Centrality estimates using the Scintilator Tiles in the Multiplicty Array
BRAHMS Analysis note # 22

Claus E. Jørgensen* Christian H. Christensen†

February 13, 2001

Abstract

We outline a method to estimate the centrality of a Au+Au collision at BRAHMS using the scintillating plastic tiles of the multiplicity array. It is shown, that to obtain a centrality cut in the most bias–free way, one *must* take into account the wide spread of the primary collision vertex, as observed during the summer 2000 run and explained elsewhere.

Contents

*Niels Bohr Institute, email: <ekman@nbi.dk>
†Niels Bohr Institute, email: <cholm@nbi.dk>

1 Introduction

The basic assumption is that there is a monotonic relation between the centrality and the charged particle multiplicity M of the collisions.

We can obtain a measure for the number of charged particles in the acceptance of the tiles from the energy deposited in them. It should be emphasized that what we label M is strictly speaking, not the charged particle multiplicity; how it is calculated is described in detail in appendix C. By cutting in M we can thus select event centrality.

2 The Problems

In figure 1 is shown dN/dM for events with an inclusive trigger 4 i.e., all events for which trigger 4 fired, plus the requirement that there is at least one hit in the scintilating tiles array above threshold.

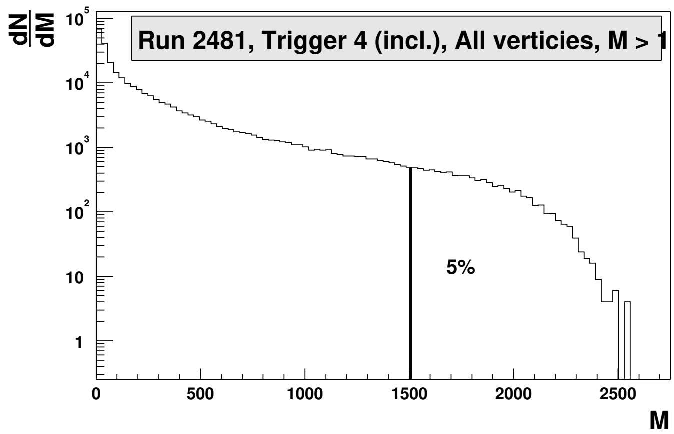

Figure 1: dN/dM for inclusive trigger 4 events

So far, much of the analysis has been done with centrality cuts as constant cuts in M.

One problem with this approach is that the acceptance of the tile array depends strongly on the position of the vertex, which means that M is not only a function of the collision centrality, but also of where the collision occurs. The dependency of M on the position of the primary collision vertex is illustrated in figure 2.

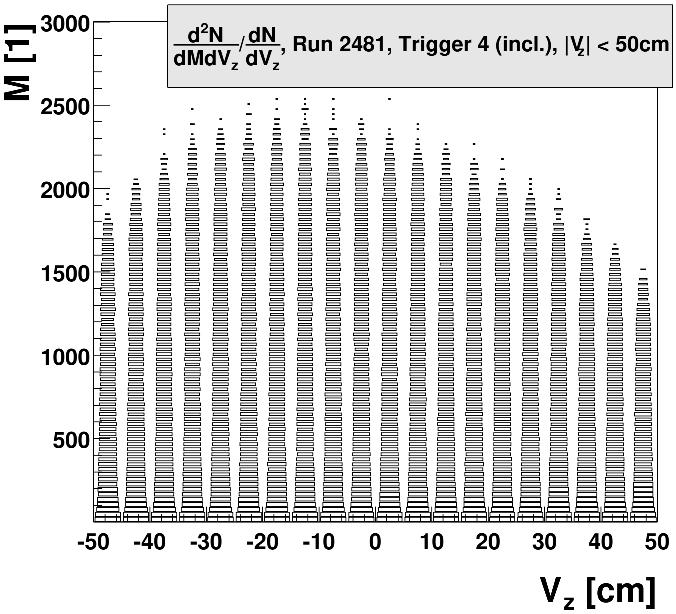

Figure 2: $\frac{d^2N}{dMdV_z}/\frac{dN}{dV_z}$ for inclusive trigger 4 events, Run 2481.

