Status of Biasing:

- Generic Biasing &
- Reverse Monte Carlo

Marc Verderi
LLR/Ecole polytechnique
Wollongong Collaboration Meeting
September 2017

Generic Biasing

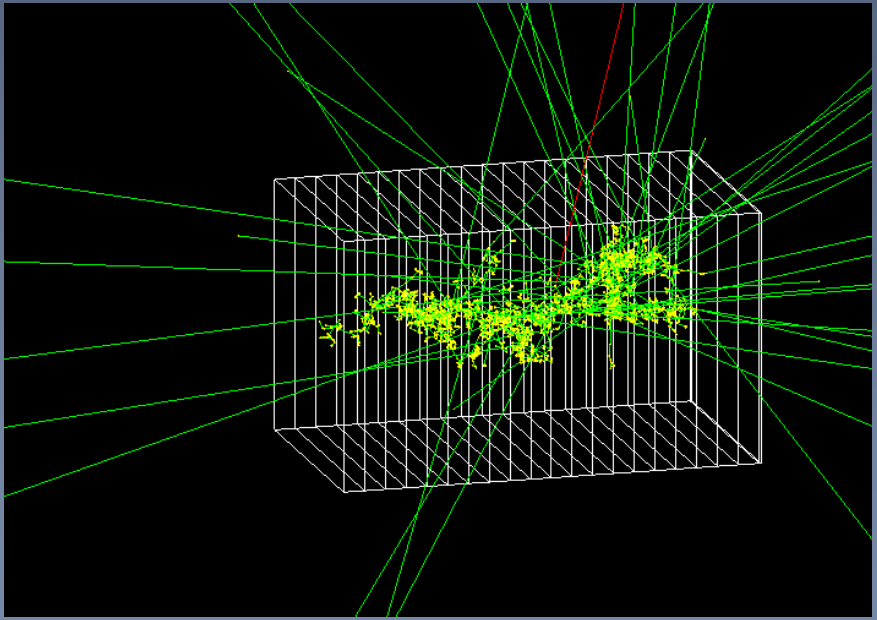
Parallel Worlds (1/2):

- 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:

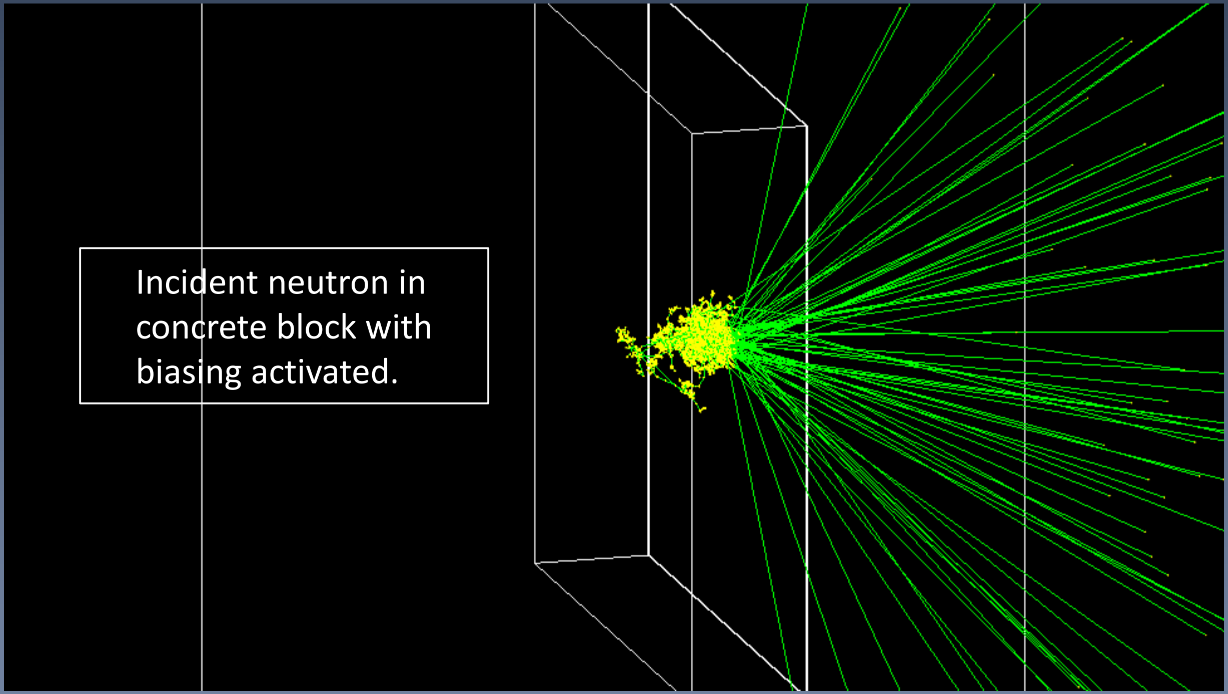
```
G4GenericBiasingPhysics* biasingPhysics = new G4GenericBiasingPhysics();
biasingPhysics->Bias("neutron");
physicsList->RegisterPhysics(biasingPhysics);
```

