G4HepEm: a Geant4 EM physics WG R&D

Jonas Hahnfeld, Benjamin Morgan, Mihály Novák

Outline

Contents

G4HepEm: **motivations** in a nutshell

- initiated within the **Geant4 EM physics Working Group** as part of the solutions aiming **to reduce the computing performance bottleneck** experienced by the **HEP detector simulation** applications (see the **initial presentation**)
- **targeting** the most performance critical part of the HEP detector simulation applications, i.e. the **EM shower generation** covering e^-/e^+ and γ particle transport
- the main idea was to investigate the **possible computing performance benefits of**:
  - **alternative simulation stepping loops highly specialised for the particle types** (e.g. one for e^-/e^+ and other one for γ) and **HEP** applications
    $\Longrightarrow$ **giving up the** *"unutilised"* **flexibility** of the *generic stepping loop* **for performance gain**
  - with a **compact and efficient implementation of** the related **run-time functionalities**
    $\Longrightarrow$ **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** $\Longrightarrow$ compact run-time library: only what is really necessary
  - separation of data and functionality $\Longleftarrow$ since data are filled at initialisation- while used at run-time
- several very attractive properties: stateless, compact, self contained run-time library for EM shower simulation naturally supporting devices side computations

Contents

`G4HepEm`: library **structure** is determined by the main goals and design

- **clear separation of run-time and initialisation-time functionalities**:
  - many information are needed at initialisation time but only a small fraction of that is used at run time
  - e.g. computation of some integrated quantities (e.g. the restricted stopping-power) requires lots of information and functionalities but at run-time these are not more than $\{E, y\}_i$ (or $\{E, y, y''\}_i$) data that are interpolated
  - provides **as compact run-time library as possible**

G4HepEm: library **structure** is determined by the main goals and design

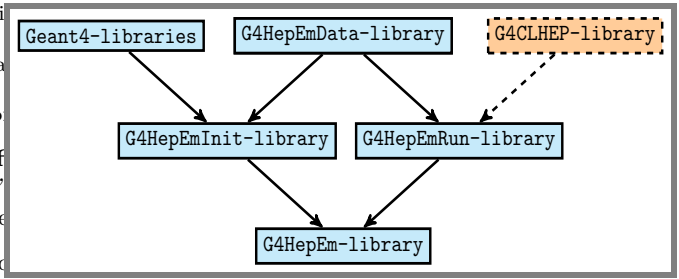
- **clear separation of run-time and initialisation-time functionalities**:
  - many information are needed at initialisation time but only a small fraction of that is used at run time
  - e.g. computation of some integrated quantities (e.g. the restricted stopping-power) requires lots of information and functionalities but at run-time these are not more than $\{E, y\}_i$ (or $\{E, y, y''\}_i$) data that are interpolated
  - provides **as compact run-time library as possible**
- results in **separation of** the **data definitions and functionalities** (i.e. very often more C-style than C++): isolated, *"single function"* implementation of the **run-time functionalities**, acting on and according to their input arguments (mostly primitive or internal data structures)

`G4HepEm`: library **structure** is determined by the main goals and design

- **clear separation of run-time and initialisation-time functionalities**:
  - many information are needed at initialisation time but only a small fraction of that is used at run time
  - e.g. computation of some integrated quantities (e.g. the restricted stopping-power) requires lots of information and functionalities but at run-time these are not more than $\{E, y\}_i$ (or $\{E, y, y''\}_i$) data that are interpolated
  - provides **as compact run-time library as possible**
- results in **separation of** the **data definitions and functionalities** (i.e. very often more C-style than C++): isolated, *"single function"* implementation of the **run-time functionalities**, acting on and according to their input arguments (mostly primitive or internal data structures)
- all these above have lots of benefits and result in attractive properties (we sill see some soon)

G4HepEm: library **structure** is determined by the main goals and design

- **clear separation of run-time and initialisation-time functionalities**:
  - many information are needed at initi
  - e.g. computation of some integrated information and functionalities but a that are interpolated
  - provides **as compact run-time lib**
- results in **separation of** the **data def** than C++): isolated, *"single function"* on and according to their input argume
- all these above have lots of benefits and
- G4HepEm is structured along this separation:
  - G4HepEmData: definition of all data structures **filled at initialisation** and **used at run-time**
  - G4HepEmInit: all **initialisation time functionalities** (e.g. for **constructing** and **populating** the above **data structures**); **relies heavily on core Geant4 functionalities**
  - G4HepEmRun: all **run-time functionalities** (e.g. **reading**/(interpolating) the above **data structures**, **computing the step lengths** and **performing physics interactions**)
  - G4HepEm: a tiny library for connecting all the above



