# Reviving the parametrised Kalman filter

Lennart H. Uecker[1], Michel De Cian[1], Thomas Boettcher[2]

[1]Physikalisches Institut, Uni Heidelberg

[2]University of Cincinnati

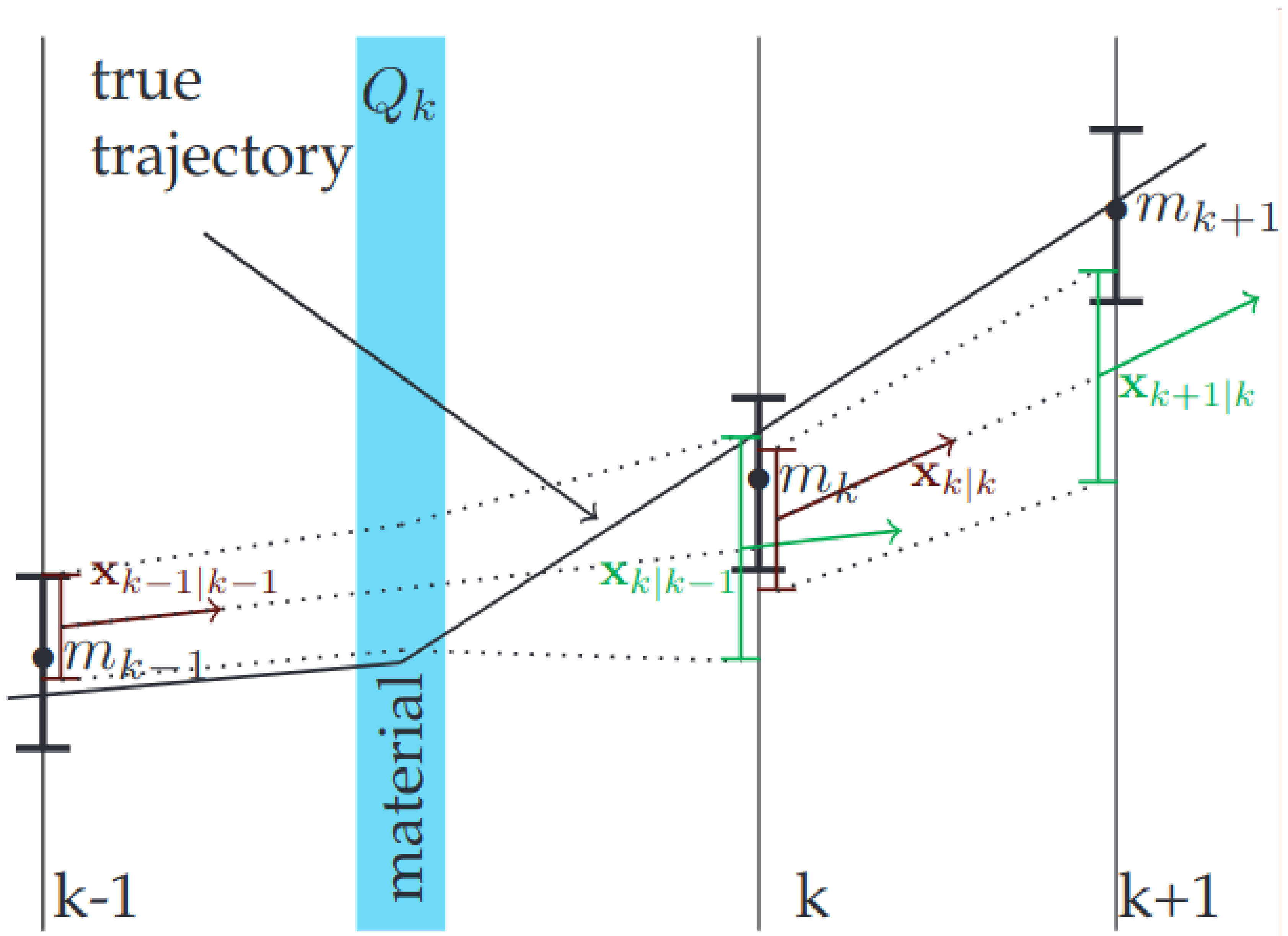
02.12.24

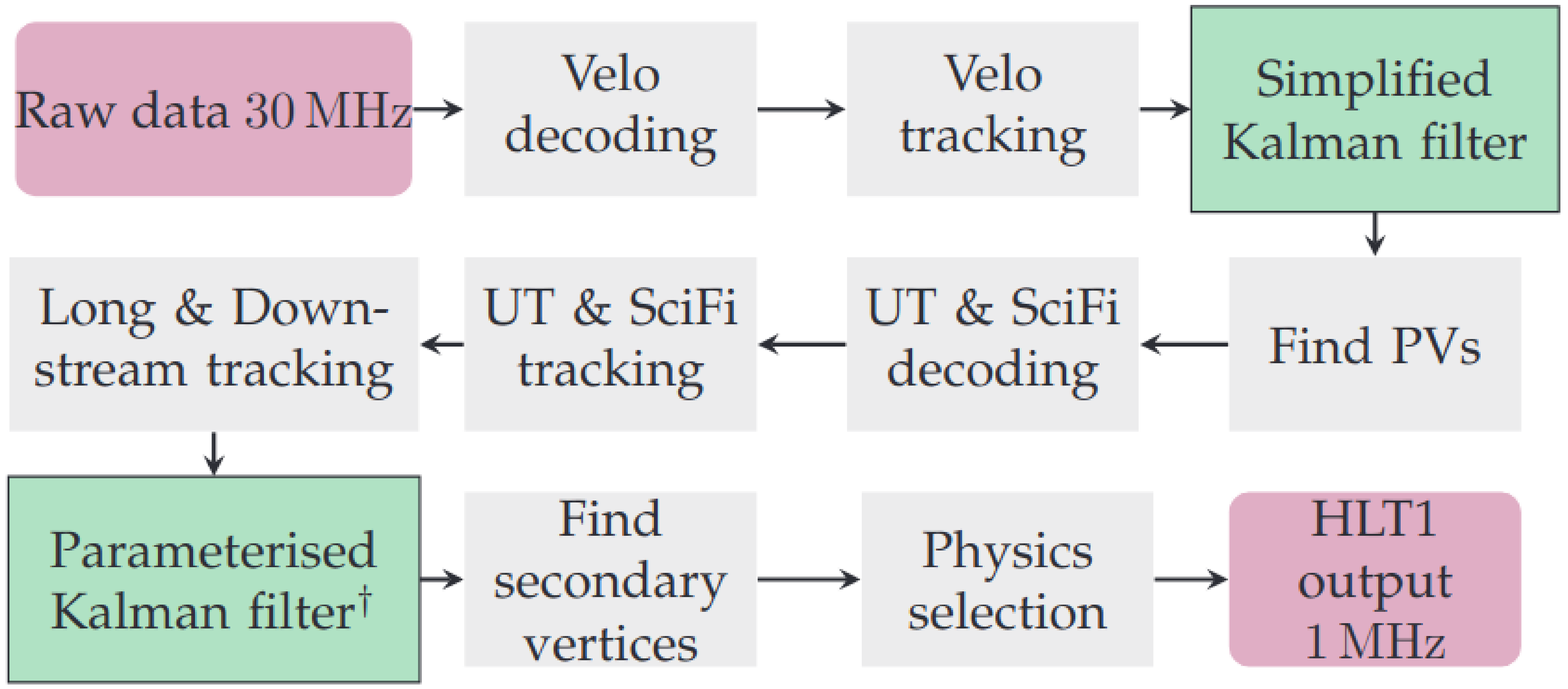
114th LHCb Week, Computing and software

FSP LHCb
Erforschung von
Universum und Materie

# What?

- Reviving the parametrised Kalman filter (parKF).
  Merge Request: !1693




† Currently simplified Velo fit in place of parameterised Kalman filter

# What is the parametrised KF

| #Pars | |
|---|---|
| V: | 20 |
| V$\rightarrow$ UT: | 60 |
| UT: | 144 |
| UT $\rightarrow$ T: | 3000 x 28 |
| T: | 944 |

- Solving the differential equation for the magnetic field is hard.
- So is looking at the geometry for scattering.
  - Large computing and memory demand. Not suited for GPUs/Allen.
- Do the hard part offline and parametrise the solution.

- Hardest part is UT $\rightarrow$ FT extrapolation. 9(7)-order polynomial for x(y) in 60 x 50 bins over x & y.
- Parametrise scattering in each layer for noise matrix Q.
- Parametrise transport between layers of VP, UT & FT $\rightarrow$ Much simpler.

Noise for UT extrapolation

```
// Define noise
KalmanFloat xErr = par[2] * fabsf(dz * x_old[4]);
KalmanFloat yErr = par[4] * fabsf(dz * x_old[4]);
KalmanFloat txErr = par[12] * fabsf(x_old[4]);
KalmanFloat tyErr = par[15] * fabsf(x_old[4]);

// Add noise
Q(0, 0) = xErr * xErr;
Q(0, 2) = par[14] * xErr * txErr;
Q(1, 1) = yErr * yErr;
Q(1, 3) = par[17] * yErr * tyErr;
Q(2, 2) = txErr * txErr;
Q(3, 3) = tyErr * tyErr;
```

Extrapolation in UT

```
// extrapolate state vector
// tx
x[2] += dz * (par[5] * ((KalmanFloat) 1.e-1) * x[4] + par[6] * ((KalmanFloat) 1.e3) * x[4] * x[4] * x[4] +
              par[7] * ((KalmanFloat) 1e-7) * x[1] * x[1] * x[4]);
// x
x[0] += dz * (par[0] * x_old[2] + (((KalmanFloat) 1.0) - par[0]) * x[2]);
// ty
x[3] += par[10] * x[4] * x[2] * std::copysign((KalmanFloat) 1.0, x[1]);
// y
x[1] += dz * (par[3] * x_old[3] + (((KalmanFloat) 1.0) - par[3]) * x[3]);
```

# History of the parKF

- First developed as a fast CPU KF
- Implemented on GPU

- This work: !1693
  (started 5 months ago)

LHCb-DP-2021-001
May 17, 2021

**A parametrized Kalman filter for fast track fitting at LHCb**

P. Billoir[1], M. De Cian[2], P. A. Günther[3], S. Stemmle[3,†]

[1] *LPNHE, Sorbonne Université, Paris Diderot Sorbonne Paris Cité, CNRS/IN2P3, Paris, France*
[2] *Institute of Physics, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland*
[3] *Physikalisches Institut, Ruprecht-Karls-Universität Heidelberg, Heidelberg, Germany*
[†] *Author was at institute at time work was performed.*

