G4HepEm and Specialized Stepping/Tracking in Geant4

Jonas Hahnfeld, Benjamin Morgan, Mihály Novák

Outline

1. Main motivations, ideas in a nutshell
2. Current state: EM shower simulation capability and verification
3. The very first performance numbers
4. Specialised tracking through a `G4VTrackingManager`
5. Some additional notes

Contents

1. Main motivations, ideas in a nutshell
2. Current state: EM shower simulation capability and verification
3. The very first performance numbers
4. Specialised tracking through a `G4VTrackingManager`
5. Some additional notes

G4HepEm: **motivations** in a nutshell

- initiated by the **Geant4 EM physics working group** as part of looking for solutions **to reduce the computing performance bottleneck** experienced by the **HEP detector simulation** applications
- **targeting** the most performance critical part of the HEP detector simulation applications, i.e. the **EM shower generation** covering(initially) e^-/e^+ and γ particle transport
- the main goal is to investigate the **possible computing performance benefits of**
  - **providing alternative, highly specialised** (for particle types, e^-/e^+, γ and HEP applications) **optional stepping loops** beyond the current general one
    $\Longrightarrow$ **giving up the** *"unutilised"* **flexibility with the hope of some performance gain**
  - having a very **compact and efficient implementation of** all the related **run time functionalities** required for an EM shower simulation
    $\Longrightarrow$ **compact run time library** and **data layout with the hope of some performance gain**
- the main design principles
  - separation of initialisation- and run-time functionalities $\Longrightarrow$ in order to have a compact run-time library
  - separation of data and functionality $\Longleftarrow$ since data are filled at initialisation- while used at run-time
- resulted in a run-time **EM shower simulation library with** many attractive characteristics such as the device(**GPU**) side **support** of all related computations (utilised in **AdePT**) or its **stateless** property that, together with its simplicity, provides an excellent domain to check many further interesting ideas
- see the **initial presentation** or the one at the last **Geant4 technical forum** on G4HepEm for more details

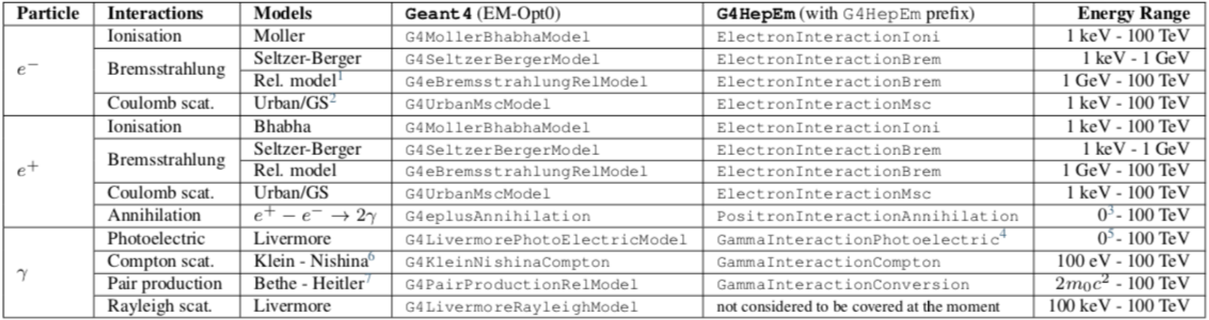
Contents

1. Main motivations, ideas in a nutshell
2. Current state: EM shower simulation capability and verification
3. The very first performance numbers
4. Specialised tracking through a `G4VTrackingManager`
5. Some additional notes

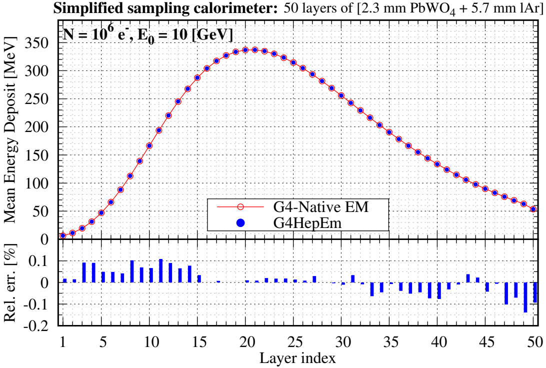
Physics coverage

- the core part of the physics, required for an EM shower simulation in HEP detectors, has just been implemented (see below with all details in the **documentation**)
- this is the essential set required for the very first performance evaluations
- the remaining parts (such as energy loss fluctuation or gamma- and lepto-nuclear interactions) will be handled in case this core part provides promising results

Table 1.1: Summary of the physics interactions and models used in “Geant4“ and “G4HepEm“ (current state).



