

Three Track Cascade and NTrack Vertexing, with constraints

Matt Graham, SLAC

HPS Collaboration Meeting, JLAB

October 21, 2026

SLAC NATIONAL ACCELERATOR LABORATORY

Current Vertexer: Billior/BilloirVertexer

I've seen it spelled both ways

P. Billoir, R. Fruhwirth, and M. Regler, "Track element merging strategy and vertex fitting in complex modular detectors," Nucl. Instrum. Meth. A 241 (1985) 115.

Technique makes linear approximation for track trajectories near trial vertex (x_0,y_0,z_0), allowing the resulting least-squares minimization to solved using all tracks at once

- initial guess is the beamspot mean at target….this is not a good guess for highly displaced A'!
- in ReconParticleDriver, we iterate (once) swimming the tracks to the vertex z obtained using the initial iteration.
- iterating this once seems to be enough to get accurate vertex

We use this to get N-track vertex (including our V0 2-track vertex)

beamspot and pointing-to-beamspot constraints also implemented but using Kalman gain matrix approach (similar to what's discussed in this talk)

Vertexing with Kalman Filter

Whereas the Billior fit is a “one-shot” fit, Kalman filter vertexing is sequential

- each track and each constraint is added as a “measurement” one-by-one and processed through the prediction-gain-update cycle
- Billoir et al say their approach is mathematically equivalent (you do the math), but the calculations are not the same so may not be the same numerically

W. Hulsbergen, “Decay Chain Fitting with a Kalman Filter”, https://arxiv.org/abs/physics/0503191

For fitting with constraints, this is a bit more of a unified approach than what we have now

New class in the recon.vertexing package: `KalmanVertexFitterGainMatrix`

I’ve implemented this on a branch (for now) “`displaced3prongvertex`”

CascadeVertexer and NTrackVertexer

Started out trying to implement: W. Hulsbergen, “Decay Chain Fitting with a Kalman Filter”, https://arxiv.org/abs/physics/0503191

- Gain-Matrix approach: each track and each constraint is added as a “measurement” one-by-one and processed through the prediction-gain-update cycle
- Billoir et al say their approach is mathematically equivalent (you do the math), but the calculations are not the same so may not be the same numerically
- Does not include the track parameters in the “state”, so we don’t get track refit…I want that so I dropped this approach

CascadeVertexer and NTrackVertexer

NTrackVertexer:

- All three tracks are fit to common vertex (our Billior vertexer does this now)
- Unconstrained fits use exactly same logic as Billior (ain't broke….)
  - Checked against N=2 Billior and give identical results
- Constrained fits use KKT block-elimination + Newton-Ralphson to solve for vertex & fitted track parameters.
  - Non-linear fit, assumes b-field at target

CascadeVertexer:

- Fit V0+recoil, allowing V0 to fly; fits two vertices V1 (the V0) and V2 (V1+recoil) [sorry for naming]
- The V0 taken from V0 list (e^+e^- / top-bottom)
- KKT/Newton-Raphson block-elimination used for all (and un) constrained fits

I've implemented this on a branch (for now) "`displaced3prongvertex`"

Beam Spot and Beam Momentum Constraints

Beam Spot Constraint:

- Primary vertex (V2 for cascade) must be at the target-z (+/-20um) with (x,y) given by beam conditions db and (sx,sy) = (55um,45um).

Beam Momentum Constraint:

- The vector sum of the 3-momentum must be the beam momentum –> (0,0,3.74GeV) rotated to detector frame. Include covariance based on mean nuclear recoil

Note, both of these constraints are soft….include covariance based on our knowledge of the beam and the physics.

From a math-standpoint, these are basically treated as measurements (though not included in the state variable)

Data samples

so far only run on 2021 MC samples:

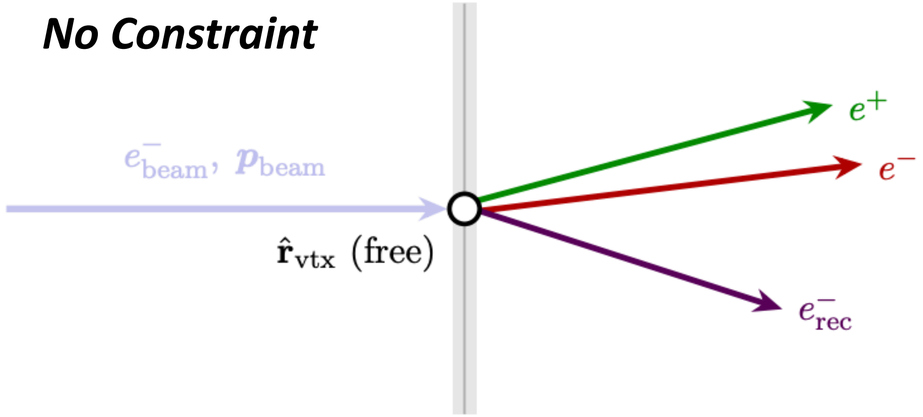
- 100 MeV A′, uniformly displaced, no pulser overlay
  - `/sdf/data/hps/physics2021/mc/readout/ap_nooverlay/pass5_v9/ap100MeV/HPS_Run2021Pass1_v9_14596/`
- 180 MeV A′ , uniformly displaced, with pulser overlay
- tridents with overlay
  - `/sdf/data/hps/physics2021/mc/readout/tritrig_pulser/pass5_v9/HPS_Run2021Pass1_v9_14272`

Mostly looked at truth matched pairs/triplets

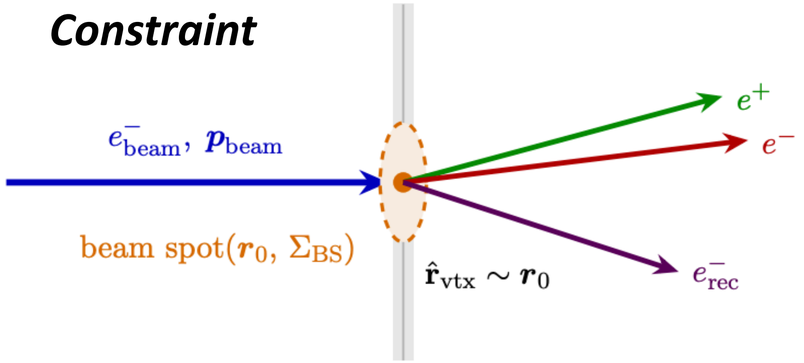
- for A′ sample, the V0 e+e- must be A′ daughters, recoil from nucleus
- for tridents, all three tracks from nucleus

