FTPC sub-system and the ITTF project

María Mora for the FTPC group

The FTPC group has been asked to look at the ITTF work in order to discuss with the ITTF crew if the actual design complies with the claims of the FTPC group and allows for future expansion, in particular the integration for tracking of new detectors as the Forward TPC.

1 Detector integration.

This is about the physical integration of the detector and the way to load hits for later tracking. The main task is to decide which part of the detector should be integrated and how this can be done using the available detector geometry, so called shapes. The integration was done creating the following classes: StiFtpcDetectorBuilder, StiFtpcDetectorGroup, StiFtpcHitLoader and StiFtpcIsActiveFunctor.

Detector –

This part is mainly coded in the classes StiFtpcDetectorBuilder and StiFtpcDetectorGroup. The idea is to create the different detector elements and the most important scattering centers (this will be used for the energy loss calculation). Each part will have a particular **shape** (StiShapes) from the actual StiShapes available: StiPlanarShape, StiCylindricalShape and StiConicalShape. They will be **placed** together and with the other detectors (using the StiPlacement) and they will be made of a particular **material** (StiMaterial). For the placement of the different parts of the detector in the z coordinate we use the values from the FTPC geometry which are set in the database. This allows us to propagate changed values immediately and easily.

Hit Loader –

This part is coded in the StiHitLoader class. The idea is to take the hits (at the moment this is done from hits in a event.root file where hits are stored in global coordinates) and load them to be able to start tracking. For the FTPC code in ITTF, the detector is placed where it theoretically should be, that means, without all the slight rotations in both chambers. To keep things consistent we decided to apply the rotations at the hit level. In StEvent the hits are stored in the global coordinate system and, in this first stage of the ITTF program (when hits are read from the StEvent), they will be transformed to local coordinate in the StiHitLoader class. For this we used the already existing function StFtpcTrackMaker/StFtpcPoint::TransformGlobal2Ftpc(). This function is based on the database values for the rotation which allows an easy propagation of rotation changes.

This task is already completed but not fully tested. Personally I would say that the code is flexible and easy to understand (only C++ knowledge is needed). In general, it is clear that four tasks must be completed for every sub-system detector: the creation of different objects from the class StiShape, the creation of a StiMaterial object for each shape, the placement of the shapes and the load of the hits. It was easy to understand the different parameters for the different kind of StiShapes and I think that this code is flexible enough to be used for other detectors. Anyway one could always include new shapes if needed without a big effort. The

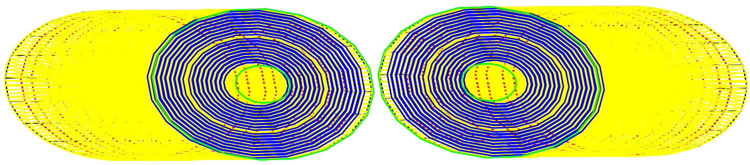

Figure 1: FTPC detector and scattering centers included in the ITTF code: the gas volumes divided in sectors and pad-rows (in yellow), the inner field-end rings in aluminum (in blue) and the inner mylar cap (green).

use of the StiMaterial class is obvious. My only personal problem was trying to understand the different kind of coordinate systems used for placing the StiShape objects and for assemble all the pieces. This was not straight forward and a better description should be provided to the users. In my opinion, no more that a week will be needed to complete the integration of any detector.

2 Tracking in the detector

I will ask some question about the ITTF code and I will raise some points to make sure that the future tracker for STAR will be ready to track in other detectors as for example the Forward TPCs.

Seed Finder –

- Does the ITTF code allow for different seed finder parameters for each sub-system?

A FTPC seed finder will obviously use different parameters from the ones in the TPC. Due to the geometry of the detector and its position: the position of the rows changes in the z direction, with a bigger distance in between (8.5 cm and 12.5 cm) than for the TPC, the number of rows is limited at 10, the density of tracks at forward rapidity can be very high, and due to the distance from the vertex, all tracks (primary and secondary) will be directed to the collision vertex in a first approximation. So parameters such as number of points in a seed, size of volume where to look for a next hit and even the direction where to look for the next hit in the seed should be different for every sub-system and easily accessible. The ability of having a different set of parameters when working with laser tracks could be necessary.

The actual seed finder in the FTPC code starts looking for hits in the region where the density is lower (the outest pad-row at higher radius). The next hit is look for in the next pad row, in the straight line vertex direction. Our actual seeds have up to three hits. For laser tracks no helix approximation is used during the tracking stage. Instead the seeds are extended to up to 10 hits.

In case the ITTF seed finder with optimized parameters for the FTPC does not do better than the actual seed finder :

- Does the ITTF code allow for the use of a different seed finder for every sub-system?

Track Finder –

- Does the ITTF code allow for he use of different track finder parameters for every sub-system?

For the same reason as for the seed finder the parameters for the track finder will be different from the TPC ones. The code should allow to change it or to add new kinds of parameters: the minimum or maximum number of hits in a track, the number of rows allowed to be skipped, size of volume where to look for a next hit, etc.

The actual track finder in the FTPC code has a maximum and minimum number of hits of 10 and 5 and two rows can be skipped without aborting the track. Each Ftpc chamber is divided in 200000 three dimensional volumes (10 divisions in the z dimension, 100 divisions in phi and 200 divisions in eta). Starting from a seed the next hit on track is searched for in the nine (3X3) adjacent volumes, in the next pad-row. Inside this volume it will search for the nearest hit to a helix: a straight line in track length vs. z coordinate and a circle in x and y coordinates. I want to point out that the segmentation in eta and phi angle implies some vertex constraint.

In case the ITTF track finder with optimized parameters for the FTPC does not do better than the actual track finder :

- Does the ITTF code allow for the use of a different track finder for every sub-system?
- The ITTF tracker should be able to track in real data (with and without magnetic field) and with laser data.

3 Special requirement from the FTPC

There are two special needs for the FTPC calibration and FTPC tracking that should be taken into account by the ITTF group:

Vertex determination –

In order to complete the FTPC calibration we need the ability of vertex determination using only FTPC tracks (using tracks from both chambers, only FTPC east tracks, only FTPC west tracks, and for all FTPC sector independently). The ITTF code must be able to provide these features.

Track Fitter –

Unfortunately the magnetic field in the FTPC chambers is not homogeneous. A correction for this at the track level is needed. The ITTF code must be able to include in its fitting routine a correction for this modification.

Actually this is done in the track fitter as following:

The actual correction is done in order to preserve the use of a a perfect helix for the fit and a homogeneous field assumption. They way to do this is shifting the hits before fitting. With a created track and assuming a homogeneous magnetic field, a helix is fitted to the track and a momentum is therefore calculated. This leads to the calculation of the particle velocities in the plane of the pad-rows. Then, the Lorentz force can be calculated at each point for the real (non homogeneous) field. In this way, every hit will obtain a different shift (in x and y, I think z shift is neglected) due to that force. Since the measured hits already experienced this force, we shift each point back by the value gotten from the calculation, assuming that the effect is almost the same at the position where we found the hit and where it was actually produced. In the end, we have a track consisting of all these shifted points. Now we apply the helix fit again and this value of momentum is the one which we take as the correct momentum.