Track-EMC Matching

SuperB FastSim Meeting
2009/03/11

Chih-hsiang Cheng
Caltech

Introduction

- Track-EMC matching is based on track DOCA with respect to the cluster centroid.
  - There are many matching methods available in Babar. I don't know exactly what is used and did not investigate how each one is done.
- The template for track-emc match: `AstSTLMatch`
  - a template <class Key, class Value, class Quality>
    - which uses container std::map<Key*, std::map<Value*, Quality*> >
- Two maps are used:
  - AstSTLMatch<AbsRecoCalo, TrkRecoTrk, PacEmcTMInfo> >
  - AstSTLMatch<TrkRecoTrk, AbsRecoCalo, PacEmcTMInfo> >
- `PacEmcTMInfo` is a simplified version of Babar's `EmcTMInfo`.

PacEmcTMInfo

- Containing track-emc matching quality. Members:

```
protected:

  HepPoint _entryPoint;
  mutable double _deltaXY, _deltaZ;
  const EmcTrkInterInfo    _trkInterInfo;
  mutable const TrkRecoTrk    * _track;
  mutable int                   _charge;

  const AbsRecoCalo   * _calo;

  Consistency _consistency;

private:

  TrkPocaBase* _poca;
```

- The main difference from EmcTMInfo is that the latter can store consistency for each track pdt hypothesis.
  - my interpretation of the code, may not be accurate.

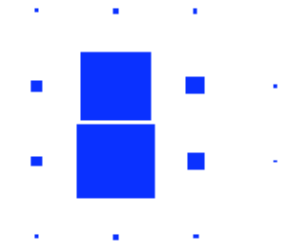
Uncertainty in DOCA

- The uncertainty is dominated by EMC clusters. We totally ignore the tracking error here (to save time).
- The resolution is determined cluster by cluster, unlike in Babar's `EmcPocaMatchMethod.cc`, which has hard-coded resolution functions.

Cluster resolution

- Add a function (returning a 2x2) `HepSymMatrix PacEmcCluster::secondMomentMatrix()` to calculate the RMS in θ, φ and their correlation, taking into account the crystal width.
  - $m_{00} = \Sigma_i\, w_i\, [\theta_i^2 + \Delta\theta_i^2/\sqrt{12}]$
  - $m_{11} = \Sigma_i\, w_i\, [\varphi_i^2 + \Delta\varphi_i^2/\sqrt{12}]$
  - $m_{12} = \Sigma_i\, w_i\, \theta_i\varphi_i$; $w_i = E_i/E_{total}$.
- The χ^2 of the match is
  - $\chi^2 = x^T M^{-1} x$, where $x = \begin{pmatrix} d_\theta \\ d_\phi \end{pmatrix}$

difference between track poca and cluster centroid



0.0009164 0.0000136
0.0000136 0.0015422

0.0011359 0.0002231
0.0002231 0.0017784

PmcTrkClusterMatch module

- Run after PmcReconstruct (and PmcCaloSplitMerge)
- Loop over track and calo lists. For each AbsRecoCalo-TrkRecoTrk pair,

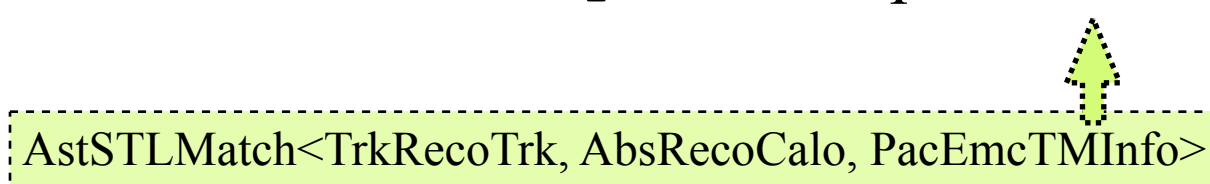
```
const TrkFit* fit= trk->fitResult();
TrkPocaXY poca(fit->traj(), fit->endFoundRange(),poscalo);
double flightlen= poca.flt1();
HepPoint trkPos= fit->traj().position(flightlen);
//...
da[0]= deltaTheta;
da[1]= deltaPhi;
int ierr;
const HepSymMatrix invm2mat = m2mat.inverse(ierr);
double chisq= (da.T() * invm2mat * da)[0];

if ( sqrt(chisq) < _maxSeparation.value() ) {
   EmcTrkInterInfo interinfo(flightlen,fit,trk);
   ChisqConsistency con(chisq,2);
   PacEmcTMInfo* tminfo= new PacEmcTMInfo(trkPos,fit->charge(),interinfo,
calo, deltaPhi, deltaTheta);
   tminfo->setConsistency(con);
   tminfo->setPoca(poca);
   emcTrkMap->insertMatch(calo, trk, tminfo);
}
```

