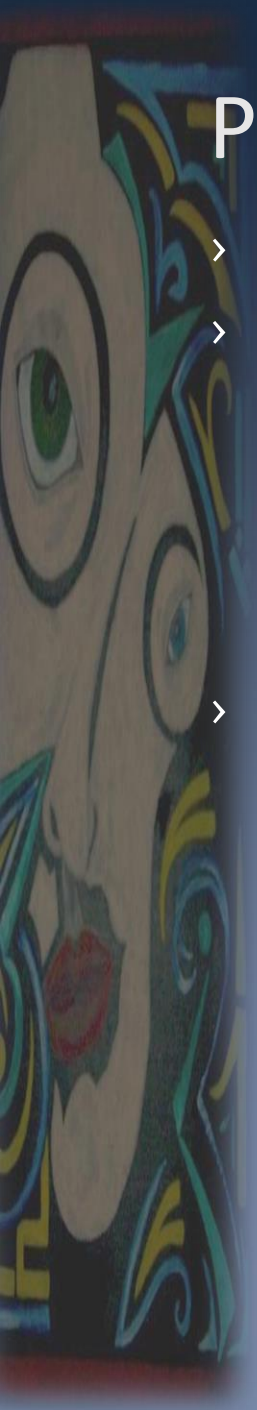
Status of Generic Biasing

Parallel – 8B

Marc Verderi
LLR/Ecole polytechnique
Wollongong Collaboration Meeting
September 2017

Released in 10.3



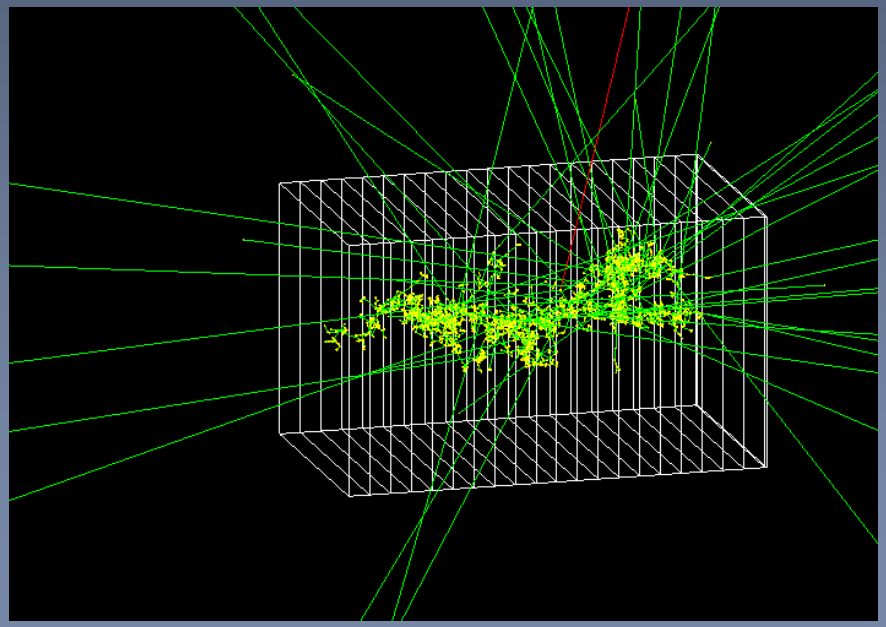
Parallel Worlds (1/2):

- Generic Biasing scheme extended to allow for parallel geometries in 10.3
- Navigation and interface to geometries provided by:
  - `G4ParallelGeometriesLimiterProcess`:
    - new process to limits the step on the boundaries of the parallel geometries;
    - one instance handles all parallel geometries the generic biasing has to be aware of;
  - `G4BiasingProcessInterface`:
    - The process which makes the interface between the tracking and the biasing classes;
    - Extended to check for biasing operator in mass and parallel geometries;
- And facility classes:
  - `G4BiasingProcessSharedData`:
    - information shared among biasing processes;
    - extended to carry information on the limiter process, if any;
  - `G4GenericBiasingPhysics`:
    - physics constructor, a helper class to configure physics list for activating the biasing;
    - extended for adding a `G4ParallelGeometriesLimiterProcess` instance:

```
FTFP_BERT* physicsList = new FTFP_BERT;
G4GenericBiasingPhysics* biasingPhysics = new G4GenericBiasingPhysics();
biasingPhysics->Bias("neutron");
biasingPhysics->AddParallelGeometry("neutron","parallelWorld1");
biasingPhysics->AddParallelGeometry("neutron","parallelWorld2");
physicsList->RegisterPhysics(biasingPhysics);
```

Parallel Worlds (2/2):