Contents

Some of the properties and their benefits:

- **separating run- and initialisation-time functionalities** results:
  - compact `G4HepEmRun` library: only what is needed at run-time
  - data memory layouts resonate with their run-time access pattern

$\Longrightarrow$ **enhanced cache efficiency through improved locality** $\rightarrow$ expected performance benefit

Some of the properties and their benefits:

- **separating run- and initialisation-time functionalities** results:
  - compact `G4HepEmRun` library: only what is needed at run-time
  - data memory layouts resonate with their run-time access pattern

$\Longrightarrow$ **enhanced cache efficiency through improved locality** $\rightarrow$ expected performance benefit

- especially when a complete simulation step can be done within the compact `G4HepEmRun` library
- even more if sub-sequent steps can be done
- even more when all these can be done with more than one particles simultaneously (see next)

Some of the properties and their benefits:

- **separating run- and initialisation-time functionalities** results:
  - compact `G4HepEmRun` library: only what is needed at run-time
  - data memory layouts resonate with their run-time access pattern

  $\Longrightarrow$ **enhanced cache efficiency through improved locality** $\rightarrow$ expected performance benefit
- **separation of data definition** and **functionalities** results:
  - self contained, *"single-function"* implementation of most of the run-time functionalities
  - functions do not contain or interact with further objects: no long chains of calls
  - simply act on and according to their input arguments: (mainly) internal data structures or primitives

  $\Longrightarrow$ **`G4HepEmRun` library is stateless** $\rightarrow$ chance for (opportunistic) multi particle computation

Some of the properties and their benefits:

- **separating run- and initialisation-time functionalities** results:
  - compact `G4HepEmRun` library: only what is needed at run-time
  - data memory layouts resonate with their run-time access pattern

  $\Longrightarrow$ **enhanced cache efficiency through improved locality** $\rightarrow$ expected performance benefit
- **separation of data definition** and **functionalities** results:
  - self contained, *"single-function"* implementation of most of the run-time functionalities
  - functions do not contain or interact with further objects: no long chains of calls
  - simply act on and according to their input arguments: (mainly) internal data structures or primitives

  $\Longrightarrow$ **`G4HepEmRun` library is stateless** $\rightarrow$ chance for (opportunistic) multi particle computation

  - e.g. popping-up more than one secondary e^-/e^+ or γ tracks (from the internal stacks) and computing their steps together
  - even **partial computation of the step** can be done (e.g. only the physics step computation part)

Some of the properties and their benefits:

- **separating run- and initialisation-time functionalities** results:
  - compact `G4HepEmRun` library: only what is needed at run-time
  - data memory layouts resonate with their run-time access pattern

  $\Longrightarrow$ **enhanced cache efficiency through improved locality** $\rightarrow$ expected performance benefit
- **separation of data definition** and **functionalities** results:
  - self contained, *"single-function"* implementation of most of the run-time functionalities
  - functions do not contain or interact with further objects: no long chains of calls
  - simply act on and according to their input arguments: (mainly) internal data structures or primitives

  $\Longrightarrow$ **`G4HepEmRun` library is stateless** $\rightarrow$ chance for (opportunistic) multi particle computation
- rather supportive for device side computations:
  - the self contained, stateless run-time functions can be easily reused on the device as device side functions or transformed to kernels (see next talk)

  $\Longrightarrow$ ***implicit*** **device side support**

Some of the properties and their benefits:

- **separating run- and initialisation-time functionalities** results:
  - compact `G4HepEmRun` library: only what is needed at run-time
  - data memory layouts resonate with their run-time access pattern

  $\Longrightarrow$ **enhanced cache efficiency through improved locality** $\rightarrow$ expected performance benefit
- **separation of data definition** and **functionalities** results:
  - self contained, *"single-function"* implementation of most of the run-time functionalities
  - functions do not contain or interact with further objects: no long chains of calls
  - simply act on and according to their input arguments: (mainly) internal data structures or primitives