cluster centroid (on the surface of the 2D fast sim model

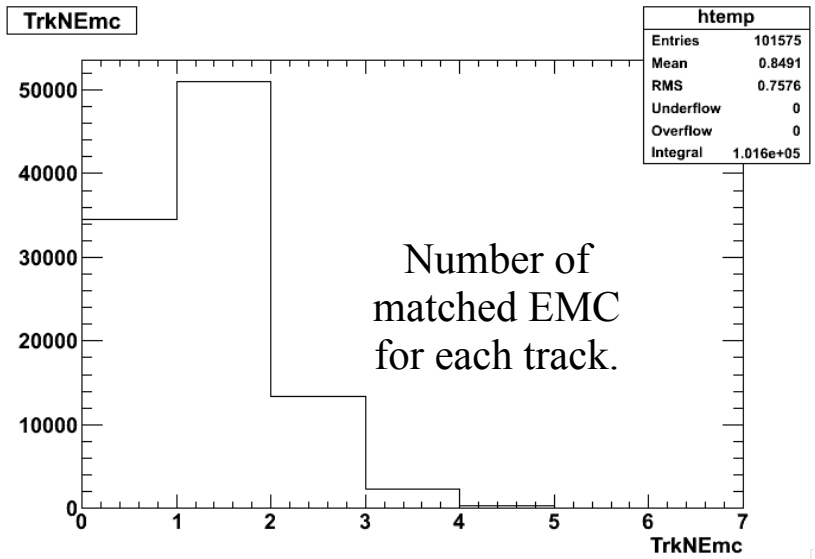
AstSTLMatch<AbsRecoCalo, TrkRecoTrk, PacEmcTMInfo>

