G4HepEm: a Geant4 EM physics working group R&D

Mihály Novák, Jonas Hahnfeld, Ben Morgan

Outline

Contents

`G4HepEm`: **motivations & description** 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 replacing the** current **general** particle transport **simulation stepping-loop** of Geant4 **by alternatives, highly specialised for particle types** (i.e. for e^-/e^+ and γ) and **tailored for HEP detector simulations**
- **identifies and extracts all the data and functionalities, required for EM shower simulation** in HEP detectors, **in a** very simple and **compact form**
- **this** clean and **compact** and well documented **environment** for EM shower generation also **provides an excellent domain for further related R&D activities**
- `G4HepEm` provides **special support for** the related R&D **activities targeting** EM shower simulation on **GPU devices**
- it has been made available in order **to facilitate and catalyse correlated** R&D **activities by** providing and **sharing** the related **expertise** and specific knowledge

Contents

Contents

`G4HepEm`: library **structure**

- **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

G4HepEm: library **structure**

- **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

Example: material description

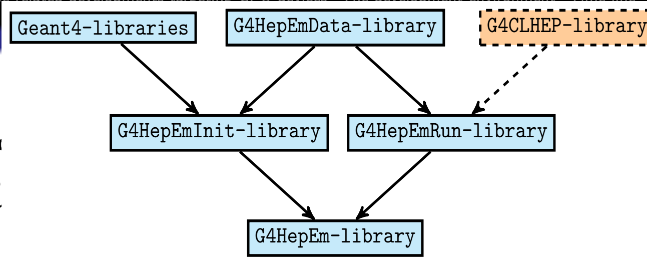
- Geant4 provides a **very rich and sophisticated material description library** with all the extended properties and built in material data bases needed for a wide range of simulations
- but **most of these** functionalities and data **are** actually **used for the** (user) **material definition and at initialisation time computations** (e.g. computation of density correction in the stopping power)
- **only a couple of these** material properties **are used at run-time** during the EM shower generation
- therefore, **a very simple data structure** (with couple of double/integer fields) **is perfectly sufficient** to keep all material related information needed at run-time
- as a consequence, there is no any run-time dependence on the Geant4 material description library
- the same is true for the element, material-cuts couple data and many more complex data

G4HepEm: library **structure**

- **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
  - in order to obtain **as compact run-time library as possible**

G4HepEm: library structure

- **clear separation of run-time and initialisati**
  - many information are needed at initialisatio
  - in order to obtain **as compact run-time libr**
- 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** (based on the given application setup) **relying heavily on core** Geant4 **functionalities**
  - G4HepEmRun: all **run-time functionalities**, e.g. for **reading**/(interpolating) **the data structures** constructed and populated at the initialisation time, **compute the step lengths** and **perform the physics interactions**
  - G4HepEm: a tiny library for connecting all the above; also provides the connection to the Geant4 applications



`G4HepEm`: library **structure**

- **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
  - in order to obtain **as compact run-time library as possible**
- `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** (based on the given application setup) **relying heavily on core** `Geant4` **functionalities**
  - `G4HepEmRun`: all **run-time functionalities**, e.g. for **reading**/(interpolating) **the data structures** constructed and populated at the initialisation time, **compute the step lengths** and **perform the physics interactions**
  - `G4HepEm`: a tiny library for connecting all the above; also provides the connection to the `Geant4` applications
- 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 their input arguments (data structures) **without** storing **any states**!
- all these above have lots of benefits that makes `G4HepEm` a perfect **environment for exploring** several **further R&D ides** (see shortly)

