Proposal for ECAL simulation and reconstruction chain update

D.Peresunko
RRC “Kurchatov institute”

List of Data structures

- Hit (class MpdEmcHit): energy deposited by one particle entered ECAL in one tower
- Digit (class MpdEmcDigit): energy deposited by all particles (+ noise) in one tower. Same as signal in real data
- Cluster (class MpdEmcCluster): list of digits and energy depositions + cluster parameters: energy, position, time, dispersion, distance to track extrapolation. Keeping list of digits allows re-calculation cluster parameters in case of improvement of calibration.

General code considerations

- Put all geometry-dependence parts into one class (singleton MpdEmcGeoUtils)
  - To reduce number of bugs appeared in case of update of geometry
- Put all simulation and reconstruction parameters to one class (singleton MpdEmcSimParams)
  - Flexible change of parameters in config. Macros
  - Reduce number of bugs due to variation of parameter in different pieces of code.

MpdEmcGeoUtils

Introduce (arbitrary) numbering of towers: detID. Based on this index one can calculate position or check neighbours.

```
///
/// \return the pointer of the _existing_ unique instance of the geometry
/// It should have been set before with GetInstance(name) method
///
static MpdEmcGeoUtils* GetInstance() { if(!sGeom) { sGeom = new MpdEmcGeoUtils() ;}
                                       return sGeom; }

// Check if two towers have common side (for clustering)
// \return -1: second from prev. sector, 0: no, 1: yes, 2: towers too far apart, no sense to continue searching for neighbors
int AreNeighbours(int detId1, int detId2) const;

// Check if two towers have common vertex (for unfolding)
// \return -1: second from prev. sector, 0: no, 1: yes, 2: towers too far apart, no sense to continue searching for neighbors
int AreNeighboursVertex(int detId1, int detId2) const;

int GeantToDetId(int chamberH,int chamber,int sector,int crate, int box) const ;   //Convert Geant volume indexes to abs ID of a channel
void DetIdToRelIndex(int detId, int & chamber, int &sector, int &iphi, int &iz) const ; //Convert detId to iphi,iz indexes within one sector

int GetTotalNCells() const { return fNTowersPerChamber * fNChambers ; }

void DetIdToGlobalPosition(int detId, double &x, double &y,double &z) const ; //calculates senter of front surfase of tower with index detId
```

Class MpdEmcSimParams

```
bool fSmearLightCollection = true; // Emulate collecting light using fEdepToLightYield photoelec
bool fSimulateNoise = true ;       // Simulate electronic noise in HitCreation
bool fApplyNonlinearity = false;   // Apply energy non-linearity in HitCreation
bool fApplyDigitization = true ;   // Apply digitization of energy in HitCreation
bool fApplyTimeResolution = true ; // Apply time resolution in HitCreation

double fElectronicNoiseWidth = 0.005 ; //Width of Gaussian electronic noise in GeV
double fCelNonlinParamA =-0.02 ;   // Cell energy non-linearity parameterization
double fCelNonlinParamB = 0.5  ;   // in the form
double fCelNonlinParamC = 1.0  ;   // e=e*c(1-a*exp(-e/b))
double fADCWidth = 0.005 ;         // Widht of one ADC count in GeV (used in energy digitization in HitCreation)
double fZSthreshold = 0.005;       // ZeroSuppression threshold (remove digits below) in GeV
double fTimeResolitionParamA=5.e-10 ; // Parameters used for time resolution simulation
double fTimeResolitionParamB=2.e-11 ; // in the form width = a + b/e (in seconds)
double fNoiseTimeMin =-100.e-9 ;   // simulate noise signal
double fNoiseTimeMax = 100.e-9 ;   // in this range (in seconds)
double fEdepToLightYield=200000.;  // Number of photoelectrons per GeV

//Clusterization
double fLogWeight = 3.;            // cutoff used in log. weight calculation
double fDigitMinEnergy=0.005;      // Minimal energy of digits to be used in cluster (GeV)
double fClusteringThreshold=0.030; // Minimal energy of digit to start clustering (GeV)
double fLocalMaximumCut=0.030;     // minimal height of local maximum over neighbours
double fClusteringTimeGate=1000.;  // maximal time difference between digits to be accepted to clusters (in ns)
bool   fUnfoldClusters=true ;      // to perform cluster unfolding
double fUnfogingEAccuracy=1.e-4 ;  // accuracy of energy calculation in unfoding prosedure (GeV)
double fUnfogingXZAccuracy=1.e-2;  // accuracy of position calculation in unfolding procedure (cm)
double fEcoreCut1 = 0.001;         // threshold for Ecore calculation E_p1
double fEcoreCut2 = 0.002;         // threshold for Ecore calculation E_p2
double fChi2radiusCut = 0.0001;    // cut in dispersion Chi2
int    fNMaxIterations=6;          // maximal number of iterations in unfolding procedure
int    fNLMMax = 30;               // Maximal number of local maxima in unfolding
int    fNPrimMax=5;                // Maximal number of primaries in list (sorted with deposited energy)
```