  $\Longrightarrow$ **`G4HepEmRun` library is stateless** $\rightarrow$ chance for (opportunistic) multi particle computation
- rather supportive for device side computations:
  - the self contained, stateless run-time functions can be easily reused on the device as device side functions or transformed to kernels (see next talk)

  $\Longrightarrow$ ***implicit*** **device side support**
  - all the data, required by these run-time functions, can be made available on the device (by a single call)

  $\Longrightarrow$ ***explicit*** **device side support**

Some of the properties and their benefits:

- **separating run- and initialisation-time functionalities** results:
  - compact `G4HepEmRun` library: only what is needed at run-time
  - data memory layouts resonate with their run-time access pattern

  $\Longrightarrow$ **enhanced cache efficiency through improved locality** $\rightarrow$ expected performance benefit
- **separation of data definition** and **functionalities** results:
  - self contained, *"single-function"* implementation of most of the run-time functionalities
  - functions do not contain or interact with further objects: no long chains of calls
  - simply act on and according to their input arguments: (mainly) internal data structures or primitives

  $\Longrightarrow$ **`G4HepEmRun` library is stateless** $\rightarrow$ chance for (opportunistic) multi particle computation
- rather supportive for device side computations:
  - the self contained, stateless run-time functions can be easily reused on the device as device side functions or transformed to kernels (see next talk)

  $\Longrightarrow$ ***implicit*** **device side support**
  - all the data, required by these run-time functions, can be made available on the device (by a single call)

  $\Longrightarrow$ ***explicit*** **device side support**

The implicit device side support has been greatly extended: the **same code can be utilised** now for the EM shower simulation **both on the host and device** (**see the details in the next talk by Jonas)**

Contents

Physics coverage, verification:

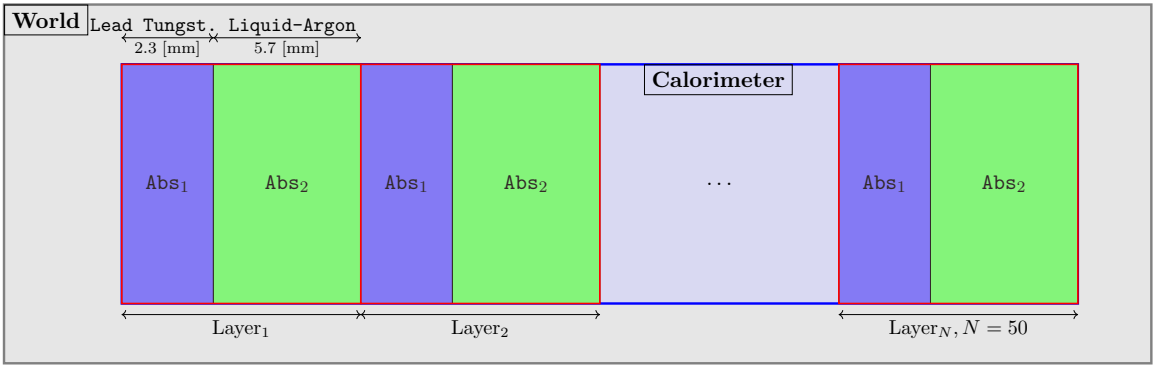
- **complete set of physics**, required for EM shower simulation of e^-/e^+ and γ in HEP detectors over the ∼[keV]-100 [TeV] kinetic energy range, is **available and verified** (see **documentation**)
- hadronic channels (e.g. gamma- and lepto-nuclear interactions) are coming soon

Table 1.1: Summary of the physics interactions and models used in `Geant4` and `G4HepEm` (current state). Energy loss fluctuation, corresponding to the `Geant4-11.p01` version of the `G4UniversalFluctuation` model, has also been implemented for e^-/e^+ and used in `G4HepEm` beyond the listed models.

