Tracking and ECAL reconstruction

Anne-Marie Magnan
Imperial College London

How well does the current implementation compared to the model ?

Does it fulfill its expectations ?

Introduction

The model rules : summary

I. Tracking reconstruction
   1. Objectives
   2. Current implementation in real data and MC
   3. Critique

II. ECAL reconstruction
   1. Objectives
   2. Current implementation in real data and MC
   3. Critique

Conclusion

My vision of the rules

- use of ILC software tools:
  - data format = **LCIO**
  - coding environment = **Marlin processors**
  - conditions data database = **CondDBMySQL**
  - database interface = **LCCD**
- **Identical steps in data and MC** in one processor
  - data and MC need to have common code as early as possible/needed
- code **transparent to setup** (i.e. Desy, CERN, Fermilab): handling of different steering parameters/conditions data folder without user intervention
- main repository for **conditions data: database**. Any code using local in-file copy should be identical as accessing directly from the database.
- Allow for the **maximum level of details**
  - being able to implement in the MC the technical issues discovered in data.
- **Performances:** 9720 channels (ECAL only) and 60 Millions data events to process regularly.
- Rules: maximum number of **users should be able to contribute**, so coding rules are mandatory.
- **Interface** to the analysis users **simple but flexible**, and **well documented**.
- **Avoid duplication of code**: provide conveniently usable common code

The first processors

- RunInfoProcessor: gather the information from the database on configuration of the run and write into the runHeader.
- ConditionsProcessor: through LCCD, give conditions data collections for each folder specified as steering parameter.
- CaliceTriggerProcessor : through the TriggerHandler class, give access e.g. to Cerenkov+scintillators information.

✓ use of LCCD/Marlin processors and database as source of information
x In RunInfoProcessor: still some information missing: e.g. angle configuration, main particle type
x In conditionsProcessor, folder names are hardcoded as steering parameters, and read inside the "init" method, implying different steering files for different setups. Proposal: automatically setup according to RunInformation in the "processRunHeader" method

Objectives of the tracking

- Reconstruct tracks from the position of hits in the 4 drift chambers, and **extrapolate the position and direction** of the incoming particle(s) :
  - to the ECAL (HCAL) front face.
  - backwards to the beam origin to measure the beam size
- Take into account **multiple scattering**: energy dependence implying error matrices are run dependent.
- **Feedback loops** to consider between MC and Data:
  - MC needed to calculate MS matrices at each energy
  - Data track reconstruction needed before final Monte Carlo generation for beam size
- Simple study of **systematics**, on reconstructed file whenever possible to avoid the CPU consuming part. Systematics:
  - effect of misalignement
  - incorrect material modelling
  - error on hit position
  - error on drift velocity

Tracking reconstruction

Raw data
LCIO LCIntVec
DESY/CERN/FNAL dependency
Electrical channels timestamps

mapping

MC data
SimTrackerHit
position, cellID and E for each hit.

digitisation

Raw data
LCIO LCIntVec
Geometrical channels hit timestamps,
X and Y values of each channels

✓ **Independent of location**
✓ **collections saved in output file**

Analysis classes for objects and constants provided in calice_reco/include(src)/TB TrackUtil
Move in "calice_analysis" package ??

reconstruction

Tracks, in X and in Y,
including **access to extrapolated position and direction in any detector point**

Conditions Data handling

/cd_calice_beam/ = DESY /cd_calice_cernbeam/ = CERN

Raw data
LCIO LCIntVec
Electrical channels amplitude

MC data
SimTrackerHit
position, cellID and E for each hit.

x mapping currently hardcoded

TBTrack/SimConstants

Raw data
LCIO LCIntVec
Geometrical channels amplitude,
4 channels, each X and Y values

x All folders (i.e. desy+cern) need to be given to the conditionsProcessor and the tracking processors take care of choosing the right one according to the runInformation

- **Constants given through the LCEvent**

TBTrack/AlnConstants
TBTrack/FitConstants

Tracks, in X and in Y, including access to extrapolated position and direction in any detector point

Transmission of parameters to analysis

- No steering parameters: all cut values are written in the database as part of the relevant constants classes.
- database=main source of information. Only max. 4 layers: all conditions data/geometry/etc… also written in the file for independent reprocessing and to know what was used to produce the file.
- Simulation: geometry saved in file during digitisation step: reco will access the same data ➔ self-consistent.

Study of systematics

- all conditions data + geometry + intermediate collections saved in file: possible to reprocess one of the reco processor on the reco file, and “delete +rewrite” the existing tracks collections and constants.

