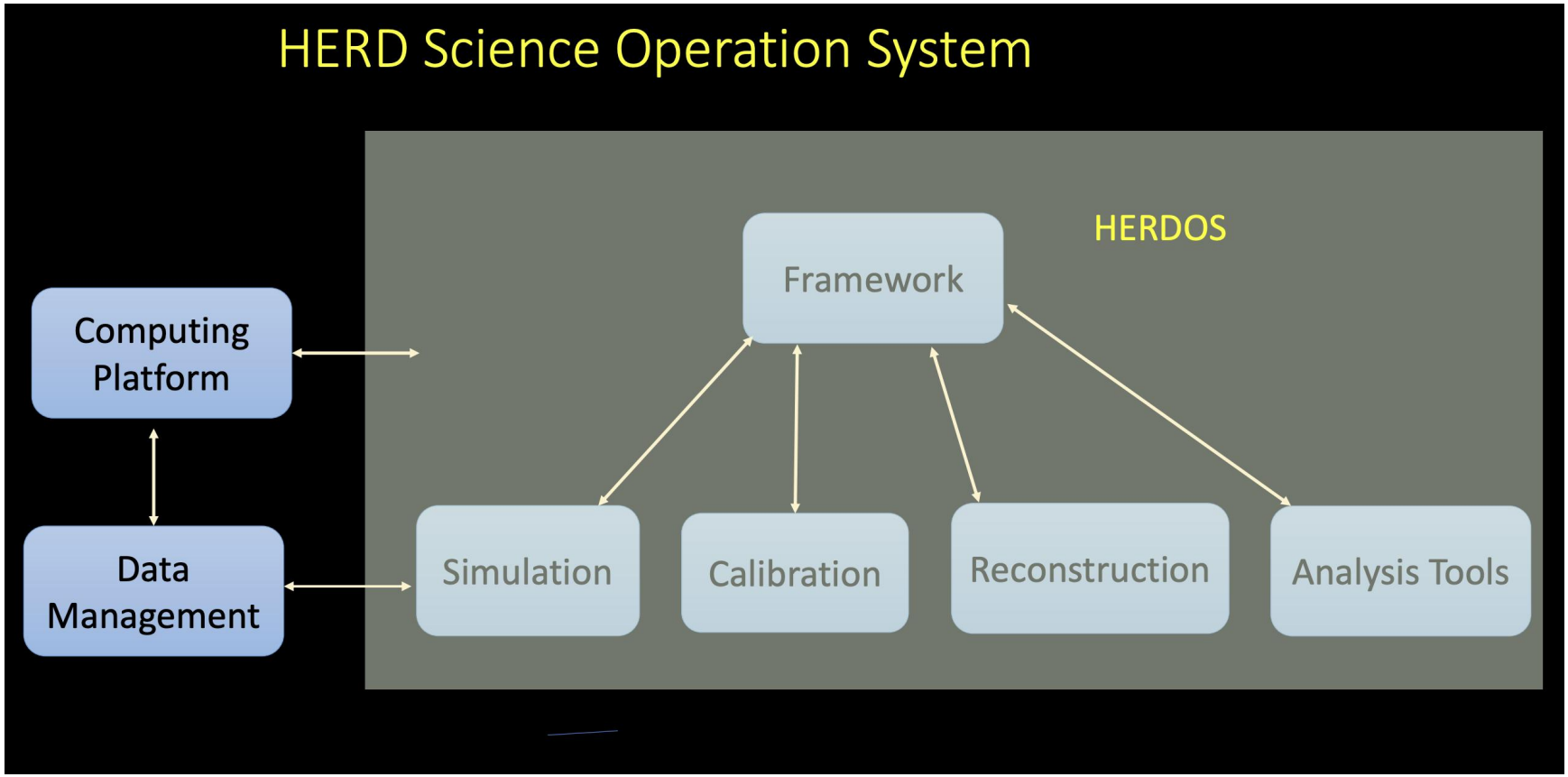
Brief introduction of data reconstruction in HERDOS

