Geant4 Fast Simulation Interface

10th International Geant4 Tutorial in Korea

6-10 November 2023

Dennis Wright

using slides of Alexei Sytov (INFN, KISTI) and

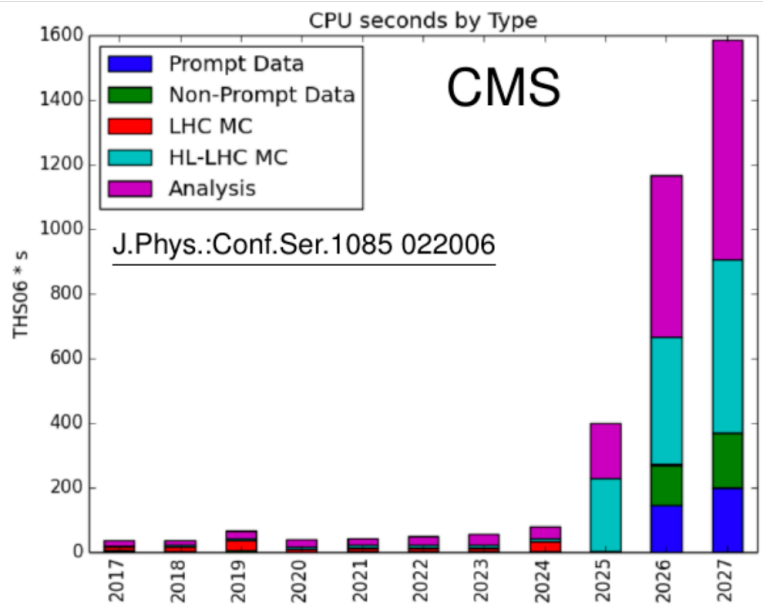
Anna Zaborowska (CERN)

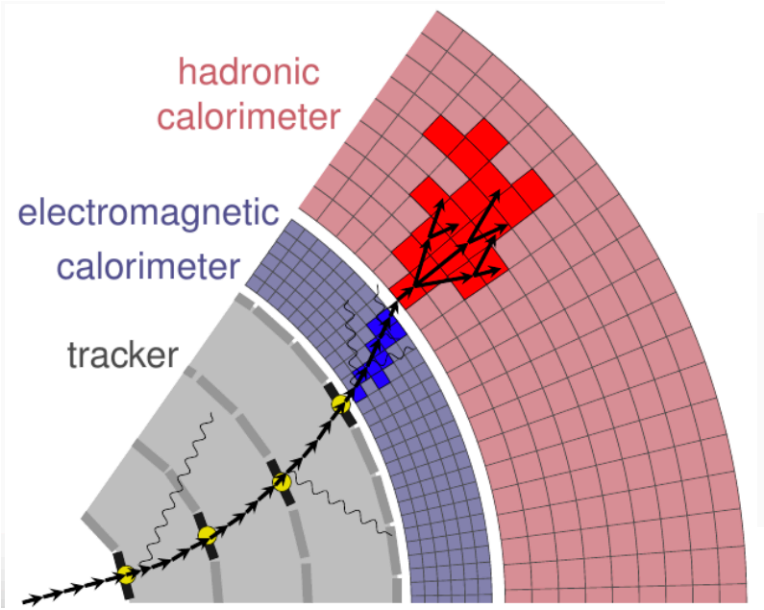
Outline

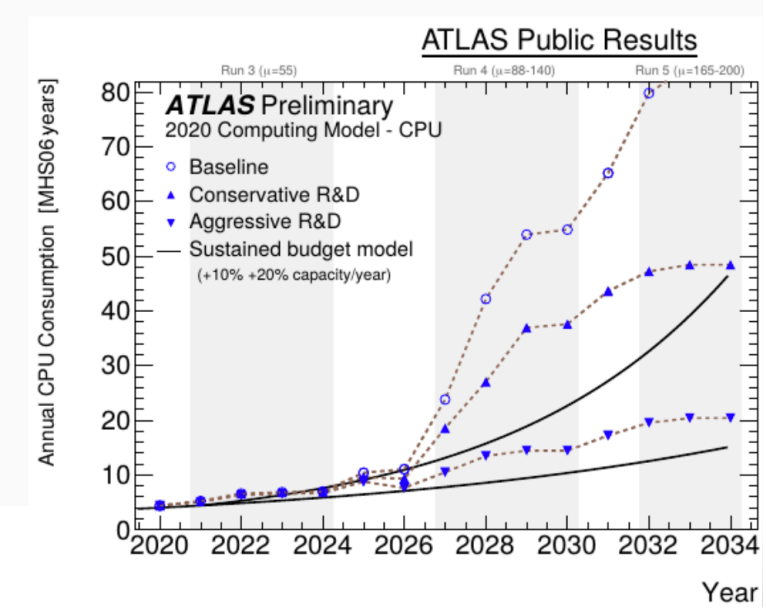
- Why is fast simulation needed?
- What is fast simulation in Geant4?
- Creating a fast simulation model in Geant4
- Geant4 fast simulation process definition and parallel worlds
- Applications
  - Detector simulations
  - Machine learning model
  - Channeling in crystals

Why is Fast Simulation Needed?

