Tracking-efficiency in TPM1

Bjørn H. Samset

September 29, 2000

A BRAHMS-note describing efforts by the Oslo-group to estimate the efficiency of the two TPC tracking-modules for TPM1. The Oslo-group currently consists of Trine Tveter, Truls M. Larsen and Bjørn H. Samset.

Contents

1 Introduction

In this brief note I will give a preliminary estimate of our tracking-efficiency in the first mid-rapidity TPC (TPM1). The data are based on an algorithm inserting a geant-track in a real event and then trying to reconstruct it - see section 2.

Please note that this is all preliminary - a more thorough analysis (hopefully including an estimate of primaries to secondaries and $\frac{dN}{d\eta}$from TPM1) is under production.

2 The efficiency-algorithm

The efficiency-estimate is generated by the following algorithm: I have thrown 4 GeV pions through TPM1 (using gbrahms) with an angle-spread of ± 8 degrees in the global xz-plane and ± 3 degrees in the xy-plane. This corresponds to a section 31.8 cm by 11.8 cm at the back of the TPC, so I have not yet tested the very edges of its acceptance. This will be done in the next round to look for edge-effects etc.

For each event in the analysis, I digitize one of these tracks and run the clusterfinder on the results. This gives me 12 clean clusters (which at this point means BrDetectorHit-objects) in a straight line, which I then substitute with real BrDetectorHits taken from events in a different sequence. This is done to get a realistic uncertainty-distribution in cluster-positions, as this uncertainty and the centroid-position is all that is passed on to the trackfinder.

I then add these hits to the ones found from the real data, do trackfinding, and see if I have reconstructed the geant-track. A "good" reconstruction is deemed to be one where exactly one of the found tracks has an overlap with the geant-track of more than 80% (as given by the algorithm RelativeOverlap in BrTrackFinderBase). A figure showing why I chose this cut is shown in Figure 1.

Efficiency-results can be found in section 4. I have also checked to see if this adding of clusters causes problems for the trackfinders - see section 5. (For the experts: My cylinder-radius in RelativeOverlap was set to 0.2 cm.)

Note that while this approach is probably a bit simplistic, it gives a good upper limit on our efficiency. In the next round, we will do the track-adding on the TPCsequence-level using software now being developed by our new master-student Truls M. Larsen. This will hopefully give a more realistic figure, as we will then also test the clusterfinder.

3 The trackfinders

For those unfamiliar with our TPC-tracking, I will give a brief (and incomplete) overview of our two algorithms.

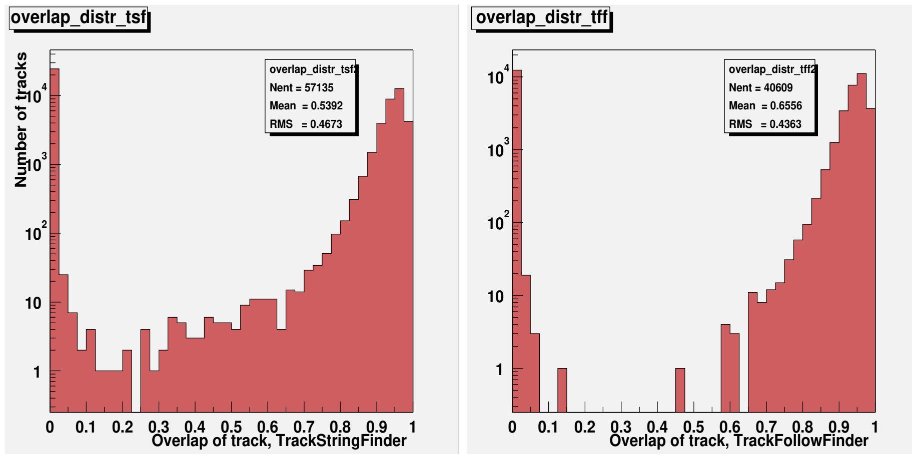

Figure 1: Distribution of overlap with geant-track for all tracks. Note that this is in log-scale. From these pictures, an overlap-cut of 0.8 was chosen.

3.1 BrTrackFollowFinder

BrTrackFollowFinder is a traditional “track-snooper”. It starts out with one cluster in the first padrow, and then searches for another within a defined region in the second. If it finds one it goes on to the third and does the same etc. If it finds more than one cluster in the allowed area it splits the track-candidate into two and searches on for both. The algorithm can miss a preset number of padrows and still mark the candidate as a good track.

