Update to add phi and z energy weighted centroids for truth track hits in the TPC

David Stewart

2022.07.25

Output: for each event: std::map<[track-id],[energy centroid]>

- Looks like:
- -----TrkrHitTruthClustersv1-----
- Number of associations: 5
- Centroids for track 2
- layer: 7 phi (-1.58466 pm 0.00222725) z (4.21672 pm 0.186501) sum_E (14)
- layer: 8 phi (-1.58211 pm 0.00197996) z (4.32955 pm 0.129447) sum_E (8)
- layer: 9 phi (-1.57796 pm 0.00213988) z (4.38344 pm 0.142205) sum_E (42)
- layer: 10 phi (-1.57568 pm 0.00206769) z (4.39438 pm 0.160166) sum_E (36)
- layer: 11 phi (-1.5735 pm 0.00180131) z (4.46925 pm 0.144059) sum_E (29)
- layer: 12 phi (-1.57101 pm 0.00290951) z (4.50531 pm 0.131201) sum_E (23)
- layer: 13 phi (-1.56867 pm 0.00138582) z (4.49897 pm 0.0937207) sum_E (5)
- layer: 14 phi (-1.56516 pm 0.00214332) z (4.63785 pm 0.167533) sum_E (11)
- layer: 15 phi (-1.5629 pm 0.00249072) z (4.678 pm 0.150239) sum_E (27)
- layer: 16 phi (-1.55964 pm 0.00199582) z (4.78149 pm 0.14152) sum_E (33)
- layer: 17 phi (-1.55686 pm 0.00176379) z (4.86455 pm 0.181018) sum_E (46)

…

Some details

- Sum_E is just the number of electron hits from gas amplication in the GEMS in the simulation/g4simulation/g4tpc/PHG4TpcElectronDrift.cc code
- Code keeps the PHG4TpcElectronDrift.cc loop: loops through all PHG4Hits, checks each one if it I belongs to an embedded track
- Assumes that hits for embedded tracks will be stored contiguously in the node tree
- Modified return in simulation/g4simulation/g4tpc/PHG4TpcPadPlane.h::MapToPadPlane to unsigned int in order to return the gem pad row layer number

Merge request with Github

- **Truth emb energy centroids in phi and Z in GEMS for embedded tracks #1588**

Code question:

- Inside of the TrkrHitTruthClusters.h added a struct for Energy Centroids:

```
struct EnergyCentroid {
  short layer_id;
  float phi_ave, phi_stdev, z_ave, z_stdev, sum_E;
  EnergyCentroid( short _layer_id, std::array<float,5> input) :
    layer_id  {_layer_id},
    phi_ave   {input[0]},
    phi_stdev {input[1]},
    z_ave     {input[2]},
    z_stdev   {input[3]},
    sum_E     {input[4]} {};
  EnergyCentroid() : layer_id{0}, phi_ave{0}, phi_stdev{0}, z_ave{0}, z_stdev{0}, sum_E{0.} {};
  void set_values( short _layer_id, std::array<float,5> input) {
    layer_id = _layer_id;
    phi_ave   =input[0];
    phi_stdev =input[1];
    z_ave     =input[2];
    z_stdev   =input[3];
    sum_E     =input[4];
  }
  ~EnergyCentroid() {};
};
```

- Used in TrkrHitTruthClusters.h

```
class TrkrHitTruthClusters : public PHObject
{
  public:

  //! typedefs for convenience
  static const int N_GEM_LAYERS = 55;
  /* using CentroidsFor1Track = std::array<EnergyCentroid, N_GEM_LAYERS>; */
  using VecEC = std::vector<EnergyCentroid>; // the data
```

Code question:

- Made helper class simulation/g4simulation/g4tpc/TrkrTruthCentroidBuilder.{h,cc}, used in PHG4TpcElectronDrift.cc
- Will this work with the overall sPHENIX code structure?

Use question:

- What kind of accessors/pointer getters do we want to get at the energy centroid data? For now the user just can get:
- Keys from the map to which tracks have energy centroid data
- From map, vectors of the centroids, which each contain the layer-id. The vectors are always sorted from smallest layer-id to largest
  - At a minimum should implement a comparison to unsigned int for EnergyCentroid in order to use a std::binary_search to check for individual pads