FERMILAB-SLIDES-25-0012-CSAID

Fermilab U.S. DEPARTMENT OF ENERGY Office of Science

Event Biasing and Fast Simulation

Soon Yung Jun (Fermilab)

12th International Geant4 Tutorial in Korea,

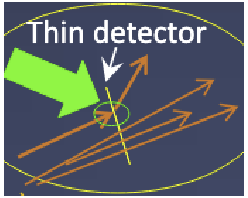
Feb 3-7, 2025@Pohang Accelerator Laboratory, Pohang

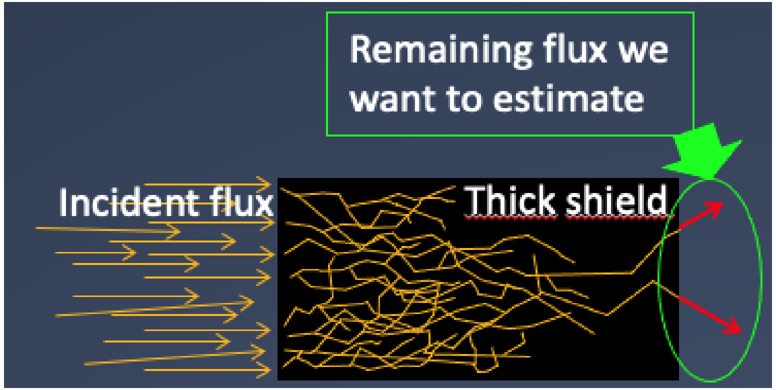
Contents

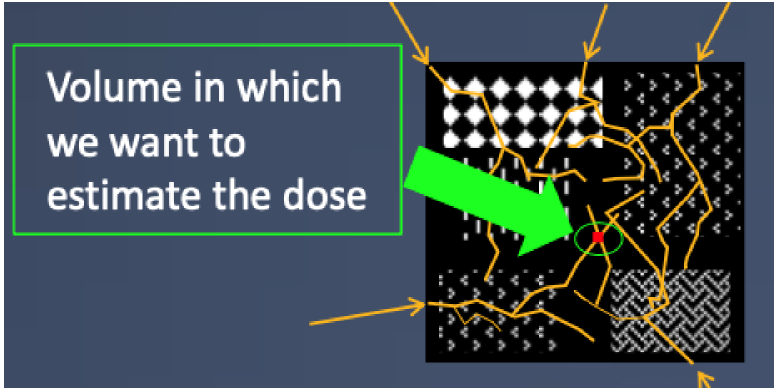
- Event Biasing
  - Overview of Event Biasing
  - Biasing Options in Geant4
  - Examples
- Fast Simulation
  - Interfaces
  - Examples

Rare Events

- There are simulation problems in which what we are interested in is "rare" because of the physics or the setup (or both)
- Examples of such problems
  - Estimating the efficiency of a shield or obtaining responses of a very thin detector
  - Estimating the dose in a very small part of a big setup (e.g., an electronic chip inside a satellite)



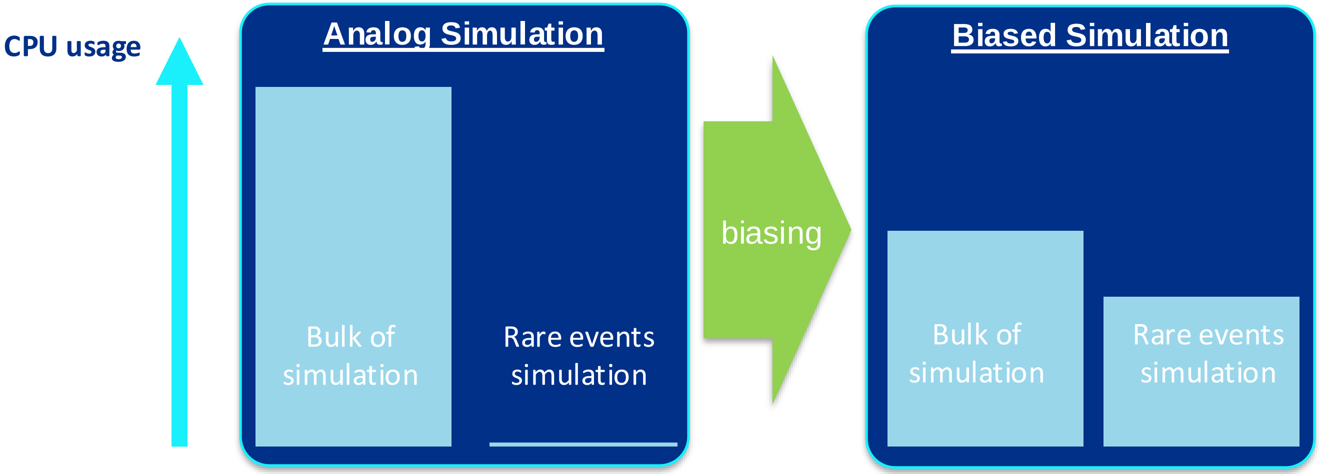




- An analog simulation is inefficient in addressing these problems → Event biasing technique

What is “Event biasing” ?