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:
  - Need for verifying statistical correctness of weight application
    - Verifications done with “private” tests up to know
  - Sharing between biasing options possible:
    - Many variables are common to the various biasing options
    - Procedure is always to compare biasing against analog
  - At present private development, under test49
    - Allows several geometries
    - Cross-section change and forced interaction scheme first tested
- Biasing option developments:
  - Implicit capture
    - Will benefit from easy cross-section access
  - DXTRAN:
    - Will likely use parallel geometry (to define the region of interest)
  - Prototyping of charged particle biasing (difficult !)

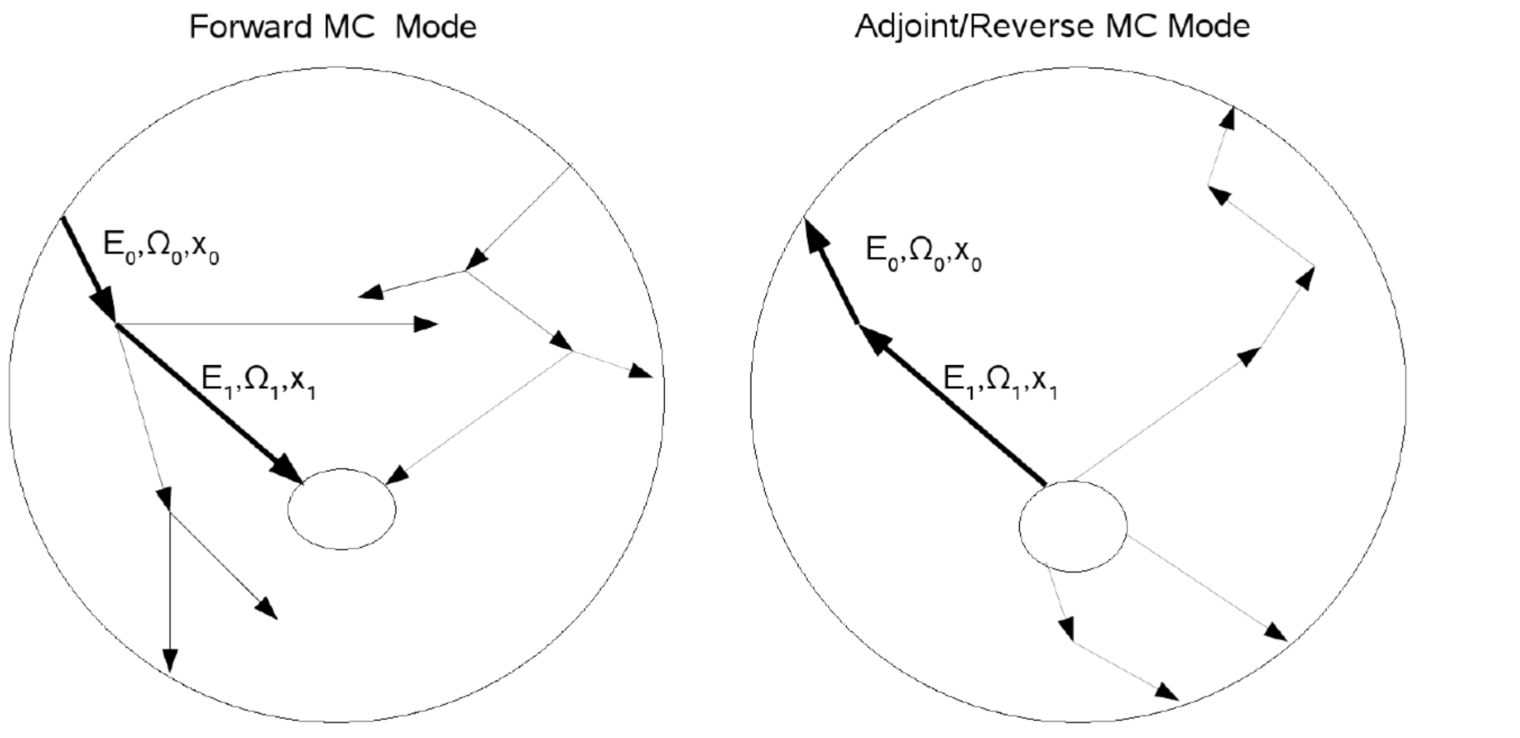
Reverse Monte Carlo

Material from Laurent Desorgher

Centre hospitalier universitaire vaudois (CHUV)

(Text in blue : Marc's addition to Laurent's slides)

Reverse MC method



Reverse MC

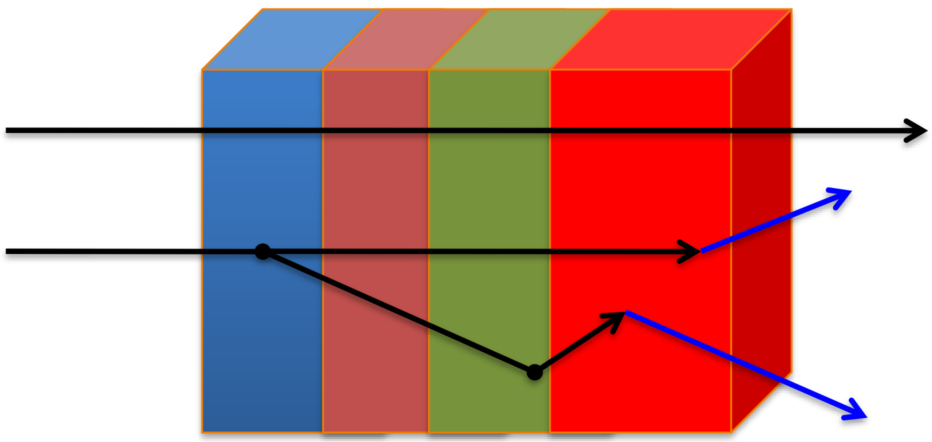
