Korea Institute of
Science and Technology Information

Geant4 Fast Simulation Interface

Dr. Alexei Sytov

9th International Geant4 Tutorial in Korea 2022

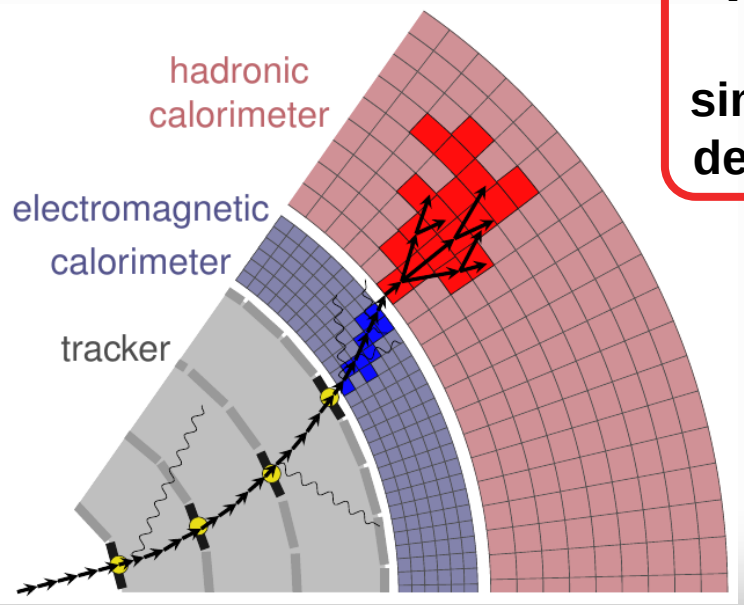
Daejeon, 2022/12/05

Outline

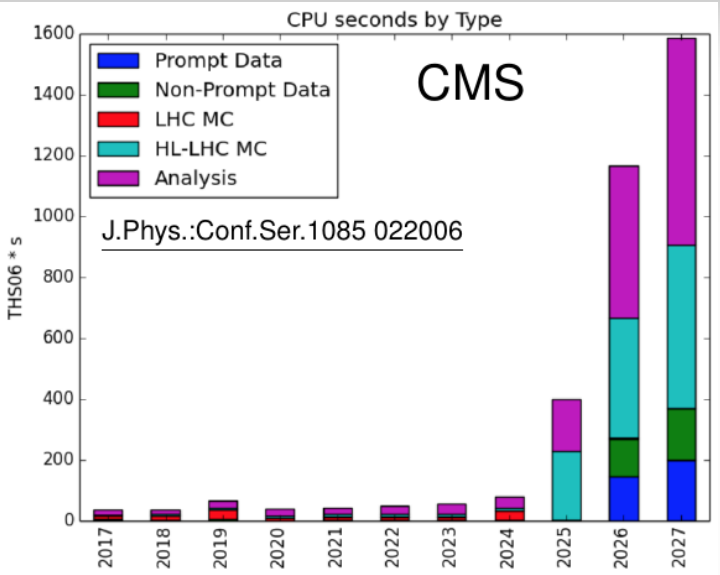
- **Why do we need fast simulations?**
- **What is fast simulation in Geant4?**
- **How to create Geant4 Fast Simulation Model?**
  - Which?
  - What?
  - Where?
- **Geant4 Fast Simulation Process definition and parallel worlds**
- **Applications**
  - Detector simulations
  - Machine Learning model
  - Channeling in crystals

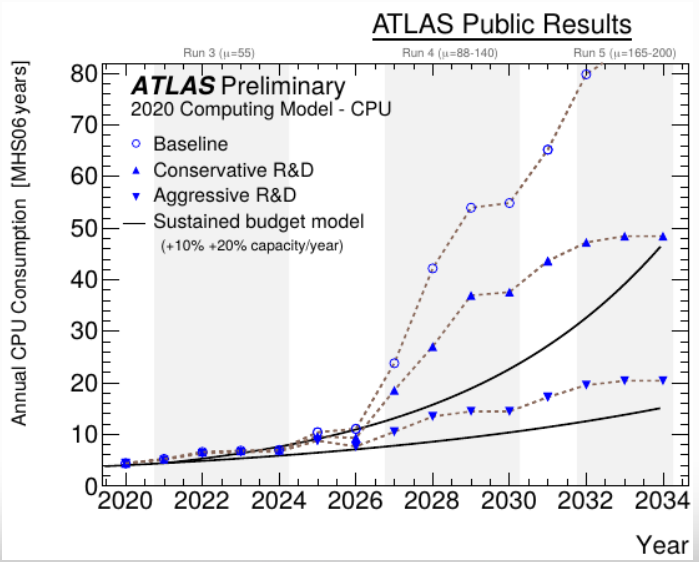
Why we need to simulate fast?