NTrack Constraint Diagrams

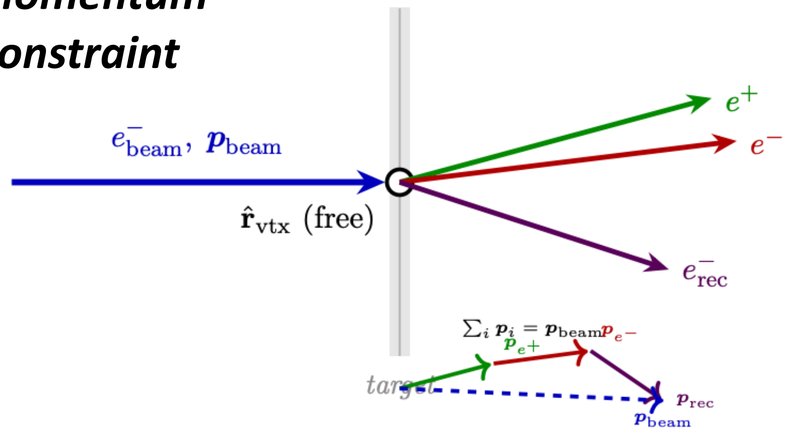
No Constraint



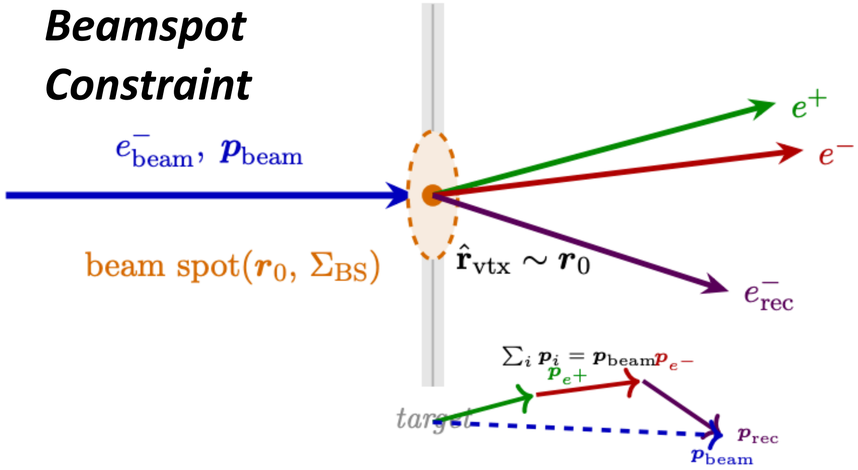
Beamspot Constraint



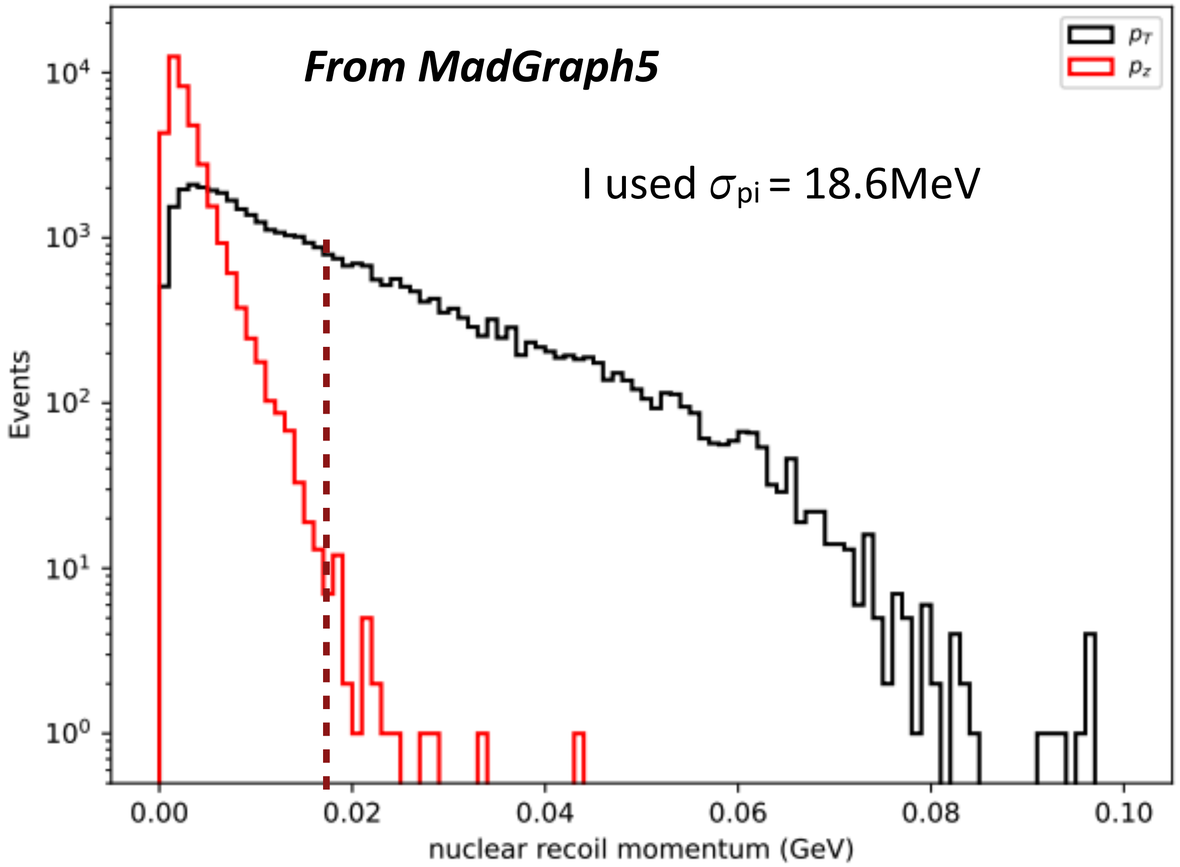
Momentum Constraint



Momentum + Beamspot Constraint

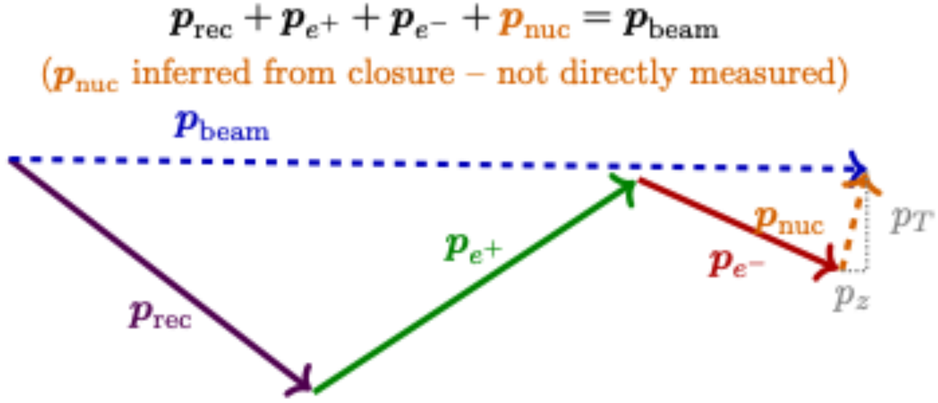


Beam-momentum constraint



