On Geant4 and differentiable programming

Mihály Novák

28-09-2023

Outline

Contents

A bit of background information:

- there is a relatively new ($\sim$ 2020 $\rightarrow$) ***MODE (Machine-learning Optimized Design of Experiments) collaboration***

  "...is a collaboration of physicists and computer scientists who target the use of differentiable programming in design optimization of detectors for particle physics applications..."

- part of the collaboration organised the ***MIAPbP (Munich Institute for Astro- Particle- and Biophysics) workshop on Differentiable Programming for Fundamental Physics*** at the Technical University of Munich this June

  "...bring together physicists and computer scientists for an extended period of time...The topics of the workshop are still new to fundamental physics."

- special workshop format: one month long, manly for discussions and work, participants are invited and stay at least for 2 weeks
- some of the organisers (*Lukas Heinrich* (TUM), *Michael Kagan* (SLAC), interested to make the entire HEP computing chain, including detector simulation, differentiable) contacted me last year and I was invited to this workshop (they wanted to have someone from the simulation/`Geant4` side)
- at the end we (with *Alberto*) decided not to go...

Some background information:

- then a ***Third MODE Workshop on Differentiable Programming and Experimental Design*** was organised this July at the Princeton University
- our collaboration (more precisely *Alberto*) was contacted by the main organiser and invited us to give a talk on `Geant4` due to the increasing interest in `Geant4` from the DP community side
- we were asked to give a talk *especially focusing on differentiable programming....* whatever it means
- so we had to understand what the DP community actually would like to have from our side and how we can help them
- at the end I went to Munich for 2-3 days to discuss what the DP community would need from us (in order to be able to give a meaningful presentation at the MODE workshop)
- actually what they need is rather understandable and I will get back to this soon but before let me show what they do without a differentiable simulator

Contents

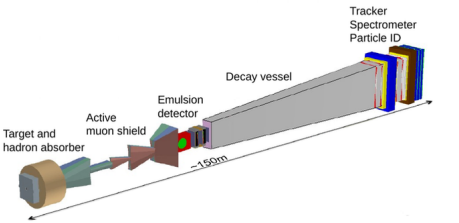
Differentiable Programming and Geant4: **other** possibilities ?

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



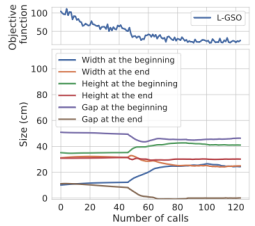


Figure 6: Magnet objective function (top) and six ψ parameters (bottom) dur-

[1] *"Black-Box Optimization with Local Generative Surrogates"* (S. Shirobokov at 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 differentiable[2]

[2] *"Etalumis: Bringing Probabilistic Programming to Scientific Simulators at Scale"* (A.G. Baydin at 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];
rnge->flatArray(3, rndArray);
if (rndArray[0] < qprb) {
  if (rndArray[1] < prob) {
    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 {
  rnge->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 can expect the *Differential Programming* community to answer these questions.

How can we help? What they need?

What the DP community would need is a **standalone**, **simple, compact** version of at least **a representative part** of our complete, generic simulation

What the DP community would need is a **standalone**, **simple, compact** version of at least **a representative part** of our complete, generic simulation

- **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 the DP community would need is a **standalone**, **simple, compact** version of at least **a representative part** of our complete, generic simulation

- **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 the DP community would need is a **standalone**, **simple, compact** version of at least **a representative part** of our complete, generic simulation

- **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

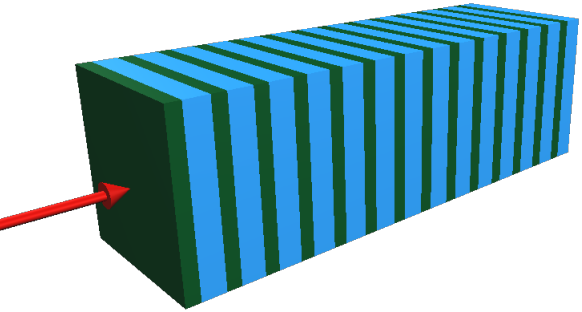
What the DP community would need is a **standalone**, **simple, compact** version of at least **a representative part** of our complete, generic simulation

