

The Geant4 particle transport simulation toolkit: differentiable?

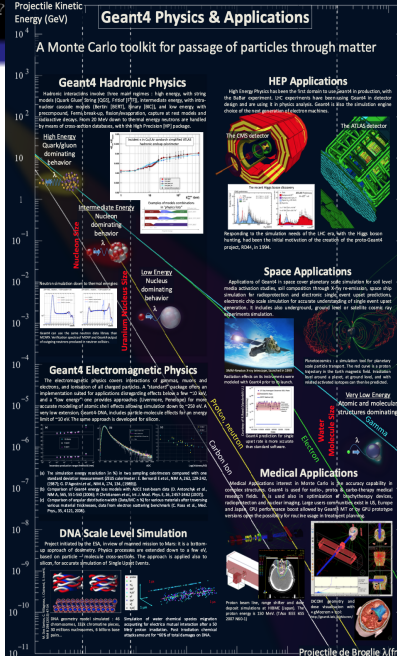
Mihály Novák
(on behalf of the **Geant4** collaboration)

Geant4, CERN-EP-SFT (simulation)



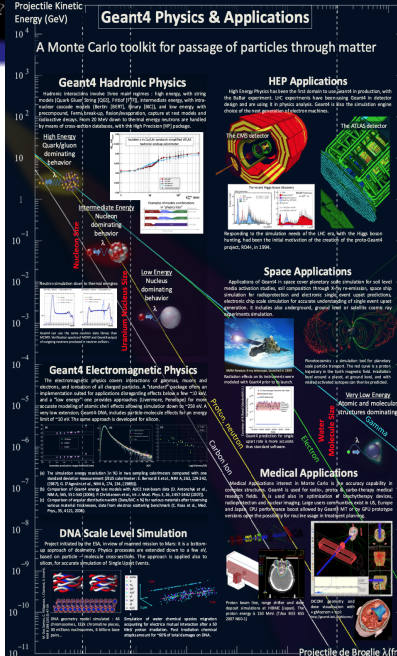
- 1 Geant4: when trying out something new
- 2 What if a differentiable Geant4 is not available?
- 3 How the Geant4 community can help?

- 1 Geant4: when trying out something new
- 2 What if a differentiable Geant4 is not available?
- 3 How the Geant4 community can help?



What is Geant4?

- toolkit for **simulating the interaction of radiation/particles with matter** while passing through complex geometrical setup
- with a robust and **powerful geometry description**
- covering a large set of particles with their rather diverse interactions over a wide energy range
- many different application domains, including high energy, nuclear, medical and bio-medical physics, space engineering, radiation protection, etc.
- all these with a **single simulation kernel** (e.g. one generic stepping loop handles all the different particles and interactions)
- its carefully designed interfaces and abstraction layers provide **high level flexibility**
- **open source**, extensible and **dynamic**: being extended (e.g. new physics interactions, models, etc.) and updated



What is Geant4?

- toolkit for simulating the interaction of radiation/particles with matter while passing through complex geometrical setup
- with a robust and **powerful geometry description**
- covering a large set of particles with their rather diverse interactions over a wide energy range
- many different application domains, including high energy, nuclear, medical and bio-medical physics, space engineering, radiation protection, etc.
- all these with a **single simulation kernel** (e.g. one generic stepping loop handles all the different particles and interactions)
- its carefully designed interfaces and abstraction layers provide **high level flexibility**
- **open source**, extensible and **dynamic**: being extended (e.g. new physics interactions, models, etc.) and updated

All these are excellent!

[illegible]

- All these are excellent! For our users!
But when trying out something new....?

When trying out something new, (at least) the followings need to be resolved:
(not only in case of *Differentiable Programming*, but in general e.g. recently simulations on GPU, FPGA)

When trying out something new, (at least) the followings need to be resolved:

(not only in case of *Differentiable Programming*, but in general e.g. recently simulations on GPU, FPGA)

- Geant4 is a **toolkit**: → not a single simulation application, but rather infinite
 - ⇒ need to **find an "appropriate" simulation application**
 - ⇒ "appropriate": **domain** and possibly even **sub-domain specific**
 - ▶ e.g. 1. HEP, detector simulation: EM shower simulation in a calorimeter
 - ▶ e.g. 2. medical, radiotherapy: accurate, high resolution 3D dose distribution in a phantom