For 3-prong, the momentum vector sum should be almost beam momentum…but there is some nuclear recoil.

Treat this as a covariance in the beam-momentum constraint

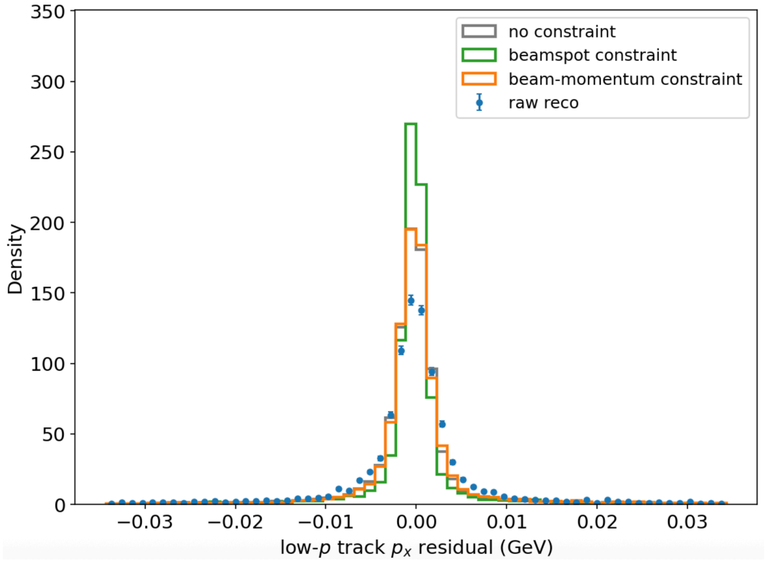


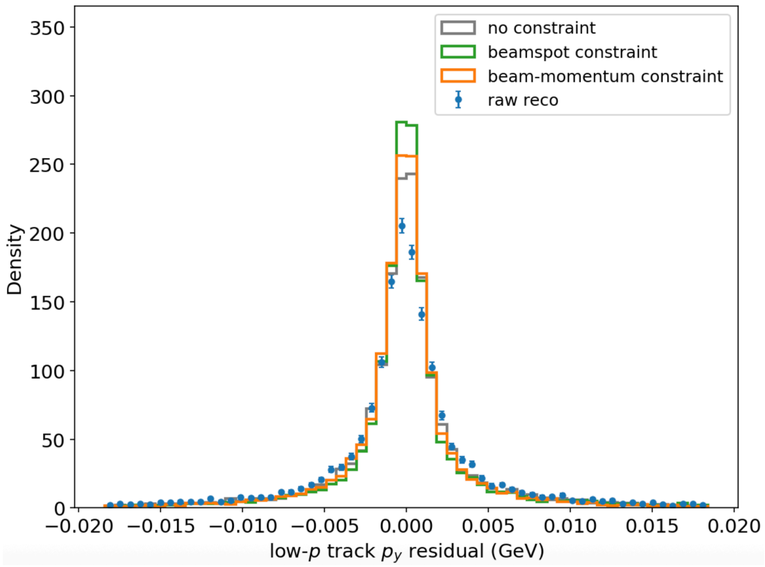
NTrack Vertex Position: Tridents (Matched)

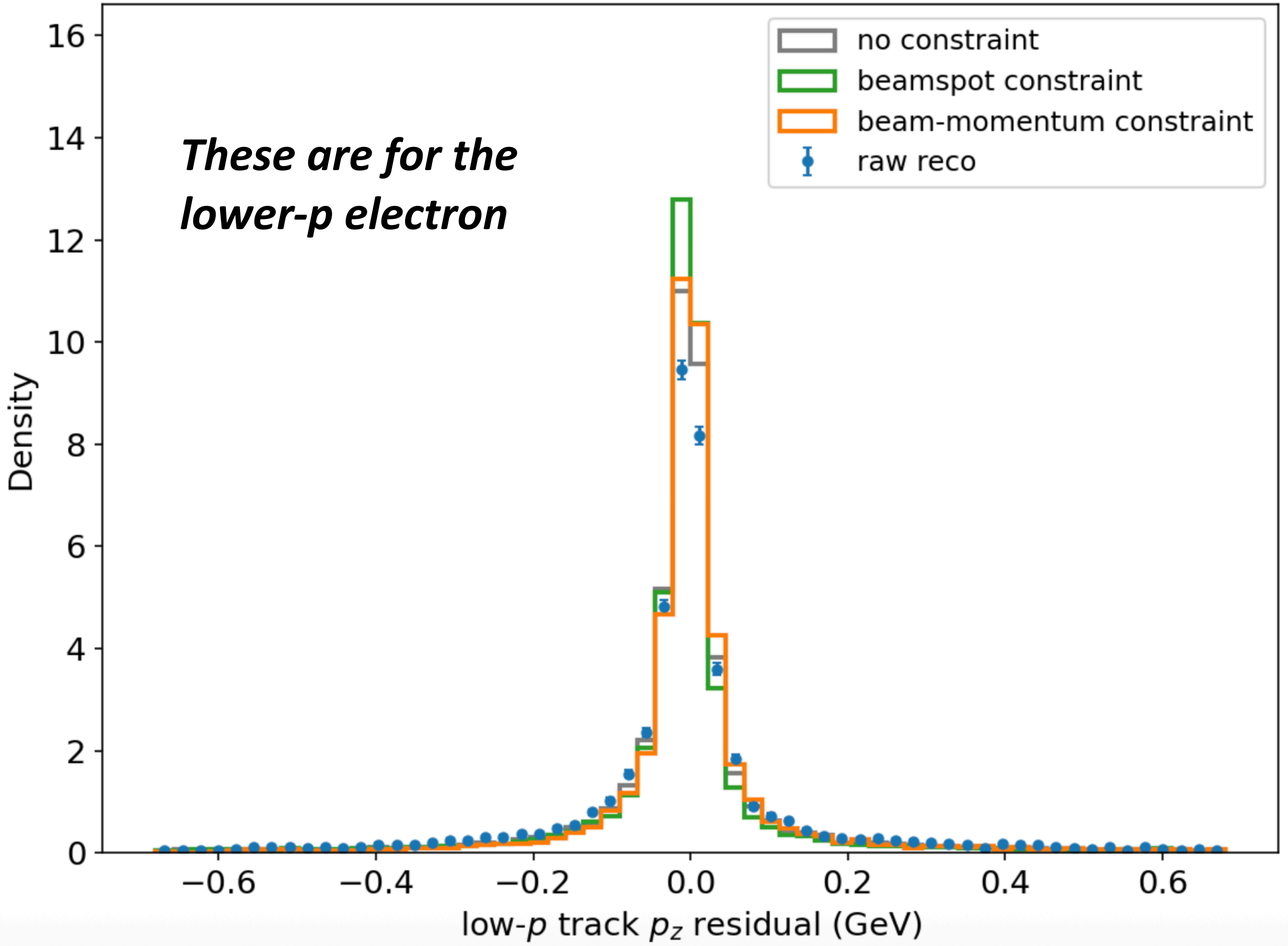
….momentum constraint doesn’t help vertex resolution….

