Enhancing the efficiency of MC simulations of radiation transport

F. Salvat Pujol\textsuperscript{1}, M. Novak\textsuperscript{1}, S. Guatelli\textsuperscript{2}

15\textsuperscript{th} EURADOS School

\textit{Computational Methods in Dosimetry
State of the Art and Emerging Developments}
Overview

- **Convergence** of a Monte Carlo (MC) simulation

- **Figure of merit** (efficiency) of a MC simulation

- Focus on **essential physics** and **simulation parameters** *FIRST*

- Efficiency enhancement:
  - Software/algorithms side: **variance reduction/biasing** techniques
  - Hardware side: **distributed/parallel** MC runs

- **Exploratory outlook**
  - Applications of **GPUs** in MC simulations of radiation transport
  - Machine learning applications
Convergence and efficiency of a Monte Carlo simulation
Statistical uncertainty in a MC simulation

- 150-MeV p beam impinging on water
- Scoring energy deposition density
- Averaged over transverse plane
- Displayed as a function of depth
Statistical uncertainty in a MC simulation

- $N=500$ primaries
- CPU time: $T_0 \sim 1 \text{s}$
- We focus on the indicated error bar
Statistical uncertainty in a MC simulation

- $N=4N_0=2000$ primaries
- $T \sim 4 \text{ s} = 4T_0$
- Error bar has halved
- $N = 16N_0 = 8000$ primaries
- $T \sim 16 \text{ s} = 16T_0$
- Error bar has halved again

- The relative uncertainty of a MC estimator $\sigma_f/f$ scales like $\sigma_f/f \sim 1/\sqrt{N}$
- The CPU time scales like $T \sim N$
Figure of merit of a MC simulation algorithm

- Figure of merit (efficiency)
  \[ \epsilon = \left( \frac{\bar{f}}{\sigma_f} \right)^2 \frac{1}{T} \]

- Scaling with N:
  - \( \sigma_f / f \sim 1/\sqrt{N} \) and \( T \sim N \)
  - For a given MC simulation problem, \( \epsilon \) is independent of \( N \) (when \( \sim \) converged!)

- \( \epsilon \) is a relative measure of how well computational time is spent towards convergence

- For simulation problems with pathologically slow convergence / low efficiency, one wishes to have techniques to lower \( T \) and/or \( \sigma \), overall increasing \( \epsilon \)
Before “fancy/sophisticated” attempts to enhance the efficiency of MC simulations, one better have a reasonable grasp of

- Underlying physics
- Monte Carlo simulation parameters
Example: Setting particle transport thresholds

- **Energy deposition by 150-MeV protons in water**
  - Dominated by proton ionization losses (collisions with target e-)
  - Mean free path for nuclear inelastic scattering of 150-MeV p in water: **106.8 cm**
    (a few protons undergo a nuclear reaction -> n production -> contribute mostly to tails of the distribution, modulate a bit the intensity of the Bragg peak)
  - Simulation bottleneck: e- production/transport threshold i.e. condensed (dE/dx) vs detailed delta-ray simulation
Threshold settings

- **Exponential increase** of CPU time as one lowers e- thresholds
- An e- threshold of 100 keV is OK if one cares just about a coarse depth-dose curve:
  - CSDA range of 100 keV e- in water: \(~0.014\) cm
  - Histogram spatial resolution: \(~0.16\) cm -> we could have used even higher e- thresholds!
- **Factor 1000 speed-up** just for being minimally aware of what governs the problem

<table>
<thead>
<tr>
<th>e- transport threshold (keV)</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>3</td>
</tr>
<tr>
<td>50</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>72</td>
</tr>
<tr>
<td>5</td>
<td>176</td>
</tr>
<tr>
<td>1</td>
<td>3000</td>
</tr>
</tbody>
</table>
Particle transport/production thresholds

- MC codes typically provide **default threshold values**, but they are **not guaranteed to be meaningful for your problem**.

- Following e-/e+ to energies lower than one really needs is a ruthless time-intensive **CPU eater**.

