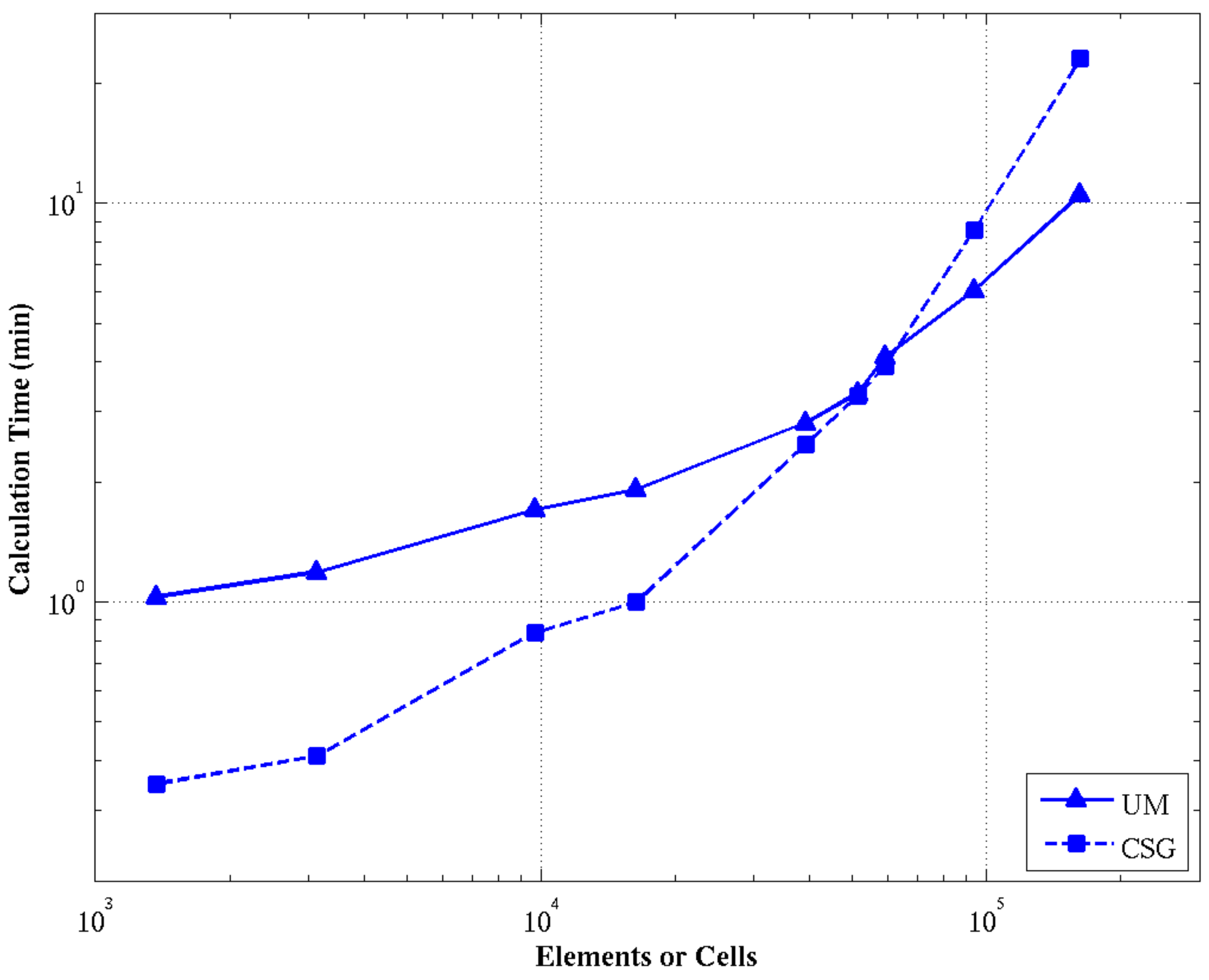

Figure 2. Calculation time on one processor as a function of geometry detail for the Godiva benchmark problem. 120 total cycles with 5000 histories per cycle.