….beamspot constraint makes the resolution the beamspot you put in…

NTrack Individual Track Momentum: Tridents (Matched)

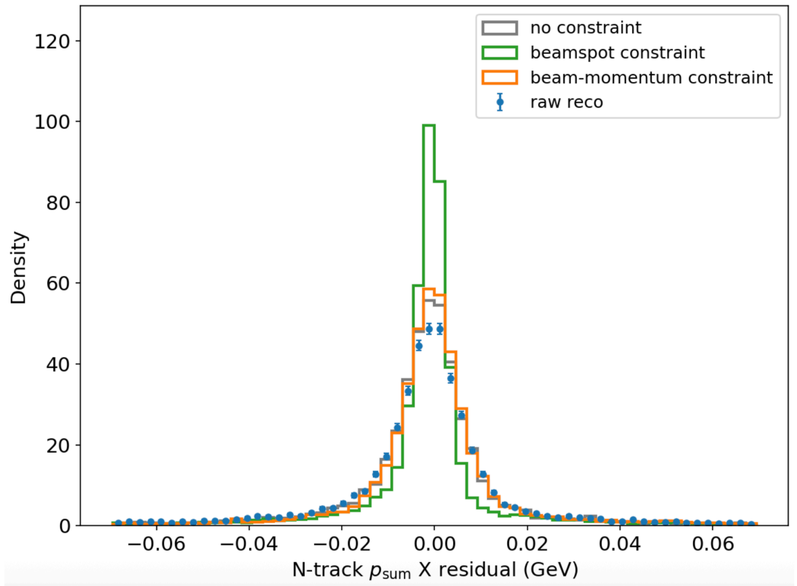


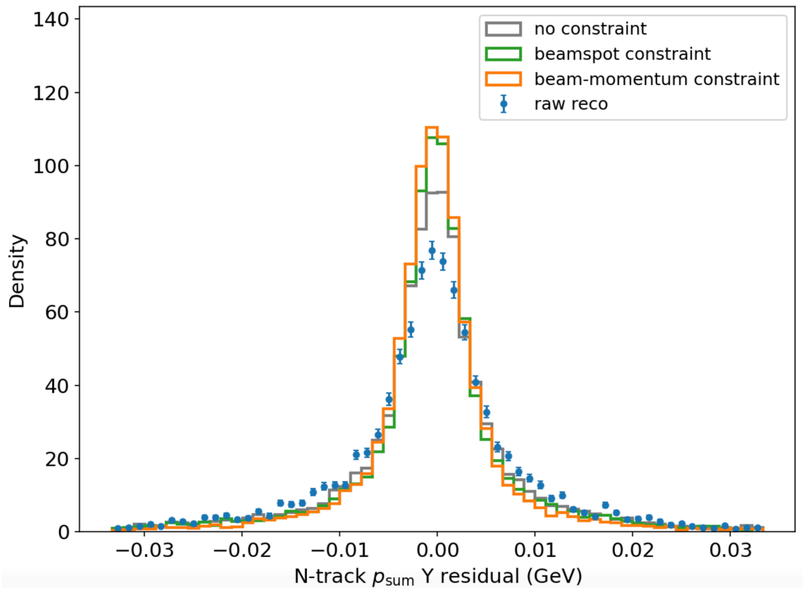


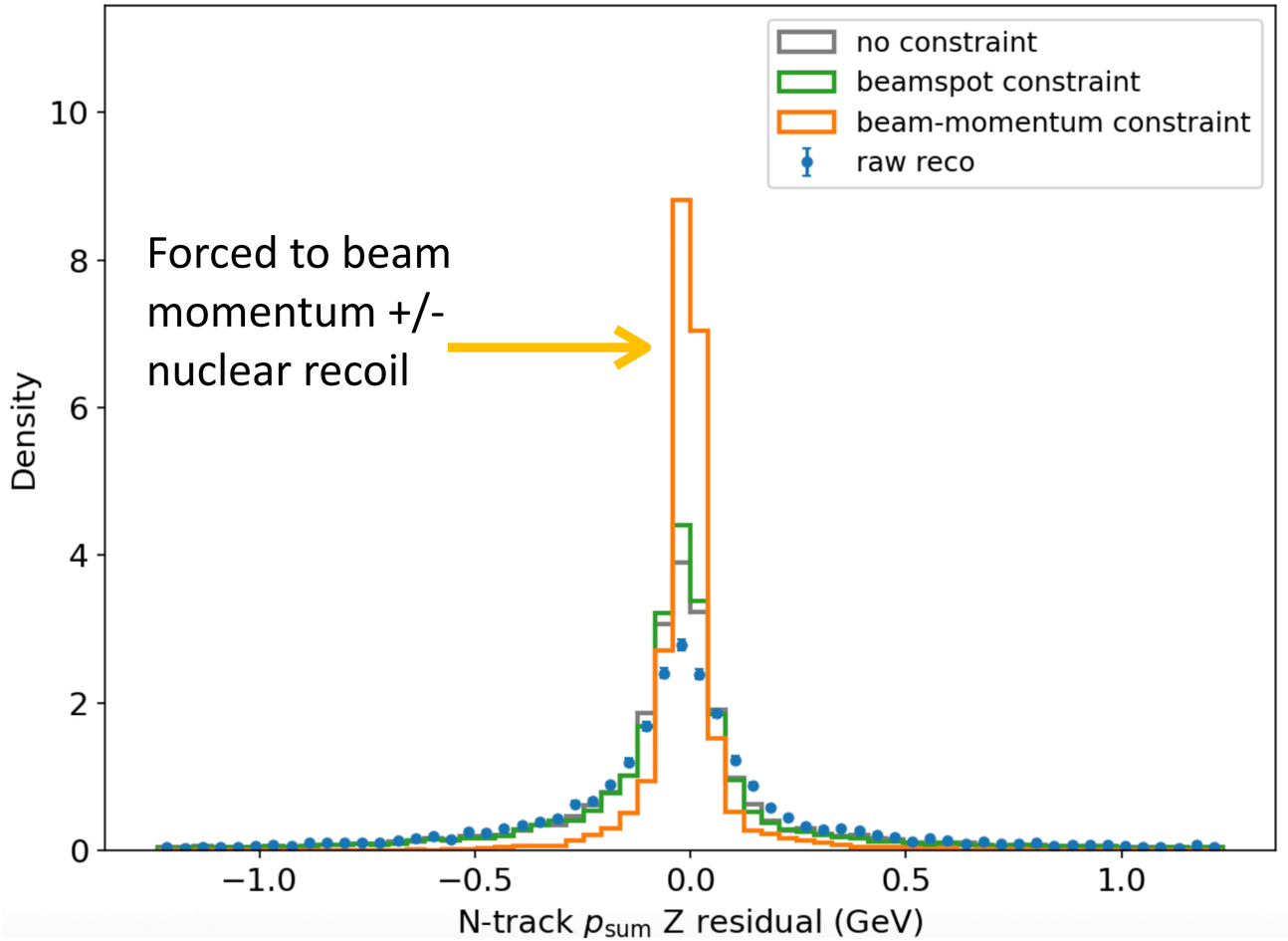


Beam spot & momentum constraints perk up the momentum residual nicely

NTrack Momentum Sum: Tridents (Matched)

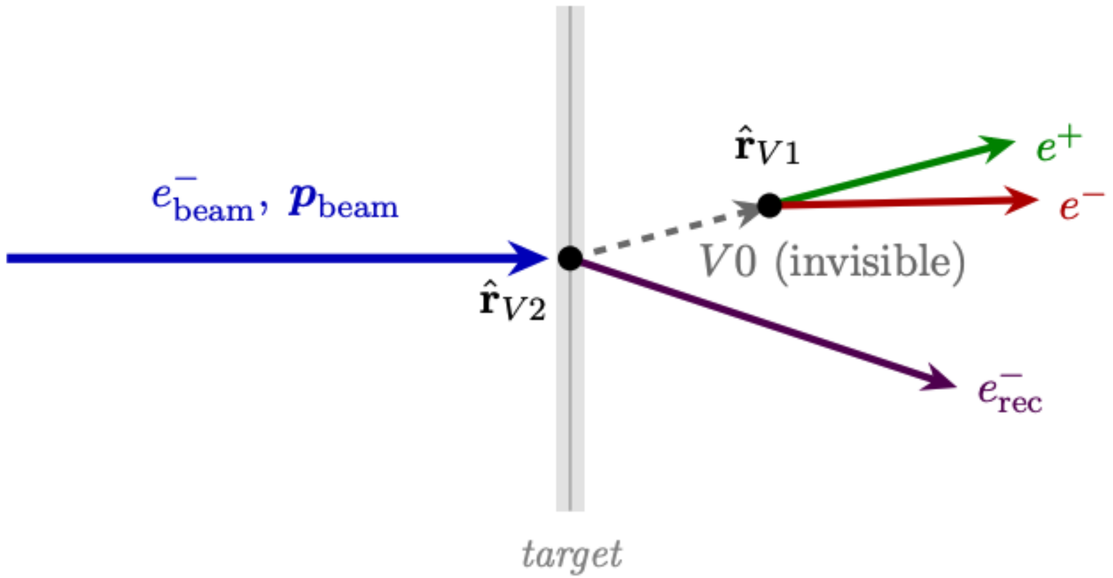






Beam spot constraints perks up the already perked psum…momentum constraint cheats

Cascade Constraint



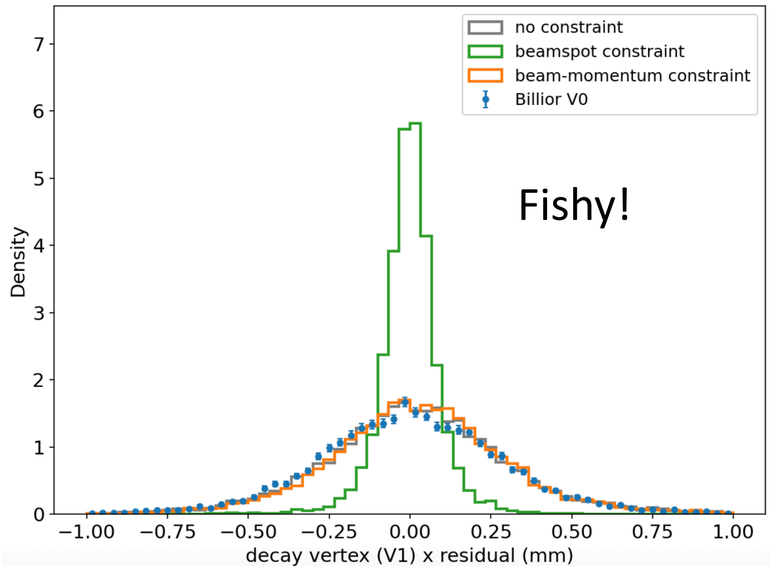
this is a 2-step constraint:

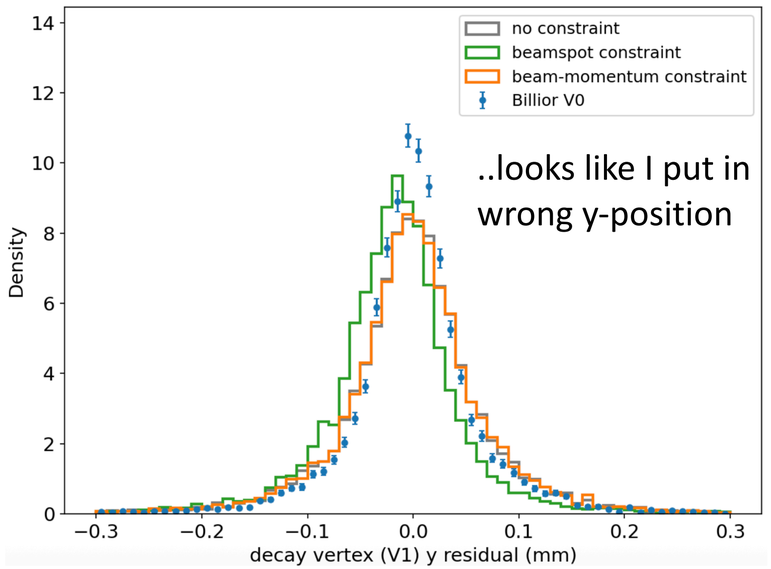
– V1: e+e- vertex (top/bottom)

