Cluster*** Killing using Kalman Tracks (2016)

Matt Graham

Tuesday January 16, 2024

*** I might say or label "strip-hit killing" but what I mean (and do) is remove SVT clusters from event

Intro

- I’m doing this for 2016 because:
  - Alic/Tom need this for SIMP analysis
  - we have comparison with GBL slope-based killing for 2016 <link to note>
- there are a few steps to get here:
  - obtain svt cluster-on-track efficiency of each sensor for data and MC (tritrig-beam) vs channel number (actually position→channel number)
  - calculate data/MC ratio-of-efficiency vs channel number
  - in recon of MC, use tables of ratios to remove clusters from the cluster collection prior to KF track finding
  - test if this is working by a) check that killed MC HOT efficiencies agree with data and b) check agreement of selected trident rates & shapes between data and killed MC
- For this talk I’ve only done killing on the first module (sensors 1 & 2) but code is set up to any combination of sensors.
  - The track-slope cluster killing used previously only removed clusters in module 1, so that’s where comparison with GBL-based work is valid
  - This is might be all we need

Setup for Cluster-on-Track Efficiency

- Data used: 10% of run 7800 (2016) from evio
- MC (not killed): on sdf, from Alic Spellman's are: /sdf/group/hps/users/alspellm/projects/THESIS/mc/2016/tritrig_beam/hps-java_v5pt2pt1/pass4/recon_20231006/lcio/
    - I think this is just the 2016 production MC but reconstructed with KF by Alic
    - I re-reconstruct these lcio
- MC (killed): /scratch/mgraham/hitKilling/StripBasedTrackHitKillData/kalman-withStripBasedKilling
    - this will go away (if it hasn't already)
- Steering files:
    - efficiency: analysis/PhysicsRun2016SVTHitEffKalman(MC).lcsim
    - cluster killing: recon/PhysicsRun2016FullReconMC_KF_TrackClusterMatcher_StripHitKiller.lcsim
- Other than a +/- 10ns cut on the track time, no requirements are made on tracks included
    - I make plots for each nHits-on-track** separately (and all combined)
    - also have plots for [electrons, positrons, either]
- hps-java branch: iss476 – this includes both efficiency and killing code
- JIRA issues HPSANA-19, HPSANA-20, HPSANA-21

Changes to KF code for efficiency/killing

- The Kalman filter allows you to easily calculate the unbiased track position at a sensor…in other works the track position as if the hit from the sensor is not included in the track fit.
    - this is how we get the unbiased residuals from KF tracks
    - of course it can also give you the track position if the was NOT in fact a hit from that sensor included in the track
    - because of this, we don't have to run tracking multiple times with different sensors turned off and then compare collections with/without that sensor…just run tracking once.
- Saving the unbiased track positions at every sensor layer required some changes to KalPatRecDriver and KalmanTrack (with Robert's help)

