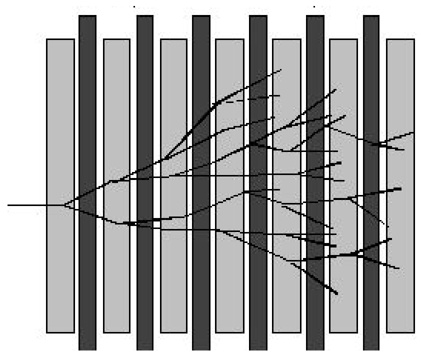
Proposal for ECAL simulation and reconstruction chain update

D.Peresunko
RRC “Kurchatov institute”

Tracking in ECal

- Geant provides particle parameters and energy loss on each step to Detector to calculate its response in active volume
  - Add energy depositions in all scintillator layers
  - Add contributions from all secondaries, originated from particle entered front surface of calorimeter
  - Remember time and trackID of primary particle
- Implemented in class MpdEmcKI
- Produce hits (class MpdEmcPointKI)



Result of tracking:
- MCTrack index
- E: sum of energy depositions in Sc layers
- time of particle appearance
- Tower ID

Hit structure (class MpdEmcPointKI)

- Hit: Energy deposited by one particle entered Ecal in one tower
  - =>(sum over EM shower inside detector; do not keep history of EM shower in MC stack)
- class MpdEmcPointKI : public FairMCPoint
- Inhereted data members:

trackID	Index of MCTrack
detID	Detector ID
pos	Coordinates at entrance to active volume [cm]
mom	Momentum of track at entrance [GeV]
tof	Time since event start [ns]
length	Track length since creation [cm]
ELoss	Energy deposit [GeV]

- Methods how to add, compare and sort hits modified

Keep class as simple as possible to reduce disk/memory usage

Geometry class

4 kinds of tower coordinates:

- Geant (active) volume name (string)
- DetID (integer)
- Hardware address (integer/list of integers?)
- (x,y,z) coordinates in global MPD frame

Class: MpdEmcGeoUtils

```
/// \return the pointer of the unique instance of the geometry
static MpdEmcGeoUtils* GetInstance();

// Convert Geant volume indexes to abs ID
int GeantToDetId(int chamberH, int chamber, int sector, int crate, int box) const;

// 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;

//calculates center of front surface of tower with index detId
// Uses TGeoManager and constructed geometry
void DetIdToGlobalPosition(int detId, double &x, double &y,double &z)const ;
```

```
root [0] MpdEmcGeoUtils * geom = MpdEmcGeoUtils::GetInstance();
root [1] geom->GeantToDetId(0,1,3,12,32)
(int) 29984
root [2] geom->GeantToDetId(0,1,3,12,33)
(int) 29985
root [3] geom->GeantToDetId(0,1,3,13,32)
(int) 30048
root [4] geom->AreNeighbours(29984,30048)
(int) 1
root [5] geom->AreNeighbours(29985,30048)
(int) 0
```

Common side: neighbors

Common vertex: not neighbors

Global coordinates: Create geometry first:

```
root [0] .x $VMCWORKDIR/macro/mpd/mpdloadlibs.C
root [1] FairRunSim* fRun = new FairRunSim();
root [2] .L $VMCWORKDIR/macro/mpd/geometry_stage1.C
root [3] geometry_stage1(fRun)
[INFO] Media file used: /opt/mpdroot/geometry/media.geo
root [4] fRun->SetName("TGeant3");
root [5] fRun->Init()
[INFO] ==============  FairRunSim: Initialising simulation run
Info in <TGeoManager::TGeoManager>: Geometry FAIRGeom, FAIR
geometry created
[INFO] FairGeoMedia: Read media
....................
[INFO] Monte Carlo Engine Initialisation with: TGeant3TGeo
root [6] MpdEmcGeoUtils * geom = MpdEmcGeoUtils::GetInstance() ;
root [7] Double_t x,y,z;
root [8] geom->DetIdToGlobalPosition(30048,x,y,z) ;
root [9] cout << x << "," << y << "," << z << endl ;
-96.2949,185.741,-136.514
```