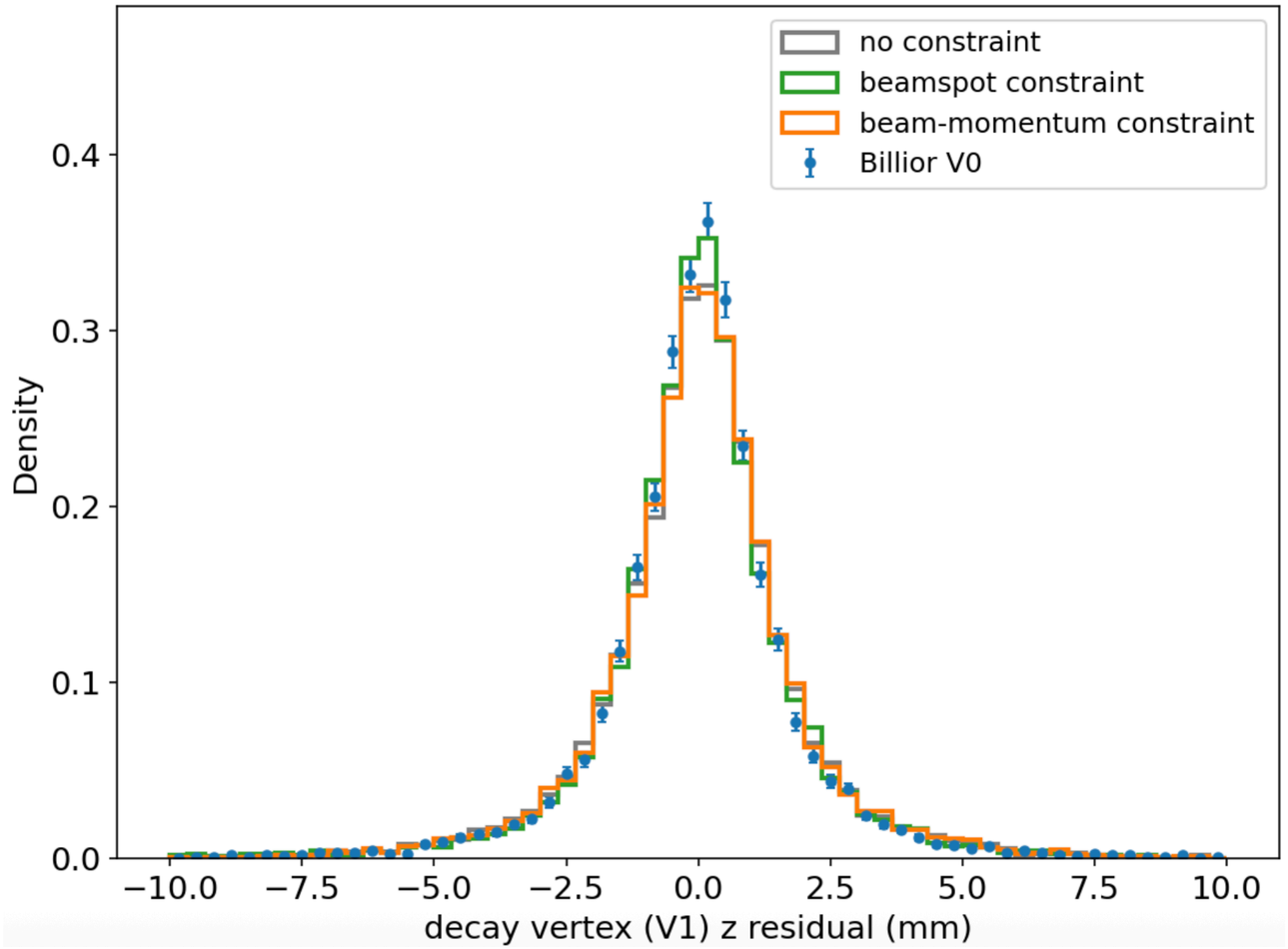
– V2: V0-recoil vertex

Constraints are similar to NTrack

Displaced A' 100 MeV: Cascade V1 (ie V0) vertex residuals