- It pays off to set transport threshold such that residual range is small compared to geometry / scoring mesh dimensions (and such that you don’t cut out any relevant physics process...).
Enhancing the MC simulation efficiency in problems with strong attenuation - Region importance biasing
Shielding example

- 500-MeV p beam
- 20 cm W target in air
- Concrete shielding, 3 layers of 25 cm width
- Estimate H*(10) ambient dose equivalent outside shielding
The basic physics

- Proton undergoing nuclear inelastic interactions, mostly in W

- Secondaries produced per incident proton (tallied with FLUKA):
  - 10.8 n -> undergo inelastic interactions mostly in target and concrete
  - 7.4 photons
  - 1.6 p
  - <0.5: d, t, 3He, 4He

- n and photons might manage to make it through the shielding and contribute to the H*(10) ambient dose outside, by and large stopped in concrete
Neutron and gamma fluence

- Particle fluence past shielding is **dominated by neutrons and photons**
- Neutron and photon fluence is gradually **attenuated** by the shielding
- But we still want a statistically significant estimate of the dose outside of the shielding
H*(10) ambient dose equivalent

- $N_{\text{prim}} = 4000$
- $T_{\text{CPU}} = 43$ seconds

NOTE: only meaningful in air/outside shielding...

$\epsilon \sim (0.8^2 \times 43)^{-1} \sim 0.03 \text{ s}^{-1}$
$H^*(10)$ ambient dose equivalent, 4x more primaries

- $N_{\text{prim}} = 16000$
- $T_{\text{CPU}} = 171$ seconds

$\epsilon \sim (0.4^2 \times 171)^{-1} \sim 0.03\, s^{-1}$
Biasing

- Figure of merit of a Monte Carlo simulation:

\[ \epsilon = \left( \frac{\bar{f}}{\sigma_f} \right)^2 \frac{1}{T} \]

- Convergence of desired physical observable might be slow, e.g.:
  - Problems with strong attenuation of relevant particle fluence
  - Processes with low cross section (e.g. photonuclear interactions)

- Biasing techniques aim at enhancing the simulation efficiency:
  - Reduce the variance and/or CPU time
  - Leading to an overall larger \( \epsilon \)
Region importance biasing

- Assign numerical **importance** to regions in your geometry

  - **Splitting**
    - Crossing into region with larger importance
    - Particle split into $I_2/I_1$ particles
    - Reduced statistical weight

  ![Diagram](image)

  \[ I_1 = 1 \quad I_2 = 3 \]

  \[ w' = w/3 \]

  \[ w' = w/3 \]

  \[ w' = w/3 \]

  - **Russian roulette**
    - Crossing into region with lower importance
    - Particle reduced to $I_2/I_1$ particles
    - Enhanced statistical weight

  ![Diagram](image)

  \[ I_1 = 3 \quad I_2 = 1 \]

  \[ w' = \frac{I_1}{I_2}w = 3w \]
Region **importance** biasing for our shielding problem

(ideally one wishes to avoid importance discrepancies in contiguous regions…)

Regions we do not care so much about

Regions we care a lot about
\( H^*(10) \) ambient dose equivalent, original \( N_{\text{prim}} \), biased

- \( N_{\text{prim}} = 4000 \)
- \( T_{\text{CPU}} = 42 \) seconds

\[ \epsilon \sim (0.2^2 \times 42)^{-1} \sim 0.6 \text{ s}^{-1} \]

(efficiency increased by a factor ~20!)

- Particle population is maintained (suppressed) in regions of high (low) importance
- Efficiency **enhancement in the right-hand regions** comes at the **detriment of left-hand regions**
- 20% uncertainty is still a bit far from convergence -> from now on it’s a matter of running for more primaries
A word of caution

- Biasing techniques effectively concentrate simulation effort in desired regions of the geometry / phase space

- It’s the user’s responsibility to ensure no contributions from relevant regions are left out by a too careless biasing scheme

- Particle shower correlations are lost*: no event-by-event analyses
Standard biasing techniques

- Region importance biasing
- Mean free path biasing
- Weight windows
- Ant colony algorithm
- ...