When trying out something new, (at least) the followings need to be resolved:

(not only in case of *Differentiable Programming*, but in general e.g. recently simulations on GPU, FPGA)

- Geant4 is a **toolkit**: → not a single simulation application, but rather infinite
 - ⇒ need to **find an "appropriate" simulation application**
 - ⇒ "appropriate": **domain** and possibly even **sub-domain specific**
 - ▶ e.g. 1. HEP, detector simulation: EM shower simulation in a calorimeter
 - ▶ e.g. 2. medical, radiotherapy: accurate, high resolution 3D dose distribution in a phantom
- Geant4 offers a **large set of particles, interactions** → what is relevant?
 - ⇒ **determined by the selected application** (and the domain and sub-domain)
 - ▶ fixing these collapses the large Geant4 phase space of particles, interactions and models
 - ▶ e.g. in case of 1. above: only e^-/e^+ and γ are relevant with their EM interactions described by models valid at higher energies

When trying out something new, (at least) the followings need to be resolved:

(not only in case of *Differentiable Programming*, but in general e.g. recently simulations on GPU, FPGA)

- Geant4 is a **toolkit**: → not a single simulation application, but rather infinite
 - ⇒ need to **find an "appropriate" simulation application**
 - ⇒ "appropriate": **domain** and possibly even **sub-domain specific**
 - ▶ e.g. 1. HEP, detector simulation: EM shower simulation in a calorimeter
 - ▶ e.g. 2. medical, radiotherapy: accurate, high resolution 3D dose distribution in a phantom
- Geant4 offers a **large set of particles, interactions** → what is relevant?
 - ⇒ **determined by the selected application** (and the domain and sub-domain)
 - ▶ fixing these collapses the large Geant4 phase space of particles, interactions and models
 - ▶ e.g. in case of 1. above: only e^-/e^+ and γ are relevant with their EM interactions described by models valid at higher energies
- Geant4 provides **high level flexibility** through its **interfaces** and **abstraction layers**
 - ⇒ **requires significant expertise, time and effort** to see and understand the details
 - ▶ all that is usually missing; new technology experts more often come from outside of the field
 - ▶ essential details are hidden beneath these abstraction layers

When trying out something new, (at least) the followings need to be resolved:

(not only in case of *Differentiable Programming*, but in general e.g. recently simulations on GPU, FPGA)

- Geant4 is a **toolkit**: → not a single simulation application, but rather infinite
 - ⇒ need to **find an "appropriate" simulation application**
 - ⇒ "appropriate": **domain** and possibly even **sub-domain specific**
 - ▶ e.g. 1. HEP, detector simulation: EM shower simulation in a calorimeter
 - ▶ e.g. 2. medical, radiotherapy: accurate, high resolution 3D dose distribution in a phantom
- Geant4 offers a **large set of particles, interactions** → what is relevant?
 - ⇒ **determined by the selected application** (and the domain and sub-domain)
 - ▶ fixing these collapses the large Geant4 phase space of particles, interactions and models
 - ▶ e.g. in case of 1. above: only e^-/e^+ and γ are relevant with their EM interactions described by models valid at higher energies
- Geant4 provides **high level flexibility** through its **interfaces** and **abstraction layers**
 - ⇒ **requires significant expertise, time and effort** to see and understand the details
 - ▶ all that is usually missing; new technology experts more often come from outside of the field
 - ▶ essential details are hidden beneath these abstraction layers
 - ⇒ even investing all this time and effort, the **full generic case** will be seen
 - ▶ often there is no interest for generic solution (but for a specific case/application)
 - ▶ better to start by solving first a specific problem

What would be needed instead is a standalone, simple, compact, small version of at least **a representative part** of the complete generic problem

What would be needed instead is a **standalone, simple, compact, small** version of at least **a representative part** of the complete generic problem

- representative:

- ▶ the **algorithmic properties**, applied **techniques well represent** those, that one would face with in **the generic, complete solution**
- ▶ covers **only part** of the phase-space **of the generic solution**, but that is actually **an essential part** across many different applications and domains

What would be needed instead is a **standalone, simple, compact, small** version of at least **a representative part** of the complete generic problem