**The LHCb GPU high level trigger and measurements of neutral pion and photon production with the LHCb detector**

by

Thomas J. Boettcher

B.S., Indiana University (2015)

Submitted to the Department of Physics
in partial fulfillment of the requirements for the degree of

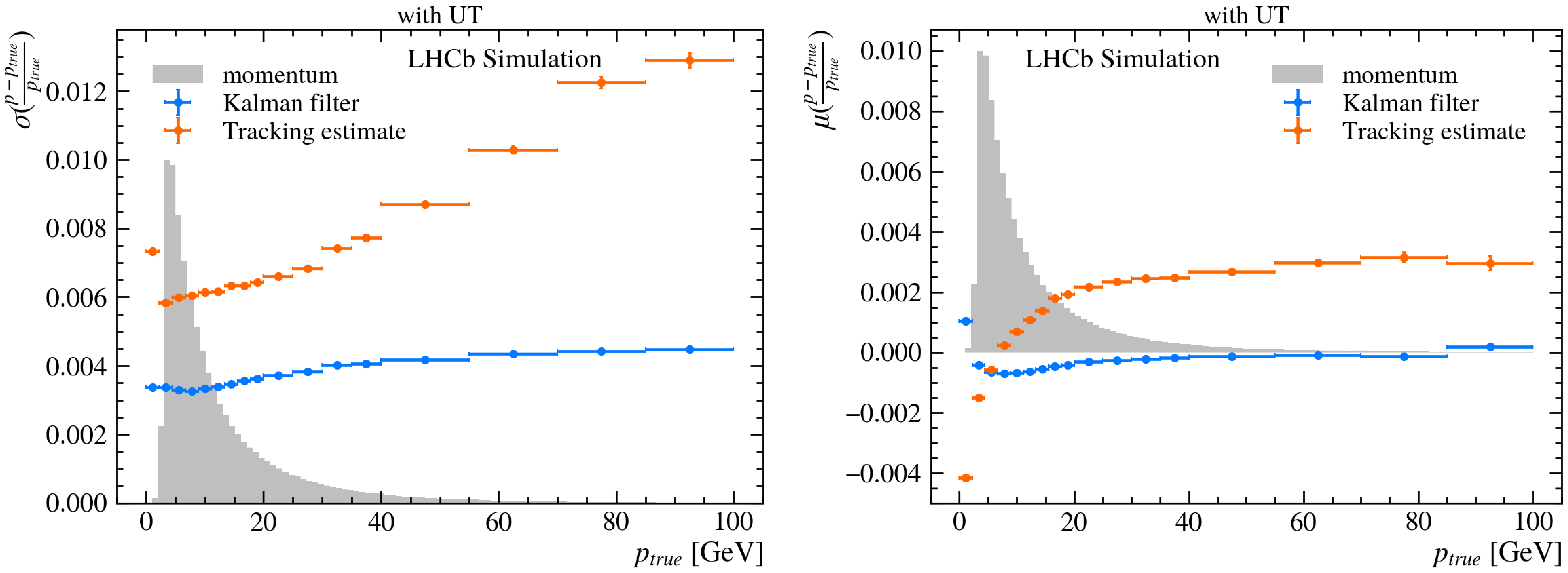
Doctor of Philosophy

at the

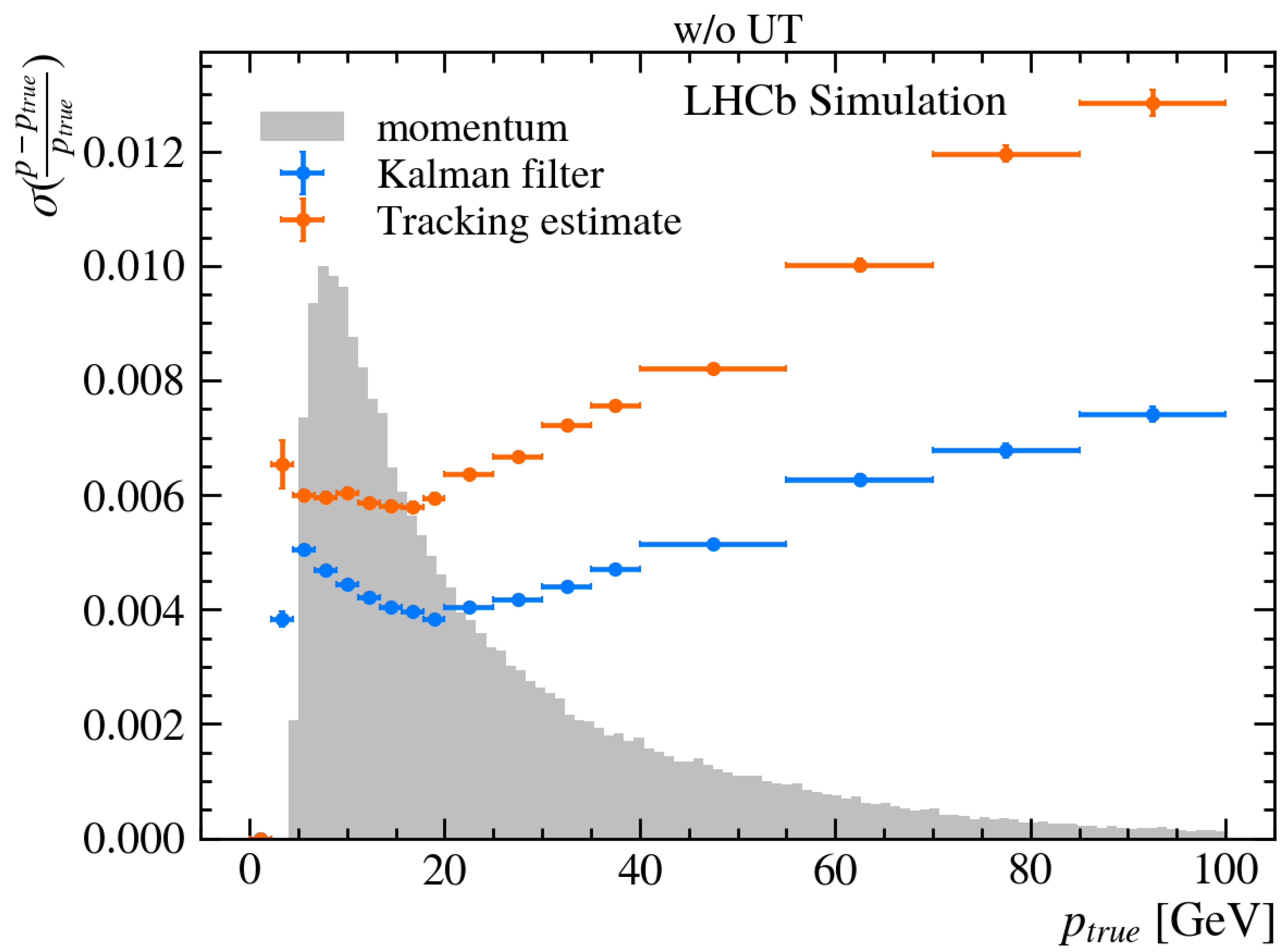
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