- Start from the tiny sensitive volume
- Reverse tracking until external source with occurence of reverse processes
- Geant4 Reverse Compton, Photoe-electric, bremsstrahlung, ionisation e-,protons

Reverse MC development in GEANT4 since last collaboration meeting

- Implementation of the forced interaction for reverse gamma
- First release and default mode in geant4.10.3

- One issue was convergence difficulty with thick shields:
  - Appearance of large weights that affected the convergence
  - Identified as a “typical” symptom of insufficient sampling of some phase space region(s)
- Laurent worked on forced interactions
  - Looks to solve/improve the problem

Principle of forced interaction for gamma reverse physics

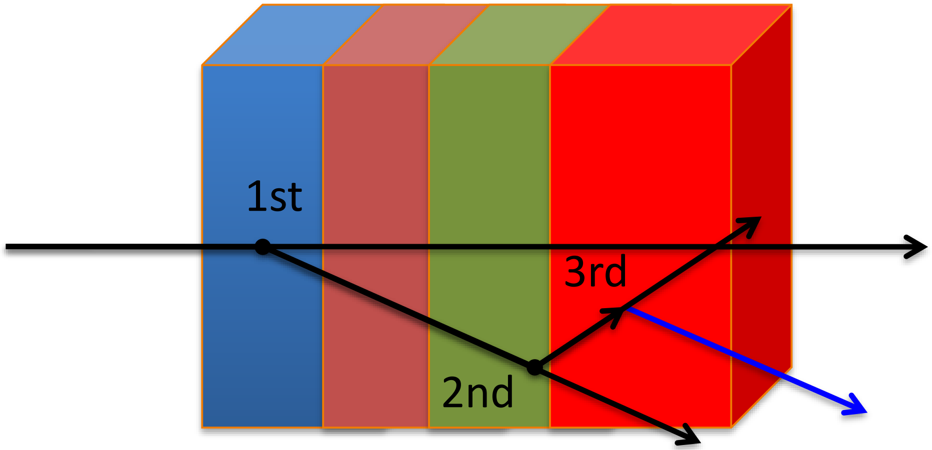


- Most of the time the reverse/forward gamma will not interact in the geometry
- Idea -> Force interactions along the track of the reverse gamma to go back to a primary electron on the other side of the shielding