3.2 BrTrackStringFinder

BrTrackStringFinder sequentially combines all clusters in the back of the TPC with those in the front, and for each matching defines a cylinder between them and sees how many clusters are within this volume. If this number exceeds a certain value and we have hits in at least the minimum number of padrows the combination is deemed a good track.

3.3 BrTrackFinderBase

BrTrackFinderBase is the parent-class for both trackfinders. This is where ghost-busting and final selection of tracks is done, so the trackers really only differ in their method for finding probable track-candidates.

4 Efficiency-results

Below I present the current efficiency-estimates. For this I have analyzed run 2336 seq’s 0, 1, 2 and 3 for a grand total of ~35K events with MRS @ 90 degrees,

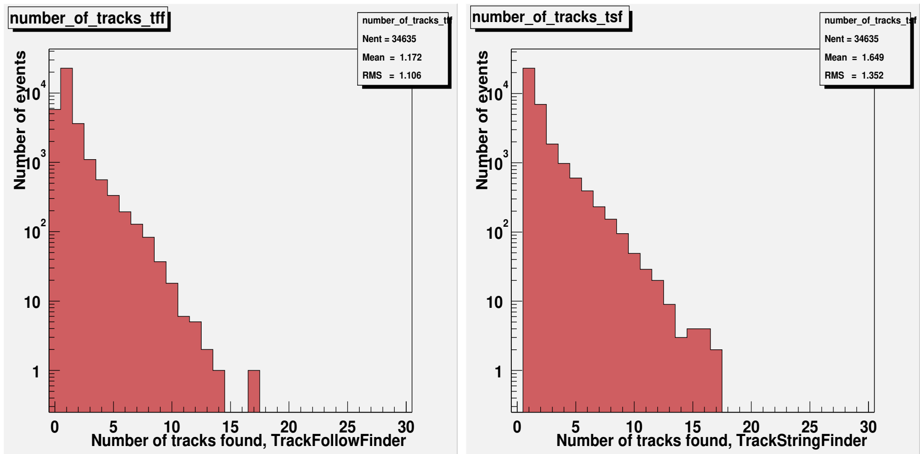

Figure 2: Number of tracks found in one event, INCLUDING the added geant-track.

and run 2513, seq's 0 and 1 totaling ~7K events with MRS @ 40 degrees. I have checked for dependencies on the angle of the geant-track and the multiplicity of the event (taken to be the SumADC from the tiles).

The lower statistics in the 40 degree-analysis is because of the high multiplicity in the TPC at this angle. The reconstruction-time increases several-fold.

4.1 MRS @ 90 degrees

First, please refer to figure 2, which shows the number of tracks the two algorithms found in each event. BrTrackStringFinder seems to find a slightly higher absolute number of tracks per event than BrTrackFollowFinder.

4.1.1 BrTrackFollowFinder

BrTrackFollowFinder reconstructed the geant-track in 28031 of the 34635 events, giving an overall efficiency of 80.9%.

Figure 3 shows first the dependency on multiplicity (row 1) and then on θ and ϕ of the geant-track (rows 2 and 3). The left column shows the total number of events in a light shade and the number of events with reconstructed geant-track in a dark shade, and the right column shows the ratio between these.

As can be easily seen, the efficiency does not seem to depend on either of these parameters - at least within this parameter-space. It may well be that we see edge-effects of the TPC if we look at tracks with steeper angles, but within this area the efficiency seems to be ~80% in all cases.