- Generate more data in the same CPU time
- More simulation (statistics) and data analysis will be required in future experiments
- Economy of simulation resources
- Limit power consumption







Geant4 Fast Simulation Interface

- To simulate an electromagnetic calorimeter with different physics, need to replace EM physics in a certain volume, under certain conditions and only for certain particle types

- Could G4Biasing be used instead?
  - It is a special class allowing a Geant4 process to be modified during simulation execution - very useful in certain cases
  - But G4Biasing works only for discrete processes, not for multiple scattering, bremsstrahlung or pair production

- At times, need to replace Geant4 processes with external simulation code

What is Fast Simulation in Geant4?

- A trade-off between simulation time and accuracy
- In some regions, detailed situations are not needed
  - replace them by parameterizations - faster approximations to physics processes
- At its first step, fast simulation stops the standard Geant4 processes, then resumes them when it is complete
- Activated only in a particular G4Region under certain conditions for certain particle types

How Does Parameterization Work?

- Where are parameterized processes used?
  - determine which G4Region
- Which particles and processes are parameterized?
  - determined by static conditions such as particle type, PDG data, etc.)
  - also by dynamic conditions (energy, direction, …)
- What happens instead of detailed simulation?
  - decide where particle is to be moved
  - determine which secondaries are created
  - Is the primary particle killed or not?
  - How much energy is deposited and where?

Define G4Region in DetectorConstruction

- Add to DetectorConstruction::Construct()

```
//my volume
G4Box* MySolid = new G4Box("MyBox",SizeX/2,SizeY/2,SizeZ/2.);
G4LogicalVolume* MyVolumeLogic = new G4LogicalVolume(MySolid,MyMaterial,"MyVolume");
new G4PVPlacement(Rotation,Position,MyVolumeLogic,"MyVolume",logicWorld,false,0);

//my region (necessary for the FastSim model)
fRegion = new G4Region("MyRegion");
fRegion->AddRootLogicalVolume(MyVolumeLogic);
```

- Add to DetectorConstruction::ConstructSDandField()

```
void DetectorConstruction::ConstructSDandField()
{
    // --------------- fast simulation ------------------------
    //extract the region of the crystal from the store
    G4RegionStore* regionStore = G4RegionStore::GetInstance();
    G4Region* MyRegion = regionStore->GetRegion("MyRegion");

    //create the channeling model for this region
    MyFastSimModel* MyModel = new MyFastSimModel("ChannelingModel",MyRegion);

    //some options of the model...
```

Create Your Own Fast Simulation Model

- MyFastSimModel.hh

```
class MyFastSimModel : public G4VFastSimulationModel
{
public:
  //-------------------------
  // Constructor, destructor
  //-------------------------
  MyFastSimModel (G4String, G4Region*);
  MyFastSimModel (G4String);
  ~MyFastSimModel ();

  //-----------------------------
  // Virtual methods of the base
  // class to be coded by the user
  //-----------------------------

  // -- IsApplicable
  virtual G4bool IsApplicable(const G4ParticleDefinition&);
  // -- ModelTrigger
  virtual G4bool ModelTrigger(const G4FastTrack &);
  // -- User method DoIt
  virtual void DoIt(const G4FastTrack&, G4FastStep&);
```

Create Your Own Fast Simulation Model

- MyFastSimModel.cc

```
MyFastSimModel::MyFastSimModel(G4String modelName, G4Region* envelope)
: G4VFastSimulationModel(modelName, envelope)
{
}

//....oooOOOOOooo........oooOOOOOooo........ooo      )oo........oooOOOOOooo

MyFastSimModel::MyFastSimModel(G4String modelName)
: G4VFastSimulationModel(modelName)
{
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo

MyFastSimModel::~MyFastSimModel()
{
}
```

