Formal quality control for a proton Monte Carlo system in radiation therapy

Formal quality control for a proton Monte Carlo system in radiation therapy

J Perl[1]
SLAC National Accelerator Laboratory, Menlo Park, CA, USA

E-mail:perl@slac.stanford.edu

Abstract: TOPAS (TOol for PArticle Simulation) is a Monte Carlo particle transport tool being released to a wide variety of proton therapy users worldwide. Because TOPAS provides unprecedented ease in 4D placement of geometry components, beam sources and scoring, including options to place geometry components, beam sources or scorers within each other, Quality Control (QC) for TOPAS is both critical and challenging. All simulation details (geometry, particle sources, scoring, physics settings, time-dependent motions, gating, etc.) are specified in the TOPAS Parameter Control System (which catches many user errors). QC includes Unit and End-to-End Testing. Each code unit is tested (each geometry component, particle source option, scoring option, etc.) and these unit testing procedures are shared with end users so they can reproduce tests. End-to-End testing of several full clinical setups is routinely performed. End-to-End testing presents a challenge since one cannot anticipate all the ways users will combine TOPAS flexible units for their specific project. Automated checking catches geometry overlaps and some other problematic setups, but one can never rule out the potential for problems when users combine units in new setups. QC is ultimately a partnership between the tool developer and the user. Key is that the developer be clear to the end user about what has been tested and what has not.

1. Introduction

Monte Carlo particle transport simulation (MC) codes have become an important tool for radiation therapy research and practice. As the variety and complexity of radiation therapy modes has increased, and as therapies have moved from simple to highly conformal to four dimensional (4D), 3D plus time, both the value of and the requirements on MC have increased. We present some thoughts on quality control (QC) for highly configurable and 4D MC codes. This work has grown out of five years of effort to develop the TOPAS Tool for Particle Simulation [1], a project to make Monte Carlo particle transport simulation faster and easier to use for proton therapy. The conclusions we draw here are not specific to TOPAS. Rather we use the TOPAS experience to raise QC issues that apply to any simulation code, especially those that are highly configurable and 4D.

TOPAS is an NIH funded project of the Massachusetts General Hospital, the University of California at San Francisco and the SLAC National Accelerator Laboratory. TOPAS lets you model a treatment head, import a patient, simulate all kinds of beams, has advanced scoring and graphics, and, because proton therapy involves so many moving parts, TOPAS is, from the ground up, 4D. Because TOPAS provides unprecedented ease in 4D setup, QC for TOPAS is both critical and challenging.

[1] To whom any correspondence should be addressed.

2. 4D Behaviors

TOPAS lets you place geometry components, particle sources and scoring regions in 3D, that is, each component can have its own translation and rotation specified however the user wishes relative to its mother component. TOPAS allows one to simulate an infinite variety of setups with just one pre-compiled software application, but different control files. All simulation details (geometry, particle sources, scoring, physics settings, time-dependent motions, gating, etc.) are specified within a TOPAS Parameter Control System, which was designed using principles of safety engineering to help prevent user errors. Automated checking provided by the underlying Geant4 toolkit [2] catches some geometry overlaps and some other problematic setups, but TOPAS adds many layers of additional safety. The system not only provides a unified repository for all parameters, but provides a unified manner in which parameters can be set to change over time, that is, a unified approach to 4D behavior [3].

Supported 4D behaviors can go beyond simple motion to include multiple moving parts, motions within motions, time-dependent shape changes, patient motion, time-dependent electric and magnetic fields, beam current modulation, sources (even multiple sources) that turn on and off, and gated scoring (figure 1). All of this can be put it together into some very complex systems, like the full IBA double-scattering nozzle at the Francis H. Burr Proton Therapy Center (figure 2) which combines moving range modulator wheels (RMW), beam current modulation, adjustable second scatterers and jaws and patient-specific apertures and compensators built on the fly, different for every field and every patient.

