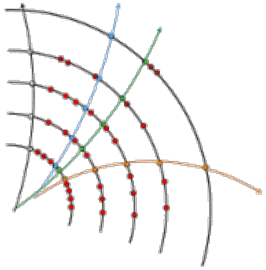


Status of ACTS at CEPC

ZhangJin

2021-03-08

中国科学院高能物理研究所
Institute of High Energy Physics
Chinese Academy of Sciences

Outline

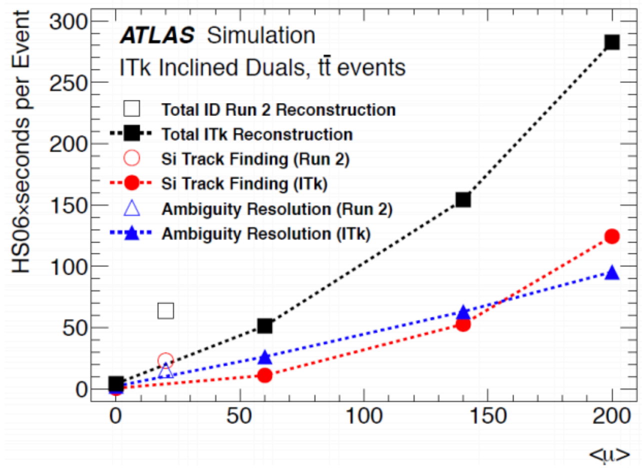
- A review of ACTS
- CEPC ACTS activities
  - Studies in Acts standalone framework
  - Integration to CEPC Core Software
- Summary and Next

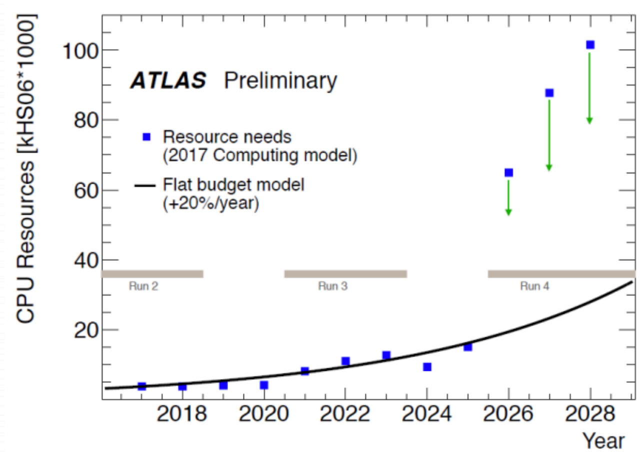
A review of ACTS - ACTS Motivation

- LHC Run-1/2 exceeded all expectations in terms of provided data
  - Design pile-up ~21 for Run-1 and ~40 for Run-2
  - Track reconstruction worked extremely well

- HL-LHC will bring great challenges to computing in track reconstruction

	LHC Run-1	LHC Run-2	LHC Run-4
muon	21	40	150-200
Tracks	~280	~600	~7-10k





Keep physics performance && Tackle computing resource problem for future LHC era

A review of ACTS : A Common Tracking Software

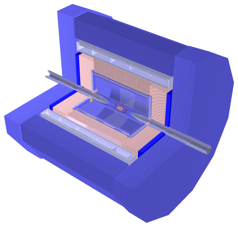
- Derived from ATLAS, driven by the core idea to become **A C**ommon **T**racking **S**oftware
  - Encapsulating the well-tested ATLAS tracking code – high performance in the past
  - Independent from detectors and framework

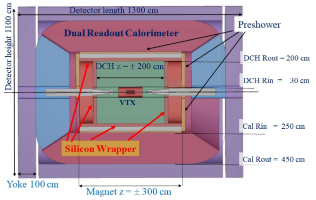
- Modern technologies
  - Deal with the CPU problem in dense tracking environment
  - Generic programming with C++17
  - Thread-safety design and efficient memory allocation

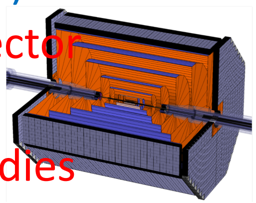
- Active group for the developing
  - Potential to become the future ATLAS tracking software
  - Other experiments are also trying
    - BELLE-2, sPHENIX, FASER, CEPC ... *

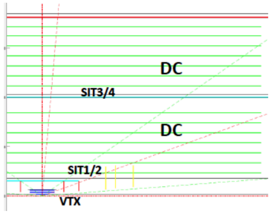
CEPC Tracking System and Requirements

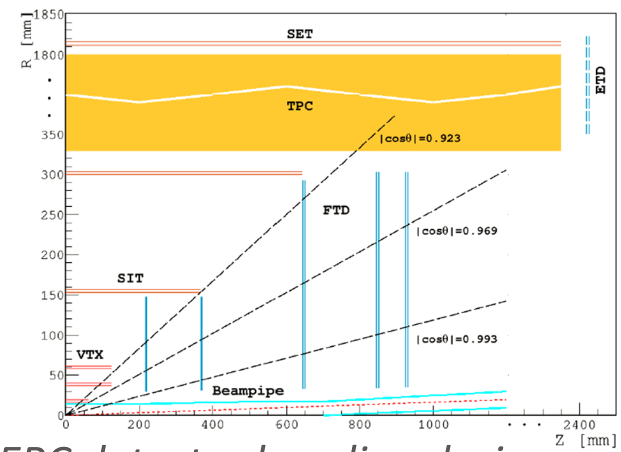
- Three CEPC detector concepts
  - Baseline detector (silicon + TPC)
  - Full silicon detector
    - FST2
    - T2 reference detector (silicon + drift chamber)
- Requirement of an accurate and efficient tools for detector studies
  - Flexibility in layout optimizations and material studies
  - Evaluating the performances of different designs
  - With the potential of becoming the future tracking software