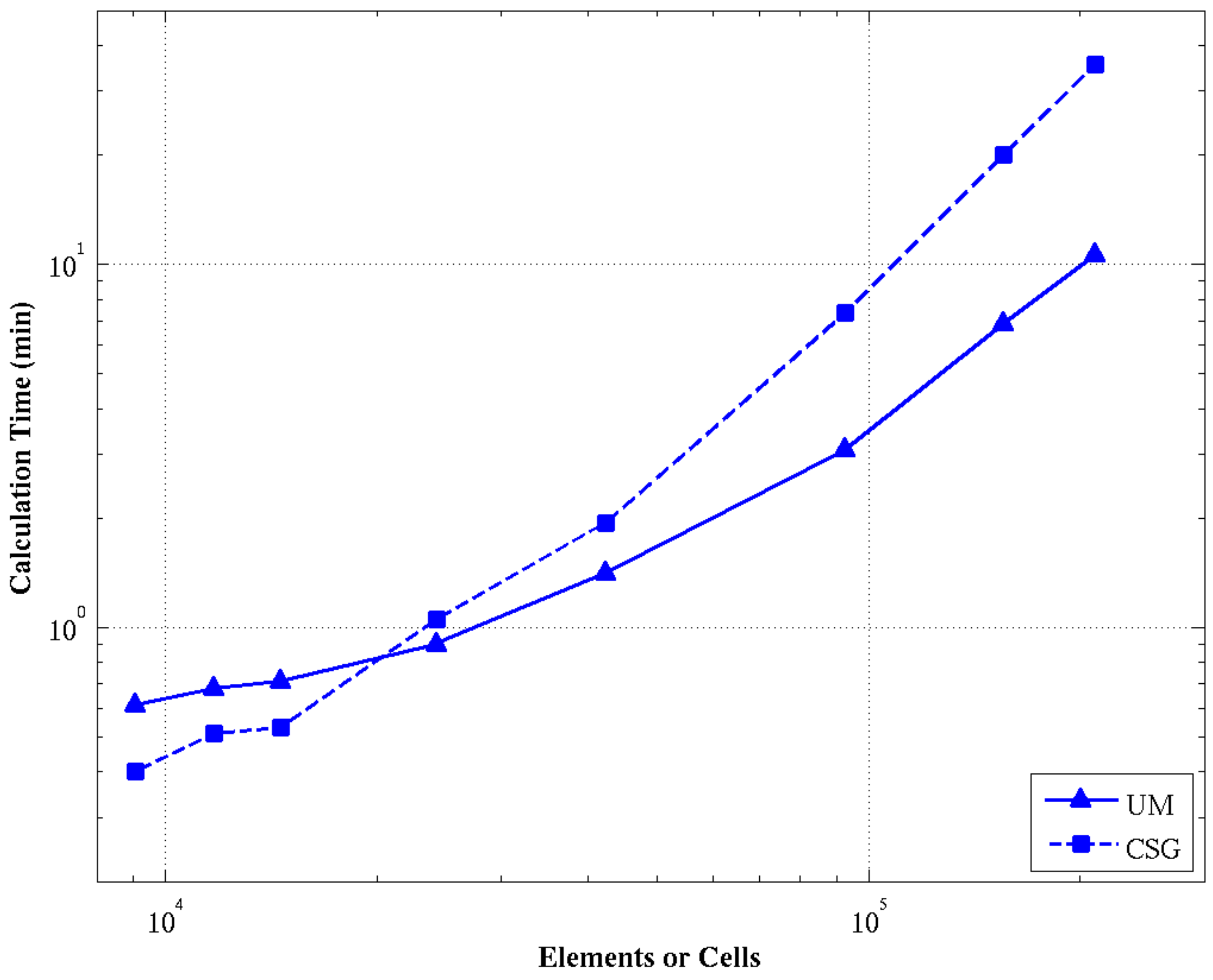

Figure 3. Calculation time on one processor as a function of geometry detail for the Osaka benchmark problem. 100,000 total histories.

