GEANT4
A SIMULATION TOOLKIT

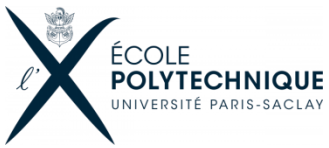
Version 11.2

Event Biasing

CERN Geant4 Advanced Course,
14th – 18th October 2024,

Marc Verderi
LLR, Ecole polytechnique

IMPERIAL



Jefferson Lab
Thomas Jefferson National Accelerator Facility

Geant4 ASSOCIATES INTERNATIONAL
Experts in Radiation Simulation

MANCHESTER 1824
The University of Manchester

Overview

I. Introduction

II. Existing Biasing Options before v10.0
- Primary Particle Biasing
- Options In Hadronic
- Geometry based importance biasing
- Weight Window Technique
- User defined biasing

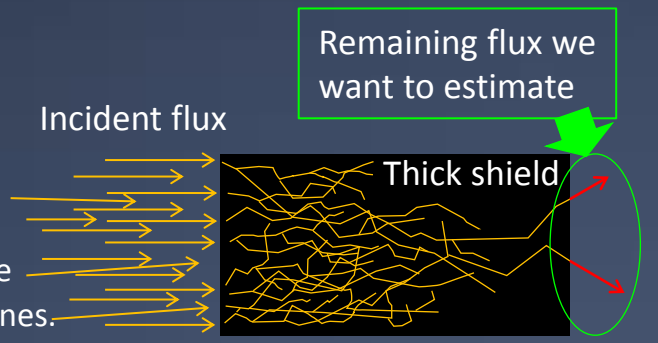
III. Reverse Monte-Carlo

IV. Generic Biasing Scheme

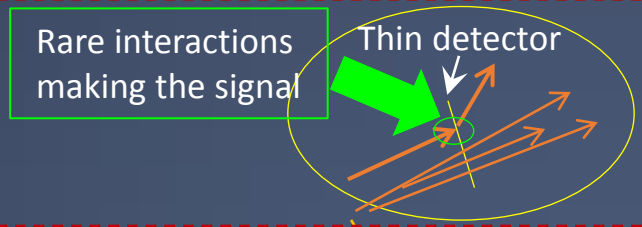
I. Introduction

Rare events

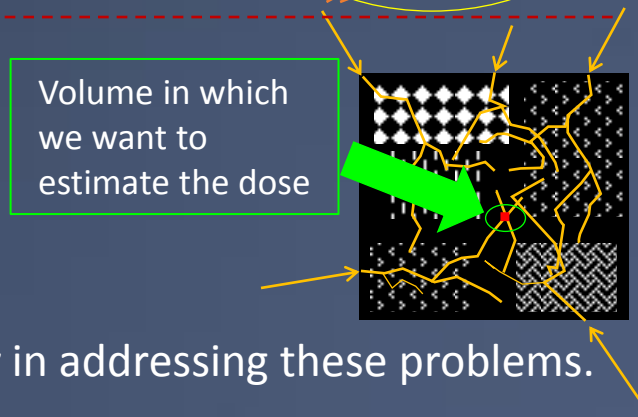
- There are simulation problems in which what we are interesting in is "*rare*"
  - rare because of the physics,
  - or because of the setup,
  - or both, etc.
- Examples of such problems:
  - Estimating the efficiency of a shield
    - A large flux enters a shield…
    - … with a lot of interactions inside, which is CPU intensive
    - … and only a tiny fraction of particles exiting : the rare ones.



  - Obtaining the response of a very thin detector
    - For example to obtain the response to neutrons
    - Most of them do not interact in
    - But the signal is made by the rare ones which interact



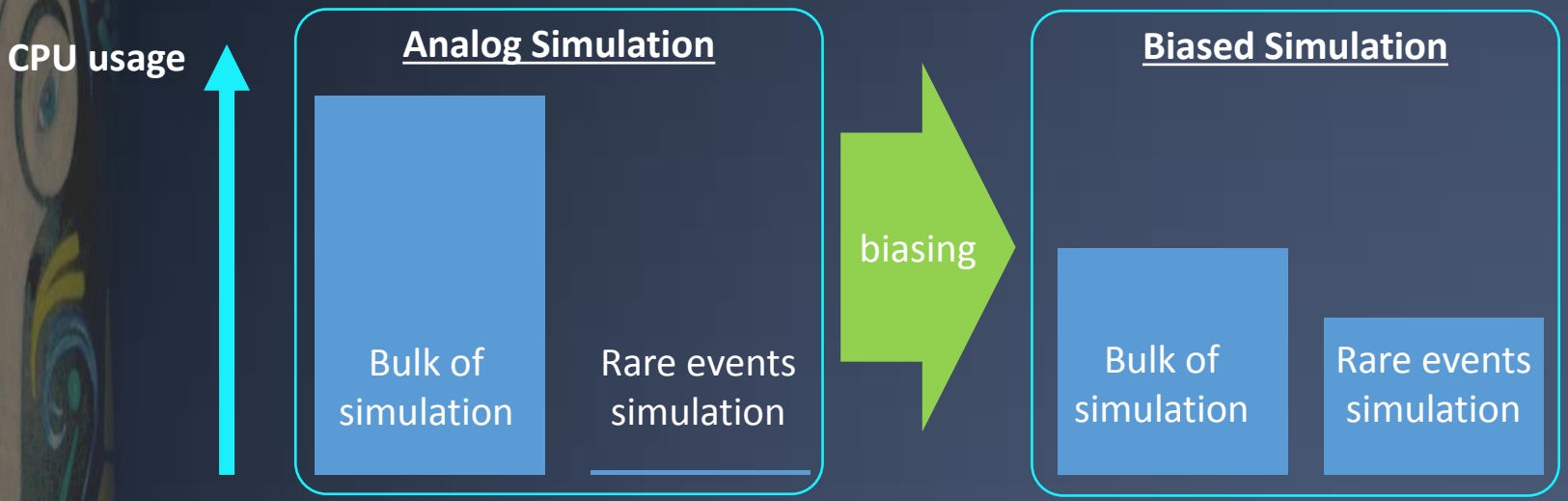
  - Estimating the dose in a very small part of a big setup
    - For example an electronic chip inside a satellite
    - Most of the time is spent in simulating showers that do not contribute the dose
    - While we want to tally this dose



- An "analog" (usual) simulation is ***very inefficient*** in addressing these problems.

What is “event biasing” ?

- “Event biasing” (or “biasing” or “variance reduction”) is a set of **techniques to simulate rare events efficiently**
  - It is an accelerated simulation with respect to these events
    - And to these events only : ie not everywhere !
  - It focuses/transfers the CPU power usage on what we want to tally



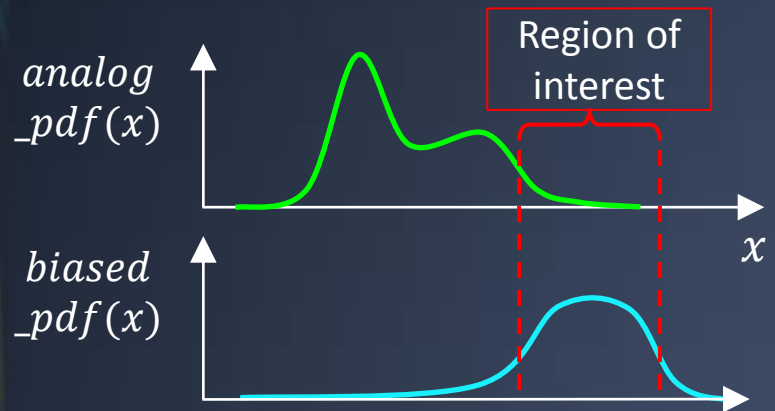
  - It can provide LARGE CPU improvements
    - Several orders of magnitude ! Depending on the problem.
- “Biasing” stands for “biased simulation”:
  - “Biased” because rare events are enhanced compared to the analog simulation.

Main Principles

› There is a large variety of techniques (sometimes with ≠ names for ≈ techniques) but there are essentially two underneath classes of techniques:

Importance Sampling

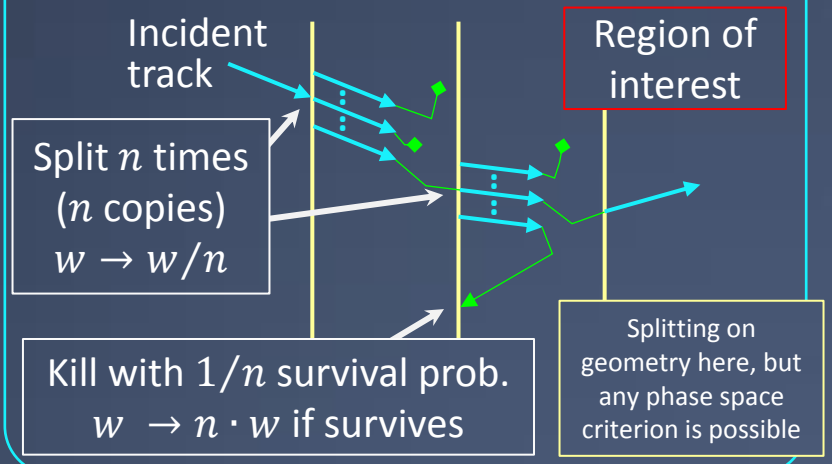
- Modify physics pdf's enhancing "important" parts of them



$$w(x) = \frac{analog\_pdf(x)}{biased\_pdf(x)}$$

Splitting (& Killing)

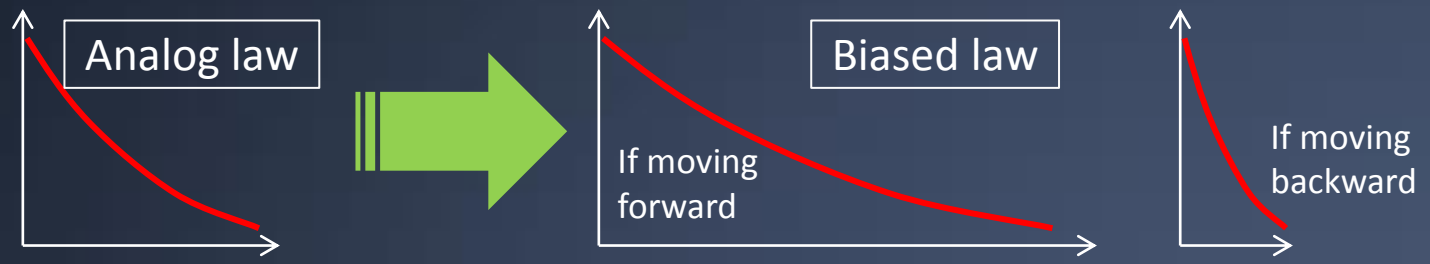
- Keep pdf's unchanged but split (kill) when moving towards (receding from) region of interest



