Status Report: Run 3 Electron Reconstruction Studies

Furkan Cetin

Heidelberg University

cetin@physi.uni-heidelberg.de

January 15, 2024

Motivation

- electrons heavily emit bremsstrahlung
- electron track-finding underperforms compared to other particles

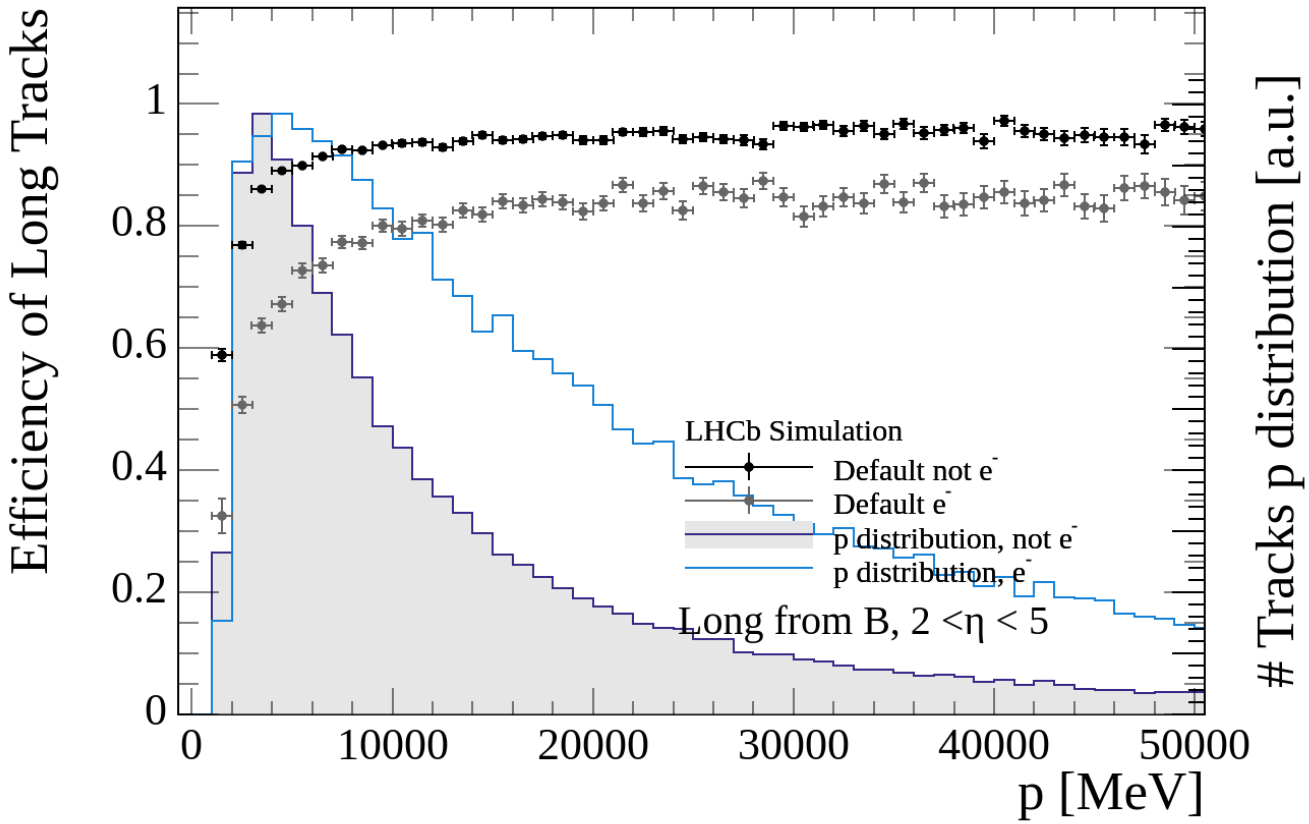


Figure: BestLong Efficiency for electrons and not-electrons using baseline reco for the $B^0 \to K^* e^+ e^-$ decay