- One gamma is sent, not interacting
  - But collects the amount of material traversed
- A second gamma if forced to interact in this amount of material

- Scheme is being extended to incident electrons

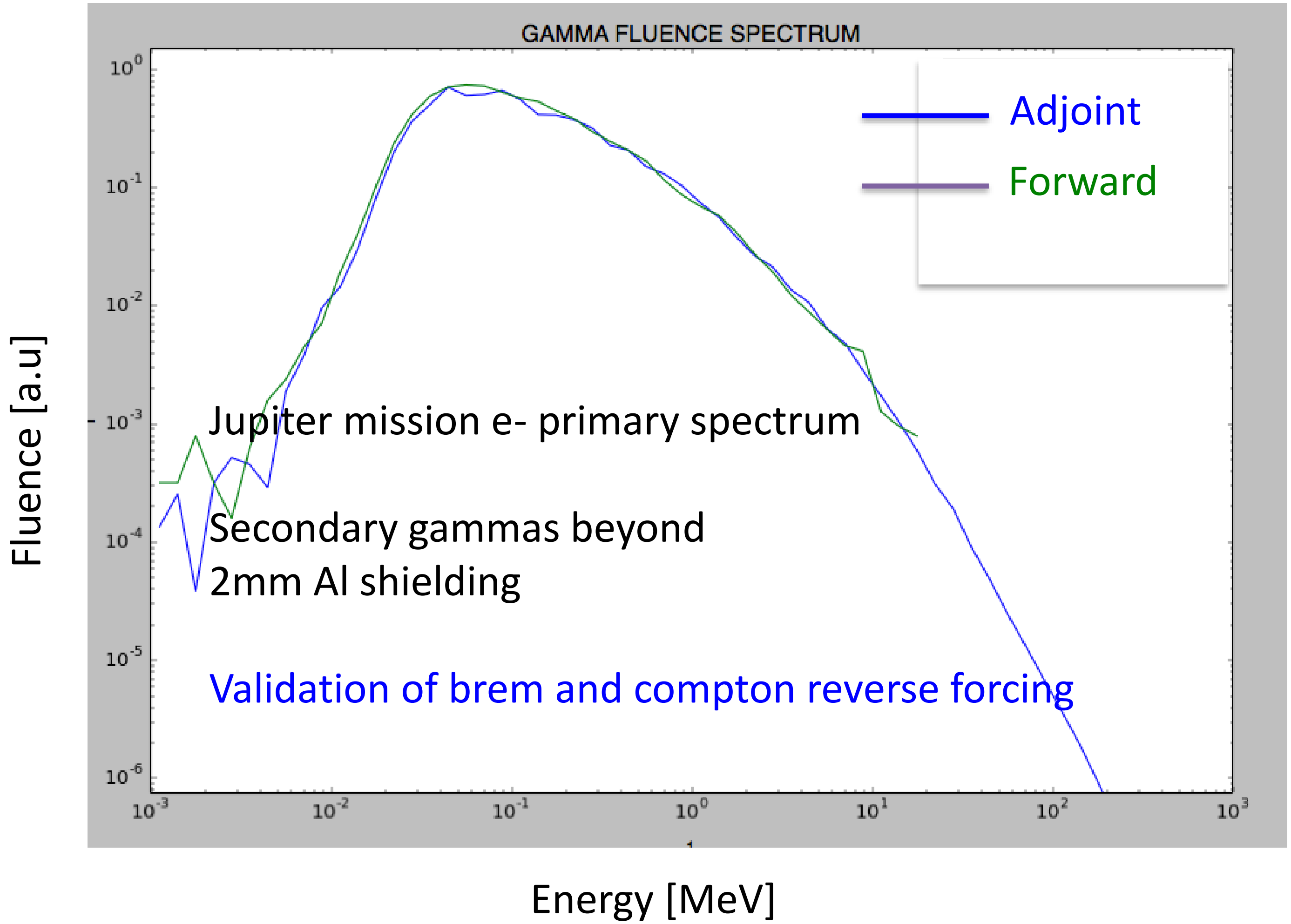
Forced interaction applied to all gammas till an electron is produced