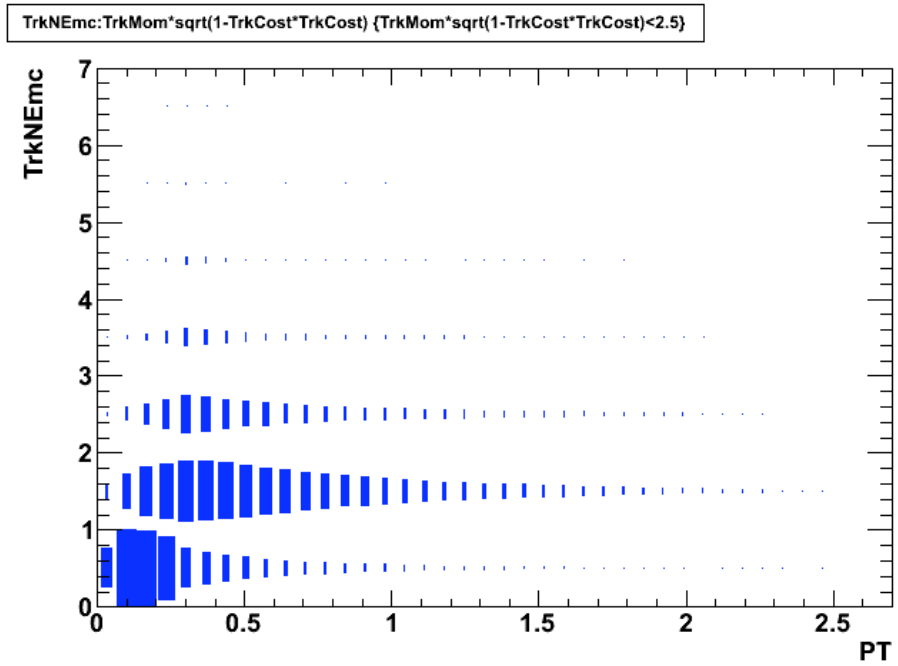
PmcTrkClusterMatch module

- Continue...
  - Going through elements in `emcTrkMap` and fill up `trkEmcMap`

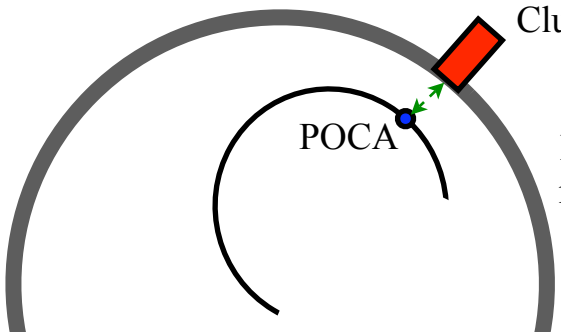


Test with B0B0bar generic

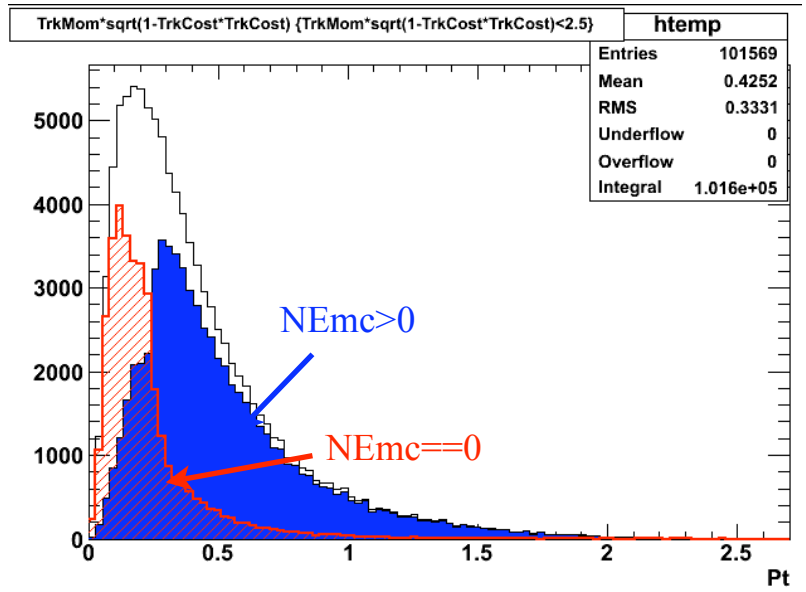




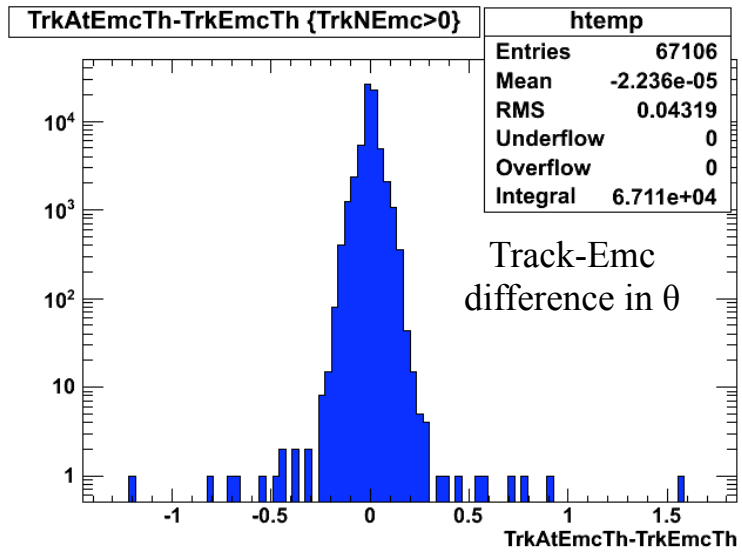
- Max allowed $\chi^2 = 25$.
- Don't understand why some high-pt tracks do not have a match.
- Some very low pt do have a match, probably because...