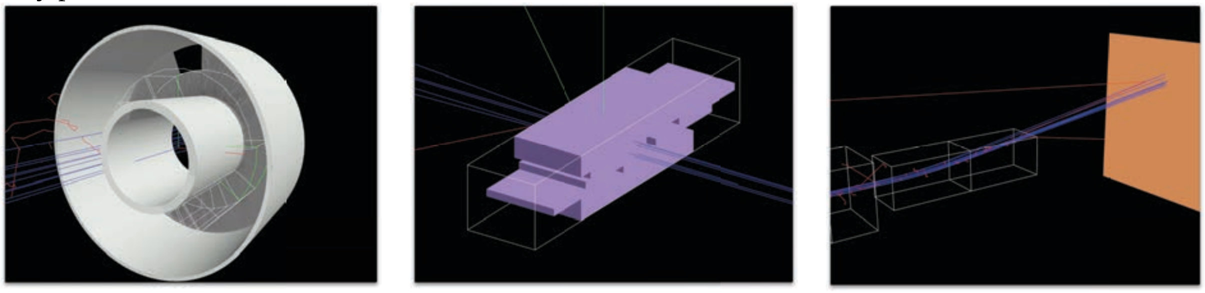


Figure 1. Components with 4D behaviors. a) Rotating Range Modulator Wheel. b) Dynamic Multileaf Collimator (simplified here to only three leaves). c) Time-varying dipole for magnetic beam steering.

For a fully configurable 4D simulation system, the user should be able to vary almost any parameter over time. The goal is to allow extreme flexibility of 4D behaviors so users can test entirely novel hypotheses. So, for example, TOPAS allows a user to change the angular extent of a cylinder over time even though we have yet to find a reason that one might wish to do this.

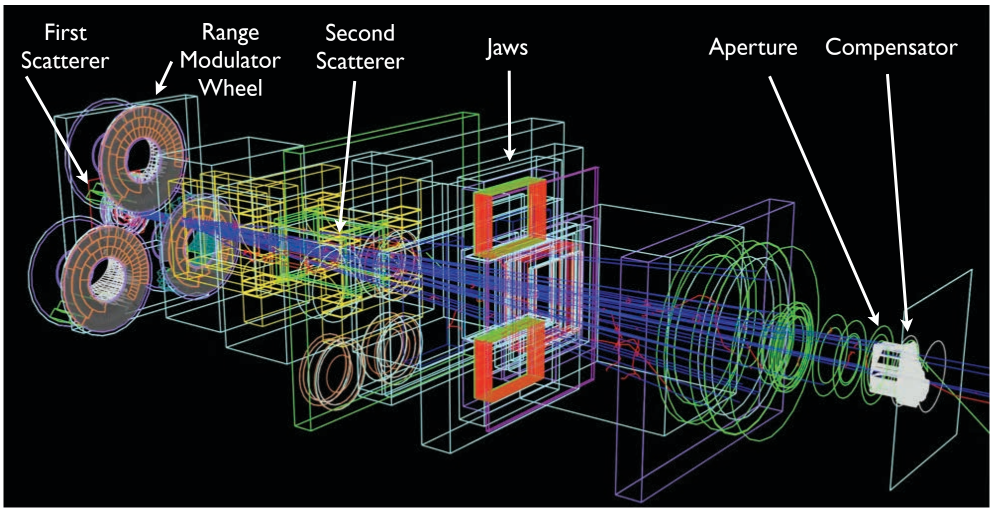


Figure 2. Proton gantry at the Francis H Burr Proton Therapy Center in double-scattering mode. Protons move left to right (blue lines). Secondary particles include electrons (red) and gammas and neutrons (green).

3. Limitations of 2 and 3D Visualization

4D behaviors are more difficult to test than 3D. There are ways in which an MC may appear to work correctly in 4D while in reality it does not. An example is in the use of visualization tools. TOPAS' underlying general-purpose MC toolkit, Geant4, contains an extensive visualization system [4] and includes capabilities for 4D OpenGL viewing. While Geant4's 4D graphics can be compelling, visualization of moving components do not necessarily prove the correctness of 4D scored results. Key is that visualization code is not the same as MC particle transport code. Visualization has the luxury of working slowly, at the sluggish pace of the human visual system. To produce visualizations, Geant4 follows the most straightforward approach, recalculating the whole image after every motion. But Monte Carlo transport needs to work much faster. One rarely wants to simulate just a few histories, but rather millions or billions. So MCs incorporate speedup strategies. We will discuss one that is particular to Geant4, the "Smart Voxel" system, but every code fast enough for therapy calculations has speedup techniques, and all such techniques must be re-considered in 4D.

Any general purpose transport code spends a lot of time trying to figure out where a particle will go when it leaves a volume - which of many other volumes will be the next one hit. There can be thousands of such volumes, and their shapes can make the “Am I going to hit you next?" question mathematically complex. Figure 3 shows an example of such a collection of volumes (the figure does not represent any actual therapy system but is just a random set of somewhat complicated objects - a few of the many solids supported by Geant4). The geometrical situation is complicated by factors beyond the complexity of the individual solids - the trajectory may be curved by a magnetic or electric field, there may be coulomb scattering, and all of this has to be solved in 3D.