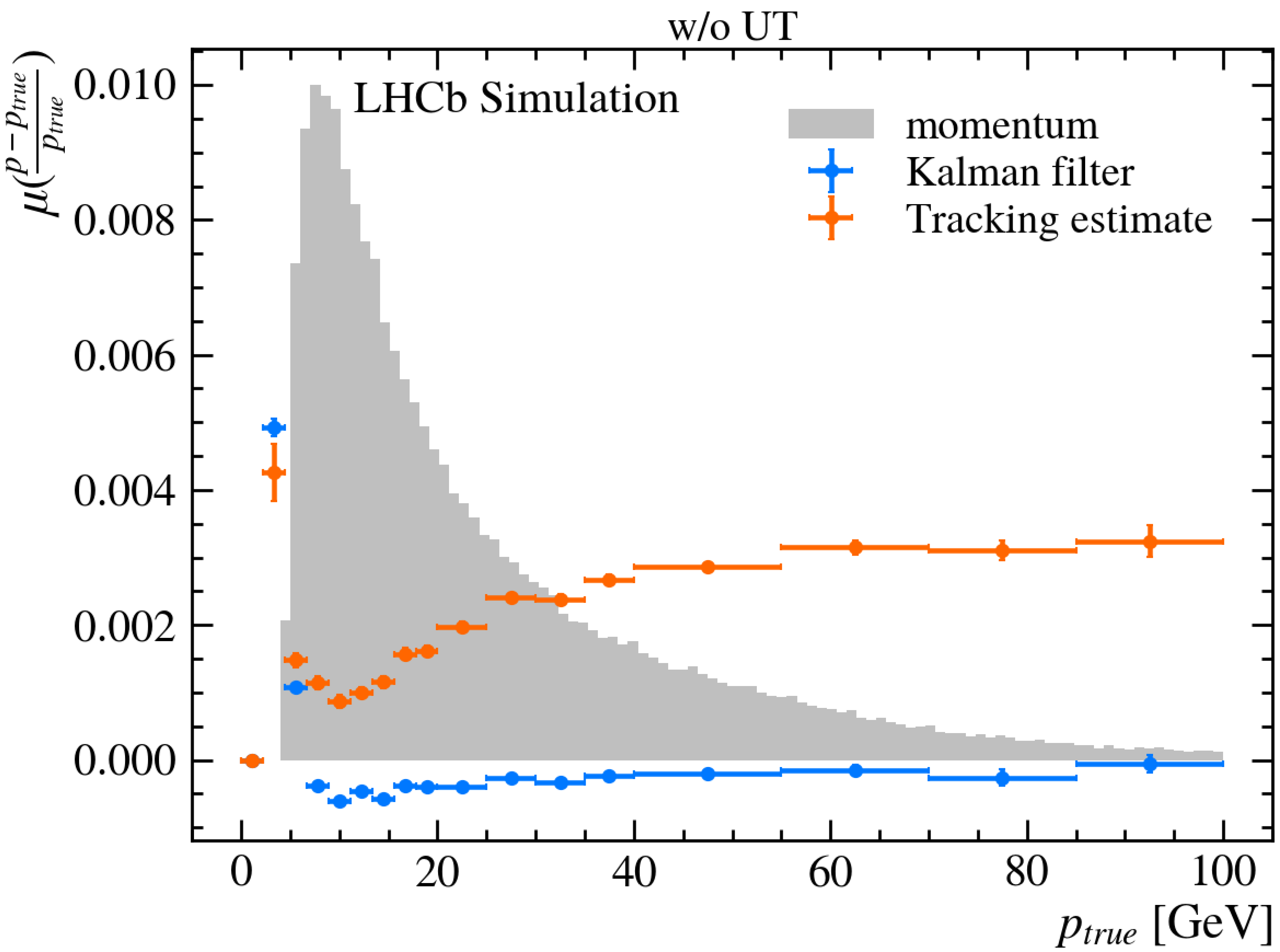
February 2021

# Momentum resolution – w/ UT

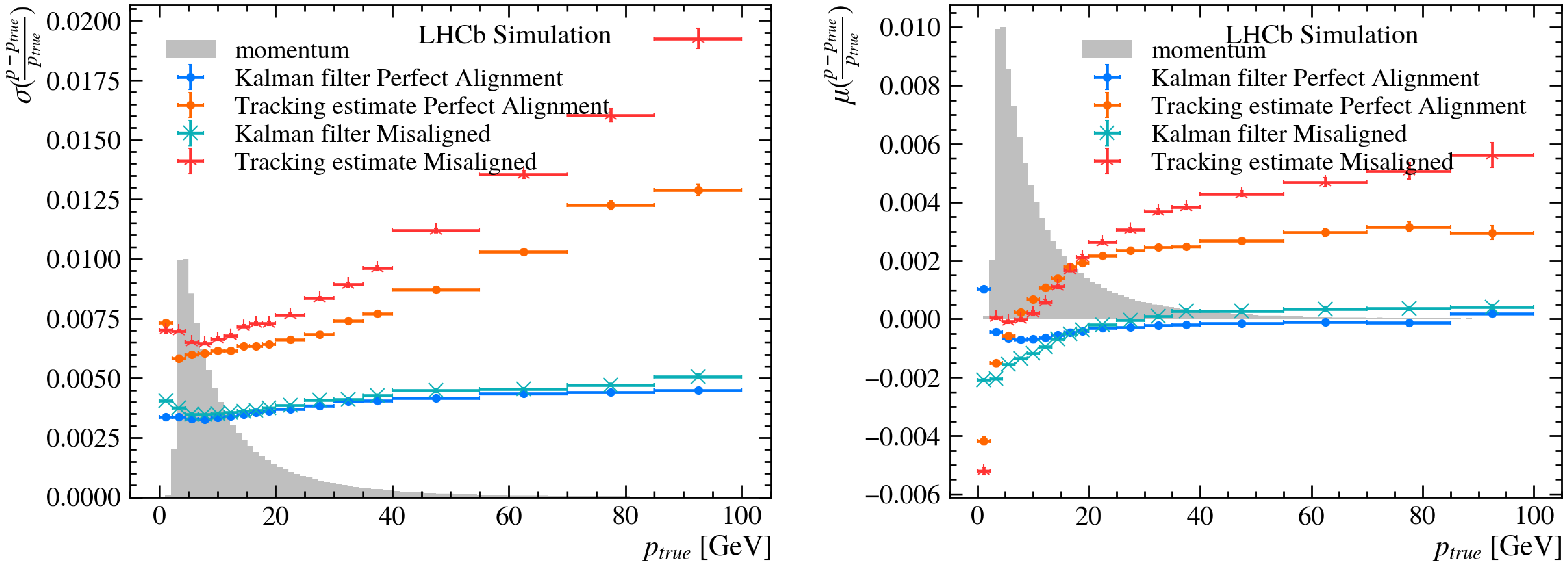


# Momentum resolution – w/o UT

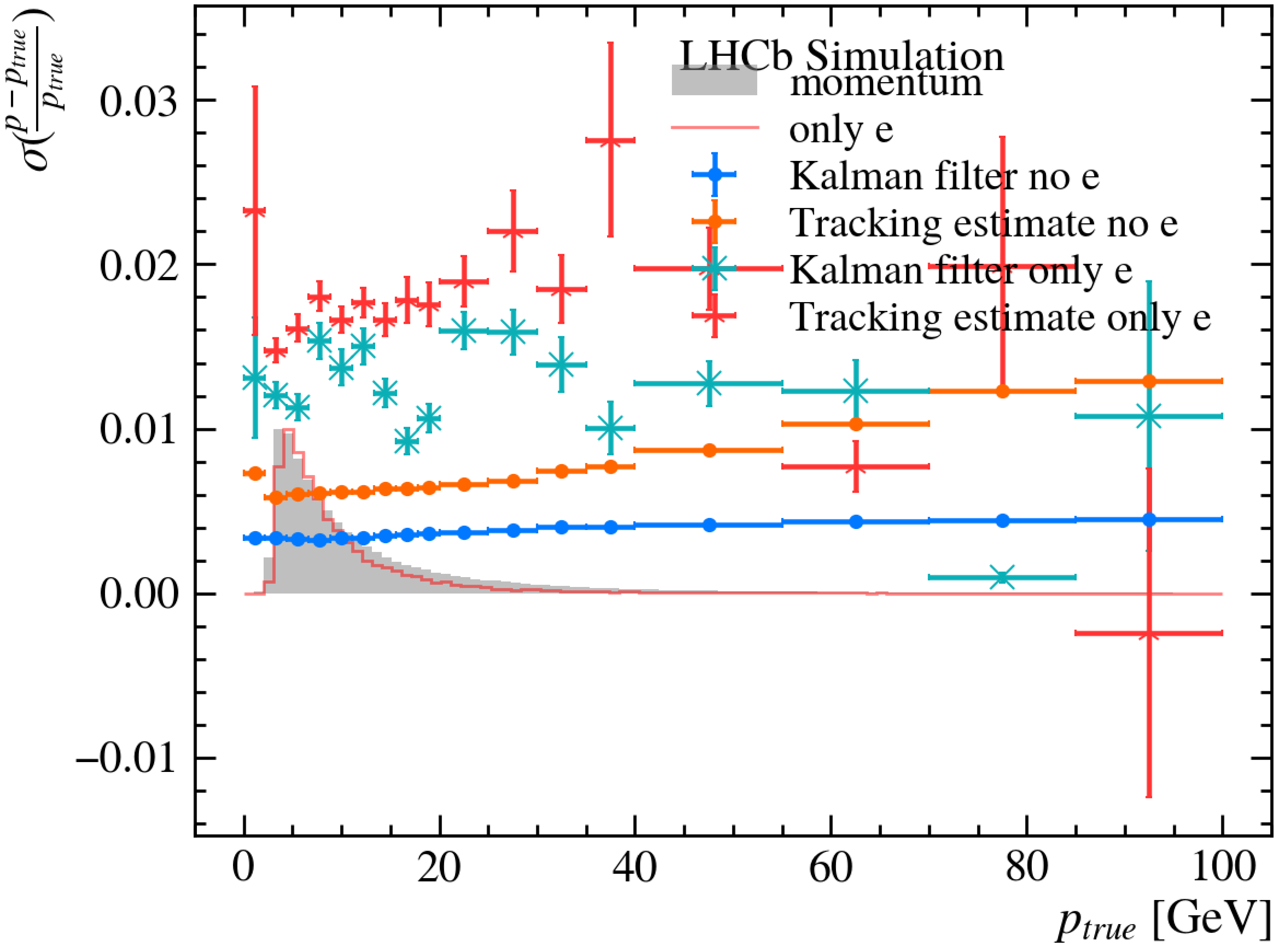