- Example `extended/biasing/GB06`:
  - Illustrates usage of parallel geometry with a classical shield problem
  - i.e. a geometry-based importance splitting
- Geometry:
  - Mass geometry : a single block of concrete
  - Parallel world : define the slices
    - Importance of slices being a function of their copy number



Incident neutron in concrete block with biasing activated. Slices on this figure are in the parallel geometry

Example GB05: Splitting by cross-section

- Generic biasing designed to allow invention of techniques/user's plugin
  - Many information provided to user's classes
  - Opportunities provided to modify physic process behavior and/or to split/kill tracks.
- Purpose of example `GB05` is to illustrate this with an invented (?) technique
  - « Splitting by cross-section » : mix of "physics-based" and splitting/killing technique
  - Supposed to be an invention
- Principle of the « Splitting by cross-section » :
  - Geometry-based importance biasing has to chose slice thicknesses so that:
    - There is enough splitting so that the flux does not decay in the shield
    - Not too much splitting to avoid a divergence in the (unweighted) flux
  - ☞ In this technique the splitting rate follows the one of the disappearance by physics
  - A biasing operation is introduced so that the `G4BiasingProcessInterace` process
    - Competes with other processes in the GPIL race
      - With a « cross-section » value which is the physical absorption cross-section one
      - Eg : for neutrons, this is « Decay + nCapture + neutronInelastic »
    - Has a `PostStepDoIt` that splits the track (by 2)
  - Technique is applied to tracks moving forward
    - Others are killed by Russian roulette
- Example shows the technical aspect
- Actual performances need to be studied !

Code snapshots

- Decision taking on biasing to apply:
  - Here, a decision at the beginning of the step, in the GPIL race
  - Decision taken by a « biasing operator »
  - Which decides of a « biasing operation » to be applied and sets it up

```
G4VBiasingOperation* GB05BOptrSplitAndKillByCrossSection::
ProposeNonPhysicsBiasingOperation(const G4Track* track,
                                  const G4BiasingProcessInterface* )
{
    …
    G4double totalCrossSection(0.0);
    for ( size_t i = 0 ; i < fProcesses.size() ; i++ ) {
        G4double interactionLength = fProcesses[i]->GetCurrentInteractionLength();
        if ( interactionLength < DBL_MAX/10. )
            totalCrossSection += 1./interactionLength;
    }
    if ( totalCrossSection < DBL_MIN ) return nullptr;
    G4double totalInteractionLength = 1./totalCrossSection;
    fSplitAndKillByCrossSection->SetInteractionLength( totalInteractionLength );
    return fSplitAndKillByCrossSection;
}
```

Code snapshots

- Decision taking on biasing to apply:
  - Here, a decision at the beginning of the step, in the GPIL race
  - Decision taken by a « biasing operator »
  - Which decides of a « biasing operation » to be applied and sets it up

```
G4VBiasingOperation* GB05BOptrSplitAndKillByCrossSection::
ProposeNonPhysicsBiasingOperation(const G4Track* track,
                                  const G4BiasingProcessInterface* )
{
    …
    G4double totalCrossSection(0.0);
    for ( size_t i = 0 ; i < fProcesses.size() ; i++ ) {
        G4double interactionLength = fProcesses[i]->GetCurrentInteractionLength();
        if ( interactionLength < DBL_MAX/10. )
            totalCrossSection += 1./interactionLength;
    }
    if ( totalCrossSection < DBL_MIN ) return nullptr;
    G4double totalInteractionLength = 1./totalCrossSection;
    fSplitAndKillByCrossSection->SetInteractionLength( totalInteractionLength );
    return fSplitAndKillByCrossSection;
}
```

Code snapshots

- These processes in `fProcesses` have been selected at construction time:

```
void GB05DetectorConstruction::ConstructSDandField()
{
…
    GB05BOptrSplitAndKillByCrossSection* biasingOperator =
    new GB05BOptrSplitAndKillByCrossSection("neutron");
    biasingOperator->AddProcessToEquipoise("Decay");
    biasingOperator->AddProcessToEquipoise("nCapture");
    biasingOperator->AddProcessToEquipoise("neutronInelastic");
…
}
```

- And put under biasing control in the main, with biasing physic constructor:

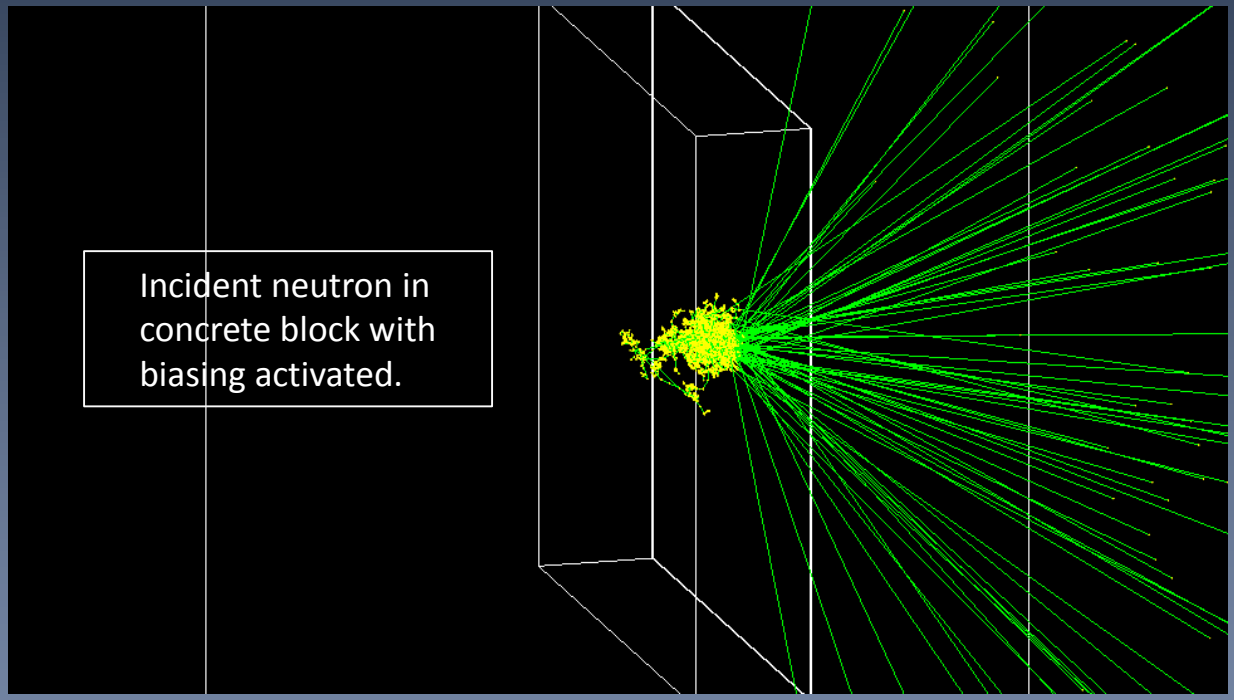
```
    G4GenericBiasingPhysics* biasingPhysics = new G4GenericBiasingPhysics();
    biasingPhysics->Bias("neutron");
    physicsList->RegisterPhysics(biasingPhysics);
```

Makes physics processes wrapped

- This makes in particular processes interaction length updated by the biasing machinery at the beginning of the step (by the first wrapper):
  - Updated physics quantities (eg: cross-sections) hence easily accessible to developer

☞ Offload a lot of internal Geant4 technicalities from the biasing developer !