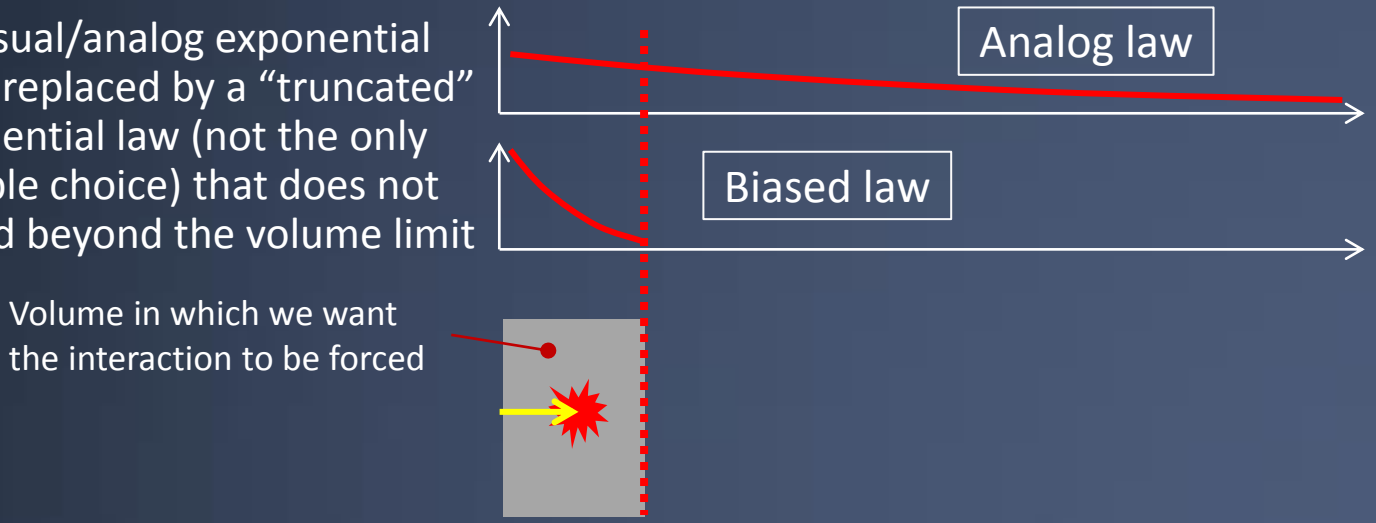
› Enhancement of rare events is tracked with statistical "weights":
  - Weights are used to "de-bias" and go back to analog quantities:
    - Eg : $E_{deposit} = \sum_{i=track} weight_i \times E^i_{deposit}$

Importance Sampling Techniques Examples

- **The "exponential transform" technique**
  - It is a change of the total cross-section
    - Made direction dependent
  - The usual/analog exponential law is replaced by an other law (still an exponential one here, but with different cross-section)

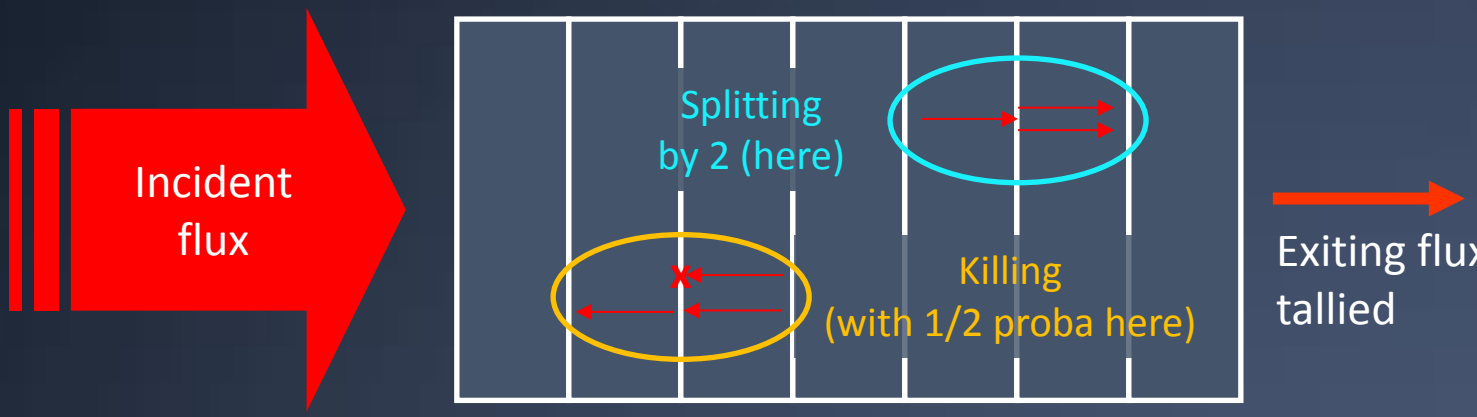


- **Interaction forcing in a thin volume:**
  - The usual/analog exponential law is replaced by a "truncated" exponential law (not the only possible choice) that does not extend beyond the volume limit



Example of Splitting

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



- Defines artificial slices (not too thick [inefficient], not too thin [divergence])
- On these slices boundaries tracks are:
  - Split (cloned in identical copies) if moving towards the “tally”
  - Randomly killed (so-called “Russian Roulette”) if moving away from the tally
  - The splitting is for compensating the physical absorption in the shield material
- Then apply splitting/killing weight calculation rule seen before, ie:
  - When splitting happens, the track copies receive a weight which is ½ (here) of the initial track weight
  - When Russian Roulette happens, the track survives with probability $p_{survival}$ = 50% (here); if it survives its weight is multiplied by $1/p_{survival}$.

Biasing : strength & difficulties

- **Strength of biasing:**
  - The physics at play in biasing is the ***same*** than with the detailed simulation
    - Biasing is “simply” an other way to process the full detailed physics
  - Hence, results obtained with biasing are **statistically identical** to 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 of biasing)
  - Ensuring a proper convergence can be difficult
    - Issue of random appearance of very large weights
      - that destroy the convergence
    - This signs that parts of the problem are not sampled enough
      - See backup slides
  - In most techniques, correlations between observables are lost

II. Existing Biasing Options before v10.0

Primary Particle Biasing

- The General Particle Source (GPS) allows to bias the particle source:
  - Position
  - Angular Distribution
  - Energy Distribution
- This is an “importance sampling” biasing
  - The physical distributions are replaced by biased ones
- Primary particles are given a weight that is the ratio of analog / biased probabilities
  - In the simulation, all daughter, gran-daughter,… particles from the primary inherit the weight of this primary

Options In Hadronic (1)

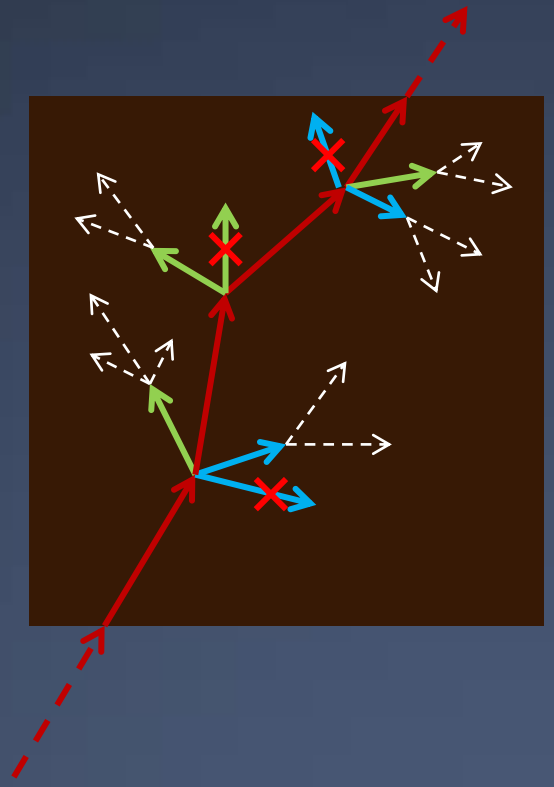
- Cross-section Biasing:
  - Possibility to enhance a (low) cross-section
    - This an “importance sampling” approach
  - Available for photon inelastic, electron & positron nuclear processes

```
//init reaction model
auto theElectroReaction = new G4ElectroNuclearReaction();

//create process and register model
G4ElectronNuclearProcess theElectronNuclearProcess;
theElectronNuclearProcess.RegisterMe(theElectroReaction);

//apply biasing
theElectronNuclearProcess.BiasCrossSectionByFactor(100);

//register process
pManager->AddDiscreteProcess(&theElectronNuclearProcess);
```

  - (no messenger for this ?)

Options In Hadronic (2)

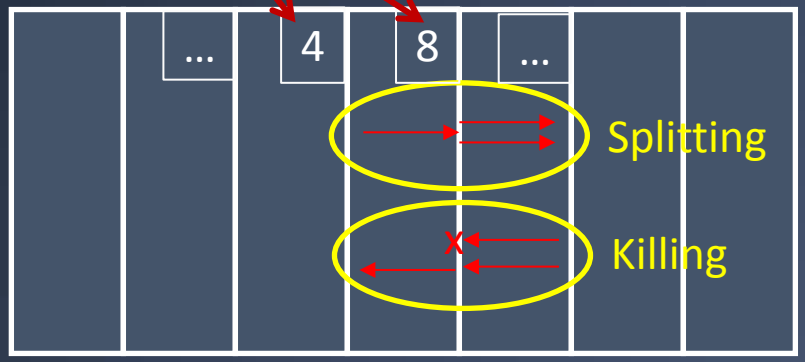
- Leading Particle Biasing:
  - It is a technique used for estimating the penetration of particles in a shield
  - Instead of simulating the full shower, at each interaction, only keep:
    - The most energetic particle
    - + randomly one particle of each species
  - This is a « killing » - based biasing



- Radioactive decay
  - Enhance particular decay within a given time window
  - Sample all decay modes with equal probabilities
  - Split parent nuclide in a user defined number of copies, letting the decay go
  - Enhance emission in a particular direction

Geometry based importance biasing

- A direct application of what was presented earlier here
- Attach “importance” to cells in geometry
  - Not to be confused with “importance sampling” : ie 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 » (eg, 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 (eg, here)

Geometry based importance biasing

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

```
auto istore = G4IStore::GetInstance();
// Loop on physical volumes:
istore->AddImportanceGeometryCell(importanceValue,  physicalVolumePointer [, nReplica]);
```

Replica number, if any

  - All volumes should be given an importance value.

- Biasing may be applied to some particle type only: eg neutron. In the main program, this is specified as:

```
G4GeometrySampler geometrySample(worldVolume,"neutron");
```

  - (worldVolume specified as there is the ability to define cells in a parallel geometry: not discussed here)

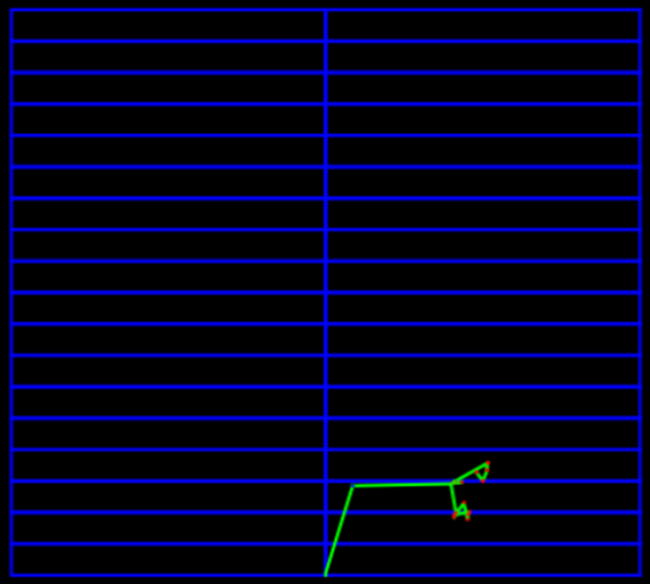
- The actual splitting and killing are handled by dedicated processes. These are to be inserted in the physics list. If using a modular physics, it is done as:

```
auto physicsList = new FTFP_BERT;
physicsList->RegisterPhysics(new G4ImportanceBiasing(&geometrySampler));
```

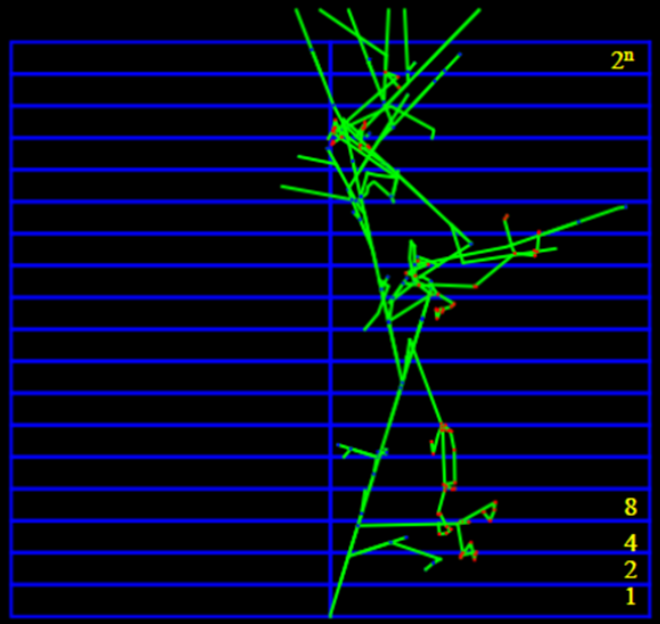
- More details can be found in examples:
  - examples/extended/biasing/B01 : geometry biasing with importance cells or weight window
  - examples/extended/biasing/B02 : same, with cells in a parallel geometry
  - examples/extended/biasing/B03 : same, for the case of a non modular physics list

