# Tracking validation update

Lia Lavezzi

2021-05-20

# Useful functions

- added to the algorithm class TestHoughTrack

```
int GetMdcRecoHitID( int mc_point_id, SmartDataPtr< Event::MdcMcHitCol > mdc_MC_point_Col,
                     SmartDataPtr< MdcDigiCol > mdc_digi_Col,
                     SmartDataPtr< RecMdcHitCol > mdc_hit_Col);

int GetMdcMCHitID( int reco_hit_id, SmartDataPtr< Event::MdcMcHitCol > mdc_MC_point_Col,
                   SmartDataPtr< MdcDigiCol > mdc_digi_Col,
                   SmartDataPtr< RecMdcHitCol > mdc_hit_Col);

int GetMdcRecoHitID( RecMdcHit *hit_in_vector, SmartDataPtr< RecMdcHitCol > mdc_hit_Col);
```

**MDC**

```
int GetCgemMCHitID( int cluster2d_id,
                    SmartDataPtr< Event::CgemMcHitCol > cgem_MC_point_Col,
                    SmartDataPtr< RecCgemClusterCol > cgem_cluster_Col);

int GetCgemCluster2dID( int mc_point_id,
                        SmartDataPtr< Event::CgemMcHitCol > cgem_MC_point_Col,
                        SmartDataPtr< RecCgemClusterCol > cgem_cluster_Col);
```

**CGEM**

```
int GetMCTrack( RecMdcTrack *hough_track,
                SmartDataPtr< Event::MdcMcHitCol > mdc_MC_point_Col,
                SmartDataPtr< MdcDigiCol > mdc_digi_Col,
                SmartDataPtr< RecMdcHitCol > mdc_hit_Col,
                SmartDataPtr< Event::CgemMcHitCol > cgem_MC_point_Col,
                SmartDataPtr< RecCgemClusterCol > cgem_cluster_Col,
                std::vector< int > &associated_mc_track,
                std::vector< int > &associated_nmdc,
                std::vector< int > &associated_ncgem);
```

**track**

Probably can be ported to a service class

# MDC - related

Retrieve **RecMdcHit** *index* inside **RecMdcHitCol** starting from the **MdcMcHit** *index*

```
int GetMdcRecoHitID( int mc_point_id, SmartDataPtr< Event::MdcMcHitCol > mdc_MC_point_Col,
                     SmartDataPtr< MdcDigiCol > mdc_digi_Col,
                     SmartDataPtr< RecMdcHitCol > mdc_hit_Col);
```

Retrieve **MdcMcHit** *index* inside **MdcMcHitCol** starting from the **RecMdcHit** *index*

```
int GetMdcMCHitID( int reco_hit_id, SmartDataPtr< Event::MdcMcHitCol > mdc_MC_point_Col,
                   SmartDataPtr< MdcDigiCol > mdc_digi_Col,
                   SmartDataPtr< RecMdcHitCol > mdc_hit_Col);
```

# MC-to-reco match

**original plan**

1) loop on the MdcMcHitCol
int digiID = MdcMCHit → getDigiIdx()

2) get the MdcDigi

3) loop on RecMdcHitCol
int recoID = RecMdcHit → getId()
match MdcDigi index with recoID

BUT RecMdcHit : : getId does not provide
- the index inside the RecMdcHitCol
- the index of the MdcDigi

It provides the index of the reco hit inside the list of RecMdcHit associated to the track

**workaround**

```
1) I retrieve the MC POINT (MdcMcHit) from the collection
-> from MdcMcHit *mdc_mc_hit I get:
- the identifier:
    Identifier identifier1 = mdc_mc_hit->identify();
- which layer/wire it belongs to:
    int id_layer1 = MdcID::layer(identifier1);
    int id_wire1 = MdcID::wire(identifier1);
- the index of the corresponding digi:
    int digi_id = mdc_mc_hit->getDigiIdx();

2) I retrieve the DIGI (MdcDigi) from the collection
-> from MdcDigi *digi I get:
- the identifier:
      Identifier identifier2 = digi->identify();
- which layer/wire it belongs to:
      int id_layer2 = MdcID::layer(identifier2);
      int id_wire2 = MdcID::wire(identifier2);

3) I loop over all the RECO HIT (RecMdcHit) in the collecti
-> for each RecMdcHit *hit I read:
- the identifier:
      Identifier identifier3 = hit->getMdcId();
- which layer/wire it belongs to:
      int id_layer3 = MdcID::layer(identifier3);
      int id_wire3 = MdcID::wire(identifier3);
```

4) The matching is done with the comparison of layer and wire from the identifier

Analogous procedure for the reco-to-MC association

# MDC - related

Retrieve **RecMdcHit** *index* inside **RecMdcHitCol** starting from the *index* in *hitvector*

```
int GetMdcRecoHitID( RecMdcHit *hit_in_vector, SmartDataPtr< RecMdcHitCol > mdc_hit_Col);
```