Example

- 1st reverse forced interaction reverse compton
- 2nd reverse forced interaction reverse compton
- 3rd reverse forced interaction reverse brem

Test of forced interaction for gamma

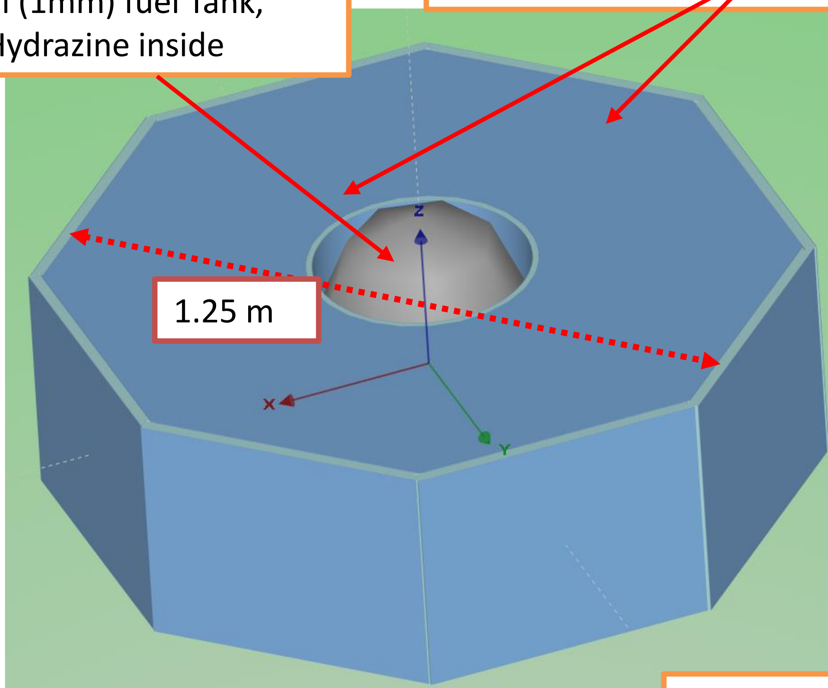


ESA Satellite geometry by M. Vuolo

External structure made of "sandwich" panels:
- 0.6 mm Carbon
- 19 mm Al Honeycomb
- 0.6 mm Carbon

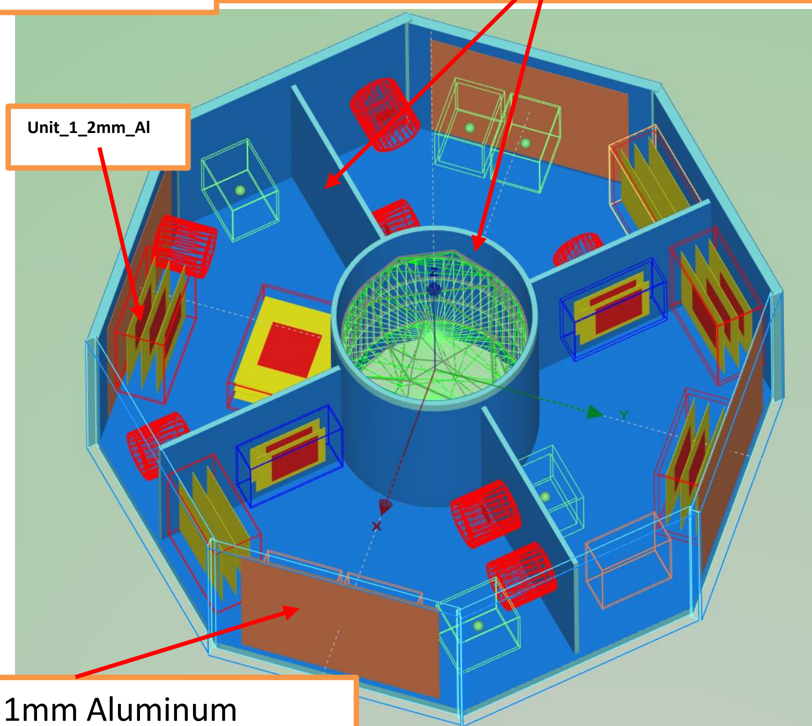
Internal structure made of "sandwich" panels:
- 0.6 mm Carbon
- 10 mm Al Honeycomb
- 0.6 mm Carbon