Example B01 - 10 MeV neutrons, thick concrete cylinder

Analogue Simulation



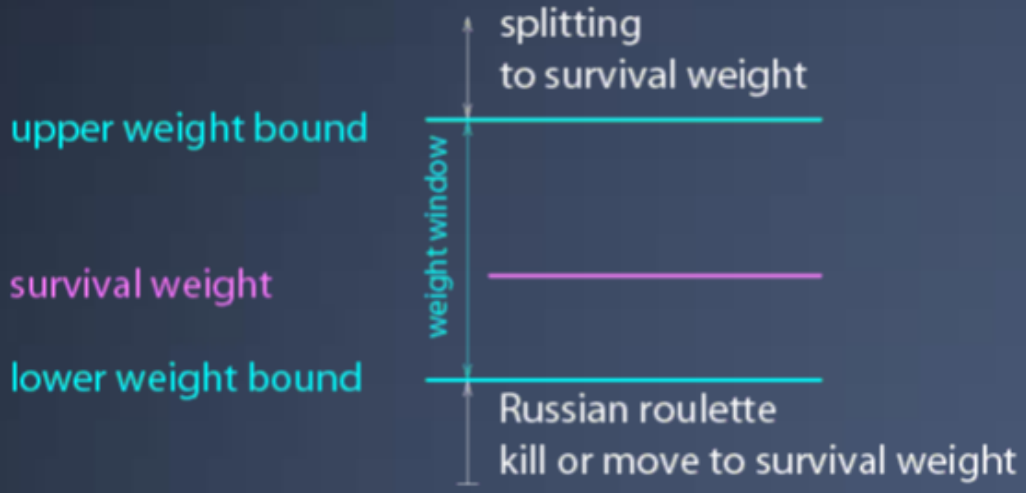
Importance Sampled



See: geant4/examples/extended/biasing/B01

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: geant4/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:
  1. Create derived class inheriting from G4WrapperProcess
  2. Override only the methods to be modified, e.g., PostStepDoIt()
  3. Register this class in place of the original
  4. Finally, register the original (wrapped) process with user class

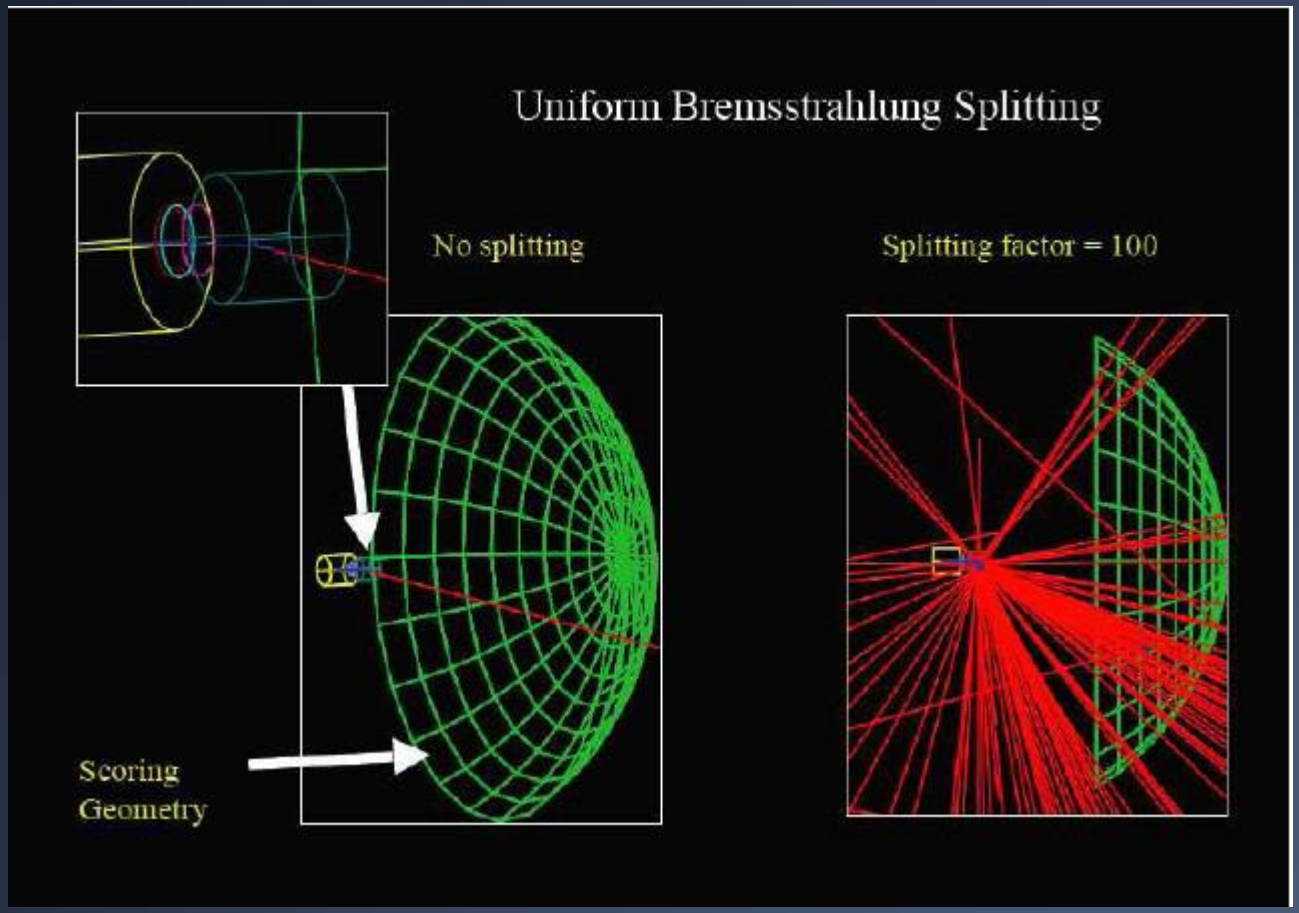
Example with brem. splitting

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

```
G4VParticleChange* myWrappedProc::PostStepDoIt(const G4Track& track, const G4Step& step)
{

    auto weight = track.GetWeight()/fNSplit;

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

    // Secondary store
    // Loop over wrapped PSDI method to generate multiple secondaries
    for (G4int i=0; i<fNSplit; i++)
        {
            particleChange = pRegProcess->PostStepDoIt(track, step);
            assert (0 != particleChange);
            particleChange->SetVerboseLevel(0);
            // Save the secondaries generated on this cycle
            for (G4int j=0; j<particleChange->GetNumberOfSecondaries(); j++)
                {
                    secondaries.push_back (new
                                           G4Track(*(particleChange->GetSecondary(j))));
                }
        }
        // -- and then pass these secondaries to the particle change…
}
```

Brem. Process in this case

Example with a brem. splitting

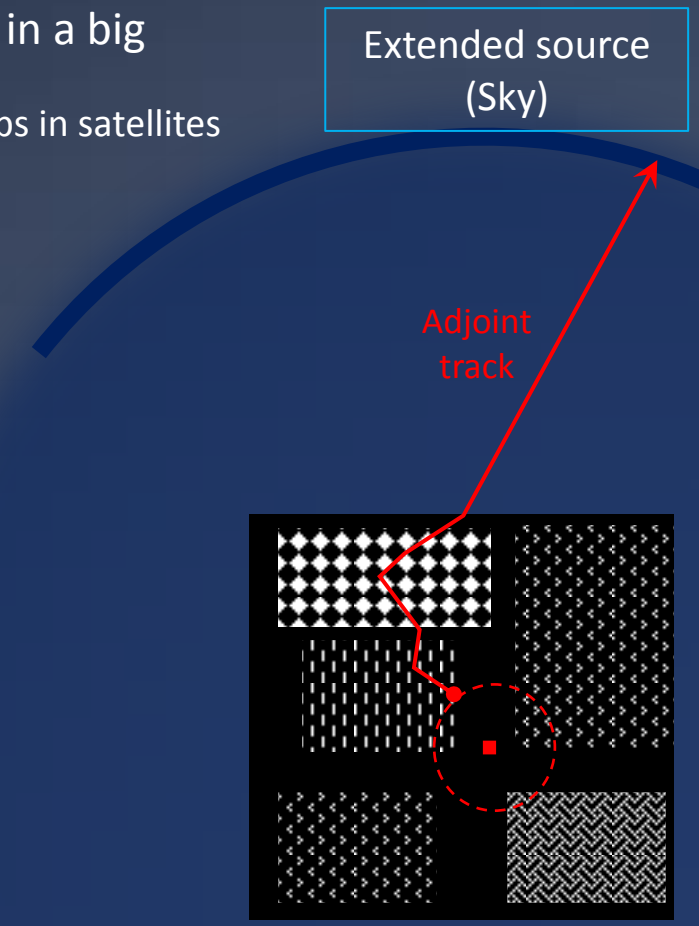


› Note that EM packages proposes a brem-splitting option also.

III. Reverse Monte-Carlo

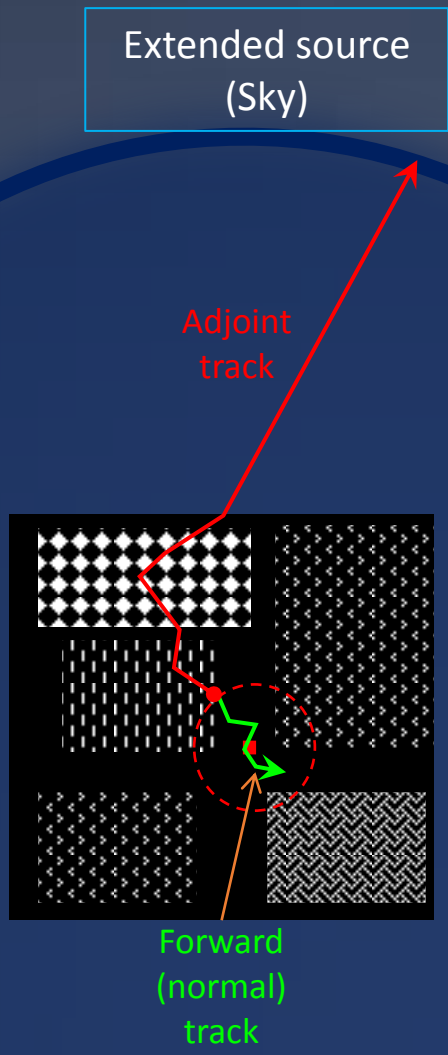
Reverse Monte Carlo

- Simulation in which particles go back in time !
- Useful in case of small device to be simulated in a big environment
  - Requested by ESA, for dose study in electronic chips in satellites
- Simulation starts with the red track
  - So-called “adjoint” track
  - Going back in time
  - Increasing in energy
  - Up to reaching the extended source