Contents

G4HepEm: connection to Geant4

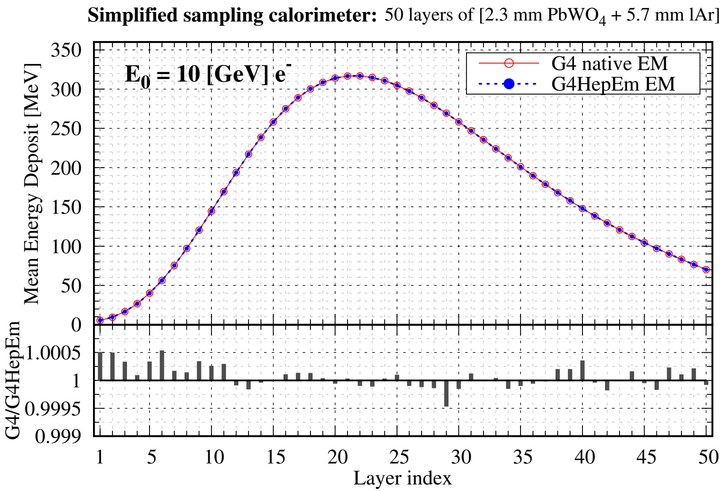
- as many other, similar activities within the Geant4 collaboration, G4HepEm can be part of the toolkit when eventually it will be proven to be successful
- this can be evaluated when the final goal is reached: having specialised simulation stepping-loops in Geant4 that are able to perform complete simulation steps based solely on G4HepEm
- while this ultimate goal requires some modifications in the Geant4 toolkit itself, the G4VProcess (top level Geant4 physics process interface) is used for connecting G4HepEm to (any) Geant4 applications during the developments
- the G4HepEm library contains an implementation of this G4VProcess interface:
  - as a special, *continuous process* covering all interactions with all their parts (continuous, discrete, at-rest)
  - currently includes *ionisation* and *bremsstrahlung* for e^- and *annihilation* $\rightarrow 2\gamma$-s additionally for e^+, conversion, Compton scattering and a simple photoelectric absorption for γ-s
  - all the details are considered (such as the condensed history aspects, i.e. secondary production threshold dependence, continuous energy loss corrections, etc.) and tested in real use cases (e.g. simplified sampling calorimeter as well as CMS detector)
  - the documentation contains a **Table** with the up-to-date state regarding the physics modelling capability
- this special process can be assigned to e^-/e^+ and γ particles in any Geant4 physics list (see more in the documentation)

G4HepEm: connection to Geant4

- **at the initialisation** of this special G4HepEm process, all **data, required at run time** for the simulation, **are extracted and computed** relying heavily on the corresponding Geant4 functionalities: the G4HepEmInit part will be active by creating and populating the data structures defined in G4HepEmData library
- **during the simulation all physics related computations**, e.g.*physics step limit and interactions*, are **performed by the compact G4HepEmRun library** instead of using the native Geant4 implementations
- this provides an excellent (non-invasive) way for **continuous verification of the development** by switching between using either the native Geant4 or the G4HepEm functionalities

G4HepEm: connection to Geant4

- **at the initi** ~~for the simulation,~~ **are extract** ities: the G4HepEmIni ed in G4HepEmData library
- **during the** *nteractions*, are **performed** mplementations
- this provide **lopment** by switching be



G4HepEm: connection to Geant4

- **at the initialisation** of this special G4HepEm process, all **data, required at run time** for the simulation, **are extracted and computed** relying heavily on the corresponding Geant4 functionalities: the G4HepEmInit part will be active by creating and populating the data structures defined in G4HepEmData library
- **during the simulation all physics related computations**, e.g.*physics step limit and interactions*, are **performed by the compact G4HepEmRun library** instead of using the native Geant4 implementations
- this provides an excellent (non-invasive) way for **continuous verification of the development** by switching between using either the native Geant4 or the G4HepEm functionalities
- however, this way of connection **still relies on the general stepping-loop** and requires more than one switches between Geant4 and G4HepEm during a single simulation step

Contents

Contents

Separating run-time and initialisation time functionalities:

- results in a **very compact `G4HepEmRun` library** $\rightarrow$ might give performance improvements, especially when a complete simulation step can be performed within this library, e.g. skipping transportation whenever possible (opportunistically) (implementing field propagation, etc..)
- **separation of data definition** and **functionalities** $\rightarrow$ **self contained, *"single-function"* implementation of most of the `G4HepEmRun` functionalities** (e.g. all interactions, step-limit, etc.)
- it means that these functions do **not contain or interact with further objects** and **act only on their input arguments** $\rightarrow$ `G4HepEmRun` **do not have any states**
- this gives the possibility in the future of e.g. popping-up more than one e^-/e^+ or γ tracks (from the internal secondary stacks) and tracking them together (opportunistically)
- having all the functionalities required for performing the simulation step available in such a compact and simple way provides and **excellent domain for testing many interesting further** R&D **ideas**