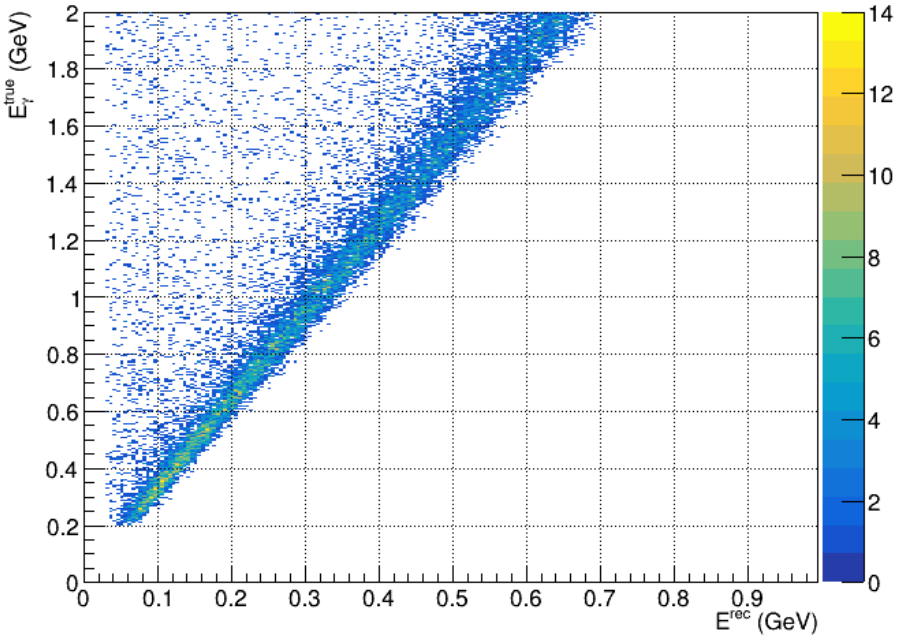
To apply changes to default parameters, call in reco.C before running:

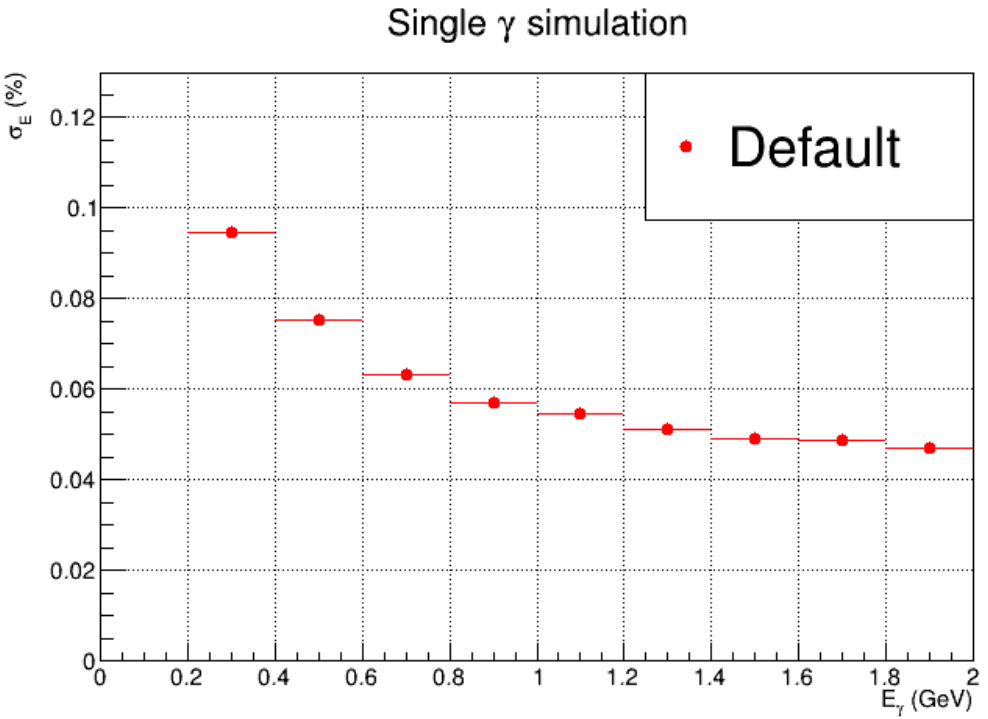
```
MpdEmcSimParams * par =
MpdEmcSimParams::GetInstance() ;

par->fSmearLightCollection = false; // Emulate LY

par->fEdepToLightYield=40000;
```