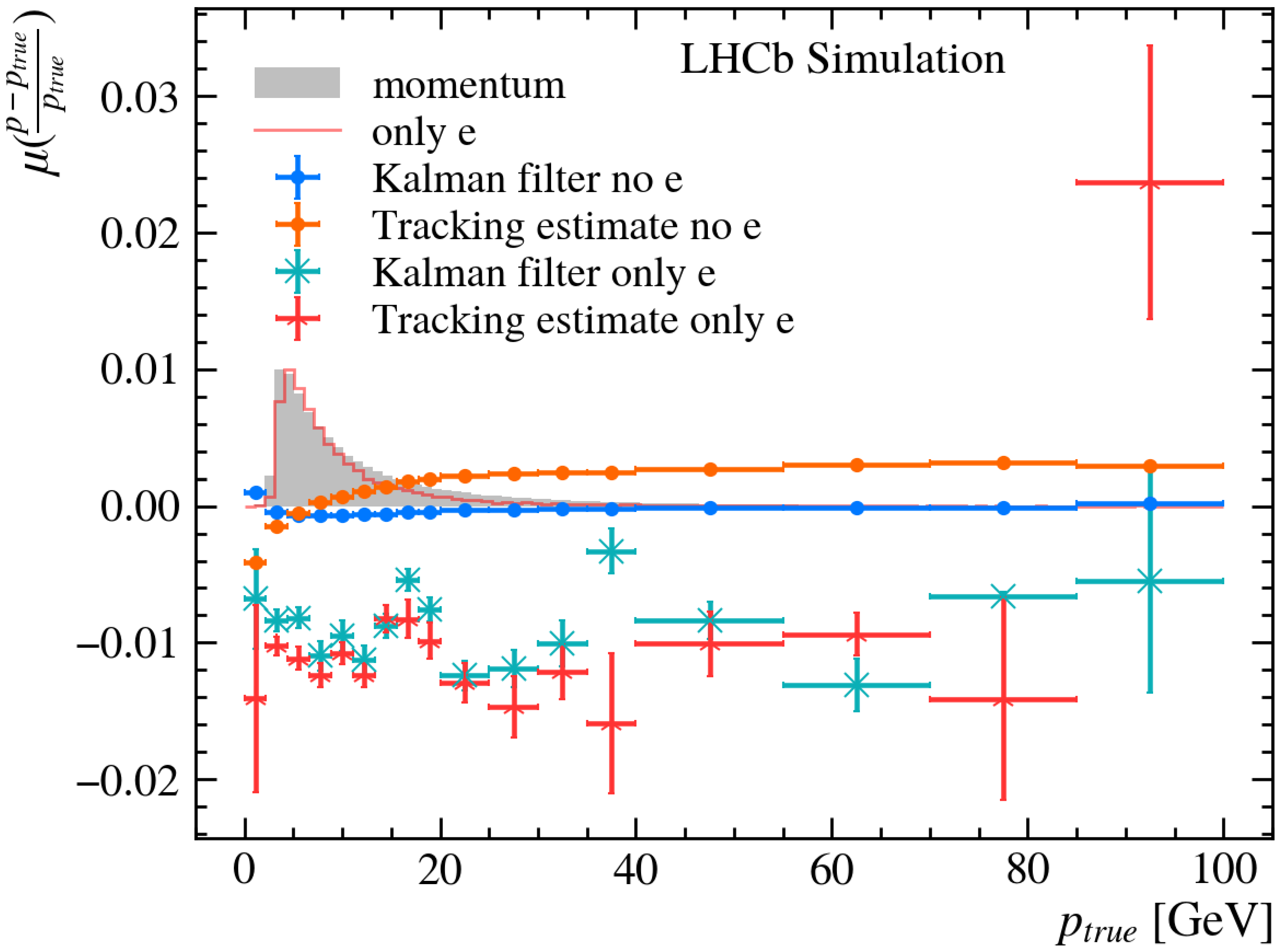



# Misaligned (in FT)



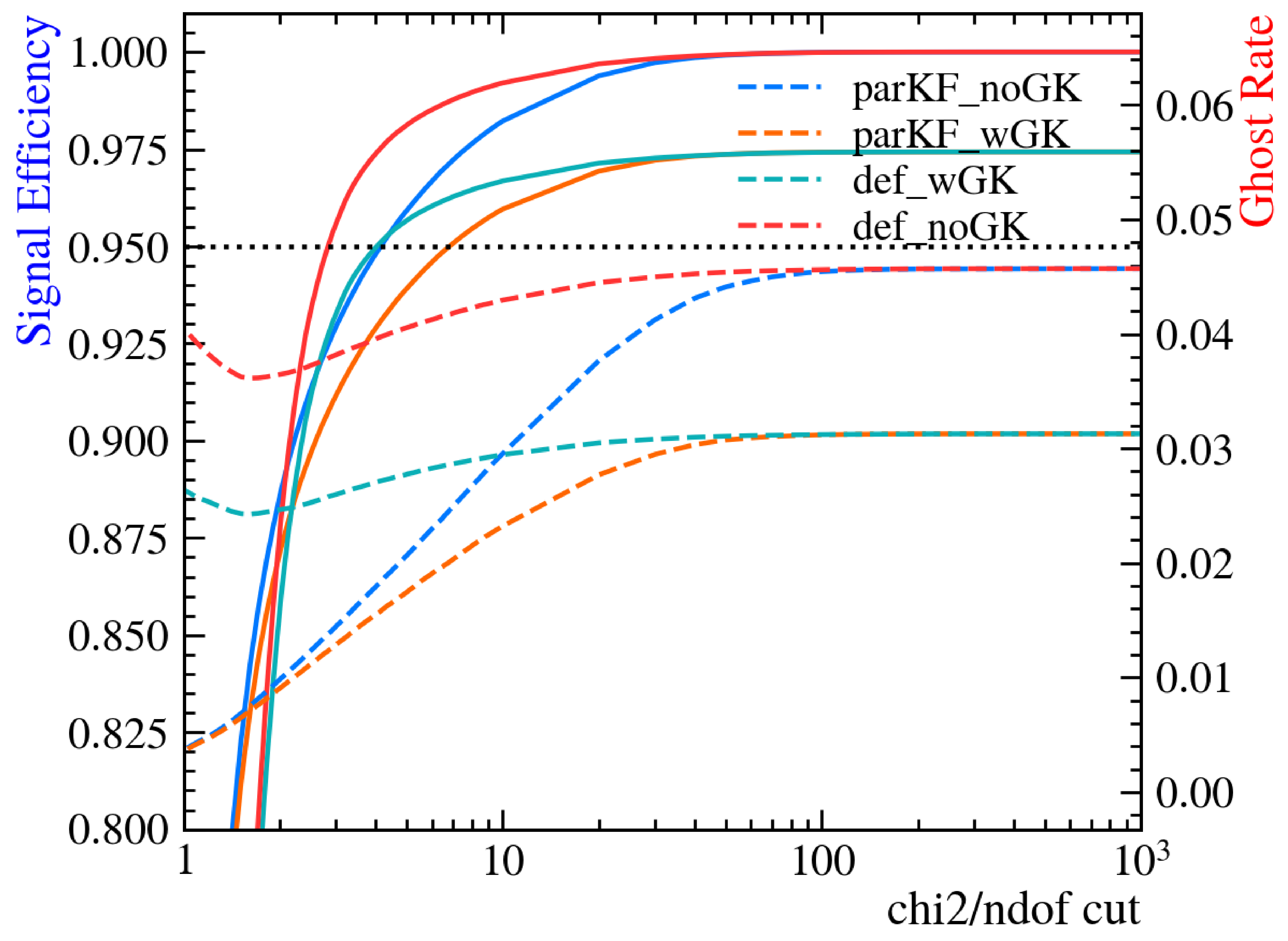
# Electrons – with misalignment

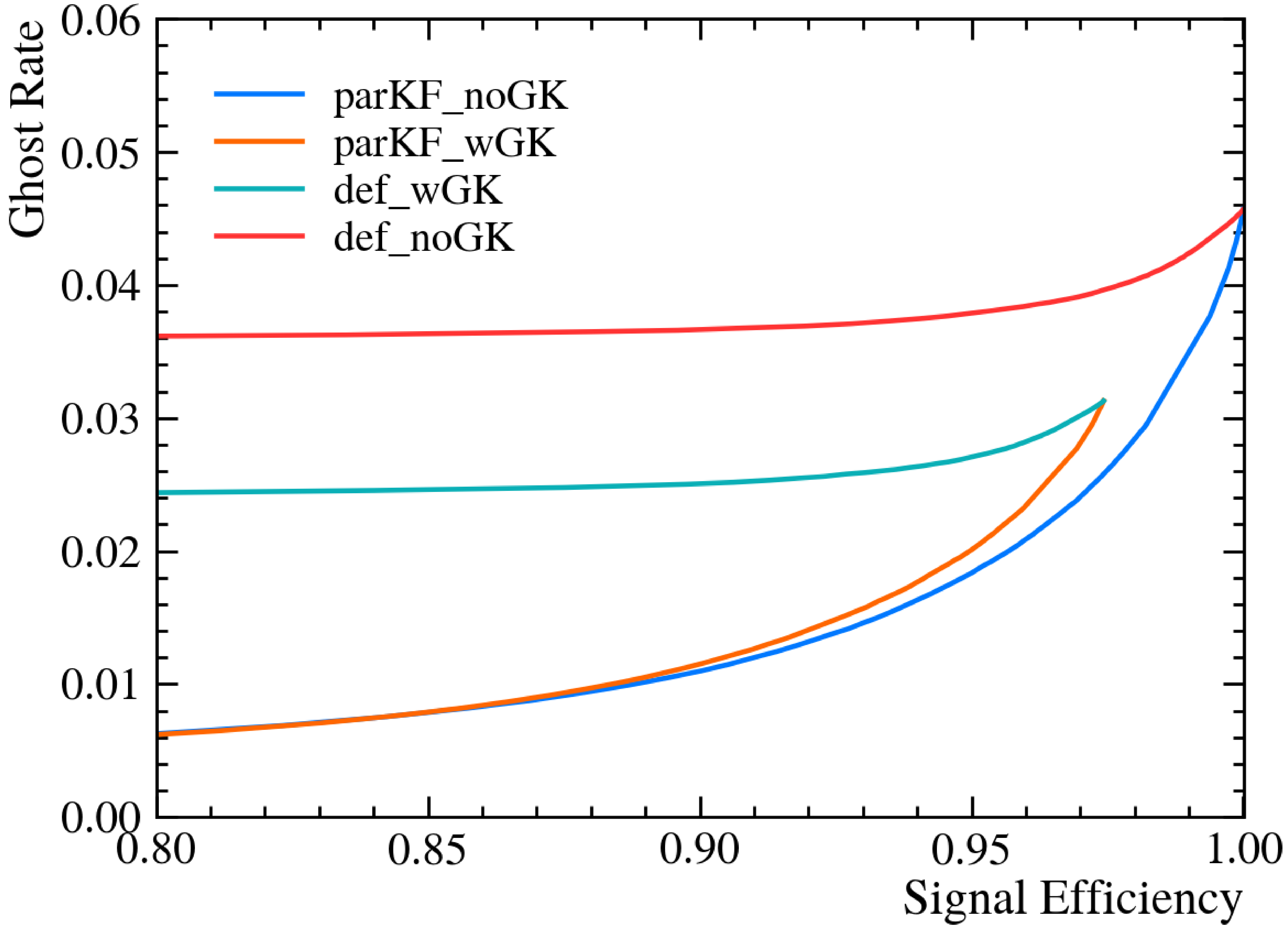