Contents

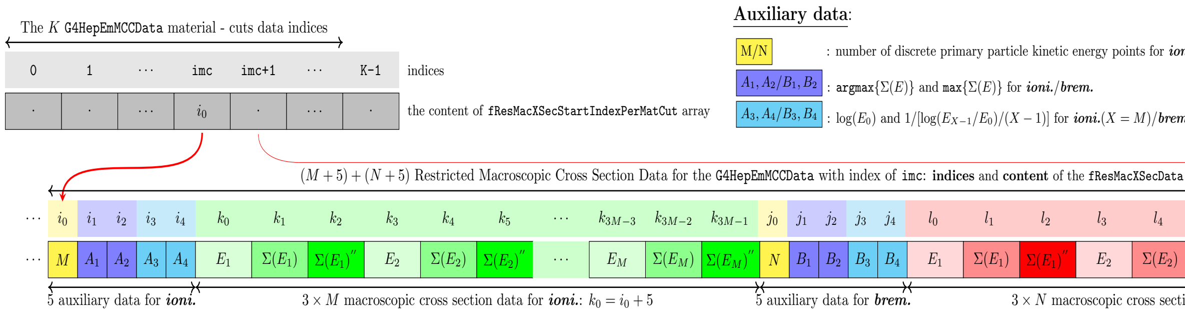
Compact data structures:

- **data structures** (defined in `G4HepEmData`, filled in `G4HepEmInit` and utilised in `G4HepEmRun`), were **designed driven by their run-time usage**
- it means that **their memory layouts** are **determined by their access patterns**

Com

Example: restricted macroscopic cross sections for e^-/e^+ **ionisation** and **bremsstrahlung**.

- **depends on the** material and secondary production threshold, i.e. on the **material-cuts** couple
- **computed**/stored **at initialisation over a discrete energy grid** (unique: for the couple and interaction)
- **used at run-time**: the inverse MFP between hard(/discrete) ionisation/bremsstrahlung events
- the **run-time** (spline) **interpolation** for a given E_{kin} is based on 6 discrete values: $E_i \leq E_{\text{kin}} < E_{i+1}$, $\Sigma(E_i), \Sigma(E_{i+1}), \Sigma(E_i)'', \Sigma(E_{i+1})''$
- evaluated for both interactions during the simulation step in the given material-cuts couple
- **all** the required **values are as close as possible in the memory**


The K G4HepEmMCCData material - cuts data indices
0
1
···
imc
imc+1
···
K-1
indices
·
·
···
i_0
·
···
·
the content of fResMacXSecStartIndexPerMatCut array
Auxiliary data:
M/N
: number of discrete primary particle kinetic energy points for ion
$A_1, A_2/B_1, B_2$
: argmax$\{\Sigma(E)\}$ and max$\{\Sigma(E)\}$ for *ioni./brem.*
$A_3, A_4/B_3, B_4$
: $\log(E_0)$ and $1/[\log(E_{X-1}/E_0)/(X-1)]$ for *ioni.*$(X=M)/$*brem*
$(M+5)+(N+5)$ Restricted Macroscopic Cross Section Data for the G4HepEmMCCData with index of imc: **indices** and **content** of the fResMacXSecData
··· i_0 i_1 i_2 i_3 i_4 k_0 k_1 k_2 k_3 k_4 k_5 ··· k_{3M-3} k_{3M-2} k_{3M-1} j_0 j_1 j_2 j_3 j_4 l_0 l_1 l_2 l_3 l_4
··· M A_1 A_2 A_3 A_4 E_1 $\Sigma(E_1)$ $\Sigma(E_1)''$ E_2 $\Sigma(E_2)$ $\Sigma(E_2)''$ ··· E_M $\Sigma(E_M)$ $\Sigma(E_M)''$ N B_1 B_2 B_3 B_4 E_1 $\Sigma(E_1)$ $\Sigma(E_1)''$ E_2 $\Sigma(E_2)$
5 auxiliary data for *ioni.*
$3\times M$ macroscopic cross section data for *ioni.*: $k_0 = i_0 + 5$
5 auxiliary data for *brem.*
$3\times N$ macroscopic cross secti

Compact data structures:

- **data structures** (defined in `G4HepEmData`, filled in `G4HepEmInit` and utilised in `G4HepEmRun`), were **designed driven by their run-time usage**
- it means that **their memory layouts** are **determined by their access patterns**
- the same is true for all energy loss related data (i.e. restricted stopping power, range, inverse range) but also for the target element selectors, etc.
- the goal is to **enhance data locality** as much as possible, that might bring performance improvements:
  - especially when a complete simulation step 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 (opportunistically as mentioned before)

Contents

Implicit support(with its structure, design, know-how):