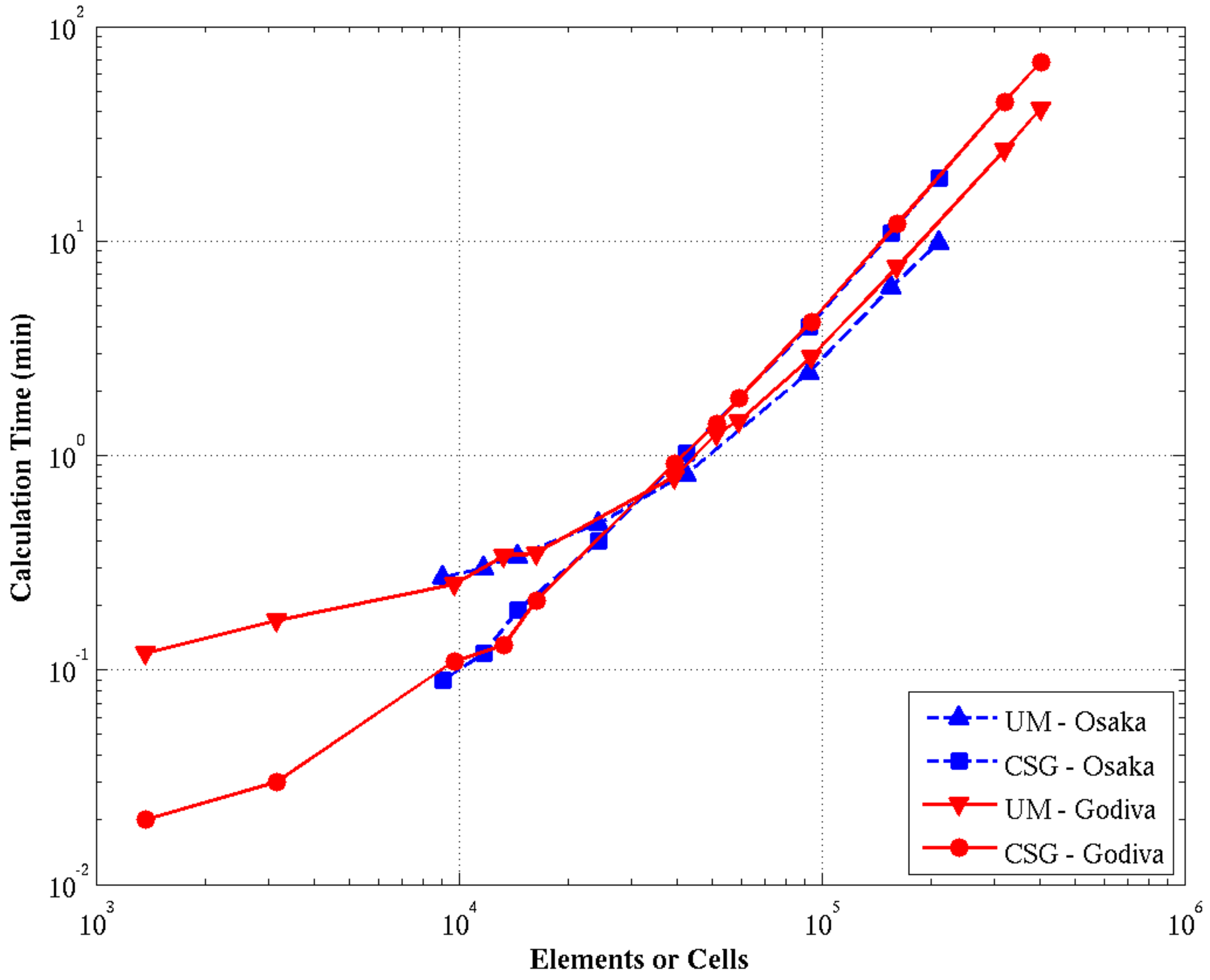

Figure 4. Calculation time on one processor as a function of geometry detail for the benchmark problem with voided material and an isotropic point source. 100,000 total histories.