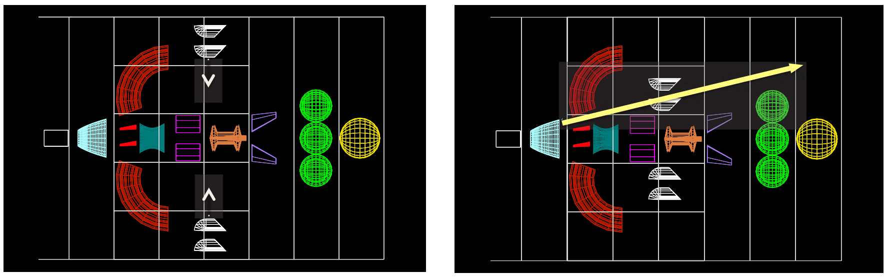


Figure 3. Geant4 smart voxel system.
Left: Arrows indicate volume moving across boundaries.
Right: Particle shown as yellow track will not "see" white volumes.

Codes use tricks to limit how many volumes must be interrogated. For the Geant4 case, before the first particle is tracked, Geant4 slices the world into what it calls "smart voxels" (the vertical lines slicing figure 3). If more than a few volumes are in any given "smart voxel", Geant4 slices the world further along different axes (the horizontal lines slicing figure 3). A lookup table is generated recording which volumes are in which of these voxels. At any given time, Geant4’s tracking need only interrogate the few volumes in the current voxel. Or if the trajectory seems to exit the current voxel, it just interrogates the volumes in the next voxel. Generating the lookup table takes some time, but it only needs to be done once at the start of the simulation, so the time burden is not significant.

The Geant4 smart voxel system works very well. Not only does it provide efficient tracking in complex arrangements of varied 3D solids, but it also works as well in regular 3D grid geometries such as a set of DICOM pixels [5]. But if a volume moves into a voxel that it wasn't in when the lookup table was generated, the trajectory will pass straight through the volume without interacting. As far as the smart voxel system is concerned, the volume is still where it was at the start of the simulation. This means one could, for example, have an MLC leaf move into the beam path yet have the beam continue to propagate as if the leaf were not there. Of course this can be fixed. The solution is to re-calculate the relevant part of the lookup table when a motion has occurred. In Geant4, it is left to the user code to explicitly trigger this re-calculation at appropriate times. The underlying Geant4 toolkit does not automatically recognize that such a change might be needed (as it would be time-consuming for Geant4 to continually check for such changes).

4. Implications for Caching

The Geant4 smart voxel system described above is just one example of "caching," a speedup technique critical to all fast MC codes. Code developers note which operations are costly and repetitive and cache results to avoid re-computation. An example comes when a code needs to check whether a particular scoring volume has been intersected. Performing a string comparison to check the name of a volume is a slow operation compared to checking the pointer to this volume in memory (an integer comparison). Accordingly, codes may look up pointers to volumes of interest only once, then cache these pointers for later rapid comparison. A 4D code may at times need to delete and re-create volumes to account for the widest possible set of 4D variations. After such changes, while the new volume may still have the same name, the memory pointer may be changed.

Another example of caching could be for storing the result of a volume calculation. Dose calculations require knowing the volume of the scored section. Such a calculation may involve time-consuming trigonometry if the volume is, for example, an arbitrary wedge of a cylinder. Because the same scoring area may be used many times, code may be written to calculate this volume only once, then cache this result. But if a 4D behavior alters this volume, the cached result may become incorrect. 4D code must therefore take care with caching. Caching itself is a 4D behavior.

Particle splitting (such as Bremsstrahlung Splitting), directional biasing and geometrical biasing [6-9] are additional examples of speedup techniques where extra care must be taken in the 4D case. Such techniques are likely to involve special actions to be taken in certain regions of a moving geometry or may involve calculations with direction vectors that may be varying over time.