Digitization

- Convert energy deposition to detector response:
  - Adding contributions from different parents
  - Poisson light collection
  - Electronic noise
  - ADC digitization

```
class MpdEmcDigitizerKI : public FairTask
{
 public:
  MpdEmcDigitizerKI();
  ~MpdEmcDigitizerKI();
  virtual InitStatus Init();
  virtual void Exec(Option_t* opt);
  void virtual Finish();

 private:
  double SimulateNoiseEnergy();                     // Simulation of noise of electronics
  double NonLinearity(const double e);              // simulate non-lineraity
  double DigitizeEnergy(const double e);            // Account final width of ADC
  double TimeResolution(const double time, const double e);
                                                    // Apply final time resolution
  double SimulateNoiseTime();                       // calculate time in noise digit
  double SimulateLightCollection(const double lostenergy);
                                        // Simulate Poissonian light production and collection
```

To run, include to reco.C:

```
FairTask * emcDig = new  MpdEmcDigitizerKI() ;
fRun->AddTask(emcDig) ;
```

Clusterization

- Collect neighbor digits to cluster
  - Seed should be above threshold
  - Neighbors have common side
    - Can be from different sectors
- Unfold clusters with several local maxima
  - Local maximum: more than C_{LM} higher than surrounding cells (including common vertex) and above seed threshold
- Calculate cluster parameters

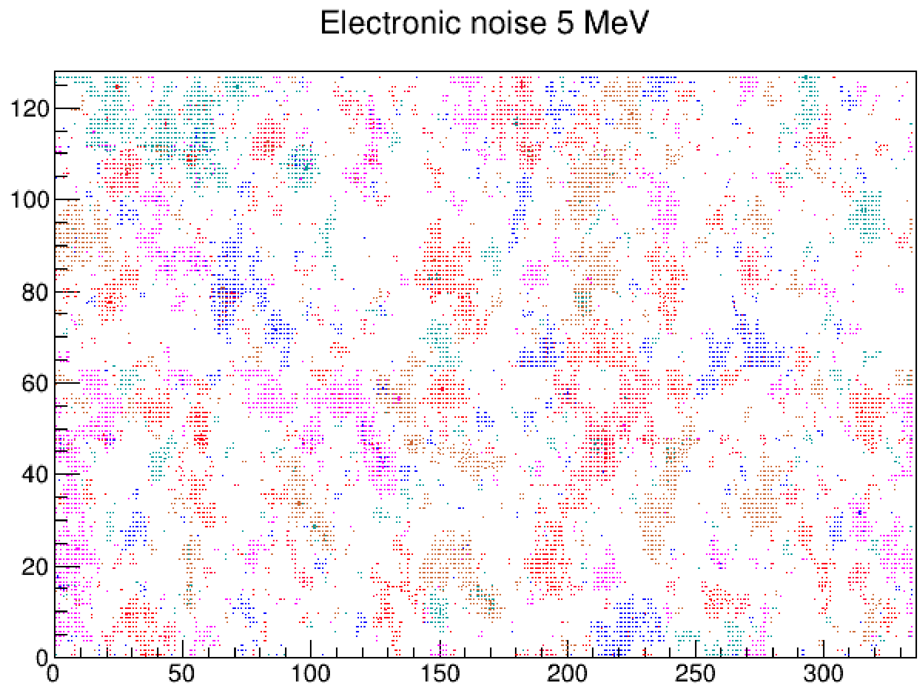
```
class MpdEmcClusterizerKI : public FairTask
{
 public:
  // Constructors/Destructors ---------
  MpdEmcClusterizerKI();
  ~MpdEmcClusterizerKI() {}
  virtual void Exec(Option_t* opt);
  static double ShowerShape(double dx, double dz);
                 // Parameterization of EM shower
 protected:
  void PrepareDigits();  // Calibrate, Allpy BadMap, clean...
  void MakeClusters();   // Do the job
  void MakeUnfoldings(); // Find and unfold clusters with few local maxima
  void UnfoldOneCluster(MpdEmcClusterKI* iniClu, Int_t nMax,
                                      int* digitId, float* maxAtEnergy);
                                       // Performs the unfolding of a cluster with
                                       // nMax overlapping showers
    void EvalClusters();     // Calculate cluster parameters: E, (x,y,z),...
```