Ti (1mm) fuel Tank,
Hydrazine inside



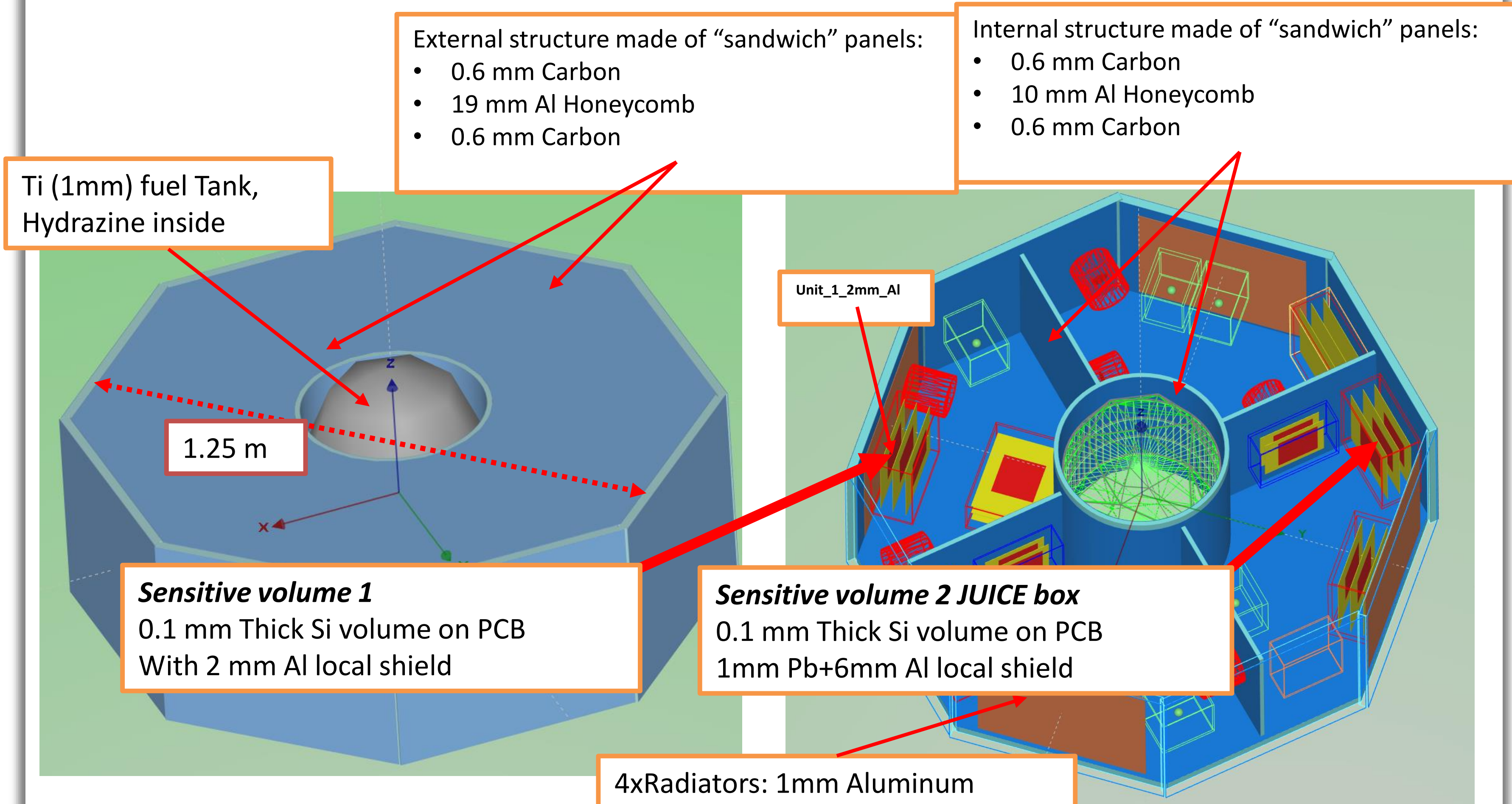
1.25 m

Unit_1_2mm_Al

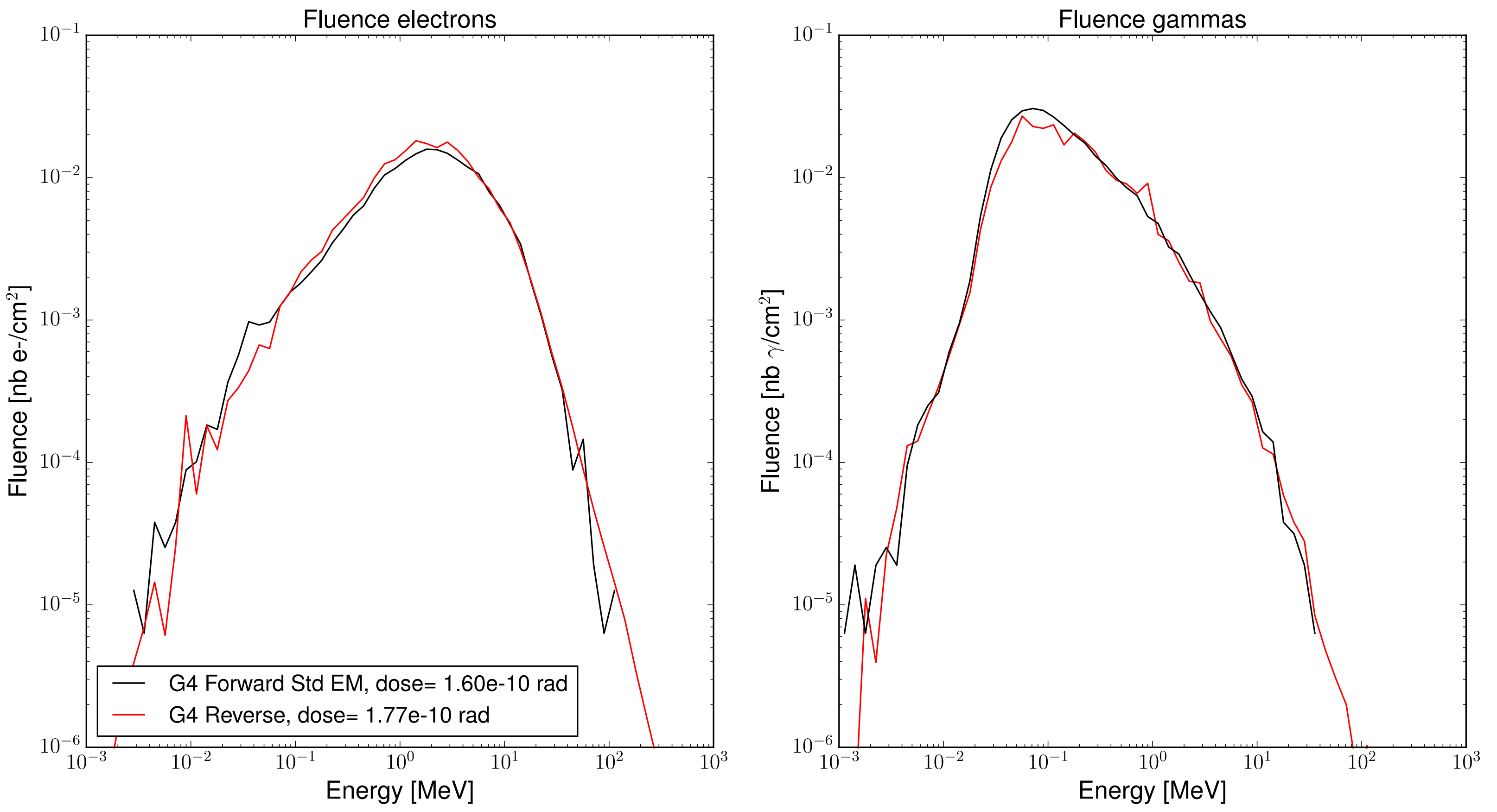


4xRadiators: 1mm Aluminum

ESA Satellite geometry by M. Vuolo

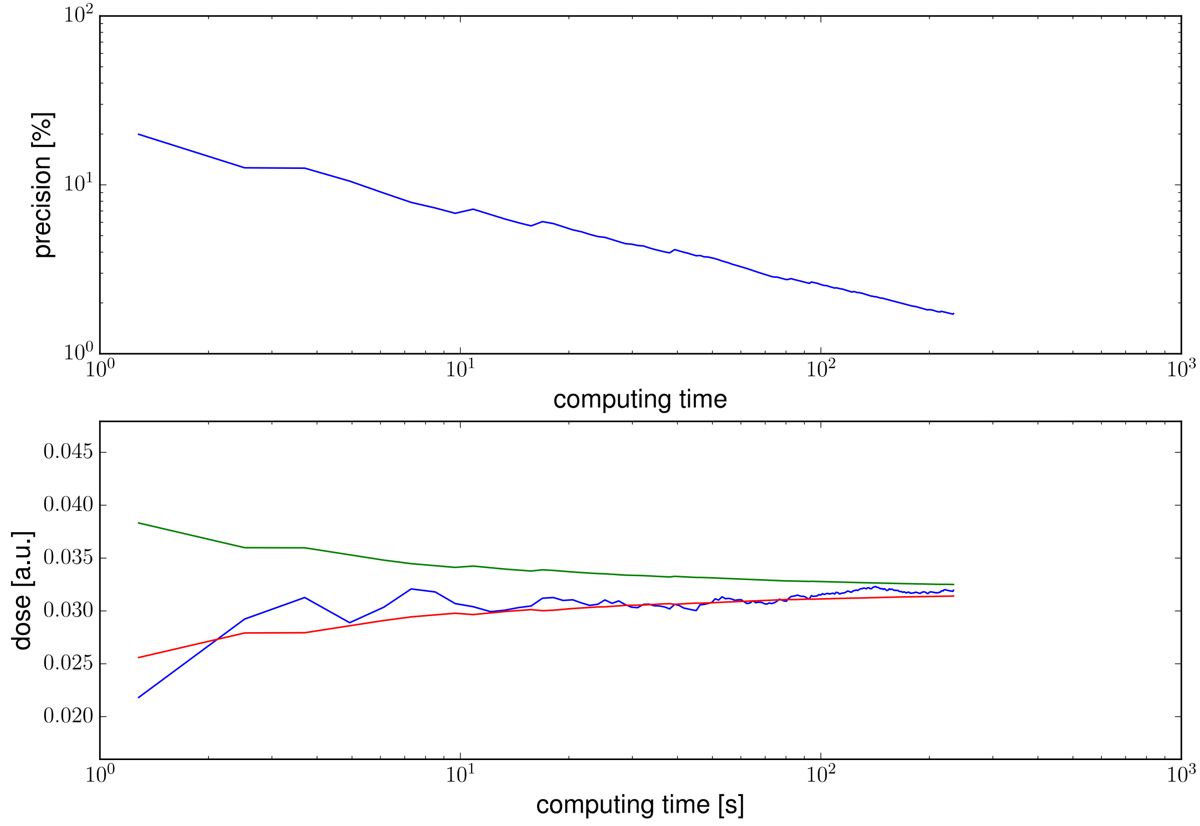


Results Sensitive volume1



Reverse and Forward MC computed agree within 10%

Convergence RMC dose Sensitive volume1



- Sensitive detector 2 shows same level of agreement and convergence

Conclusion

- Generic biasing:
  - Parallel geometry introduced in 10.3
  - And two new examples
    - Parallel geometry (GB06) & « splitting by cross-section » (GB05)
  - Statistical test suite soon available
  - Options ongoing:
    - Implicit capture & DXTRAN
    - Prototyping of charged particle biasing

- Reverse Monte Carlo:
  - Forced interaction for reverse gamma introduced
    - Forced interaction for reverse electron prototyped
  - Look to improve long pending issue of convergence
  - Updated version of RMC is now available in last release of GRAS (4.0)
  - Plan to:
    - Allow ability to do per process forced interaction or free flight for gammas
    - Add importance sampling to further improve convergence

- Manpower : is still an issue