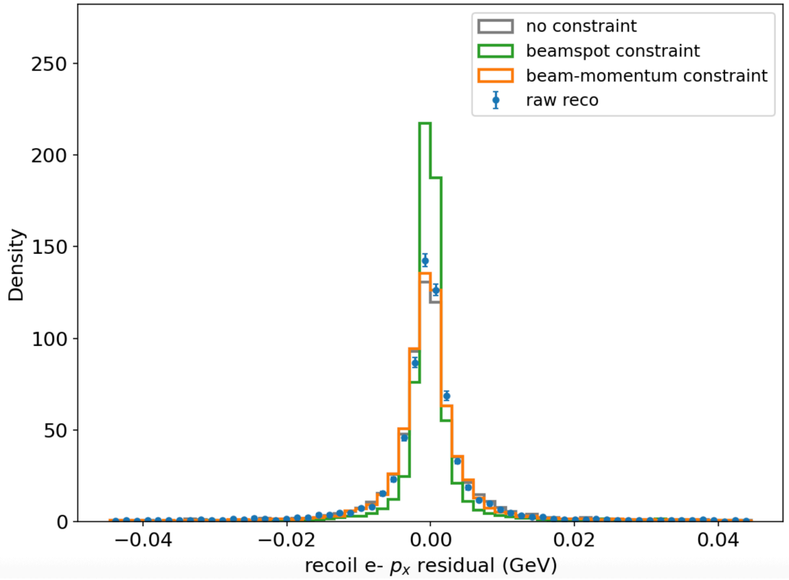


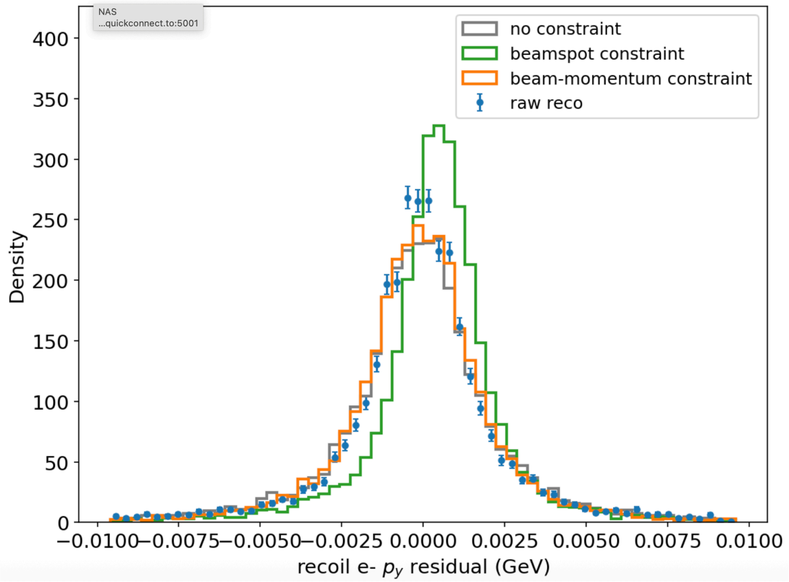


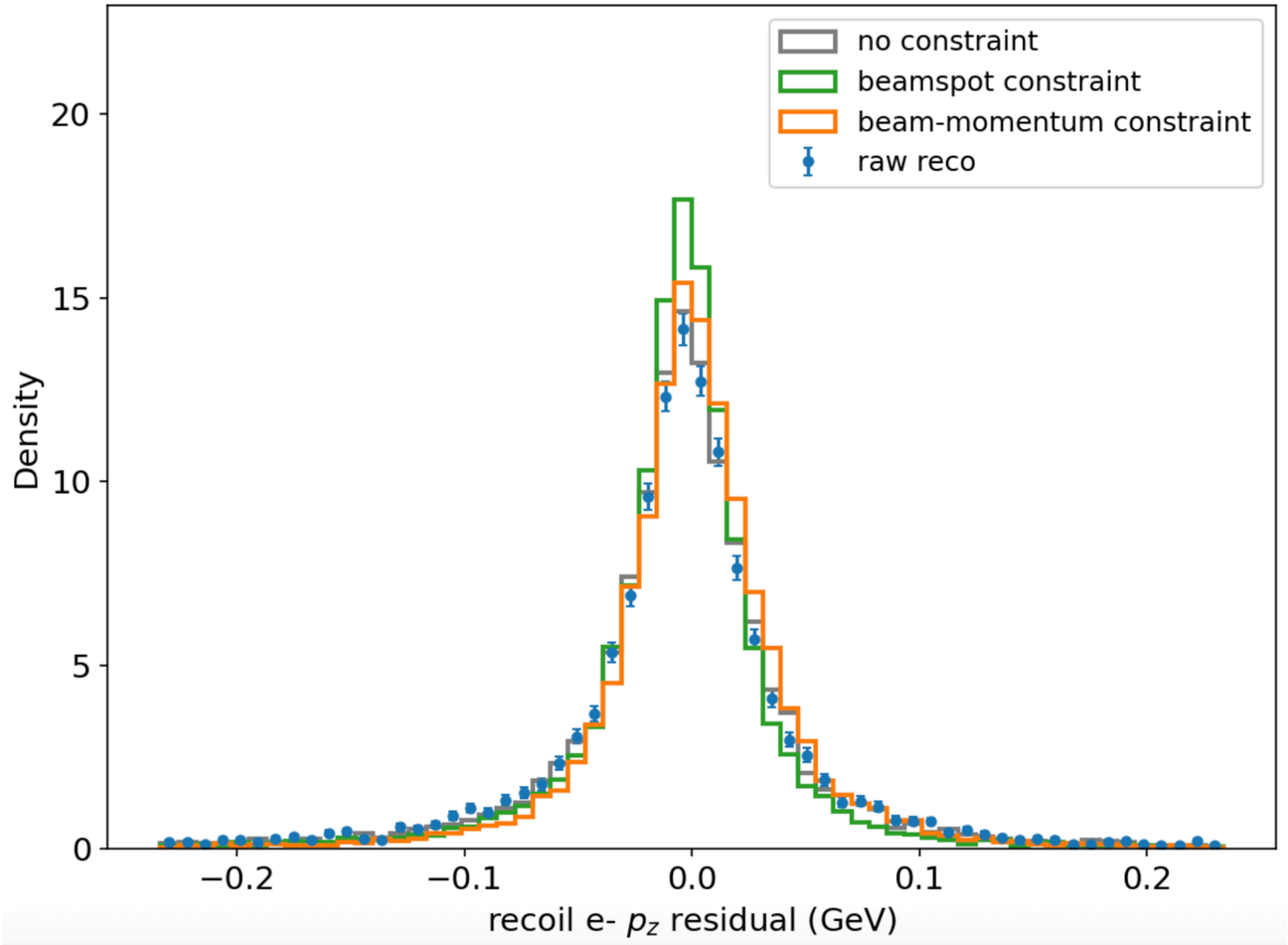


About as good as the plain 2-body Billior fit.