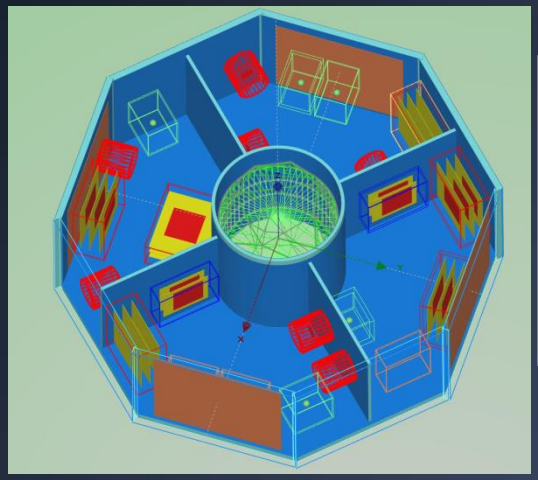


Reverse Monte Carlo

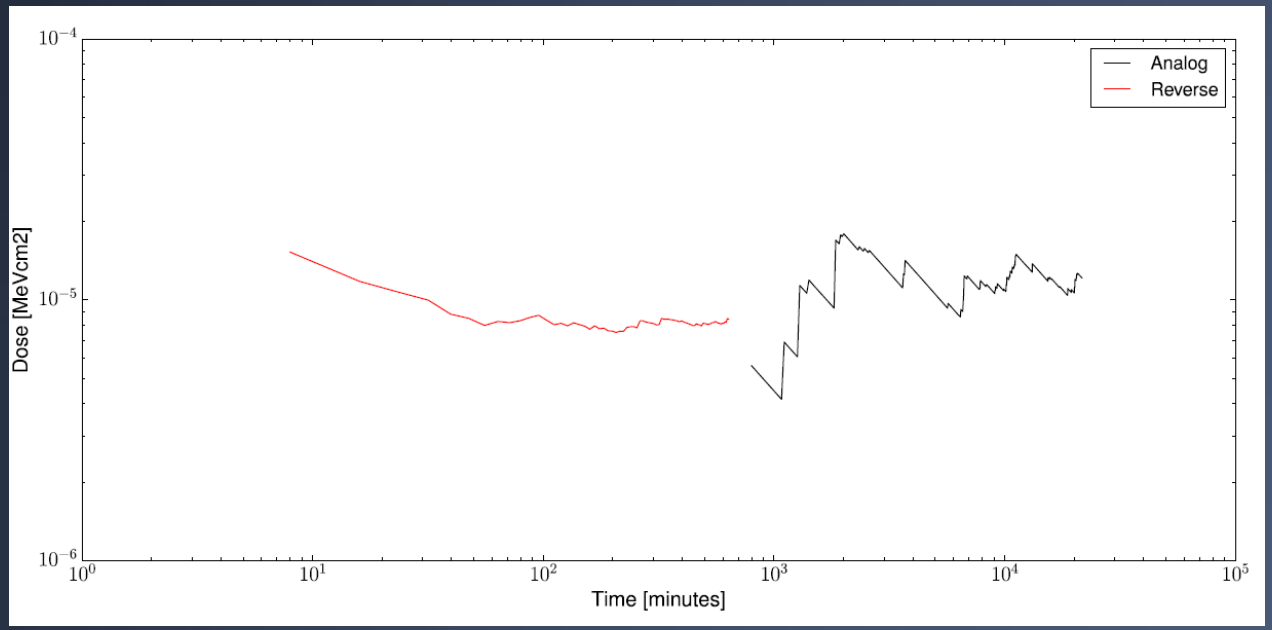
- Simulation in which particles go back in time !
- Useful in case of small device to be simulated in a big environment
  - Requested by ESA, for dose study in electronic chips in satellites
- Simulation starts with the red track
  - So-called “adjoint” track
  - Going back in time
  - Increasing in energy
  - Up to reaching the extended source
- If track energy < high energy limit of the sky source spectrum
  - Proceed with a usual “forward” simulation (green track)
- When many track simulated, adjust adjoint track weights so that energy spectrum of adjoint particle matches sky source spectrum
  - This provides the weight of the green tracks
- Dose in chip can then be computed, accounting for the forward track contribution, with the proper weight.
- See: geant4/examples/extended/biasing/ReverseMC01



Example of convergence speed



- Electron dose with ESA “Spacecraft_test” geometry
- Middle Erath Orbit spectrum
- Courtesy of Laurent Desorgher (chuv)



IV. Generic Biasing Scheme

Generic Biasing Scheme

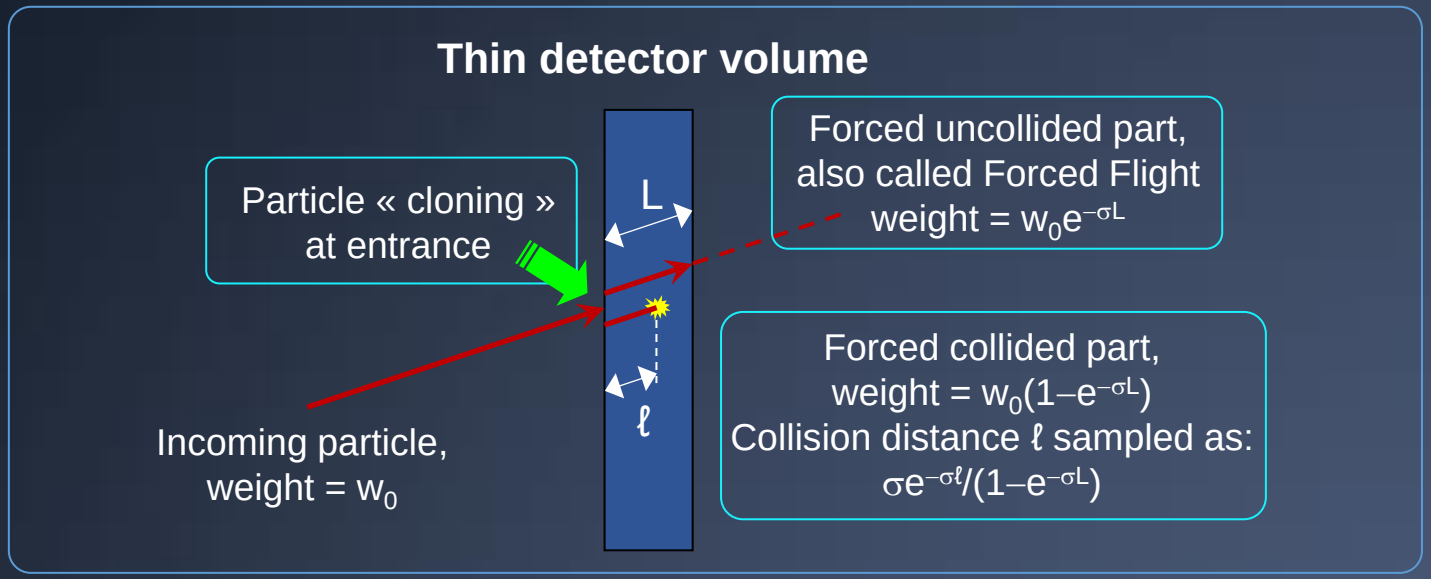
- **Adopt an "object oriented" (OO) scheme to handle any type of (forward) biasing**
  - It defines an abstract layer, general enough
  - It provides some ready-to-go options based on this layer
  - It is open to users, for customized implementations

- It handles:
  - Physics process biasing → denoted as "physics biasing"
  - Splitting/killing → denoted as "non-physics biasing"

- It is aimed at being "modular":
  - By defining simple techniques (kind of "primitive of biasing")
  - That are combined to provide a given biasing scheme

Modularity ?

› Example of “force collision” scheme à la MCNP



› The “particle cloning”, “forced non-collision”, “forced collision” actions can be seen as sort of biasing primitives, we call “biasing operations”
  - They are combined here to provide the “force collision” scheme
  - But they could be used in other schemes

› Generic Biasing Scheme idea:
  - Define a class for **“biasing operations”**
  - Define a class for **“biasing operators”**, that decide for biasing operations
  - Provide the **interface** of these classes with the tracking

Generic Biasing Components

› G4VBiasingOperation

- An abstract class
- Can act on a physics process:
  - › By modifying its interaction law
  - › By modifying its final state generation
- Can act by itself
  - › To split or randomly kill particles

› G4VBiasingOperator

- An abstract class
- Selects G4VBiasingOperation's
  - › At the beginning of the step
  - › At the final state generation level
- ***This is the entity which is making all the decisions***

› G4BiasingProcessInterface

- A concrete class
- Which can wrap a physics process, to bias it
  - › Applying "physics biasing" options

G4BiasingProcessInterface
G4GammaConversion

- Which can have no process, to apply splitting & killing techniques
  - › Applying "non-physics biasing" options

G4BiasingProcessInterface
(no process)

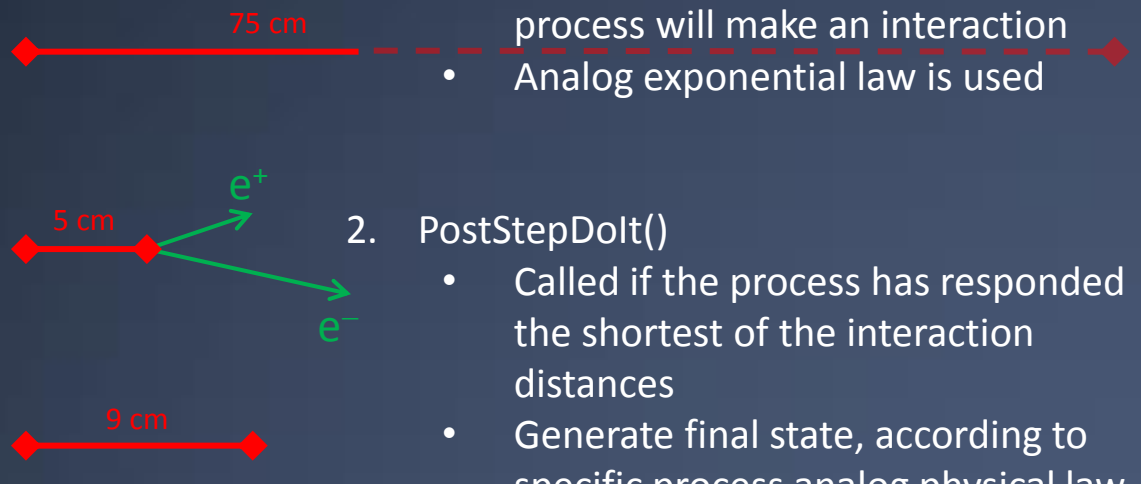
- It makes the interface between the biasing classes and the tracking
  - › By collecting and applying the decisions of the current G4VBiasingOperator (if any)

“physics biasing”