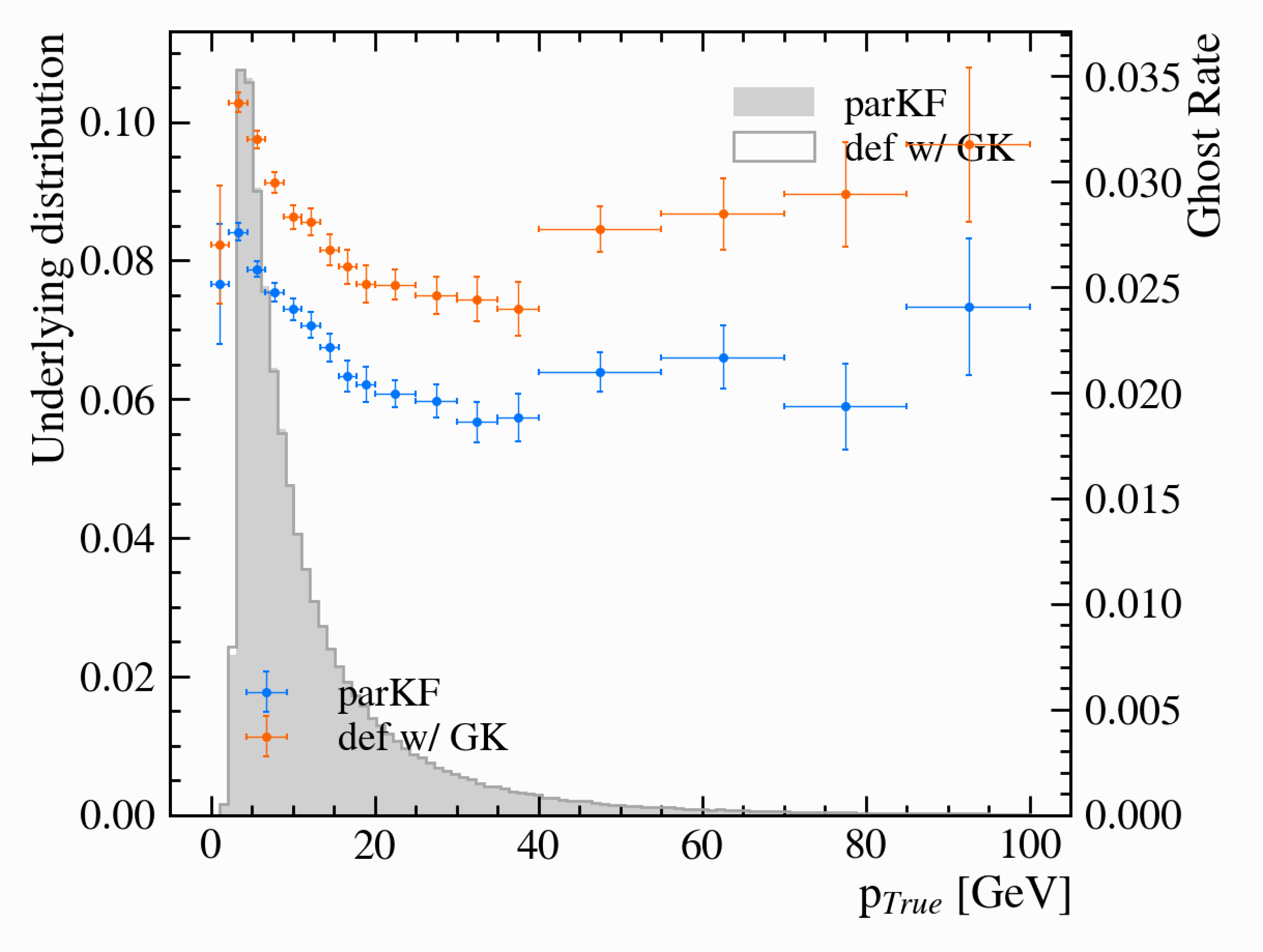
# Ghost rates and the ghost killer

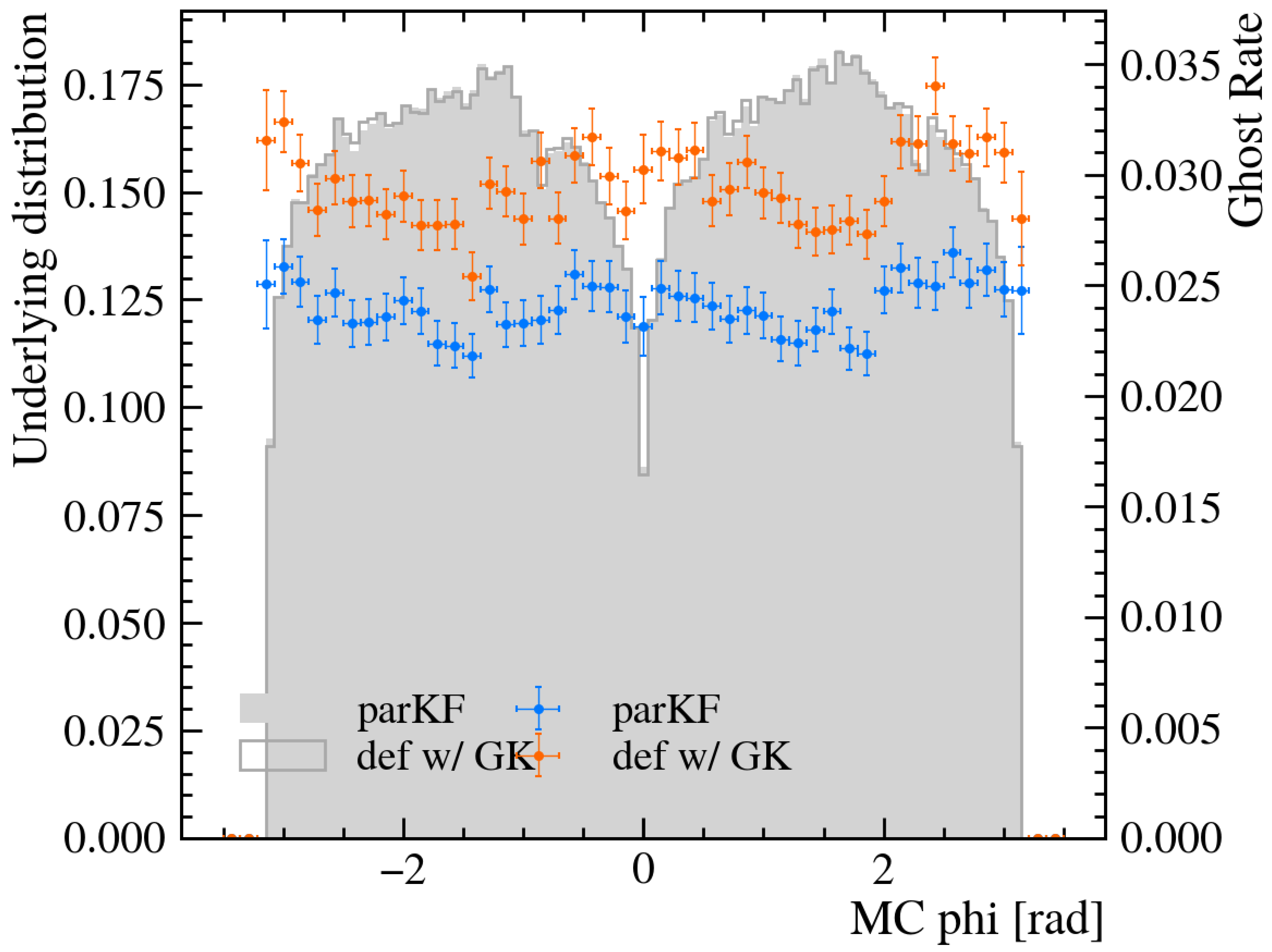
- ParKF calculates a $\chi^2$ → use to reject Ghost tracks