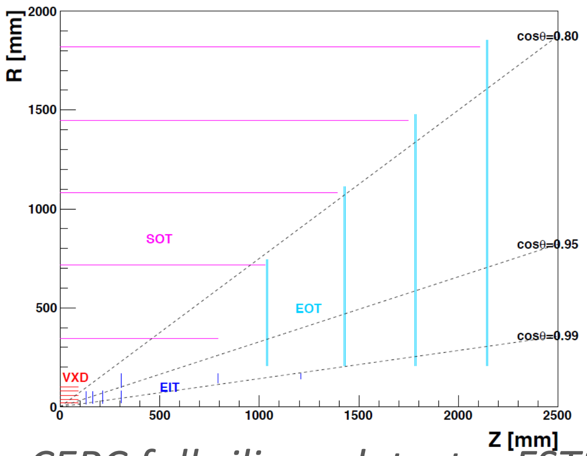








CEPC detector baseline design



CEPC full silicon detector FST2

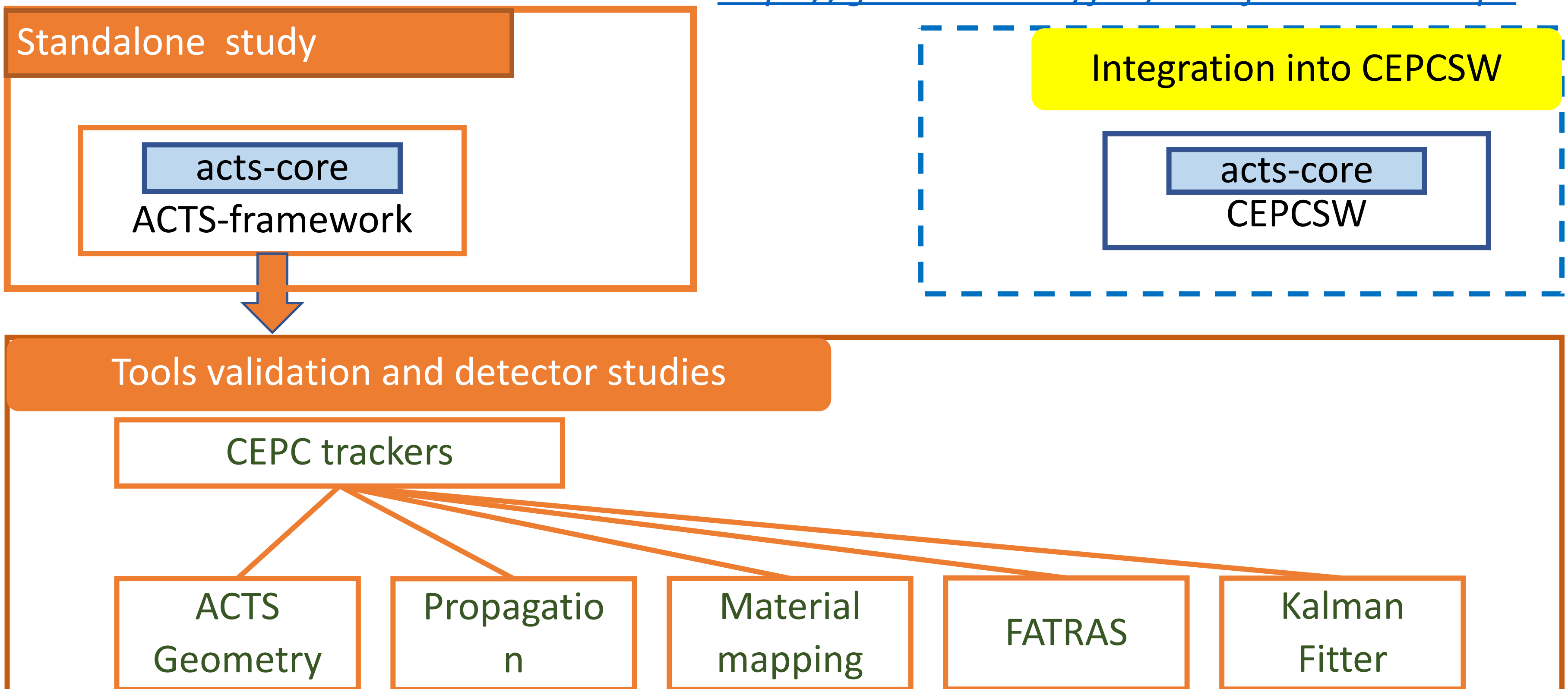
Activities and Motivations

- Take part in ACTS development
  - Gaussian Sum Filter developing
  - Gitlab Code reviewing
- Detector and Algorithm studies in the standalone framework
  - Validation tracking tools and fully understand the details
- Integration to CEPC Core software
  - Important to test the Algorithm, IO, EDM interfaces

Previously : studies in the Standalone framework

Detector studies at standalone framework

https://gitlab.cern.ch/jinz/acts-framework-cepc



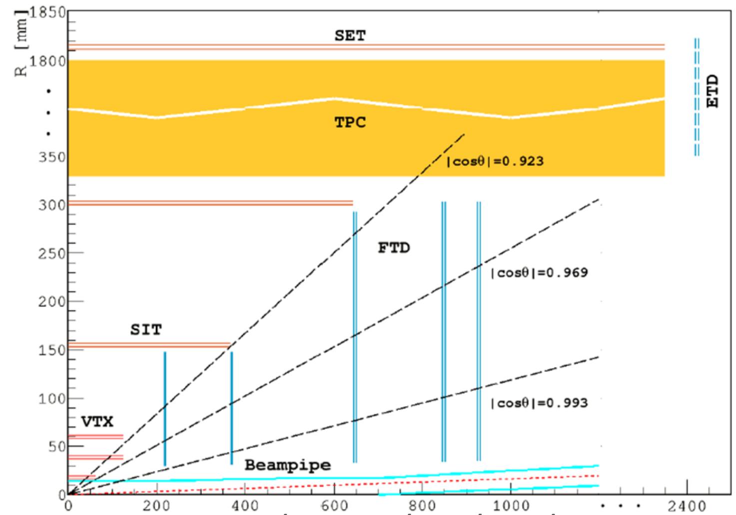
- ➢ Two designs of CEPC tracking system implemented with DD4HEP (Baseline+FST2)
- ➢ Baseline (silicon + TPC)
  - ➢ Silicon detector with material
  - ➢ Simplified TPC with 220 sensitive surfaces
  - ➢ Validated within this standalone ACTS-framework
- ➢ Full silicon detector (FST2 in CDR)
  - ➢ Geometry and materials under validating

Implementation

ACTS Geometry | Propagation | Material mapping | FATRAS | Kalman Fitter

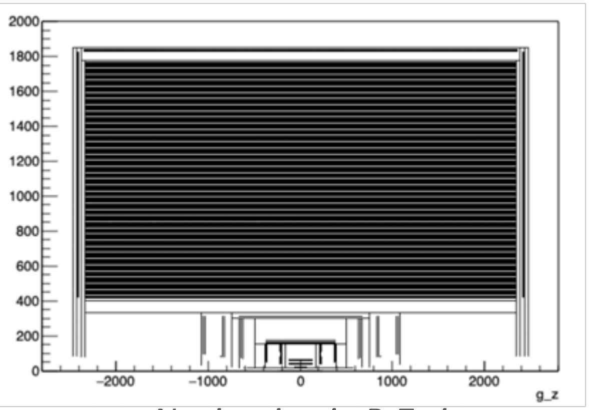
Baseline tracker

- DD4hep based geometry to describe CEPC inner tracker and built with XML file
  - Flexible to modify the detector parameters
  - Good readability
  - Easy to integrate to CEPCSW
  - May become one of the standards in the future

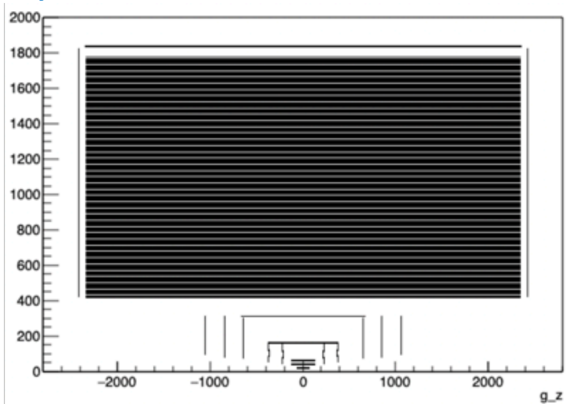


CEPC detector baseline design

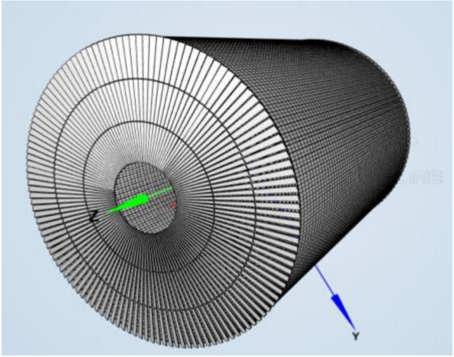
- Propagation
  - A powerful tool to debug the tracking geometry is correctly built