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` (including both models)	1 keV - 1 GeV
		Rel. model[1]	`G4eBremsstrahlungRelModel`		1 GeV - 100 TeV
	Coulomb scat.[2]	Urban	`G4UrbanMscModel`	`ElectronInteractionUMSC`	1 keV - 100 MeV
		Wentzel-VI	`G4WentzelVIModel`		100 MeV - 100 TeV
e^+	Ionisation	Bhabha	`G4MollerBhabhaModel`	`ElectronInteractionIoni`	1 keV - 100 TeV
	Bremsstrahlung	Seltzer-Berger	`G4SeltzerBergerModel`	`ElectronInteractionBrem` (including both models)	1 keV - 1 GeV
		Rel. model	`G4eBremsstrahlungRelModel`		1 GeV - 100 TeV
	Coulomb scat.	Urban	`G4UrbanMscModel`	`ElectronInteractionUMSC`	1 keV - 100 MeV
		Wentzel-VI	`G4WentzelVIModel`		100 MeV - 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

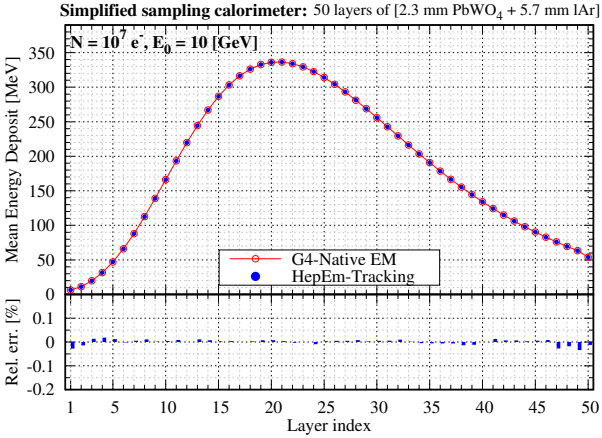
Physics coverage, verification:

- **complete set of physics**, required for EM shower simulation of e^-/e^+ and γ in HEP detectors over the ∼[keV]-100 [TeV] kinetic energy range, is **available and verified** (see **documentation**)
- hadronic channels (e.g. gamma- and lepto-nuclear interactions) are coming soon



Physics coverage, verification:

- **complete set of physics**, required for EM shower simulation of e^-/e^+ and γ in HEP detectors over the ∼[keV]-100 [TeV] kinetic energy range, is **available and verified** (see **documentation**)
- hadronic channels (e.g. gamma- and lepto-nuclear interactions) are coming soon

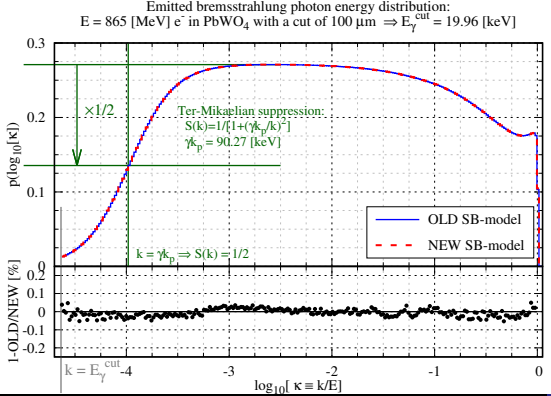


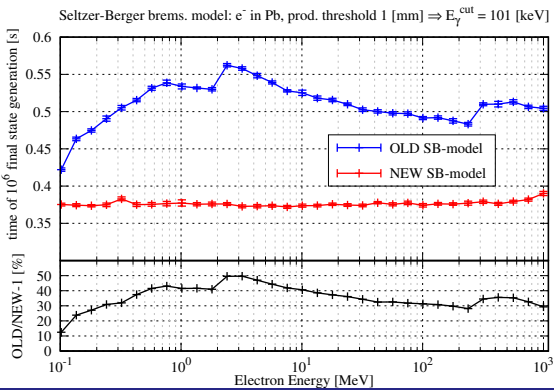
What is different compared to EM standard (EM-opt0):

- no *real algorithmic* changes: same number of simulation steps are performed in the same way but...
- some are purely technical (e.g. the *Seltzer-Berger* brem. model) $\Longrightarrow$ ***no change expected***

What is different compared to EM standard (EM-opt0):

- no *real algorithmic* changes: same number of simulation steps are performed in the same way but...
- some are purely technical (e.g. the *Seltzer-Berger* brem. model) $\Longrightarrow$ ***no change expected***
  - we use a 2D interpolation based rejection sampling in `Geant4` (2D numerical DCS)
  - we implemented a rejection free version in `G4HepEm` (with the same memory cost)





What is different compared to EM standard (EM-opt0):

- no *real algorithmic* changes: same number of simulation steps are performed in the same way but...
- some are purely technical (e.g. the *Seltzer-Berger* brem. model) $\Longrightarrow$ ***no change expected***
- *Rayleigh* scattering is not included (not relevant) $\Longrightarrow$ ***no visible change expected***
- a slightly *"relaxed"* model for the photoelectric effect $\Longrightarrow$ ***#(very low energy)*** secondary e^- ***might be slightly different but no visible change (e.g. in energy deposit) expected***