- representative:

- ▶ the **algorithmic properties**, applied **techniques well represent** those, that one would face with in **the generic, complete solution**
- ▶ covers **only part** of the phase-space **of the generic solution**, but that is actually **an essential part** across many different applications and domains

- simple, compact, small:

- ▶ all **for enhancing clarity** through a smaller scale version of the problem
- ▶ implementation: without interfaces, layers of abstractions, deep call stacks, etc.
- ▶ all what is needed for solving the small scale problem and nothing more

What would be needed instead is a **standalone, simple, compact, small** version of at least **a representative part** of the complete generic problem

- representative:

- ▶ the **algorithmic properties**, applied **techniques well represent** those, that one would face with in **the generic, complete solution**
- ▶ covers **only part** of the phase-space of **the generic solution**, but that is actually an **essential part** across many different applications and domains

- simple, compact, small:

- ▶ all **for enhancing clarity** through a smaller scale version of the problem
- ▶ implementation: without interfaces, layers of abstractions, deep call stacks, etc.
- ▶ all what is needed for solving the small scale problem and nothing more

- standalone:

- ▶ should be implemented without external (library) dependencies
- ▶ all components of the simulation (geometry, physics, stepping loop, etc.) should be available locally in a single application
- ▶ would provide **full control over the components** and freedom to modify any of them

What would be needed instead is a **standalone, simple, compact, small** version of at least **a representative part** of the complete generic problem

- representative:

- ▶ the **algorithmic properties**, applied **techniques well represent** those, that one would face with in **the generic, complete solution**
- ▶ covers **only part** of the phase-space of **the generic solution**, but that is actually an **essential part** across many different applications and domains

- simple, compact, small:

- ▶ all **for enhancing clarity** through a smaller scale version of the problem
- ▶ implementation: without interfaces, layers of abstractions, deep call stacks, etc.
- ▶ all what is needed for solving the small scale problem and nothing more

- standalone:

- ▶ should be implemented without external (library) dependencies
- ▶ all components of the simulation (geometry, physics, stepping loop, etc.) should be available locally in a single application
- ▶ would provide **full control over the components** and freedom to modify any of them

Would allow the DP community to work on a differentiable "Geant4".

We will get back to this but before ...

- 1 Geant4: when trying out something new
- 2 What if a differentiable Geant4 is not available?
- 3 How the Geant4 community can help?

Differentiable Programming and Geant4: other possibilities ?

Differentiable Programming and Geant4: other possibilities ?

- numerical calculation (estimate) of the gradient: might be a solution in some cases
 - ▶ but not really feasible when the simulator evaluation is expensive (e.g. complex setup)
 - ▶ or in case of more than a few parameters

Differentiable Programming and Geant4: other possibilities ?

- numerical calculation (estimate) of the gradient: might be a solution in some cases
- approximate the stochastic, non-differentiable simulator with a differentiable surrogate
 - ▶ example: Local-Generative Surrogate Optimisation (L-GSO)¹
 - ▶ iteratively trains and use a differentiable surrogate to approximate the simulator
 1. trains the surrogate in the current point of the phase space (on simulator generated data)
 2. uses the surrogate to estimate the gradient in the local neighbourhood of the current parameter space point
 - ▶ was used to find a more optimal geometry of a multi-stage magnet (described by $\psi \in \mathbb{R}^{42}$ parameters) for active muon shielding in the SHiP experiment

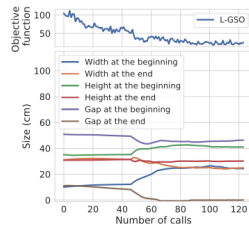
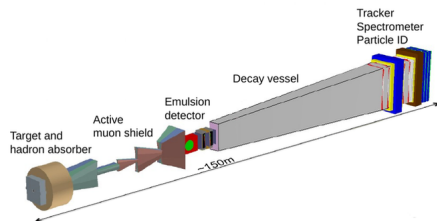


Figure 6: Magnet objective function (top) and six ψ parameters (bottom) dur-

¹ "Black-Box Optimization with Local Generative Surrogates" (S. Shirobokov et al 2020 NeurIPS2020)

Differentiable Programming and Geant4: other possibilities ?