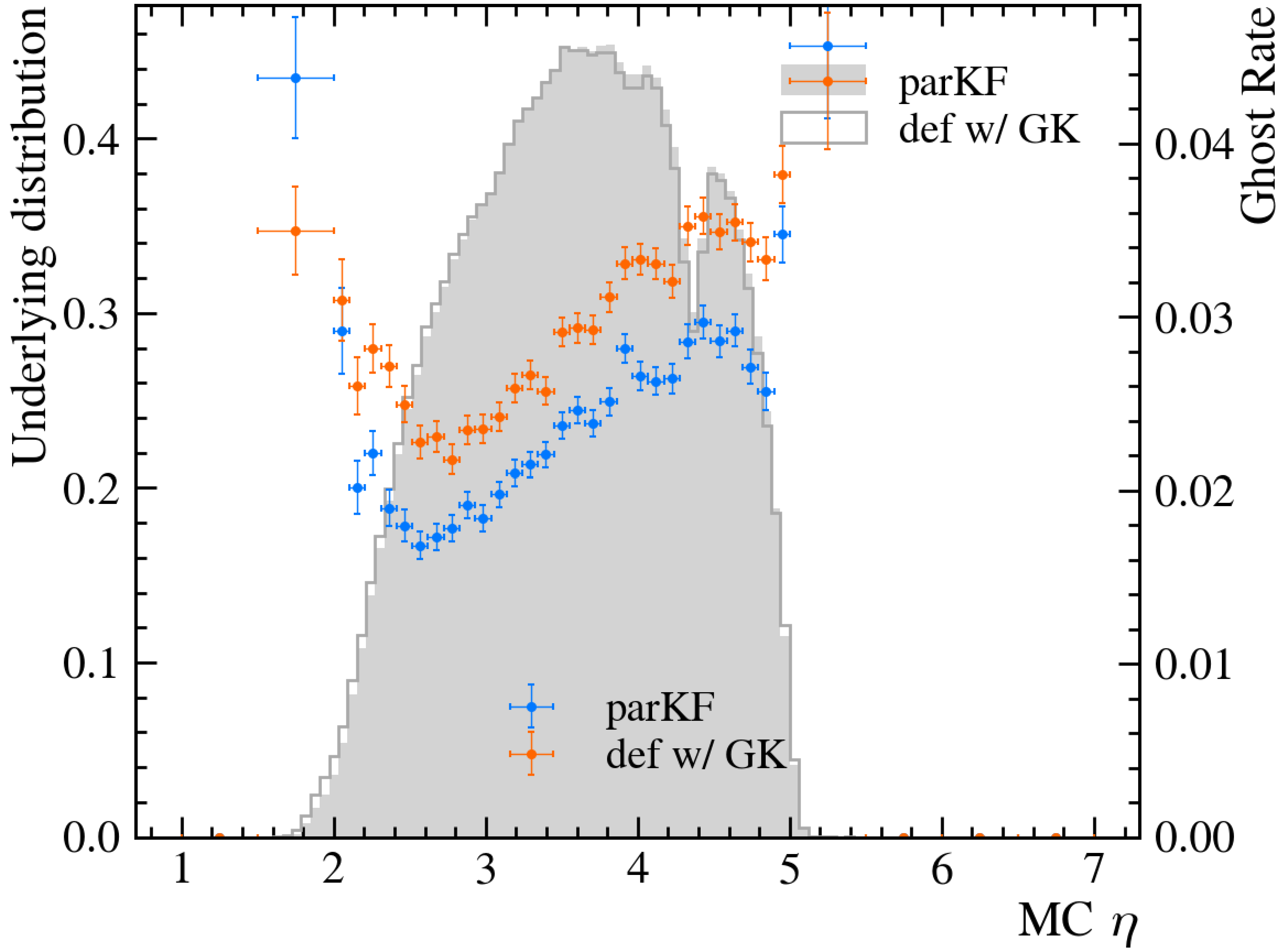



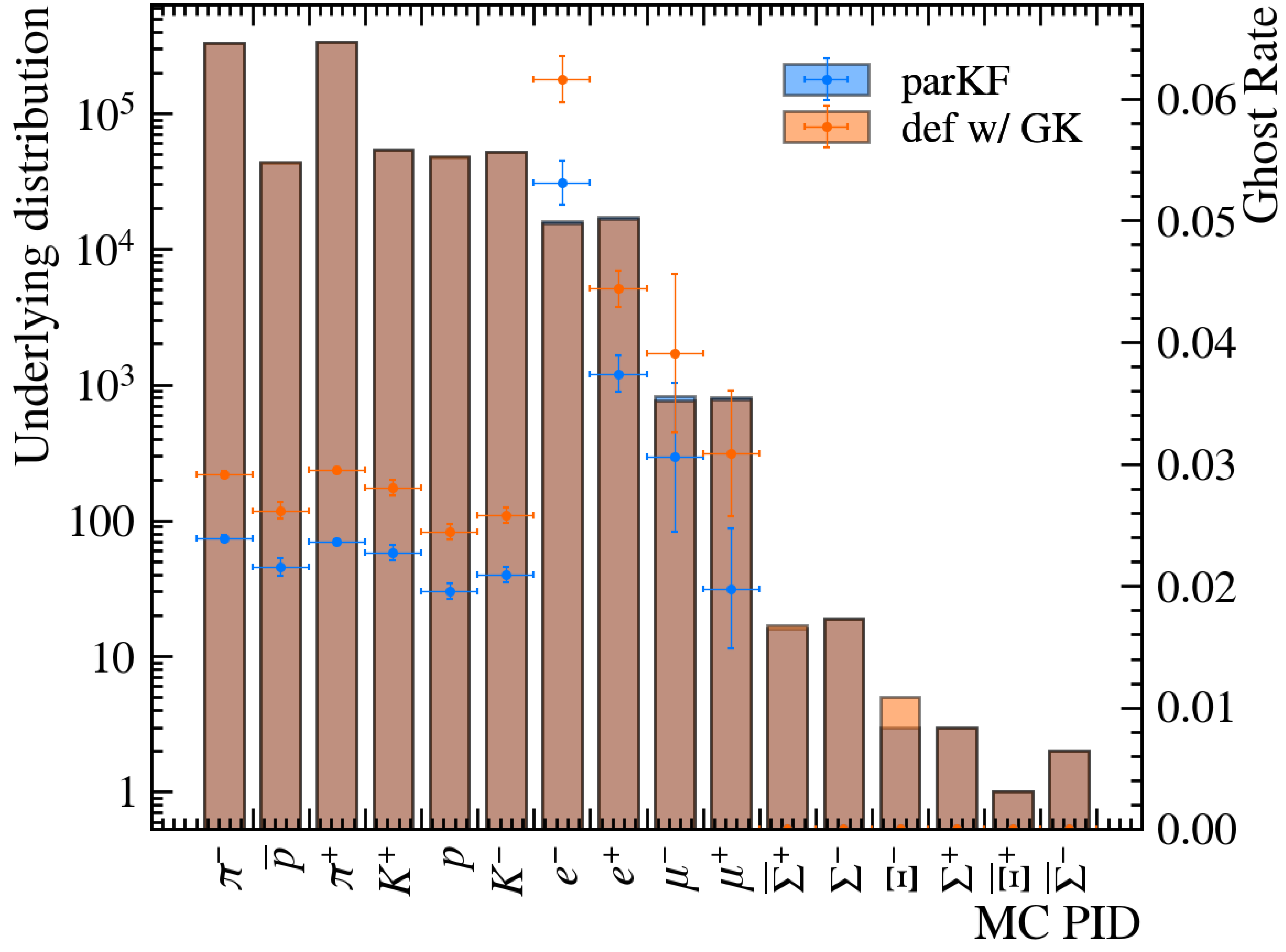
## Differential Ghost rate @ 97% signal eff.



