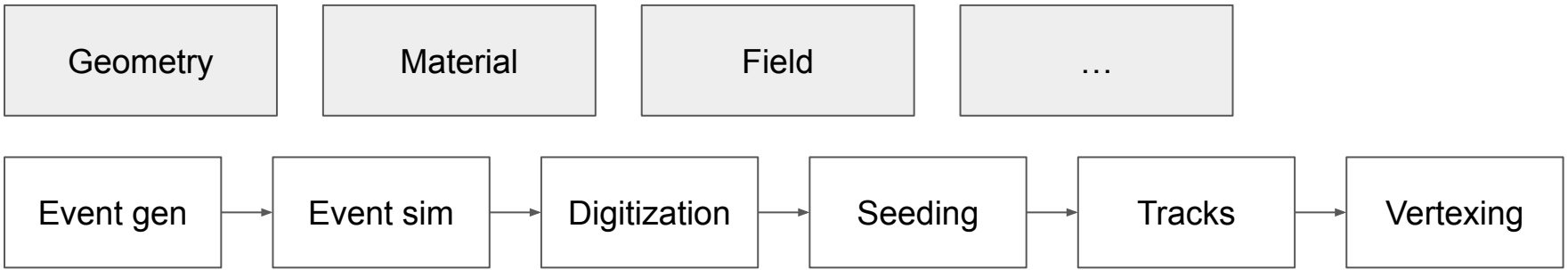
Tutorials: Full chain

ACTS Workshop 2022

Overview

- Full chain = simulation + reconstruction
- Useful for validation, testing
  - *The* integration test for tracking
  - Used in ACTS CI to catch regressions early
- Plug and play for different geometries and algorithms



Which full chain?

- ODD full chain (Examples/Scripts/Python/full_chain_odd.py)
- ITk full chain (Examples/Scripts/Python/full_chain_itk.py)
- physmon full chain (CI/physmon/physmon.py)
- ODD CI full chain (ci/full_chain_odd.py)
- Workshop full chain (full_chain/full_chain_odd.py)

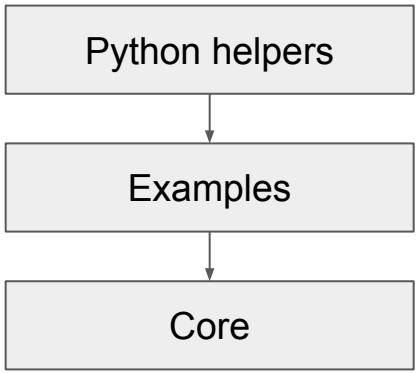
Setup

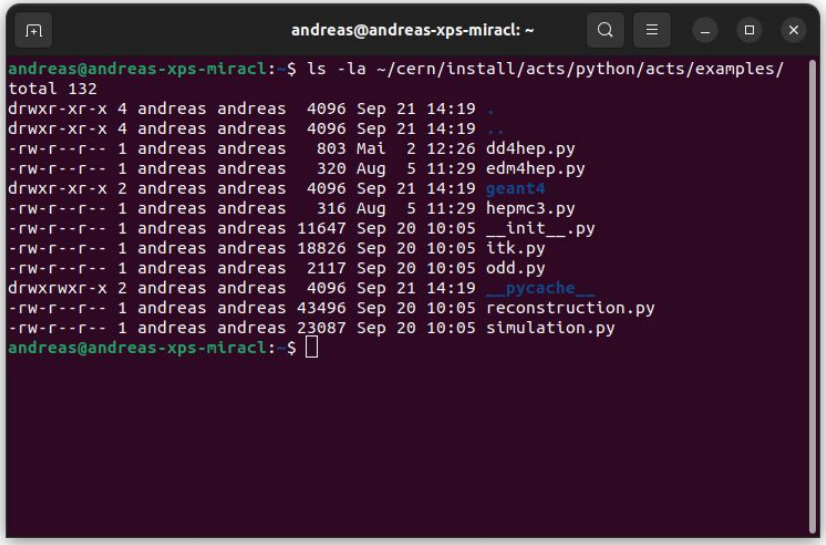
- I had hoped for using the CI docker images but didn’t work because of authentication problems
- Instead I rely on my local setup which will not be very reproducible to install
- My setup
  - Ubuntu 22.04.01
  - Root 6.20.06
  - DD4hep 1.21
  - ACTS master 48a155c
- Alternative from Paul: ACTS on lxplus (CentOS 7)

```
$ cmake -S . -B build_key4hep -GNinja \
    -DACTS_BUILD_PLUGIN_DD4HEP=ON \
    -DACTS_BUILD_EXAMPLES=ON \
    -DACTS_BUILD_EXAMPLES_PYTHON_BINDINGS=ON \
    -DACTS_BUILD_EXAMPLES_DD4HEP=ON \
    -DACTS_BUILD_ODD=ON
```