Reminder

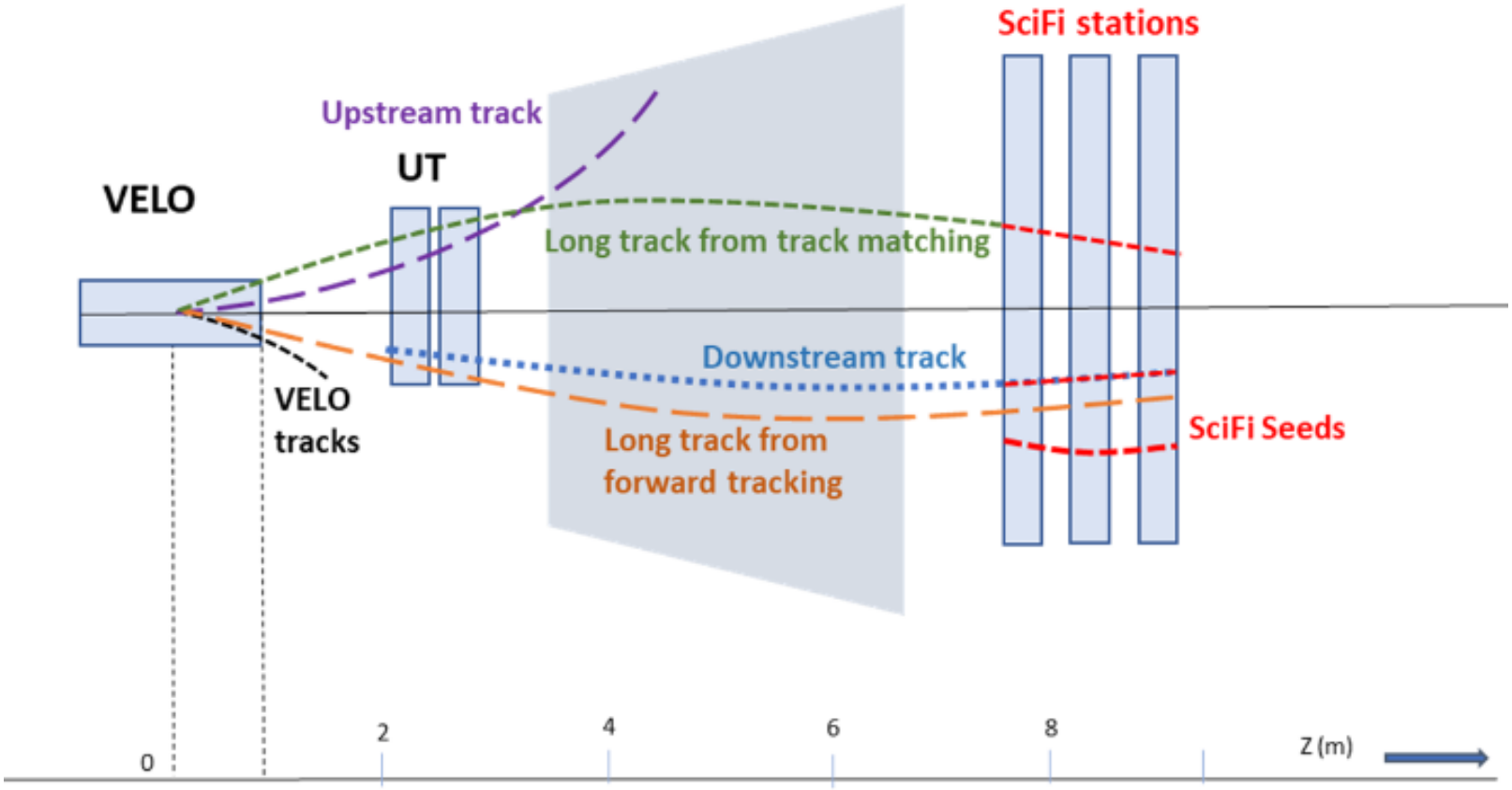


Figure: Track Types at LHCb

Idea

- we have Velo and Scifi tracks, with good $e^{\pm}$ efficiency

→ implement a Matching algorithm for electrons

- current Matching not trained for electrons
- Matching is computationally cheap → can be used in trigger

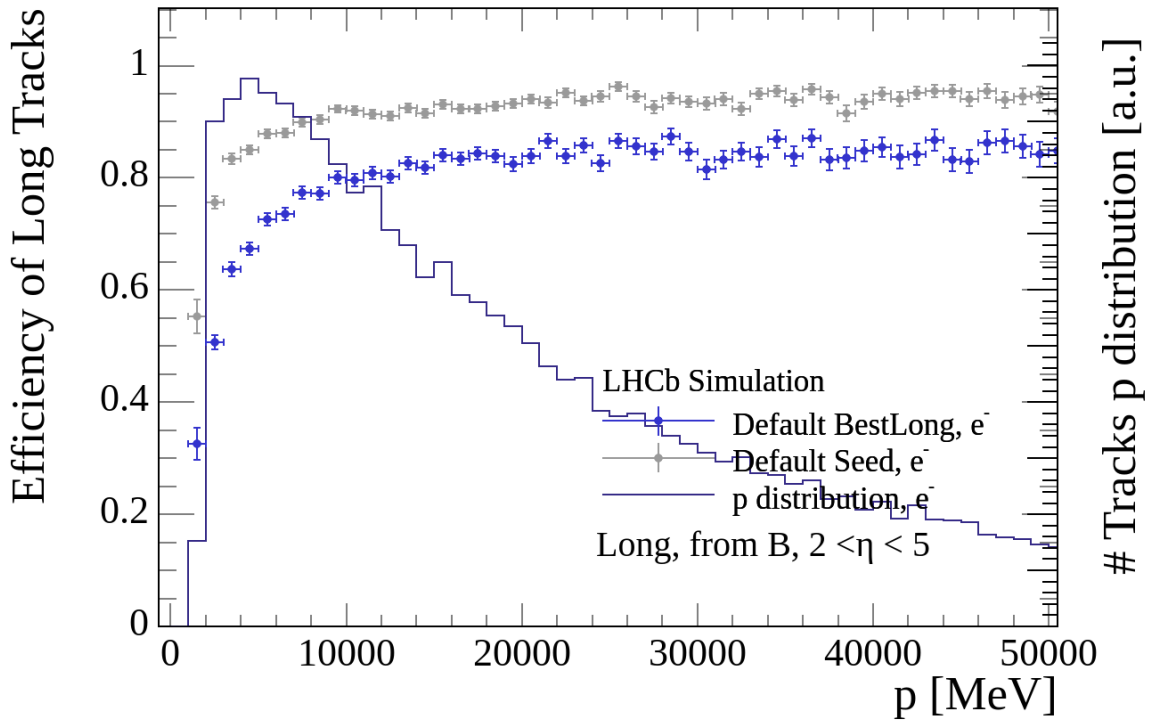


Figure: Efficiency for Electron Seed Tracks and BestLong Tracks in baseline reco

The Matching Algorithm

- use a classifier, a Multi Layer Perceptron
- quantify the level of agreement, i.e. a match

Table: Input variables of the Matching MLP

Variable	Preselection
χ^2_{match}	< 15
D_x	< 250 mm
D_y	< 250 mm
$\lvert\Delta t_x^{\text{match}}\rvert$	< 1.5
$\lvert\Delta t_y^{\text{match}}\rvert$	< 0.15
$t_x^2 + t_y^2$	

