• When last we left this saga,
  • I had generated Run 3 muon “factors” for the theta_index=1,2,3 endcap crystals using muon peaks, but
  • it was clear that the factors loaded at the end of Run 2 were not being correctly interpreted/implemented. (I.e., we had treated muon factors as if they were real Bhabha constants, so that the logarithmic energy interpolation was not properly accounted for.)
  • It is straightforward to take the interpolation into account. New muon-based Bhabha-like constants have been derived.
Method to “calibrate” inner three rings of endcap

- Use muons to “transfer” the calibration from a good EMC region into the inner three rings.

- Use ring 4 as the normalization for the calibration (e.g., similar radiation damage and negligible momentum difference for muons).
  - Using muon peaks, determine the central value for ring 4

- Calculate factors for each crystal in rings 1-3 to shift the peak for that crystal to the central value for ring 4.
  - Since ring 1 has shorter crystals, ring 1 is shifted to \((\text{ring 4})^{(16.5/17.5)}\)

- The precision of this method is limited to the few percent level by the uncertainty in muon peaks for individual crystals, but for rings 1-3, this is a substantial improvement.
Digi’s are “calibrated” by taking the raw digi energy $E$ and multiplying it by an energy-dependent coefficient that is interpolated between the source and Bhabha constants:

$$E_{\text{cal}} = E \left[ c_{\text{source}} + r (c_{\text{Bhabha}} - c_{\text{source}}) \right]$$

where

$$ r = \frac{\ln E - \ln 6.13}{\ln E_{\text{Bhabha}} - \ln 6.13} $$

For crystals in the forward endcap, muon peaks are at about 215 MeV. For each crystal we determine a factor $f$ that multiplies the coefficient for that crystal to get the calibrated coefficient. We want to use this information to derive a new Bhabha constant.

$$ f \left[ c_{\text{source}} + r (c_{\text{Bhabha}} - c_{\text{source}}) \right] = \left[ c_{\text{source}} + r_{\mu} (c'_{\text{Bhabha}} - c_{\text{source}}) \right] $$

where

$$ r_{\mu} = \frac{\ln 215 - \ln 6.13}{\ln E_{\text{Bhabha}} - \ln 6.13} $$

So,

$$ c'_{\text{Bhabha}} = c_{\text{source}} + f \left[ c_{\text{source}} + r_{\mu} (c_{\text{Bhabha}} - c_{\text{source}}) \right] - c_{\text{source}} $$

$$ r_{\mu} $$
Solid line is new Run 3 Bhabha constants based on muons.

Dashed line is the starting Bhabha constants.
Solid line is late Run 2 Bhabha constants based on muons.

Dashed line is the starting Bhabha constants.
The early Run 3 and late Run 2 Bhabha constants can be compared, if the change in front-end constants is taken into account. Some variation over time is expected. A few percent uncertainty results from extrapolating the muon peak.

Comparing muon-based Bhabha constants is uncertain at a rough level of

\[
\sigma^2(\text{diff}) = \sigma^2(\text{Run 2}) + \sigma^2(\text{Run 3}) = (2\sigma_{\mu})^2 + (2\sigma_{\mu})^2 = 2(2\sigma_{\mu})^2
\]

\[
\sigma(\text{diff}) = 1.4(2\sigma_{\mu}) = 2.8 \sigma_{\mu}
\]

(Ignoring any effect from the uncertainty in source constants.)
Comparison of muon-based Bhabha constants for early Run 3 and late Run 2.
(The change in front-end constants has been accounted for.)

Taking this comparison at face value, the mean Bhabha constant is 2.2% higher, and the implied uncertainty in the muon peaks is about 5.0%/2.8=1.8%.
Conclusion

• Starting Run 3 Bhabha-like constants have been derived.

• Constants for the late Run 2 data (03 May +) have been derived. These appear to be consistent with Run 3 constants within expected uncertainties intrinsic to the method.

• A number of complexities have become clear. For instance, source calibrations are normally done frequently (when there is a working neutron generator), so that two or three different source calibrations will be loaded for a period long enough to get sufficient single-crystal muon statistics.

• To put things in perspective, the Bhabha constants in the database for theta_index=1,2,3 in the past were not being updated at all, and were typically wrong by 30% at the end of Run 2. The muon-based method is good to a few percent, but not better than that.