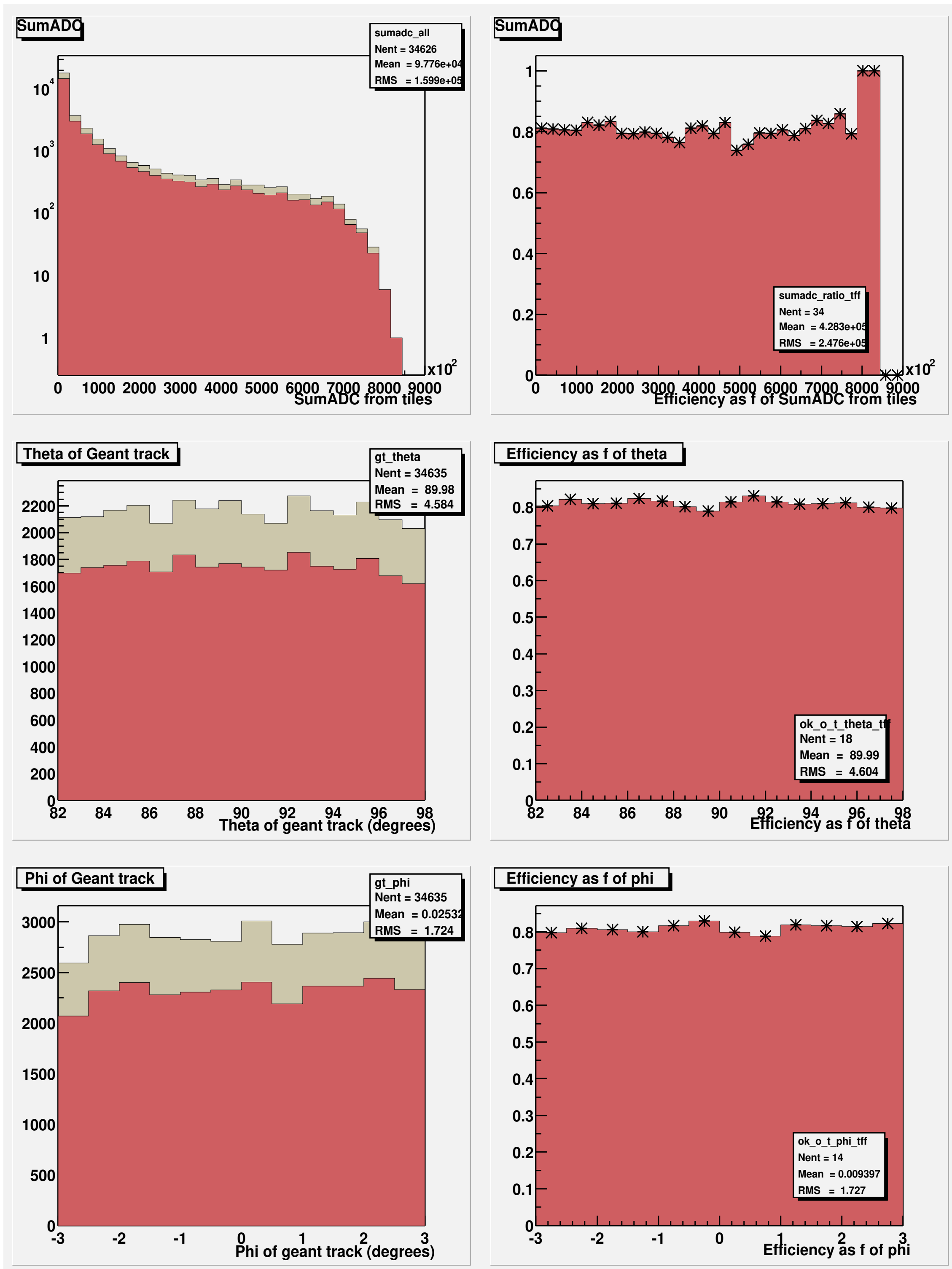

Figure 3: Efficiency-estimate, BrTrackFollowFinder, MRS @ 90 degrees

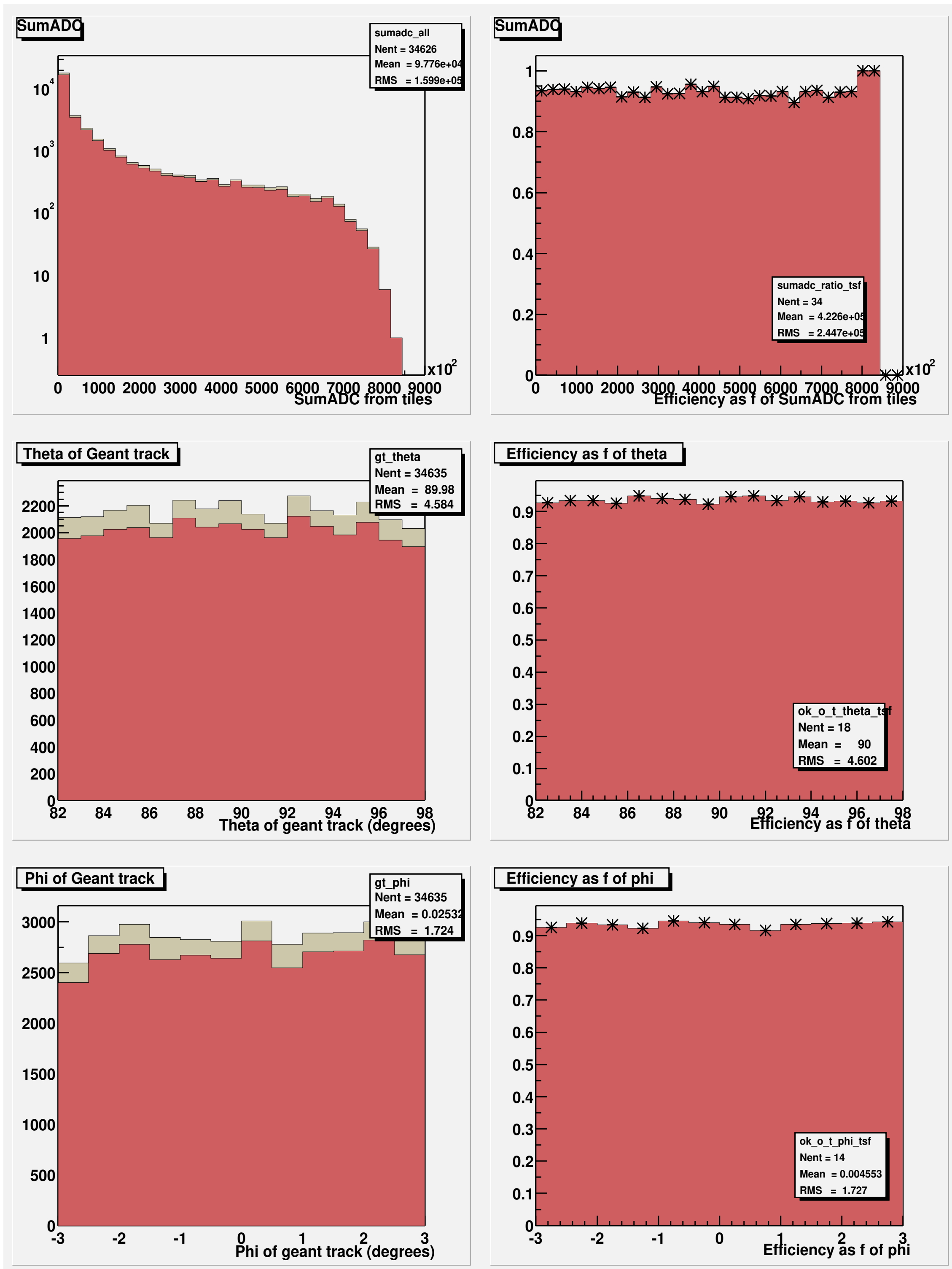