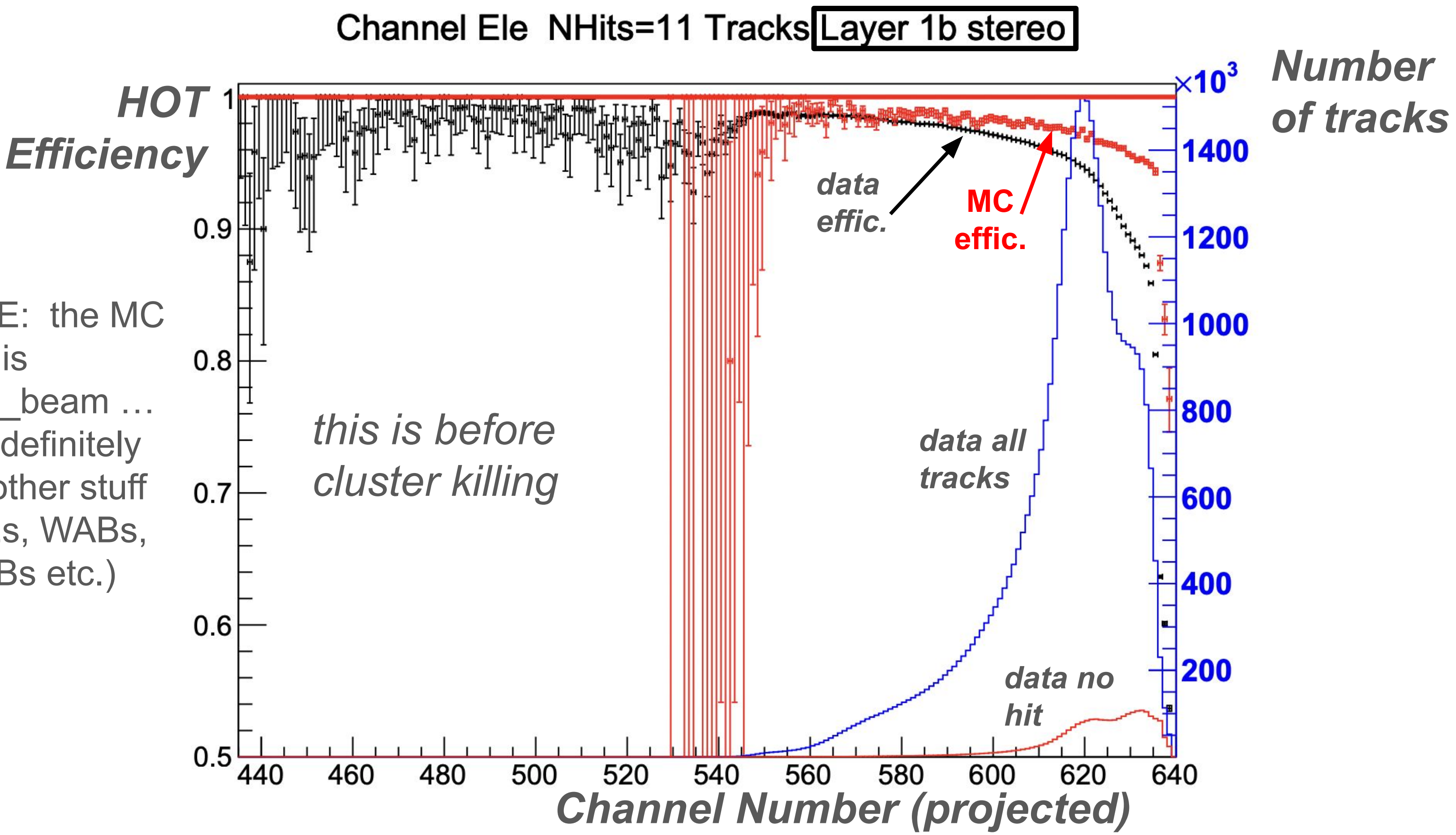
Changes to KalmanPatRecDriver:

- add collection trackIntersects collection and relations to track to event
- add appropriate stuff to fill intersects in "prepareTrackCollections" method

Changes to KalTrack:

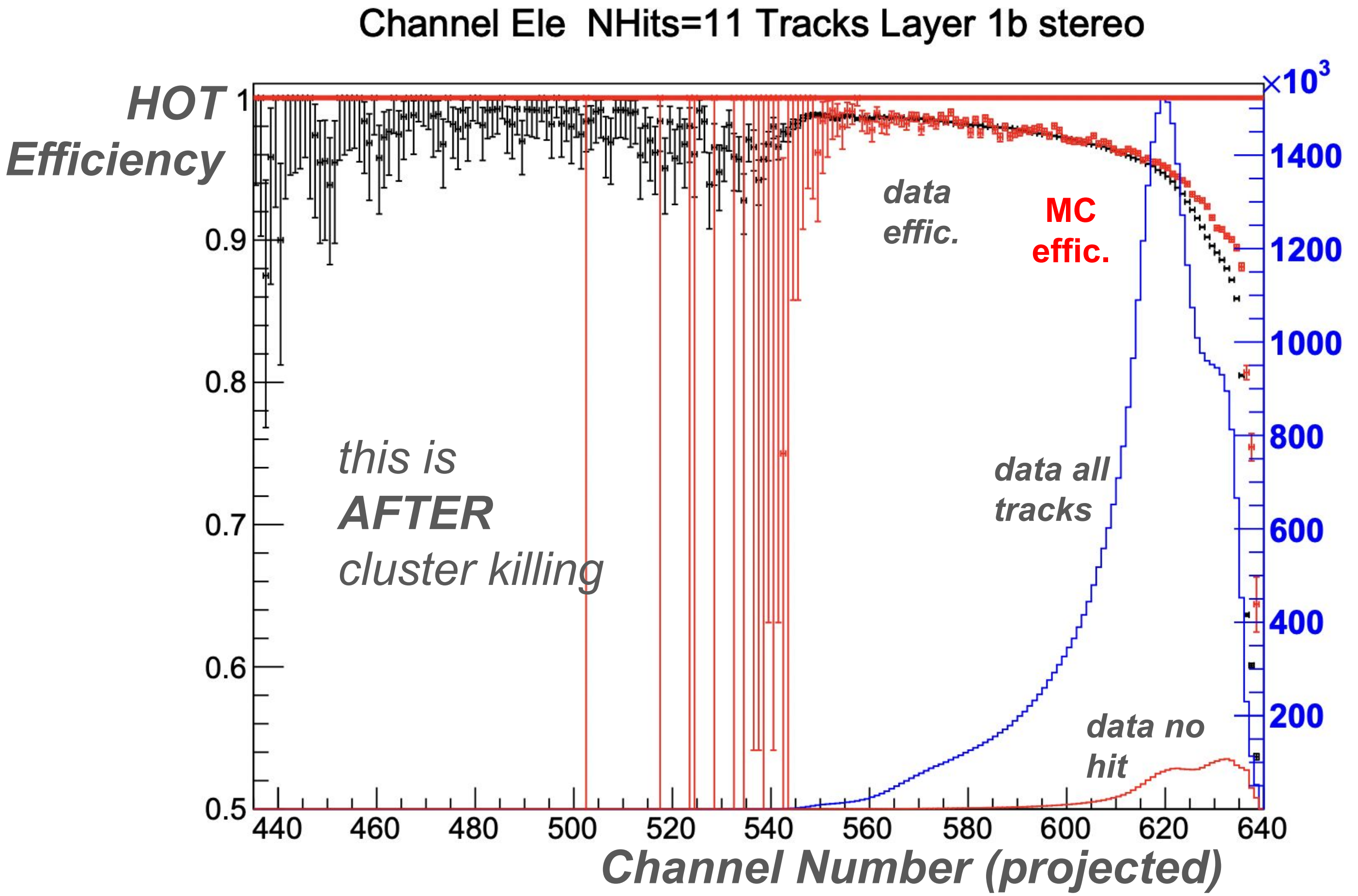
- remove the "trimSites" option so that it looks for intersects beyond the bounds of the first/last layers hit
- add method "unbiasedIntersect" which is similar to "unbiasedResidual" but returns the track intersect with the sensor in sensor frame and variance in u.

Sample of efficiency plot: 11(/12) hit electrons



NOTE: the MC here is tritrig_beam … data definitely has other stuff (FEEs, WABs, cWABs etc.)