Navigation in R-Z view



Sensitive in R-Z view


SET & ETD

Implementation

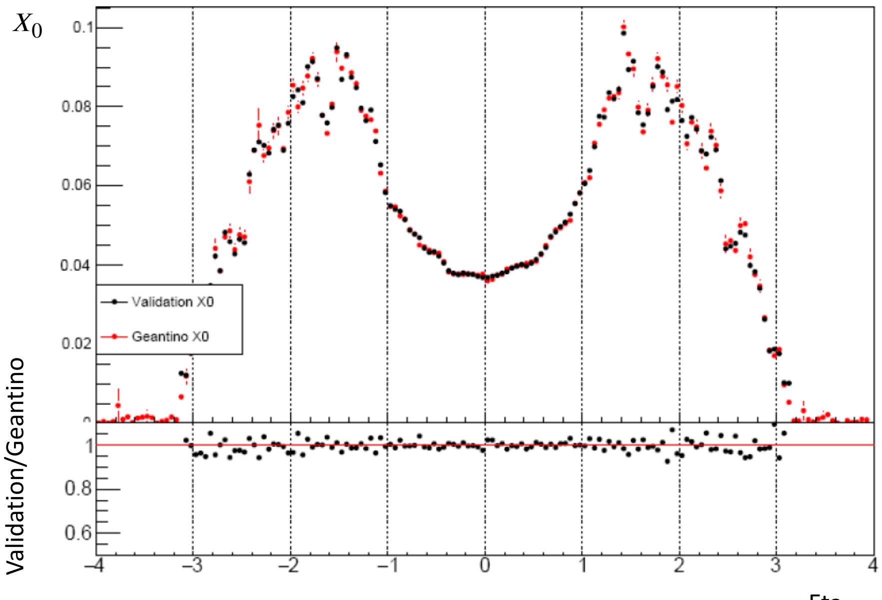
Baseline tracker

➢ Material mapping

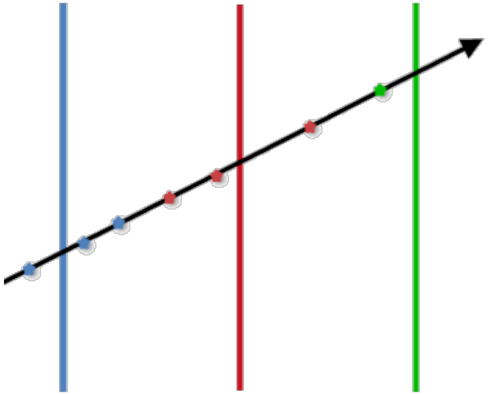
project complex material onto tracking geometry

- Details of Material in the DD4hep xml
- Geantino to record the original material
- Original material is mapped to surfaces – json ouput


```
"volumes": {
    "14": {
        "Geoid": "[  14 |   0 |   2 |   0 |    0 ]",
        "Name": "",
        "layers": {
            "2": {
                "Geoid": "[  14 |   0 |   2 |   0 |    0 ]",
                "representing": {
                    "bin0": [
                        "binPhi",
                        "closed",
                        1,
                        [
                            -3.1415927410125732,
                            3.1415927410125732
                        ]
                    ],
                    "bin1": [
                        "binR",
                        "open",
                        25,
                        [
                            70.0999984741211,
                            300.9956970214844
                        ]
                    ],
```

Validation of X_0



Surface mapping

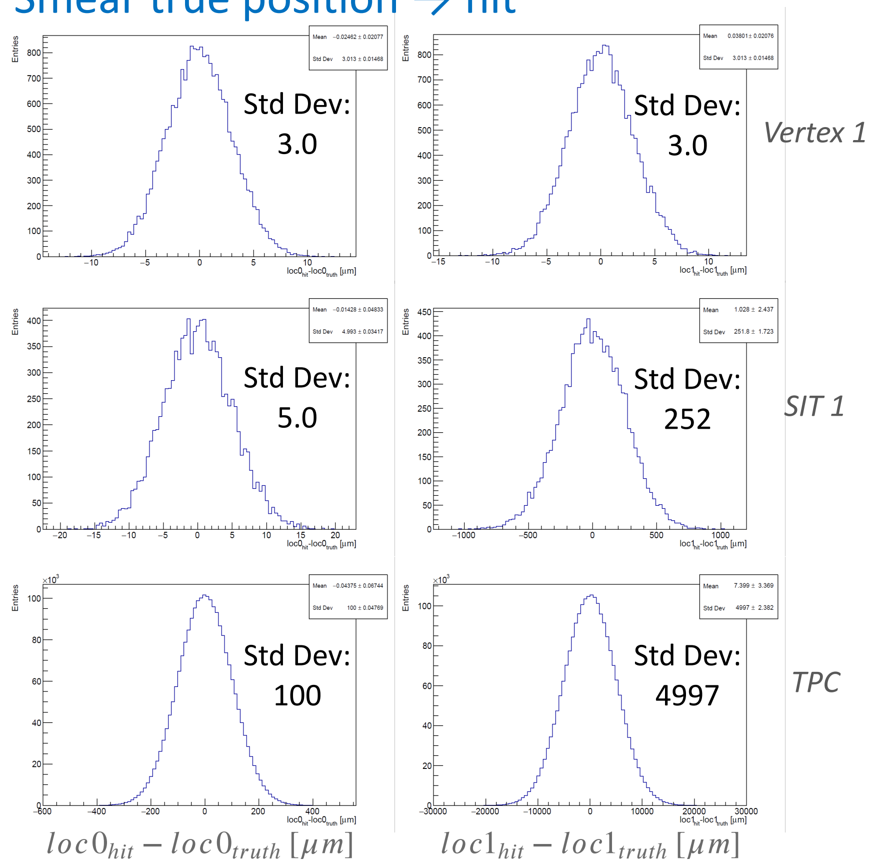
Generally match with Geant4 output. The simplified material distribution is consistent with the actual material.

Implementation

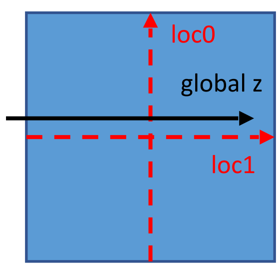
- Resolutions of sub-detectors in CDR

Sub-detector			loc0_res [μm]	loc1_res [μm]	
Barrel	Vertex	1	3	3	pixel
		2	4	4	pixel
		3	4	4	pixel
	SIT 1, 2		5	250	strip
	TPC		100	5000	TPC
	SET		5	250	strip
Endcap	FTD 1, 2		3	3	pixel
	FTD 3, 4, 5		5	250	strip
	ETD		5	250	strip

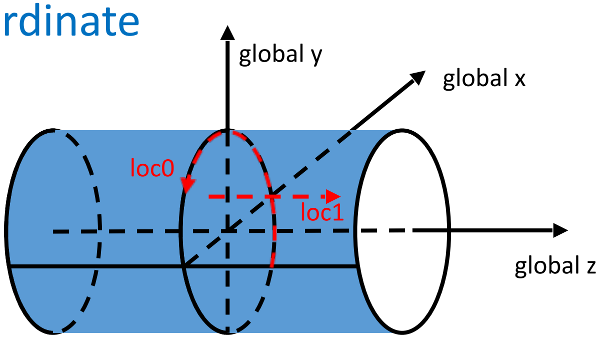
- FATRAS (Fast ATLAS Track Simulation) to do the simulation
  - Smear true position → hit



- Local coordinate



Silicon module



TPC layer

Fast Simulation results are correct

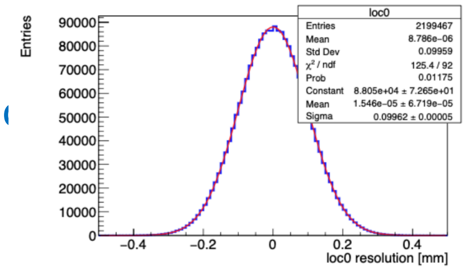
Performance study

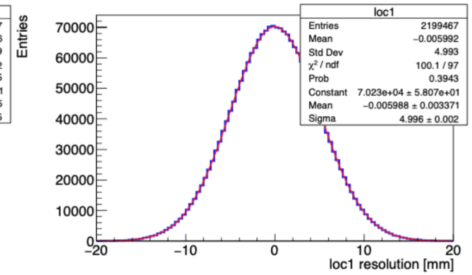
ACTS Geometry | Propagation | Material mapping | FATRAS | Kalman Fitter