What is different compared to EM standard (EM-opt0):

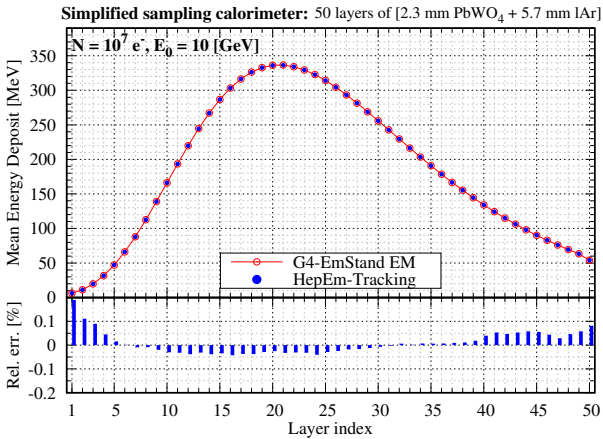
- no *real algorithmic* changes: same number of simulation steps are performed in the same way but...
- some are purely technical (e.g. the *Seltzer-Berger* brem. model) $\Longrightarrow$ ***no change expected***
- *Rayleigh* scattering is not included (not relevant) $\Longrightarrow$ ***no visible change expected***
- a slightly *"relaxed"* model for the photoelectric effect $\Longrightarrow$ ***#(very low energy)*** secondary e^- ***might be slightly different but no visible change (e.g. in energy deposit) expected***
  - instead of the most sophisticated *Livermore* cross section based model
  - a simpler one is used considering only the *K-shell* binding energies
  - (very low energy) secondary e^- is not generated when the photon energy is below this energy
  - i.e. it's assumed that the corresponding secondaries wouldn't go far anyway

What is different compared to EM standard (EM-opt0):

- no *real algorithmic* changes: same number of simulation steps are performed in the same way but...
- some are purely technical (e.g. the *Seltzer-Berger* brem. model) $\Longrightarrow$ ***no change expected***
- *Rayleigh* scattering is not included (not relevant) $\Longrightarrow$ ***no visible change expected***
- a slightly *"relaxed"* model for the photoelectric effect $\Longrightarrow$ ***#(very low energy)*** secondary e^- ***might be slightly different but no visible change (e.g. in energy deposit) expected***
- e^-/e^+ Coulomb scattering is described by a single model (*Urban* MSC) over the entire energy range $\Longrightarrow$ ***no significant change expected***

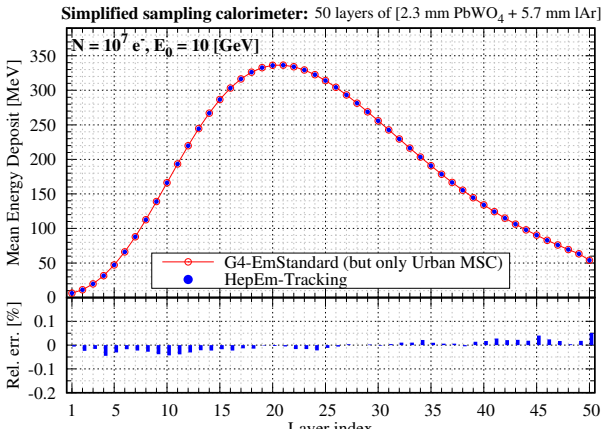
What is different compared to EM standard (EM-opt0):

- no *real algorithmic* changes: same number of simulation steps are performed in the same way but...
- some are purely technical (e.g. the *Seltzer-Berger* brem. model) $\Longrightarrow$ ***no change expected***
- *Rayleigh* scatt ***expected***
- a slightly *"rel* ***nergy)*** secondary e^- ***might be slig*** ***osit) expected***
- e^-/e^+ Coulor $\Longrightarrow$ ***no sign*** er the entire energy range



What is different compared to EM standard (EM-opt0):

- no *real algorithmic* changes: same number of simulation steps are performed in the same way but...
- some are purely technical (e.g. the *Seltzer-Berger* brem. model) $\Longrightarrow$ ***no change expected***
- *Rayleigh* scatt ***expected***
- a slightly *"rel* ***nergy)*** secondary e^- ***might be slig*** ***osit) expected***
- e^-/e^+ Coulor er the entire energy range $\Longrightarrow$ ***no sign***