Particle	Interactions	Models	Geant4 (EM-Opt0)	G4HepEm (with G4HepEm prefix)	Energy Range
e^-	Ionisation	Moller	G4MollerBhabhaModel	ElectronInteractionIoni	1 keV - 100 TeV
	Bremsstrahlung	Seltzer-Berger	G4SeltzerBergerModel	ElectronInteractionBrem	1 keV - 1 GeV
		Rel. model[1]	G4eBremsstrahlungRelModel	ElectronInteractionBrem	1 GeV - 100 TeV
	Coulomb scat.	Urban/GS[2]	G4UrbanMscModel	ElectronInteractionMsc	1 keV - 100 TeV
e^+	Ionisation	Bhabha	G4MollerBhabhaModel	ElectronInteractionIoni	1 keV - 100 TeV
	Bremsstrahlung	Seltzer-Berger	G4SeltzerBergerModel	ElectronInteractionBrem	1 keV - 1 GeV
		Rel. model	G4eBremsstrahlungRelModel	ElectronInteractionBrem	1 GeV - 100 TeV
	Coulomb scat.	Urban/GS	G4UrbanMscModel	ElectronInteractionMsc	1 keV - 100 TeV
	Annihilation	$e^+ - e^- \to 2\gamma$	G4eplusAnnihilation	PositronInteractionAnnihilation	0[3] - 100 TeV
γ	Photoelectric	Livermore	G4LivermorePhotoElectricModel	GammaInteractionPhotoelectric[4]	0[5] - 100 TeV
	Compton scat.	Klein - Nishina[6]	G4KleinNishinaCompton	GammaInteractionCompton	100 eV - 100 TeV
	Pair production	Bethe - Heitler[7]	G4PairProductionRelModel	GammaInteractionConversion	$2m_0c^2$ - 100 TeV
	Rayleigh scat.	Livermore	G4LivermoreRayleighModel	not considered to be covered at the moment	100 keV - 100 TeV



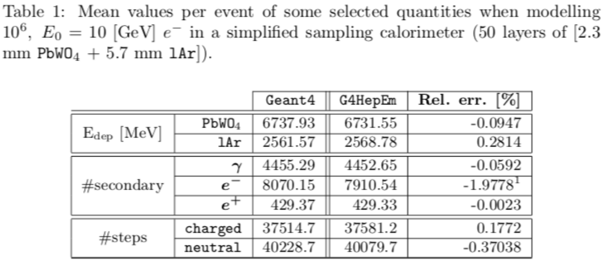


Table 1: Mean values per event of some selected quantities when modelling 10^6, $E_0 = 10$ [GeV] e^- in a simplified sampling calorimeter (50 layers of [2.3 mm $\texttt{PbWO}_4$ + 5.7 mm $\texttt{lAr}$]).

		Geant4	G4HepEm	Rel. err. [%]
E_{dep} [MeV]	$\texttt{PbWO}_4$	6737.93	6731.55	-0.0947
	lAr	2561.57	2568.78	0.2814
#secondary	γ	4455.29	4452.65	-0.0592
	e^-	8070.15	7910.54	-1.9778[1]
	e^+	429.37	429.33	-0.0023
#steps	charged	37514.7	37581.2	0.1772
	neutral	40228.7	40079.7	-0.37038