It can be seen from figure 2, that a global constant cut in M does not correspond to a cut in centrality, applicable to all values of V_z.

One way to get around this problem is to make the M distribution for a number of vertex bins and to determine centrality cuts for each bin. This is a kind of centrality calibration.

The Calibration Method

Having realized that the centrality is highly vertex dependent, we are forced to revisit our way of determining the centrality.

1. Starting from raw data, we look at minimum bias events (trigger 4 $\wedge 1 <$ tile hit above pedestal) with $|V_z| < 50cm$, where V_z is determined by the ZDCs. This is to ensure that V_z stays inside the coverage of the scintillator tiles, and to reject background from beam–gas, etc.

2. The multiplicity M and the vertex V_z are histogrammed to build up $\frac{d^2N}{dMdV_z}$, with 8 V_z–bins, corresponding to the 8 rings of tiles in the multiplicity array.

3. After having run through many events ($\mathcal{O}(100,000)$), we select a vertex bin and plot dN/dM for that vertex bin. See also figure 3

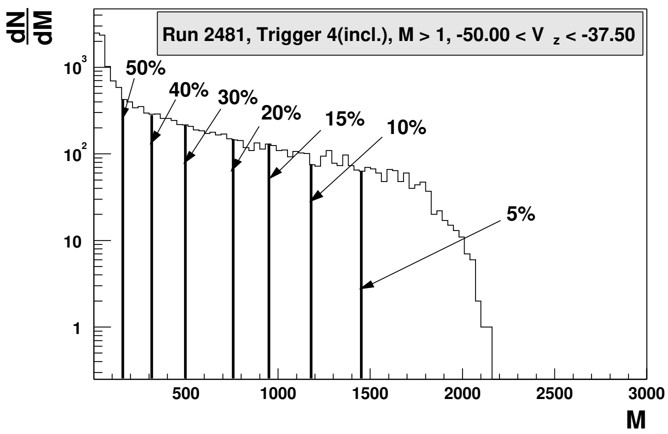

Figure 3: dN/dE at $-50cm < V_z < -37.5cm$ for inclusive trigger 4 events

4. This distribution dN/dM for a given vertex bin is then integrated in steps from M_{max} to 1, and the centrality cut, M_{cut}, is determined.

5. Steps 3 and 4 are repeated for all vertex bins, and the values M_{cut} are plotted as a function of V_z. This gives us series of pairs of numbers (V_z, M_{cut}), one series for each cut value.

   These points are then fitted with 3^{rd}–degree polynomial in V_z, to give us functions

$$S_{cut}(V_z) = a_{cut,0} + a_{cut,1}V_z + a_{cut,2}V_z^2 + a_{cut,3}V_z^3 \quad (1)$$

   See also figure 4.

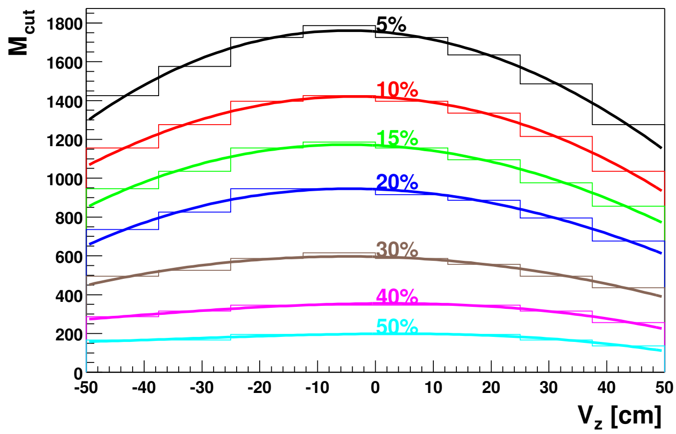

Figure 4: Cut functions

A third degree polynomial is used to give a reasonable fit with the fewest possible free parameters.

Centrality determination for a given event

For each subsequent event, one can, using $S_{cut}(V_z)$, now find the upper and lower limit for the collision centrality. For example, one could get the numbers 5 and 10 for upper and lower centrality limit – if the $S_{5\%}$ and $S_{10\%}$ functions are defined (in the calibration procedure) and the collision is in this centrality interval.