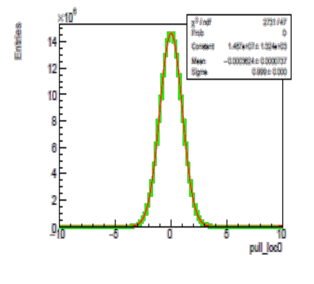
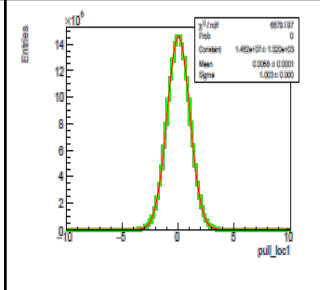
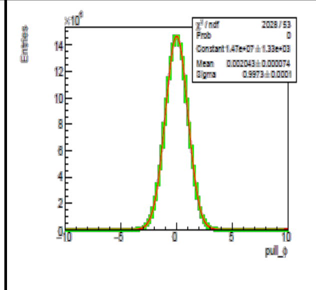
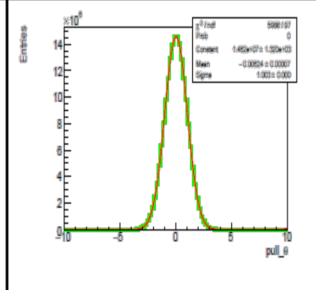
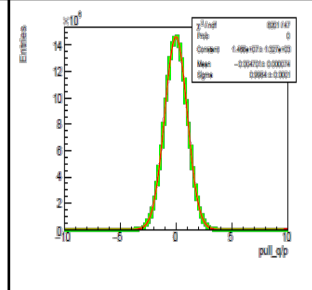
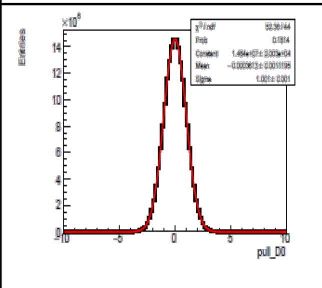
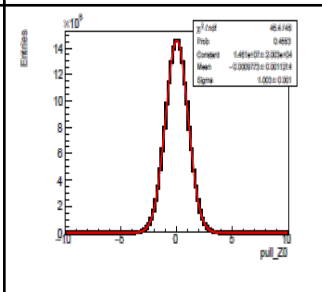
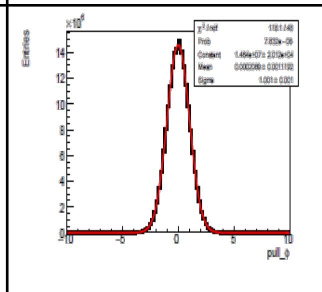
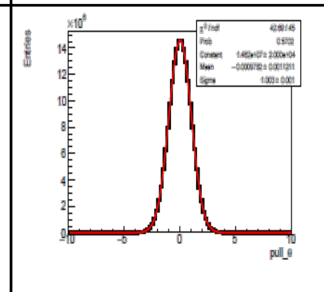
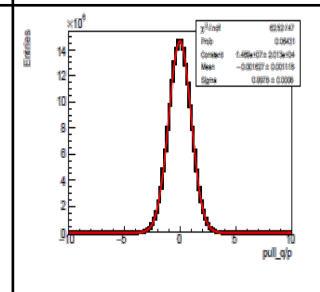
Baseline tracker

- FATRAS
  - Particle gun: 800,000 single μ^- from (0, 0, 0)
  - Magnetic field: (0, 0, 3T)
  - p_T: 100GeV, θ: 85°, φ: uniform distributi
- Kalman Filtering
  - Pull distribution of track parameters

TPC measurement Errors





	Loc0 or D0	Loc1 or Z0	phi	theta	q/p
Smooth					
Perigee					

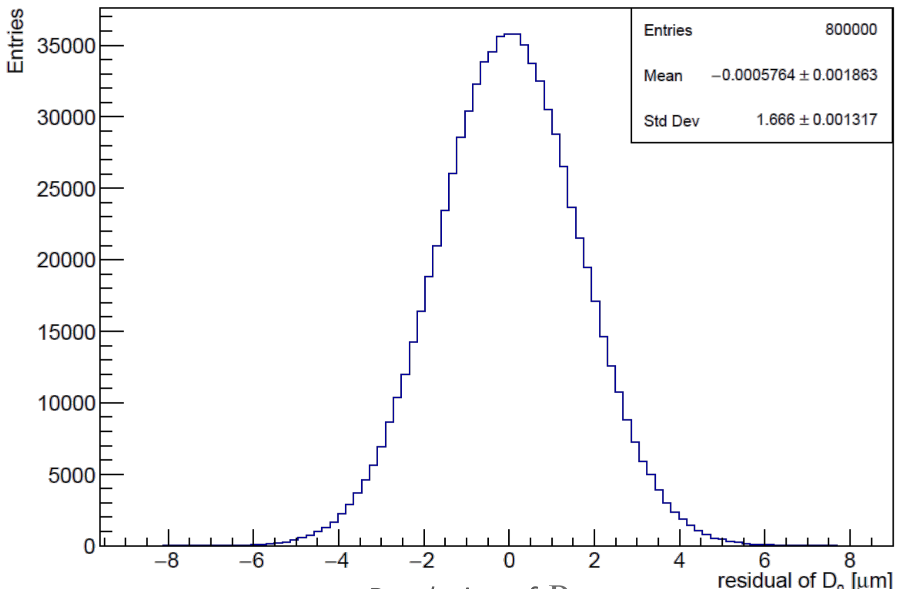
- Fitting results are convincible： All Fitting States follows standard normal distribution; measurement errors are reasonable

Performance study

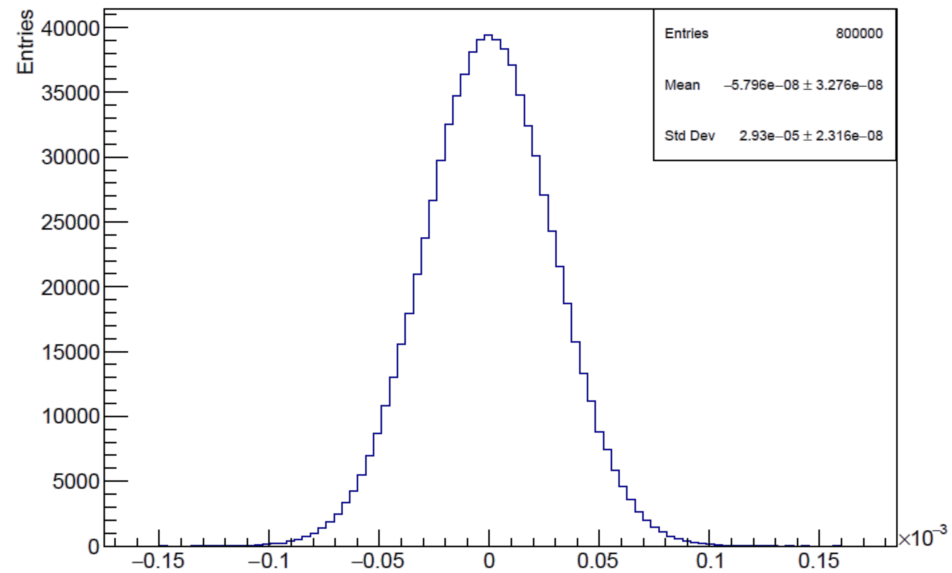
ACTS Geometry | Propagation | Material mapping | FATRAS | Kalman Fitter