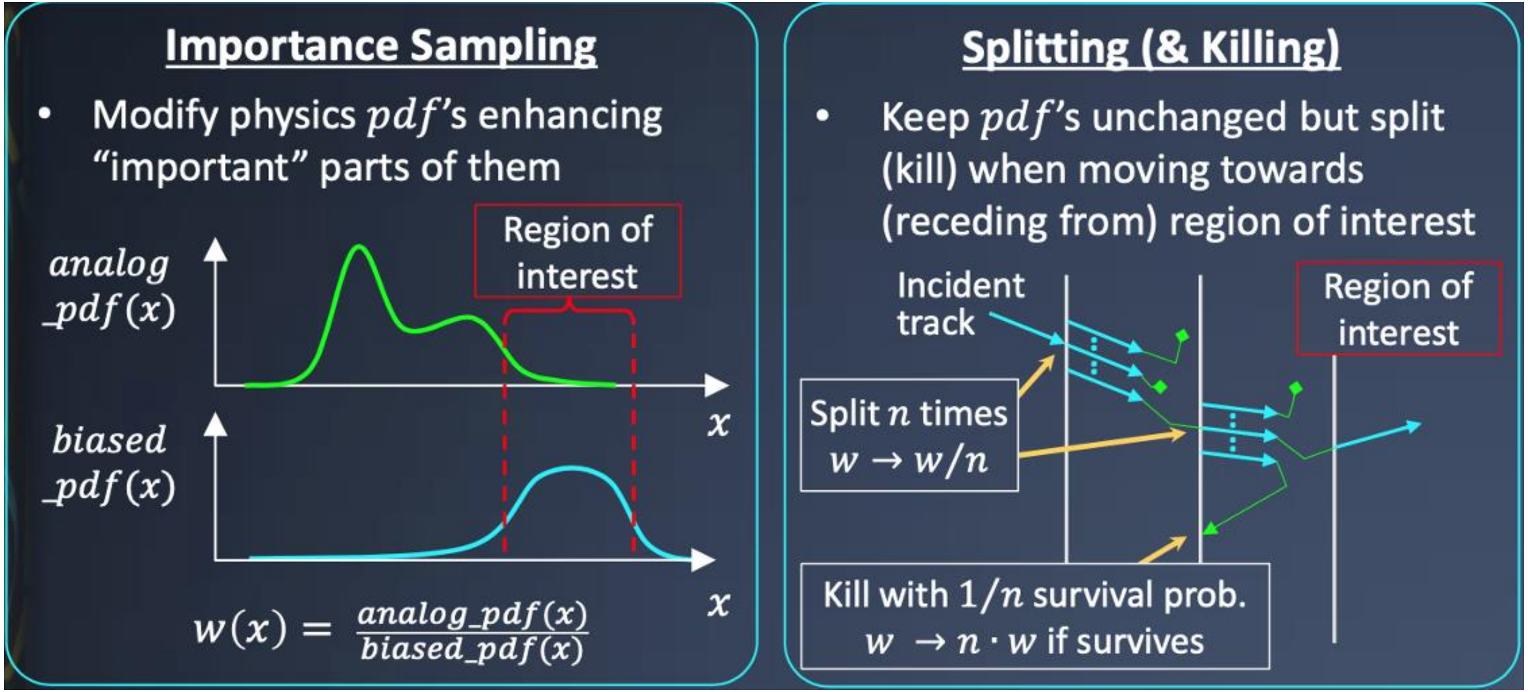
- “Event biasing” (or “biasing” or “variance reduction”) is a set of techniques to simulate rare events efficiently
  - It focuses/transfers the CPU power usage on what we want to tally



  - It can provide LARGE CPU improvements (several orders of magnitude depending on the problem!)
- “Biasing” stands for “biased simulation”: “Biased” because rare events are enhanced compared to the analog simulation.

Main Principle

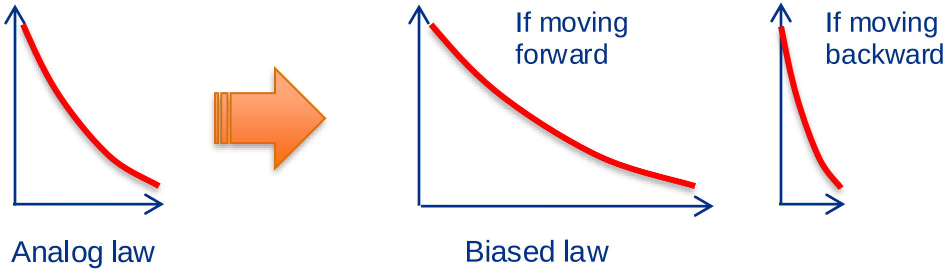
- There is a large variety of techniques, but there are essentially two underneath classes



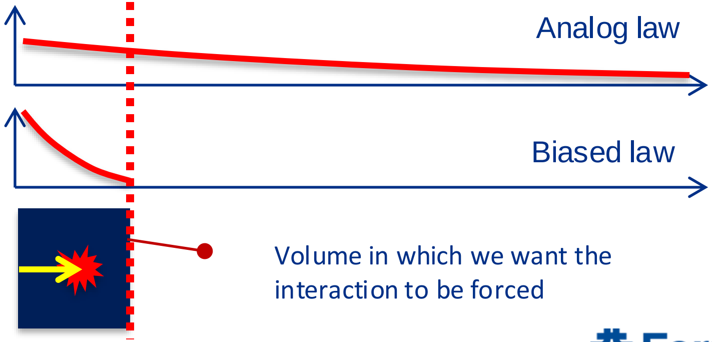
- Enhancement of rare events is tracked with statistical weights (w_i) which are used to de-bias and go back to analog quantities: e.g., $E_{deposit} = \Sigma_{i=track}\, w_i\, E^{i}_{deposit}$

Example of Importance Sampling Techniques

- Cross section biasing: the “exponential transform” technique
  - The exponential law is replaced by another law to enhance (or suppress) cross section (often with direction dependent). Available for hadron inelastic, electron and positron nuclear processes.