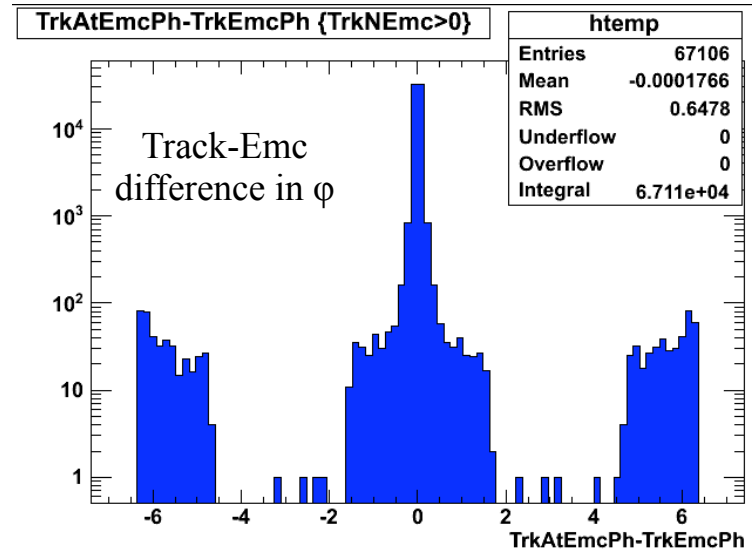


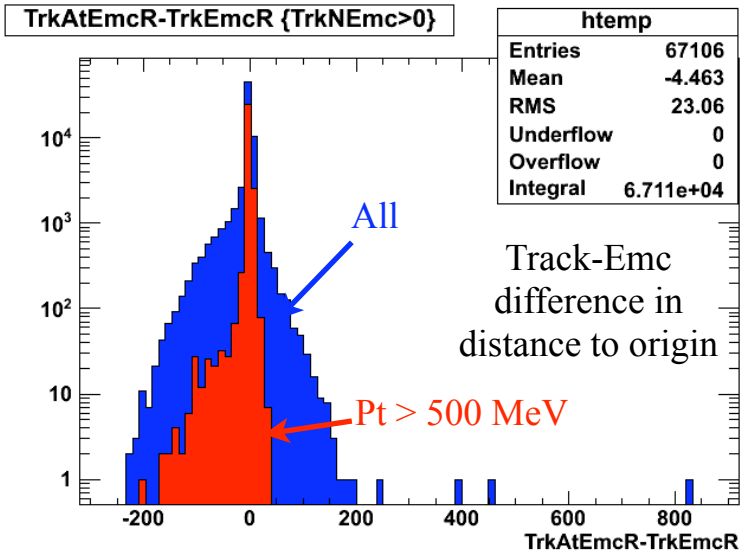
Distance in R not taken into account



Track-EMC residuals







- Δθ looks normal.
- Δφ has long tails (low momentum charged particles leave a long cluster along φ.
- Many have large distance in R, at least partially due to the reason explained in the previous page.

Conclusions

- Track-EMC matching basically works.
- Need to add distance information to the χ^2 (what is the uncertainty?)
- How do we decide the χ^2 cut?
- Because we model the EMC in 2D, a track has to reach the center (in radius) of the EMC to create a cluster. But in reality a lower pt track can create a cluster as long as it reaches the EMC inner radius.
  - How big is this problem?
  - Should we solve this by making EMC multi-layer with the first (thin) layer near the inner radius?