This was necessary since RecMdcHit : : getId does not provide neither the index inside the RecMdcHitCol nor the index of the associated MdcDigi, but just the index inside the hitvector, i.e. the index of the reco hit inside the list of RecMdcHit associated to the track

```
// HOUGH TRACK
RecMdcTrackCol::iterator iter_hough_track = hough_track_Col->begin();
for(iter_hough_track = hough_track_Col->begin(); iter_hough_track != hough_track_Col->end(); iter_hough_track++) {
  RecMdcTrack *hough_track = (RecMdcTrack*) (*iter_hough_track);
//
// hit vector ---------------------------------
HitRefVec hitvector = hough_track->getVecHits();
ClusterRefVec clustervector = hough_track->getVecClusters();
```

# CGEM - related

Retrieve **RecCgemCluster** *index* inside **RecCgemClusterCol** starting from **CgemMcHit** *index*

```
int GetCgemCluster2dID( int mc_point_id,
                        SmartDataPtr< Event::CgemMcHitCol > cgem_MC_point_Col,
                        SmartDataPtr< RecCgemClusterCol > cgem_cluster_Col);
```

Retrieve **CgemMcHit** *index* inside **CgemMcHitCol** starting from **RecCgemClusterCol** *index*

```
int GetCgemMCHitID( int cluster2d_id,
                    SmartDataPtr< Event::CgemMcHitCol > cgem_MC_point_Col,
                    SmartDataPtr< RecCgemClusterCol > cgem_cluster_Col);
```

# CGEM MC-to-reco

1) get cluster 1d x/v associated to RecCgemCluster 2d

```
//  cout << "GET CGEM MC POINT ID " << cluster2d_id << endl;
RecCgemClusterCol::iterator iter_cgem_cluster = cgem_cluster_Col->begin() + cluster2d_id;
RecCgemCluster *cluster = (*iter_cgem_cluster);
int id_x1 = cluster->getclusterflagb();
int id_v1 = cluster->getclusterflage();
```

2) loop over the collection of CgemMCHit

```
// --> loop over cgem MONTECARLO points and association of the reco track to MC track
Event::CgemMcHitCol::iterator iter_cgem_MC_point;
int counter=0;
for(iter_cgem_MC_point = cgem_MC_point_Col->begin(); iter_cgem_MC_point != cgem_MC_point_Col->end(); iter_cgem_MC_point++) {
  Event::CgemMcHit *mc_hit = (*iter_cgem_MC_point);
```

3) for each CgemMCHit retrieve the indices corresponding to the RecCgemCluster 1d x/v

```
  vector<int> xclusteridxvector = mc_hit->GetXclusterIdxVec();
  vector<int> vclusteridxvector = mc_hit->GetVclusterIdxVec();
  int id_x2 = xclusteridxvector.at(0);
  int id_v2 = vclusteridxvector.at(0);
```

4) match the RecCgemCluster to the CgemMcHit if

- id_x1 == id_x2
- id_v1 == id_v2

# CGEM MC-to-reco

FIRST ERROR

```
Event::CgemMcHit *hit = (*iter_cgem_MC_point);
vector<int> digiidxvector = hit->GetDigiIdxVec();
vector<int> xclusteridxvector = hit->GetXclusterIdxVec();
vector<int> vclusteridxvector = hit->GetVclusterIdxVec();
```

BUT in some events I have CgemMcHit which has no associated RecCgemCluster

```
---------------------------------------------- EVENT 34 RUN -9989 is error 0
ERROR_13 mc point 7 is not associated to a reco, i.e. xvec 0 vvec 0
ERROR_13 mc point 8 is not associated to a reco, i.e. xvec 0 vvec 0
```

I put some printouts in **CgemClusterCreate : : ToyCluster**

# ToyCluster

I put some printouts in **CgemClusterCreate : : ToyCluster**

```
mc point 3
creatorProcess Generator
getXStripID 676
X_ID 676 V_ID 707
associate idx 6
associate idv 7
mc point 4
creatorProcess Generator
getXStripID 363
X_ID 363 V_ID 638
associate idx 8
associate idv 9
mc point 5
creatorProcess Generator
getXStripID 478
X_ID 478 V_ID 665
associate idx 10
associate idv 11
mc point 6
creatorProcess Generator
getXStripID 450
X_ID 450 V_ID 543
associate idx 12
associate idv 13
mc point 7
creatorProcess PionMinusInelastic
mc point 8
creatorProcess PionMinusInelastic
```

It happens in the cases where the process is neither a Generator nor a Decay

Specifically this is for Inelastic scattering of negative pions

Is it correct that these associations are missing?

# CGEM MC-to-reco

## SECOND ERROR

```
---------------------------------------------- EVENT 13 RUN -9989 is error 0
ERROR_4a id_x1 6 == id_x2 6 but id_v1 13 != id_v2 7
ERROR_4b id_x1 6 != id_x2 12 but id_v1 13 == id_v2 13
ERROR_8a (WARNING) cgem reco hit not connected to cgem mc point
ERROR_4a id_x1 10 == id_x2 10 but id_v1 17 != id_v2 11
ERROR_4b id_x1 10 != id_x2 16 but id_v1 17 == id_v2 17
ERROR_8a (WARNING) cgem reco hit not connected to cgem mc point
```

Can this be due to cases where inside the same event two CgemMcHit fire on the same strips?

How are these cases handled in ToyCluster?