2024-07-12

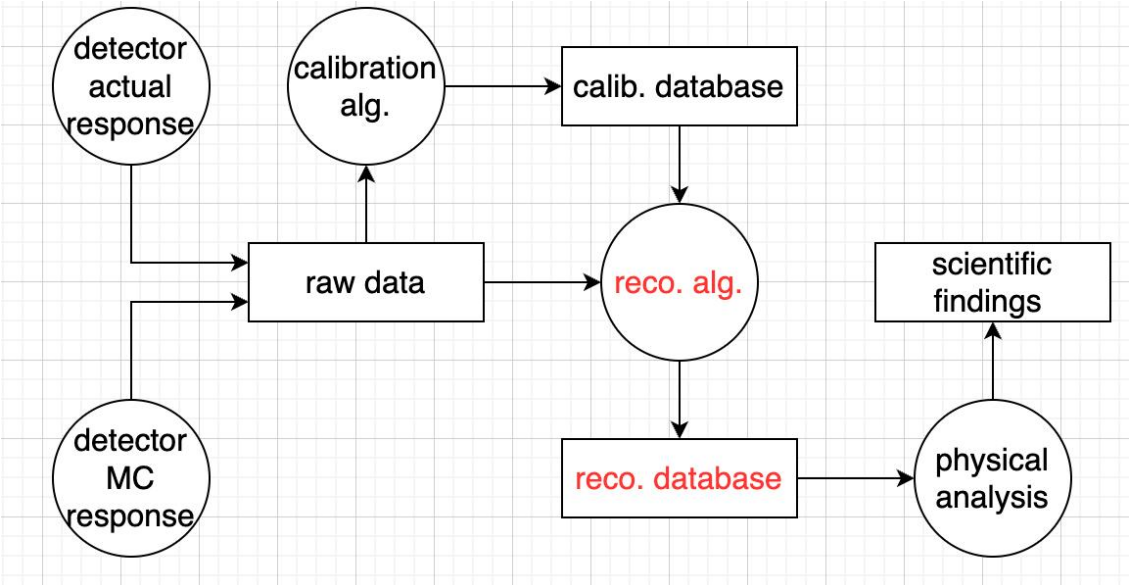
HERD science operation system



reconstruction is 1/7 of the sub-systems in HERD science operation system.
it takes care of the official(standard) process for extracting meaningful physical information from raw detector data.
for general SOC introduction, please refer:
https://indico.ihep.ac.cn/event/22450/

data reconstruction

- the official(standard) process for extracting meaningful physical information from raw detector data
  - convert the raw electrical signal data(raw data) recorded by detectors in both MC and actual way
  - process with reco alg. (with the calibration database)
  - finalize into physical quantities(reco. database) for further analysis/science



organizations in China

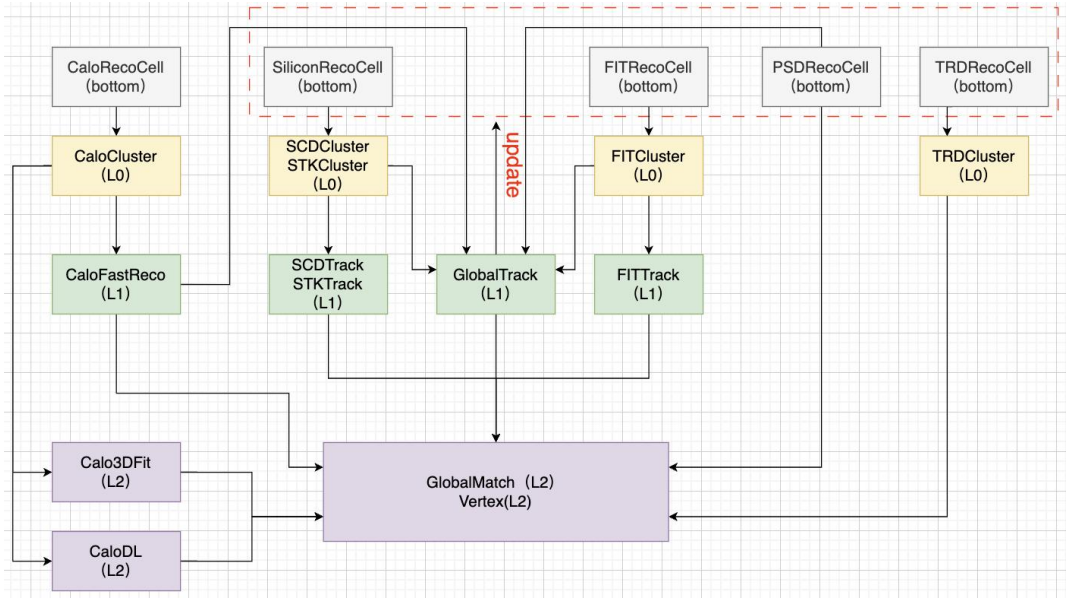
- established in 2022
- ~ 25 members from 7 institutes and universities
  - GXU, IHEP, SDU, SDIAT, SWJT, USTC, ZJU
- regular meetings
  - weekly meetings to report active progress and discuss issues
  - weekly machine learning related technical meetings
- dedicated discussions as needed for specific issue
  - workflow of camera calibration and bottom level reconstruction
  - 2023 test beam data calibration and reconstruction by using HERDOS
  - technical solutions for gamma-ray reconstruction
  - technical solutions for track reconstruction
  - workflow of TRD
  - overall scheme design
  - ...