G4PhotoelectricEffect

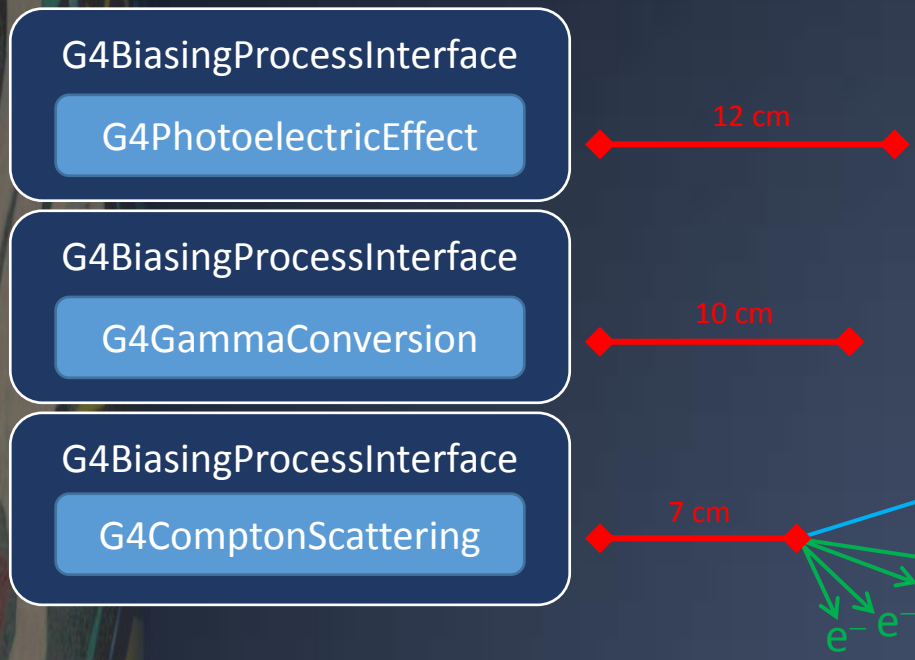
G4GammaConversion

G4ComptonScattering



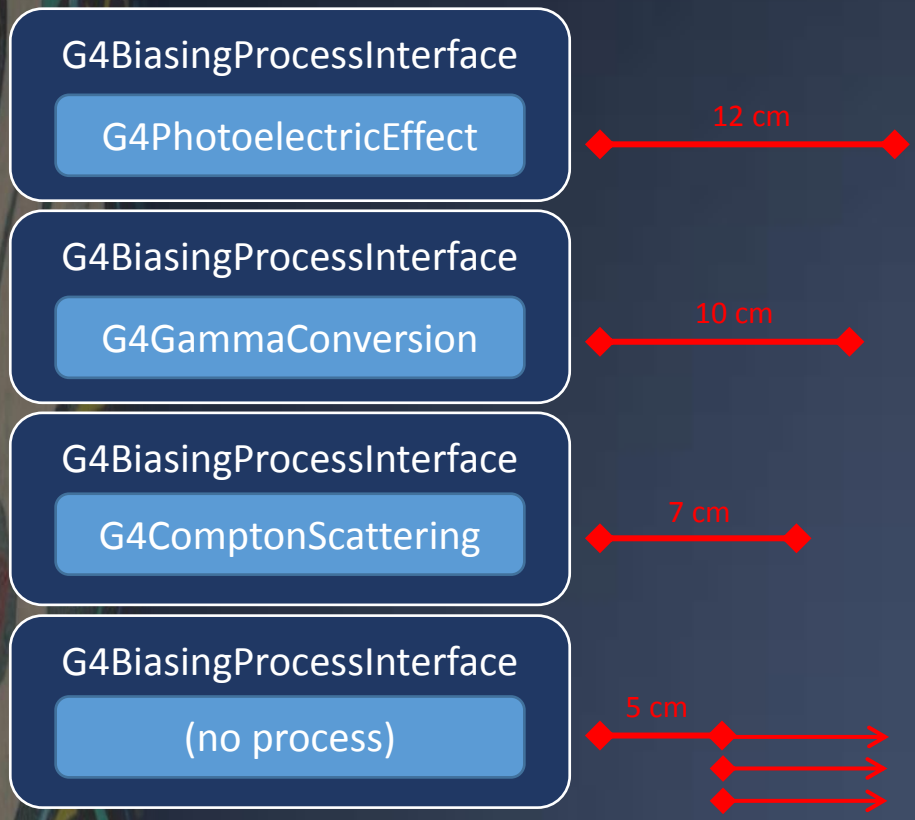
1. GetPostStepPhysicalInteractionLength()
   - Returns the distance at which the process will make an interaction
   - Analog exponential law is used
2. PostStepDoIt()
   - Called if the process has responded the shortest of the interaction distances
   - Generate final state, according to specific process analog physical law

“physics biasing”



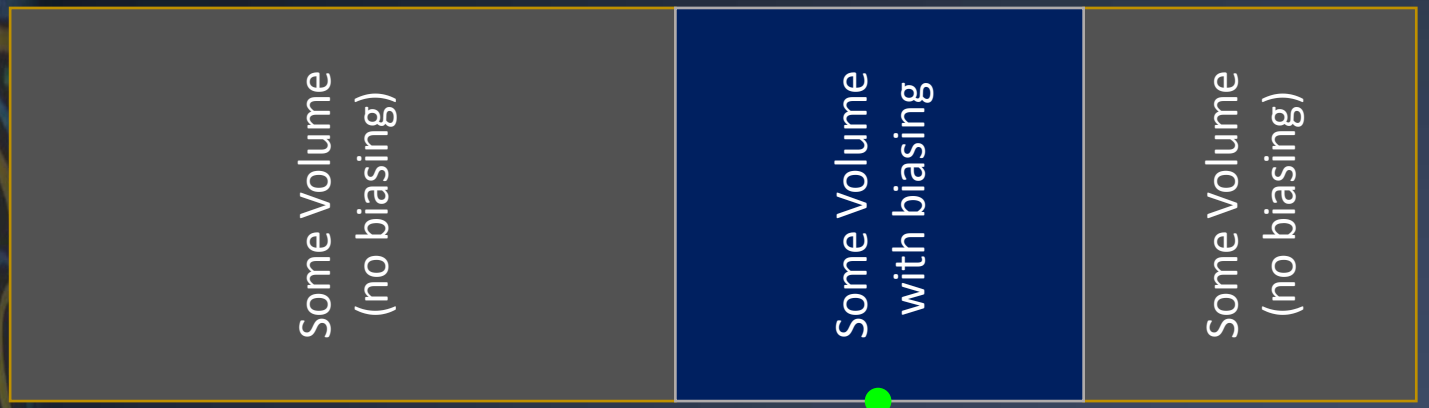
1. GetPostStepPhysicalInteractionLength()
   - Returns the distance at which the process will make an interaction
   - ~~Analog exponential law is used~~
   - A biased interaction law is used (if wished)
2. PostStepDoIt()
   - Called if the process has responded the shortest of the interaction distances
   - ~~Generate final state, according to specific process analog physical law~~
   - Generate final state, according to biased law (if wished)

“physics biasing” + “non-physics biasing”



1. GetPostStepPhysicalInteractionLength()
   - Returns the distance at which the process will make an interaction
   - ~~Analog exponential law is used~~
   - A biased interaction law is used (if wished)
2. PostStepDoIt()
   - Called if the process has responded the shortest of the interaction distances
   - ~~Generate final state, according to specific process analog physical law~~
   - Generate final state, according to biased law (if wished)

- “non-physics biasing” competes with other processes (as any process)
  - Here, winning the race
  - Here, applying a splitting by 3



G4BiasingProcessInterface
(no process)

G4BiasingProcessInterface
G4PhotoelectricEffect

G4BiasingProcessInterface
G4GammaConversion

G4BiasingProcessInterface
G4ComptonScattering

MyBiasingOperator

G4FlatForce-InteractionOperation

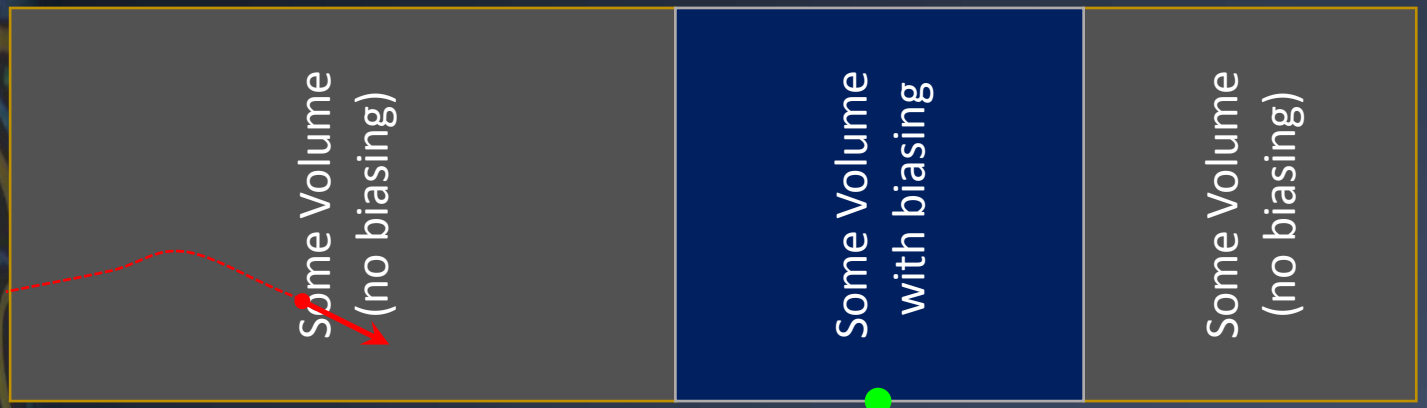
G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator



G4BiasingProcessInterface
(no process)

G4BiasingProcessInterface
G4PhotoelectricEffect

G4BiasingProcessInterface
G4GammaConversion

G4BiasingProcessInterface
G4ComptonScattering

MyBiasingOperator

G4FlatForce-InteractionOperation

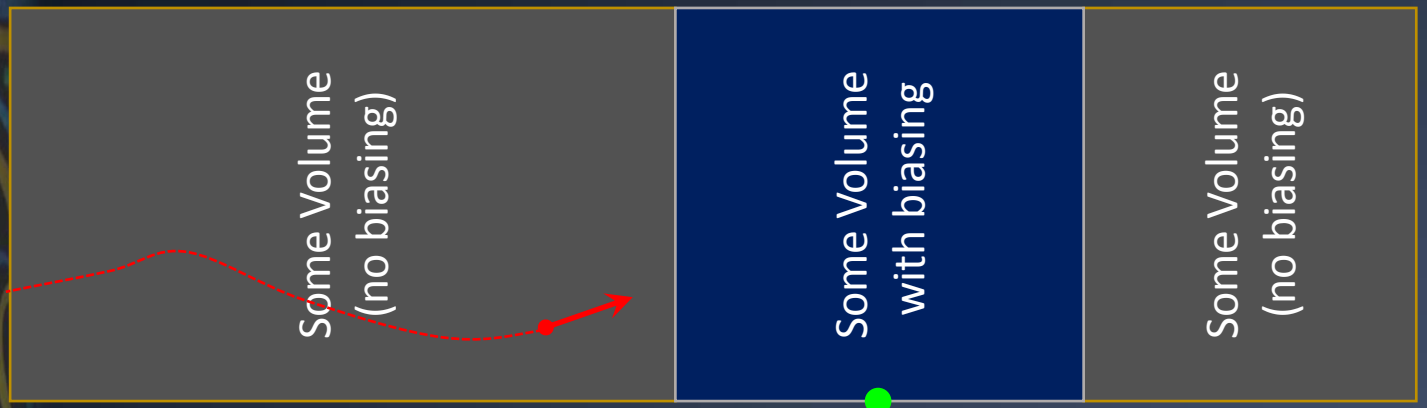
G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator



G4BiasingProcessInterface
(no process)

G4BiasingProcessInterface
G4PhotoelectricEffect

G4BiasingProcessInterface
G4GammaConversion

G4BiasingProcessInterface
G4ComptonScattering

MyBiasingOperator

G4FlatForce-InteractionOperation

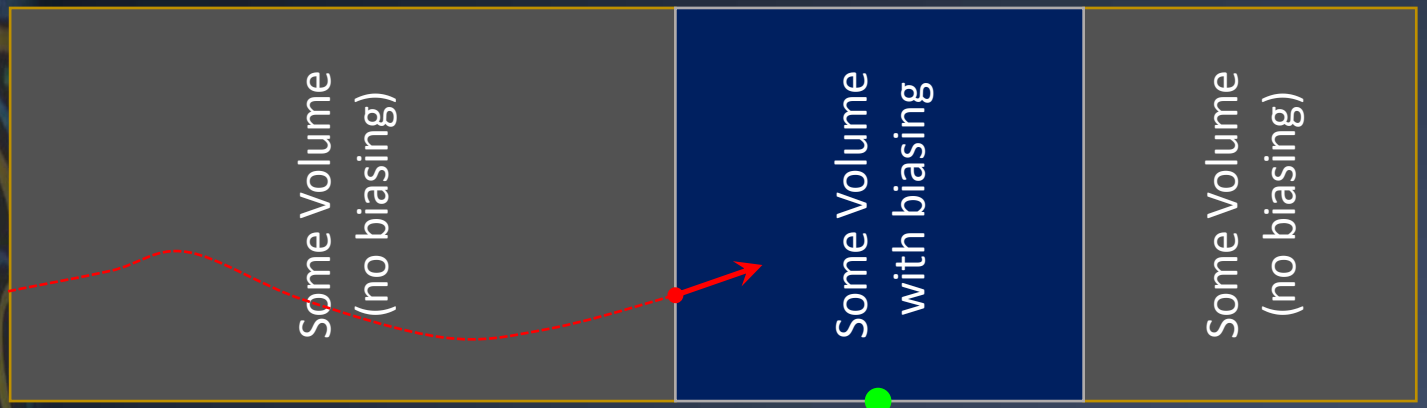
G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator



G4BiasingProcessInterface
(no process)

G4BiasingProcessInterface
G4PhotoelectricEffect

G4BiasingProcessInterface
G4GammaConversion

G4BiasingProcessInterface
G4ComptonScattering

MyBiasingOperator

G4FlatForce-InteractionOperation

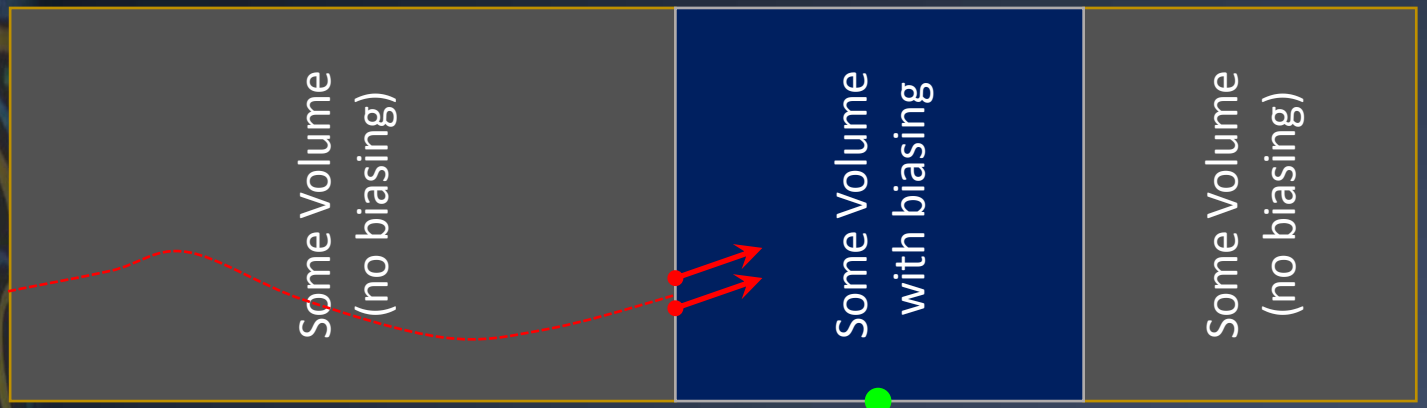
G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator



G4BiasingProcessInterface
(no process)

G4BiasingProcessInterface
G4PhotoelectricEffect

G4BiasingProcessInterface
G4GammaConversion

G4BiasingProcessInterface
G4ComptonScattering

MyBiasingOperator

G4FlatForce-InteractionOperation

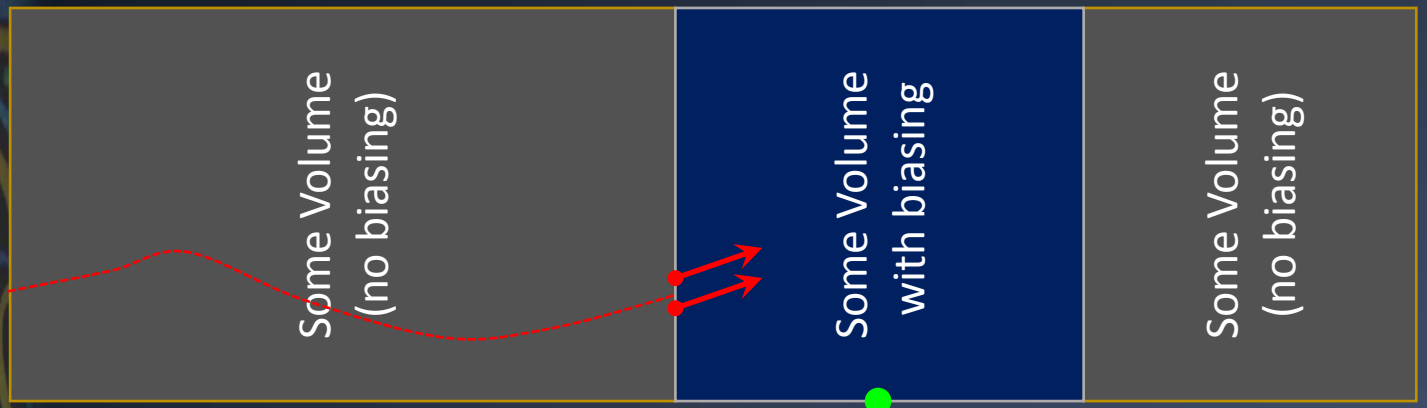
G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator



G4BiasingProcessInterface
(no process)

G4BiasingProcessInterface
G4PhotoelectricEffect

G4BiasingProcessInterface
G4GammaConversion

G4BiasingProcessInterface
G4ComptonScattering

MyBiasingOperator

G4FlatForce-InteractionOperation

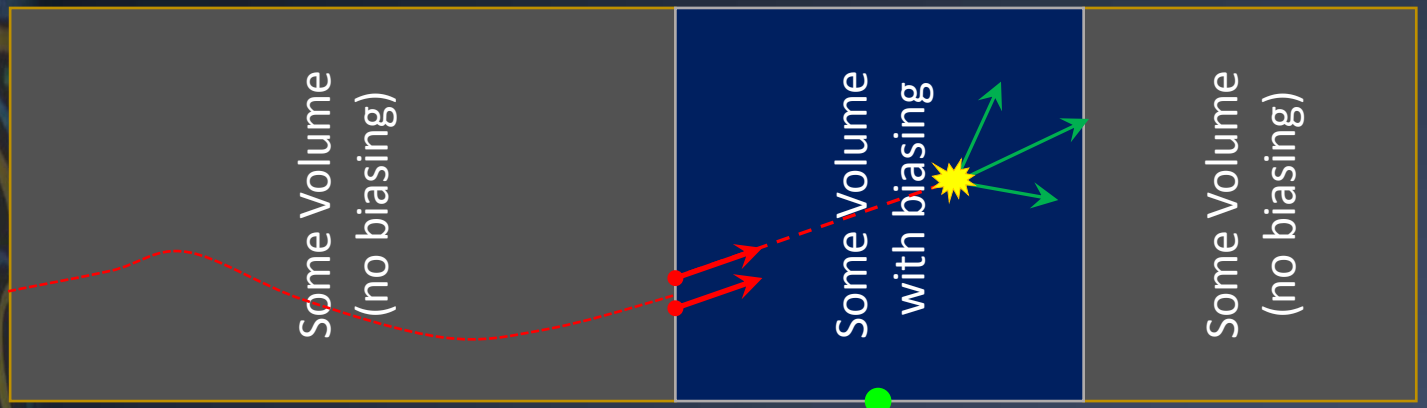
G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator



G4BiasingProcessInterface
(no process)

G4BiasingProcessInterface
G4PhotoelectricEffect

G4BiasingProcessInterface
G4GammaConversion

G4BiasingProcessInterface
G4ComptonScattering

MyBiasingOperator

G4FlatForce-InteractionOperation

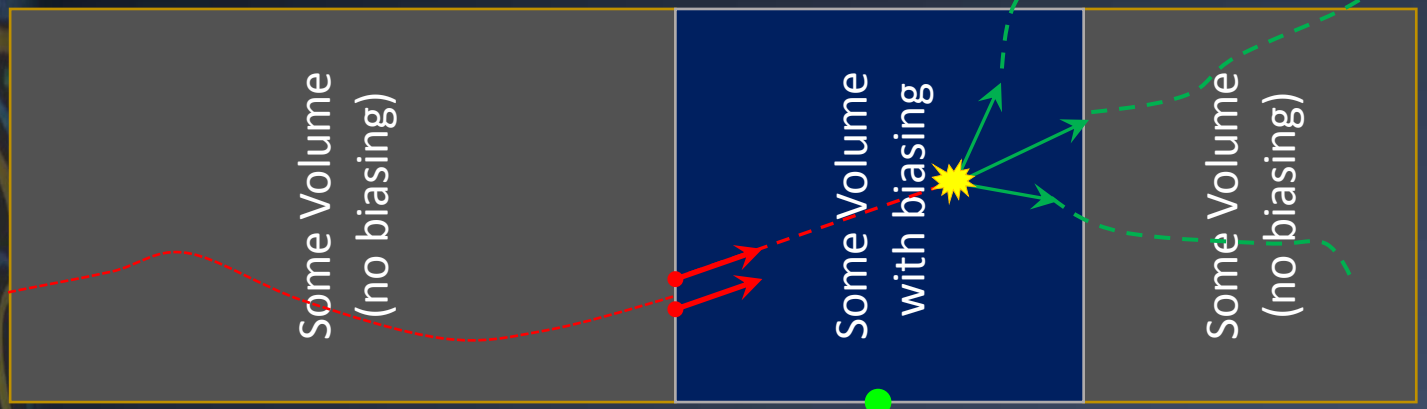
G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator



G4BiasingProcessInterface
(no process)

G4BiasingProcessInterface
G4PhotoelectricEffect

G4BiasingProcessInterface
G4GammaConversion

G4BiasingProcessInterface
G4ComptonScattering

MyBiasingOperator

G4FlatForce-InteractionOperation

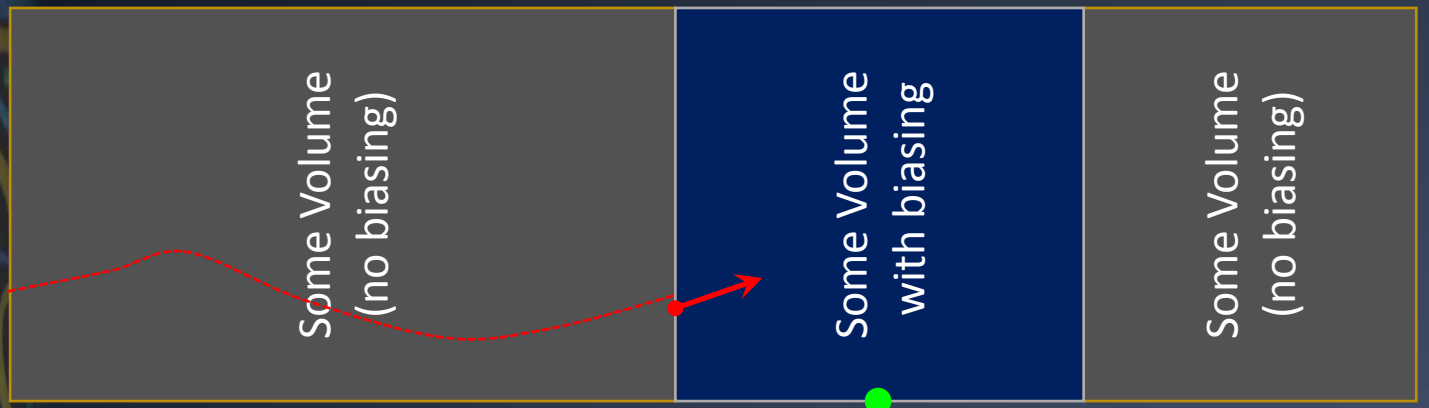
G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator



G4BiasingProcessInterface
(no process)