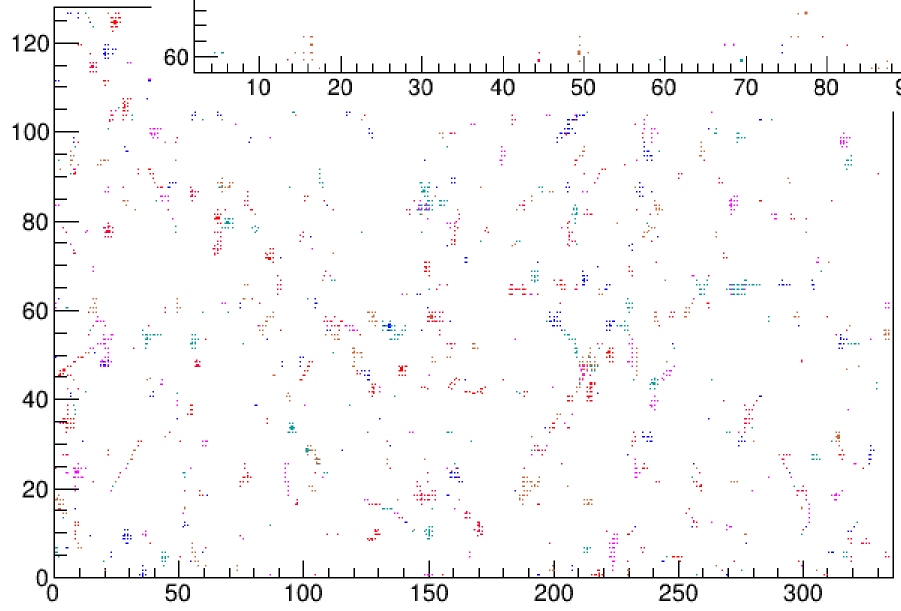
To run, include to reco.C:

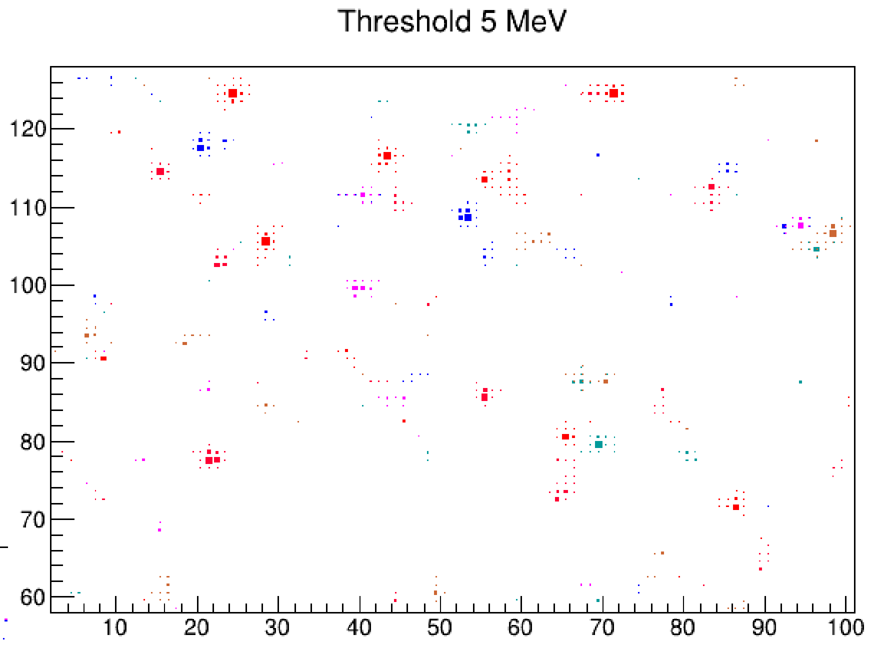
```
FairTask * emcClu = new MpdEmcClusterizerKI() ;
fRun->AddTask(emcClu) ;
```

Examples of clusterization: event with 50 photons

Test event reconstructed with different parameters: with and without minimal energy threshold. Different colors represent different reconstructed clusters







Track matching

- Assign each cluster the closest track extrapolated to Ecal
- Assign each track closest cluster
- Correct cluster position for displaced vertex
- Tracks should be extrapolated to
  - ECAL inner surface?
  - Some depth in ECAL (~5X0, ½ of full depth,...?)
- “Closest”: smallest distance in phi and z directions
  - Different measure is being considered:

$$R^2 = \frac{(D\phi)^2}{\sigma_\phi^2} + \frac{(Dz)^2}{\sigma_z^2}$$

```
class MpdEmcMatchingKI : public FairTask
{
 public:
  MpdEmcMatchingKI();

  virtual void Exec(Option_t* opt);
  void virtual Finish();

 protected:
  void ExtrapolateTracks();   // Fill array with points of track
                              // extrapolations to ECAL surface
  void MakeMatchToClusters(); // find best match to clusters
  void MakeMatchToTracks();   // Find best match to tracks
  void CorrectClustersVtx();  // correct clulster position for
                              // Zcoordinate of vertex
```