3 Comparison with the old method

In this section the centrality cut method described in this note and the `Get_centrality` function[1] are investigated.

It is fairly simple to check if, say a 10 percent centrality cut really corresponds to the 10 percent of events with the highest multiplicity. From a sample of N minimum bias event the cut should give $N/10$ events — in other words $N_{10\%}/N = 0.10$.

Approximately 400,000 minimum bias events (i.e. inclusive trigger 4 with at least one hit in the tiles) from run 2481 have been analysed. For 16 vertex bins the ratio N_{cent}/N has been found for the different (inclusive) centrality cuts in the two methods. Figures 5 and 6 (blow–up) show the results.

It's clear from the figures that the centrality cuts are improved, but they are still not perfect. It seems that there is some systematic effect on the edge bins, which means that the centrality of events with $|V_z| > 40$cm is still overestimated. The N_{cent}/N ratio for the 5%, 10% and 15% cuts shows a systematic decrease for growing V_z. We believe that these effects are due to the improper S fits. We will systematically investigate the fitting, and try to come up with a better calibration.

4 Examples

An example of using `BrTileCentModule` is given in

```
brahms_app/cholm_app/centrality/findcent.cxx
```

and one using `BrCentFilter` to cut in centrality is given in

[1] `http://www.rhic.bnl.gov/brahms/WWW/private/list_hyper/brahms-cent-l/att-0041/01-Get_centrality.C`, see also [2]

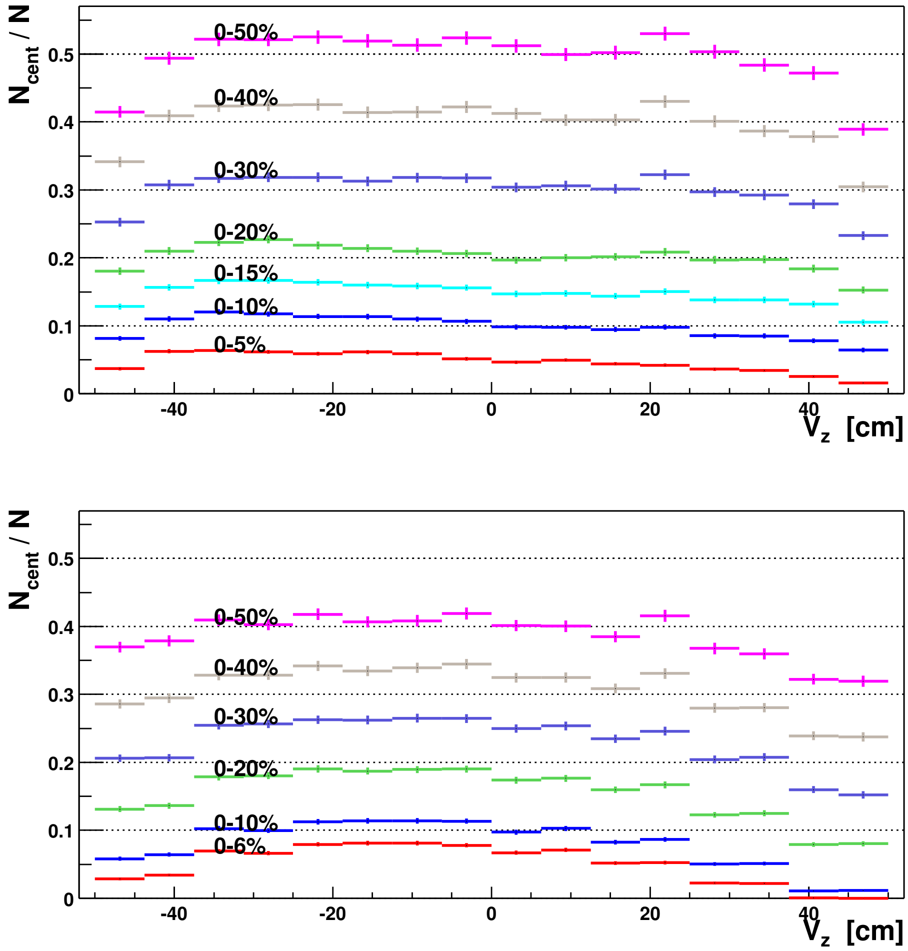

Figure 5: N_{cent}/N from `BrTileCentModule`, based on fitted cut values in M (top), from `Get_centrality`, based on constant cut values in M (bottom)