G4BiasingProcessInterface
G4PhotoelectricEffect

G4BiasingProcessInterface
G4GammaConversion

G4BiasingProcessInterface
G4ComptonScattering

MyBiasingOperator

G4FlatForce-InteractionOperation

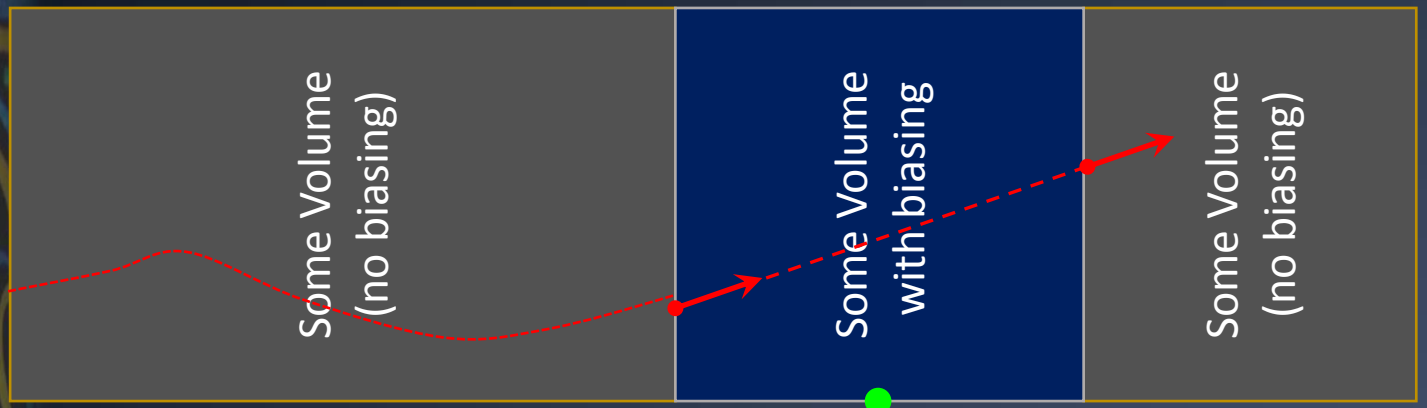
G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator



G4BiasingProcessInterface
(no process)

G4BiasingProcessInterface
G4PhotoelectricEffect

G4BiasingProcessInterface
G4GammaConversion

G4BiasingProcessInterface
G4ComptonScattering

MyBiasingOperator

G4FlatForce-InteractionOperation

G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator

How to use

- In the main:

```
// -- Select a modular physics list
auto physicsList = new FTFP_BERT;
// -- And augment it with biasing facilities:
auto biasingPhysics = new G4GenericBiasingPhysics();
biasingPhysics->Bias("gamma");
biasingPhysics->Bias("neutron");
physicsList->RegisterPhysics(biasingPhysics);
runManager->SetUserInitialization(physicsList);
```

  - More options exist for the G4GenericBiasingPhysics constructor:
    - Activate biasing for processes only
    - Or for doing splitting/killing only
    - Or to activate biasing for all charged
    - Or all neutral particles
    - Etc.

- In ConstructSDandField() of detector construction:

```
auto biasingOperator = new MyBiasingOperator();
biasingOperator->AttachTo( logicalVolumeToBias );
```

Existing functionalities & examples

- geant4/examples/extended/biasing/GB01 :
  - Individual process cross-section biasing
  - Implemented for neutral particles
  - Charged particle case requires development
    - Issue : variation of cross-section during step because of energy loss
- geant4/examples/extended/biasing/GB02 :
  - Force collision à la MCNP
  - Implemented for neutral particles (as MCNP)
- geant4/examples/extended/biasing/GB03 :
  - Geometry importance + further option
  - Scheme augmented compared to classical geometry importance
    - Allows kinds of “intermediate” non-integer splitting values
- geant4/examples/extended/biasing/GB04 :
  - Re-implementation of a classical Bremsstrahlung splitting
- geant4/examples/extended/biasing/GB05 :
  - Illustrates a "splitting by cross-section“
  - An invention (to my knowledge) where the splitting rate is directly adjusted from absorption cross-sections
- geant4/examples/extended/biasing/GB06 :
  - Parallel geometries with generic biasing.
- geant4/examples/extended/biasing/GB07 :
  - Implement a “particle leading” biasing scheme

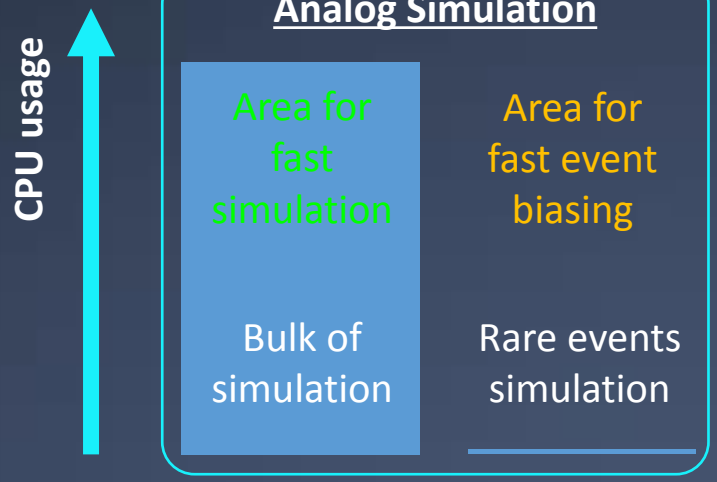
Summary

- Biasing techniques can provide very large acceleration factors in problems in which we must tally rare events
- Geant4 proposes biasing options
  - Primary source (GPS) , leading particle, cross-section (had), radioactive decay, splitting with importance in geometry, weight window, user biasing with G4WrapperProcess, and bremsstrahlung splitting
  - Reverse Monte-Carlo
- Generalized “OO” approach since v10.0
  - Biasing of process interaction and process final state
  - Together with with killing, splitting
  - Flexible scheme, allowing implementation of many options, and open to users for further customized extensions
- Biasing techniques delicate to handle, but sometimes unmissable when dealing with rare event problems !

Backup

Biasing versus other acceleration techniques ?

- Other acceleration techniques exist:
  - "fast simulation" : use approximate but faster model (eg: EM showers)
    - Accelerates the bulk of the simulation
  - Deterministic computation
    - in some cases (eg : medical, space)
    - no such techniques in Geant4



- Strength of biasing:
  - The physics at play in biasing is the ***same*** than with the detailed simulation
    - Biasing is "simply" an other 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 very large weights
      - that destroy the convergence
    - This signs that parts of the problem are not sampled enough (ex. in backup slides)

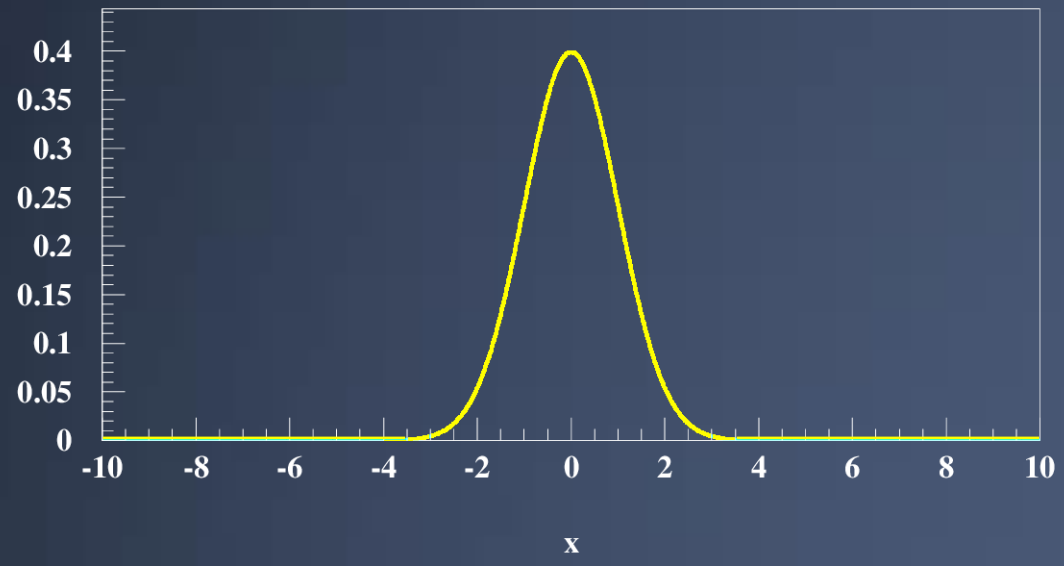
Some Words of Caution with Convergence

Introduction

- Biasing techniques are powerful
  - If a technique is well suited to a problem, gains in simulation efficiency by several order of magnitudes are usual
- In contrast to an analog simulation where all “events” have the same weight (==1), biasing is not “democratic”
  - Because of the weight, some events are more important than others
- This weight disparity may cause convergence problems
  - That can be nasty !
  - What happens is that, from time to time, an event with big weight may come, and change drastically the –say- estimated current mean value
  - Always a fear that this may happen…
- Proper convergence is the issue the user of biasing has to care about
  - And this is the price to pay for using biasing
- Let’s illustrate a “bad convergence case” with a simple problem
  - But this allows to spot the difficulties