- To speed up simulations in order to generate more data within the same CPU time
- **More simulations** and data analysis will be **required** in the **future experiments**
- Economy of simulation resources
- Economy of electricity



We need new models to simulate certain detectors faster





From Anna Zaborowska presentation

Why do we need any special interface in Geant4?

What about G4Biasing?

- **G4Biasing** is a special class allowing one to **modify Geant4 processes** during simulation execution

In certain cases extremely useful, **but:**

- **G4Biasing works only** for **discrete processes**, it **does not work** for multiple scattering, bremsstrahlung, pair production (**continuous discrete processes**)

To simulate **electromagnetic calorimeter** with a different physics we need **to replace electromagnetic physics** in a **certain volume**, at **certain condition** and for **certain particles**

Sometimes we need to **replace** the **Geant4 processes** by an **external simulation code**

Geant4 Fast Simulation Interface

What we do mean by Fast Simulation in Geant4?

- **Fast simulation** **is not** a simulation that '**magically**' produces results faster.
- **Fast simulation** is a **trade-off** between simulation **time** and **accuracy**.
- In **some regions** we do **not need** too **detailed simulations** => we can replace them by faster simulation processes, so-called **parameterisation**
- **Fast simulation** completely **stops** the **standard Geant4 processes** at the step of Fast Simulation model and then resumes them
- Is activated **only** in a **certain G4Region** at a **certain condition** and only for **certain particles**

Fast Simulation Interface: where? which? what?

- **Where the particles are parameterised, in which region?**
- **Which particles are parameterised?**
  - static conditions (particle type, PDG, charge, . . .)
  - dynamic conditions (energy, direction, . . .)
- **What happens instead of the detailed simulation:**
  - where the particle is moved?
  - what are the created secondaries?
  - is the primary particle killed?
  - what (and where) energy is deposited?

Where: in a G4Region defined 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);
```

Volume declaration (completely standard)

G4Region declaration

Add Logic Volume to G4Region declaration

- **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...
}
```

Get G4Region

Declare your FastSim model

How to 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&);
```

Inheritance of **G4VFastSimulationModel**

Functions to compile:

How to create your own Fast Simulation Model?

MyFastSimModel.cc

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

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

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

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

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

Constructor

Destructor

How to create your own Fast Simulation Model: which?

MyFastSimModel.cc

The model is applicable for these particles:

```
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;
}
```

The model is activated at this condition

How to create your own Fast Simulation Model: what?

MyFastSimModel.cc

```
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();
}
```

Write your code

Get parameters

Propose new parameters

You may kill the particle

Secondary particle production

MyFastSimModel.cc

```
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);
}
```

The last step: Fast Simulation Process registration

Register FastSimulationPhysics

- **Add to main:**

```
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 );
```

That's it. Enjoy! :)

Important:

- If any **condition** of the model is **not fulfilled** (IsApplicable, ModelTrigger), **standard Geant4** processes will be **active** just as usual
- If there are **several Fast Simulation models**, the **first** model in the list will be **activated** for which the conditions are fulfilled

Parallel worlds for different types of particles

- **Add to main:**

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
```