MyFastSimModel.cc : Particles and Conditions

```
G4bool MyFastSimModel::IsApplicable(const G4ParticleDefinition& particleType)
{
  return
    &particleType == G4Electron::ElectronDefinition() ||
    &particleType == G4Positron::PositronDefinition() ||
    &particleType == G4Gamma::GammaDefinition();
    //& my particles ...
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo...

G4bool MyFastSimModel::ModelTrigger(const G4FastTrack& fastTrack)
{
  //my code:
  //...
  return MyCondition;
}
```

MyFastSimModel.cc : Model Implementation

```
void MyFastSimModel::DoIt(const G4FastTrack& fastTrack,
                          G4FastStep& fastStep)
{

  //get some necessary information
  G4double Etotal = fastTrack.GetPrimaryTrack()->GetTotalEnergy();
  G4double mass = fastTrack.GetPrimaryTrack()->GetParticleDefinition()->GetPDGMass();
  G4double charge = fastTrack.GetPrimaryTrack()->GetParticleDefinition()->GetPDGCharge();
  G4ThreeVector MomentumDirection=fastTrack.GetPrimaryTrackLocalDirection();
  G4ThreeVector xyz = fastTrack.GetPrimaryTrackLocalPosition();
  G4double TGlobal = fastTrack.GetPrimaryTrack()->GetGlobalTime();
  //fastTrack.Get...

  //do very important simulations
  //my code ...

  //set new parameters:

  //set global time
  fastStep.ProposePrimaryTrackFinalTime(TGlobal);
  //set final position
  fastStep.ProposePrimaryTrackFinalPosition(xyz);
  //set final kinetic energy
  fastStep.ProposePrimaryTrackFinalKineticEnergy(Etotal-
           fastTrack.GetPrimaryTrack()->GetParticleDefinition()->GetPDGMass());
  //set final momentum direction
  fastStep.ProposePrimaryTrackFinalMomentumDirection(MomentumDirection);

  //kill a primary particle if necessary
  fastStep.KillPrimaryTrack();
}
```

MyFastSimModel.cc : Secondary Particle Production

```
void MyFastSimModel::DoIt(const G4FastTrack& fastTrack,
                          G4FastStep& fastStep)
{
    //some code ...

    //there is a default but it is better to do:
    fastStep.SetNumberOfSecondaryTracks(MaxParticlesProducedPerStep);

    //particle declaration
    const G4DynamicParticle theGamma =
            G4DynamicParticle(G4Gamma::Gamma(),PhotonMomentumDirection,Ephoton);

    //generation of a secondary photon
    fastStep.CreateSecondaryTrack(theGamma,PhotonCoordinateXYZ,PhotonGlobalTime,true);
}
```

MyFastSimModel.cc : Register Fast Sim Process

- Add to physics list:

```
G4FastSimulationPhysics* fastSimulationPhysics = new G4FastSimulationPhysics();
fastSimulationPhysics->BeVerbose();
// -- activation of fast simulation for particles having fast simulation models
// -- attached in the mass geometry:
fastSimulationPhysics->ActivateFastSimulation("e-");
fastSimulationPhysics->ActivateFastSimulation("e+");
// -- Attach the fast simulation physics constructor to the physics list:
physicsList->RegisterPhysics( fastSimulationPhysics );
```

- Important:
  - If any condition of the model is not fulfilled (isApplicable, ModelTrigger), the standard Geant4 process will be active as normal
  - If there are several fast simulation models, the first model whose conditions are fulfilled will be activated

Parallel Worlds for Different Particle Types

- For mass and parallel geometry
  - examples/extended/parameterisations/Par01/examplePar01.cc

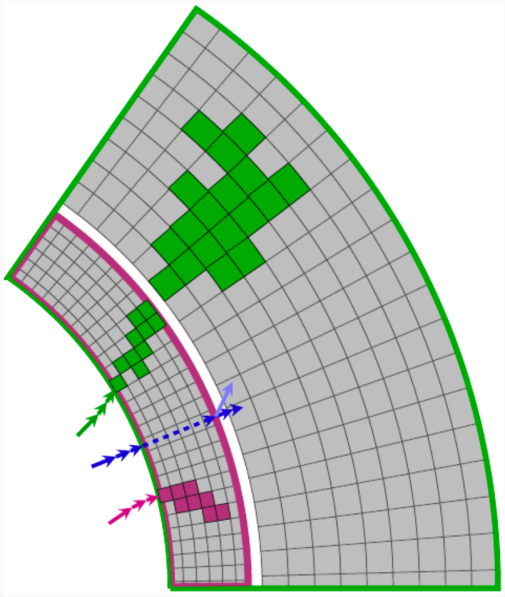
```
FTFP_BERT* physicsList = new FTFP_BERT; // G4VModularPhysicsList
G4FastSimulationPhysics* fastSimulationPhysics = new G4FastSimulationPhysics(); // helper
fastSimulationPhysics->BeVerbose();
// - activation of fast simulation for particles having fast simulation models attached
↪  in the mass geometry:
fastSimulationPhysics->ActivateFastSimulation("e-");
fastSimulationPhysics->ActivateFastSimulation("e+");
fastSimulationPhysics->ActivateFastSimulation("gamma");
// - activation of fast simulation for particles having fast simulation models attached
↪  in the parallel geometry:
fastSimulationPhysics->ActivateFastSimulation("pi+","pionGhostWorld");
fastSimulationPhysics->ActivateFastSimulation("pi-","pionGhostWorld");
physicsList->RegisterPhysics( fastSimulationPhysics ); // attach to the physics list
```