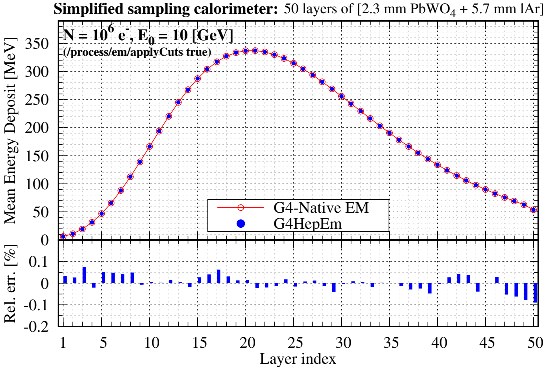
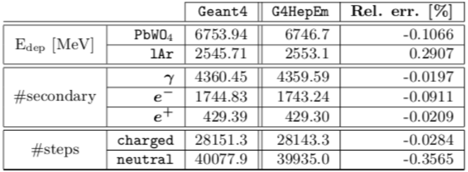


Table 1: Mean values per event of some selected quantities when modelling 10^6, E_0 = 10 [GeV] e^- in a simplified sampling calorimeter (50 layers of [2.3 mm PbWO$_4$ + 5.7 mm lAr]). Using /process/em/applyCuts true



		Geant4	G4HepEm	Rel. err. [%]
E$_{dep}$ [MeV]	PbWO$_4$	6753.94	6746.7	-0.1066
	lAr	2545.71	2553.1	0.2907
#secondary	γ	4360.45	4359.59	-0.0197
	e^-	1744.83	1743.24	-0.0911
	e^+	429.39	429.30	-0.0209
#steps	charged	28151.3	28143.3	-0.0284
	neutral	40077.9	39935.0	-0.3565

/process/em/applyCuts true: mean number of secondary e^- is reduced by ∼ 80 % (charged steps by ∼ 25 %)!

Contents

1. Main motivations, ideas in a nutshell
2. Current state: EM shower simulation capability and verification
3. The very first performance numbers
4. Specialised tracking through a `G4VTrackingManager`
5. Some additional notes

Configurations and notations:

- G4HepEm build: dynamic and it is still not optimised, e.g. the standard mathematical functions are used instead of the optimised versions of GEANT4
- Geant4 build: master with static libraries; -DGEANT4_BUILD_STORE_TRAJECTORY=OFF and -DGEANT4_BUILD_VERBOSE_CODE=OFF; on AMD Ryzen 9 3900
- hardware: 12 core AMD Ryzen 9 3900

- application TestEm3: 10 [GeV] e^- in a simplified sampling calorimeter with 50 layers of 2.3 [mm] $PbWO_4$ and 5.7 [mm] liquid-Ar using the default EM settings (e.g. /process/em/applyCuts false)
  $\Longrightarrow$ pure EM shower simulation
- application cms2018: CMS geometry with gg2ttbar events with physics settings similar to CMSSW, including hadronic physics, gamma-lepto-nuclear processes, a propagation in constant field and optimisations such as /process/em/applyCuts true
  $\Longrightarrow$ close to production settings with its realistic EM fraction

- results Physics List: using the usual physics list interface, i.e. the general stepping loop, either with G4NativeEm processes or G4HepEm
- results Specialised Tracking: using a specialised stepping/tracking loop implementation, i.e. NOT the general stepping loop, either with G4NativeEm processes or G4HepEm

Setup: TestEm3, 100k e^-, 10 GeV, 24 threads on AMD Ryzen 9 3900 (**default EM settings**)

	Physics List	Specialised Tracking	difference
G4NativeEm	500 s	426 s	-14.8 %
G4HepEm	459 s	373 s	-18.7 %
difference	-8.2 %	-12.4 %	-25.4 %

Setup: cms2018, 1000x the same gg2ttbar event, 24 threads on AMD Ryzen 9 3900 (**optimised EM**)

	Physics List	Specialised Tracking	difference
G4NativeEm	2889 s	2747 s	-4.9 %
G4HepEm	2847 s	2660 s	-6.6 %
difference	-1.5 %	-3.2 %	-7.9 %

Note: significant performance gain due to the specialised tracking of e^-/e^+ and γ even already using Geant4 native processes that is boosted further with G4HepEm (even in its current, preliminary phase)

Using native GEANT4 processes with a single thread. The minimum time of three runs are reported.

setup	Physics List	Specialised Tracking	difference
TestEm3, 1000 e$^-$, 10 GeV	65.99 s	57.40 s	-13.02 %
plus magnetic field, $B_z = 1T$	78.45 s	71.59 s	-8.74 %
cms2018, one gg2ttbar event*	42.35 s	40.29 s	-4.85 %

Note: significant performance gain due to the specialised tracking of e^-/e^+ and γ even already using GEANT4 native processes while obtaining numerically identical results!

