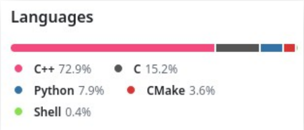
LHeC SW developments

LHeC IP SR study monthly meeting – August 8, 2024

Laurent Forthomme (AGH University of Kraków)

Latest software developments – July/August 2024

- **Disclaimer**: heavy (test beam) period, all developments presented here date back from last week…!
- **Reminder**: public github repository: https://github.com/forthommel/lhecsw
  - Originated from Peter's DD4hep fork, stripped LHeD geometry definition part from repository, and multiple geometry scenarii studied in the past



- **NEW**: interfacing to Key4hep, a Gaudi-based offline SW framework, built with a modular structure (“packages” or “packages/subpackages” definition)
  - Fully compatible with FCC-ee/FCC-hh/CEPC/… simulation/reconstruction tools, better handling of multithreading runs than “pure” DD4hep
  - Geometry definition & toolsets (surfaces definitions) still handled by DD4hep
  - Gaudi-translation of the few “producer” modules already mentioned for “single-config” event generation: “trivial” particle gun, BDSIM scorer planes output, Pythia 8, CepGen (Pythia 6 + Sherpa interfaces still in preparation)
  - Geant4 propagation into all sensitive volumes, currently relies on “stock” tracker/calorimeter hits collection production (flexible ; can be customised to detectors-specific collections storing additional Geant4 attributes for e.g. DIGI/RecHits conversion)
  - Reusing major data formats definition from Key4hep/EDM4hep (SimHits/(Rec)Hits/DIGIs/…), derivatives can be defined for LHeC detector-specific usages

Example steering file – Pythia 8 configuration (→ RecHits)

- **Python steering** of simulation/reconstruction jobs
  - Combination of standard Gaudi Configurables options and LHeC framework-specific includes (can be used to define a common/shared set of algorithms + parameterisation for future studies)
  - Attempt to automate **internal conversion** between transient data formats (e.g. Pythia 8 → HepMC3 → Gaudi)
  - **Output data model** definition from "standard" podio library, with ROOT (TTree/TNtuple) and SIO (SLAC) formats I/O management
- Implementation of first Geant4 SimHits → (Rec)Hits **conversion algorithms**
  - Currently handling a few algos w/ Geometry-sentient spatial/temporal resolution smearing for vertex/pixel trackers (SimAlgos/Tracker), and energy-smearing for calorimeters (SimAlgos/Calorimetry) ; more to follow
  - All collections of interest can be saved and reused for later stages of processing ; standard "producer/consumer" I/O structure

```python
from Gaudi.Configuration import *
from Configurables import GenAlg
from Configurables import PodioOutput, FCCDataSvc
from Configurables import ApplicationMgr

from Generator.pythia8Interface_cff import *
from Geometry.geoservice_cfi import geoservice
from SimG4.sim_cff import geantservice, geantsim
from SimAlgos.digi_cff import digis

pythia8.preInitCommands = [
    'Beams:idA = 2212',      # beam 1 = proton
    'Beams:idB = 11',        # beam 2 = electron
    'Beams:frameType = 2',   # beams are back-to-back, but with different energies
    'Beams:eA = 7000.',      # proton energy (GeV)
    'Beams:eB = 50.',        # electron energy (GeV)
    'PDF:lepton2gamma = on',
    'PhotonCollision:gmgm2mumu = on',
]
genalg = GenAlg("Pythia8", SignalProvider = pythia8)
genalg.hepmc.Path = "hepmc"

geantsim.eventProvider = pythia8Particles

out = PodioOutput("out",  # PODIO output algorithm
    outputCommands = ["keep *"],
    filename = "output.root",
    OutputLevel = DEBUG,
)
podioevent = FCCDataSvc("EventDataSvc")

ApplicationMgr(  # wrap everything together
    TopAlg = [
        genalg,
        pythia8HepMCConverter,
        Geantsim,
        *digis,
        out
    ],
    EvtSel = 'NONE',
    EvtMax   = 100,
    ExtSvc = [podioevent, geoservice, geantservice]
)
```

Key4hep/Gaudi-specific

LHeC-specific (conditions/algos)

$\gamma\gamma \rightarrow \mu^+\mu^-$

Updated geometry definition

Instead of one single DD4hep “compact” geometry definition file, split into several components-specific files

```
data/compact/
├── elements.xml
├── LHeD  <──────  scenario-specific folder
│    ├── Beampipe.xml
│    ├── Ecal.xml → (EcalBarrel.xml, EcalCryostat.xml, EcalPlugs.xml)
│    ├── Fields.xml
│    ├── Hcal.xml → (HcalBarrel.xml, HcalEndcaps.xml, HcalPlugs.xml)
│    ├── Magnets.xml
│    ├── materials.xml
│    ├── Muon.xml → (MuonBarrel.xml, MuonEndcaps.xml)
│    ├── SiTracker.xml → (SiTrackerBackward.xml, SiTrackerBarrel.xml, SiTrackerForward.xml, SiTrackerOBarrel.xml, SiTrackerReadout.xml, SiTrackerSupports.xml)
│    └── SiVertex.xml → (SiVertexBarrel.xml, SiVertexReadout.xml, SiVertexSupports.xml)
├── LHeD.xml  <──────  main definition file
└── materials.xml
```

Increased flexibility in selecting sub-detectors to be added/excluded from simulation geometry

- Drops potential code duplication in defining various geometry scenarii (e.g. symmetric/asymmetric designs)
- Still requires some polishing in the definition of constants (dimensions/distances/elements multiplicity/…), can be delegated to subdetector definition files

How many parallel scenarii to be maintained? Symmetric/asymmetric designs, or more?