Python examples

- Are on top of the Python bindings, Example framework, Core
- Small Python library with helper functions (Examples/Python/python/acts/examples/)
  - Since recently ACTS will install some Python modules (CMakeLists.txt)
- simulation.py and reconstruction.py
  - Capture different steps/algorithms of the chain
  - Patter: addSomething(sequencer, ...)





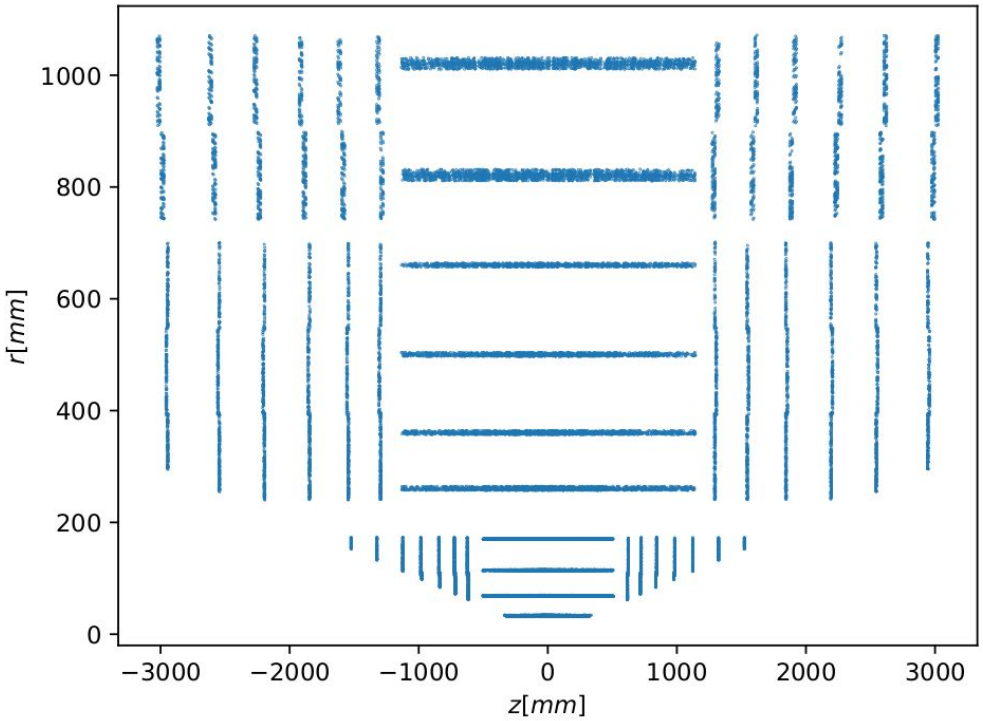
Python helpers

```python
def addParticleGun(
    s: acts.examples.Sequencer,
    outputDirCsv: Optional[Union[Path, str]] = None,
    outputDirRoot: Optional[Union[Path, str]] = None,
    momentumConfig: MomentumConfig = MomentumConfig(),
    etaConfig: EtaConfig = EtaConfig(),
    phiConfig: PhiConfig = PhiConfig(),
    particleConfig: ParticleConfig = ParticleConfig(),
    multiplicity: int = 1,
    vtxGen: Optional[EventGenerator.VertexGenerator] = None,
    printParticles: bool = False,
    rnd: Optional[RandomNumbers] = None,
    logLevel: Optional[acts.logging.Level] = None,
) -> None:
    customLogLevel = acts.examples.defaultLogging(s, logLevel)

    # Preliminaries
    rnd = rnd or RandomNumbers(seed=228)

    # Input
    evGen = EventGenerator(...)

    s.addReader(evGen)

    if printParticles:
        s.addAlgorithm(
            ParticlesPrinter(
                level=customLogLevel(),
                inputParticles=evGen.config.outputParticles,
            )
        )
```

```python
    if outputDirCsv is not None:
        outputDirCsv = Path(outputDirCsv)
        if not outputDirCsv.exists():
            outputDirCsv.mkdir()

        s.addWriter(
            CsvParticleWriter(
                level=customLogLevel(),
                inputParticles=evGen.config.outputParticles,
                outputDir=str(outputDirCsv),
                outputStem="particles",
            )
        )

    if outputDirRoot is not None:
        outputDirRoot = Path(outputDirRoot)
        if not outputDirRoot.exists():
            outputDirRoot.mkdir()

        s.addWriter(
            RootParticleWriter(
                level=customLogLevel(),
                inputParticles=evGen.config.outputParticles,
                filePath=str(outputDirRoot / "particles.root"),
            )
        )

    return s
```