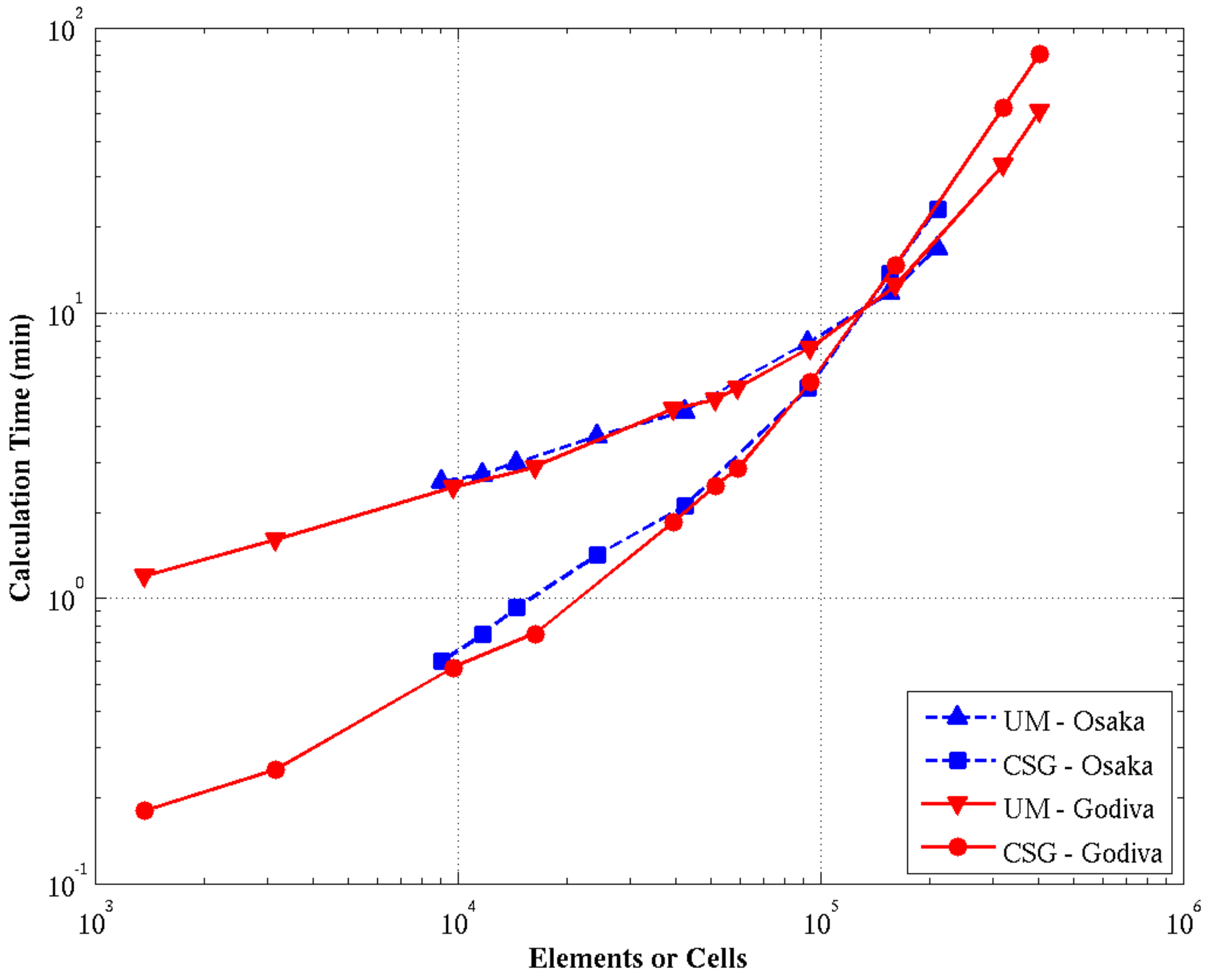

Figure 5. Calculation time on one processor as a function of geometry detail for the benchmark problem with voided material and an isotropic point source. 1 million total histories.