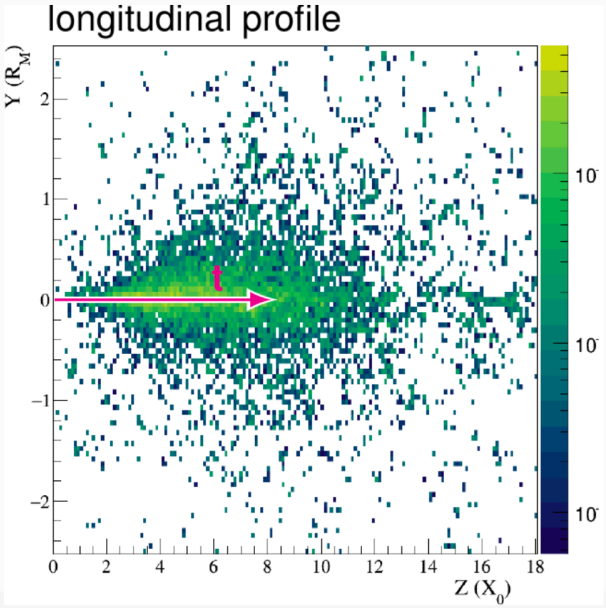
- For parallel geometry
  - examples/extended/parameterisations/Par01/Par01ParallelWorldForPion.cc

```
  G4Region* ghostRegion = new G4Region("GhostCalorimeterRegion");
// ghostLogical is a G4LogicalVolume in parallel geometry, a box made of air encompassing
↪  both EM&H calorimeters
ghostRegion->AddRootLogicalVolume(ghostLogical);
```

Applications

- Simulation of electromagnetic showers in matter
  - EM calorimeters
- Simulation of sampling calorimeters
- Machine learning
- Implementation of external codes in Geant4