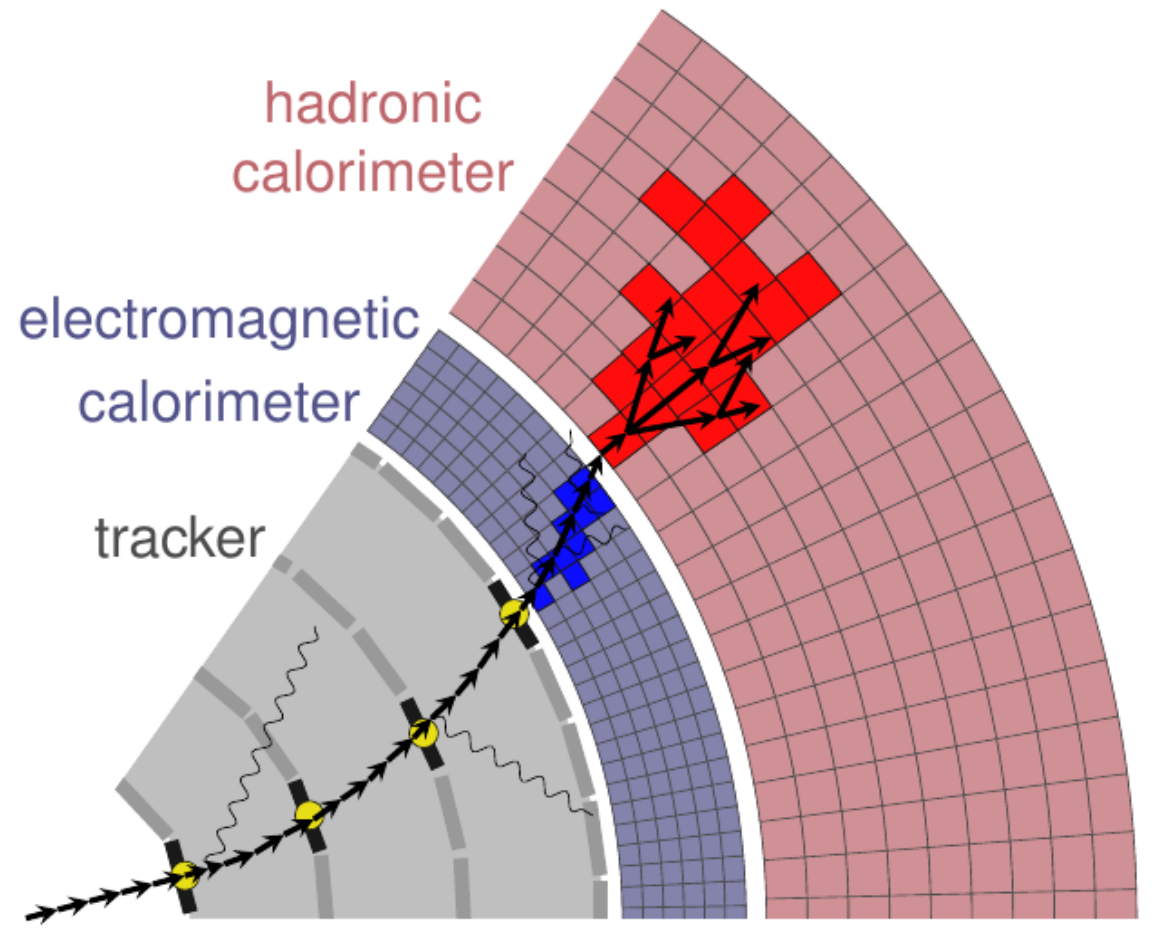
- **Add to DetectorConstruction**

for parallel geometry:

examples/extended/parameterisations/Par01/src/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