Figure 4: Efficiency-estimate, BrTrackStringFinder, MRS @ 90 degrees

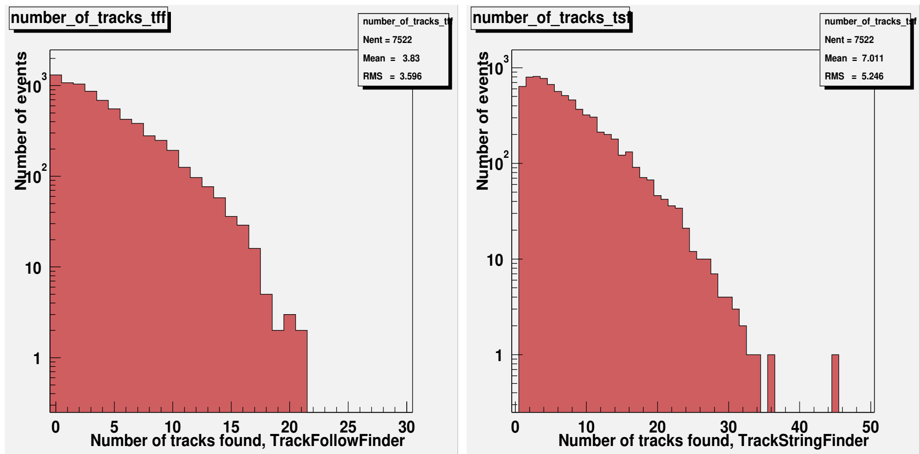

Figure 5: Number of tracks found in one event, INCLUDING the added geant-track.