5. Importance of 4D Tests

In the smart voxel case described above, the potential errors from mishandling 4D were obvious - a particle either encountered a volume or missed the volume completely. But in real world applications, effects are more subtle. A classic example from proton therapy concerns ribs motion during 4D lung treatment. A beam targeted on a lung tumor may encounter only soft tissue on its way to the lung in some breathing phases, but rib on some other breathing phases. Not only is the presence or absence of rib dependent on breathing phase, but even the angle of the rib relative to the treatment beam may vary. As the proton range is highly dependent on atomic structure of materials encountered, this subtle motion can have profound effects on the end of proton range. A 4D visualization tool that simply confirms that anatomy is moving is not a sufficient QC aid. There must be tools that confirm that the code correctly accounts for all 4D changes. Figure 4 shows how TOPAS 4D scoring can effectively confirm 4D behavior, in this case from a rotating modulation propeller in an eye treatment beamline.

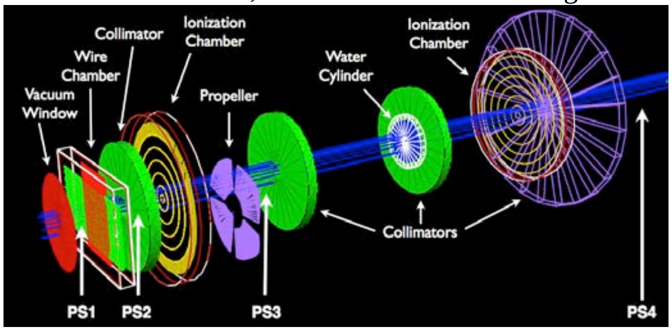
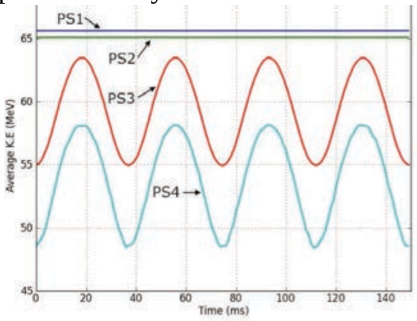
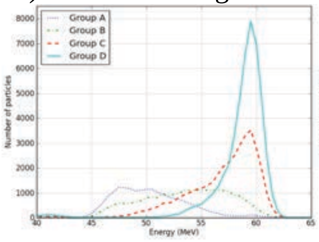


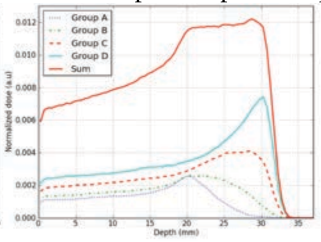
Figure 4. Propeller rotates in UCSF Eye Treatment beamline. Protons (blue lines) travel left to right.



TOPAS scores energy as function of time at four phase space output planes PS1 to PS4.

Energy delivered to a water phantom located at right (beyond PS4) for four different angular intervals of propeller rotation.





SOBP for the same four angular intervals of propeller rotation (lower four lines) and the overall SOBP (upper line).

To make visualization more trustworthy in 4D, we gave TOPAS a feature whereby scored volumes visibly darken as dose is accumulated (figure 5).

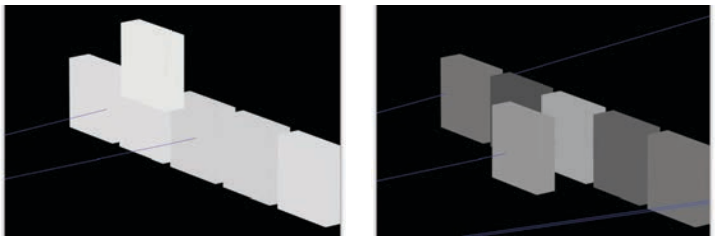


Figure 5. A beam of protons hits an array of sensitive films. Top film starts above beam path, moves downwards during irradiation. Final position, at right, shows moving film has darkened and has shielded film behind it.

6. Unit Testing versus End-to-End Testing

The more complex and flexible a simulation tool, the more formal its testing must be. Formal testing must include both Unit Testing and End-to-End testing. "Unit" refers to the smallest piece of the code, one particle source, one treatment head component, one scorer. To make the distinction, we step into metaphor. We show a single unit from the world of railroads: one brake shoe from one railroad car (figure 6). For a unit test, we use the minimum combination of parts to be able to test some fundamental behavior. Put a brake and the wheel together to test the stopping power.

Figure 6. Unit test. One wheel. One brake. Stopping power can be measured.

(Klotzbremse MaK 450 C.jpg, ©MdE Wikimedia Commons, CC-BY-SA 3.0 German)

Figure 7. Unit testing is not sufficient in complex systems, and end-to-end testing can never anticipate all situations. The train wreck at Montparnasse, 1895 (Studio Lévy & fils).