- numerical calculation (estimate) of the gradient: might be a solution in some cases
- approximate the stochastic, non-differentiable simulator with a differentiable surrogate
- direct coupling with the simulator using dedicated Probabilistic Programming framework
 - ▶ example: Etalumis: managed to make Sherpa differentiable²

² "Etalumis: Bringing Probabilistic Programming to Scientific Simulators at Scale— (A.G. Baydin et al. 2019 SC19)

Differentiable Programming and Geant4: other possibilities ?

- numerical calculation (estimate) of the gradient: might be a solution in some cases
- approximate the stochastic, non-differentiable simulator with a differentiable surrogate
- direct coupling with the simulator using dedicated Probabilistic Programming framework

But what if the target is still to **make Geant4**, the stochastic simulator **differentiable**?

- is what we do differentiable at all?
- what are the main obstacles in our algorithms?
- what are the actual benefits if we eliminate those (if possible)?

```
double rndArray[3];
rng->flatArray(3, rndArray);
if (rndArray[0] < qprb) {
    if (rndArray[1] < probb) {
        return 1. + G4HepEmLog(dumEa + rndArray[2]/dumEaa)*thex;
    } else {
        const double var0 = (1.0 - d)*rndArray[2];
        if (var0 < 0.01*d) {
            const double var = var0/(d*dumC1);
            return -1.0 + var*(1.0 - var*0.5*parC)*b1;
        } else {
            return 1.0 + thex*(parC - parXsi - parC*G4HepEmPow(var0 + d, -1./dumC1));
        }
    }
} else {
    return 2.0*rndArray[1] - 1.0;
}
```

```
double greject = 0.;
double eps = 0.;
double rndmv[3];
do {
    rng->flatArray(3, rndmv);
    if (normCond > rndmv[0]) {
        eps = 0.5 - epsRange * G4HepEmX13(rndmv[1]);
        const double delta = deltaFactor/(eps*(1.-eps));
        greject = (ScreenFunction1(delta)-fz)*invF10;
    } else {
        eps = epsMin + epsRange*rndmv[1];
        const double delta = deltaFactor/(eps*(1.-eps));
        greject = (ScreenFunction2(delta)-fz)*invF20;
    }
} while (greject < rndmv[2]);
```

Differentiable Programming and Geant4: other possibilities ?

- numerical calculation (estimate) of the gradient: might be a solution in some cases
- approximate the stochastic, non-differentiable simulator with a differentiable surrogate
- direct coupling with the simulator using dedicated Probabilistic Programming framework

But what if the target is still to **make Geant4**, the stochastic simulator **differentiable**?

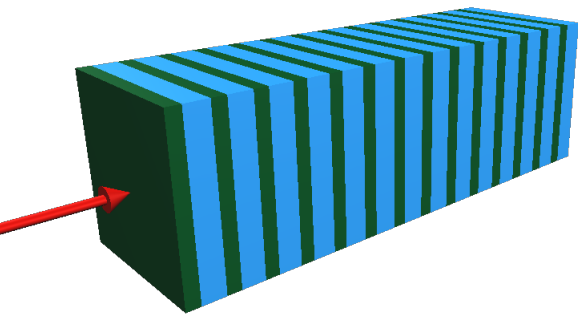
- is what we do differentiable at all?
- what are the main obstacles in our algorithms?
- what are the actual benefits if we eliminate those (if possible)?

We expect the *Differential Programming* community to answer these questions.

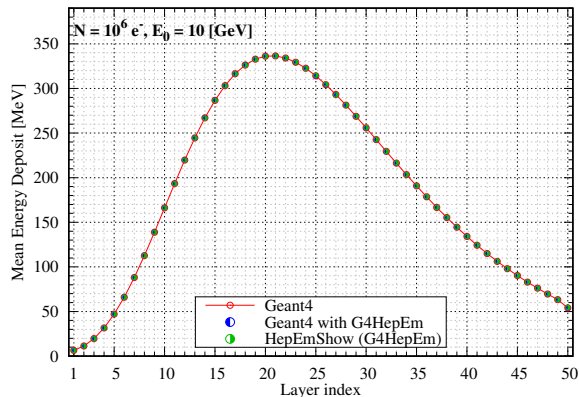
While we can help by providing a suitable starting point: **HepEmShow**

- 1 Geant4: when trying out something new
- 2 What if a differentiable Geant4 is not available?
- 3 How the Geant4 community can help?