To run, include to reco.C:

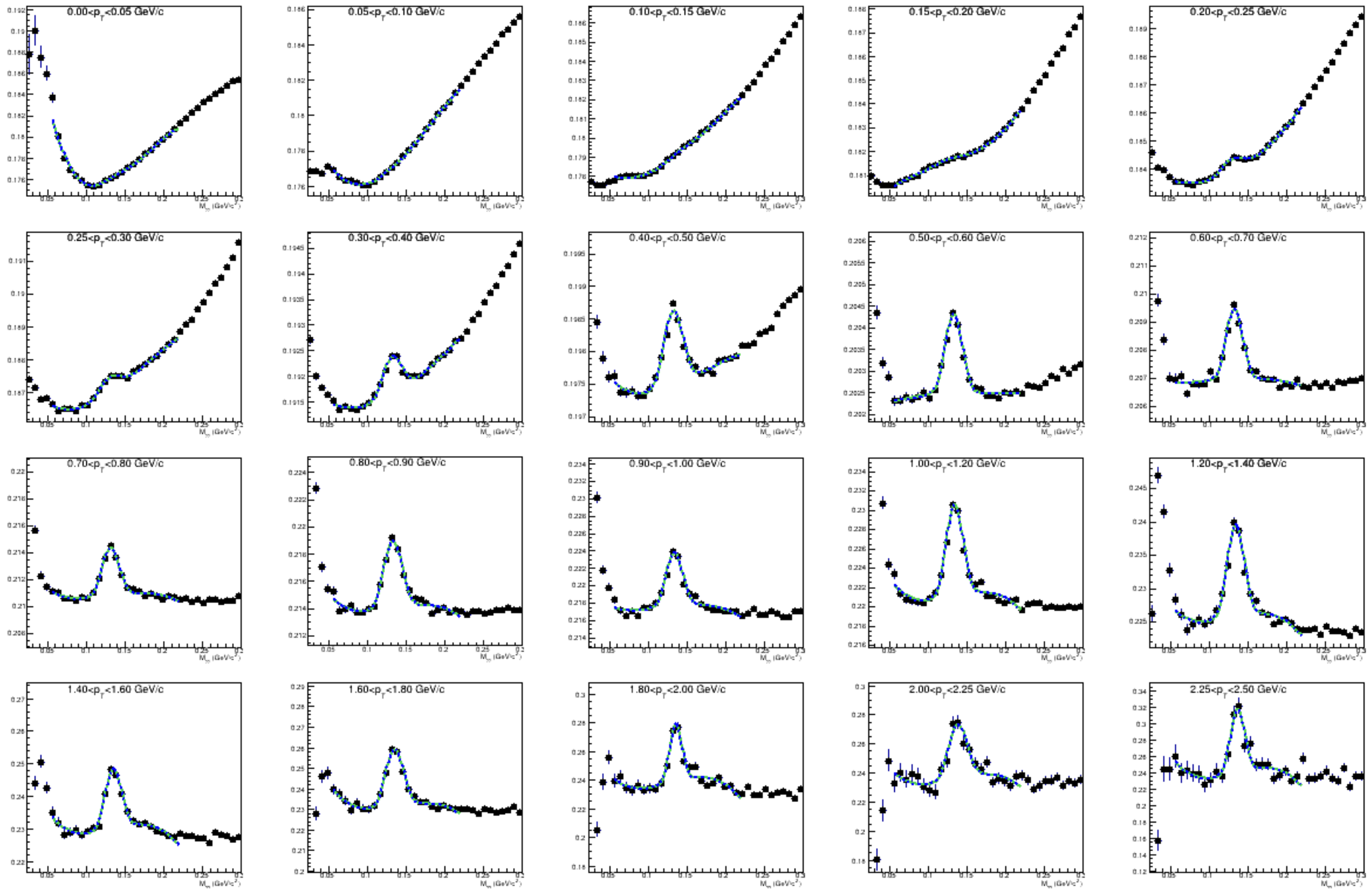
```
FairTask * emcTM = new MpdEmcMatchingKI() ;
fRun->AddTask(emcTM) ;
```