OpenDataDetector

- Constructed via DD4hep
- Can be found on Gitlab
- Design
  - Full silicon
  - 4 pixel layers + 7 endcaps
  - 4 short strip layers + 6 endcaps
  - 2 long strip layers + 6 endcaps



Full chain

```python
detector, trackingGeometry, decorators = getOpenDataDetector(
    geoDir, mdecorator=oddMaterialDeco
)
field = acts.ConstantBField(acts.Vector3(0.0, 0.0, 2.0 * u.T))
rnd = acts.examples.RandomNumbers(seed=42)

s = acts.examples.Sequencer(events=100, numThreads=-1, outputDir=str(outputDir))

addParticleGun(
    s,
    MomentumConfig(1.0 * u.GeV, 10.0 * u.GeV, transverse=True),
    EtaConfig(-3.0, 3.0, uniform=True),
    ParticleConfig(2, acts.PdgParticle.eMuon, randomizeCharge=True),
    rnd=rnd,
)

addFatras(
    s,
    trackingGeometry,
    field,
    rnd=rnd,
    outputDirRoot=outputDir,
)

addDigitization(
    s,
    trackingGeometry,
    field,
    digiConfigFile=oddDigiConfig,
    rnd=rnd,
    outputDirRoot=outputDir,
)

addSeeding(
    s,
    trackingGeometry,
    field,
    geoSelectionConfigFile=oddSeedingSel,
    outputDirRoot=outputDir,
)

addCKFTracks(
    s,
    trackingGeometry,
    field,
    CKFPerformanceConfig(ptMin=1.0 * u.GeV, nMeasurementsMin=6),
    outputDirRoot=outputDir,
)

addVertexFitting(
    s,
    field,
    TrackSelectorRanges(pt=(1.0 * u.GeV, None), absEta=(None, 3.0), removeNeutral=True),
    vertexFinder=VertexFinder.Iterative,
    trajectories="trajectories",
    outputDirRoot=outputDir,
)

s.run()
```

Initialisation

- Units alias
- Geometry paths
- Output directory
- Geometry config
- Algorithm config
- Geometry construction
- Magnetic field
- Random numbers

```
22  u = acts.UnitConstants
23  geoDir = getOpenDataDetectorDirectory()
24  outputDir = pathlib.Path.cwd() / "odd_output"
25
26  oddMaterialMap = geoDir / "data/odd-material-maps.root"
27  oddDigiConfig = geoDir / "config/odd-digi-smearing-config.json"
28  oddSeedingSel = geoDir / "config/odd-seeding-config.json"
29  oddMaterialDeco = acts.IMaterialDecorator.fromFile(oddMaterialMap)
30
31  detector, trackingGeometry, decorators = getOpenDataDetector(
32      geoDir, mdecorator=oddMaterialDeco
33  )
34  field = acts.ConstantBField(acts.Vector3(0.0, 0.0, 2.0 * u.T))
35  rnd = acts.examples.RandomNumbers(seed=42)
```