Comparison with model

- ✓ use of ILC software tools:
  - data format = LCIO
  - coding environment = Marlin processors
  - conditions data database interface = LCCD
- ✓ Identical steps in data and MC in one processor
  data and MC need to have common code as early as possible/needed
- ✓ code transparent to setup (i.e. Desy, CERN, Fermilab): handling of different steering parameters/conditions data folder without user intervention
- ✓ main repository for conditions data: database. Code using local in-file copy should be identical as accessing directly from the database.
- ✓ Allow for the maximum level of details
  being able to implement in the MC the technical issues discovered in data.
- ✓ Performances: only 4 layers max, not a real issue….
- x Rules: maximum number of users should be able to contribute, so coding rules are mandatory.
- x Interface to the analysis users simple but flexible, and well documented.
- ✓ Avoid duplication of code: provide conveniently usable common code, through "calice_analysis" classes

Comparison with objectives

- ✓ Reconstruct tracks from the position of hits in the 4 drift chambers, and extrapolate the position and direction of the incoming particle(s) :
  - ✓ to the ECAL (HCAL) front face.
  - ✓ backwards to the beam origin to measure the beam size: track reconstruction needed before final Monte Carlo generation.

- ✓ Take into account multiple scattering: energy dependence implying error matrices are run dependent.

- x **Feedback loops** to consider between MC and Data:
  - x MC needed to calculate MS matrices at each energy
  - x Data track reconstruction needed before final Monte Carlo generation for beam size

- ✓ Simple study of systematics, on reconstructed file whenever possible to avoid the CPU consuming part. Systematics:
  - ✓ effect of misalignement
  - ✓ incorrect material modelling
  - ✓ error on hit position
  - ✓ error on drift velocity

Objectives of the ECAL reconstruction

- provide **ECAL hits** (clusters) with position, cellID, **calibrated energy** in MIPS, **in aligned detectors**.
- **Subtract pedestals**
- procedure to **correct for pedestal shifts**
- store **dead channels** information
- flag noisy channels
- **flag pathological events** (e.g. square events)
- calculate parameters (e.g. pedestals, noise) for use in the MC.
- allows simple study of **systematics**, whenever possible **on the reco file**. Systematics:
  - gain variation
  - noise+pedestal subtraction uncertainties
  - alignment

ECAL reconstruction

Raw data
LCIO LCIntVec
Electrical channels amplitude

handling of DAQ type: beamData, pedestals, calibration runs. Pedestal calculation on pedestal events. Pedestal subtraction. Bad event rejection. Pedestal corrections. Mapping to geometrical channels

MC data
LCIO SimCalorimeterHit
position, cellID and E for each hit.

decalibration
add noise

x Amplitude = Int*10000 to have float precision!

Raw data
LCIO RawCalorimeterHit
Geometrical channels amplitudes

Calibration, threshold cut.

Reconstructed hits
LCIO CalorimeterHits position, CellID, E

Conditions Data handling
R= read W=write

R Geometry from Mokka database

Raw data
LCIO LCIntVec
Electrical channels amplitude

R -- calibration constants
W – average noise per channel
R – mapping hardware->geom

MC data
LCIO SimCalorimeterHit
position, cellID and E for each hit.

R – mapping geom-hardware
(x accessible only if cell was connected in real data)
R -- calibration constants
R -- noise per channel

Raw data
LCIO RawCalorimeterHit
Geometrical channels amplitudes

R – mapping Geom->hardware
R -- calibration constants
W(-in file only) – dead channels list

Reconstructed hits
LCIO CalorimeterHits
position, CellID, E

GEOMETRY INACCESSIBLE FOR MC: if use of CALICE one, INCONSISTENCY

Study of systematics

- **No possibilities** currently to study systematics on the reconstructed file, **without having to rerun everything**.
- Proposal:
  - accessing all constants by geometrical channel instead of hardware, so that they are usable on a reconstructed file.
  - alignment: do not rely on CalorimeterHit position, but **recalculate the position from the geometry data** at analysis time. (Removing position could also save disk space….)
  - calibration: **lower the threshold cut, to allow gain and noise effects to be studied** without a big influence on the final sample.

Transmission of parameters to analysis

- Lots of steering parameters, most of them for experts use only.
- Nearly all constants written to the database
- but no "geometrically-based" classes existing to access constants from reconstructed hits: e.g. a user wanting to access the calibration constants of a cell would currently have to first convert the "geometrical cellID" into a "moduleID", "module_type" and "cell_index".
- No geometry information in MC files: not possible currently to reconstruct the x,y,z position of a hit in agreement with the geometry used for generation.