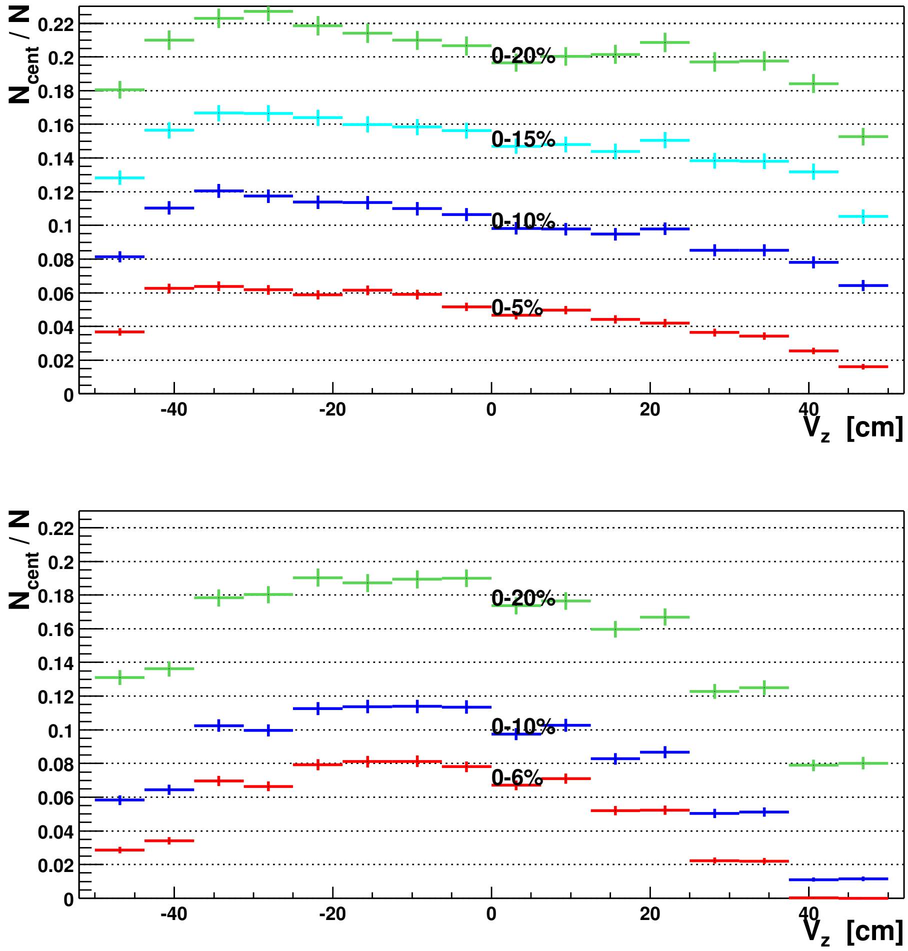

Figure 6: Same as figure 5, but only up to 20%, to emphasise the change from `Get_centrality` (bottom) to `BrTileCentModule` (top).

```
brahms_app/cholm_app/centrality/selectcent.cxx
```

both in the regular BRAHMS CVS repository.

A The Scintillating Tiles Detector

The multiplicity array of BRAHMS measures the deposit of energy by charged particles, as they pass through the sensitive matter of the detectors two components: the plastic scintillating tiles, and the silicon strip detectors.

Although the silicon strip detectors have a much better intrinsic resolution than the scintillating tiles, no attempt have been made to use these detectors for determining the centrality. This is mainly because the silicon strip detectors cover a lesser solid angle than the tiles, and the general consensus is that the scintillating tiles do well enough for the purpose at hand, determining the centrality.

The geometry of the scintillating tiles is described elsewhere [2], suffice to say, that the tiles are distributed around the nominal collision point in a slightly asymmetric way, as can be seen on figure 7.