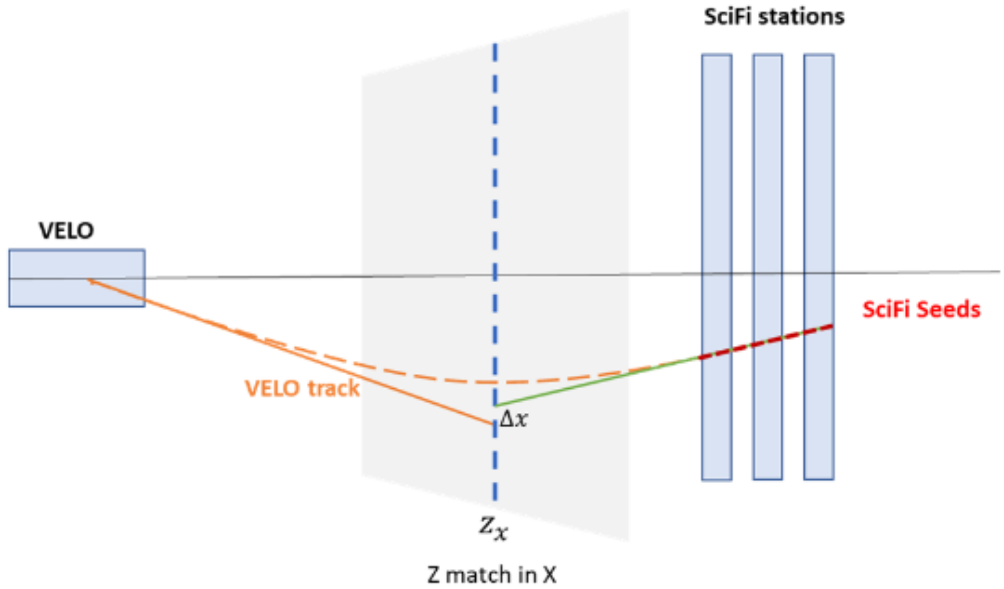


Figure: Matching Distance D_x

Matching Variables in baseline reconstruction

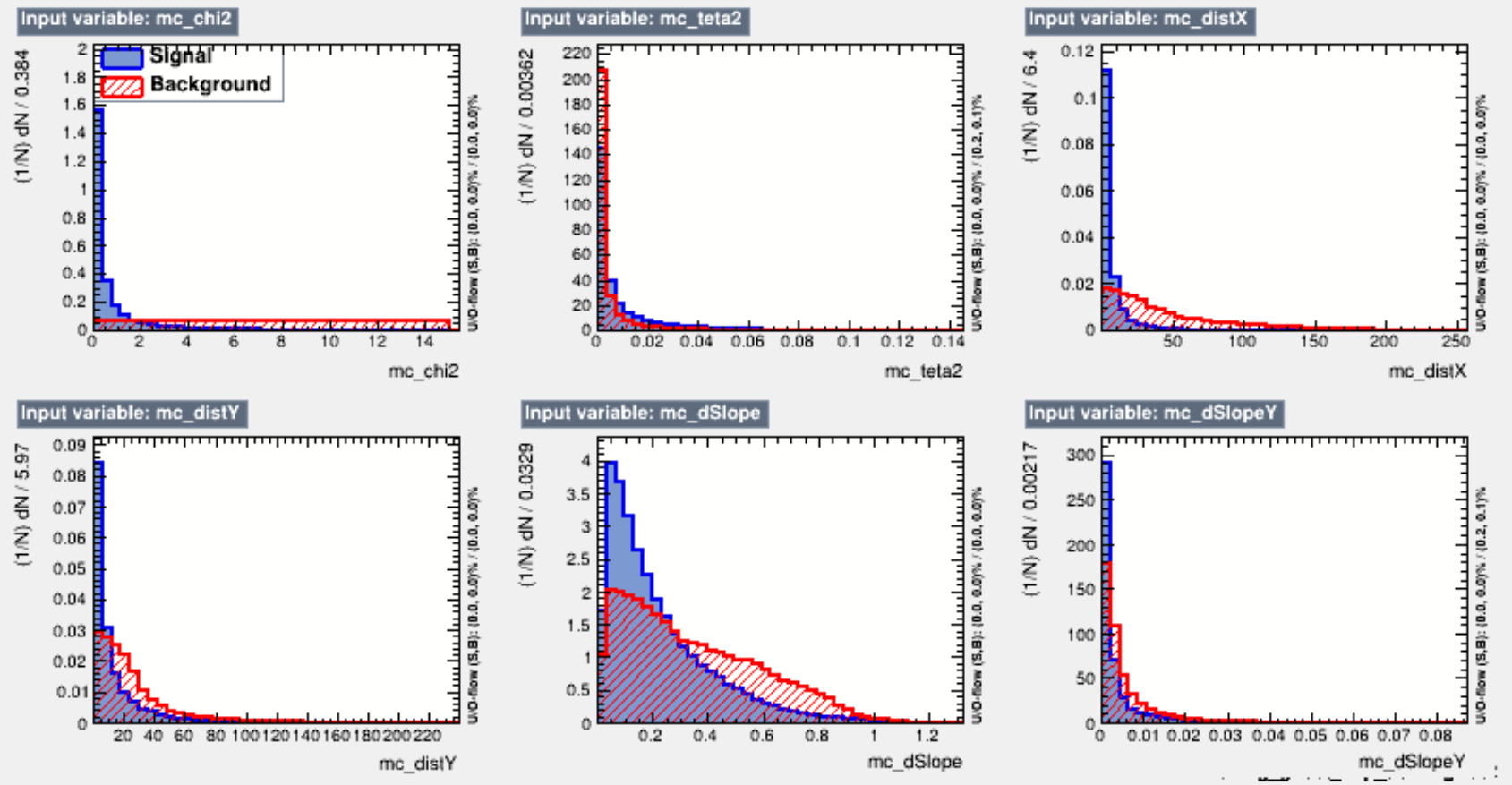


Figure: Input variables for baseline reco, i.e. explicitly excluding electrons as signal

Matching Variables in electron-specific reconstruction

Trained a NN with true long tracks of $e^{\pm}$ as signal and true ghosts of $e^{\pm}$ and not-$e^{\pm}$ as background.