When we combine many units into a larger system, that is an end-to-end test. End to end tests can get very complicated and include a huge number of independently moving parts or even human interaction. And though we get the stopping power right in our unit test, we may still get an unexpected result from an end-to-end test. End-to-End testing of a very complex system can never be perfect (figure 7) as there are too many possible overall states for each to be tested and human factors must be considered (this topic is addressed in detail in the growing field of “system safety” [10,11]).

Unit tests should be included in simulation codes. An example of such a test from TOPAS is as follows: since TOPAS scoring allows the user to segment a cylinder into rho, phi and z sections, we need to confirm that dose computed in one of those sections uses the correct volume of that section. The calculation involves trigonometry, correct counting of the number and extent of segments and involves caching values to avoid repeated calls to trigonometric functions during the MC’s frequently-visited stepping loop. As these complications provide areas where a programming mistake could creep in, we provide a unit test - we shine a broad flat beam on that section and confirm that the scored dose equals what we expect from first principles of energy, density and volume. TOPAS has many such tests, from checking fluences to checking that the direction of our plus phi angle is as intended. The tests are distributed to our users as part of our included examples so they can replicate the results.

As End-to-End testing, we measure dose output from several entire beamlines in realistic setups. We don't rely on these tests to tell us that the units work; we rely on them to tell us that our process of putting modules together works. In an end-to-end test, two wrongs can make a right. For example, if we have a beam energy too high, but it passes through a block that is too thick, the dose beyond the block may be correct. But if you move one of those components, the errors then manifest.

7. Conclusion - QC is a Partnership

Developers of highly configurable, 4D MC simulation tools should unit test as many parts of their system as possible. They should End-to-End test some complicated setups, but must make it clear to end users that the tool developer cannot end-to-end test everything. QC is ultimately a partnership between the tool developer and the user. Key is that the developer be clear to the end user about what has been tested and what has not. As TOPAS developers, we have been fortunate that our users are medical physicists - by the nature of the profession they are the most skeptical of all scientists. As codes move to increasing flexibility, increasing 4D, medical physicists must stay skeptical, must insist on formal unit testing and must understand the severe limits of any one particular end-to-end test.

8. References

[1] Perl J, Shin J, Schümann J, Faddegon B and Paganetti H 2012 TOPAS: An innovative proton Monte Carlo platform for research and clinical applications. *Med. Phys.* **39,** 6818–6837
[2] Agostinelli S et al. 2003 Geant4—a simulation toolkit. *Nucl. Instrum. Methods A* **506,** 250–303
[3] Shin J, Perl J, Schümann J, Paganetti H and Faddegon B A 2012 A modular method to handle multiple time-dependent quantities in Monte Carlo simulations. *Phys. Med. Biol.* **57,** 3295–3308
[4] Allison J, Asai M, Barrand G, Donszelmann M, Minamimoto K, Perl J, Tanaka S, Tcherniaev E and Tinslay J 2008 The Geant4 visualisation system. *Comp. Phys. Comm.* **178**, 331-365
[5] Schümann J, Paganetti H, Shin J, Faddegon B A and Perl J 2012 Efficient voxel navigation for proton therapy dose calculation in TOPAS and Geant4. *Phys. Med. Biol.* **57,** 3281–3293
[6] Kawrakow I and Fippel M 2000 Investigation of variance reduction techniques for Monte Carlo photon dose calculation using XVMC. *Phys. Med. Biol.* **45**, 2163-2183
[7] Kawrakow I, Rogers D W O and Walters B R B 2004 Large efficiency improvements in BEAMnrc using directional splitting. *Med. Phys.* **31**, 2883– 2898
[8] Brualla L, Salvat F and Polanco-Zamora R 2009 Efficient Monte Carlo simulation of multileaf collimators using geometry-related variance-reduction techniques. *Phys. Med. Biol.* **54**, 4345–4360
[9] Ramos-Méndez J, Perl J, Faddegon B, Schümann J and Paganetti H 2013 Geometrical splitting technique to improve the computational efficiency in Monte Carlo calculations for proton therapy. *Med. Phys.* **40,** 041718
[10] Chou D 2012 Health IT and Patient Safety: Building Safer Systems for Better Care. *JAMA: The Journal of the American Medical Association, 308*(21), 2282-2282
[11] Leveson N G 2012 *Engineering a safer world: Systems thinking applied to safety*. MIT Press

Acknowledgments
This work was supported by NIH/NCI under R01 CA 140735-01.