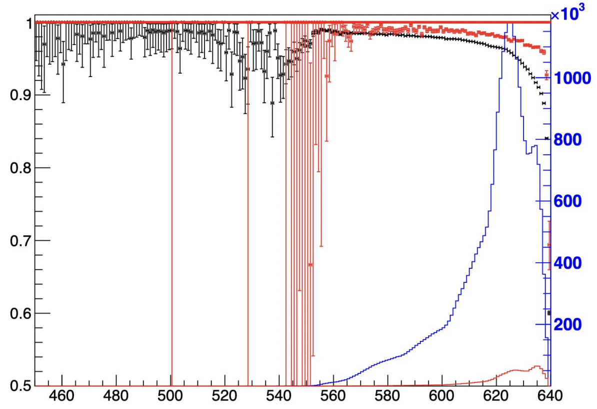
Same plot, but for cluster-killed recon



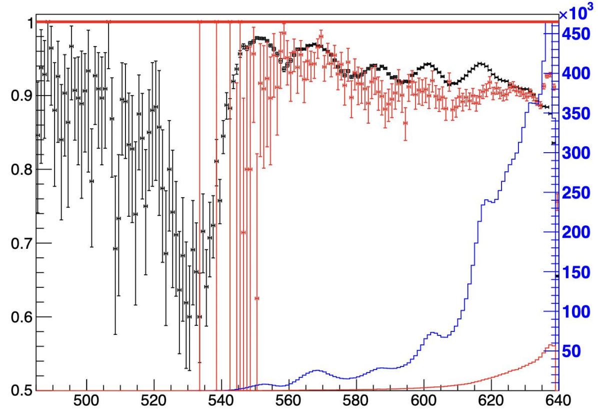
Number of tracks

doesn't quite match yet at very edges… but definitely improved

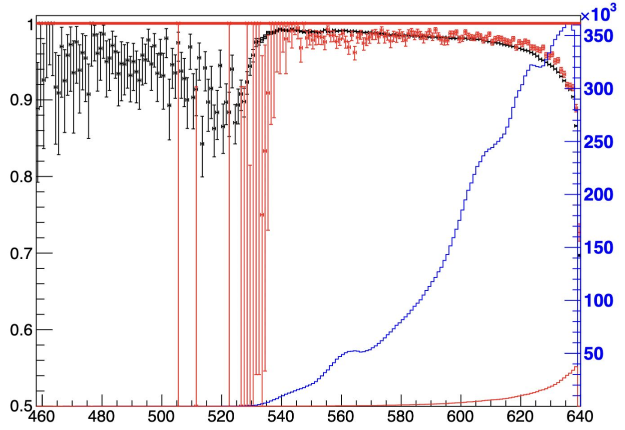
Channel Ele NHits=11 Tracks Layer 2t stereo



Channel Ele NHits=10 Tracks Layer 2t stereo



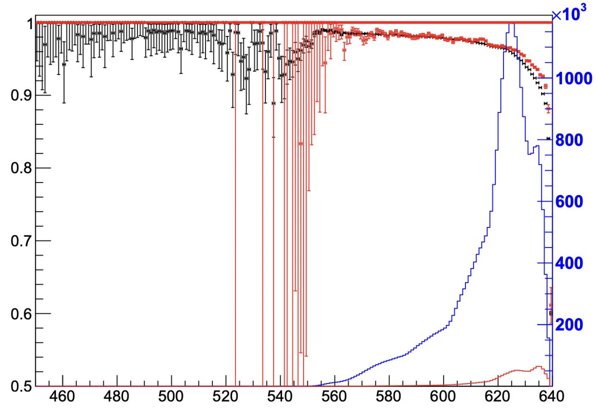
Channel Ele NHits=9 Tracks Layer 2t stereo



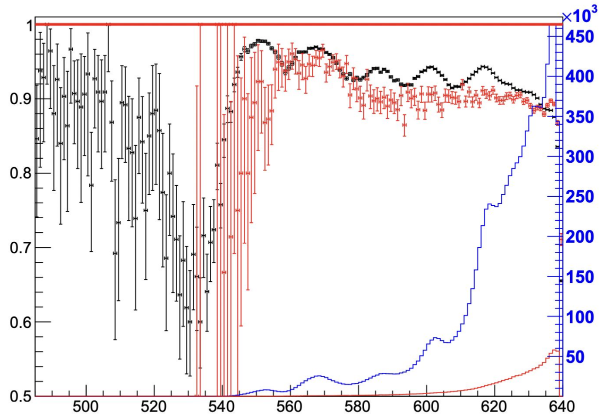
No Killing

Cluster Killed

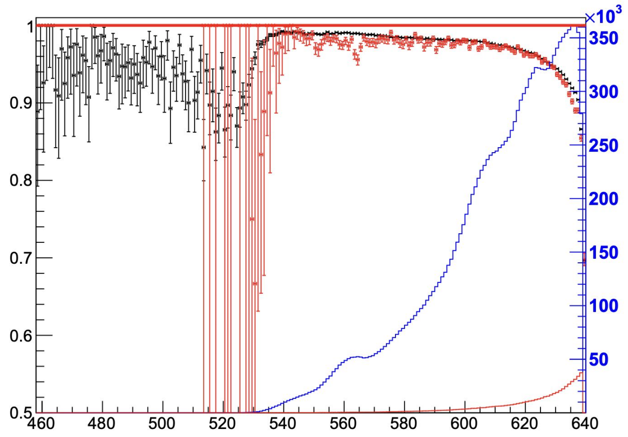
Channel Ele NHits=11 Tracks Layer 2t stereo