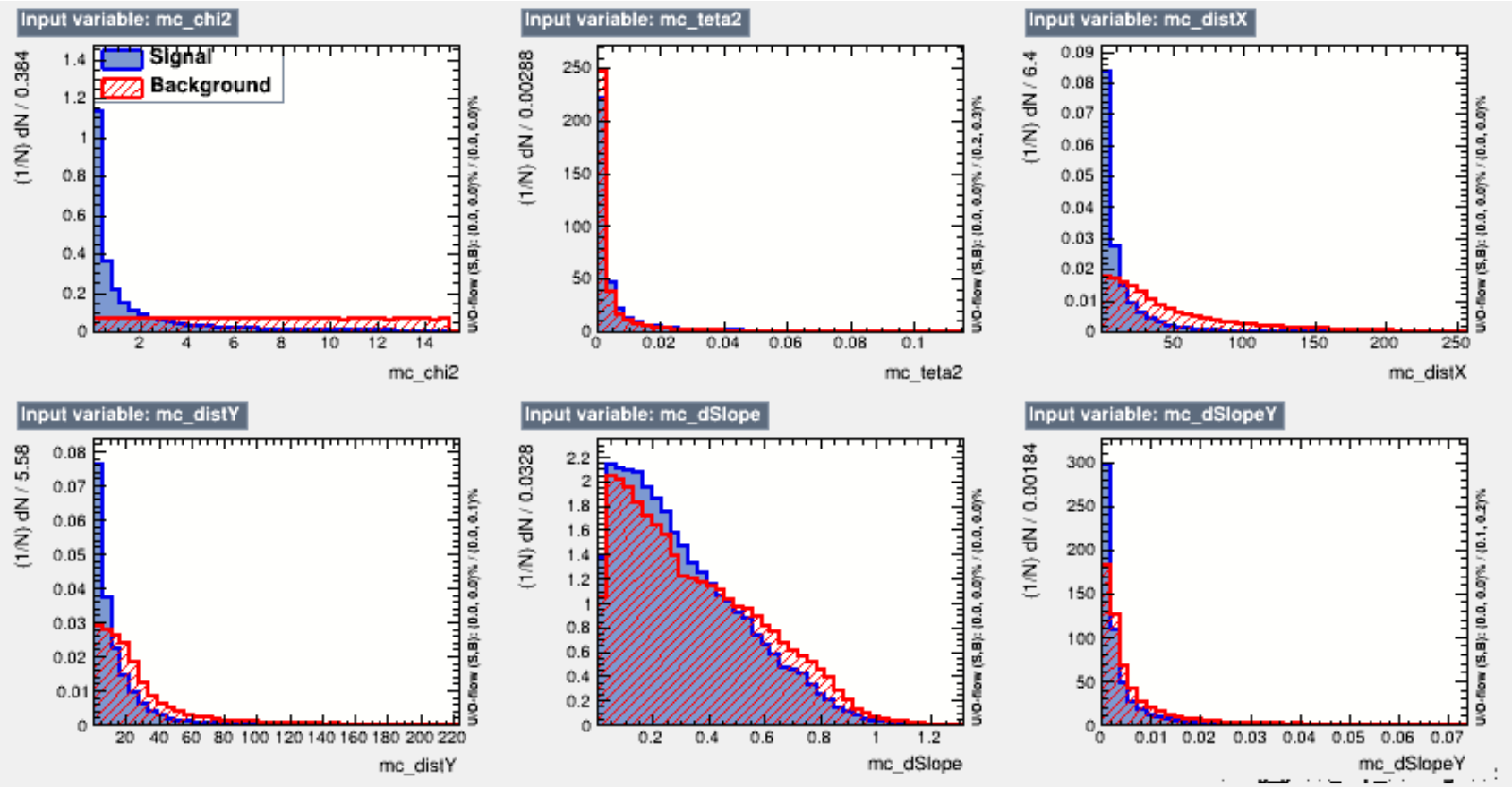
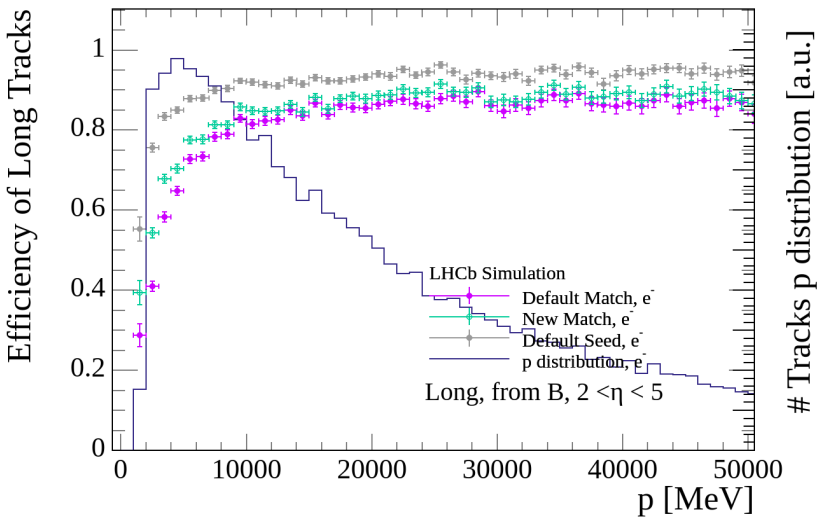
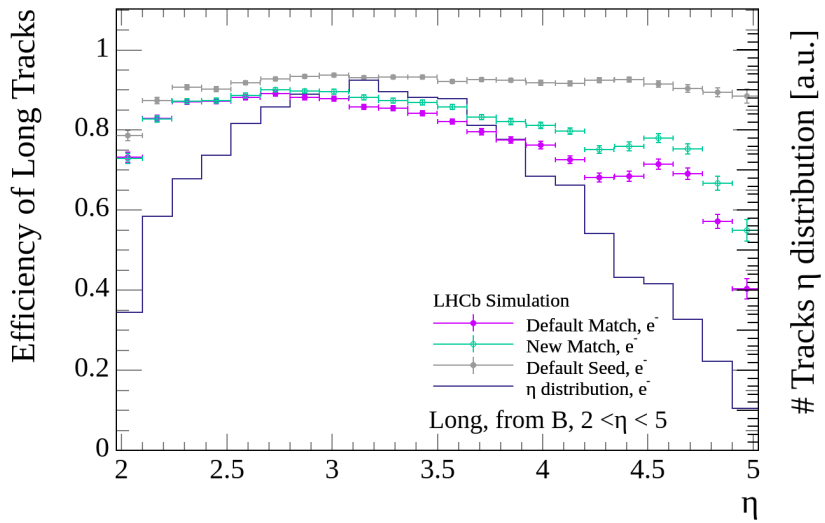
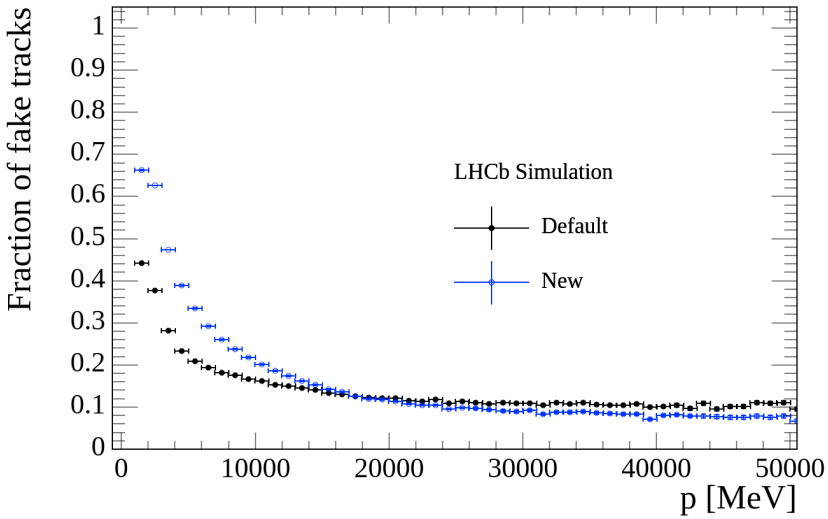