Single photon simulation

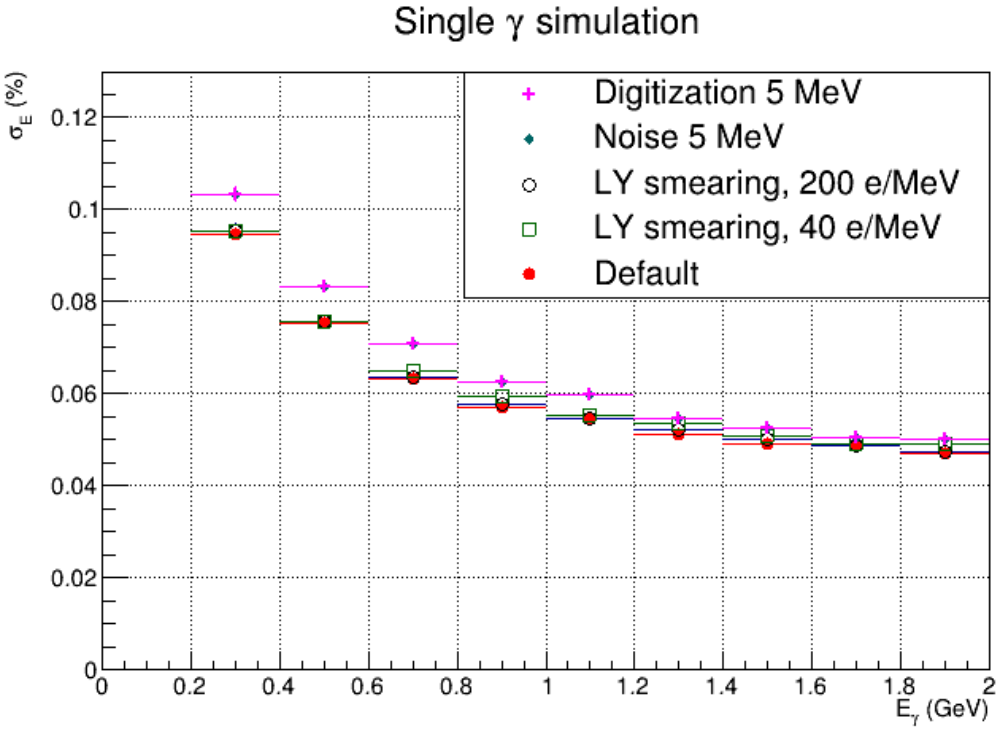




Simulate only energy deposition in scintillators: best possible resolution

Single photon: additional smearing

- Light collection: Poisson smearing
  - Minor effect even with extremely low number of photons/MeV~40
- Electronic noise
  - Optimistic 5 MeV (deposited energy)
- Digitization
  - 5 MeV/ADC count



Electronic noise and digitization have similar effects because of similar parameters.

Need realistic input for reasonable simulations!

Conclusions

- Proposal:
  - commit code to git (who?)
  - Test by community
- Todo:
  - Bug fixing and comparing to default reconstruction chain (1-2 weeks)
  - Test options of track matching (1-2 weeks)
  - Implement realistic response parameters (need input from beam-test => time scale??)
  - Implement realistic time resolution (need parameterization from beam-test => time scale??)
  - Implement (de-)calibration, mis-alignemnt, bad map (2 months)
    - Interface with calibration database?
  - More requests?