Baseline tracker

➢ Resolution of vertex and momentum



Resolution of D_0



Resolution of $1/p_T$

- ➢ Result (p_T: 100GeV, θ: 85°)
  - ➢ $\sigma_{r\varphi} = 1.67\ \mu m$
  - ➢ $\sigma_{1/p_T} = 2.93 \times 10^{-5}\ c/GeV$

- ➢ Full simulation resolution in CDR
  - ➢ $\sigma_{r\varphi} = 1.89\ \mu m$
  - ➢ $\sigma_{1/p_T} = 2.75 \times 10^{-5}\ c/GeV$

Performance study

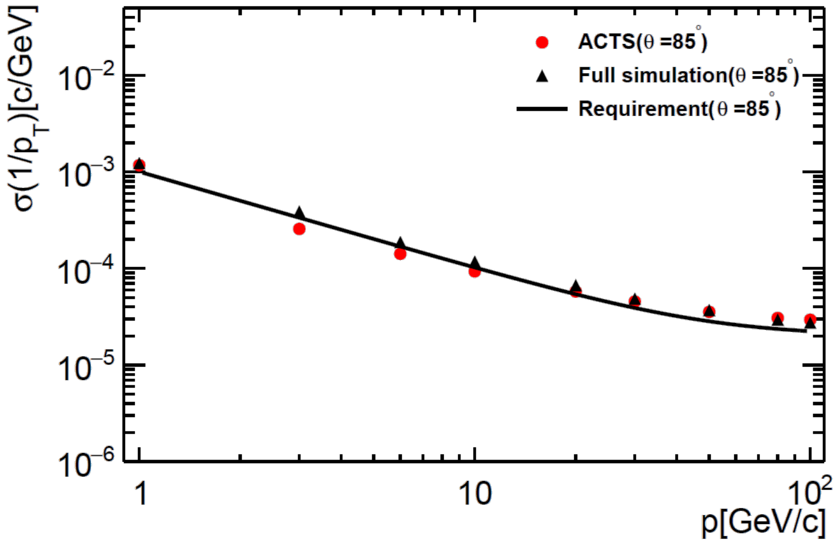
ACTS Geometry | Propagation | Material mapping | FATRAS | Kalman Fitter

Baseline tracker

- Resolution of vertex and momentum
  - Full simulation data are according to CDR
  - The CEPC physics program requires

- $\sigma_{1/p_T} = a \oplus \frac{b}{p sin^{3/2}\theta}$, $a \sim 2 \times 10^{-5} c/GeV$ and $b \sim 1 \times 10^{-3}$

Resolution of $1/p_T$

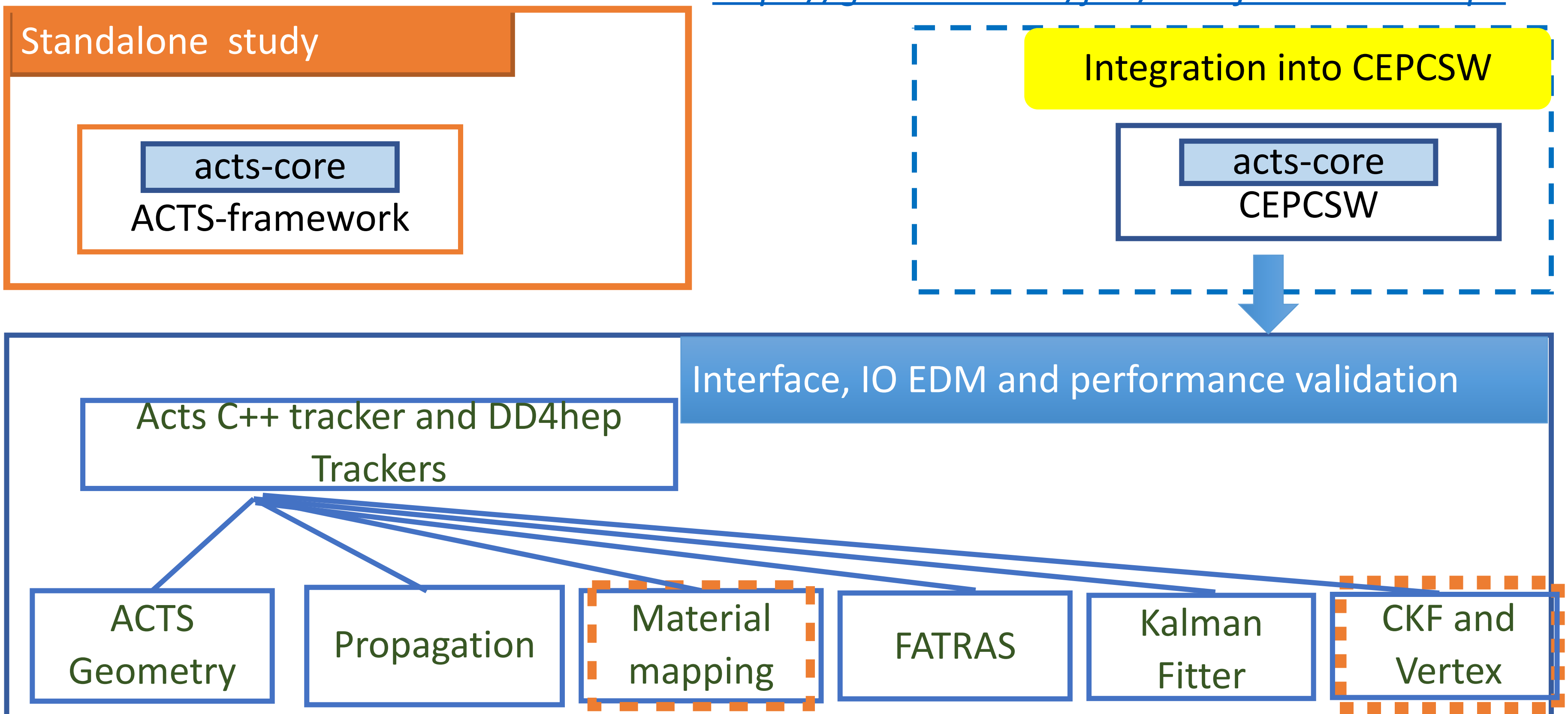


- Generally match with full simulations in CDR

Presently : From standalone framework to the CEPC core software

From Standalone framework to the CEPC core software

https://gitlab.cern.ch/jinz/acts-framework-cepc



- Start to do ACTS - CEPCSW integration - **CEPCSW key4hep Env, ACTS v5.0.0, lxslc7**
  - Focus on Algorithm Integration and performances at the first step
  - Most Key Algorithms are integrate to CEPCSW (Material mapping is not ready)
  - To be validated with the results in standalone framework

Geometry Building Tools Integration

- Building 3 basic detectors *CEPCSW/Examples/options/GenericActs.py*
  - Generic Detector - to check geometry tools and building procedure are correct
  - DD4hep Detector *CEPCSW/Examples/options/DD4hepActs.py*
    - Demonstrator - a simple silicon layer to check dd4hep geometry building and acts extension
    - FullSilicon detector - tracking performance validation and comparing