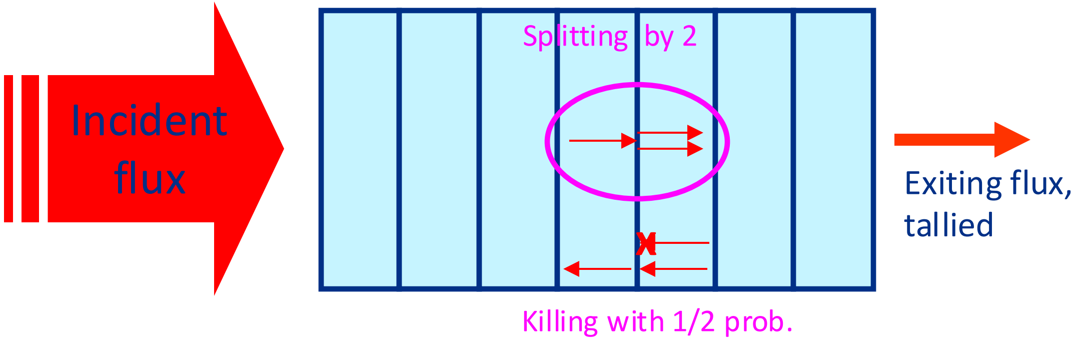


- Forcing the interaction in a thin volume:
  - The usual/analog exponential law is replaced by a truncated exponential law that does not extend beyond the volume limit



Example of Splitting

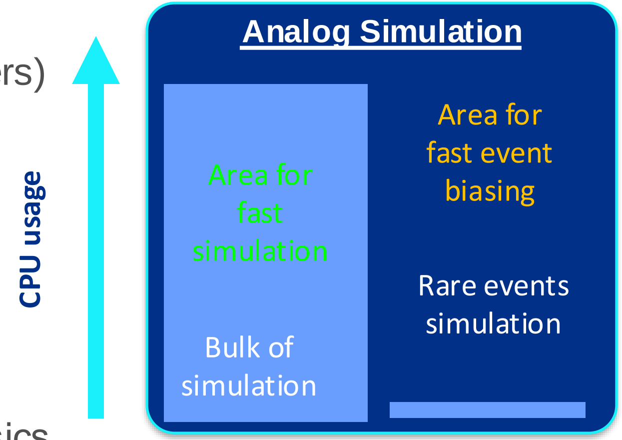
- A classical application of splitting is the “shield” simulation problem



  – Particles moving toward exit are cloned biasing the physical absorption in the shield material
  – When cloning happens, clones receive a weight of the initial track
- Splitting comes together with killing/Russian Roulette
  – For particles going opposite direction than the one we want
  – In above example, particles moving backward are killed with a probability ½ and, if it survives, its weight is multiplied by 2.
  – “Too numerous” particles are killed (population control). It is used for example in CMS, for slow neutrons

Biasing vs. Other Acceleration Techniques?

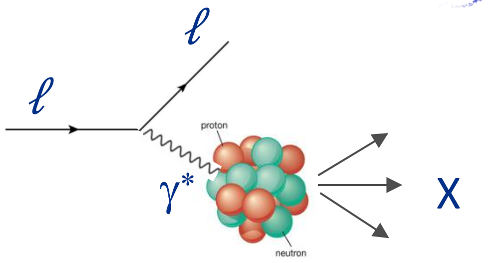
- Other acceleration techniques exist:
  - Fast simulation: use approximate but faster model (e.g.: EM showers)
    - Accelerates the bulk of the simulation
  - Deterministic computation (medical, space) (not in Geant4)
- Strength of biasing:
  - The physics at play in biasing is same with the detailed simulation
    - Biasing is “simply” another way to process the full detailed physics
  - So, results obtained with biasing are statistically the same than the ones obtained with a large/huge processing of standard simulation
- Difficulties with biasing:
  - Only “rare events” problems can be treated this way (by nature)
  - Ensuring a proper convergence can be difficult - issue of random appearance of large weights



Biasing Options in Geant4

Options In Hadronic Process (1)

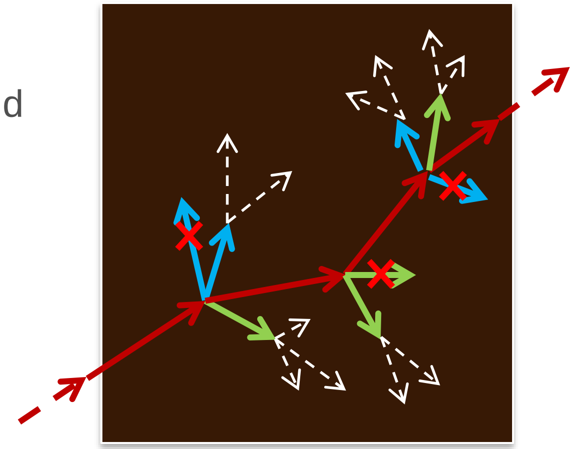
- Cross-section Biasing: BiasCrossSectionByFactor
  - Possibility to enhance a (low) cross-section
    - This an "importance sampling" approach
  - Available for photon inelastic, electron & positron nuclear processes




```
// Initialize a reaction model
G4ElectroNuclearReaction* electroReaction = new G4ElectroNuclearReaction();

// Create an associated process and register the model
G4ElectronNuclearProcess electronNuclearProcess;
electronNuclearProcess.RegisterMe(electroReaction);

// Apply biasing
electronNuclearProcess.BiasCrossSectionByFactor(100);

// Register the process
procManager->AddDiscreteProcess(&electronNuclearProcess);
```

Options In Hadronic (2)

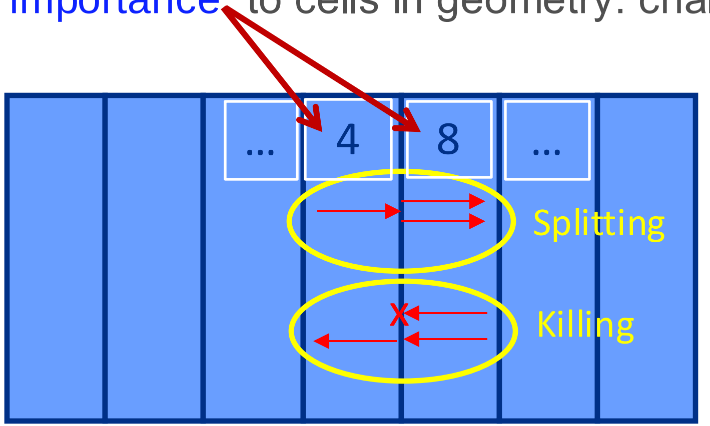
- Leading particle biasing:
  - A technique used for estimating the penetration of particles in a shield
  - Instead of simulating the full shower, at each interaction, only keep:
    - The most energetic particle
    - Randomly one particle of each species
  - This is a killing-based biasing



