Residual MTR efficiency estimation and usage

Dario Berzano, Pascal Dupieux, Xavier Lopez,
Bogdan Vulpescu, whoever...

Laboratoire de Physique Corpusculaire - Clermont-Ferrand, France

Version 1 - 20100721

Residual «R» efficiencies

- It is the «residual efficiency»: useful to estimate systematic errors, such as misestimated hadrons amount in simulations, with the assumption of the factorization of different efficiency loss causes. **Is it right?**
- They are calculated using simulations with 100% RPC efficiency and data
- It is already given in OCDB format: `MUON/Calib/TriggerEfficiency` - use it with `AliCDBManager::SetSpecificStorage()`
- For now they are calculated per local board: no difference between bending/nonbending plane → there are $234 \cdot 4$ R values
- **Note:** indeed, there are $234 \cdot 4 \cdot 2$ values, but values for bending and nonbending plane are the same, for now
- **See** `MtrEff02.pdf` (with my highlights, underlines and comments)

From «hardware» efficiencies to phase space efficiencies

- R values are calculated by running a simulation (100% RPC eff) and reconstructing real data; efficiencies thereby calculated are divided to produce R
- We normally re-run the simulation with R eff (no more 100%!): simulation results are subsequently used to calculate efficiency in phase space
- With a table of efficiencies in phase space, that in principle varies from run to run, we can correct distributions in analysis of real data

Fast and slow methods to apply the hardware R values

- Remember: apply them to simulations!
- **Slow method (standard & affordable)**
  Rs are stored in a custom OCDB: automatically used, if SetSpecificStorage(), when simulating+reconstructing. It is slow because the complete simulation is ran once to calculate Rs, and the 2nd time to use R
- **Fast method (to be validated)**
  We use them at AliESDs.root level. These ESDs are products of sim+rec with 100% RPC efficiency, and by loading R values we can «throw dice» and decide whether to keep or reject the track. It is fast because simulation is run only once (to calculate Rs)

Track hits diffusion

- Does a track stay in the same local board or same RPC, or what else?
- Read this information in ESD with `AliESDMuonTrack::GetEffFlag( muTrack->GetHitsPatternInTrigCh() )`:
  - `kNoEff` → track not good for efficiency calculation; it should not be taken into account neither in «efficient» nor in the «total number» of tracks
  - `kChEff` → use chamber level efficiency granularity: in my analysis it is estimated with average of all local boards
  - `kSlatEff` → use RPC level efficiency granularity: estimated with average too
  - `kBoardEff` → directly use Bogdan's values!

Different levels of granularity already at R calculation level?

- While using values in the sim («slow»), maybe (I do not know for sure how simulation+reconstruction works for this) local board granularity will suffice
- But when using efficiency «a posteriori» (that is: «fast» method, at AliESDs.root level), for some kind of events we need less granular efficiencies
- Why? In ESD we don't have the hits! Only the first local board crossed, and the «eff flag» (see prev. slide) that says if the track changed local board or RPC
- Lesser granularity (at chamber and RPC, *i.e.* «slate» level) is now estimated by properly averaging local board values: is it correct? Wouldn't it be better to directly calculate it?
- **Important (for me):** differentiate bending and nonbending plane, and use, in OCDB, the recently introduced correlation («both planes») values

Definitions, for clarity!

- We have got two kinds of efficiencies, let's call them *(def.)*:
  - **residual chamber efficiency:** the ratio of sim over data, or the so-called «R»s!
  - **track efficiency:** how is it calculated?
- The first is distributed in the X, Y projective trigger plane by using local boards + (eventually) bending/nonbending plane granularity
- The second would be distributed in the phase space plane (θ, ϕ, P, DCA) and it was already calculated by Bogdan (see some of the `MtrEff02.pdf`'s colored plots!), so the code exists!

Doubts and observations on the track efficiency

- For me to do: have a look at Bogdan's code to produce these plots! Will the ESD suffice, *i.e.* **can it be calculated at the analysis level?** I think so, but I (still) don't know how!
- This is **Very Important™** because:
  - These distributions will be used for correcting distributions from Real Data
  - The comparison between the track efficiencies calculated with the «slow» and «fast» methods (on data coming from the same simulations) provides a validation of the latter! **Is it true?**

Why use extrapolation?

* Speaking of track extrapolation: it seems that Bogdan uses it… I don't know how and exactly where, I should look at the code

* See Slide 8 from MtrEff02.pdf

* Extrapolation is performed by AliMUONTrackExtrap class, normally used to match trigger and tracker tracks

* We may think of using it for only a reason: trigger-only tracks do not have kinematics! By prolonging the track into the tracker we can obtain it

Why NOT use extrapolation?

- For me (after a conversation with Philippe Pillot), extrapolation should **not** be used! Why?
- There is an iron wall 120 cm thick!
- Ok, tracks that do pass the wall are mostly at high-energy; so energy loss correction is negligible (which may be performed for the class)
- But Multiple Coulomb Scattering occurs! Even if we apply the proper correction, low Pt tracks may be badly positioned by **several centimeters**!
- The Bottom Line: we do all this R calculation to eliminate as much as possible systematic errors in efficiency determination, but we highly risk to introduce a systematic error ourselves which may be concealed inside the track extrapolation algorithm!

Please make comments :)