Output: MpdEmcClusterKI

```
class MpdEmcClusterKI : public TObject{
 public:
  // Returm momentum of photon assuming it came from the provided vertex
  void GetMomentum(TLorentzVector& p, const TVector3* vertex) const;

  void AddDigit(const MpdEmcDigitKI* digit, Double_t edep = 0);   //Add digit and deposited energy
  void EvalAll();              // Evaluate cluster parameters
  int GetNumberOfLocalMax(int* maxAt, float* maxAtEnergy) const;     // Finds local maxima
  void GetDigitParams(int i, int& detId, float& eDigit) const ;                // Get tower ID and energy of i-th contributing digit
  int GetNumberOfTracks() const { return fNPrimaries; }                        // Number of MC tracks
  void GetMCTrack(Int_t i, Int_t& trackId, Float_t& trackEdep) const ;   //Track number in stack and energy deposited by it
  Float_t GetE() const { return fE; };
  Float_t GetEcore() const { return fEcore; };
  Float_t GetChi2() const { return fChi2; };
  Float_t GetTime() const { return fTime; };
  Float_t GetX() const { return fX; };                //Coordinates
  Float_t GetY() const { return fY; };                //calculated at the inner
  Float_t GetZ() const { return fZ; };                //surface of ECAL tube
  Float_t GetPhi() const { return TMath::ATan2(fY, fX); };
  Float_t GetRho() const { return TMath::Sqrt(fX * fX + fY * fY); };
  Float_t GetRad() const { return TMath::Sqrt(fX * fX + fY * fY + fZ * fZ); };

  Int_t GetTrackIndex() const { return fTrackId; };
  Float_t GetDPhi() const { return fdPhi; };
  Float_t GetDZ() const { return fdZ; };

  Int_t GetMultiplicity() const;
  Int_t GetNLM() const { return fNExLM; }

  void GetLambdas(Float_t& l1, Float_t& l2);      // Dispersion parameters
```

```
TTree * t= (TTree*)f->Get("mpdsim") ;
TObjArray * clusterBr  = new TObjArray() ;
t->SetBranchAddress("EmcCluster",&clusterBr) ;
TClonesArray *vtxs = 0;
t->SetBranchAddress("Vertex",&vtxs);

TVector3 primVert;
TLorentzVector p1,p2;

for(Int_t i=0; i<t->GetEntries(); i++){
   t->GetEntry(i) ;
   MpdVertex *vtx = (MpdVertex*) vtxs->First();
   vtx->Position(primVert);
   for(Int_t j=0; j<clusterBr->GetEntries(); j++){
     MpdEmcClusterKI * clu1 =
                 (MpdEmcClusterKI*)clusterBr->At(j);
     clu1->GetMomentum(p1, &primVert);
```

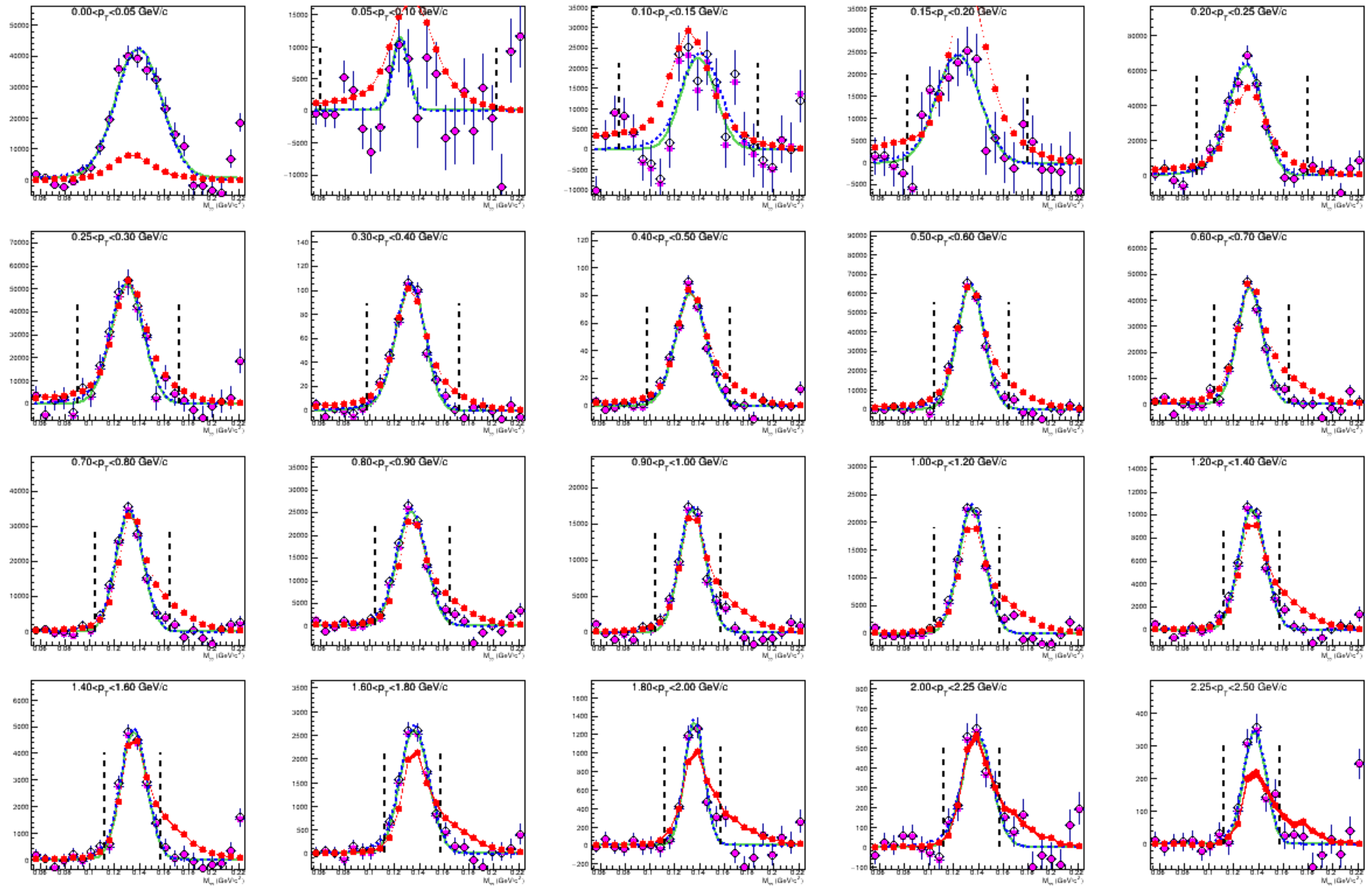
Example of ratio of Real/Mixed inv. Mass distributions UQRMD, 120 000 events (thanks Viktor)