development guideline

- code development based on IHEP cluster and git
  - some developments works on branches in reconstruction/repo.(restricted), as IO definition(EDM) and fundamental alg.(bottom level, clustering...)
  - some developments works on private fork, as high-level algorithms(shower fitting...)
- two level of review processes (outdated flow due to historical background)
  - by reconstruction group members:
    - within the reconstruction/repository, dedicated branch->master
    - code, technical details(physics), code documentation(doxygen), algorithms documentation, unit test will be checked
  - by framework maintainers: T. Li, Q.Q. Shi, Z.C. Tang ...
    - from reconstruction/repo. to HERDOS/repo.
- direct development at HERDOS/repo. as centralized workplace, under switching
  - discard the reconstruction/repo.

four level definition in reconstruction

- why multi-level definition?
  - data selection
  - error handling
  - computational efficiency, optimized storage
  - analysis strategy
  - collaboration on the modular design



bottom level: ADC to Edep

L0: Edep to clusters or hits

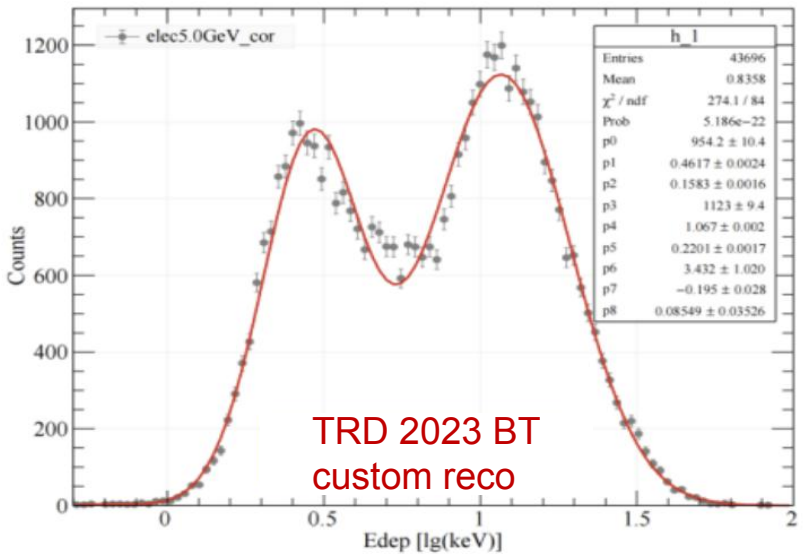
L1: primary reconstruction to form track feature(charge, energy) and particle feature(id)