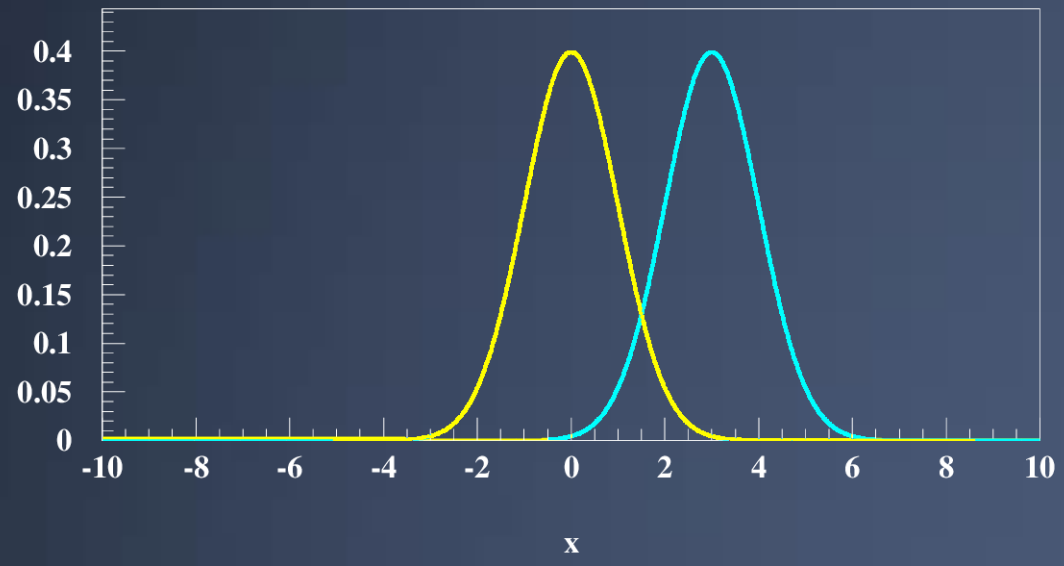
Convergence

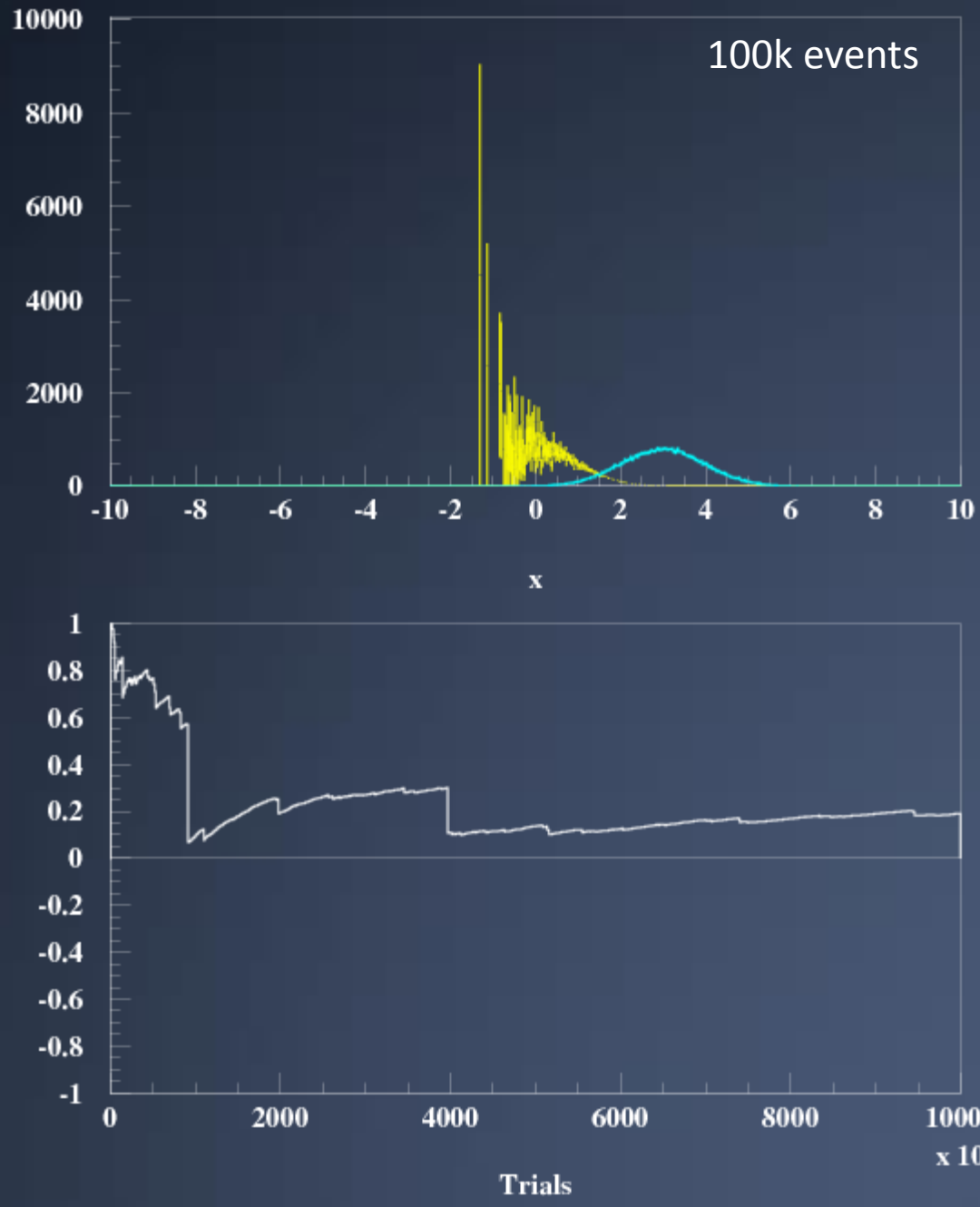
- A « toy » example of a bad sampling:
  - We want to estimate the mean value of the unknown « yellow » distribution.
  - As we don’t know where it is, we try to sample it with the « blue » distribution
    - And we know how to compute the weight



Convergence

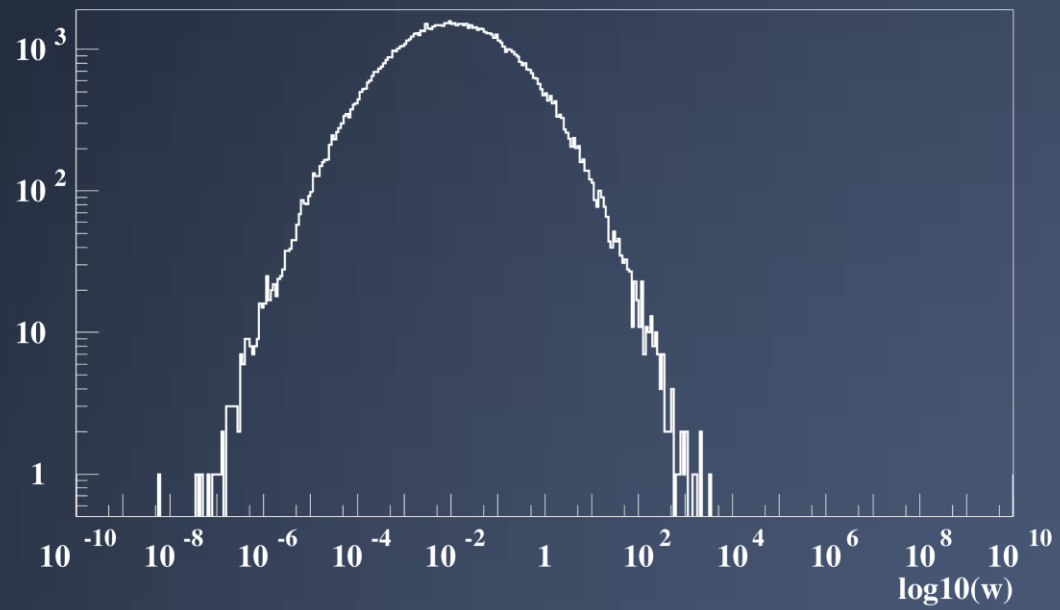
- A « toy » example of a bad sampling:
  - We want to estimate the mean value of the unknown « yellow » distribution.
  - As we don’t know where it is, we try to sample it with the « blue » distribution
    - And we know how to compute the weight



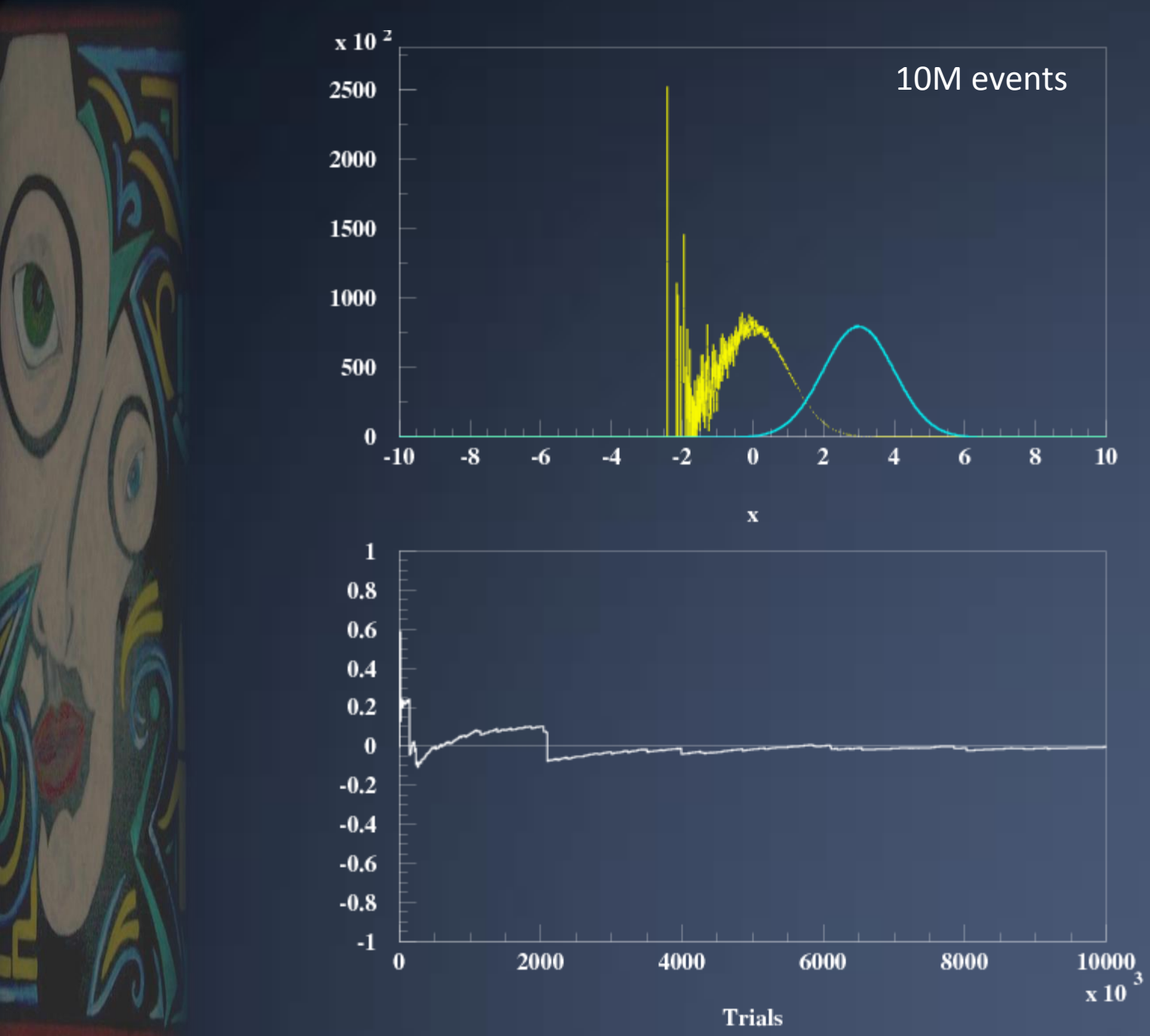

100k events
10000
8000
6000
4000
2000
0
-10
-8
-6
-4
-2
0
2
4
6
8
10
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
2000
4000
6000
8000
10000
x 10
Trials

Weight distribution…

- Lots of tiny weights…



- Few huge ones…


x 10 2
2500
2000
1500
1000
500
0
-10
-8
-6
-4
-2
0
2
4
6
8
10
x
10M events
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
2000
4000
6000
8000
10000
x 10 3
Trials

Some qualitative observations

- Observations that can help spotting the inappropriate sampling:
  - We have a wide variety of weights
    - Many tiny ones
      - Waste of time to determine the mean value
    - Few huge ones
      - Responsible for jumps
    - Such problem –if can not be improved- could at least be alleviated with a weight window technique
      - In a real problem (useless here in our toy problem)
  - We have huge weights from time to time
    - These are not wrong !
    - Temptation would be to “dismiss” these events
    - But in our case, these events bring down the mean value to the correct value
    - These are not wrong, but ***their presence is a sign of the problematic sampling***
  - We observe monotonic increase of the mean value
    - We are sampling only “one side” of the problem
- Convergence criteria have been established in other packages
  - Like MCNP
- You are invited at staying critical when using such biasing techniques !

COMPARISON WITH FLUKA AND MCNPX

Few words of caution

- Compare the existing FLUKA and MCNPX functionalities with the existing and planned Geant4 ones:
  - Ie : comparison is not exactly fair with FLUKA and MCNPX
- Need to say the above to be fair ;)

FLUKA / Geant4 biasing functionalities

Biasing options in FLUKA from http://www.fluka.org/content/manuals/fluka2011.manual	**Options in Geant4,** Present, new, or future
Leading particle biasing for electrons and photons: region dependent, below user-defined energy threshold and for selected physical effects. ?	Done.
Russian Roulette and splitting at boundary crossing based on region relative importance.	Existing and re-done with generic scheme.
Region-dependent multiplicity tuning in high energy nuclear interactions.	Can be done and more general.
Region-dependent biased downscattering and non-analogue absorption of low-energy neutrons.	Can be done.
Biased decay length for increased daughter production.	Done.
Biased inelastic nuclear interaction length.	Done.
Biased interaction lengths for electron and photon electromagnetic interactions.	Done and validated for gamma.
Biased angular distribution of decay secondary particles.	Can de done.
Region-dependent weight window in three energy ranges (and energy group dependent for low energy neutrons).	Existing. Can be "re-provided" and more general.
Bias setting according to a user-defined logics.	(need more info in FLUKA, but is actual purpose of this dev.)
User-defined neutrino direction biasing.	Can be done, easily.
User-defined step by step importance biasing.	Can be done, easily.

Simple for neutral. More difficult for charged but understood.

MCNPX / Geant4 biasing functionalities

Biasing options in MCNPX From LA-UR-03-1987, MCNP5 manual	**Options in Geant4,** Present, new, or future
Energy Cutoff & Time Cutoff	Existing (not considered as biasing)
Geometry Splitting with Russian Roulette	Existing and "re-provided" with generic scheme.
Energy Splitting/Roulette and Time Splitting/Roulette	Can be done easily with generic scheme.
Weight Cutoff	Existing (in some way). Easy in generic scheme.
Weight Window	Existing and will be "re-provided" with new design. Can be made more general.
Exponential Transform	Done (with generic scheme).
Implicit Capture (or "Implicit capture," "survival biasing," and "absorption by weight reduction")	Can be done. Planned for this year.
Forced Collisions	Done (with generic scheme).
Source Variable Biasing	Existing (GPS).
Point Detector Tally (?)	(not biasing ?)
DXTRAN	Planned, need more work. Doable.
Correlated Sampling	Not planned for now "à la MCNP". But doable with user's invest.