Channel Ele NHits=10 Tracks Layer 2t stereo



Channel Ele NHits=9 Tracks Layer 2t stereo



SVT Cluster Efficiency/Killing Summary

- I think the hps-java code to make plots needed for HOT efficiency is good to go…I have a python script which takes it from there
- The hps-java that reads in specified data/mc efficiency ratios for sensors and removes clusters based on these also is working as expected
- See HPSANA-ISS20 for details and killed & unkilled MC and data comparisons for all sensors and nHits
- Looking through these, tracks with “nHits=”-even-numbered (e.g. 10, on previous slide) look strange.
  - I think this track-subsample is mostly made up of tracks that have some number of “module hits” (both axial & stereo) and we ask if it has “just one other hit” but not it’s module-partner hit (otherwise it would be in another nHits plot)
  - I’m going to chalk up most of the disagreement between data/MC for the even numbered nHits plots to the differences in physics between tritrig_beam and data
- For the efficiency ratios used in cluster-hit killing, I chose ***nHits=11, electrons***!
  - positrons from cWABs give fake inefficiencies in first few layers;
- Next…let’s see how data/MC comparisons look for tridents with these changes.

Tridents without and with “L1” hit killing

- I only remove hits from sensor layers 1 & 2, which I’ll call module-1 = “L1”
- Take a look at HPSANA-22 for more details and lots of plots:

My working directory for this is:
/sdf/group/hps/users/mgraham/Tridents20XX/physrun2016

using data:
/sdf/group/hps/data/physrun2016/BLPass4b/rereco/**7800**.root
nominal MC:
/scratch/mgraham/hitKilling/StripBasedTrackHitKillData/kalman-noKilling
killed MC:
/scratch/mgraham/hitKilling/StripBasedTrackHitKillData/kalman-withStripBasedKilling
...see issue 21 for how these files in "/scratch" were made and remembery they will not hang around forever

I used hpstr to do this tuplization and analysis:
hpstr_base = /sdf/home/m/mgraham/sdf/hpstr-iss89

tuplization config: recoTuple_trkEff_cfg.py
analysis config: anaTridentKF_2016_cfg.py

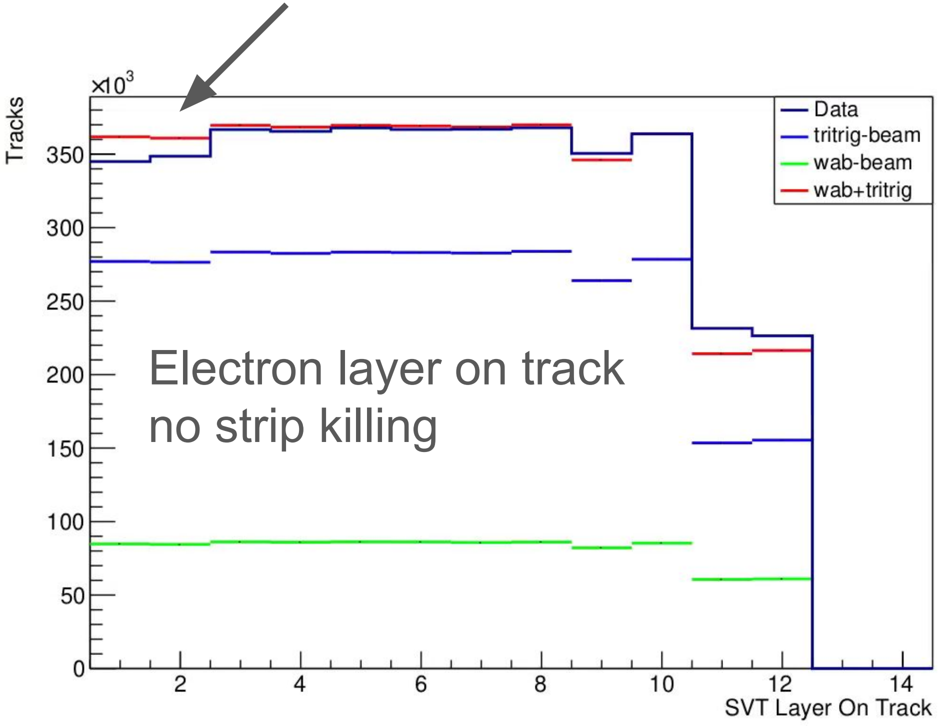
I use the script: plotTridentRates.py
for make plots & getting numbers