- the bare minimum of **functionalities and data**, required for the EM shower generation in the HEP detector simulation domain, are **extracted** from the large `Geant4` code base and provided in a simple form (i.e. provides clean example implementation of the target)
- moreover, this is done by **separating the run-time and initialisation time functionalities**
- furthermore, run-time **functionalities and data are decoupled**
- the required run-time functionalities are implemented as *"single-function"*s, receiving information through and acting on solely on their input arguments
  - the `G4HepEm` functionalities do not have any state
  - do not rely on any further implicit objects (or their functionalities)
  - these functions (used to perform the simulation steps) define the corresponding GPU kernels
  - moreover, due to their implementation, they can be transformed to the corresponding kernels very easily
- all these above are **excellent properties** regarding the corresponding developments **targeting** (EM shower generation in HEP detectors on) **GPU devices**

Implicit support(with its structure, design, know-how):

- the bare minimum of **functionalities and data**, required for the EM shower generation in the HEP detector simulation domain, are **extracted** from the large `Geant4` code base and provided in a simple form (i.e. provides clean example implementation of the target)
- moreover, this is done by **separating the run-time and initialisation time functionalities**
- furthermore, run-time **functionalities and data are decoupled**
- the required run-time functionalities are implemented as *"single-function"*s, receiving information through and acting on solely on their input arguments
    - the `G4HepEm` functionalities do not have any state
    - do not rely on any further implicit objects (or their functionalities)
    - these functions (used to perform the simulation steps) define the corresponding GPU kernels
    - moreover, due to their implementation, they can be transformed to the corresponding kernels very easily
- all these above are **excellent properties** regarding the corresponding developments **targeting** (EM shower generation in HEP detectors on) **GPU devices**
- **especially if the** (complicated and diverse) **data** structures, **required** by the above **run-time** functionalities **would also be available on** the GPU **device**

Implicit support(with its structure, design, know-how):

- the bare minimum of **functionalities and data**, required for the EM shower generation in the HEP detector simulation domain, are **extracted** from the large `Geant4` code base and provided in a simple form (i.e. provides clean example implementation of the target)
- moreover, this is done by **separating the run-time and initialisation time functionalities**
- furthermore, run-time **functionalities and data are decoupled**
- the required run-time functionalities are implemented as *"single-function"*s, receiving information through and acting on solely on their input arguments
  - the `G4HepEm` functionalities do not have any state
  - do not rely on any further implicit objects (or their functionalities)
  - these functions (used to perform the simulation steps) define the corresponding GPU kernels
  - moreover, due to their implementation, they can be transformed to the corresponding kernels very easily
- all these above are **excellent properties** regarding the corresponding developments **targeting** (EM shower generation in HEP detectors on) **GPU devices**
- **especially if the** (complicated and diverse) **data** structures, **required** by the above **run-time** functionalities **would also be available on** the GPU **device**
- **good news:** `G4HepEm` **provides this *explicit* GPU support!** (and even more...)

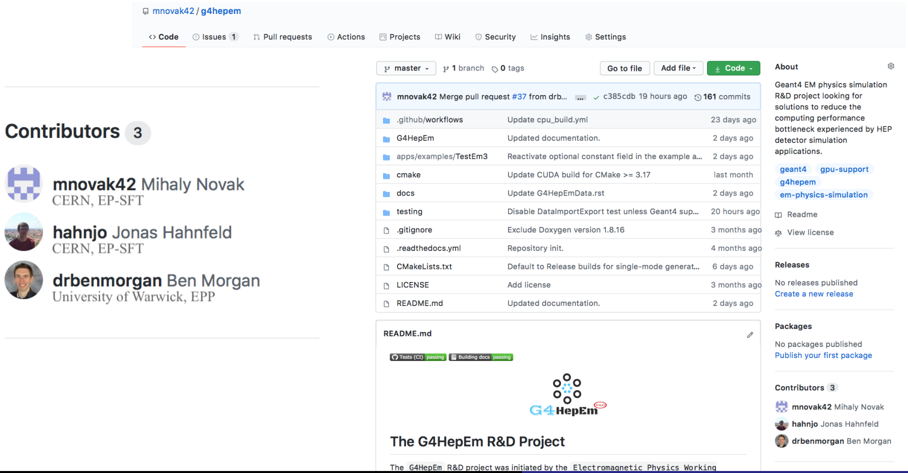
Explicit GPU support:

- when built with the `-DG4HepEm_CUDA_BUILD=ON` CMake option, `G4HepEm` **provides the functionalities to make all data** structures (constructed and populated at initialisation time and) **required at run-time available on the main device memory**
- originally:
  - differences between the host and device side memory access patterns **were accounted** by using a **special device side memory layout to enhance coalesced memory access**
  - special CUDA kernels, accessing these data structures, were also provided by `G4HepEm`
- this **has been dropped**(at least temporarily) and **the same memory layout is used now on the host and device sides**:
  - **thanks to the design**, **the host side data access functions can be reused on the device**: **no need of special CUDA kernels**, **exactly the same code can be used on the host and device sides**
  - **not only the** data and **data access functions**, but practically **all host side, run time functionalities can be reused on the device side** (after abstracting away the random number generator, the only run time object dependence)
- as a result, **the same code** (related to **more than 95%** of the `G4HepEm` host side run time functionalities) **can be reused** as it is **to perform** (the physics related part of) the **EM shower simulation on the GPU**
- **many advantages for R&D activities** targeting EM shower simulation **on GPU**:
  - **significantly accelerating the development process** since all the physics related parts are provided by `G4HepEm` on the device as well
  - **verification/validation** of the simulation or maintenance(no code duplications), etc...

Contents

Development environment:

- at the moment, dedicated **GitHub** repository: **mnovak42/g4hepem**



mnovak42 / **g4hepem**

Code | Issues 1 | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

master · 1 branch · 0 tags · Go to file · Add file · Code

mnovak42 Merge pull request #37 from drb... ✓ c385cdb 19 hours ago · 161 commits

.github/workflows	Update cpu_build.yml	23 days ago
G4HepEm	Updated documentation.	2 days ago
apps/examples/TestEm3	Reactivate optional constant field in the example a...	2 days ago
cmake	Update CUDA build for CMake >= 3.17	last month
docs	Update G4HepEmData.rst	2 days ago
testing	Disable DataImportExport test unless Geant4 sup...	20 hours ago
.gitignore	Exclude Doxygen version 1.8.16	3 months ago
.readthedocs.yml	Repository init.	4 months ago
CMakeLists.txt	Default to Release builds for single-mode generat...	6 days ago
LICENSE	Add license	3 months ago
README.md	Updated documentation.	2 days ago

README.md

Tests (CI) passing · Building docs passing

G4HepEm R&D

The G4HepEm R&D Project

The `G4HepEm` R&D project was initiated by the `Electromagnetic Physics Working`

About

Geant4 EM physics simulation R&D project looking for solutions to reduce the computing performance bottleneck experienced by HEP detector simulation applications.

geant4 · gpu-support · g4hepem · em-physics-simulation

Readme

View license

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 3

mnovak42 Mihaly Novak
hahnjo Jonas Hahnfeld
drbenmorgan Ben Morgan

Contributors 3

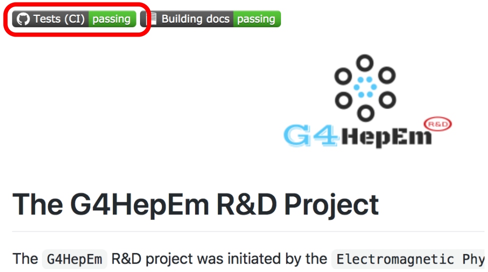
mnovak42 Mihaly Novak
CERN, EP-SFT

hahnjo Jonas Hahnfeld
CERN, EP-SFT

drbenmorgan Ben Morgan
University of Warwick, EPP

Development environment:

- at the moment, dedicated **GitHub** repository: **mnovak42/g4hepem**
- a minimum **Continuous Integration** testing is in place:
  - `Geant4-10.6.p03` on `Ubuntu-20.04` with `gcc-9.3`
  - no CUDA build and only to make sure that all the tests are fine

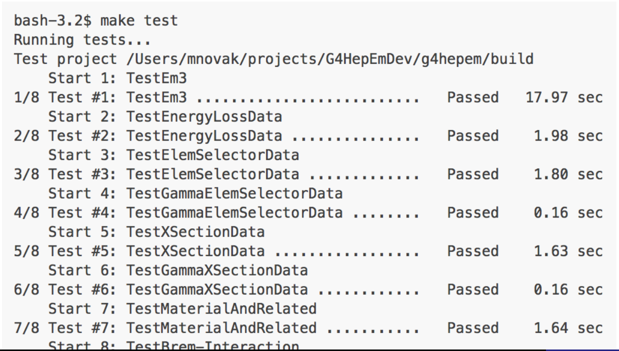


The G4HepEm R&D Project

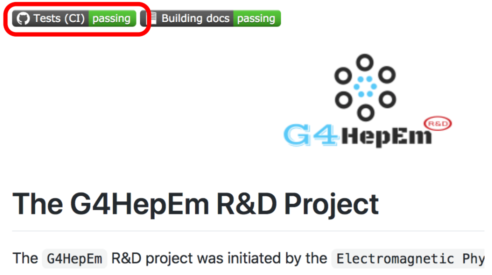
The `G4HepEm` R&D project was initiated by the `Electromagnetic Phy`