Comparison with model

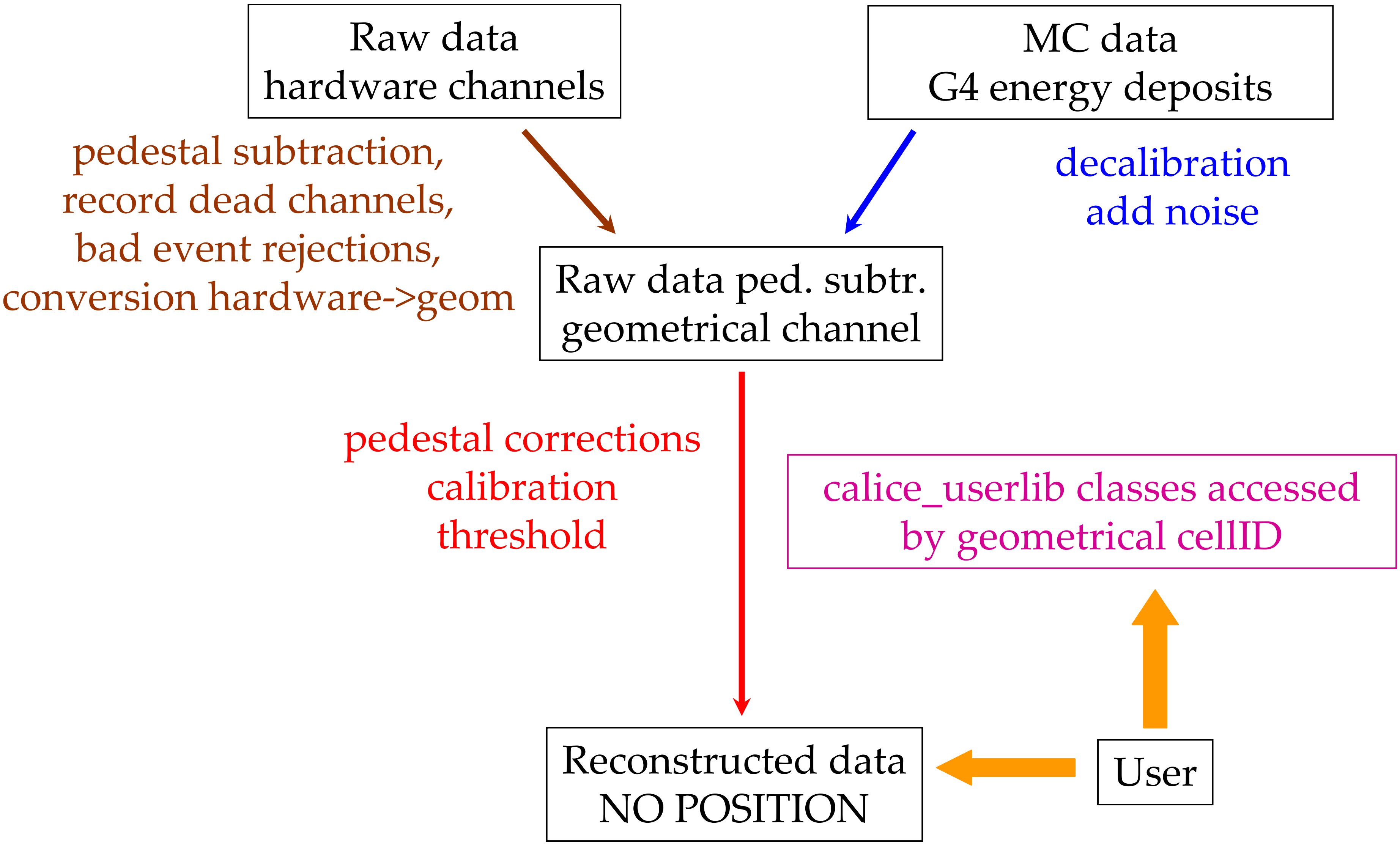
- ✓ use of ILC software tools:
  - data format = LCIO
  - coding environment = Marlin processors
  - conditions data database = CondDBMySQL
  - database interface = LCCD
- vx Identical steps in data and MC in one processor
  - data and MC need to have common code as early as possible/needed
- ✓ code transparent to setup (i.e. Desy, CERN, Fermilab)
- ✓ main repository for conditions data: database. Any code using local in-file copy should be identical as accessing directly from the database.
- x Allow for the maximum level of details: being able to implement in the MC the technical issues discovered in data.
- ✓ Performances: 9720 channels (ECAL only) and 60 Millions data events to process regularly.
- x Rules: maximum number of users should be able to contribute, so coding rules are mandatory.
- x Interface to the analysis users simple but flexible, and well documented.
- x Avoid duplication of code: provide conveniently usable common code.

CALICE
Calorimeter for LC

Comparison with objectives

- ✓ Subtract pedestals
- ✓ procedure to correct for pedestal shifts x but not checkable in MC!!!
- ✓ provide ECAL hits (clusters) with position, cellID, calibrated energy in MIPS, in aligned detectors,
- x but not possible to access easely the geometry offline.
- x store dead and bad channels information in database
- ✓ flag pathological events (e.g. square events) (however not yet checked… and not in database)
- ✓ calculate parameters (e.g. pedestals, noise) for use in the MC.
- x allows simple study of systematics, whenever possible on the reco file. Systematics:
  - x gain variation
  - x noise+pedestal subtraction uncertainties
  - x alignment

Proposal to join data and MC earlier



Conclusion

- Complete software chain is THERE and WORKING
- BUT: designed to run primarily on DESY testbeam data, then modified to accommodate CERN, then modified to accommodate MC, then….
- Need a bit or reorganisation in 3 main points:
  - i. provide a convenient geometry interface for both data and MC
  - ii. calice_userlib classes accessed by geometrical indices to be used on both reconstructed and MC files
  - iii. Data and MC joined earlier in the chain: separation of the current "data" processor in 2: one "data only" and one common.
- And :
  - i. conditionsProcessor allowing steering files independent of location,
  - ii. organise **feedback loop** between MC and Data
  - iii. write **code rules** and **DOCUMENTATION**.