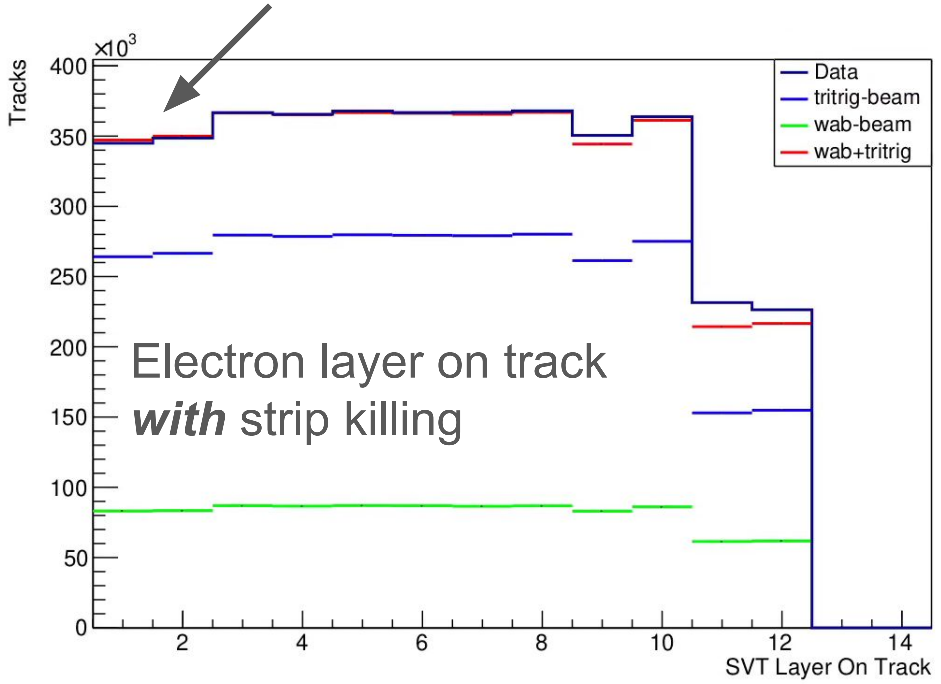
Trident selection & categorization (beyond hps-java)

- In keeping with 2016 GBL-based analysis:
    - require both electron & positron tracks match with ECal clusters
    - require both electron & positron tracks have both axial & stereo hits in L2
- Additionally:
    - nHits(track)>= 10
    - chiSq(track) < 50
    - p(track) > 250 MeV
    - p(electron) < 1.8 GeV
    - 1.0 <p(ele+pos) < 3.0 GeV
- Along with plots for all layers-hit combos, I make separately for all combinations of sensors in L1 hit-or-miss for either electron or positron
- Integrated over layer combos, I see a ~13% excess rate of MC over data…this is in line with what I had for the GBL analysis (~10%).   ***Everything I show comparing data & MC tridents have the MC scaled by 0.87.***