4.1.2 BrTrackStringFinder

Figure 4 shows the same picture for BrTrackStringFinder. It reconstructed the geant-track in 32353 of the 34653 events, giving an estimated efficiency of 93.4%. The same observations as above apply in this case - I see a constant ratio of 92-94% for all angles and multiplicities.

4.2 MRS @ 40 degrees

Figure 5 shows how many tracks the two algorithms found in each event at 40 degrees. Now it is very clear that BrTrackStringFinder finds more tracks than BrTrackFollowFinder per event - note the different scales on the x-axis. Also, for BrTrackFollowFinder the distribution peaks at around ten tracks more here than at 90 degrees (see figure 2), which does not seem all that unreasonable.

4.2.1 BrTrackFollowFinder

Figure 6 shows that the picture at 40 degrees for BrTrackFollowFinder is roughly the same as for 90 degrees. Out of 7522 analyzed events the geant-track was found in 6180 cases, giving an estimated efficiency of 82.2%.

One notable exception is that in this case the efficiency seems to be multiplicity-dependent. It ranges from ~90% at low SumADC's to ~60% at the highest. This may be an effect of the lower statistics, but the tendency seems clear - we see a lower efficiency at the very high occupancy we get with TPM1 at such forward angles. This may also be partly helped if we reduce the search-width of the tracker.

