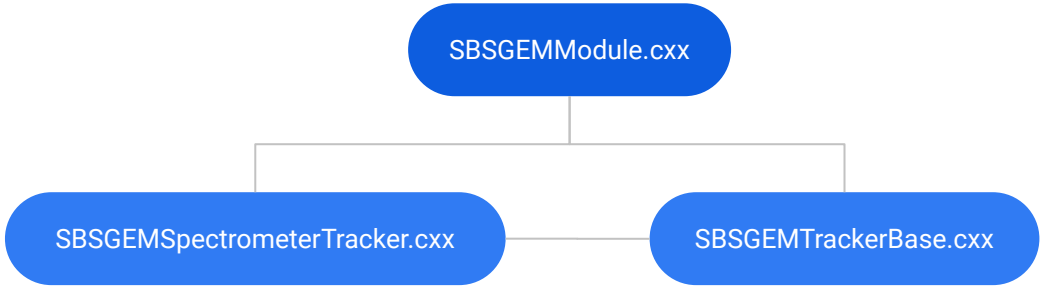
SBS-Offline GEM Analysis Overview

Sean Jeffas

University of Virginia

SBS Student Meetings June 27, 2022

GEM File Structure



- SBSGEMModule.cxx processes the raw data and saves raw strip information - 4,800 lines of code.
- SBSGEMTrackerBase.cxx takes the strip information and does clustering and tracking - 2,500 lines of code.
- SBSGEMSpectrometerTracker.cxx holds the main functions that call the other two files - 700 lines of code.

Raw Data Decoding

- **SBSGEMModule::Decode()** is 1,200 lines long and contains most of the important features.

1) Loop over all the raw data and calculate
   a) Possibly correct the common mode if the value is far from expectation.
2) Loop over all the raw data again.
   a) Check if a raw strip passes zero suppression.
   b) Record as a strip with a hit and all strip information is written to arrays.
      i) Check if strip passes varius timing cuts.
      ii) Set fKeepStrip to false if any single cut fails.

Hit Clustering

- **SBSGEMTrackerBase::hit_reconstruction()** performs the GEM clustering.

1) Form 1D clusters in each dimension.
   a) Loop over strip hit arrays and search for maximums in ADC.
   b) Save cluster information in an array with the class **sbsgemcluster_t**.
2) Form all possible 2D combinations from the 1D clusters.
   a) Save each individual combination in an array with the class **sbsgemcluster_t**.

Tracking

- **SBSGEMTrackerBase::find_tracks()** performs the GEM tracking.

1) Look at all possible 2D hit combinations and fit straight lines to get tracks (see algorithm on the next slide).
2) If a track passes the **fTrackChi2Cut** threshold and meets the minimum hits on layers requirement (**nhitsrequired**) then it added to the track variables in the tree, **SBSGEMTrackerBase::AddNewTrack()**
   a) Also must pass cuts on target and calorimeter constraints.

Tracking Algorithm

```
//The basic algorithm should do the following:

// 1. Loop over all possible layer combinations for which the number of layers is equal to the current minimum hit requirement:
// 2. Within each layer combination at the current minimum hit requirement, check that every layer has at least one free hit. If so, proceed:
// 3. For all possible combinations of one hit from each of the two outermost layers, calculate the straight line between the two points, and project the
//    straight line to each layer
// 4. At each intermediate layer, populate the list of hits in the grid bins pointed to by the candidate track. If the projected track is close to the edge of
//    the bin in either X or Y, also consider hits in the adjacent X and/or Y bins
// 5. IFF every intermediate layer contains at least one hit in the grid bin(s) sufficiently close to the projected track, proceed to loop over the
//    hits in the intermediate layers, using either the "fast" method (find the hit in each layer closest to the projected track) or the "brute force" method
//    (loop on all possible hit combinations in the intermediate layers using "odometer" algorithm), which is probably more accurate, but slower
// 6. For each candidate hit combination, calculate the chi2 of a straight line fit.
```