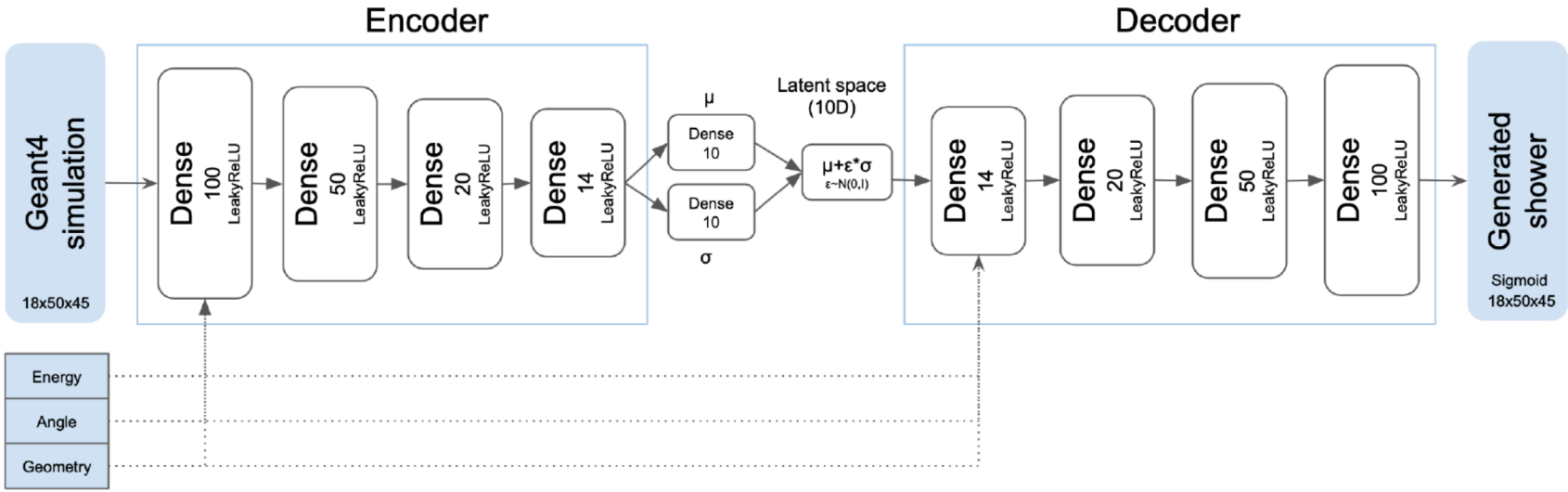




Applications - Examples

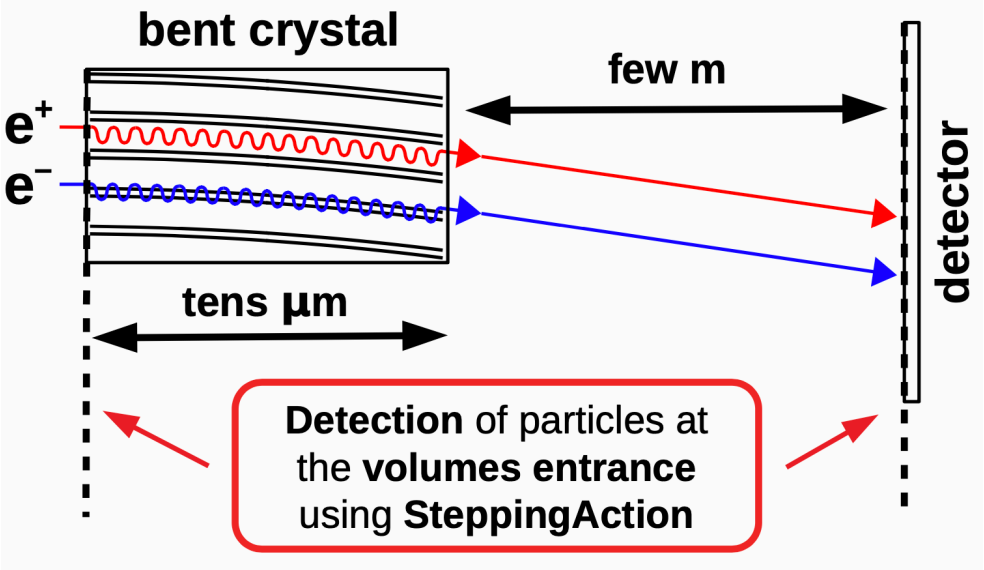
- In examples/extended/parameterisations
  - /Par01/src
    - Par01EMShowerModel.cc (crude e^-, e^+, γ shower parameterization)
    - Par01PionShowerModel.cc (crude π^+, π^- shower model in ghost volume)
    - Par01PiModel.cc (shows how a parameterization can create secondaries)
  - /Par02/src
    - Par02FastSimModelEMCal.cc (e^-, e^+, γ in EM calorimeter using energy smearing)
    - Par02FastSimModelHCal (hadrons in hadronic calorimeter using energy smearing)
    - Par02FastSimModelTracker.cc
  - /Par03/src
    - Par03EMShowerModel.cc (creates multiple energy deposits)
  - /Par04/src
    - Par04MLFastSimModel.cc (uses machine learning to create multiple energy deposits)
  - /gflash
    - GFlashShowerModel (uses GFLASH EM parameterization library)

Machine Learning



- Variational auto-encoder: one of the best ways to randomly generate a distribution using initial parameters
  - such as energy, angle of track, geometry, …
- Fast simulation model can upload neural network parameters for inference using
  - Lightweight Trained Neural Network or
  - Open Neural Network Exchange (ONNX) libraries

Channeling in Crystals



- Can use fast simulation to speed up transport of e^- and e^+
- Geant4 simulation developed based on 855 MeV electrons from Mainz Mikrotron on ultra-short crystal
- Multithreaded version of this has run on NURION@KISTI supercomputer

Summary

- Fast Simulation Interface was created to replace standard Geant4 processes during code execution to speed up simulations
- FastSimulation completely stops the standard Geant4 processes, uses the FastSimulation model instead and, when it is finished, resumes the standard Geant4 processes
- It is activated only in a particular G4Region under certain conditions and for certain particles
- Many applications for fast simulation
  - homogeneous and sampling calorimeters
  - Electromagnetic shower
  - Machine learning and more
- Fast Simulation Interface is the simplest way to implement external code into Geant4 processes