- Radioactive decay (by α, β emission or electron capture, γ-emission by atomic relaxation)
  - Enhance a particular decay within a given time window (half-life time)
  - Sample all decay modes with equal probabilities (branching ratio)
  - Split parent nuclide in a user defined number of copies, letting the decay go
  - Enhance emission in a particular direction

Geometry based importance biasing

- Attach "importance" to cells in geometry: change of probability density functions of interaction laws



  - Applies splitting if the track moves forward
    - with splitting factor 8/4 = 2, if track goes from « 4 » to « 8 » (e.g., here)
    - each copy having a weight = ½ of the incoming track
  - Applies killing if the track moves the other way
    - it is killed with a probability ½
    - If particle survives, its weight is multiplied by 2 (e.g., here)

Geometry-based Importance Biasing

- Geometry cells, meant to carry importance values, are created and associated to physical volumes in the detector construction:

```
// Create an "importance store"
G4IStore *istore = G4IStore::GetInstance();

// Assign an importance value to (all) physical volumes
istore->AddImportanceGeometryCell(importanceValue, physicalVolume);
```

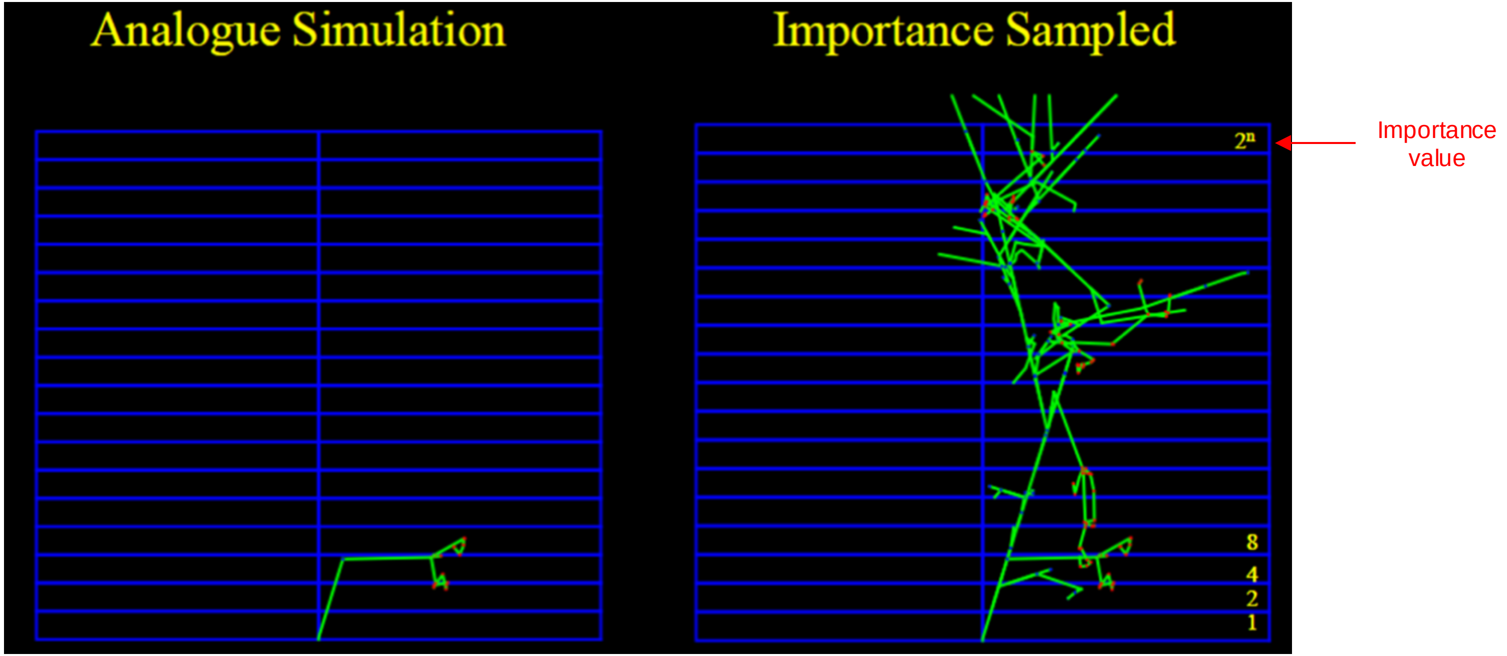
- The actual splitting and killing are handled by dedicated processes and biasing may be applied to some particle type only: e.g., neutron

```
// Apply to neutrons
G4GeometrySampler geometrySampler(worldVolume, "neutron");

// Register to the modular physics list
G4VModularPhysicsList* physicsList = new FTFP_BERT;
physicsList->RegisterPhysics(new G4ImportanceBiasing(&geometrySampler));
```

- More details can be found in examples/extended/biasing: B01, B02, B03

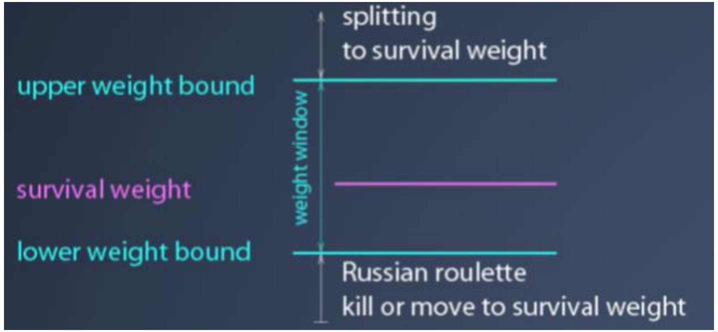
Example B01

- Geometry-based importance biasing for 10 MeV neutrons in a thick concrete cylinder: see examples/extended/biasing/B01
  - Assign importance values with a power of 2 (2^n)