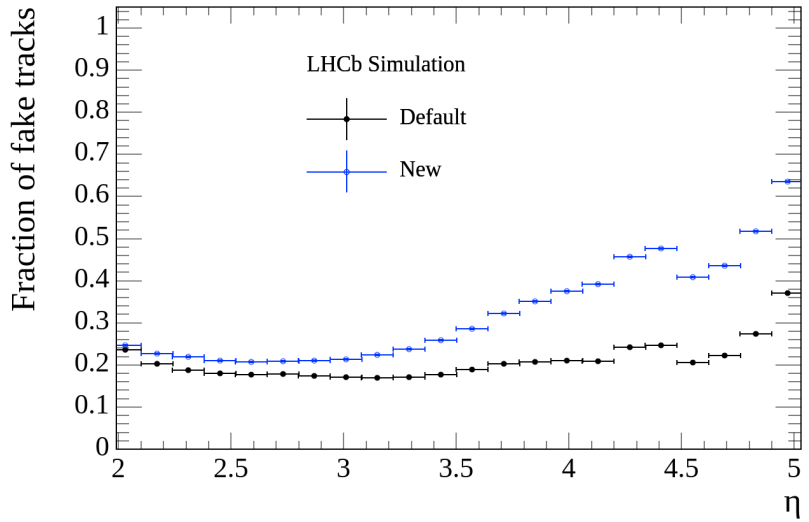
Figure: Input variables for electron-specific reco, i.e. only electrons as signal

Preliminary Efficiency and Ghost Rates









Conclusion

- Matching ansatz looks promising for electrons
- simple retraining already gains efficiency

To do:

- look into other possible input variables, exploiting the detector geometry
  - parametrisation of radiation length $\langle E \rangle = E_0 e^{z/X_0}$
  - more use of geometric variables such as ϕ
- try different NN architecture

Ideas to control ghosts:

- filter Seed tracks using calorimeter information
- use Velo and/ or Seed tracks left over after Long tracking

Backup: Baseline Reconstruction and Residual Matching

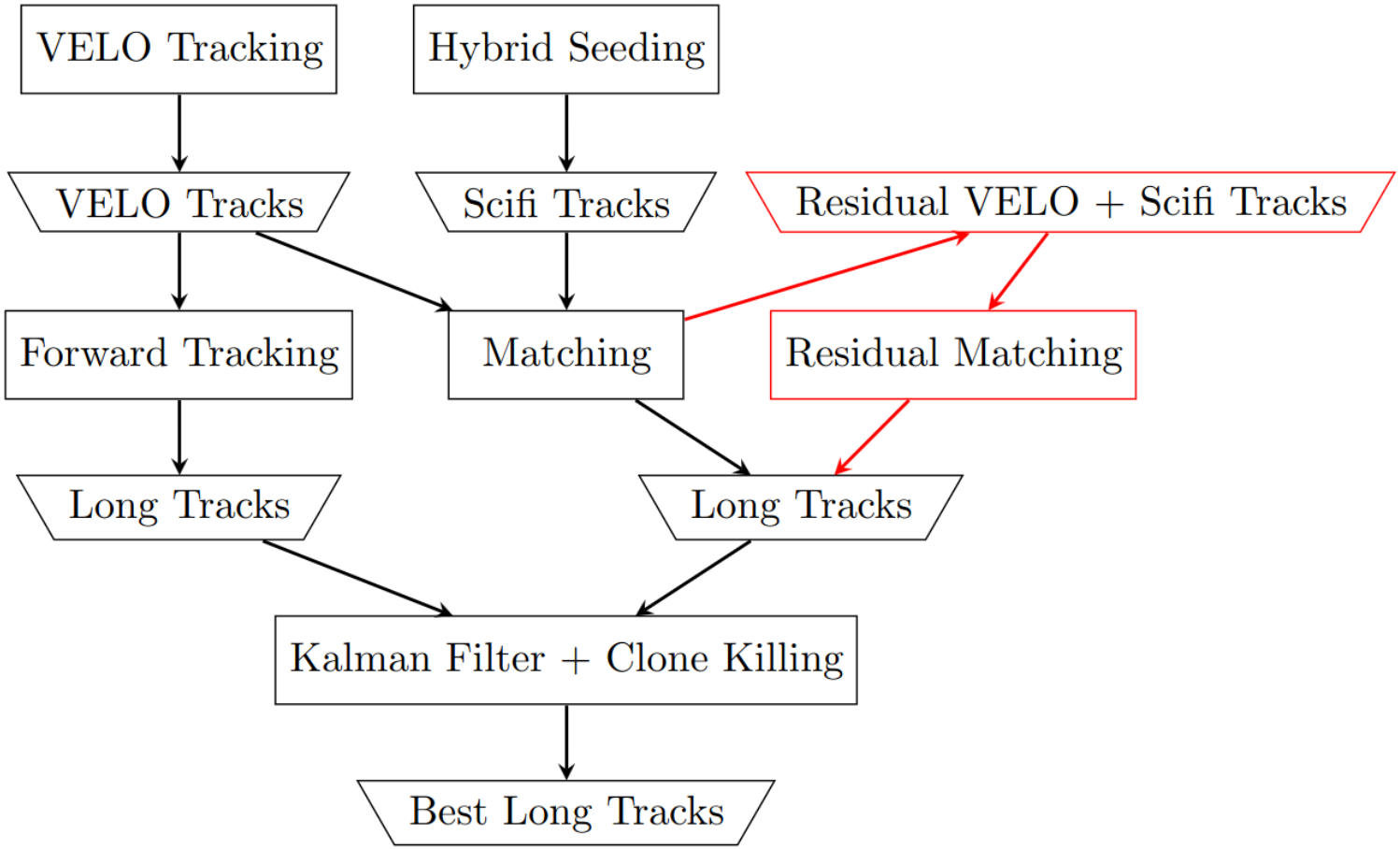
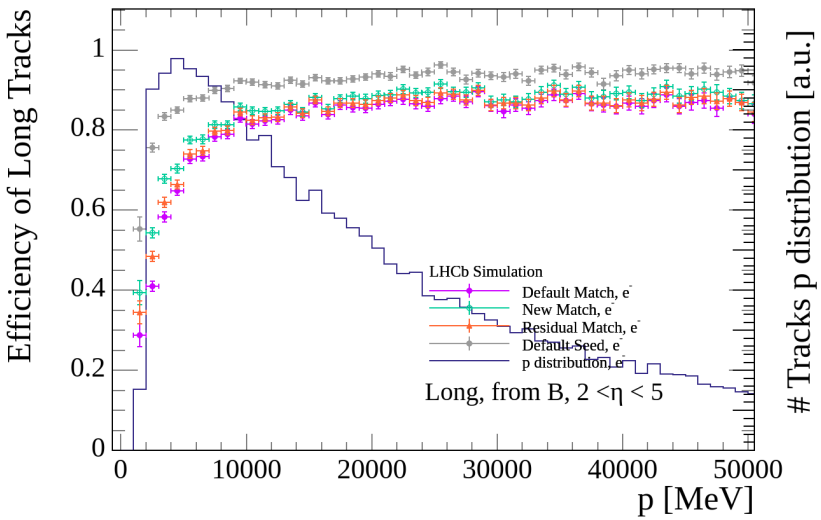


