PROTOCOL FOR *vpg_tle*

VARIANCE REDUCTION TECHNIQUE FOR PROMPT GAMMA SIMULATION

August 28, 2025

Contents

1 INTRODUCTION

The vpgTLE module (voxelized prompt gamma Track-Length Estimator) is a variance reduction technique that enables fast and accurate estimation of prompt gamma production.

In standard analogue Monte Carlo simulations, a prompt gamma is only detected if it is emitted following a nuclear reaction and reaches the detector — a rare event requiring a large number of incident particles to achieve good statistics.

vpgTLE is based on the Track-Length Estimator (TLE) approach [1], which assigns an expected prompt gamma yield to each particle step, based on its energy E_p and the material crossed, regardless of whether an actual reaction occurs.

This allows spectra to be estimated from particle tracks alone, without simulating prompt gamma emission directly. The method comprises three main stages [2, 3].

- **Stage 0:** Generate databases of prompt gamma emission spectra as functions of E_{pg} for each element Z and incident energy E_p. These are reusable but must be regenerated if the physics list changes.
- **Stage 1:** simulation of particle (hadron or ion) transport through a voxelized volume (typically derived from a CT scan). For each traversed voxel, two spectra are built: one as a function of prompt gamma energy E_{pg}, and one as a function of emission time t_{pg}. This step is illustrated in Figure 1.2, alongside stage 0.

- **Stage 2:** simulation of prompt gamma emission based on the maps calculated in the previous step. This last step for the GATE 10 vpgTLE module is still not builded.

2 Source codes

The source codes for stages 0 and 1 are available in a dépôt GitHub repository that mirrors the full OpenGATE directory structure, enriched with programs specific to the vpgTLE module. These programs are organized under the `contrib/vpgTLE` folder.

The C++ code for Stage 0 is located in the stage 0 directory : `contrib/vpgTLE/stage0`.

The scripts for Stage 1 are available in the stage X directory : `contrib/vpgTLE/stageX/stage1`.

3 Stage 0

This is done once for all with a large statistics on the CC, eg 10^{10} protons split in 1000 jobs of 107 protons each. It has to be redone each time the physics is changed (Geant4 version, production cuts, ...).

Modifications had to be made to internal Geant4 libraries due to technical constraints. The files `G4ParticleHPInelastic.cc` and `G4ParticleHPInelastic.hh`, adapted for the vpgTLE module, are available in the `modified_G4HP_file` directory. After integrating these files into the Geant4 source code and compiling, the stage0-vpgtle script can be executed with the following options:

-n: number of collisions per energy bin (default: 1×10^4);
-p: type of incident particle for the PG database (proton or neutron); "proton" is used by default;
weight: if specified, the standard vector is computed and stored in the output .root file.

After compilation on the cluster, run typically 1000 simulations of 10^7 protons (hard-coded in stage0-vpgtle.cc): "sbatch –array=1-1000 run_stage0.slurm -n {nb collisions} -p {particle of interest} {weight}". It should take about 6 hours.

Combine all 1000 outputs by executing the script `merge_root.py` in the stage0 folder. This is a specific merging code for the prompt-gamma database – to merge the variables GammaZ, Kapa Inelastique, EpEpg, and NrPG, and normalize them by the number of jobs.

Output: The output of Stage0 is therefore the files pgdb_neutron.root and pgdb_proton.root, which will be used in Stage1 python scripts, both analog (just for the prompt-gamma-energy sampling and limits) and vpg. Don't forget to copy the pgdb.root files in the data directory of stageX.

4 stage 1

The data folder contains all the necessary input files for running the simulation on a CT volume voxelized at a resolution of $(4mm)^3$. The databases generated in the stage 0 should also be stored here. The scripts for Stage 1.a (Monte Carlo simulation) and Stage 1.b (post-processing) are located in the `src` folder, along with auxiliary scripts `multijobs.py` and `variables.py`, which are used respectively for parallel execution and for managing the `tle_simu` class used in the processing. Additional options will be added in the future to improve usability. The C++ programs used for the Monte Carlo simulation

(particle processing at step level) are available in the `actor` folder for both the analog and vpgTLE cases.

- **1.a :** For cluster execution, memory usage is relevant since four 4D volumes are created. You can run the `vpg_tle` scripts (`python stage_1a.py`) locally with 10^7 protons (about 1 hour of computation), which is sufficient to obtain relevant results.
- **1.b :** No cluster is needed for this step, as it is only a post-processing of the merged (or single) output from stage 1.a. You do not need to modify the script, even if you changed stage 1.a, since parameters are automatically transferred from 1.a to 1.b. This stage processes only the energy spectrum (the ToF spectrum is a direct result of stage 1.a). Its output consists of three images — spectra for protons, neutrons, and both — which should be used as direct input for the future stage 2. This step takes about 15 minutes to compute.

References

[1] J F Williamson. "Monte Carlo evaluation of kerma at a point for photon transport problem." In: *Phys. Med. Biol. 67 184001* (1987). DOI: 10.1118/1.596069.

[2] W El Kanawati et al. "Monte Carlo simulation of prompt gamma-ray emission in proton therapy using a specific track length estimator." In: *Phys. Med. Biol. 60 8067* (2015). DOI: 10.1088/0031-9155/60/20/8067.

[3] B F B Huisman et al. "Accelerated prompt gamma estimation for clinical proton therapy simulations". In: *Phys. Med. Biol. 61 (2016) 7725–7743* (2016). DOI: 10.1088/0031-9155/61/21/7725..