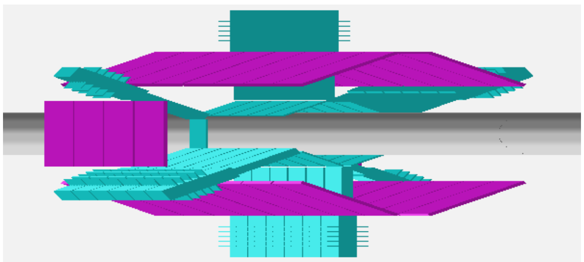

Figure 7: The multiplicity array seen from FFS side.

B Centrality

Naïve Approach

In a first attempt to define the centrality C' as dependent on the measured multiplicity M deposited in the scintillating tiles, we write

$$C'\left(M_m\right) \equiv 1 - \frac{\int_0^{M_m} dM\, \frac{dN}{dM}}{\int_0^{M_{max}} dM\, \frac{dN}{dM}} \quad , \tag{2}$$

where M_{max} is the maximum observed deposited energy in the tiles, and dN is the number of events with a given deposited energy dM.

The distribution dN/dM should be measured over all collisions, since the measured multiplicity M is most likely to be correlated with other conditions one might impose.

However, as is most often the case, we do not have a clear, unambiguous signal for a collision; hence we must rely on some sort of measurement to tell us that a collision has occurred.

Minimum Bias

Hence the need for a clear *minimum bias trigger*. The signal from the two Zero Degree Calorimeters (ZDCs) being closely spaced in time (trigger 4) and the requirement at least one tile has a ADC over pedestal, is generally considered to be a good minimum bias trigger [2]. This is because trigger 4 is the most *inclusive*[2] of the available triggers in BRAHMS, and the the requirement that at least one tile fires, vetoes pure Columb interactions between the colliding nuclei, and other kinds of events that may be considered as 'bad' events.

Taking the Primary Vertex Position into Account

For minimum bias events, we plot $\frac{d^2N}{dMdV_z}$ (see figure 2), and find that

$$N|_{M_m} = \int_0^{M_m} dM\, \frac{dN}{dM}$$

is not constant over V_z for a given E. This can be expressed be re–writing equation 2:

$$\frac{dC(M_m)}{dV_z} \equiv 1 - \frac{\int_0^{M_m} dM\, \frac{d^2N}{dMdV_z}}{\int_0^{M_{max}} dM\, \frac{d^2N}{dMdV_z}} \tag{3}$$

C ADC to Multiplicity

The first thing to be aware of is how the translation from ADC counts into M is performed[3]. The ADCs for the tiles are dual range ADC, so a preliminary correction of the ADCs must be done:

$$ADC'_i = \begin{cases} ADC_i & \text{for } ADC_i \leq Gap_{i,low} \\ ADC_i - (Gap_{i,high} - Gap_{i,low}) & \text{for } ADC_i > Gap_{i,low} \end{cases},$$

[2]Meaning it selects events that all other triggers also select, as well as less central events that may not satisfy other trigger requirements.

[3]The procedure presented here uses the calibration parameters given in [2], and is more or less them same.

where $Gap_{i,high}$ is the lower limit of the upper part of the ADC, and $Gap_{i,low}$ is the upper limit of the lower part of the ADC.

Next, we exclude all ADCs that are are lower than a certain threshold T:

$$T_i = P_i + a\Delta P_i \quad ,$$

where P_i is the ADC's pedestal, ΔP_i the pedestal width, and a some constant factor. a is usually set to 2.

The next step, is to subtract the pedestal from the ADC values, to give us the calibrated ADC_{cal}:

$$ADC_{cal,i} = ADC'_i - P_i$$

As described elsewhere [2], one must make a geometrical correction for the spread in path length through a given tile, when determining the individual tile multiplicity. This correction factor f is highly vertex dependent, and is given by:

$$f_i(V_z) = \sin\left(\theta_i(V_z)\right) = \frac{R}{\sqrt{(Z_i^2 - V_z^2) + R^2}} \quad ,$$

where R is the tile's radial distance from the beam axis, Z_i the i^{th} tile's position along the beam axis, and V_z the primary vertex position along the beam axis.