HepEmShow: HEP style simulation of EM showers in a configurable simplified sampling calorimeter (a **simple, compact, small** but a **representative part** of the generic problem)



Simplified sampling calorimeter: 50 layers of [2.3 mm PbWO₄ + 5.7 mm lAr]



HepEmShow: HEP style simulation of EM showers in a configurable simplified sampling calorimeter (a **simple, compact, small** but a **representative part** of the generic problem)

- representative: EM shower simulation is at **the core** of the complete detector simulation
 - ▶ **the same simulation** (algorithm, interactions, models, etc.) **of the EM showers** that is used today e.g. **in the ATLAS and CMS detectors**
 - ▶ plays **essential** role in many different application domains beyond HEP detectors as well

HepEmShow: HEP style simulation of EM showers in a configurable simplified sampling calorimeter (a **simple, compact, small** but a **representative part** of the generic problem)

- **representative**: EM shower simulation is at **the core** of the complete detector simulation
 - ▶ **the same simulation** (algorithm, interactions, models, etc.) **of the EM showers** that is used today e.g. **in the ATLAS and CMS detectors**
 - ▶ plays **essential** role in many different application domains beyond HEP detectors as well
- **standalone**: implemented locally and via headers
 - ▶ the **entire physics component is provided** by pulling-in G4HepEm(🔗) headers:
 - a compact implementation of the **Geant4** "*standard*" EM physics (i.e. used in HEP)
 - clear separation of data definition, data initialisation and run-time functionalities
 - ⇒ **run-time functionalities**: stateless, **header only**, **Geant4 independent**
 - moreover, **all data can be dumped** after the initialisation and **re-loaded/re-used**
 - ⇒ **Geant4 style but Geant4 independent EM shower simulation based only on G4HepEm run-time headers and data**
 - see **[the latest presentation on G4HepEm](#)** for more details (or the 🔗 **repository**)

HepEmShow: HEP style simulation of EM showers in a configurable simplified sampling calorimeter (a **simple, compact, small** but a **representative part** of the generic problem)

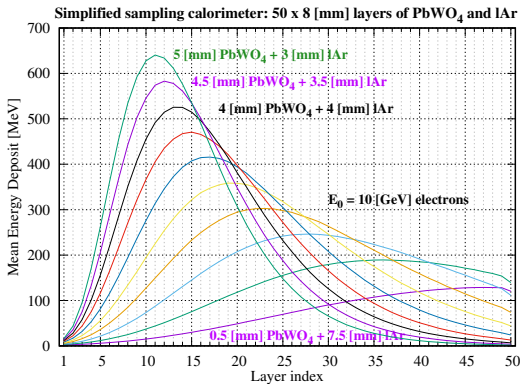
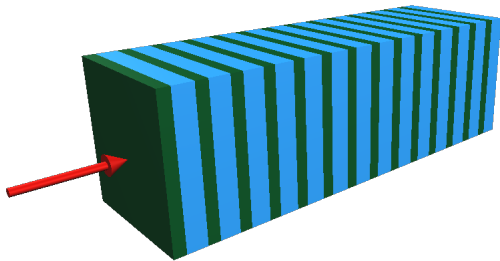
- **representative**: EM shower simulation is at **the core** of the complete detector simulation
 - ▶ **the same simulation** (algorithm, interactions, models, etc.) **of the EM showers** that is used today e.g. **in the ATLAS and CMS detectors**
 - ▶ plays **essential** role in many different application domains beyond HEP detectors as well
- **standalone**: implemented locally and pulling-in headers
 - ▶ the **entire physics** component **is provided** by pulling-in **G4HepEm headers**
 - ▶ **all other** components (geometry, stepping loop, primary generation, random number generator, etc.) are **implemented locally**
 - ▶ **full control over all components** and freedom to change any codes

HepEmShow: HEP style simulation of EM showers in a configurable simplified sampling calorimeter (a **simple, compact, small** but a **representative part** of the generic problem)