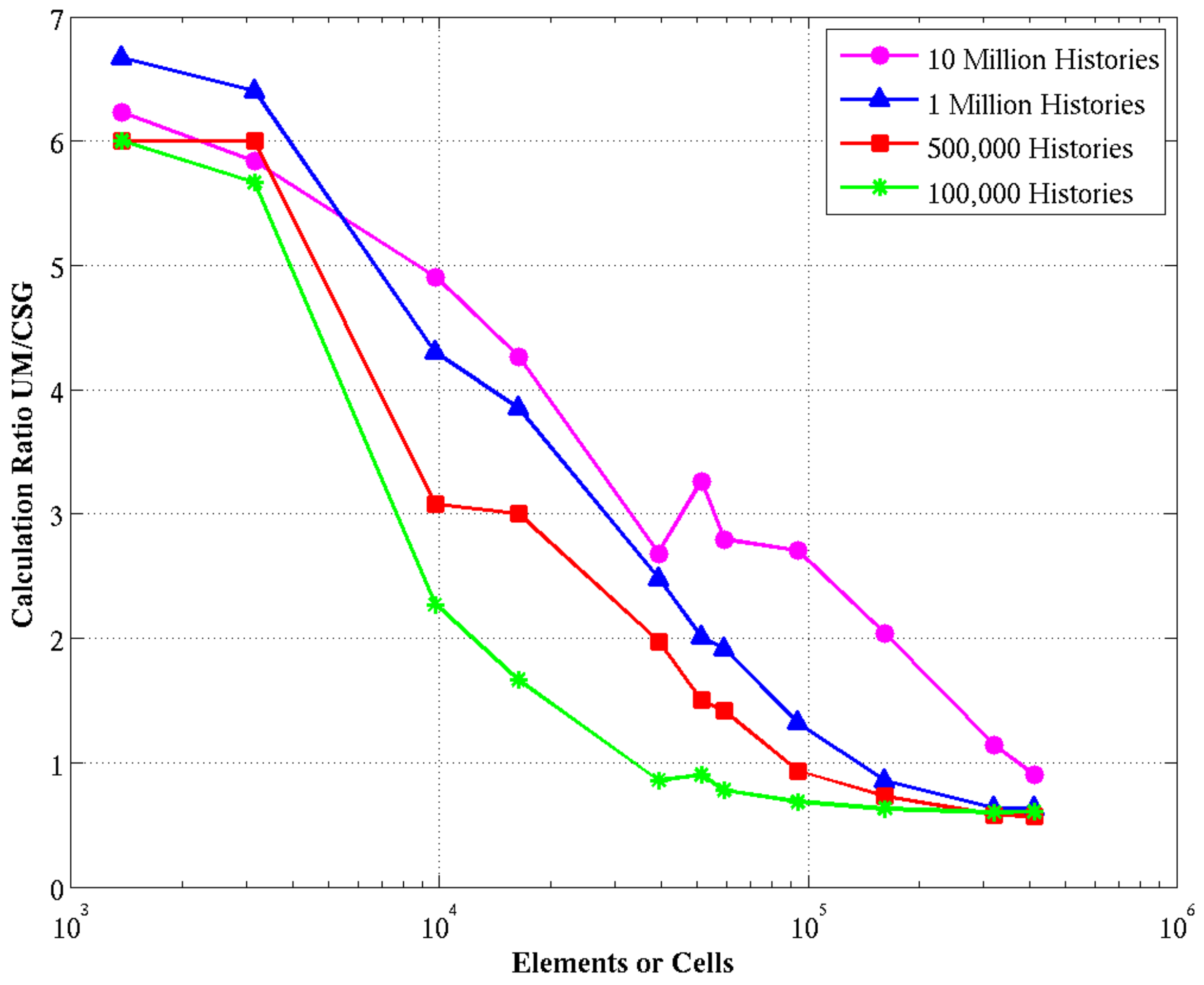

Figure 6. Ratios of calculation time (UM / CSG) on one processor as a function of geometric detail using the voided Godiva benchmark geometry.