Displaced A’ 100 MeV: Cascade recoil momentum residuals

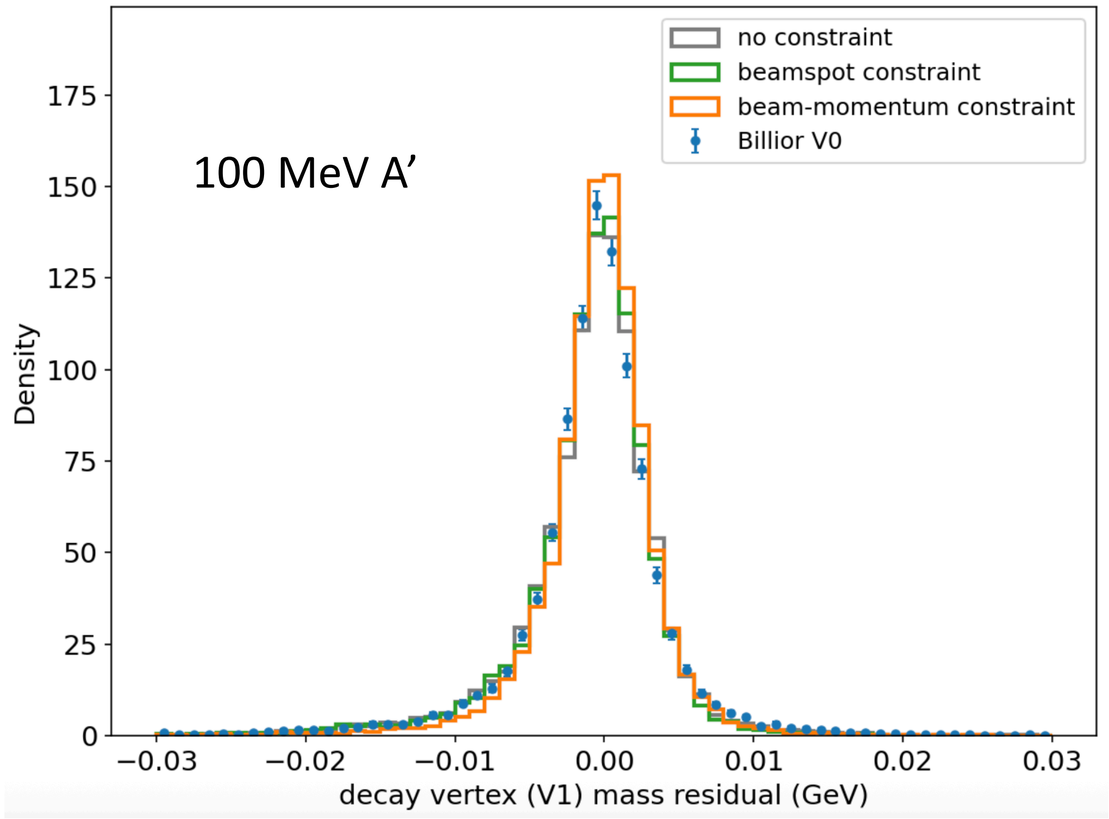


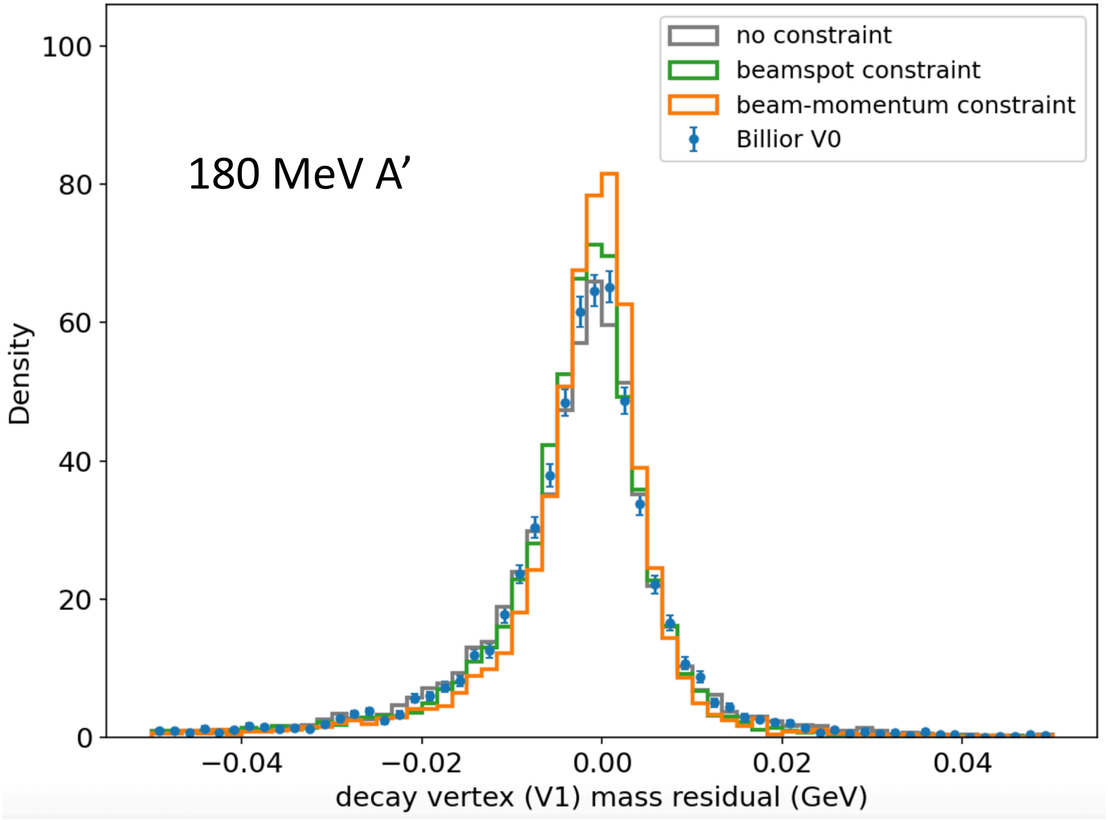




Beamspot definitely helps recoil directions, and a bit for pz

Cascade V1 invariant mass





The mass also tightens up a tiny bit with the beam-momentum constraint…more improvements with >> mass?

Summary (for now)

The cascade and N-track vertexing seem to be working ok

- Right now I have it set up to make collections for all 4 constraints for both NTrack and Cascade
- Should we get this into hps-java? Useful uses?
- Other configurations? W_D^+ W_D^- ?