Using native GEANT4 processes with a single thread. The minimum time of three runs are reported.

setup	Physics List	Specialised Tracking	difference
TestEm3, 1000 e^{-}, 10 GeV	65.99 s	57.40 s	-13.02 %
plus magnetic field, $B_z = 1\,T$	78.45 s	71.59 s	-8.74 %
cms2018, one gg2ttbar event*	42.35 s	40.29 s	-4.85 %

Note: significant performance gain due to the specialised tracking of e^{-}/e^{+} and γ even already using GEANT4 native processes while obtaining numerically identical results!

Both the specialised tracking/stepping and G4HepEm (already in its current state) provides significant performance improvements! How do we provide the possibility of implementing these specialised/external tracking?

Contents

1. Main motivations, ideas in a nutshell

2. Current state: EM shower simulation capability and verification

3. The very first performance numbers

4. Specialised tracking through a `G4VTrackingManager`

5. Some additional notes

Before showing how we implemented these external tracking, **please note**:

- this is our current and suggested solution that fulfils the requirements needed for these specialised/external stepping/tracking (and a bit more)
- due to the nature of `G4HepEm`, the current solution might or might not be optimal for other external tracking such as GPU based EM shower simulation (I will get back to this)
- a **WIP merge request (`geant4/geant4-dev!2105`)** has been opened in order to share our current solution and start a discussion on this topic

Interface – `G4VTrackingManager`:

- Proposal: introduce new (optional) interface `G4VTrackingManager`
  - Attached to `G4ParticleDefinition` (similar to process manager)
  - Invoked from `G4EventManager` instead of generic stepping loop (see next slides)
  - Draft merge request: `geant4/geant4-dev!2105`

Interface – G4VTrackingManager:

- Proposal: introduce new (optional) interface `G4VTrackingManager`
  - Attached to `G4ParticleDefinition` (similar to process manager)
  - Invoked from `G4EventManager` instead of generic stepping loop (see next slides)
  - Draft merge request: geant4/geant4-dev!2105

- Important methods:
  - `virtual void HandOverOneTrack(G4Track* aTrack) = 0;`
    - Implementation can decide what to do (e.g. track immediately, defer processing, etc.)

Interface – `G4VTrackingManager`:

- Proposal: introduce new (optional) interface `G4VTrackingManager`
  - Attached to `G4ParticleDefinition` (similar to process manager)
  - Invoked from `G4EventManager` instead of generic stepping loop (see next slides)
  - Draft merge request: `geant4/geant4-dev!2105`

- Important methods:
  - `virtual void HandOverOneTrack(G4Track* aTrack) = 0;`
    - Implementation can decide what to do (e.g. track immediately, defer processing, etc.)
  - `virtual void FlushEvent() {}`
    - Called by GEANT4 after no tracks left for the current event.
    - If tracks were deferred, implementation *must* process buffered tracks.
    - GEANT4 kernel is prepared that secondaries might be stacked.

Integration into G4EventManager:

Current code in G4EventManager::DoProcessing (simplified)

```
while (track = trackContainer->PopNextTrack(/* ... */)) {

  trackManager->ProcessOneTrack(track);

  /* ... (get trajectory, stack secondaries, delete track) */

}
```

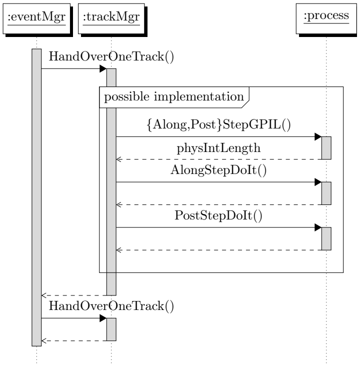
(G4TrackingManager* trackManager is a member of G4EventManager)