Weight Window Technique

- This is a splitting & killing technique, to avoid having
  - Too high weight tracks at some point
    - As this makes the convergence of the estimated quantities difficult
    - Tracks with weight above some value are splitted
  - Too low weight tracks
    - As these are essentially a waste of time
    - Tracks below some value are “Russian roulette” - killed



- As with importance, this is configured per cell
  - And can be configured per energy
- See: examples/extended/biasing/B01

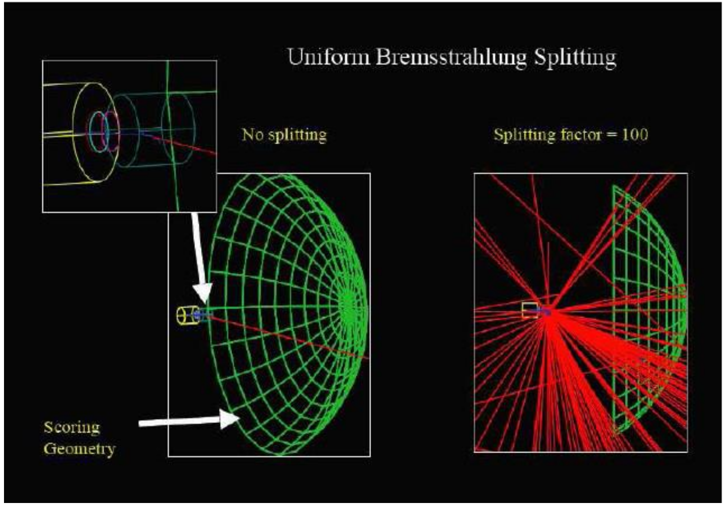
User Defined Biasing

- *G4WrapperProcess* can be used to implement user defined event biasing
- G4WrapperProcess, which is a process itself, wraps an existing process
- All function calls forwarded to wrapped process
- Needed steps:
  - Create derived class inheriting from G4WrapperProcess
  - Override only the methods to be modified, e.g., PostStepDoIt()
  - Register this class in place of the original
  - Finally, register the original (wrapped) process with user class

Example of Bremsstrahlung Splitting

- Splitting is a technique used for example in medical applications:
  - Radio-therapy with photon beam generated by bremsstrahlung
  - Interest is in photons, not in the electron → enhance photon production
  - The bremsstrahlung process is repeated N times, All bremsstrahlung photons are given a weight 1/N
  - The electron continues with one of the state among N ones generated

```
G4VParticleChange* MyWrappedProc::PostStepDoIt(const G4Track& track, const G4Step& step)
{
  G4double weight = track.GetWeight()/fNSplit;

  // Secondary store
  std::vector<G4Track*> secondaries(fNSplit);

  // Loop over the wrapped PSDI method to generate multiple secondaries
  for (G4int i = 0 ; i < fNSplit ; i++)
  {
    particleChange = pRegProcess->PostStepDoIt(track, step);
    assert (0 != particleChange);

    // Save the secondaries generated on this cycle
    for (G4int j=0; j<particleChange->GetNumberOfSecondaries(); j++)
    {
       secondaries.push_back (new G4Track(*(particleChange->GetSecondary(j))));
    }
  }

  return particleChange;
}
```



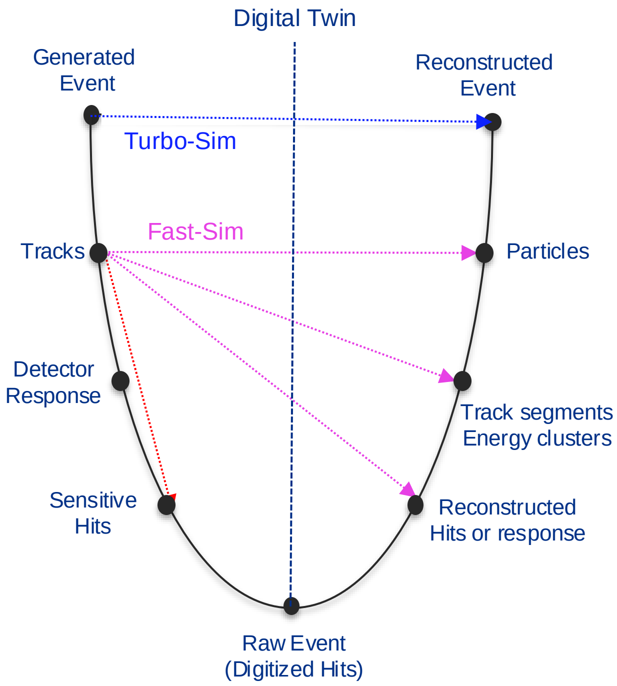
Existing Functionalities and Examples

- geant4/examples/extended/biasing
  - GB01: Individual process cross-section biasing for neutral particles
  - GB02: Force collision à la MCNP for neutral particles (as MCNP)
  - GB03: Geometry importance + further options
    - Scheme augmented compared to classical geometry importance
  - GB04: Re-implementation of a classical Bremsstrahlung splitting
  - GB05: Illustrates a "splitting by cross-section“
  - GB06: Parallel geometries with generic biasing
  - GB07: Implement a “particle leading” biasing scheme
- See other topics on “Event Biasing” by M. Verderi at https://indico.cern.ch/event/1172490/
  - Reverse Monte Carlo
  - Generic biasing scheme