Utilisation:

- `G4HepEm` is still in **R**&**D** phase with its external `GitHub` repository: **mnovak42/g4hepem**
- it can be utilised in any `Geant4` application in different ways: (or in any applications in many ways)

Utilisation:

- `G4HepEm` is still in **R&D** phase with its external `GitHub` repository: **mnovak42/g4hepem**
- it can be utilised in any `Geant4` application in different ways: (or in any applications in many ways)

1. using the `G4HepEmProcess` in the `G4VPhysicsConstructor` (i.e. *physics list*) interface:

- `G4HepEmProcess` is a single process implementation of all e^-/e^+ and γ interactions
- can be easily utilised by replacing all native `Geant4` processes with this single one
- when doing so, all physics related computations will be done (for e^-/e^+ and γ) by `G4HepEm`
- **note: the generic stepping loop is utilised** in this case
- the only **performance benefit** might be **due to the compact implementation**
- very useful for the development as well as for the first round of the physics validation

Utilisation:

- `G4HepEm` is still in **R**&**D** phase with its external `GitHub` repository: **mnovak42/g4hepem**
- it can be utilised in any `Geant4` application in different ways: (or in any applications in many ways)

1. using the `G4HepEmProcess` in the `G4VPhysicsConstructor` (i.e. *physics list*) interface:

- `G4HepEmProcess` is a single process implementation of all e^-/e^+ and γ interactions
- can be easily utilised by replacing all native `Geant4` processes with this single one
- when doing so, all physics related computations will be done (for e^-/e^+ and γ) by `G4HepEm`
- **note: the generic stepping loop is utilised** in this case
- the only **performance benefit** might be **due to the compact implementation**
- very useful for the development as well as for the first round of the physics validation

2. using the **new**(in `Geant4-11.0`) `G4VTrackingManager` interface:

- the proposed solution for **specialised stepping**: per particle types and actually much more
- the full control, over the simulation of a given particle type, can be handed over: **even external tracking**
- great flexibility for diversifying the simulation solutions: e.g. special solutions for critical points
- a nice example for `Geant4` **R**&**D**: one year from idea to release

Contents

Configuration:

- Geant4: `version-11.0`

only static libraries	`-DBUILD_SHARED_LIBS`	=	`OFF`
	`-DBUILD_STATIC_LIBS`	=	`ON`
non-verbose build	`-DGEANT4_BUILD_VERBOSE_CODE`	=	`OFF`
no trajectories stored	`-DGEANT4_BUILD_STORE_TRAJECTORY`	=	`OFF`
MT build	`-GEANT4_BUILD_MULTITHREADED`	=	`ON`

- application: ***simplified sampling calorimeter*** with 50 layers of 2.3 [mm] $PbWO_4$ and 5.7 [mm] `liquid-Ar` using the default EM settings $\Longrightarrow$ pure EM shower simulation
- notations:
  1. `Physics List`: using either the Geant4 ***native*** or the `G4HepEmProcess`
  $\Longrightarrow$ using the **generic stepping** loop for e^-/e^+ and γ
  2. `Tracking Manager`: using either the Geant4 ***native*** processes or `G4HepEm`
  $\Longrightarrow$ **specialised tracking** for e^-/e^+ and γ through the **new** `G4VTrackingManager` interface

Configuration:

- Geant4: `version-11.0`

only static libraries	-DBUILD_SHARED_LIBS	=	OFF
	-DBUILD_STATIC_LIBS	=	ON
non-verbose build	-DGEANT4_BUILD_VERBOSE_CODE	=	OFF
no trajectories stored	-DGEANT4_BUILD_STORE_TRAJECTORY	=	OFF
MT build	-GEANT4_BUILD_MULTITHREADED	=	ON

- application: ***simplified sampling calorimeter*** with 50 layers of 2.3 [mm] $PbWO_4$ and 5.7 [mm] `liquid-Ar` using the default EM settings $\Longrightarrow$ pure EM shower simulation
- notations:
  1. `Physics List`: using either the Geant4 ***native*** or the `G4HepEmProcess`
  $\Longrightarrow$ using the **generic stepping** loop for e^-/e^+ and γ
  2. `Tracking Manager`: using either the Geant4 ***native*** processes or `G4HepEm`
  $\Longrightarrow$ **specialised tracking** for e^-/e^+ and γ through the **new** `G4VTrackingManager` interface
- 10^5, 10 [GeV] e^- using 24 threads on a `12 code AMD Ryzen 9 3900` (**default EM settings**)