Potential tasks/future developments

- Start investigating **tracking/vertexing algorithms** implementations
  - work recently ongoing on ATLAS' ACTS library porting to Key4hep environment: key4hep/k4ActsTracking (currently stalled, only DD4hep → ACTS geometry conversion recipe w/o reconstruction algorithms interfacing ; need to investigate whether some development branches are available somewhere?)
- Define a few **"standard candle"** resolution/efficiency **distributions** extracted in earlier attempts (e.g. CDR I/II)
  - In a first stage, can help validating the various approaches developed so far for each subdetector
  - Can live in a (CI-oriented) test/relvals infrastructure, w/ a few plots of interest helping future developers in all forthcoming algos/data formats/conditions implementations
  - May lead to potential new interfacing of MCs/simulation toolboxes ; may be ported to the Key4hep environment in a feedback loop
- Introduce some "translation units" between DD4hep-based geometry and **Delphes** fast simulation tool
  - E.g. a few resolution/acceptance extractors (→ Delphes TCL input) given a change of conditions/geometry scenario
- Start **versioning** the current LHeC SW stack, e.g. in the LHeC **CVMFS** area (potentially w/ automation of library maintenance through CI ; work in progress @ CERN GitLab: lforthom/lhecsw)
- Other, longer-term: Gaudi is not very "CMS"-friendly: depending on future developers community building around this, introduce some translator units (EDAnalyzer/EDProducer/…) and additional Python helpers for configurations?

Spares

LHeC detector simulation/reconstruction toolbox

Latest software developments – June 2024

- **Work in progress:** publication of a custom software stack in the LHeC CVMFS area (`/cvmfs/lhec.cern.ch`)
  - E.g. core SW (lhecsw, geometry definitions) + dependencies: BDSIM, e-p customised MC versions/plugins, …
  - Not to be used for datasets storage (→ `/eos/project/l/lhec`)
  - Working on a Docker image creation (through CI) + architecture/gcc version/…-dependent publication
  - Main benefits: unified, version'ed snapshots for samples generation + analysis, accessible all over WLCG

BDSIM interfacing

Currently provides a BDSIM interfacing tool (“TTree-reader”) delivering a simple Geant4 particle (electron-photon) gun

- fixed the event parentage issue reported in May, now allows for visualisation/propagation of radiation synchrotron into DD4hep/Geant4 model

- more developments can be done to directly interface BDSIM, might be overly complex for the current usage
  - development branch currently being worked on: forthommel:ext-bdsim_direct_interface
  - requiring minor adaptation from BDSIM output objects definition (avoiding in-between ROOT buffering stage through a collections/storage object with accessors)

(Old) Pythia 8 event builder

Pythia 8 event builder, with HepMC3/DD4hep input actions interfacing (vertices/particles parentage bookkeeping)

- allows Pythia fragments directly steered in the Python configuration snippets
- e.g. $\gamma\gamma \rightarrow \mu\mu$ **production in e-p** (σ ~ 1.013 nb) with scattered proton breakup and full event hadronisation/fragmentation →
- still a few "youth sicknesses" to be cured: issues with "intermediate" particle status codes, inducing non-blocking errors on check of parentages/parton virtualities/…

```
gen = DDG4.GeneratorAction(kernel, 'Geant4InputAction/Input')
gen.Input = 'Pythia8EventGenerator'
gen.OutputLevel = Output.DEBUG
gen.Parameters = dict(
    Commands = [
        'Beams:idA = 2212',     # beam 1 = proton
        'Beams:idB = 11',       # beam 2 = electron
        'Beams:frameType = 2',  # beams are back-to-back, but with different energies
        'Beams:eA = 7000.',     # proton energy (GeV)
        'Beams:eB = 50.',       # electron energy (GeV)
        'PDF:lepton2gamma = on',
        'PhotonCollision:gmgm2mumu = on',
    ],
)
geant4.buildInputStage([gen], output_level=Output.DEBUG)
```

- full event simulation with QGSP_BERT physics list: O(~25 s)/event (single thread), no tracking/reconstruction algorithms!
- still debugging the interface, might be a unit conversion problem in initial event feeding to DD4hep model (mm → cm/m/…, rad → mrad/…)

Pythia 8 event builder – visualisation tool