Real: photons from same event; Mixed: photons from different events

Green/blue lines: parameterizations with different peak shapes

Example of Signal=Real-Mixed inv. Mass distribution UQRMD, 120 000 events (thanks Viktor)



Magenta (black) points: Signal=Real-Mixed

Red points: pairs with common ancestor pi0

Class MpdEmcSimParams

```
bool fSmearLightCollection = true;             // Emulate light smearing
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;                 // Width 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
bool   fMultiSectorClusters = false;          // allow clusters with digits in different sectors
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.;           // max. time difference btwn 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)
float fNonLinCorrection[2] = {0.0269775, 3.07082} ;   //Parameters for Nonlinearity correction: Ecorr=[0]+[1]*E
float fZcorrSinA[2] = {-0.5588, -0.042};  // Parameters for cluster position correction (See MpdEmcCluster::CorrectVertex)
float fZcorrSinW[2] = {52.7, 0.4};           // Parameters for cluster position correction
float fZcorrA[2] = {0.07887, 0.0101};        // Parameters for cluster position correction
float fZcorrB[2] = {0.00256, 0.00032};       // Parameters for cluster position correction
```

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

```
MpdEmcSimParams * par =
MpdEmcSimParams::GetInstance() ;
par->fSmearLightCollection = false;
par->fSimulateNoise = false ;   //Simulate el. noise
par->fApplyNonlinearity = false;  //Apply E non-lin.
par->fApplyDigitization = false ;   // Apply E digitization
par->fZSthreshold=0.;
par->fEdepToLightYield=40000;
par->fElectronicNoiseWidth=0.005*0.34;
par->fADCWidth = 0.005*0.34;
par->fUnfoldClusters=true ;
```

Conclusions

- Status:
  - Code committed to git (Thanks, Alexandr)
  - To be tested by community
- Todo:
  - Implement and test new geometry
  - **Validate MC simulations with beam-test data in consistent configuration**
    - Single electron energy resolution and single electron non-linearity
      - Implement/fix realistic response parameters once beam-test results will be available
    - Check shower shape for EM shower (electrons, $\pi^- + A \rightarrow \pi^0 + A \rightarrow 2\gamma+X$)
      - Optimize Dispersion cuts
    - Implement realistic time resolution (need parameterization from beam-test)
  - Test options of track matching
  - Implement (de-) calibration, mis-alignemnt, bad map
    - Interface with calibration database?
  - Develop class for realistic analysis using clever mixing
    - Develop general analysis manager?
  - More requests?