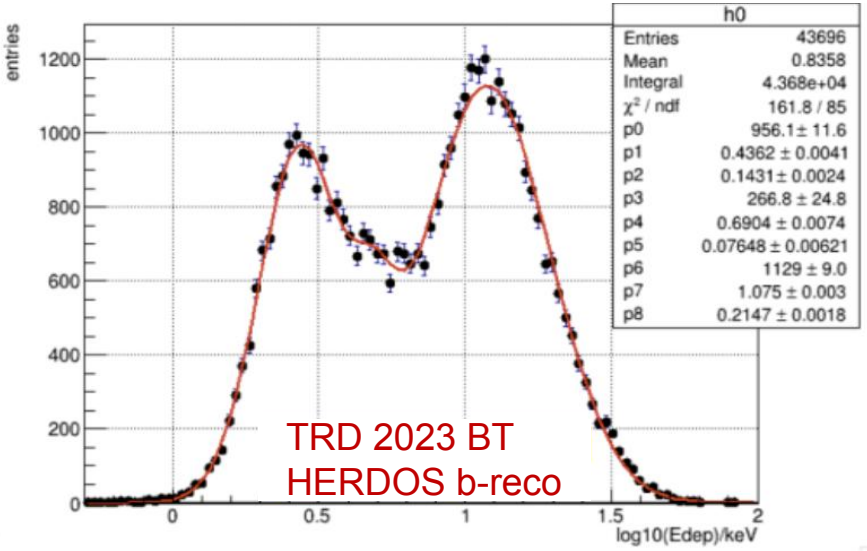
L2: high level reconstruction with advanced algorithms, to form events feature