Development environment:

- at the moment, dedicated `GitHub` repository: **mnovak42/g4hepem**
- a minimum `Continuous Integration` testing is in place:
    - `Geant4-10.6.p03` on `Ubuntu-20.04` with `gcc-9.3`
    - no CUDA build and only to make sure that all the tests are fine
    - tests can be activated by the `-DBUILD_TESTING=ON` `CMake` configuration option (then `make test`)



```
bash-3.2$ make test
Running tests...
Test project /Users/mnovak/projects/G4HepEmDev/g4hepem/build
    Start 1: TestEm3
1/8 Test #1: TestEm3 ..........................   Passed   17.97 sec
    Start 2: TestEnergyLossData
2/8 Test #2: TestEnergyLossData ...............   Passed    1.98 sec
    Start 3: TestElemSelectorData
3/8 Test #3: TestElemSelectorData .............   Passed    1.80 sec
    Start 4: TestGammaElemSelectorData
4/8 Test #4: TestGammaElemSelectorData ........   Passed    0.16 sec
    Start 5: TestXSectionData
5/8 Test #5: TestXSectionData .................   Passed    1.63 sec
    Start 6: TestGammaXSectionData
6/8 Test #6: TestGammaXSectionData ............   Passed    0.16 sec
    Start 7: TestMaterialAndRelated
7/8 Test #7: TestMaterialAndRelated ...........   Passed    1.64 sec
    Start 8: TestBrem-Interaction
```



The G4HepEm R&D Project

The `G4HepEm` R&D project was initiated by the `Electromagnetic Phy`

Development environment:

- at the moment, dedicated **GitHub** repository: **mnovak42/g4hepem**
- a minimum **Continuous Integration** testing is in place:
  - Geant4-10.6.p03 on Ubuntu-20.04 with gcc-9.3
  - no CUDA build and only to make sure that all the tests are fine
  - tests can be activated by the -DBUILD_TESTING=ON CMake configuration option (then make test)
- documentation:
  - reStructuredText (for the in-depth documentation) and Doxygen (for the code documentation) combined by using Sphinx (and Breathe)
  - generated automatically after each merge to the master through a Read the Docs webhook

Tests (CI) passing Building docs passing

The G4HepEm R&D Project

Development environment:

- at the moment, dedicated **GitHub** repository: **mnovak42/g4hepem**
- a minimum **Continuous Integration** testing is in place:
  - Geant4-10.6.p03 on Ubuntu-20.04 with gcc-9.3
  - no CUDA build and only to make sure that all the tests are fine
  - tests can be activated by the -DBUILD_TESTING=ON CMake configuration option (then make test)
- documentation:
  - reStructuredText (for the in-depth documentation) and Doxygen (for the code documentation) combined by using Sphinx (and Breathe)
  - generated automatically after each merge to the master through a Read the Docs webhook
  - **available online** (i.e. html) as well as in pdf formats

Docs » The G4HepEm R&D Project Documentation

Edit on GitHub

The G4HepEm R&D Project Documentation

- Introduction
  - Description and goals

Contents

Timeline of `G4HepEm` (till the next evaluation)

April	complete e^-/e^+ physics by including (multiple) Coulomb scattering complete γ physics by providing the final version of photoelectric
May	Alternative stepping-loops, specialised for e^-/e^+ and γ transport, in `Geant4`
June	**Evaluation and further directions**
…	Complete physics: energy loss fluctuation, γ nuclear interaction, etc. Propagation in field Investigate possible extension(s) for (opportunistic) multi-particle computations ⋮

Contents

Summary:

- the ongoing `G4HepEm` **R**esearch & **D**evelopment activity of the `Geant4` **EM physics working group** has been presented in a nutshell
- the **motivations**, leading to this research activity, has been reviewed together with the ultimate **goals and some of the expected results**
- it has been shown, `G4HepEm` **offers the possibilities for exploring several further interesting R&D ideas** thanks to its design
- moreover, `G4HepEm` **provides support for** the related **R&D** activities **targeting EM shower simulation on GPU** devices
- sharing our expertise by making this development public, the **Geant4 EM physics working group**(the collaboration) tries to **facilitate and catalyse all the related R&D activities**