Hardware acceleration
MC as a naturally distributed calculation
Efficiency enhancement from distributed runs

- MC simulation efficiency:
  \[ \epsilon = \left( \frac{\bar{f}}{\sigma_f} \right)^2 \frac{1}{T} \]
  
  Simulation time
  Relative statistical uncertainty (squared)

- For a fixed number of primaries \( N \) distributed in \( n \) jobs running at the same time, the cumulative CPU time \( T \) is the same, but if one takes \( T \) as a *walltime*, the simulation efficiency is enhanced by a factor of nearly \(* n\)

- Negligible coding overhead, no synchronization issues
[Possible bottleneck for large memory requirements]

- n distributed runs $\rightarrow$ n x memory

- **Each instance replicates its own memory** for geometry, cross section, scoring, etc.

- Extreme limit (complicated geometry + e.g. plenty of low-energy neutron cross sections to load + very dense scoring meshes), insufficient memory e.g. if running 16 threads on one CPU

- Codes like e.g. Geant4 allow for **shared memory** (cross sections and geometry) among threads

- A bit of coding overhead / thread synchronization

Ref: https://indico.cern.ch/event/776050/contributions/3240673/attachments/1788898/2913542/Multithreading1.pdf
Best of both worlds: exploit both biasing and distributed/parallel runs!

Twice as many jobs now, leading to:

- $N_{\text{prim}} = 8000$
- $T_{\text{wall}} = 42$ seconds

$\epsilon \sim (0.14^2 \times 42)^{-1} \sim 1.2 \text{ s}^{-1}$

(efficiency increased by a factor $\sim 40$ wrt to the initial efficiency)

For a vast majority of practical situations, a combination of biasing + distributed runs suffices
Exploratory outlook (hardware): GPUs
GPUs

• **GPU:** graphics processing unit
  • Parallel processing of thousands of computational threads

• **Naturally advantageous scenarios:**
  • Tasks requiring millions of *identical* operations (problem reducing to linear algebra)
  • *Direct, uniform, contiguous memory access*

• **Challenging scenarios:**
  • Tasks with *thread divergence* and *random memory access*  
    (…as in a MC simulation of radiation transport!)

• Requires heavy recoding of MC simulation  
  (CUDA programming model)
MPEXS

- KEK-based tool for radiotherapy:
  - Limited set of physics: e-, e+, gamma
  - Simple geometry (infinite medium)
  - Water-equivalent material

- Process thousands of independent particle histories in parallel

- Thread divergence: ~50% (!!)
- Nevertheless, speed-up factor of ~400 attainable against single-core CPU.

- Ref: https://indico.cern.ch/event/921244/contributions/3870624/attachments/2045775/3427426/HSF-200527-MPEXS.pdf
Electromagnetic interactions + geometry are among the most CPU time consuming aspects for HEP detector simulations

Ongoing R&D attempting to cast HEP particle transport problem to benefit from massive parallelization on GPU architectures

**AdePT:**
- Workload balancing, reduce impact of shower tails, maximize number of tracks in flight, etc
- Speed-up observed in simple geometries, pending real geometry (ATLAS/CMS calorimeters)

**Celeritas:**
- Targetting EM+hadronic physics, re-implementation of subset of G4 physics for GPU, focusing on EM showers

Refs (talks and git repos):
- [https://indico.cern.ch/event/1156147/contributions/4854699/attachments/2444243/4188160/HSFGPU_report.pdf](https://indico.cern.ch/event/1156147/contributions/4854699/attachments/2444243/4188160/HSFGPU_report.pdf)
- [https://github.com/apt-sim/AdePT](https://github.com/apt-sim/AdePT)
- [https://github.com/celeritas-project/celeritas](https://github.com/celeritas-project/celeritas)
Exploratory outlook (algorithms): Machine learning attempts

Material kindly provided by Florian Mentzel

Do not miss Habib Zaidi’s interesting talk at 16h!
MC+ML attempts for medical physics applications