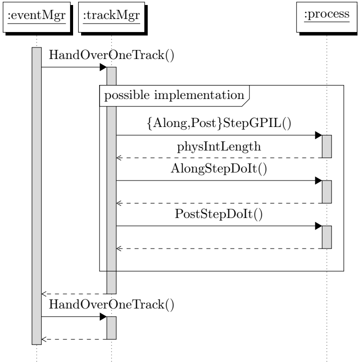
Integration into G4EventManager:

Relevant part of G4EventManager::DoProcessing (simplified)

```
std::unordered_set<G4VTrackingManager *> trackingManagersToFlush;
do {
  while (track = trackContainer->PopNextTrack(/* ... */))) {
    auto* partDef = track->GetParticleDefinition();
    auto* particleTrackingManager = partDef->GetTrackingManager();
    if (particleTrackingManager != nullptr) {
      particleTrackingManager->HandOverOneTrack(track);
      trackingManagersToFlush.insert(particleTrackingManager);
    } else { /* ... (call generic G4TrackingManager as before) */ }
  }

  for (G4VTrackingManager *tm : trackingManagersToFlush) {
    tm->FlushEvent();
  }
  trackingManagersToFlush.clear();
} while (trackContainer->GetNUrgentTrack() > 0);
```


:eventMgr
:trackMgr
:process
HandOverOneTrack()
possible implementation
{Along,Post}StepGPIL()
physIntLength
AlongStepDoIt()
PostStepDoIt()
HandOverOneTrack()



:eventMgr

:trackMgr

:process

HandOverOneTrack()

possible implementation

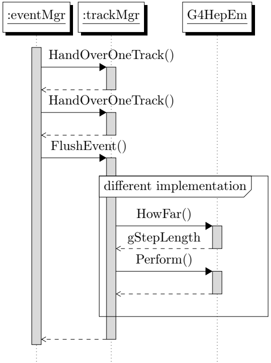
{Along,Post}StepGPIL()

physIntLength

AlongStepDoIt()

PostStepDoIt()

HandOverOneTrack()



:eventMgr

:trackMgr

G4HepEm

HandOverOneTrack()

HandOverOneTrack()

FlushEvent()

different implementation

HowFar()

gStepLength

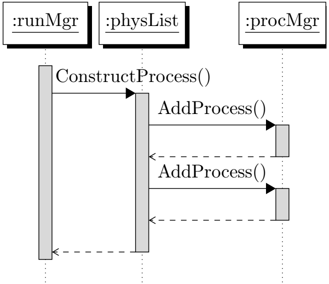
Perform()

Registration of `G4VTrackingManager`:

- Specializing physics processes → same mechanisms as processes
  - Registration in physics list
  - Advantage: users only need to change to a different physics constructor

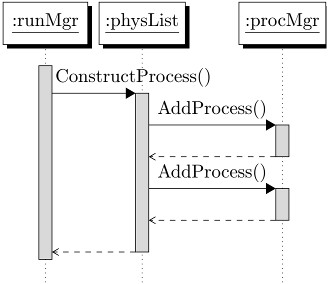
Registration of G4VTrackingManager:

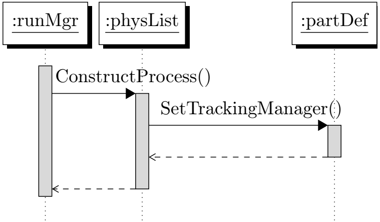
- Specializing physics processes → same mechanisms as processes
  - Registration in physics list
  - Advantage: users only need to change to a different physics constructor



Registration of G4VTrackingManager:

- Specializing physics processes → same mechanisms as processes
  - Registration in physics list
  - Advantage: users only need to change to a different physics constructor





Contents

1. Main motivations, ideas in a nutshell
2. Current state: EM shower simulation capability and verification
3. The very first performance numbers
4. Specialised tracking through a `G4VTrackingManager`
5. Some additional notes

- G4HepEm and/or specialised tracking might be applied in the entire detector since it provides performance improvements even in the case of a single step with a single particle
- this might not be the case for other external tracking such as GPU based simulation
- the latter might be applied only in a specific detector region that would require slightly different solution: the proposed solution checks only when the particle is popped from the stack while the latter would require a check at each simulation steps
- any solution is good for us that can provide the same functionalities that the proposed interface without performance penalty