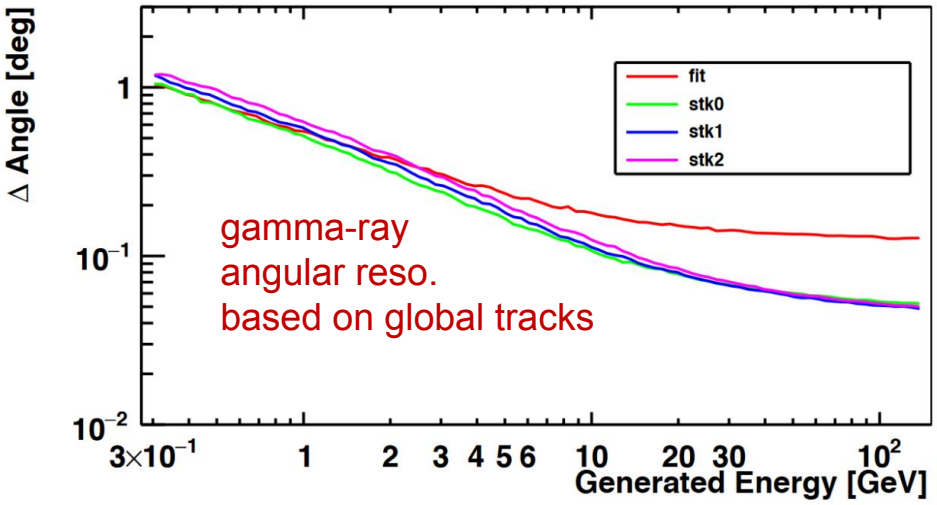
algorithm classes and status

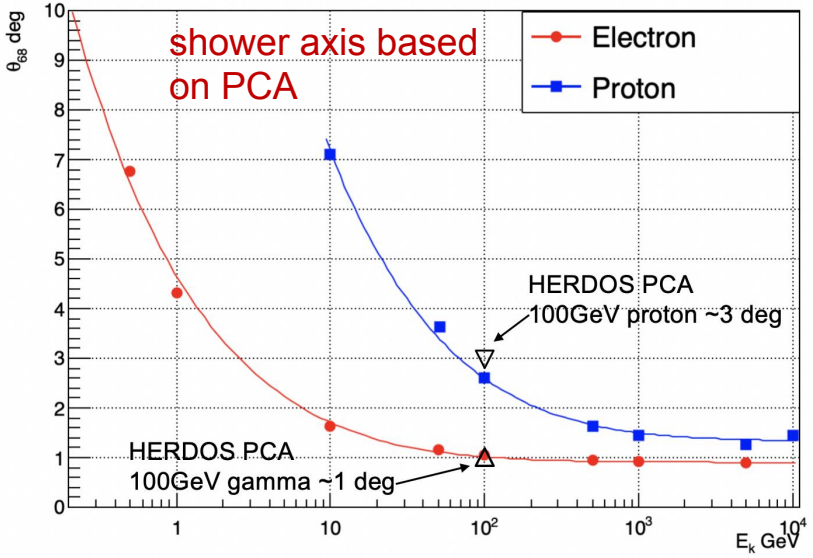
		algorithm	function	status
bottom level	1	CALOCameraBottomReco	convert camera pixel gray scale to edep	developing, v1 will be released 2024/08
	2	CALOPDBottomReco	convert PD adc to edep	not applied
	3	CALOPMTBottomReco	convert trigger PMT adc to edep	not applied
	4	FITBottomReco	convert SiPM adc to edep	not applied
	5	PSDBottomReco	convert SiPM adc to edep	developing, v1 will be released 2024/08
	6	SCD/STKBottomReco	convert strip adc to edep	developing, v1 will be released 2024/08
	7	TRDBottomReco	convert strip adc to edep	developing, v1 will be released 2024/08
L0	1	CALOClustering	cell hits into clusters	developing
	2	FITClustering	channel hits into clusters	v1 released, base on Sim. data
	3	SCDClustering	strip hits into cluster	v1 released, base on Sim. data, TB validation
	4	STKClustering	strip hits into cluster	will be merged with SCD
	5	TRDClustering	strip hits into cluster	developing
L1	1	CALOFast	fast/raw, shower axis, energy, pid	developing, and will be split
	2	FITLocalTracking	local tracks based on FIT clusters	v1 released, base on Sim. data
	3	SCDLocalTracking	local tracks based on SCD clusters	v1 released, base on Sim. data, TB validation
	4	STKLocalTracking	local tracks based on STK clusters	developing
	5	GlobalTracking	combine trajectory by integrating shower axis, FIT/SCD clusters, PSD hits	v1 released, base on Sim. data
	6	TrackUpdating	iterations of making corrections to clusters	developing
	7	PSD/STK/SCD chargeZ	hit/cluster equivalent Q, as to be iterated in tracking	developing
L2	1	CALO3DFit	precise shower axis, energy, pid based on Fitting	long term developing
	2	CALOML	precise shower axis, energy, pid based on Machine Learning	long term developing
	3	TrackML	advanced track finding based on Machine Learning	long term developing
	4	GlobalMatch	combine information and form particle info.	long term developing
	5	Vertex	fragmentation/conversion vertex finding	long term developing

some progresses in a nutshell









a developing example from user point of view

- migrate CaloPCA algorithm(shower axis reconstruction) from custom to HERDOS env.
  - custom version by Z. Quan (pure cpp with standard ROOT library)
  - migration to HERDOS by M. Xu
  - CaloPCA is the 1st reco. alg. and serves as the template for following reco. development
- useful information before the migration
  - admin/technical support, Z.C. Tang (tangzhch@ihep.ac.cn) and T. Li (tengli@sdu.edu.cn)
  - apply IHEP SSO and join HERDOS group to access git, documentation, indico...

step by step procedure(1)

- setup HERDOS environment
  - *source /cvmfs/herd.ihep.ac.cn/HERDOS/SLC7/Pre-release/ExternalLibs-GCC8.5.0/bashrc.sh*
- clone the whole offline/ from remote to local
- Create the CaloPCA package in offline/Reconstruction/CaloPCA/

```
.
|-- CMakeLists.txt           # CMake configuration file for package build and compilation settings.
|-- CaloPCA                  # CaloPCA module directory, containing related header files.
|   |-- CaloShowerAxisPCA.hh # Header file defining the CaloShowerAxisPCA class, including algorithm interface defini
|-- python                   # Directory containing Python scripts.
|   |-- CaloPCA.py           # Entry file controlling the package name, used for runtime references.
|-- scripts                  # Directory for scripts used for project maintenance, testing, or data processing.
|   |-- plotPCA.C            # ROOT script for plotting PCA-related output data.
|   |-- runtest.py           # Python script for running and testing.
|-- src                      # Source code directory, containing the core implementation code of the project.
|-- CaloShowerAxisPCA.cc # Implementation file for the CaloShowerAxisPCA class, containing the specific algorithm.
```

step by step procedure(2)