Fast Simulation

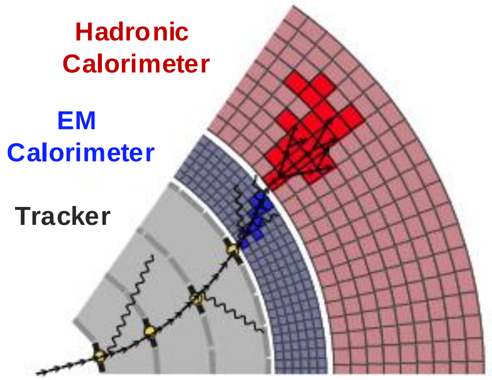
Fast Simulation

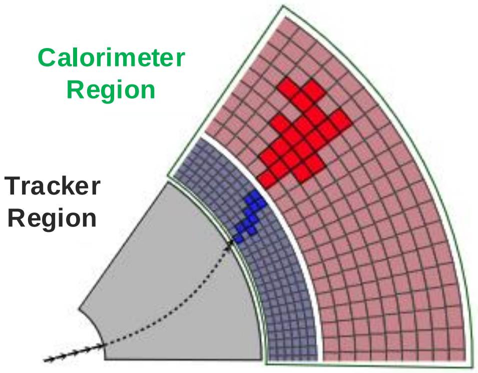
- Geant4 supports a fast simulation interface for a parameterized simulation
  - Quality (approximation) vs. Computing performance (faster than full simulation)
  - Replace Geant4 processes with external or user simulation codes
  - Allow a Geant4 process to be modified during simulation execution - very useful in certain cases
- User cases interfacing with
  - Parameterized physics (charge deposition in tracking layers, energy deposition from particle showers)
  - Fast simulation for optical photon propagation through optical fibers
  - External codes, crystal channeling as an example
  - Machine learning inference
  - Offloading a special task to other devices



How does the interface work?

- *G4FastSimulationManager*
  - Define G4Envelope (G4Region)
  - Add a model inherited from *G4VFastSimulationModel*
- *G4VFastSimulationModel* (pure virtual)
  - IsApplicable (particle definition)
  - ModelTrigger (dynamic conditions)
  - DoIt (model actions)
  - Flush (post clearance)
- *G4FastSimHitMaker*
  - make (sensitive detector)





Create Your Own Fast Simulation Model

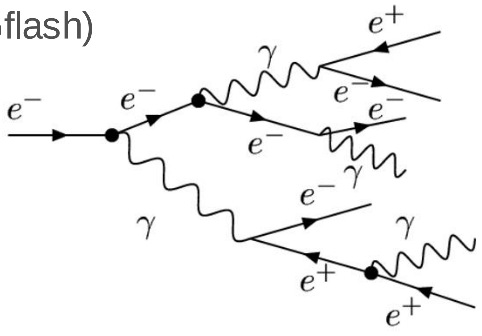
- A concrete example for parameterization, offloading, inference, etc.

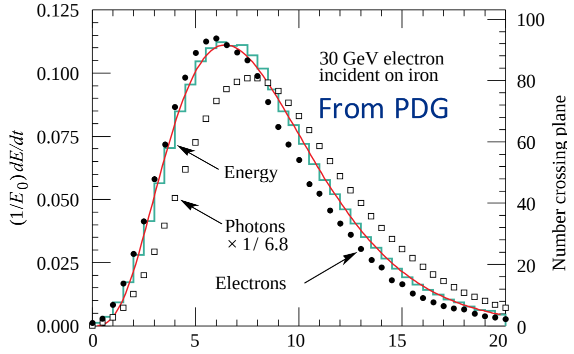
```
MyFastSimModel::MyFastSimModel(const G4String &name, G4Region *envelop)
    : G4VFastSimulationModel(name, envelope)
{
    // Action of my model (parameterization, offloading, inference, etc)
    fAction = std::make_shared<MyFastSimAction>();
}

G4bool MyFastSimModel::IsApplicable(const G4ParticleDefinition &particle)
{
    // List of applicable particles
    return (&particle == G4Electron::ElectronDefinition());
}

G4bool MyFastSimModel::ModelTrigger(const G4FastTrack &track)
{
    // Mininum energy cutoff to parameterize
    return (track.GetPrimaryTrack()->GetKineticEnergy() < GeV) ? false : true;
}

void MyFastSimModel::DoIt(const G4FastTrack &track, G4FastStep &step)
{
    // Kill the parameterised particle
    step.KillPrimaryTrack();

    // Execute the action of MyFastSimAction class
    fAction->execute(track);

    // Make sensitive hits with the list of hits from the action
    this->makeHits(track);
}
```

- Inherited from G4VFDastSimulationModel
- Appliable particles
- Conditions to trigger this model
- Actions of this model
- Create sensitive hits

EM Cascade and EM Shower (Gflash)

- Geant4 supports a fast simulation interface for EM shower simulation
  - e–γ cascade EM Parameterization (Gflash)
  - *G4VFastSimulationModel*

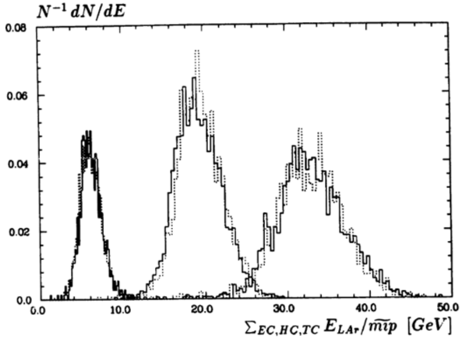




- Longitudinal energy profile
  - Gamma distribution: peak at $(\alpha\text{-}1)/\beta$

$$f(x;\alpha,\beta) = \frac{x^{\alpha-1} e^{-\beta x} \beta^{\alpha}}{\Gamma(\alpha)}$$