Sequencer

- Stears the event processing
- Events can be processed in parallel because they are independent
- Algorithms, Reader, Writer are added to the Sequencer
- Sequencer calls Reader, Algorithms, Writer in order for each event

```
37    s = acts.examples.Sequencer(events=100, numThreads=-1, outputDir=str(outputDir))

89    s.run()
```

```
20:18:56    Sequencer       INFO      Added reader 'EventGenerator'
20:18:56    Sequencer       INFO      Added algorithm 'ParticleSelector'
20:18:56    Sequencer       INFO      Added algorithm 'FatrasSimulation'
20:18:56    Sequencer       INFO      Added writer 'RootParticleWriter'
20:18:56    Sequencer       INFO      Added writer 'RootParticleWriter'
20:18:56    Sequencer       INFO      Added writer 'RootSimHitWriter'
20:18:56    Sequencer       INFO      Added algorithm 'DigitizationAlgorithm'
20:18:56    Sequencer       INFO      Added writer 'RootMeasurementWriter'
20:18:56    Sequencer       INFO      Added algorithm 'TruthSeedSelector'
```

Event generation

- Particle gun (alternative: Pythia8)

```
39  addParticleGun(
40      s,
41      MomentumConfig(1.0 * u.GeV, 10.0 * u.GeV, transverse=True),
42      EtaConfig(-3.0, 3.0, uniform=True),
43      ParticleConfig(2, acts.PdgParticle.eMuon, randomizeCharge=True),
44      rnd=rnd,
45  )
```

```
1  particle_id,particle_type,process,vx,vy,vz,vt,px,py,pz,m,q
2  4503599644147712,13,0,0,0,0,0,-4.42938805,5.41367865,1.74514735,0.105658367,-1
3  4503599660924928,-13,0,0,0,0,0,2.69452524,1.82695782,-13.3078556,0.105658367,1
```

Fatras and digitization

- Fast track simulation (alternative: Geant4)
- Hit smearing (alternative: geometric pixel activation)

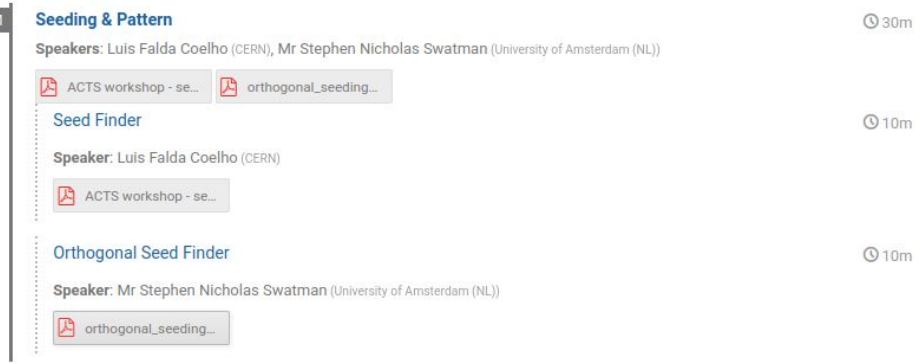
```
50  addFatras(
51      s,
52      trackingGeometry,
53      field,
54      rnd=rnd,
55      outputDirRoot=outputDir,
56  )
```

```
58  addDigitization(
59      s,
60      trackingGeometry,
61      field,
62      digiConfigFile=oddDigiConfig,
63      rnd=rnd,
64      outputDirRoot=outputDir,
65  )
```

Seeding

- For more details about the seeding: Monday session
- (alternative: truth tracking)

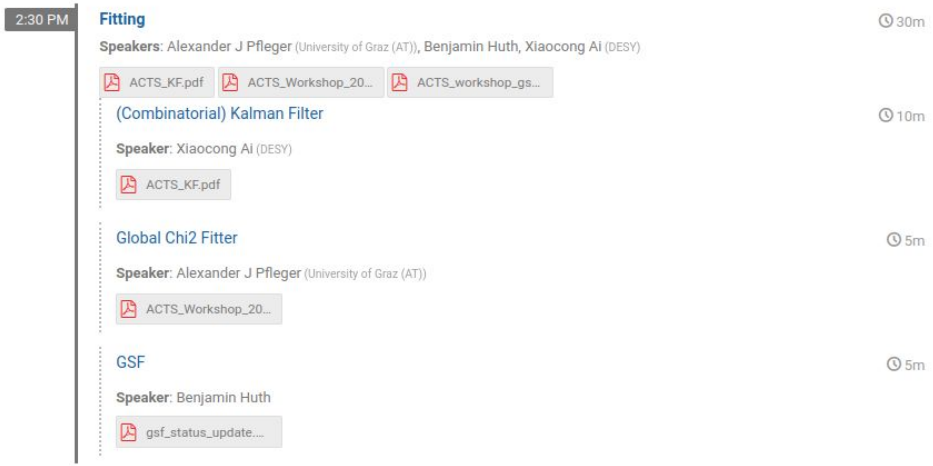
```
67  addSeeding(
68      s,
69      trackingGeometry,
70      field,
71      geoSelectionConfigFile=oddSeedingSel,
72      outputDirRoot=outputDir,
73  )
```