Tridents without & with “L1” hit killing: Electron Layers Hit

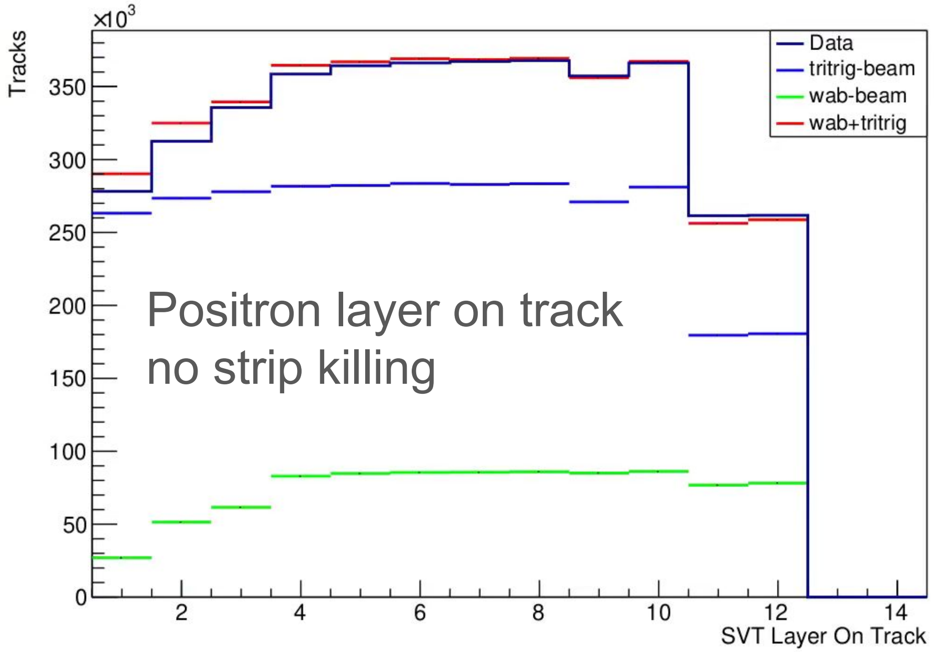
Recall, I only remove hits from sensor layers 1 & 2, which I’ll call module 1 = “L1”