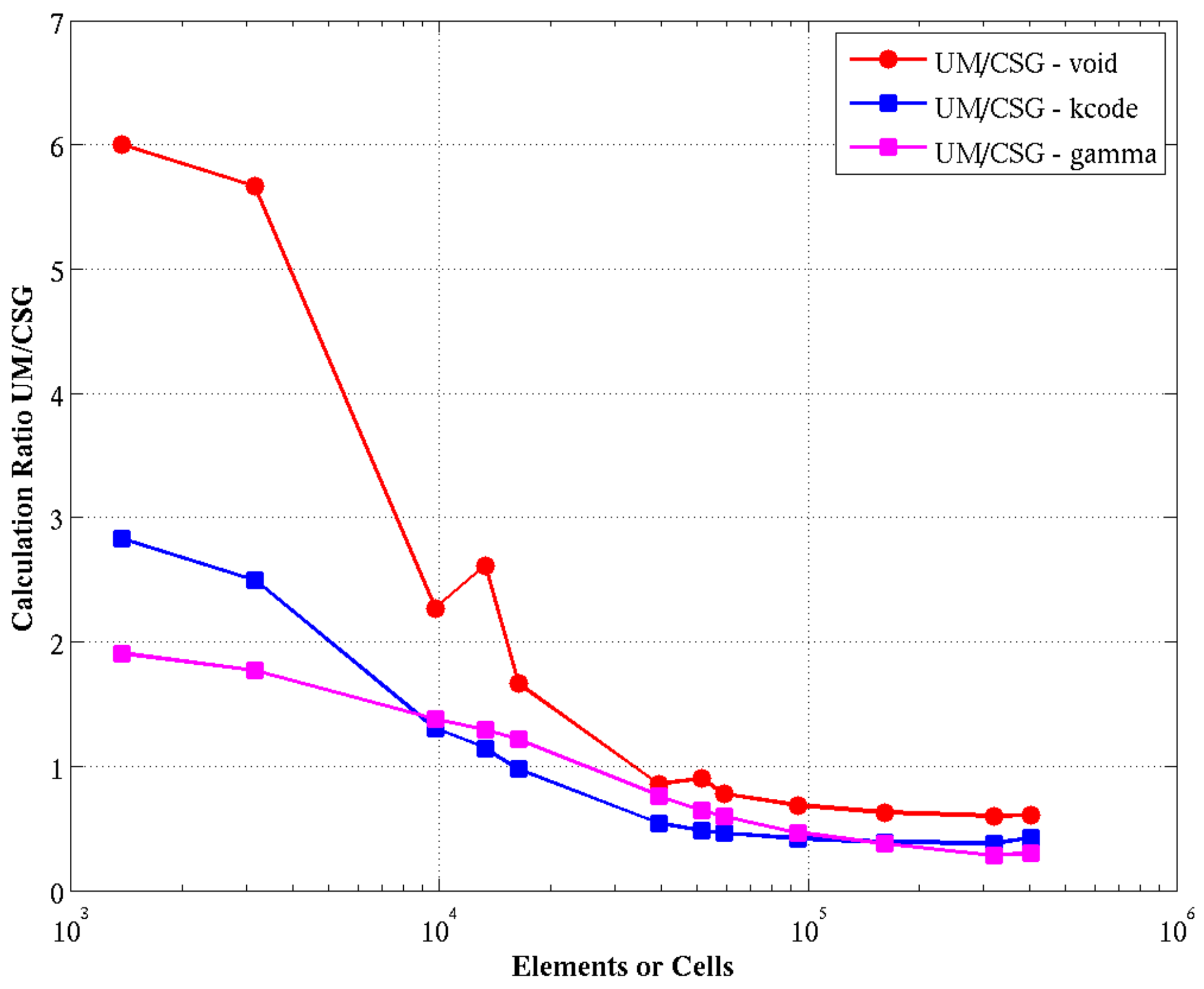

Figure 7. Ratios of calculation time (UM / CSG) on one processor as a function of geometric detail and problem type using the Godiva benchmark geometry. 100,000 histories for each calculation.