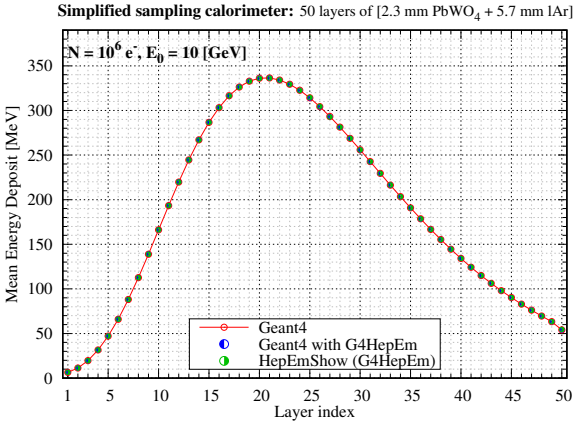
- **representative**:
  - the **algorithmic properties**, applied **techniques well represent** those, that one would face with in **the generic**, complete **solution**
  - covers **only part** of the phase-space **of the generic solution**, **but** that is actually **an essential part** across many 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 test their techniques, experiment and understand the characteristics of our simulation algorithm.

Contents

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)





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
      $\Longrightarrow$ **run-time functionalities**: stateless, **header only, `Geant4` independent**
    - moreover, **all data can be dumped** after the initialisation **and re-loaded**/re-used
      $\Longrightarrow$ `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)

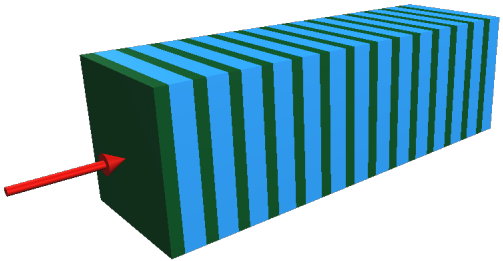
- **representative**: EM shower simulation is at **the core** of the complete detector simulation
  - **the same simulation** (**algorithm**, interactions, models, etc.) **of the EM showers** that is **used** today e.g. **in the ATLAS and CMS detectors**
  - plays **essential** role in many different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

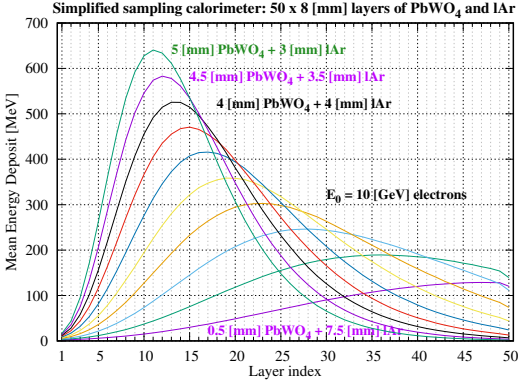
HepEmShow: HEP style simulation of EM showers in a configurable simplified sampling calorimeter (a **simple, compact, small** but **a representative part** of the generic problem)

- **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 $\rightarrow$ 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

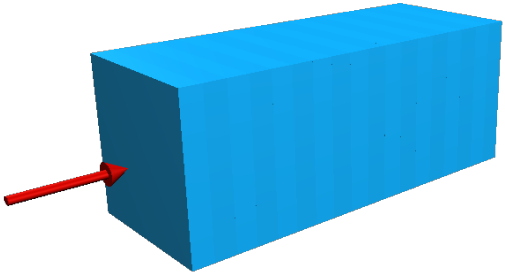
- geometry: *number* of `layers`, `absorber` and/or `gap` *thickness* (`gap` can be zero), *material*
- primary particles: `particle` *type*, kinetic *energy*

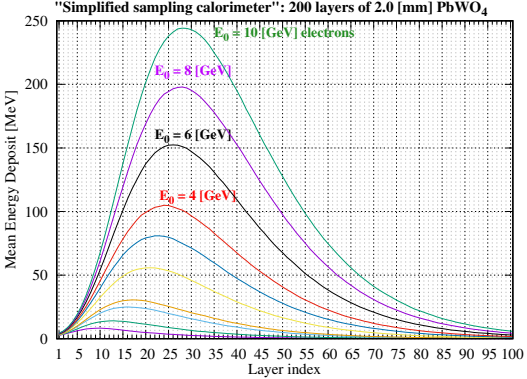




HepEmShow: HEP style simulation of EM showers in a **configurable** simplified sampling calorimeter

- 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 October ☺) in the **repository**

Marc & Alberto wanted to let you know about this