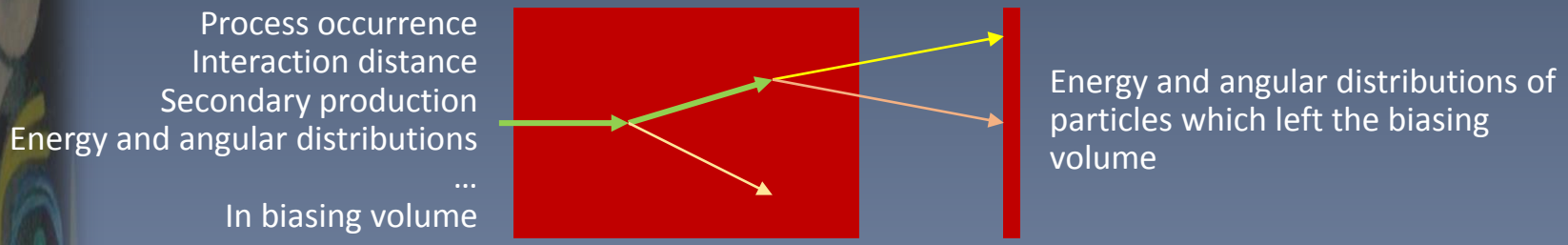
Illustration of GB05



Ongoing

Statistical test suite

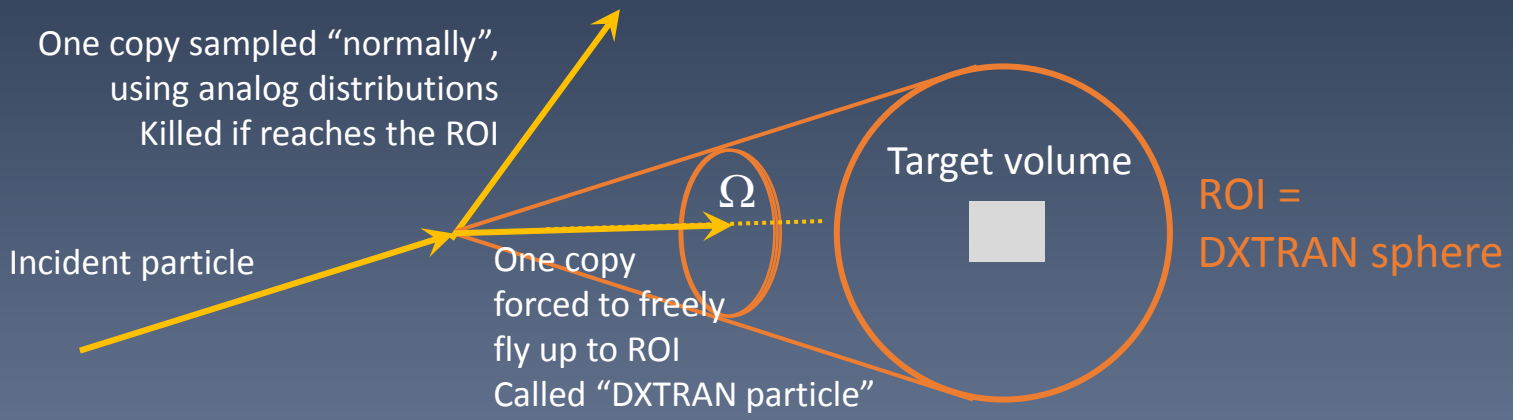
- Aim at verifying statistical correctness of weight application
  - Verifications done with “private” tests up to know
    - but with limited statistics
  - A geant4/tests/testXX would allow to run large statistics
    - And push analog/biasing statistical comparisons
- Sharing between biasing options possible:
  - Many variables are common to the various biasing options



  - Useful for cross-section change, forced collision, and future leading particle biasing, implicit capture.
- At present private development, under test49
  - Several biasing volume possible, choice of biasing option
  - Output through ntuples + root macros for comparisons
    - Help welcome to have this working in the tests !
  - Plan for histogram output when will be better defined

DXTRAN

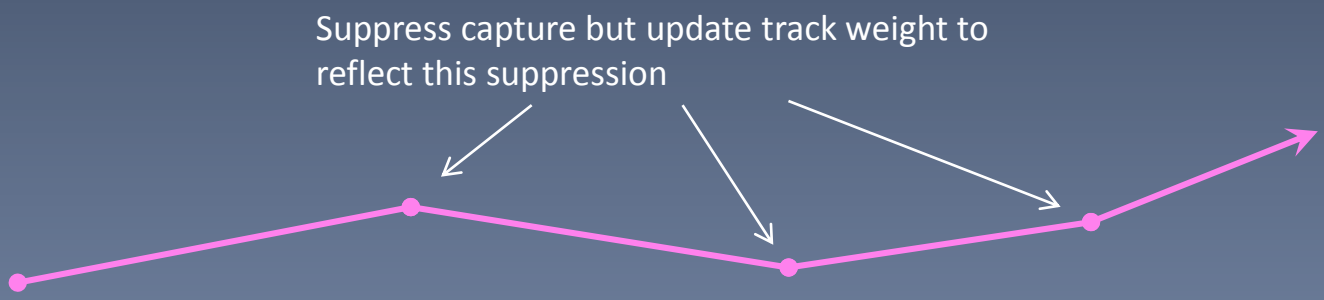
- Option imported from MCNP to scatter particles toward a preferred solid angle
- DXTRAN = stands for deterministic transportation



- Intended for elastic (or quasi-elastic) scattering :
  - In particular neutrons
- Issues:
  - Still to well understand MCNP scheme in weight calculation
  - Weight calculation:
    - Scattering of DXTRAN particle made along a biased angular distribution
    - Weight calculation needs the probability to scatter along this direction in the analog case
- Where to find these analog calculations ? How to avoid dependencies onto other physics packages ?

Implicit Capture

- MCNP option in neutron transport
  - "Implicit capture," "survival biasing," and "absorption by weight reduction" stand for the same technique
- Keep neutrons alive wrt absorption process(es)
  - Makes a same neutron "exploring" more phase space



- Needed ingredients should exist today:
  - Example GB05 makes use of same functionalities than the one needed for implicit capture
  - Namely access cross-section of –say- capture process
- "Just" the matter of finding time to implement the scheme.