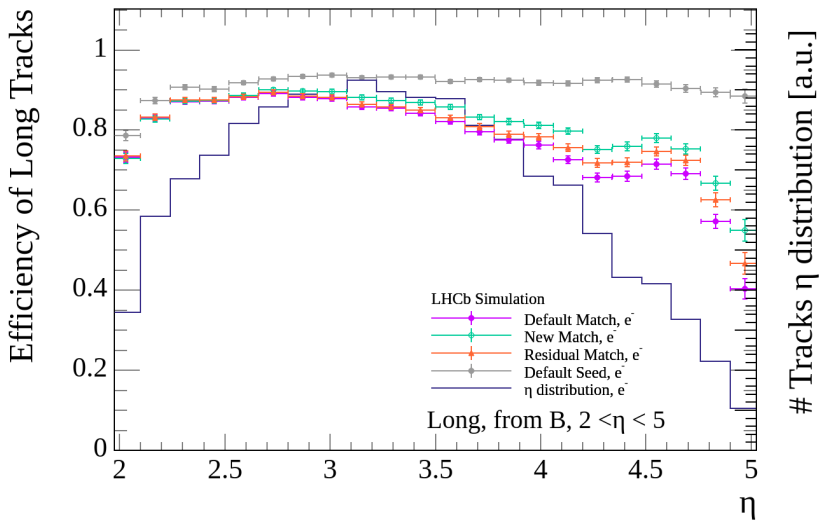
Figure: Data flow in baseline reco (black) and residual matching (red)

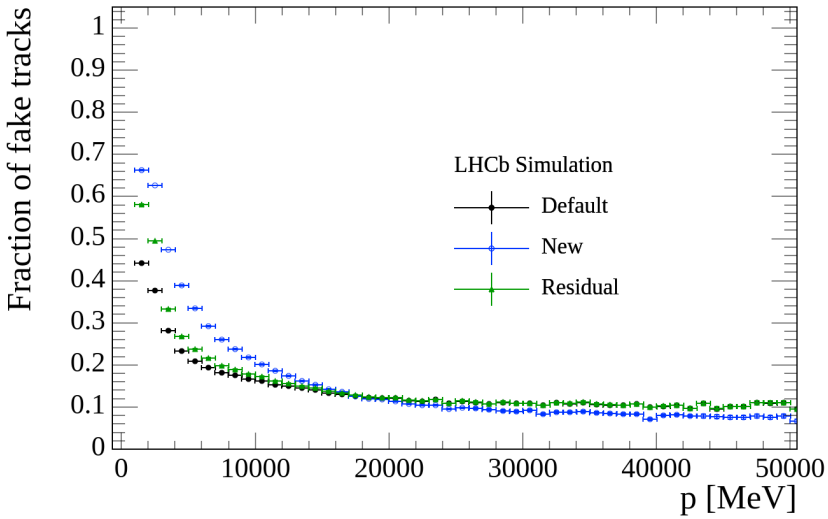
Residual Velo and Scifi tracks are given to a second Matching algorithm

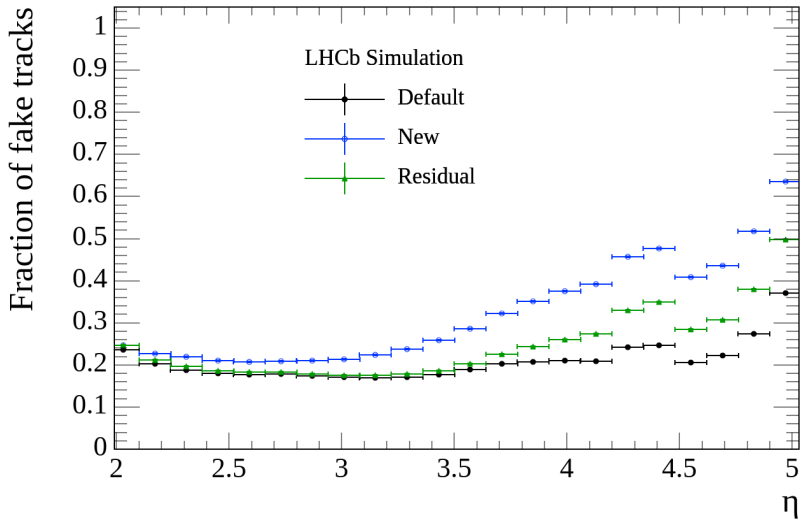
→ trained with simulated electron long tracks as signal