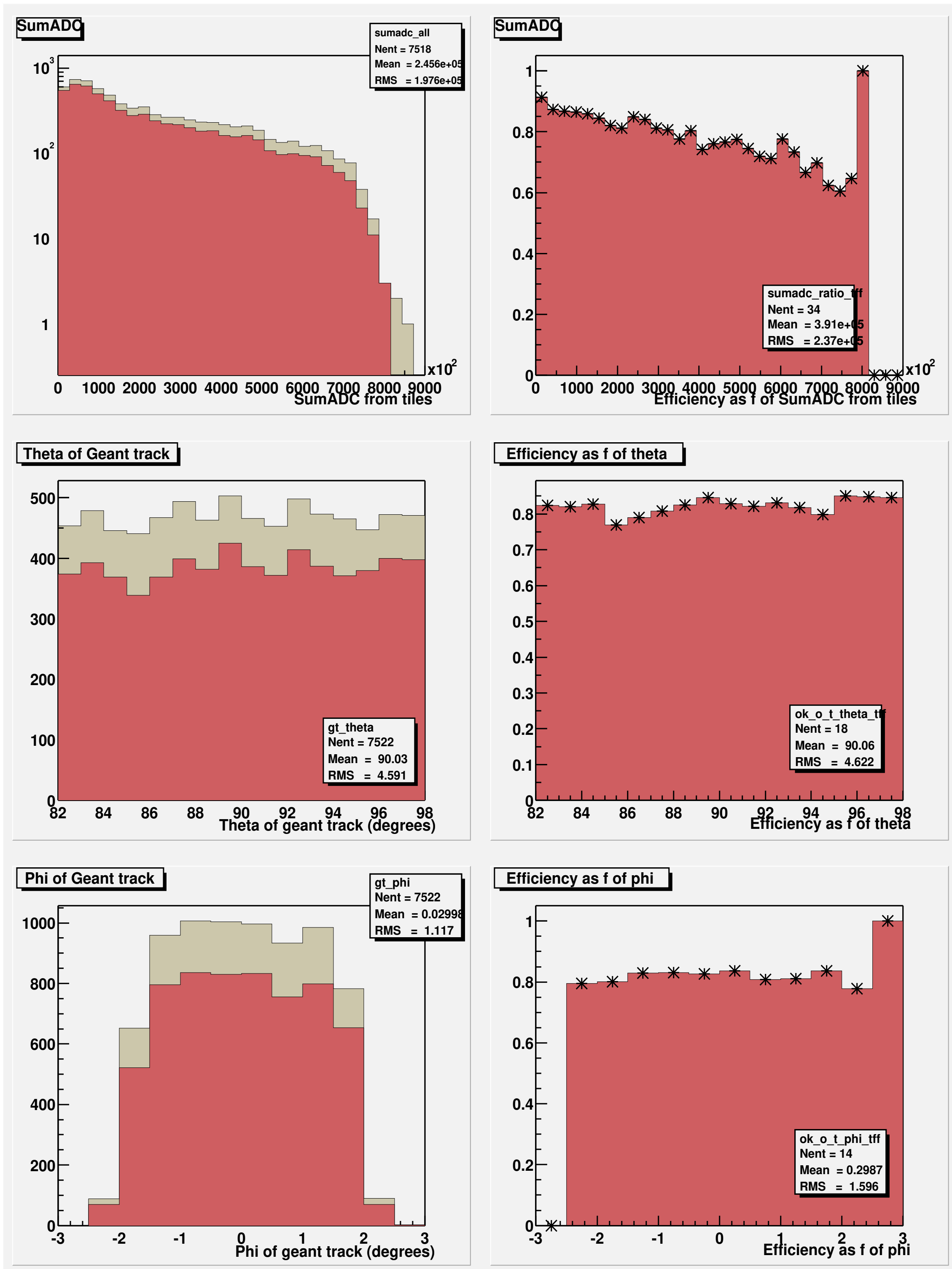

Figure 6: Efficiency-estimate, BrTrackFollowFinder, MRS @ 40 degrees

4.2.2 BrTrackStringFinder

In figure 10 BrTrackStringFinder has successfully reconstructed the geant-track in 7190 of the 7522 events, giving an estimated efficiency of 95.6% at 40 degrees. Here we again see no change with multiplicity or angle of the geant-track, so the efficiency-estimate can be said to be ~95% for all situations at this angle. See below for a comment on CPU-time, though.

4.3 Efficiency vs CPU-time

As can be seen above, BrTrackStringFinder performs 10-15% better than BrTrackFollowFinder under the same conditions. One problem for BrTrackFollowFinder is that there is still a lot of noise-clusters in TPM1, and this causes it to generate a huge amount of track-candidates. Sometimes this forces the algorithm to give up, so I expect the efficiency to rise if we can either clean up the TPC a bit or create a better handling of the track-candidates in the algorithm. We might also simply reduce the search-width a bit - this will be tried out soon.

The main problem with BrTrackStringFinder is that it is still very slow - it easily eats ten times the CPU-time of the other one. It has not yet been properly optimized for speed, but we must probably still choose between efficiency and fast reconstruction. This problem is particularly severe at the forward angles because the CPU-time of this algorithms goes as the number of clusters squared - the analysis I ran on RCF (rcas0003) to reconstruct roughly 3500 events took ~60 hours! This is not very practical, so if we are to use this tracker it needs a good deal of optimization.

5 A problem with anti-ghosts

I also ran a first preliminary check to see if including the extra clusters in the event would produce ghosts. I ran the tracker first without the geant-track, and then compared the number of tracks found to that found with the track. The result for 8704 events from run 2336 (seq 0) can be seen in Table 1, where $\Delta N = \#TracksWithGeant - \#TracksWithoutGeant$.

BrTrackStringFinder performs as expected (?), with $\Delta N = 1$ in all but 0.6% of the events. BrTrackFollowFinder, on the other hand, seems to have problems finding the same amount of tracks when more (12) clusters are added. I cannot easily see the reason for this behavior, so I hereby ask the experts at BNL and in Copenhagen: What causes this?

A more quantitative study of the geant-induced ghosts, i.e. tracks found after adding the geant-track that do not overlap with any of the tracks found before the adding or the geant-track itself is also underways here in Oslo.