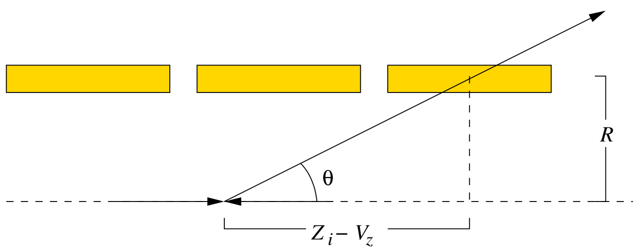

Figure 8: Geometrical correction factor for relative position of tile to vertex

Finally, we end up with something that is more or less proportional to the multiplicity in a given tile, by dividing by the mean deposited energy per particle $\overline{E}$ in a tile:

$$M_{i,m} = ADC_{cal,i}\frac{f_i(V_z)}{\overline{E}_i} \tag{4}$$

$\overline{E}$ is determined from cosmic ray data [1].

D The Vertex Algorithm

Presently, there are four ways of determining the vertex position along the beam axis V_z in BRAHMS:

1. Using the ZDC time signal as described above. This is considered the least precise measurement, as the intrinsic time resolution $\approx 115ps$ [3] giving intrinsic $\sigma_{ZDC}(V_z) \approx 3.5cm$. However, current analysis gives $\sigma_{ZDC}(V_z) \approx 5cm$ [4].

2. Using the Beam–Beam time signal, in similar fashion to the ZDCs. However, the timing resolution is $\approx 70ps$ giving a much better resolution for V_z.

3. Using reconstructed hits or clusters in the first TPC of the mid rapidity spectrometer (TPM1). This is considered the most precise algorithm so far.

4. Using reconstructed tracks from TPM1 to point at a vertex. This method shows some promise, but isn't fully understood yet.

However, we can not choose the vertex as we see fit when we build up the distribution $\frac{d^2N}{dMdV_z}$, since that would introduce a bias in the data. We are therefore restricted to using the vertex measurement obtained from ZDCs, despite the non optimal resolution, since we have already required hits in the ZDCs by choosing the trigger 4 as our minimum bias trigger, and this obtain no further bias.

It is possible using the time signal from the two ZDC, t_L and t_R, and assuming the particles that trigger the measurement in the ZDCs have $\beta \approx 1$, to obtain a very crude determination of the position of the primary vertex along the beam axis Vz:

$$V_z = c(t_R - t_0) - c(t_L - t_0) = c(t_R - t_L) \tag{5}$$

where t_0 is the, not measured, absolute time of the collision, and t_R and t_L is the average of the time signal from the first two modules in each ZDC.

E Overview of Scintillating Plastic Tile Code

After the re–organisation of the multiplicity array code we have the following class for the tiles:

`BrTileCalibration` Object to store calibration constants, such as pedestals, gain, and of course the centrality cut functions $S_{cut}(V_z)$.

`BrTileCent` Object to store, for a given event, the lower and upper bounds on centrality, as determine by `BrTileCentModule`.

`BrTileCentCalModule` Module to find the functions S_{cut} from many events with `BrTileRdo` objects in them. This class assumes that a proper vertex cut has been defined, for example using `BrVertexFilter` and that minimum bias events have been chosen, for example using `BrTriggerfilter`.

`BrTileCentModule` Module to determine the centrality of a given event, based on the input object `BrTileRdo` and the cut functions $S_{cut}(V_z)$.

`BrTileDig` Object[4] storing the raw digits from the Data Acquisition system (DAQ).

`BrTileDigModule` Module[5] that makes simulated digits from GEANT input.

`BrTileGapCalModule` *Not complete yet.* Module to find the dual range ADC's gap.

`BrTileParameters` Object[6] to store static parameters read from the file `$BRATSYS/params/DetectorParameters.txt`. Some of it's data members should really be read from a geometry database.

`BrTilePedCalModule` *Not complete yet.* Calibration module to find the ADC's pedestals and pedestal widths. This module assumes that pedestal events have been chosen before hand, for example using `BrTriggerFilter`.

`BrTileRdo` Object[7] to store calibrated data — also called a Reduced Data Object.