- Main ongoing lines of applications of ML to MC simulations:
  - Convolutional neural networks for dose estimation in radiotherapy and imaging
  - Dose denoising from low statistics Monte Carlo simulations,
  - Detector modelling
  - Event selection
  - Replacing particle sources / phase space modelling with generative models

<table>
<thead>
<tr>
<th>Application</th>
<th>Input type</th>
<th>Refs (among others)</th>
<th>Main ML types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dose denoising</td>
<td>image</td>
<td>[43, 59, 71, 101, 103, 111, 131, 159]</td>
<td>CNN, U-net</td>
</tr>
<tr>
<td>SPECT scan-time reduction</td>
<td>image</td>
<td>[62, 119, 121]</td>
<td>CNN, U-net</td>
</tr>
<tr>
<td>CBCT scatter modelling</td>
<td>image</td>
<td>[27, 58, 60, 75, 79, 84, 87, 88, 140, 145, 152, 155]</td>
<td>CNN, U-net</td>
</tr>
<tr>
<td>PET attenuation/scatter correction</td>
<td>image</td>
<td>[6, 97]</td>
<td>CNN, U-net</td>
</tr>
<tr>
<td>Detector response modelling</td>
<td>particles</td>
<td>[126, 144]</td>
<td>GAN, MLP</td>
</tr>
<tr>
<td>Source + phase space modelling</td>
<td>particles</td>
<td>[108, 125, 127]</td>
<td>GAN</td>
</tr>
<tr>
<td>Event selection</td>
<td>particles</td>
<td>[8, 12, 40, 46, 93, 98, 100, 102, 107, 157]</td>
<td>MLP, CNN</td>
</tr>
<tr>
<td>Interaction position in scintillators</td>
<td>various</td>
<td>[23, 33, 37, 99, 109, 112, 122, 150, 154]</td>
<td>MLP, CNN</td>
</tr>
</tbody>
</table>

1http://hdl.handle.net/11803/19255
2http://hdl.handle.net/2078.1/thesis:14550

Overview of ML applications in MC simulations (~medical)

- **Dose estimation with neural networks:**
  - Oscar Pastor-Serrano et al., *Millissecond speed* deep learning based proton dose calculation with Monte Carlo accuracy. *Physics in Medicine and Biology*, in press. [https://doi.org/10.1088/1361-6560/ac692e](https://doi.org/10.1088/1361-6560/ac692e)

- **Low-statistics Monte Carlo enhancement**

- **Replacing particle sources with generative models**
  - D. Sarrut et al., Generative adversarial networks (GAN) for compact beam source modelling in Monte Carlo simulations. *Physics in Medicine and Biology* 64 215004, 2019. [https://doi.org/10.1088/1361-6560/ab3fc1](https://doi.org/10.1088/1361-6560/ab3fc1)
A sobering comment

- D. Sarrut et al., *Front. Phys.* **9** 738112 (2021)

“For the moment, even if it is envisioned that deep learning can improve simulations, it does not seem certain that it can always replace Monte Carlo.”
Summary
Summary

- Basic **understanding** of **underlying physics** and code **simulation parameters** can already lead to orders of magnitude enhancement of simulation efficiency wrt a careless run.

- **Biasing techniques** as natural methods to enhance simulation efficiency e.g. in desired regions of interest in geometry:
  - Further orders-of-magnitude enhancement, but user responsible for not cutting out relevant corners of phase space.

- **MC naturally distributed** computational problem:
  - Truly parallel codes can reduce memory requirements.

- Exploratory outlook onto applications of **GPUs and ML** to MC:
  - Even beyond: field programmable gate arrays (FPGAs), MC on a chip (MCoaC):
    - Speedups of factor ~90 for TOPAS [https://doi.org/10.1016%2Fj.ejmp.2019.06.016](https://doi.org/10.1016%2Fj.ejmp.2019.06.016)
    - Less power (~30 W) than CPUs (~100 W) or GPUs (~300 W)

- **MC code developers share the blame**:
  - Efficiency of interaction/transport/sampling algorithms is on us! Physics performances **1st**, optimization **2nd**.
Thank you very much for your attention!