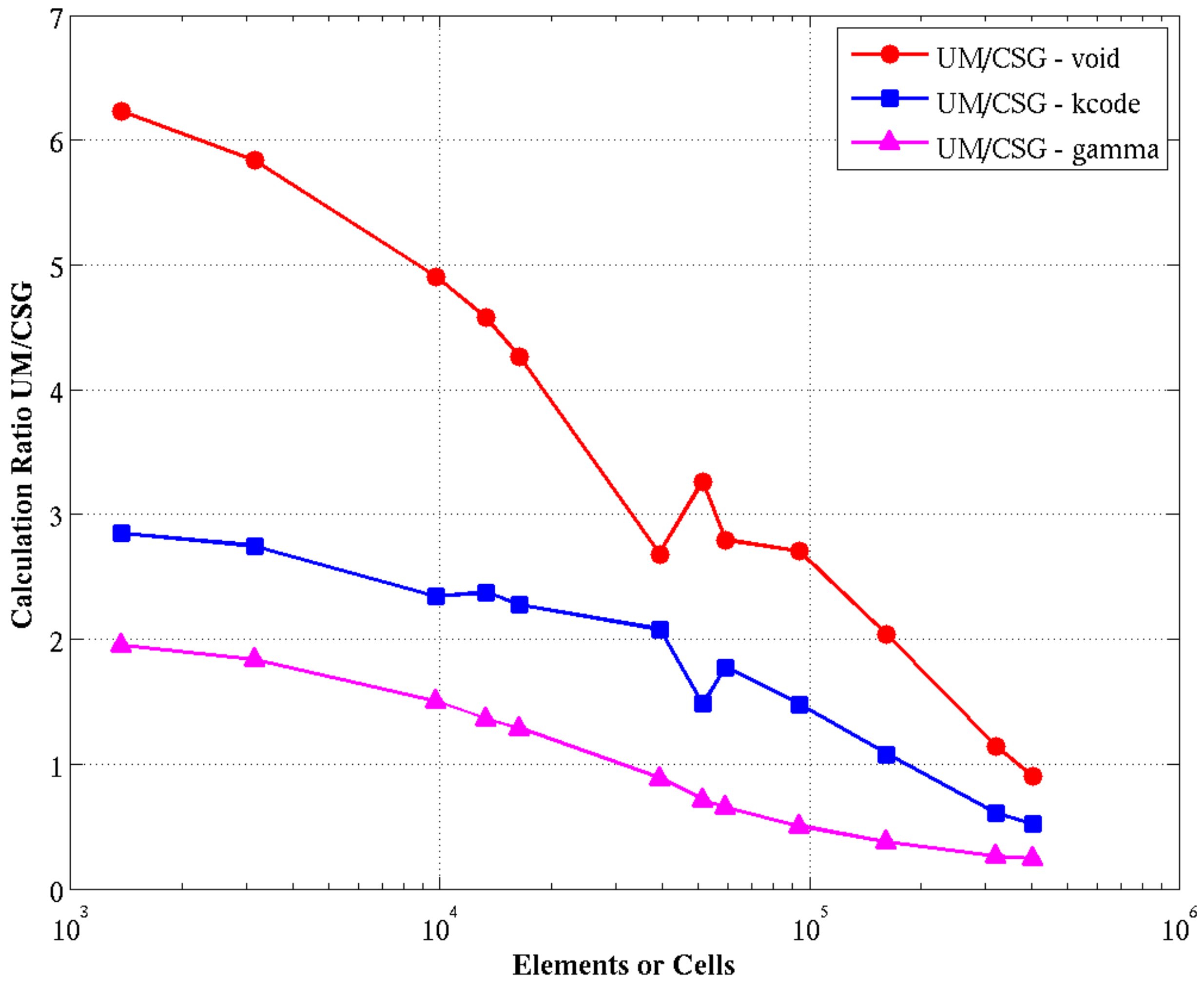

Figure 8. Ratios of calculation time (UM / CSG) on one processor as a function of geometric detail and problem type using the Godiva benchmark geometry. 10 million histories for each calculation.

6. CONCLUSIONS

A fair comparison of MCNP6's two geometry capabilities has been presented by analyzing code performance on two well-known benchmark problems, each with varying numbers of histories, geometry refinement, and problem type. Results from the problems analyzed here show that shorter problem initialization times occur when the unstructured mesh feature is used. This behavior is dependent primarily on the number of elements (either finite elements or traditional MCNP cells) in the geometry and should scale no matter how complex (vs. simple benchmarks) the geometry is. In this work for the sake of a fair comparison we were restricted to use APC's; MCNP's setup time for any of its traditional cells is similar.

When a small number of elements/cells is adequate to describe the geometry, the CSG capability outperforms the UM capability because of its ability to quickly build the "other side cell lists". If the "other side cell lists" do not exist, MCNP must perform an n-squared search over all cells to construct them. In situations where the CSG capability is still building these lists, the UM capability has shown better performance. Users whose models require very large element/cell counts (> 1million) may be curious to know which method is more efficient. The answer is dependent on the number of histories required to reach the desired precision. Based upon the results presented here, it is doubtful with these very large element/cell counts that the lists will be built in time to make the CSG capability more efficient than the UM capability.

The majority of this work was based on single part models. Extrapolation of these results to complex, multi-part models may not be straightforward, but is the next logical step to pursue and prove. In our limited experience, initialization times remain reasonable if element counts do not greatly exceed ~50,000 elements per part and execution times can vary by +/- 20%, Reference [5], if the models are constructed from multiple parts in lieu of one large part.

With the addition of the UM capability, MCNP users have another method in which to construct geometry models. The user will need to decide which of these several available methods is best for their problem under consideration. While the UM capability may not be the method of choice for simple geometry problems, it was used with these simple benchmark problems to demonstrate and understand performance issues. We expect the UM feature to perform in a similar fashion with more complex problems. Our in-house, complex-model work mentioned in the introduction to this paper leads us to believe that the findings of this paper will hold in these situations. Given the ability to more easily create complex models with CAE tools, no matter the element count, and the much shorter problem initialization times, even with a factor of 2 to 3 poorer calculational performance, the UM capability may ultimately be the better overall option to use.