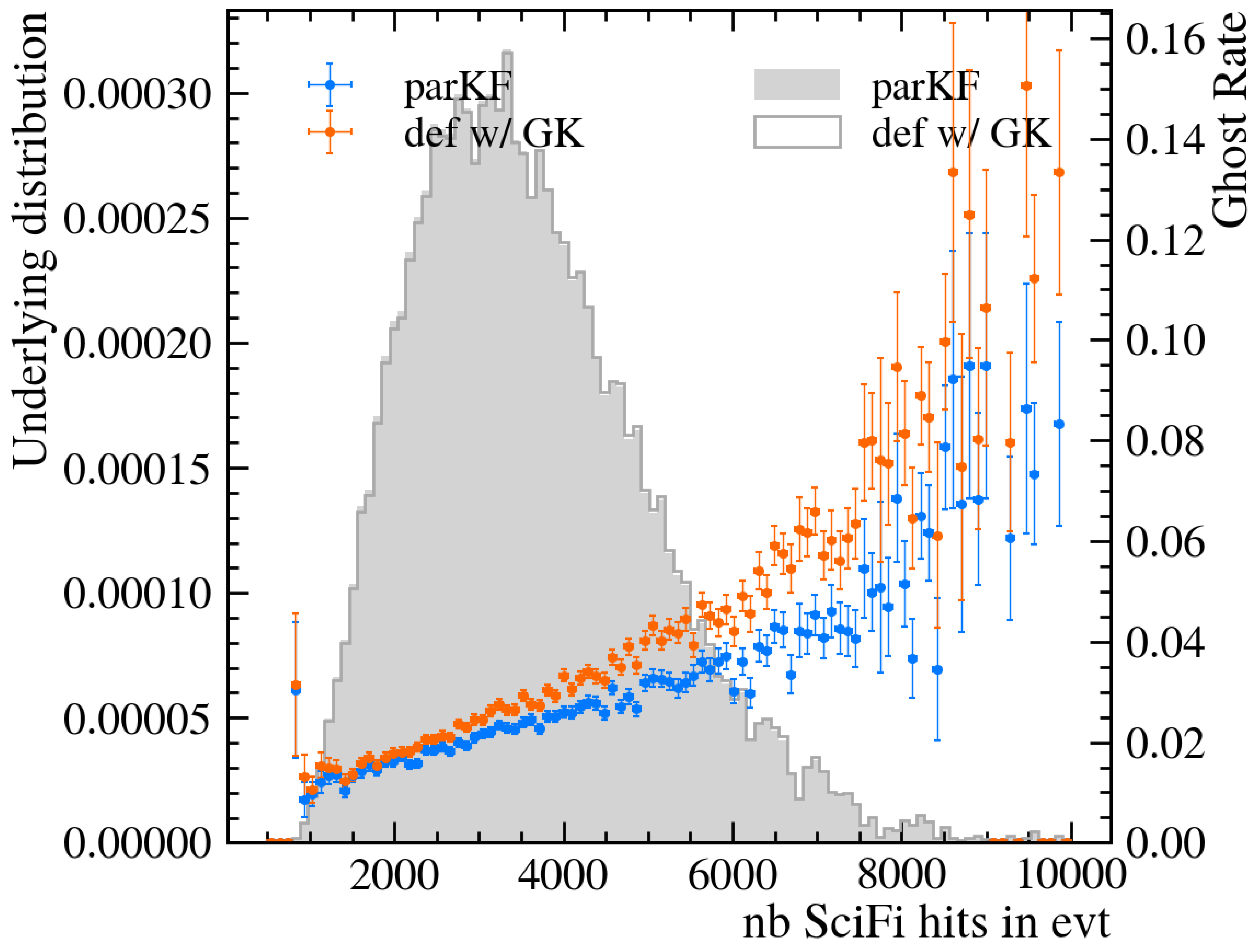



# Differential Ghost rate @ 97% signal eff.

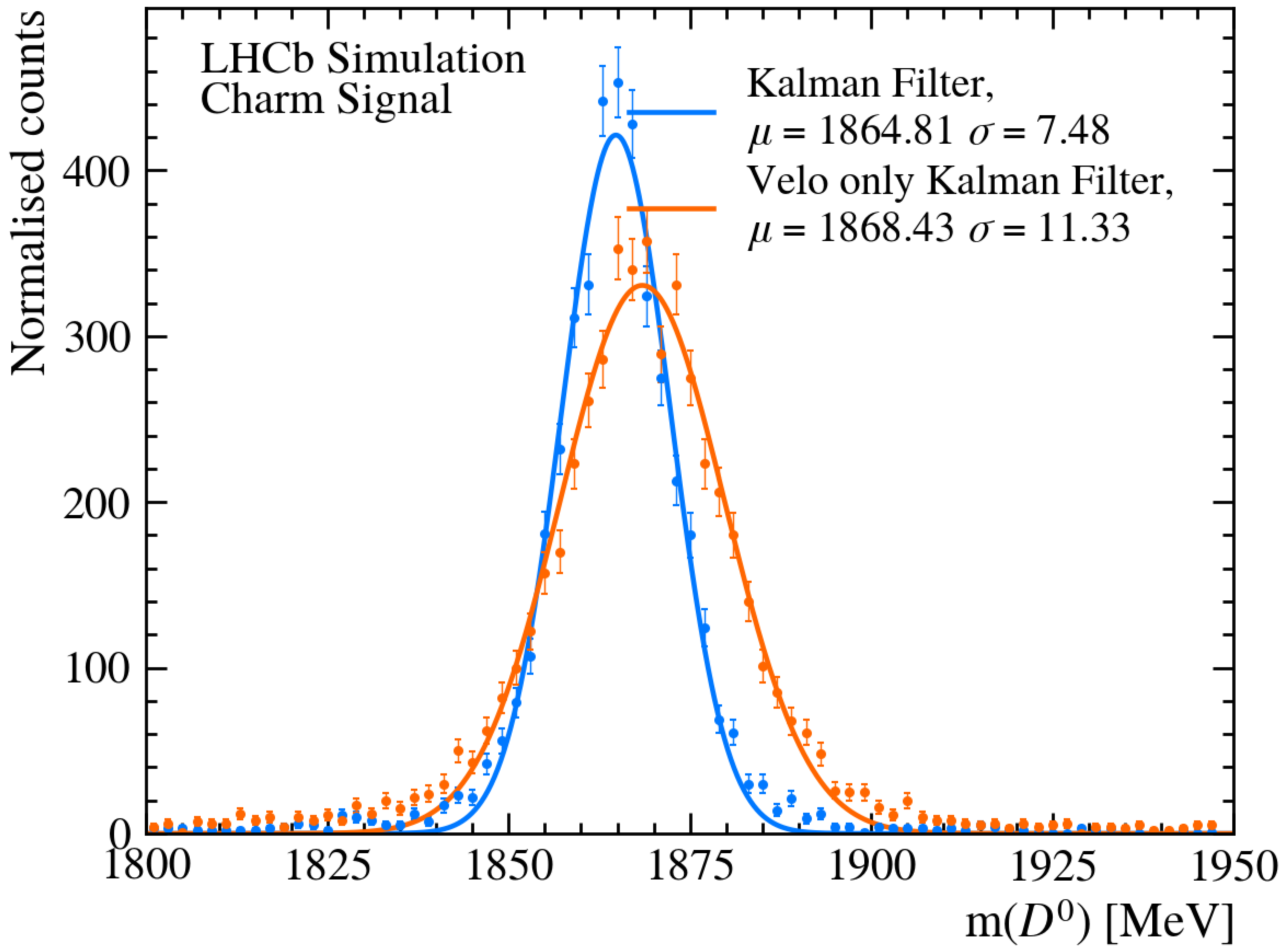


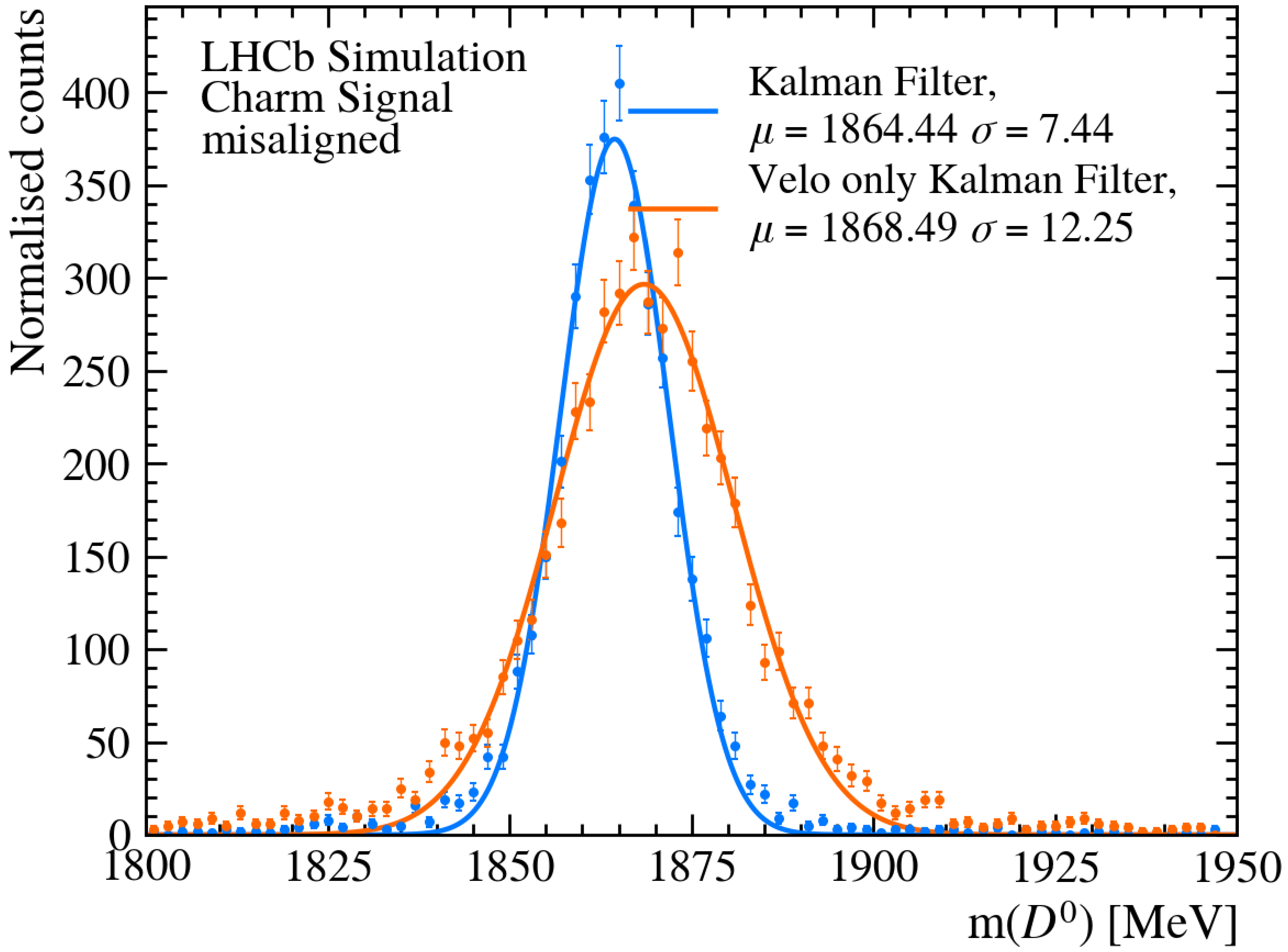
# Differential Ghost rate @ 97% signal eff.