From Anna Zaborowska presentation

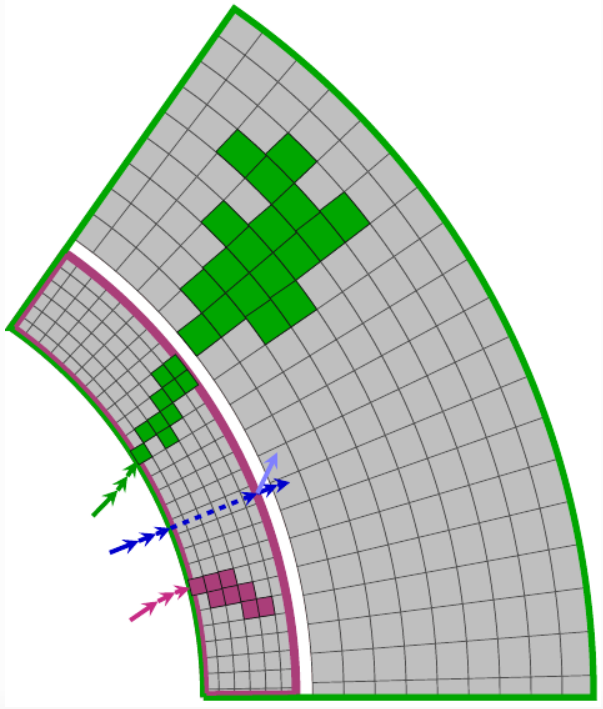
Applications

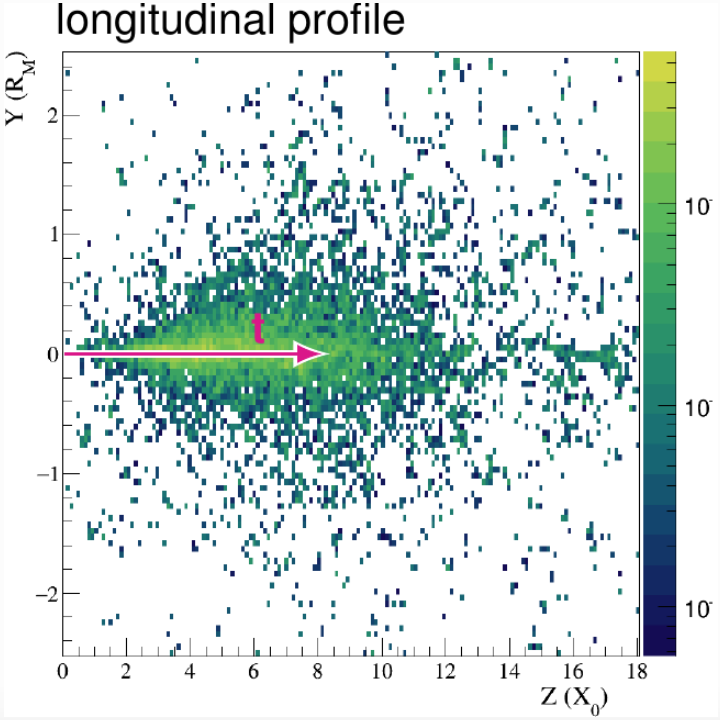
Existing examples: examples/extended/parameterisations/

- **examples/extended/parameterisations/Par01/src/**
  - Par01EMShowerModel.cc
  - Par01PionShowerModel.cc
  - Par01PiModel.cc
- **examples/extended/parameterisations/Par02/src/**
  - Par02FastSimModelEMCal.cc
  - Par02FastSimModelHCal.cc
  - Par02FastSimModelTracker.cc
- **examples/extended/parameterisations/Par03/src/**
  - Par03EMShowerModel.cc
- **examples/extended/parameterisations/Par04/src/**
  - Par04MLFastSimModel.cc
- **GFlashShowerModel**

Applications

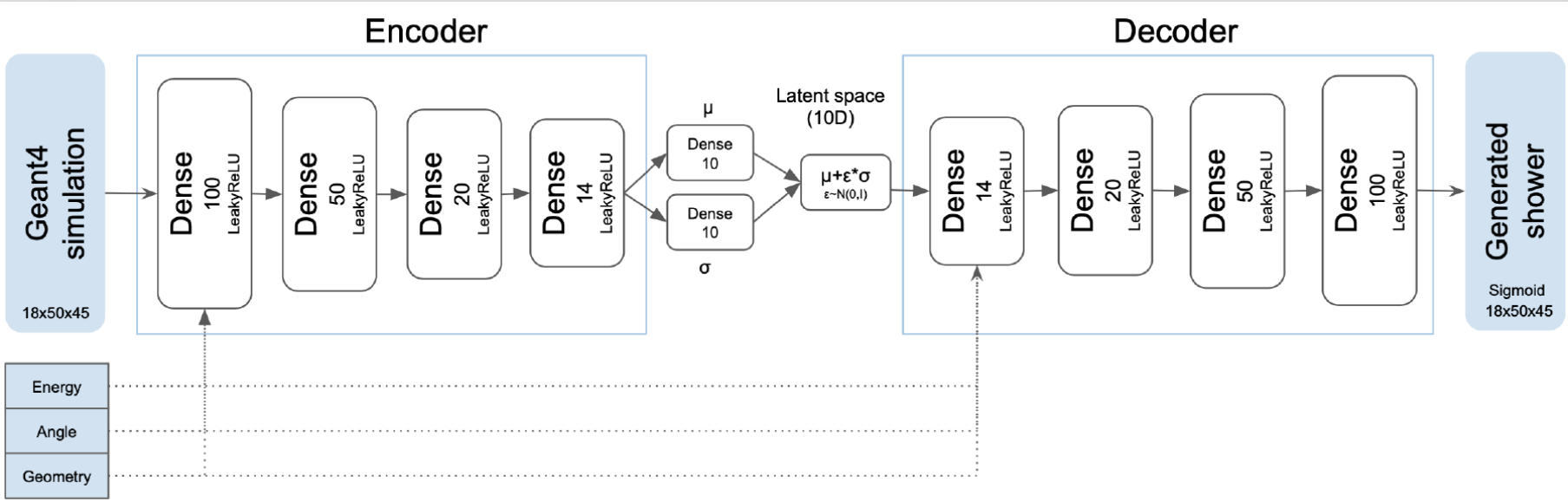
- Simulation of **electromagnetic showers** in matter (**e.m. calorimeters**, …)
- Simulation of **sampling calorimeters**
- **Machine Learning**
- Implementation of **external codes** into Geant4