- Acts Geometry constructed correctly

- Json Writer is available
  - Using Json to write out geometry and material

```
//write Json
JsonSurfacesWriter::Config sJsonWriterConfig;
sJsonWriterConfig.trackingGeometry = m_trackingGeometry;
sJsonWriterConfig.writePerEvent = true;
auto sJsonWriter = std::make_shared<JsonSurfacesWriter>(
                  sJsonWriterConfig, logLevel);
// Write the tracking geometry object
sJsonWriter->write();
```


```
{
  "acts-geometry-hierarchy-map": {
    "format-version": 0,
    "value-identifier": "surfaces"
  },
  "entries": [
    {
      "layer": 2,
      "sensitive": 1,
      "value": {
        "bounds": {
          "type": "RectangleBounds",
          "values": [
            -24.000000000000004,
            -24.000000000000004,
            24.000000000000004,
            24.000000000000004
          ]
        },
        "geo_id": 360288107628593153,
        "transform": {
          "rotation": [
            0.0,
            -0.38388499999363634,
            0.9233809109895469,
            -2.7755575615628914e-17,
            0.9233809109895476,
            0.3838849999936361,
            -0.9999999999999998,
            -2.350617682414441e-17,
            0.0
          ],
"detector.json" [noeol] 296L, 7251C
```


The purpose of these 2 two detectors are to validate the Tracking Algorithms and tools in the Standalone framework

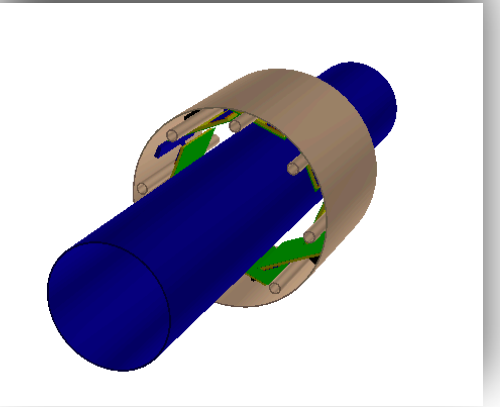
Geometry Building Tools Integration

- Demonstrator extension

```
[zhangjin@lxslc705 Demon]$ ls -R
.:
CMakeLists.txt  compact  src

./compact:
Demonstrator.xml  elements.xml  materials.xml

./src:
DemonstratorBarrel_geo.cpp  DemonstratorBeamPipe_geo.cpp
```



- FullSilicon detector - tracking performance validation and comparing

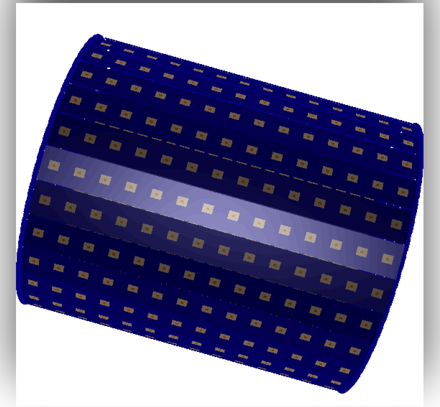
```
[zhangjin@lxslc705 FullSilicon]$ ls -R
.:
CMakeLists.txt  compact  src

./compact:
cepc  cepc_FST2.xml

./compact/cepc:
CEPC_elements.xml   cepc_Beampipe.xml  cepc_EIT_EOT.xml  cepc_VXD_SOT.xml
CEPC_materials.xml  cepc_Display.xml   cepc_IDs.xml      cepc_readouts.xml

./src:
CepcDetector

./src/CepcDetector:
CEPC_Common.cpp  CEPC_TPC_barrel.cpp  CEPC_assambleHelper.hpp  CEPC_beampipe.cpp  CEPC_layouthelper.hpp  CEPC_service.hpp
```



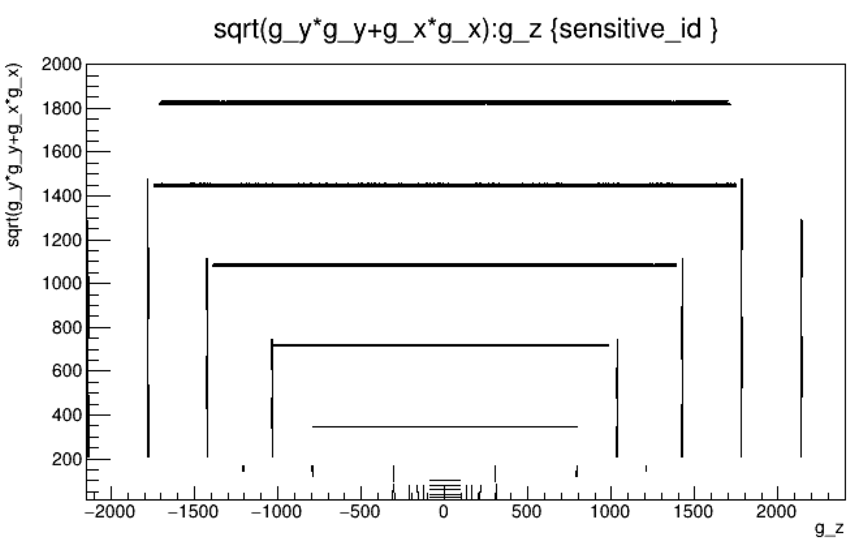
Acts Tracing geometry constructed correctly in CEPCSW

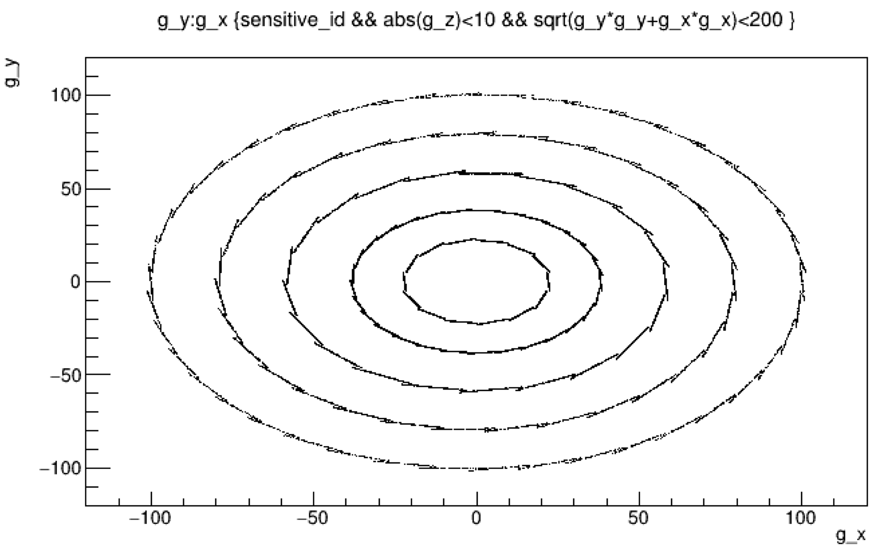
Material Json writer is to be implemented

Propagation Integration

Example : Examples/options/Fullsilicon.py

- RandomSeed to Generate “tracks”
- Propagation tool to extrapolate tracks in FST2 Detector and record sensitive/material detectors
- Root output of all sensitive/material positions and steps