# Charm in MC



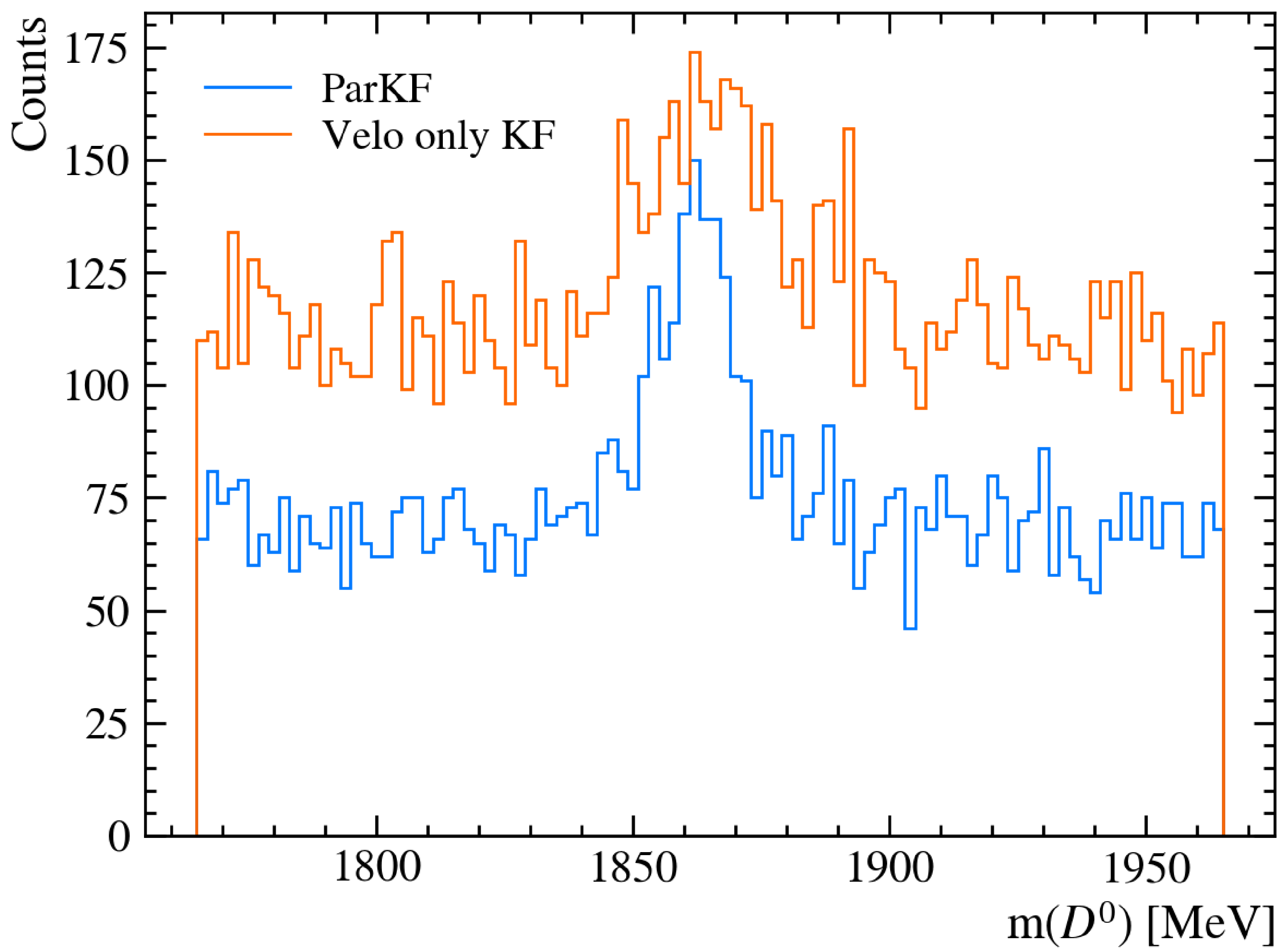

**Realistic misalignment in the FT only.**



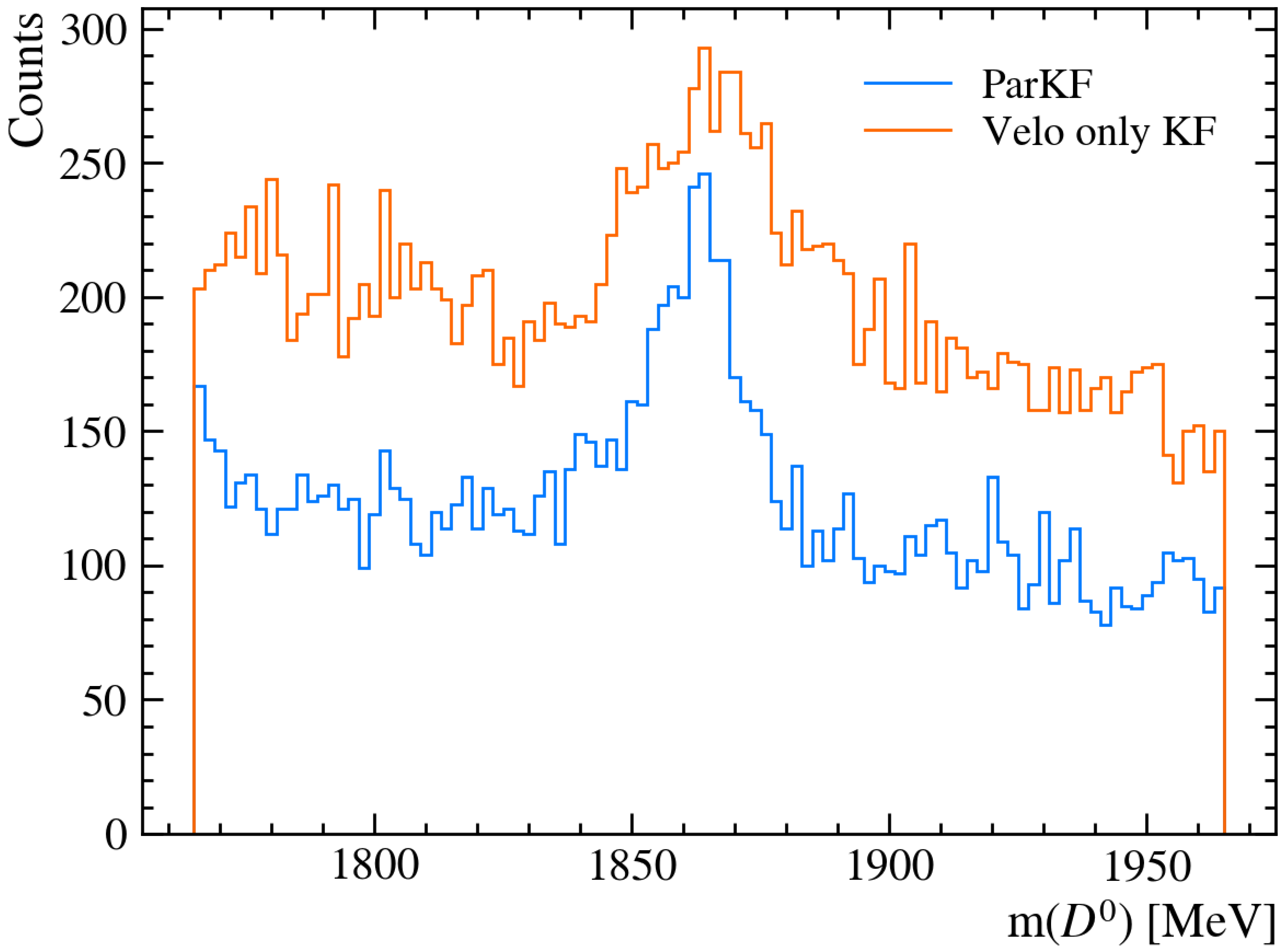
## Selection in data

- MEP dumps from 09.09.24.
- 5 Million evts
- Avg. Mu 4.3, MagUp

HLT1 TwoTrackMVA line



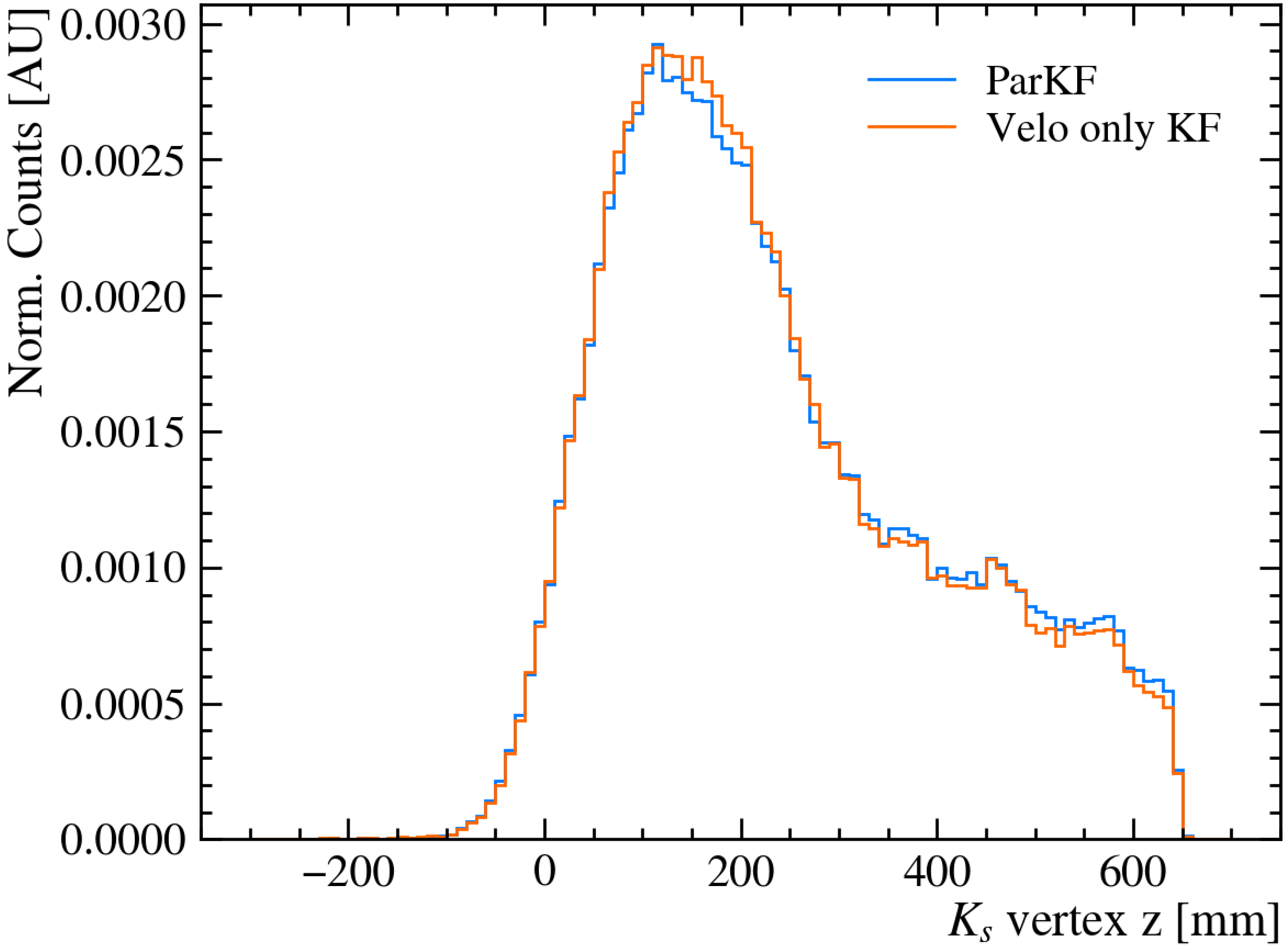

D0 → Kpi Line



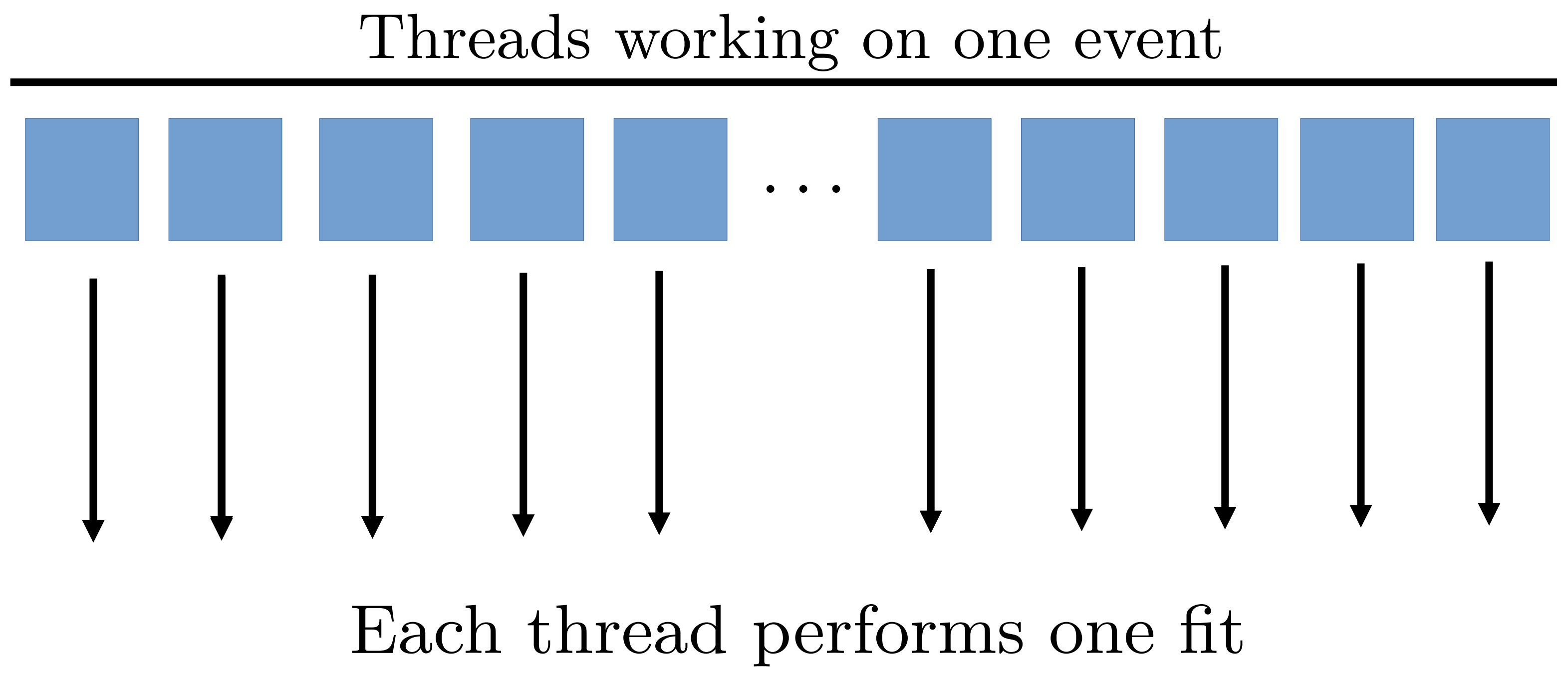
# Parametrisation & large displacements

- From first principles, no strong effects
- Parametrisation has weak assumptions on the origin vertex
  - Should still hold for all long tracks

- No large inefficiencies on Ks→ ππ trigger line based on Vertex z



# Runtime performance




- *NVIDIA RTX A5000 throughput change -29.59%†*
- Changes to thread layout and misc.
  - No physics performance impact
- **Now:** *NVIDIA RTX A5000 throughput change -22.84%*
- *WIP:*
  - *Memory optimisation. (maybe ~18%)*

*† In previous presentation a lower throughput change was claimed because I did the test wrong.*

# What's next:

- Need to rewrite code to create parametrisations
  - Except the UT $\rightarrow$ FT, we already have that.
- Implementation for Downstream tracks.
- Speed up

- Outlier removal.
- Smoothing. (out of scope of HLT1)
- ...