Track finding

- For more details about the track finding/fitting: Monday session

```
addCKFTracks(
    s,
    trackingGeometry,
    field,
    CKFPerformanceConfig(ptMin=1.0 * u.GeV, nMeasurementsMin=6),
    outputDirRoot=outputDir,
)
```



Vertexing

- For more details about the vertexing: Monday session
- (alternative: AMVF)

```
83  addVertexFitting(
84      s,
85      field,
86      TrackSelectorRanges(pt=(1.0 * u.GeV, None), absEta=(None, 3.0), removeNeutral=True),
87      vertexFinder=VertexFinder.Iterative,
88      trajectories="trajectories",
89      outputDirRoot=outputDir,
90  )
```

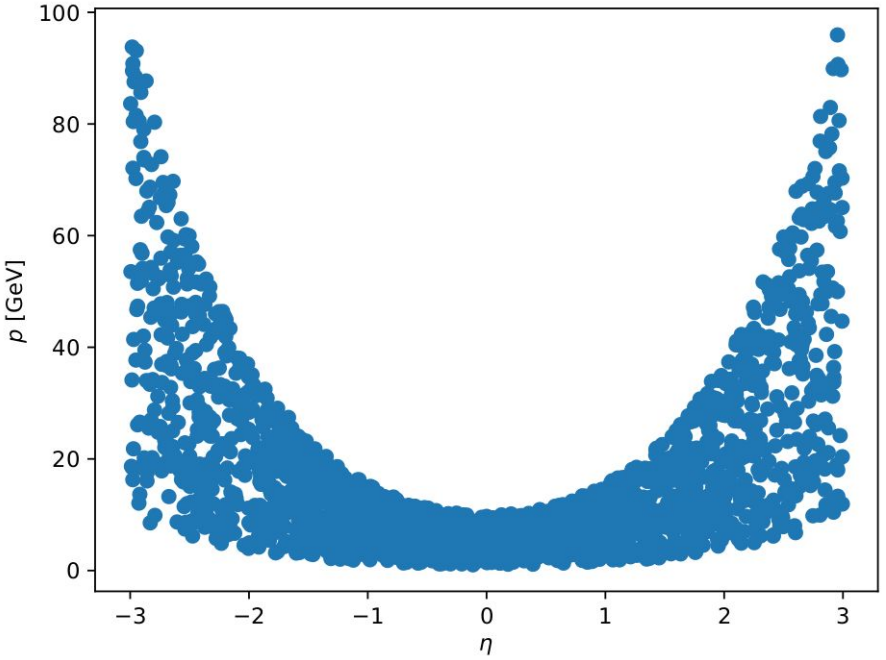
Resources

- https://github.com/andiwand/acts-workshop-2022
- https://acts.readthedocs.io/
- https://github.com/acts-project/acts
- https://indico.cern.ch/event/1184037
- https://codimd.web.cern.ch/A108z_6tRiWJaIa5yAabdg