- modify the package related for build and compile
  - offline/Reconstruction/CMakeLists.txt
    - add_subdirectory(CaloPCA)
  - offline/Reconstruction/CaloPCA/CMakeLists.txt

```
herd_add_pkg(CaloPCA
    DEPENDS  GeometrySvc PodioDataSvc EventDataModel
)

```

  - offline/Reconstruction/CaloPCA/python/CaloPCA.py

```
import Sniper

Sniper.loadDll("libCaloPCA.so") #libXXXX.so, the package name as in CMakeLists.txt
del Sniper

```

step by step procedure(3)

- migrate the custom PCA code
  - CaloPCA/CaloPCA/CaloShowerAxisPCA.hh
  - CaloPCA/src/CaloShowerAxisPCA.cc

```
86  class CaloShowerAxisPCA: public AlgBase
87  {
88      public:
89
90          CaloShowerAxisPCA(string const& name);
91
92          virtual ~CaloShowerAxisPCA();
93          virtual bool initialize();
94          virtual bool execute();
95          virtual bool finalize();
96
97      private:
98        // EDM
99          const CaloSimCellCollection* simhits;
100         const MCParticleCollection* mcParticles;
101         CaloShowerAxisCollection* mAxis;
102
103         /** name of the input collection to be used for reco.*/
104         string mInputCollName;
105
106         bool buildAxis(const CaloSimCellCollection & ); // PCA algo. to build shower axis, based on cell MC truth.
107
108         /** in GeV, those above energyThreshold then to build CALO PCA*/
109         float energyThreshold;
110         /** hits collection of an event*/
111         vector<std::vector<double> > hitsAll;
112         /** weighted hits coordinates in XYZ direction*/
113         vector<double> hits;
114         /** cellid for dd4hepID conversion in ddxmlsvc */
115         std::vector<dd4hep::long64> cellid;
116 };
117
```

(from the manual)

three member functions are mandatory:

- ✓ initialize(): SNiPER will call initialize() before the run starts. Actions like creating histograms, connecting database, opening files could be placed here.
- ✓ finalize(): SNiPER will call finalize() after the run stops. Actions like saving histograms or TTrees, closing files could be placed here.
- ✓ execute(): SNiPER will call execute() once for each event during data processing, the main body of your algorithm should be placed here.

step by step procedure(4)

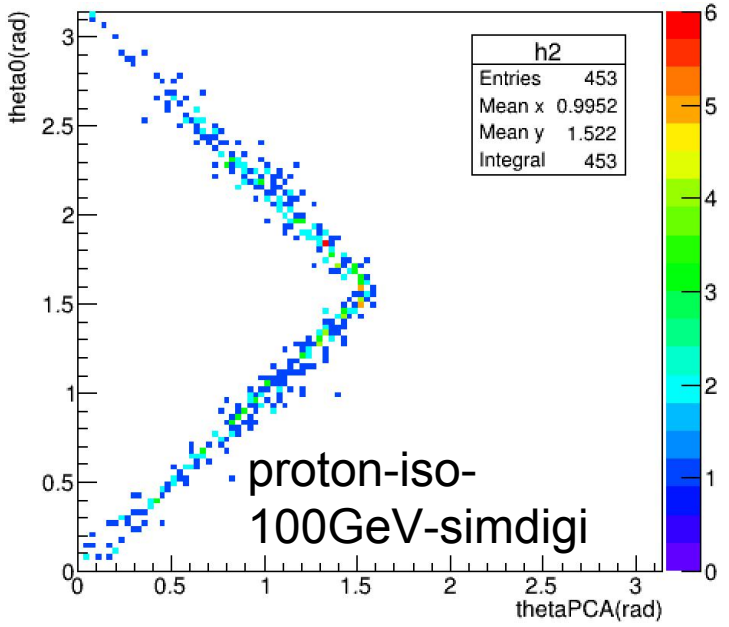
- prepare the running script CaloPCA/scripts/runtest.py and the script plotting the results plotPCA.C

```
21  isvc.property("InputFile").set("dummy.root"); #FIXME, please replace dummy.root with simulated output file.
22
23  osvc = task.createSvc("PodioOutputSvc/OutputSvc")
24  osvc.property("OutputFile").set("/dev/shm/caloPCA1.root");
25
26
27  import CaloPCA
28
29  xmlsvc = task.createSvc("GeometrySvc")
30  xmlsvc.property("GeoCompactFileName").set(hdos_install+"/compact/v2022a/v2022a-f.xml");
31  caloPCAalgo = task.createAlg("CaloShowerAxisPCA") # the Class name as in src/CaloShowerAxisPCA.hh
32  caloPCAalgo.property("inputcoll").set('calohits') # debug
33  caloPCAalgo.setLogLevel(sniperplus.DEBUG)
34
35  #task.setEvtMax(100)
36  task.setEvtMax(-1)
37  task.show()
38  task.run()
39
40
41  print('Output to ')
42  osvc.property("OutputFile").show()
43
44  import os
45  import sys
46
47  print('argv=', sys.argv)
48  dir=os.path.dirname(sys.argv[0])
49  if not dir : dir = '.'
50  os.system('root '+dir+'/plotPCA.C')
51
52
```