Backup: Residual – New

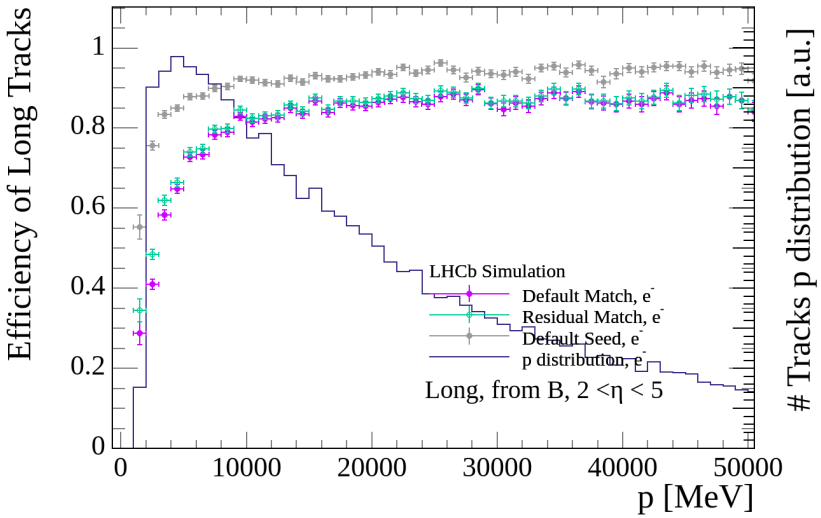


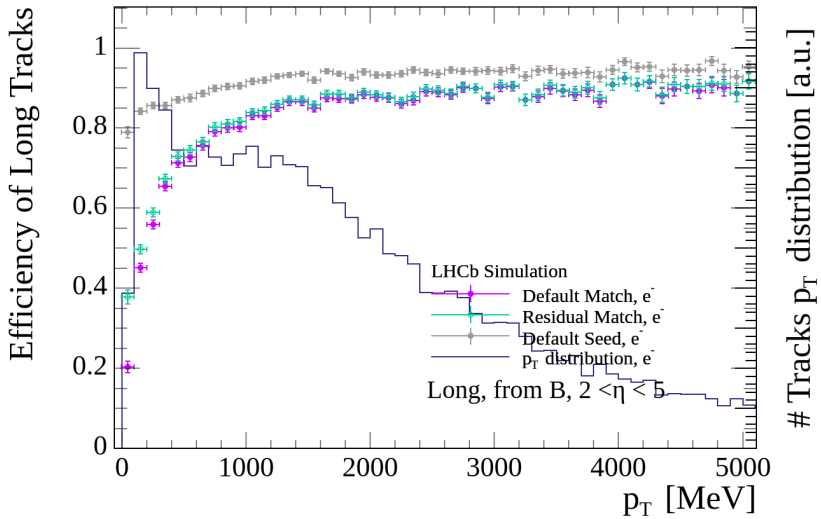


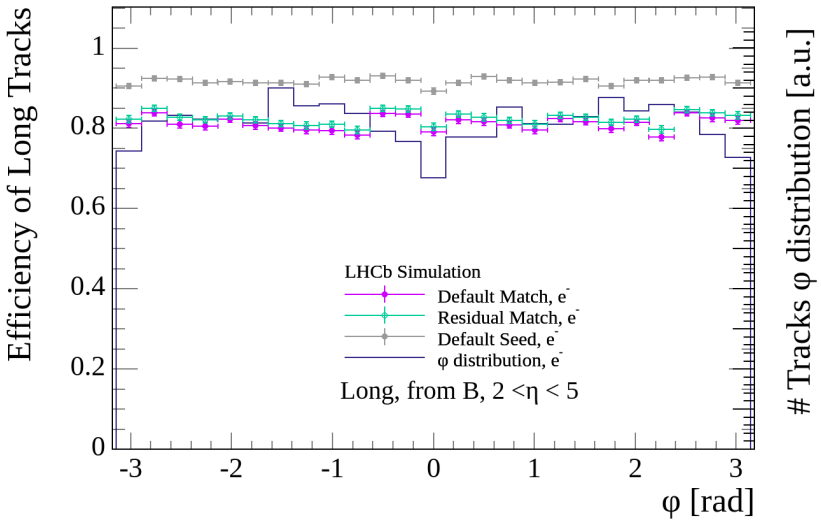


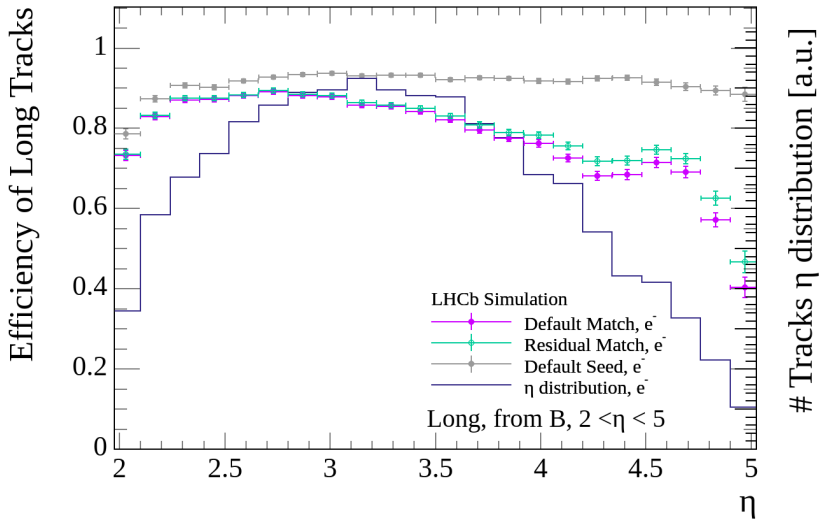


Backup: Residual Matching

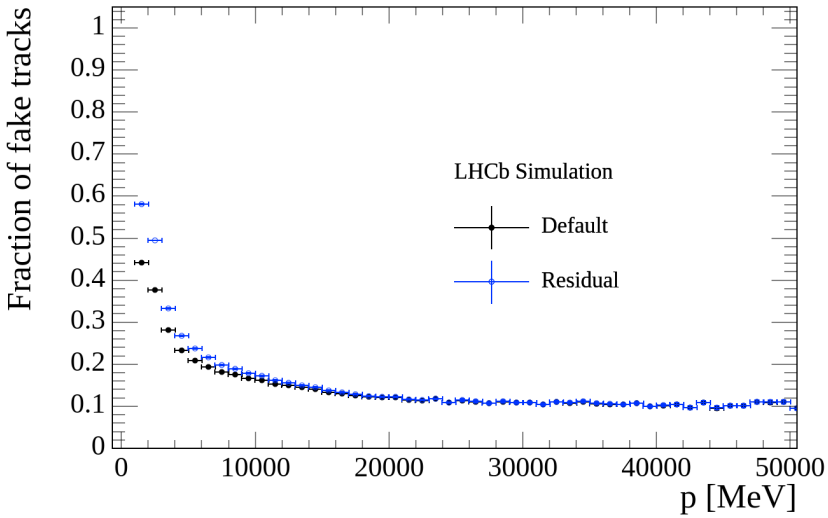


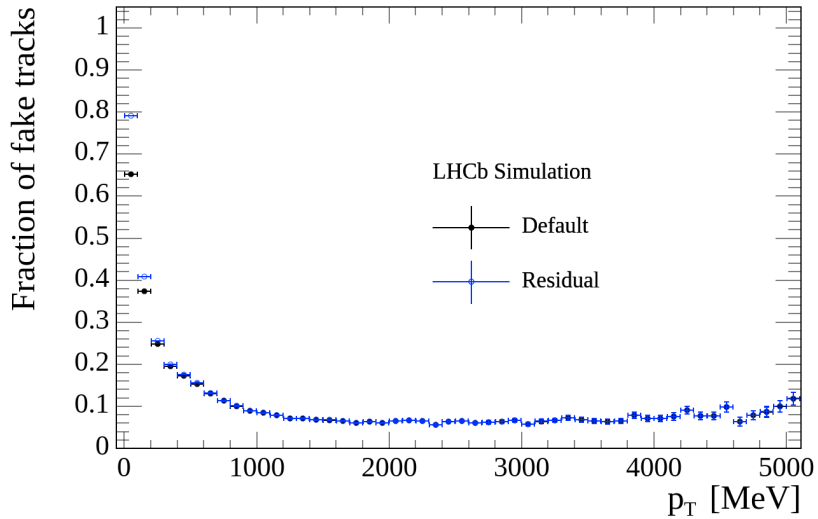


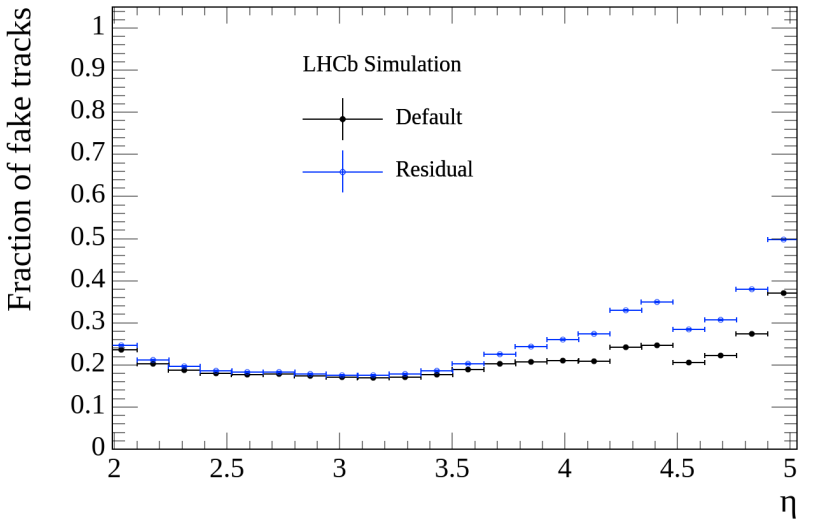


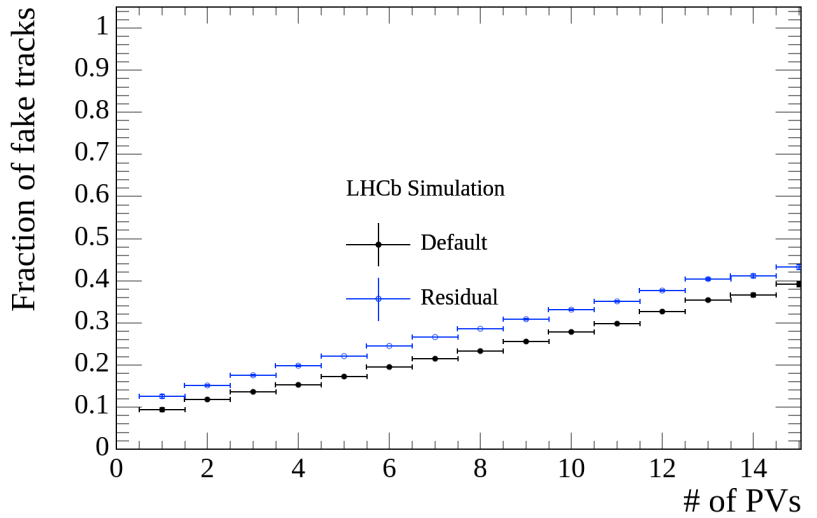


Backup: Residual Matching









Backup: Residual Weights

```
const auto ResfMin = std::array<simd::float_v, 6>{
    {2.32376150961e-05, 1.20999845876e-06, 3.0517578125e-05, 0.000152587890625,
    5.18634915352e-05, 3.16649675369e-08}};
const auto ResfMax = std::array<simd::float_v, 6>{{29.999835968, 0.448848098516,
                                                   490.75402832, 499.918823242,
                                                   1.29696559906, 0.148829773068}};
const auto ResfWeightMatrix0to1 = std::array<std::array<simd::float_v, 7>, 8>{
    {{0.972643778287334, 0.945437530240695, -1.40069143935294,
      -15.6034120045671, 1.14493675557278, 6.76331107008671, -6.58864627844693},
     {1.99177578845469, -13.3678019612632, 8.38118795560118, 1.73988710441318,
      -4.61454323644065, 5.29554800958296, 1.796743670204},
     {0.154471209290507, -6.25196675947653, 5.03239643950246, 17.3659761341648,
      -6.54695139344376, -13.0321058473978, -2.79459536100855},
     {-1.91255962568079, -8.6500289238652, 11.3312847667967, 13.5402314908838,
      -2.61341614761575, 6.63476937311634, 18.5047027165893},
     {-13.4902851128642, 5.03927112314943, -7.35289370328568,
      0.0572131890099181, -1.6142848069816, -3.07255458814266,
      -18.9635216594601},
     {1.88222476973218, 6.53087839421258, 2.08080853139342, 0.816872513930955,