`BrTileRdoModule` Module[8] that makes calibrated data from `BrTileDig` objects. It assumes that the vertex information is present, as created by some other module e.g., `BrRdoModuleBB` or `BrRdoModuleZDC`. Currently, it uses static calibration constants, but is prepared for using calibration constants retrieved from the databases.

[4] Was `BrDigTiles`, which still is present for backward compatibility.
[5] Was `BrDigitizeTiles` but renamed to fit into new naming scheme.
[6] Was `BrDetectorParamsTiles` but renamed to fit into new naming scheme.
[7] Was `BrRdoMult`, but now split into 2 classes: One for the tiles, and one for the silicon.
[8] Was `BrRdoModuleMult`, but split into 2 classes One for the tiles, and one for the silicon.

F The (almost) Complete Code

So to find the calibrated data, and store it in a `BrTileRdo` object for each input event, one could have code like

```
BrModuleContainer* mainCont =
  new BrModuleContainer("mainCont", "Main Cont");
BrRdoModuleZDC* zdcModule =
  new BrRdoModuleZDC("ZDC", "ZDC reducer");
mainCont->AddModule(zdcModule);
BrRdoModuleBB* bbModule =
  new BrRdoModuleBB("BB", "Beam-Beam Reducer");
mainCont->AddModule(bbModule);
// ... more vertex modules (?)
BrTileRdoModule* tileModule =
  new BrTileRdoModule("MultTile", "Tile Reducer");
mainCont->AddModule(tileModule);
mainCont->Init();
// From here on normal event processing
```

To find the functions $S_{cut}(V_z)$, one would take the output from above and pass it through code like the below:

```
BrModuleContainer* mainCont =
  new BrModuleContainer("mainCont", "Main Cont");
BrTriggerFilter* trigFilt =
  new BrTriggerFilter("trigFilt", "Trigger filter");
mainCont->AddModule(trigFilt);
trigFilt->AddTrigger(4);
BrRdoModuleZDC* zdcModule =
  new BrRdoModuleZDC("ZDC", "ZDC reducer");
mainCont->AddModule(zdcModule);
BrVertexFilter* vtxFilter =
  new BrVertexFilter("VTX", "VerTeX Filter");
mainCont->AddModule(vtxFilter);
vtxFilter->SetMaxZdcVertexZ(50);
BrTileCentCalModule* centcalModule =
  new BrTileCentCalModule("centTile", "Centrality Calib");
mainCont->AddModule(centcalModule);
mainCont->Init();
// From here on normal event processing
```

To find the centrality limits of a given event, one should have:

```
BrModuleContainer* mainCont =
  new BrModuleContainer("mainCont", "Main Cont");
BrRdoModuleZDC* zdcModule =
  new BrRdoModuleZDC("ZDC", "ZDC reducer");
mainCont->AddModule(zdcModule);
BrRdoModuleBB* bbModule =
  new BrRdoModuleBB("BB", "Beam-Beam Reducer");
mainCont->AddModule(bbModule);
// ... more vertex modules (?)
BrTileCentModule* centModule =
  new BrTileCalModule("centTile", "Centrality finder");
mainCont->AddModule(centModule);
BrCentFilter* centfilterModule =
  new BrCentFilter("centFilter", "Centrality filter");
mainCont->AddModule(centfilterModule);
centfilterModule->SetLowCut(0); // From 0 to 10%
centfilterModule->SetHighCut(.1); // From 0 to 10%
mainCont->Init();
// From here on normal event processing
```

References

[1] Ramio Debbe. Calibration of tiles with cosmic rays. BRAHMS Analysis Note 12, Brookhaven National Laboratory, September 2000.

[2] J. H. Lee. From ADC to dN/deta. BRAHMS Analysis Note 14, Brookhaven National Laboratory, November 2000.

[3] Micheal Murray. Zero degree calorimetry for brahms. BRAHMS Note 14, Texas A&M University, Febuary 1998.

[4] Bjørn H. Samset. Re: New BrRdoModuleZDC in brat. E–mail to `brahm-dev-l`, Febuary 2001. `http://www.rhic.bnl.gov/brahms/WWW/private/list_hyper/brahms-dev-l/0771.html`.