step by step procedure(5)

- compile and run
  - in offline/ dir., compile&install with *./build_RelWithDebInfo.sh*
  - setup the offline env., *source offline/install/setup.sh*
  - run the test scripts, *python CaloPCA/scripts/runtest.py*
    - a MC simulation file will be needed as input
  - plot the results *root plotPCA.C*
    - EDM data handle by standard ROOT

```
18  TChain *ch_mc = new TChain("events");
19  ch_mc->Add(f_mc);
20  TTreeReader reader_mc(ch_mc);
21
22  TChain *ch_pca = new TChain("events");
23  ch_pca->Add(f_pca);
24  TTreeReader reader_pca(ch_pca);
25
26  TTreeReaderArray<int> mcparts_trackID(reader_mc, "mcparts.trackID");
27  TTreeReaderArray<float> mcparts_momentum_x(reader_mc, "mcparts.momentum.x");
28  TTreeReaderArray<float> mcparts_momentum_y(reader_mc, "mcparts.momentum.y");
29  TTreeReaderArray<float> mcparts_momentum_z(reader_mc, "mcparts.momentum.z");
30
31  TTreeReaderArray<float> calopca_dir_x(reader_pca, "caloPCA.dir.x");
32  TTreeReaderArray<float> calopca_dir_y(reader_pca, "caloPCA.dir.y");
33  TTreeReaderArray<float> calopca_dir_z(reader_pca, "caloPCA.dir.z");
34
```



key points in the migration and supplements

- handle with Event Data Model for I/O

```
90    simhits = getROColl(CaloSimCellCollection, mInputCollName);
91    if(not simhits) return true; // false will terminate the event loop
92    mAxis = getRWColl(CaloShowerAxisCollection, "caloPCA");
93    if(not mAxis) return true;
```

- handle with geometry service

```
164   dd4hep::long64 cellcode = xyz2code( simhit.getIx(), simhit.getIy(), simhit.getIz() );
165   //dd4hep::VolumeID ddid = ddcellidFromCellCode(cellindex);
166   dd4hep::Position position = ddsvc->getPosition(GeometrySvc::CALO, cellcode);
167   float posx = position.X();
168   float posy = position.Y();
169   float posz = position.Z();
```

- supplements of CaloPCA
  - a comprehensive introduction, please refer to the framework guide
  - the 1st version of CaloPCA available at https://code.ihep.ac.cn/herdos/offline/tree/master/Reconstruction/CaloPCA
  - more detailed algorithm description document is being refined at https://code.ihep.ac.cn/herdos/docs/tree/4-refine-reco/sphinx_doc/source/reconstruction
  - the enhancement of unit test is under going

summary

most of the reco. algorithms are based on those well-established in the filed of particle physics, key points are more efficient and higher precision integration and application in HERD, e.g.

from narrow FOV to isotropic

from “clean” background to high contaminated

一起向未来

Together for a Shared Future