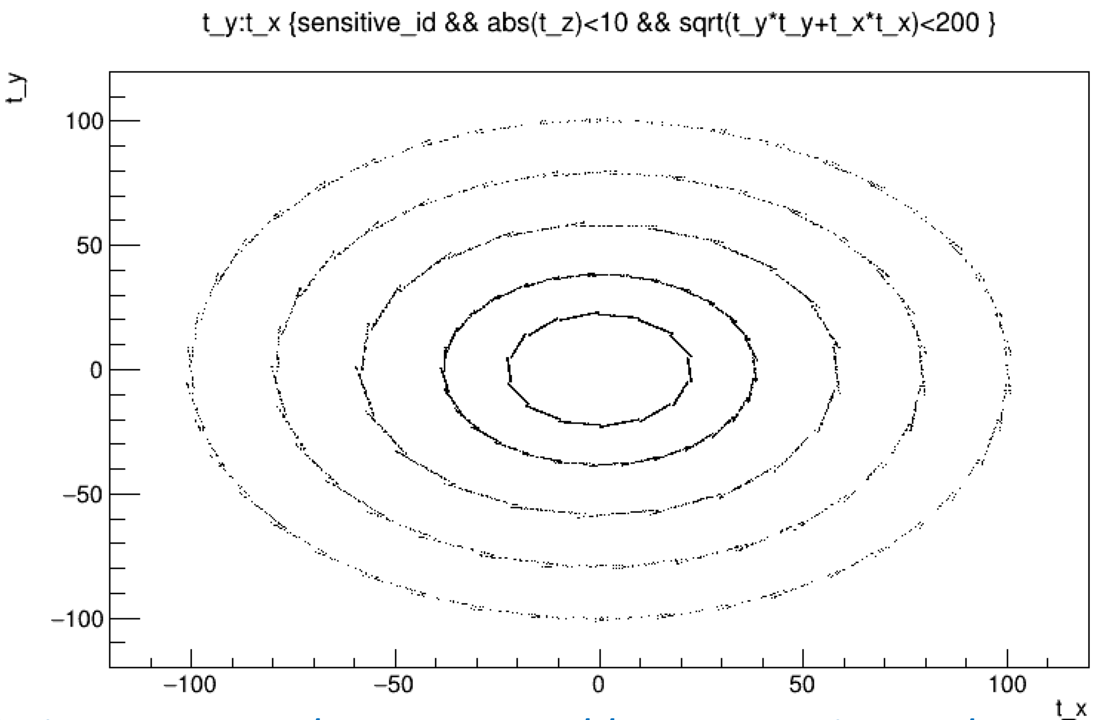


FTRAS (Fast Simulation) Integration

Example : Examples/options/DD4hepActsFatras.py

- GtGunTool as Generator, PodioOutput root file
- Read "MCParticle" from PodioInput root file

- Record all simulated particles and hits



Preliminary output shows reasonable Propagation and FastSim Algorithm

Need more detailed studies and comparisons

Kalman Filtering Integration

Example : Examples/options/DD4hepActsKalman.py

- Fatras as input
- HitSmearing, TruthTrack Finding, Particle Smearing
  - Currently write these simple functions to focus on Kalman fitting Algorithm

```
SimParticleContainer particlesInitial;
SimParticleContainer particlesFinal;

//Fast simulation
Fatras(particles,simHits,particlesInitial,particlesFinal);
sourceLinks.reserve(simHits.size());
measurements.reserve(simHits.size());
hitParticlesMap.reserve(simHits.size());
hitSimHitsMap.reserve(simHits.size());

//Hit smearing
hitSmearing(particles, simHits, sourceLinks, measurements, hitParticlesMap, hitSimHitsMap);
//Truth Track finding
TruthTrack(particles, simHits, sourceLinks, measurements, hitParticlesMap, hitSimHitsMap);
//Particle smearing
ParticleSmearing(particles, parameters);

sortSurface(tracks);
//KalmanFilter Fitting
Fitting(m_trackingGeometry,tracks);
```

Fitting Algorithm is available to run, Root output of track performance is in progress

We will check the fitting performances and compare it within the standalone framework

Summary and Next

Summary

- Acts Tracking Tools are validated in the standalone framework and show reasonable results from previously studies
- Start to Integrate ACTS to CEPCSW
- Preliminarily several key Algorithms is available, i.e., Geometry, Propagation, FastSim, KalmanFilter Fitting

Next

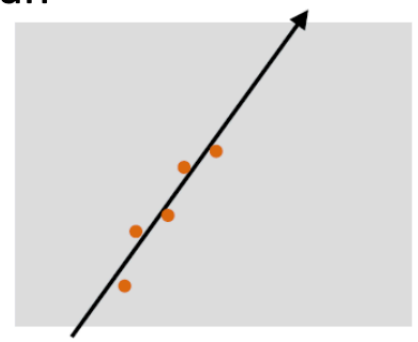
- Comparing the results with standalone framework
- Some codes/polices need to be modified

BACKUP

Gaussian Sum Filter

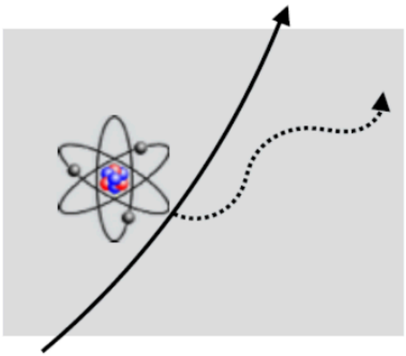
Kalman Filter : linearized filter allows all experiment noise is gaussian distributed

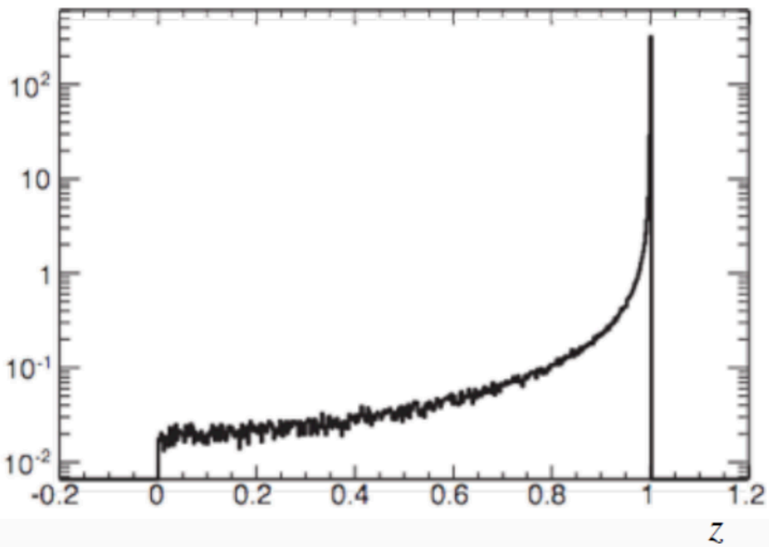
- Measurement errors – usually can be controlled
- Multiple scattering – a small gaussian tails
- Ionization loss – Landau distributed, fortunately dE<<E



The electron reconstruction is significant and difficult

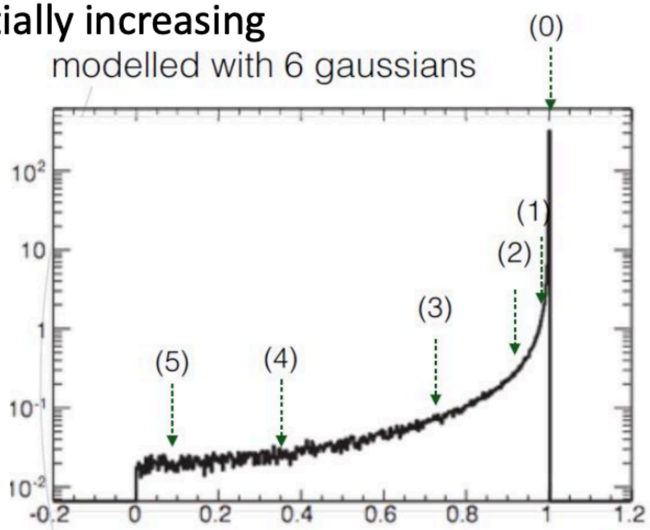
Energy loss is a bremsstrahlung effect -> strongly non gaussian