  - α and β are correlated with fluctuations



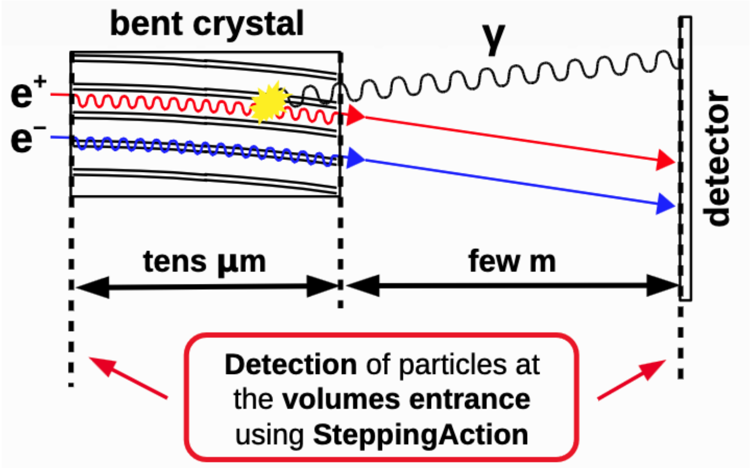
- Lateral profile
  - Core (R_C) and tail (R_T) components of the radial distribution

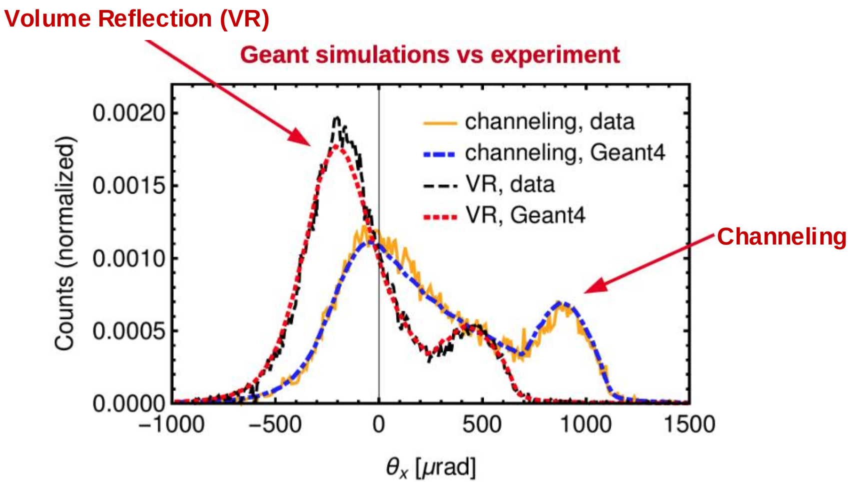
$$f(r) = p\frac{2rR_C^2}{(r^2+R_C^2)^2} + (1-p)\frac{2rR_T^2}{(r^2+R_T^2)^2}$$

- Energy deposition with a spot energy: $dE_{dep}(\vec{r}) = \frac{E_{dep}}{2\pi} f(x)\,dx\,f(r)\,dr$ $\quad \frac{\sigma_a}{E_{dep}} = \frac{a}{\sqrt{E_{inc}}}$, $E_{spot} = a^2$

FastSim Interface to External codes: Channeling in Cry stal

- Can use fast simulation to speed up transport of $e^{\pm}$ on ultra-short crystal at Mainz Mikrotron
  - 855MeV electron experiments at MAMI, A. Mazzolari et al. PRL. 112, 135503 (2014), A. Sytov et al. Eur. Phys. J. C 77, 901 (2017), JKPS 83, 132–139 (2023), see Alexei talk's at https://hep0.kisti.re.kr/event/6/
  - G4ChannelingFastSimModel (source/parameterisations/channeling)



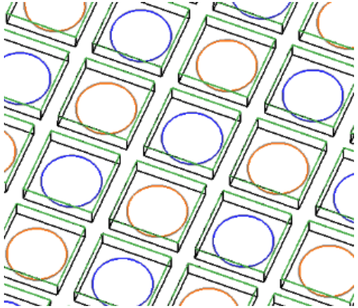


- Multithreaded version of this has run on NURION@KISTI supercomputer

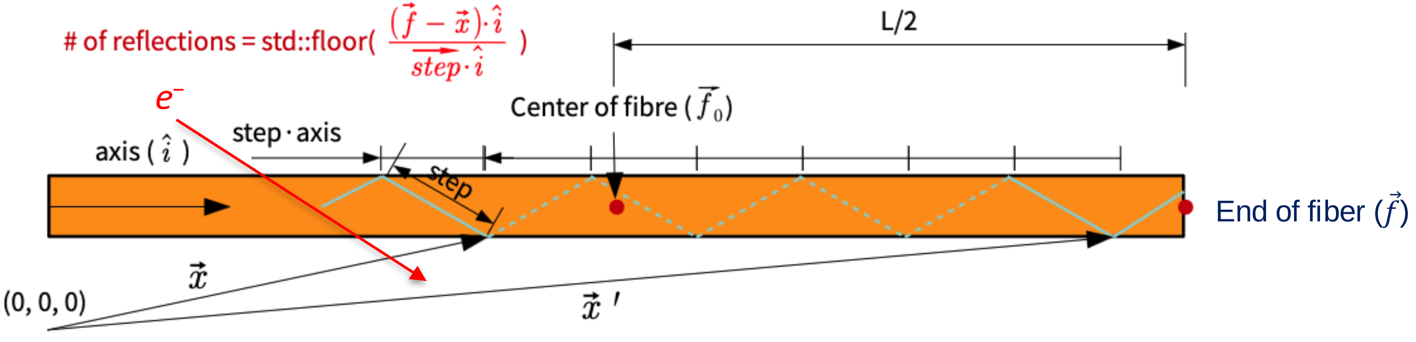
Fast Optical Photon Transportation in Dual Readout Calorimeter

- Not all intermediate transportation steep are needed for the simulation (Sanghyun Ko, SNU)