- **representative**: EM shower simulation is at **the core** of the complete detector simulation
 - ▶ **the same simulation** (algorithm, interactions, models, etc.) **of the EM showers** that is used today e.g. **in the ATLAS and CMS detectors**
 - ▶ plays **essential** role in many different application domains beyond HEP detectors as well
- **standalone**: implemented locally and pulling-in headers
 - ▶ the **entire physics** component **is provided** by pulling-in **G4HepEm headers**
 - ▶ **all other** components (geometry, stepping loop, primary generation, random number generator, etc.) are **implemented locally**
 - ▶ **full control over all components** and freedom to change any codes
- **simple, compact, small**: everything is **clear, compact and easily understandable**
 - ▶ EM showers so only e^-/e^+ and gamma particles and their EM interactions → small
 - ▶ physics in **G4HepEm** is implemented without abstraction, virtual methods, managers, etc.
 - ▶ more stateless, direct, C-style implementation of the run-time functionalities than C++
 - ▶ all other components (geometry, stepping loop, primary generation, etc.) are also implemented by ensuring only the minimum functionalities required for the application

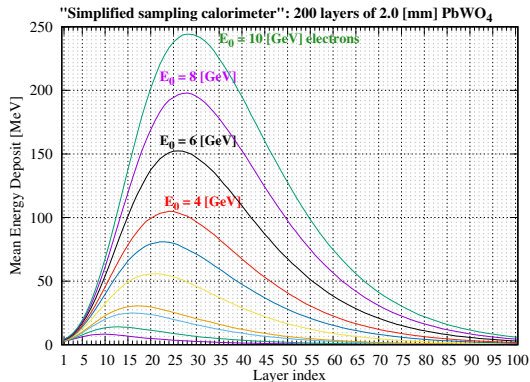
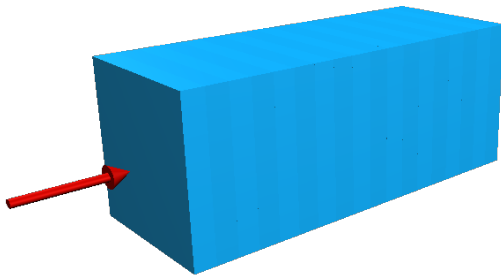
HepEmShow: HEP style simulation of EM showers in a configurable simplified sampling calorimeter

- geometry: *number of layers*, *absorber and/or gap thickness* (gap can be zero), *material*
- primary particles: *particle type*, *kinetic energy*



HepEmShow: HEP style simulation of EM showers in a configurable simplified sampling calorimeter

- geometry: *number of layers*, absorber and/or *gap thickness* (gap can be zero), *material*
- primary particles: *particle type*, *kinetic energy*



HepEmShow: HEP style simulation of EM showers in a configurable simplified sampling calorimeter

- geometry: *number of layers, absorber and/or gap thickness (gap can be zero), material*
- primary particles: *particle type, kinetic energy*

It's rather easy to:

- **use your own random engine** to generate $\xi \in \mathcal{U}(0, 1)$ (often needed)
- **simplify further by disabling complex processes** like e^-/e^+ *multiple Coulomb scattering* or *energy loss fluctuation*
- go even further by **removing simulations of e^-/e^+** , i.e. **only γ particles** interact
- even for γ -s, **introduce your own, simplified interactions** instead of the real ones


HepEmShow: HEP style simulation of EM showers in a configurable simplified sampling calorimeter

- geometry: *number of layers*, *absorber* and/or *gap thickness* (*gap* can be zero), *material*
- primary particles: *particle type*, *kinetic energy*

It's rather easy to:

- **use your own random engine** to generate $\xi \in \mathcal{U}(0, 1)$ (often needed)
- **simplify further by disabling complex processes** like e^-/e^+ *multiple Coulomb scattering* or *energy loss fluctuation*
- go even further by **removing simulations of e^-/e^+** , i.e. **only γ particles** interact
- even for γ -s, **introduce your own, simplified interactions** instead of the real ones

Please note:

- **HepEmShow** is a **result of our very first discussion** during the *MIAPbP workshop on Differentiable Programming for Fundamental Physics* just couple of weeks ago
- this is the **first version**, we will likely have **further iterations, refinements** to ensure that the *Differentiable Programming* community receives what is needed
- **HepEmShow** will be **available soon** (by September) in the  **repository**

And we are here to help Questions? :-)