	`Physics List`	`Tracking Manager`	difference
`G4NativeEm`	473 s	405 s	-14.4 %
`G4HepEm`	414 s	337 s	-18.6 %
difference	-12.5 %	-16.8 %	**-28.7 %**

- it takes $\sim$ 521 [s] using the EM standard physics: -35.3 % difference when using `G4HepEm` with the `specialised tracking`

Contents

Summary:

- `G4HepEm` and the already available *specialised tracking* show very ***promising*** *results*
  - $\Longrightarrow$ *both* the specialised stepping and the compact implementation *ideas work nicely*
  - $\Longrightarrow$ the *compete EM shower simulation is validated*/verified against the native `Geant4` version
  - $\Longrightarrow$ already under pre-validation in experimental framework (ATLAS)

Summary:

- `G4HepEm` and the already available *specialised tracking* show very ***promising*** *results*
  $\Longrightarrow$ *both* the specialised stepping and the compact implementation *ideas work nicely*
  $\Longrightarrow$ the *compete EM shower simulation is validated*/verified against the native `Geant4` version
  $\Longrightarrow$ already under pre-validation in experimental framework (ATLAS)
- thanks to its design, `G4HepEm` offers ***lots of*** *interesting* ***further ideas*** to try
  $\Longrightarrow$ opportunistic multi particle computation is one of them

Summary:

- `G4HepEm` and the already available *specialised tracking* show very ***promising*** *results*
  $\Longrightarrow$ *both* the specialised stepping and the compact implementation *ideas work nicely*
  $\Longrightarrow$ the *compete EM shower simulation is validated*/verified against the native `Geant4` version
  $\Longrightarrow$ already under pre-validation in experimental framework (ATLAS)
- thanks to its design, `G4HepEm` offers ***lots of*** *interesting* ***further ideas*** to try
  $\Longrightarrow$ opportunistic multi particle computation is one of them
- moreover, a *large fraction of the same code is* ***reusable on the GPU***
  $\Longrightarrow$ largely accelerates the related developments, easing the validation, maintenance

Summary:

- `G4HepEm` and the already available *specialised tracking* show very ***promising*** *results*
  $\Longrightarrow$ *both* the specialised stepping and the compact implementation *ideas work nicely*
  $\Longrightarrow$ the *compete EM shower simulation is validated*/verified against the native `Geant4` version
  $\Longrightarrow$ already under pre-validation in experimental framework (ATLAS)
- thanks to its design, `G4HepEm` offers ***lots of*** *interesting* ***further ideas*** to try
  $\Longrightarrow$ opportunistic multi particle computation is one of them
- moreover, a *large fraction of the same code is* ***reusable on the GPU***
  $\Longrightarrow$ largely accelerates the related developments, easing the validation, maintenance
- it was *successfully* ***utilised*** *in* `AdePT` for the EM shower simulation
  $\Longrightarrow$ *made possible the complete EM shower simulation on device*
  $\Longrightarrow$ took $\sim 2-3$ months to obtain the first EM shower simulation on the device

Summary:

- `G4HepEm` and the already available *specialised tracking* show very ***promising*** *results*
  $\Longrightarrow$ *both* the specialised stepping and the compact implementation *ideas work nicely*
  $\Longrightarrow$ the *compete EM shower simulation is validated*/verified against the native `Geant4` version
  $\Longrightarrow$ already under pre-validation in experimental framework (ATLAS)
- thanks to its design, `G4HepEm` offers ***lots of*** *interesting* ***further ideas*** to try
  $\Longrightarrow$ opportunistic multi particle computation is one of them
- moreover, a *large fraction of the same code is* ***reusable on the GPU***
  $\Longrightarrow$ largely accelerates the related developments, easing the validation, maintenance
- it was *successfully* ***utilised*** *in* `AdePT` for the EM shower simulation
  $\Longrightarrow$ *made possible the complete EM shower simulation on device*
  $\Longrightarrow$ took $\sim 2-3$ months to obtain the first EM shower simulation on the device
- the next presentation shows more details on this