From Anna Zaborowska presentation,
examples/extended/parameterisations/Par01

Machine Learning model

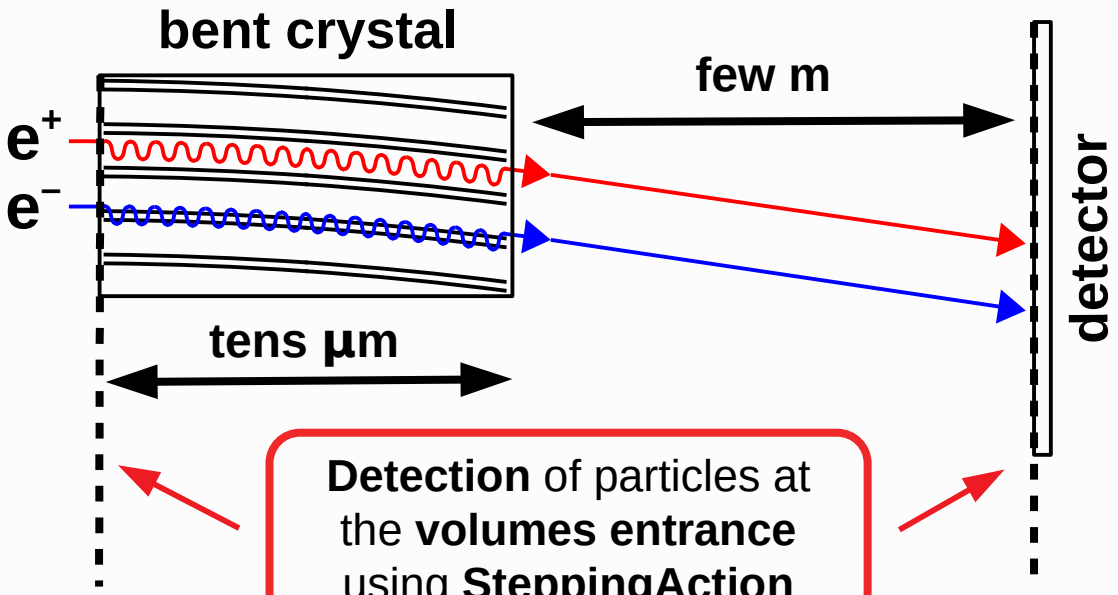


Variational autoencoder is one of the best way to randomly **generate** a distribution using initial parameters

Fast Simulation Model can upload the neural network parameters for **inference** using **Lightweight Trained Neural Network** or **Open Neural Network Exchange libraries**

Channeling model: my next presentation

- **Inspired by** our experiments* of **855 MeV electron** beam deflection by an ultrashort **bent crystal** at **Mainz Mikrotron MAMI**



Detection of particles at the **volumes entrance** using **SteppingAction**

Beam setup in **run.mac** using **GPS** commands; all the **geometry** in **DetectorConstruction**

Multithreading works! Checked at the supercomputer **NURION@KISTI** (Korea)

Output both in **root** (only primary particles) and in **textfile** (all the particles) format

*A. Mazzolari et al. Phys. Rev. Lett. 112, 135503 (2014)

A. Sytov et al. Eur. Phys. J. C 77, 901 (2017)

Conclusions

- **Fast Simulation Interface** was created to **replace standard Geant4** processes during the code execution to **speed-up** the **simulations**.
- **Fast simulation** completely **stops** the **standard Geant4 processes** at the step of Fast Simulation model and then resumes them.
- It is activated **only** in a **certain G4Region** at a **certain condition** and only for **certain particles**
- It possesses a lot of **applications** such as simulations of a homogeneous and sampling **calorimeters**, **electromagnetic shower**, **Machine Learning** and so on. This provides considerable **speed-up** of **Geant simulations**.
- **Fast Simulation Interface** is the simplest way to implement an **external code** into Geant4.