      1.76981234909237, -8.6501994076645, 3.81699174241397},
     {-0.79066972900182, -0.617757099680603, 0.740878002718091,
      0.681870030239224, -1.20759406685829, 0.769290467724204,
      -1.8437808630988},
     {1.96787188749046, 0.680940366397391, 0.050263650384077, 1.68306844400001,
      1.12938262301514, 0.122157098634831, -0.887283402159991}}};
const auto ResfWeightMatrix1to2 = std::array<std::array<simd::float_v, 9>, 6>{
    {{-2.73702380879827, 1.22468365009789, 2.40149928694528, 0.276654711632341,
      -0.947460759127638, -0.94795299724562, 1.63438201788813,
      -1.41515589667229, -0.708508928627869},
     {-0.408168817589508, -0.542699435360695, -0.336829708223667,
      -0.507220427829013, 0.533181686353704, -0.0512849135791123,
      -1.61531096417457, 0.0991539876010671, 4.00684418941464},
     {0.401110123287066, -0.82501422982477, -0.82214087163611,
      -2.13310745114762, 0.656608219190029, -1.54611499475089,
      -0.825543426908553, -1.92246825444023, -2.49920928064247},
     {0.743417630960188, -2.54297207137451, 0.868639896626588, 1.21759484724959,
      -0.432278512319556, -0.682439011110067, 1.61348068527877,
      -1.70813842427554, 0.191141321065651},
     {0.601790057732671, -2.70865568575877, -0.949516903771233,
      1.41807664967738, 0.0135866328882364, 1.63463920593405,
      -0.848898627795279, 0.794266404867267, -4.68030461730642},
     {-0.894524549453373, -0.413420422791491, -1.27841462173856,
      -0.921761527738667, 1.7613032977725, -1.20901458126865, -1.52203810494393,
      1.63899587513312, 3.18360564985773}}};
const auto ResfWeightMatrix2to3 = std::array<simd::float_v, 7>{
    {-0.468166794846483, 0.905418443044577, 0.345720533590786,
    0.626519340549303, -0.564753919345451, 0.871170117133406,
    -2.29725166588317}};
```