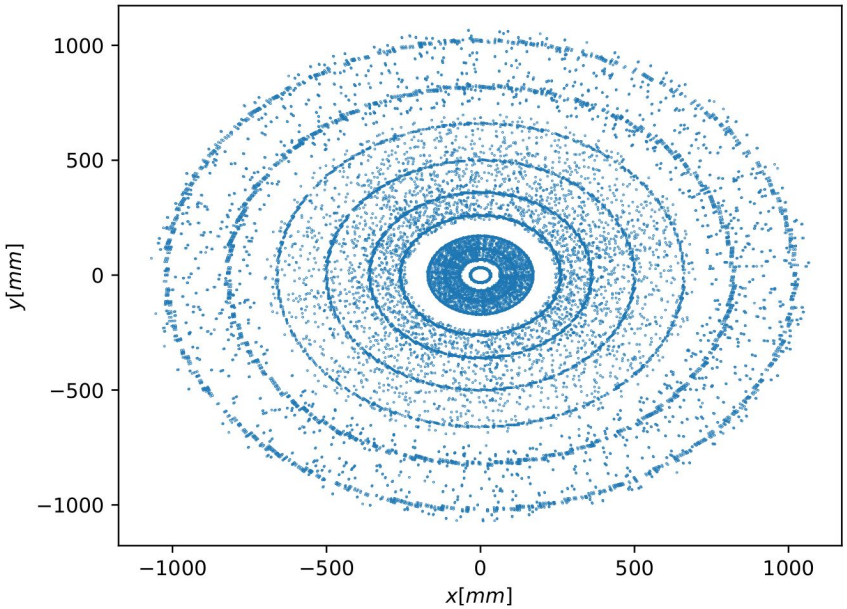
Backup

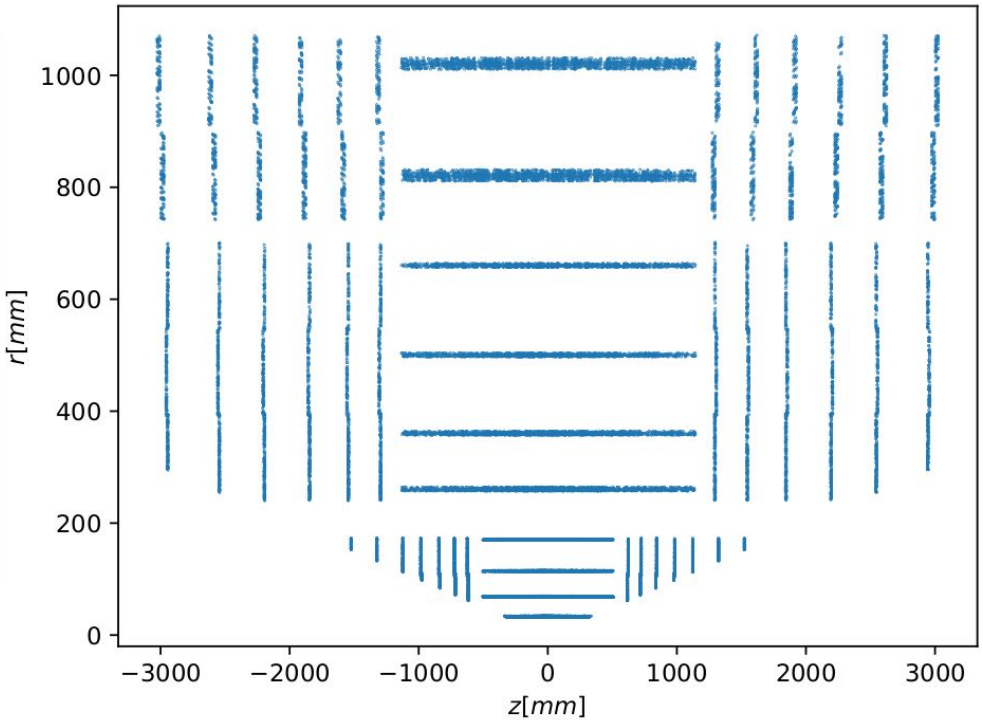
Particle gun

```
39  addParticleGun(
40      s,
41      MomentumConfig(1.0 * u.GeV, 10.0 * u.GeV, transverse=True),
42      EtaConfig(-3.0, 3.0, uniform=True),
43      ParticleConfig(2, acts.PdgParticle.eMuon, randomizeCharge=True),
44      rnd=rnd,
45  )
```



Fatras


1000
500
0
−500
−1000
y[mm]
−1000
−500
0
500
1000
x[mm]


1000
800
600
400
200
0
r[mm]
−3000
−2000
−1000
0
1000
2000
3000
z[mm]

Vertexing

```
addVertexFitting(
    s,
    field,
    TrackSelectorRanges(pt=(1.0 * u.GeV, None), absEta=(None, 3.0), removeNeutral=True),
    vertexFinder=VertexFinder.Iterative,
    trajectories="trajectories",
    outputDirRoot=outputDir,
)
```