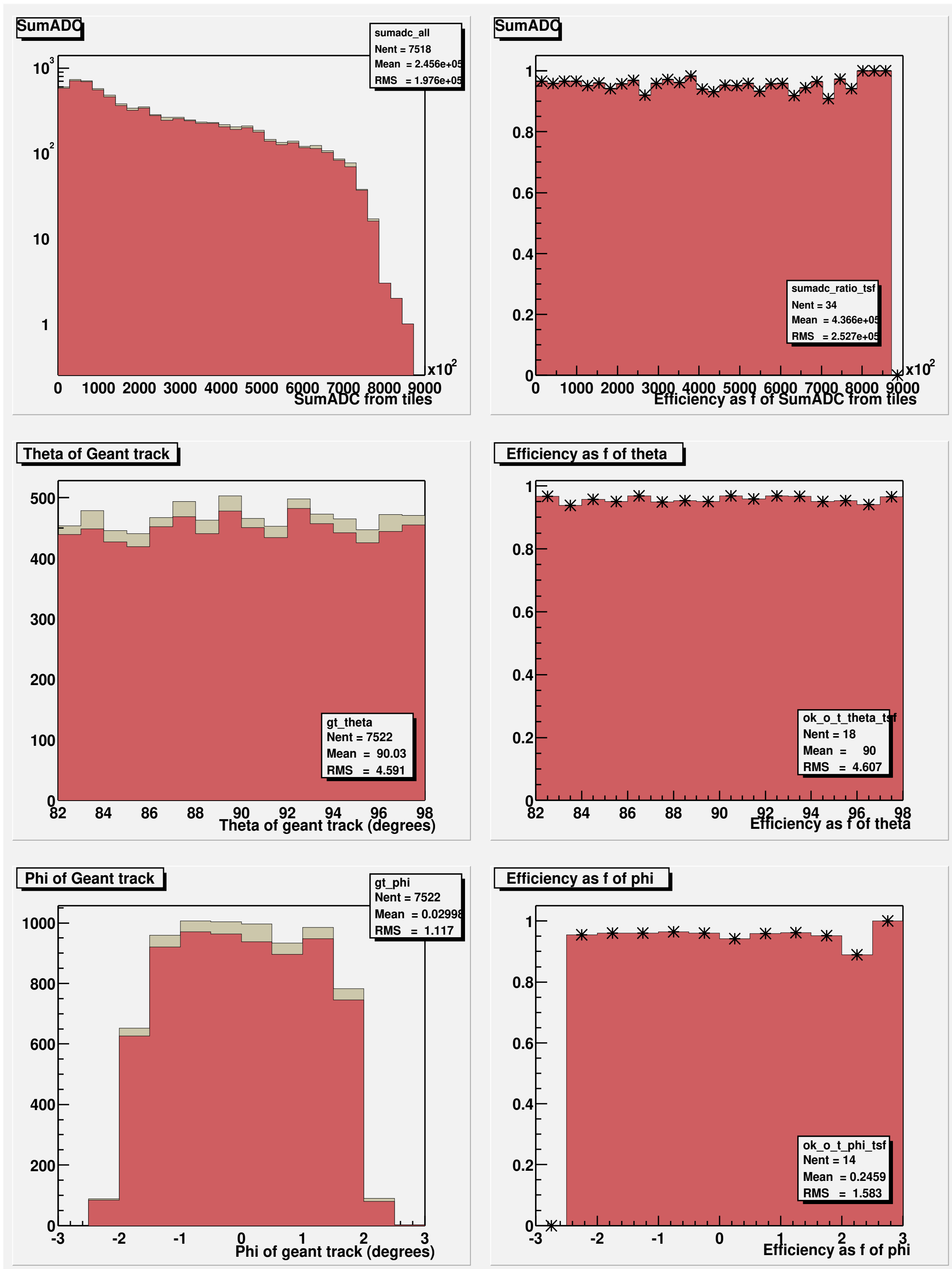

Figure 7: Efficiency-estimate, BrTrackStringFinder, MRS @ 40 degrees

ΔN	BrTrackFollowFinder	BrTrackStringFinder
<-4	5	0
-4	4	0
-3	14	0
-2	53	0
-1	123	0
0	1259	0
1	7188	8651
2	58	53
3	0	0
4	0	0
>4	0	0

Table 1: Tracks found with geant-track minus tracks found without geant-track. The entries in the table are number of events with the given ΔN.

6 Conclusions and visions

From the above figures I conclude that a reasonable first-order upper-limit for the two trackfinders is

BrTrackFollowFinder	BrTrackStringFinder
80%	90%

For BrTrackFollowFinder, some of the missing 20% may come from problems created by introducing a geant-track (i.e. something artificial) in the event, but since all I do is add more BrDetectorHits to an array it is difficult to see how this problem should affect only this case and not a real event with one more track. Hopefully testing with track-adding at the sequence level will soon help resolve this.

A problem with the seemingly most efficient tracker, though, is the CPU-time it uses. A lot of optimization is needed, but by for instance getting rid of noise-clusters and bananas/butterfiles we will also indirectly increase the tracking-speed. This should be (and is being!) pursued.

As mentioned, we will soon do this testing with track-adding at the cluster-level, and also try to tune some of the parameters for both trackers. Results from this will follow as they appear.

Please send comments on any and all of this to me and Trine.