Science and Technology Facilities Council

Particle Physics

Python Example Update

Tim Adye

Rutherford Appleton Laboratory

ACTS Developers meeting

5th April 2022

PR summary

- Refactor Python examples so they can be chained together
  1. PR #1128 (8 Feb) feat: Python particle gun and Fatras examples can be chained [@timadye]
  2. PR #1174 (1 Apr) feat: remaining Python examples can be chained [@timadye]

- At this stage, maintain exact compatibility for callers and Python tests:
  - `$ ROOT_HASH_CHECKS=on pytest`
    - Caveats:
      - fixed a few small bugs in `Examples/Python/tests/test_examples.py`, but no changes in references
      - renamed file listed in `Examples/Python/tests/root_file_hashes.txt` which was inconsistent between tests
- Plan a separate small PR tidying up some remaining complications
  - here will introduce changes in `test_examples.py` and `root_file_hashes.txt` comparisons

Main changes

- Most changes in `Examples/Scripts/Python` with new `add*` function, called by old `run*` function:
  - particle_gun.py : `runParticleGun` → addParticleGun
  - pythia8.py : `runPythia8` → addPythia8
  - fatras.py : `runFatras` → addFatras
  - digitization.py : `configureDigitization` → addDigitization
  - seeding.py : `runSeeding` → addSeeding
  - ckf_tracks.py : `runCKFTracks` → addCKFTracks
  - vertex_fitting.py : `runVertexFitting` → addVertexFitting
  - itk.py : new options `buildITkGeometry(jsonconfig, logLevel)`
  - common.py : `common.addPythia8` now just a wrapper for pythia8.addPythia8

  - full_chain_itk.py : most basic example of the use of the above…

Example: ITk full-chain

full_chain_itk.py

```python
#!/usr/bin/env python3
import pathlib, acts, acts.examples, itk

u = acts.UnitConstants
geo_dir = pathlib.Path("acts-detector-examples")
outputDir = pathlib.Path.cwd()

detector, trackingGeometry, decorators = itk.buildITkGeometry(geo_dir)
field = acts.ConstantBField(acts.Vector3(0.0, 0.0, 2.0 * u.T))
rnd = acts.examples.RandomNumbers(seed=42)

from particle_gun import addParticleGun, MomentumConfig, EtaConfig, ParticleConfig
from fatras import addFatras
from digitization import addDigitization
from seeding import addSeeding, SeedingAlgorithm, TruthSeedRanges, ParticleSmearingSigmas
from ckf_tracks import addCKFTracks

s = acts.examples.Sequencer(events=100, numThreads=-1)
s = addParticleGun(
    s,
    MomentumConfig(1.0 * u.GeV, 10.0 * u.GeV, transverse=True),
    EtaConfig(-4.0, 4.0, uniform=True),
    ParticleConfig(num=1, pdg=acts.PdgParticle.eMuon, randomizeCharge=True),
    rnd=rnd,
)
s = addFatras(
    s,
    trackingGeometry,
    field,
    outputDirRoot=outputDir,
    rnd=rnd,
)
```

use `addPythia8(...)` for physics events, e.g. ttbar + <μ>=200

All defaults come from C++. Can also default with `None`

```python
s = addDigitization(
    s,
    trackingGeometry,
    field,
    digiConfigFile=geo_dir / "atlas/itk-hgtd/itk-smearing-config.json",
    outputDirRoot=outputDir,
    rnd=rnd,
)
s = addSeeding(
    s,
    trackingGeometry,
    field,
  # SeedingAlgorithm.TruthSmeared, ParticleSmearingSigmas(pRel=0.01), rnd=rnd,
  # SeedingAlgorithm.TruthEstimated,
    TruthSeedRanges(pt=(1.0 * u.GeV, None), eta=(-4.0, 4.0), nHits=(9, None)),
    geoSelectionConfigFile=geo_dir / "atlas/itk-hgtd/geoSelection-ITk.json",
    outputDirRoot=outputDir,
)
s = addCKFTracks(
    s,
    trackingGeometry,
    field,
    TruthSeedRanges(pt=(400.0 * u.MeV, None), nHits=(6, None)),
    outputDirRoot=outputDir,
)

s.run()
```

select alternative seeding algorithm

Implementation details

- Arguments such as `MomentumConfig(min=1.0 * u.GeV, max=10.0 * u.GeV, transverse=True)` are defined as namedtuple
  - they are actually keyword arguments, but can also be specified without keyword, which can be inferred from the type, or as a tuple e.g.
    - `addParticleGun(MomentumConfig(1.0 * u.GeV, 10.0 * u.GeV, transverse=True),…)` or
    - `addParticleGun(momentumConfig=(1.0 * u.GeV, 10.0 * u.GeV, True),…)`
  - this is sorted out by the `@acts.examples.NamedTypeArgs` decorator

- C++ objects are configured with unspecified (or `None`) options removed, so they can take their C++ defaults
  - use `**acts.examples.defaultKWArgs(arg=val,…)` to automatically remove `None`s (and `tuples` of `None`)

- A useful debugging tool is to print all the `acts.examples` C++ call arguments
  - enabled with `acts.logging.DEBUG` log level

What next?

- Tidy up oddities introduced for exact consistency
- Separate documentation
  - currently just in function docstring
- More full-chain examples
  - `full_chain_generic.py` already implemented in `ckf_tracks.py`
  - OpenDataDetector
- More Python examples to use these functions / style?
- Separate `add*` functions into modules of their own for a clear interface?
  - at the moment the `add*` functions are examples in their own right
  - might then be useful to allow user configuration/access of individual algorithms constructed by `add*` functions
    - maybe using some `Sequencer` interrogation?