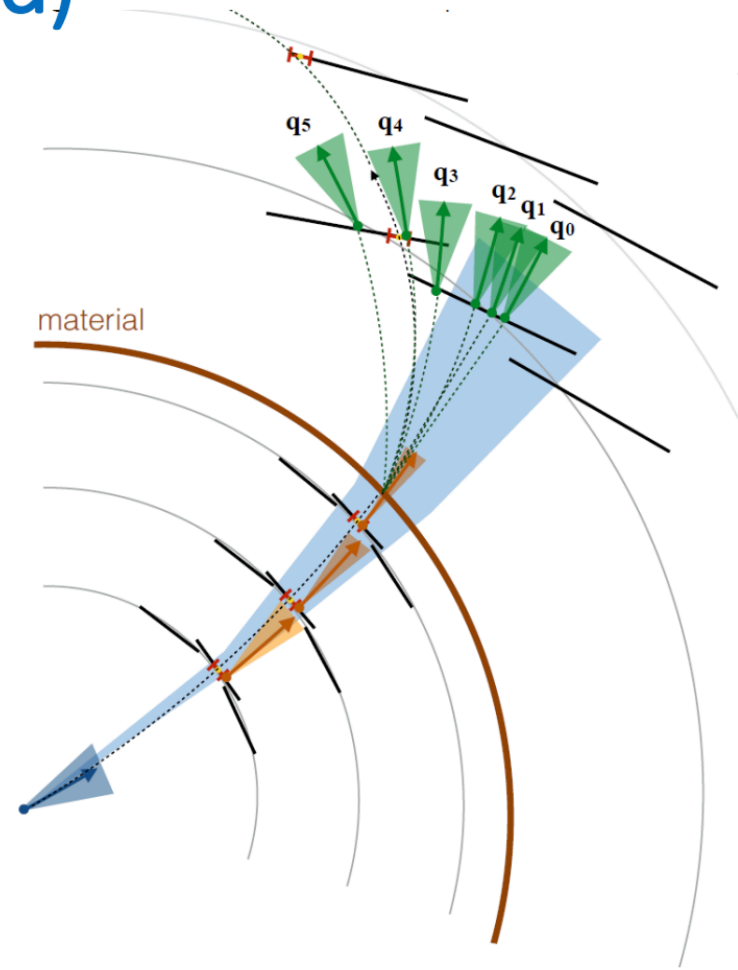




Gaussian Sum Filter(cont'd)

- Electron reconstruction are well handled with Gaussian Sum Filter, which is a parallel sets of Kalman Filter
- The bremsstrahlung energy loss distribution can be approximated as a weighted sum of gaussian components
- Each component behaves like a Kalman component, **propagate** individually
- Components should be merged to avoid the exponentially increasing

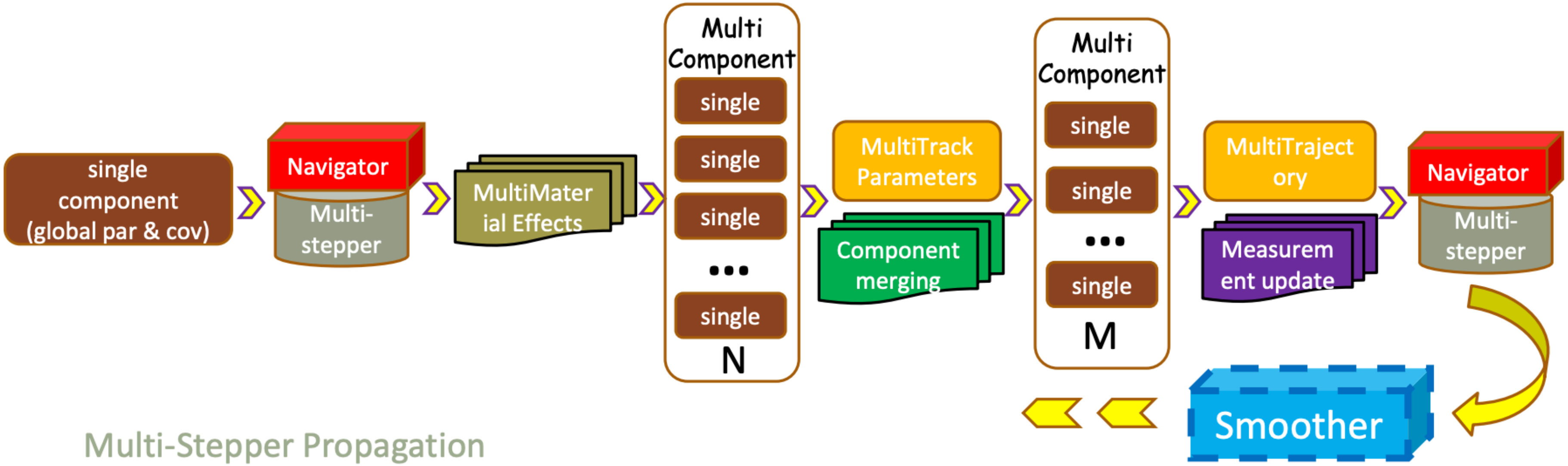




One component splits into 6 components

The (mean/cov/weight) of each component taken from ATLAS at the first step

Gaussian Sum Filter: implementation



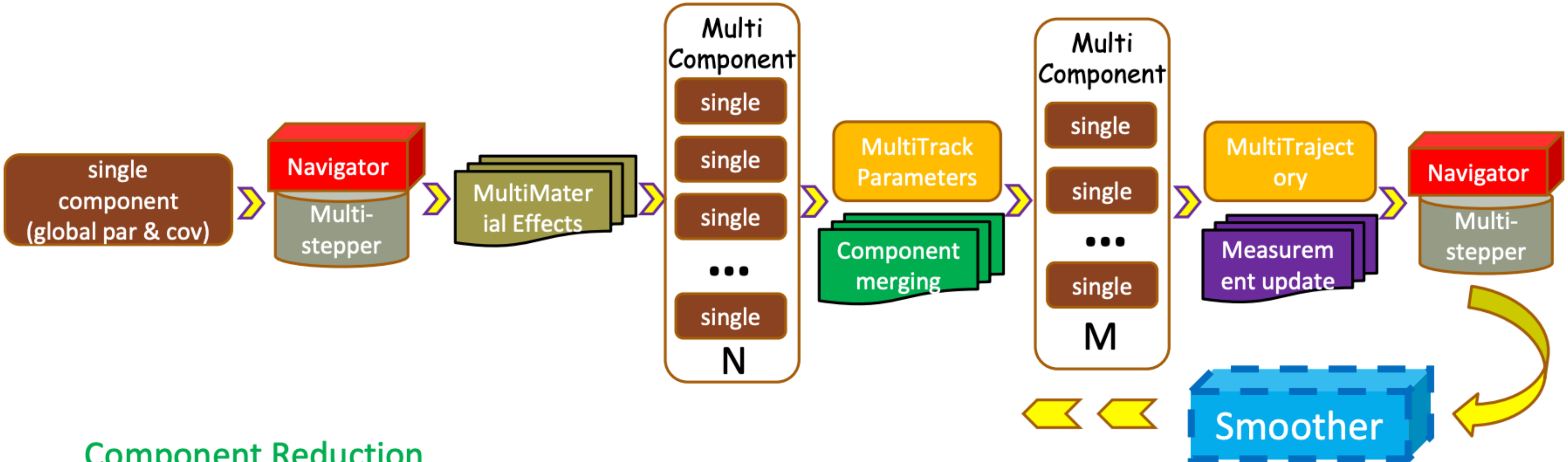
Multi-Stepper Propagation

- Take the combination of components – behave like single component in Navigation
- Each component owns its path
- Status(Free, Lock, Dead) of components decide if/when step forward

MultiMaterial Effect - Energy loss + Multiple scattering

- Bethe-Heitler – Currently take the ATLAS parameters to construct 6 components in each material effect

GSF: implementation (Cont'd)



Component Reduction
- Iteration to combine closet components to a maximum number

EDM for Gaussian sum filter
- Multitrajectory: TrackState store minimize heap allocation
- MultiTrackParameters : used for calculations of different components, e.g. component reduction

Measurement update with each component but modify the weight

Smoother: similar backward propagation, not prepared

Integration tests and validations for performance check will be done in the next step