7. FUTURE WORK

This work pointed out an issue in the unstructured mesh tracking routines regarding the lack of intelligent selection of the appropriate elemental intersection face; this should be corrected in the near future, provided the solution does not significantly impact memory requirements for implementation. The correction should only make the unstructured mesh capability faster than what currently is in the MCNP6 Beta 2 release.

Recent work by developers advocating a CAD-tracking methodology as the solution to generating and using complex geometry models in Monte Carlo calculations [8] shows that technology to be a factor of ~5 slower than tracking with the legacy CSG methodology. At this time it seems highly unlikely that a comparison similar to the one presented in this paper can be made between the UM and the CAD-tracking methodology of Reference [8], where tracking takes place on the CAD geometry. However, it would be informative to have as direct a comparison as possible between the CAD-tracking methodology and the UM capability. Based on what is known today, there probably will be situations where one geometry treatment will be faster than the others. However, where detailed results are needed for multi-physics work or visualization, it appears that the UM geometry will be the one to use since results on the mesh are produced almost automatically for little additional performance penalty. Monte Carlo practitioners should be aware of the pros and cons of each and then use the method that is best for their needs.

8. ACKNOWLEDGEMENTS

MCNP unstructured mesh development over the life of this work is due to funding from the US Department Of Energy's NNSA – Advanced Strategic Computing (ASC) and Engineering Campaign 7 (C7) programs. We thank these sponsors.

REFERENCES

1. X-5 MONTE CARLO TEAM, "MCNP – A General Monte Carlo N-Particle Transport Code, Version 5, Volume I: Overview and Theory," LA-UR-03-1987, Los Alamos National Laboratory (April 2003).
2. Timothy J. Tautges, Paul P. H. Wilson, Jason A. Kraftcheck, Brandon M. Smith, Douglass L. Henderson, "Acceleration Techniques For Direct Use Of CAD-Based Geometries In Monte Carlo Radiation Transport," *International Conference On Mathematics, Computational Methods & Reactor Physics*, Saratoga Springs, New York, May 3-7, 2009, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2009).
3. Yican Wu, Qin Zeng, and Lei Lu, "CAD-Based Modeling For 3D Particle Transport," *International Conference On Mathematics, Computational Methods & Reactor Physics,* Saratoga Springs, New York, May 3-7, 2009, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2009).

4. D. Groβe and H. Tsige-Tamirat, “Current Status of the CAD Interface Program McCad for MC Particle Transport Calculations,” *International Conference On Mathematics, Computational Methods & Reactor Physics*, Saratoga Springs, New York, May 3-7, 2009, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2009).

5. Roger L. Martz, “MCNP6 Unstructured Mesh Initial Validation And Performance Results,” to be published in Nuclear Technology, American Nuclear Society, LaGrange Park, IL.

6. Roger L. Martz, Tim Goorley, and Ryan Clement, *Implementing MCNP’s 21st Century Geometry Capability: Requirements, Issues, and Problems*, LA-UR-09-07884, presented at the ANS RPSD 2010 Topical Meeting, Las Vegas, Nevada, April 19-22, 2010, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2010).

7. Karen C. Kelley, Roger L. Martz, and David L. Crane, *Riding Bare-Back On Unstructured Meshes For 21st Century Criticality Calculations*, PHYSOR 2010 – Advances in Reactor Physics to Power the Nuclear Renaissance, Pittsburgh, Pennsylvania, May 9-14, 2010, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2010).

8. Tim D. Bohm, S.T. Jackson, M.E. Sawan, and P.H.H. Wilson, “CAD-Based Monte Carlo Code Using Fusion-Specific Experiments,” Nuclear Technology, Vol. 175, July (2011).

9. Dessault Systemes Simulia, Inc., “ABAQUS USER MANUALS, Version 6.9,” Providence, RI (2009).

10. CUBIT Geometry and Meshing Toolkit, Version 13.1, October 2011, Sandia National Laboratory, http://cubit.sandia.gov .

11. *International Handbook of Evaluated Criticality Safety Benchmark Experiments*, NEA/NSC/DOC(95) 03 Vol II, September 2009 Edition.

12. Akito Takahahi, et. al., Osaka *Nickel Sphere Benchmark Experiment (OKTAVIA)*, Shielding Integral Benchmark Archive and Database, SINBAD_2010.05, Radiation Safety Information Computational Center, Oak Ridge, TN.