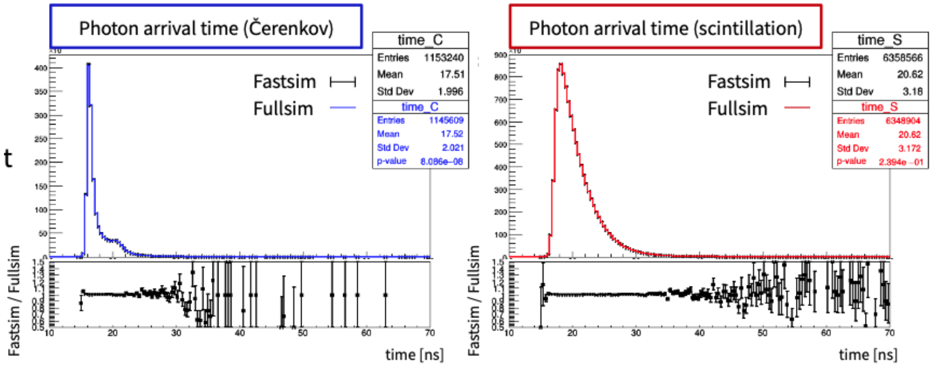
  https://indico.cern.ch/event/915715/



Dual-Readout Optical Fibers
Cherenkov and Scintillation

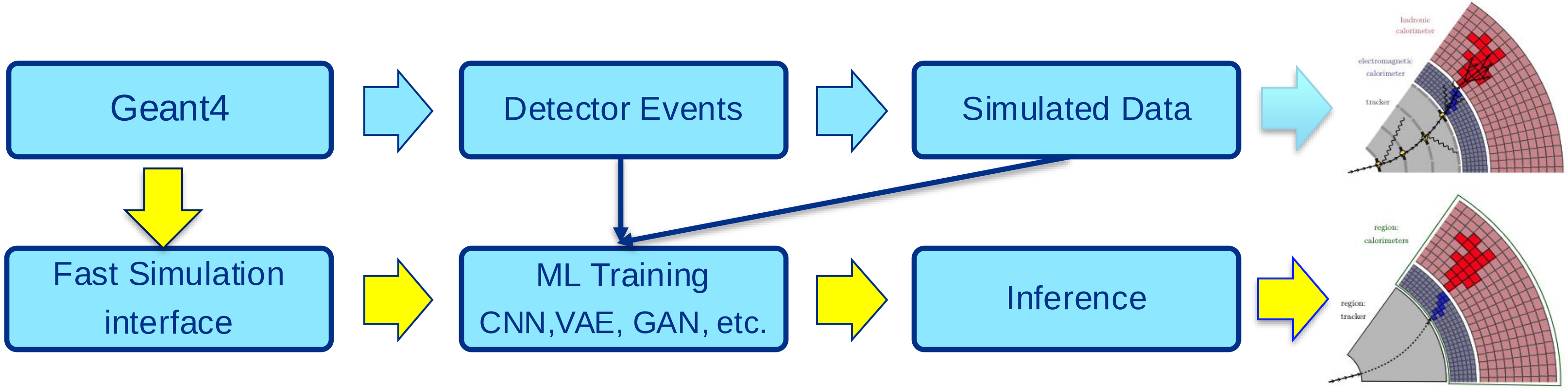


- CPU time: 1000 of 20 GeV e^- events
  - Full simulation: 304 ± 88 min
  - Fast simulation: 4.62 ± 1.17 min
  - Gain: ~70 times

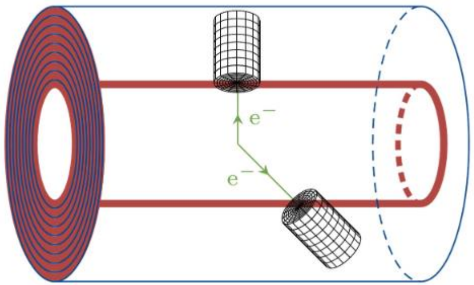


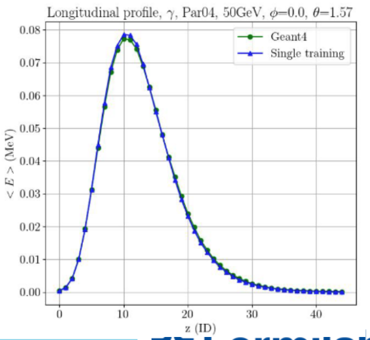
Fast Simulation Model for Machine Learning

- The Geant4 fast simulation model provides a self-contained ML framework for training and inference: each experiment requires a specific ML designing model and target



- Interface to upload neural network parameters for inference
  - Lightweight Trained Neural Network
  - Open Neural Network Exchange Libraries
  - Torch
- Calo Challenge https://arxiv.org/abs/2410.21611





Geant4 Examples

- examples/extended/parameterisations/
- Par01
  - Par01EMShowerModel (crude e^-,e^+,γ shower parameterization)
  - Par01PionShowerModel (crude π^+, π^- shower model in ghost volume)
  - Par01PiModel (shows how a parameterization can create secondaries)
- Par02
  - Par02FastSimModelEMCal.cc (e^-, e^+, γ in EM calorimeter using energy smearing)
  - Par02FastSimModelHCal (hadrons in hadronic calorimeter using energy smearing)
  - Par02FastSimModelTracker
- Par03EMShowerModel (creates and store multiple energy deposits)
- Par04MLFastSimModel (machine-learning aided fast simulation of electromagnetic showers)
- /gflash GFlashShowerModel (use of the GFLASH EM parameterization library)

Summary

- Event biasing techniques can provide very large acceleration factors in problems in which we must tally rare events. Geant4 supports a variety of biasing options
  - Leading particle, cross-section (had), radioactive decay, splitting with importance in geometry, weight window, user biasing with G4WrapperProcess, and bremsstrahlung splitting
  - they delicate to handle, but sometimes unmissable when dealing with rare event problems
- Geant4 provides "Fast Simulation Interface" which can replace standard Geant4 processes during code execution to speed up simulations or implement external codes
  - It is activated only in a particular G4Region under certain conditions and for certain particles
  - Many applications for fast simulation
    - Electromagnetic shower in homogeneous and/or sampling calorimeters
    - Machine learning inference
    - Offloading specialized tasks to accelerators or heterogenous hardware (GPUs, …)