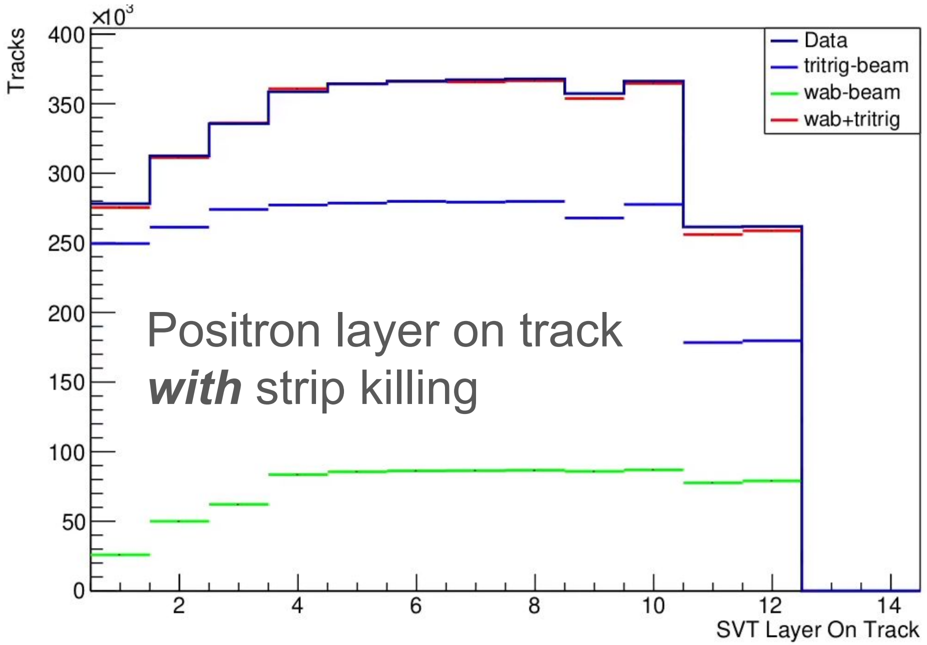


Tridents without & with “L1” hit killing: Positron Layers Hit

…layer-on-track plots look pretty great after killing…



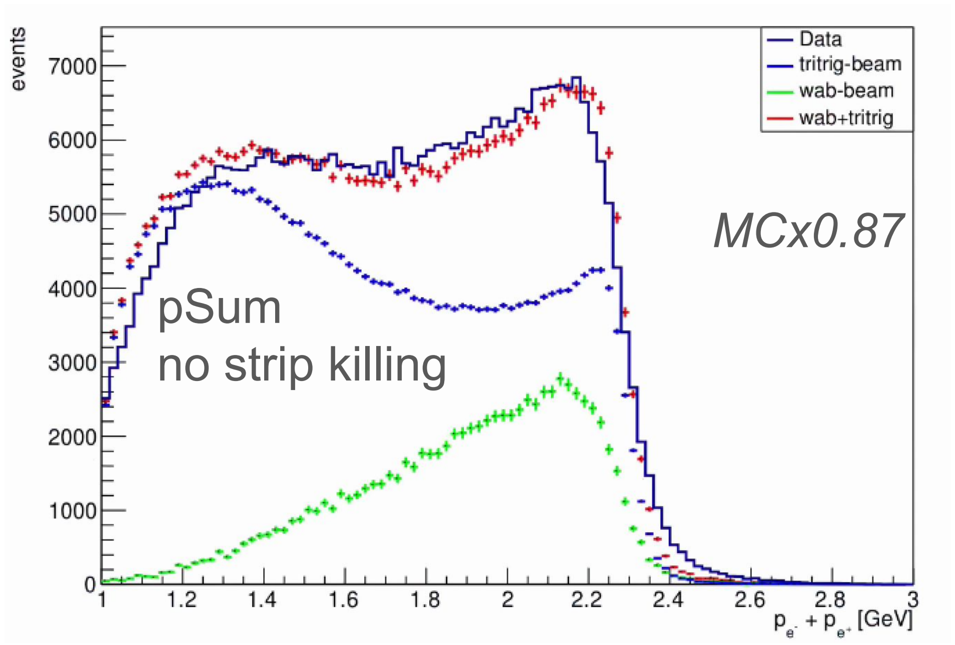
Positron layer on track
no strip killing

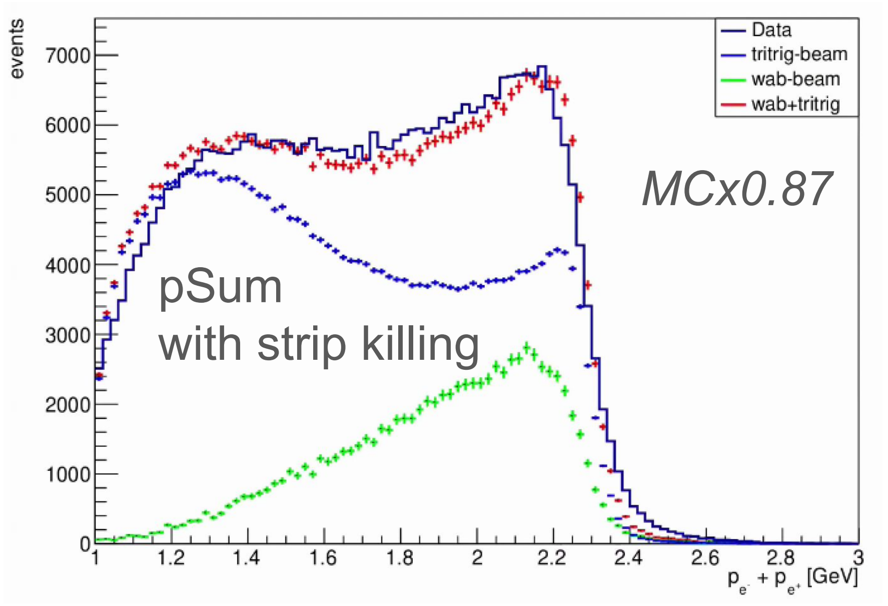


Positron layer on track
with strip killing

pSum for all layer combos…

There's really no change in rates/shapes if you look at all track combinations…this is in line with slope-based killing





…amount of tracks lost will of course increase a bit if we remove hits on all layers, but likely it will be small effect.

Module-Layer Fractions: data vs. trident+WAB MC

Recall…LXLY is first module layer with hit for X=positron, Y=electron

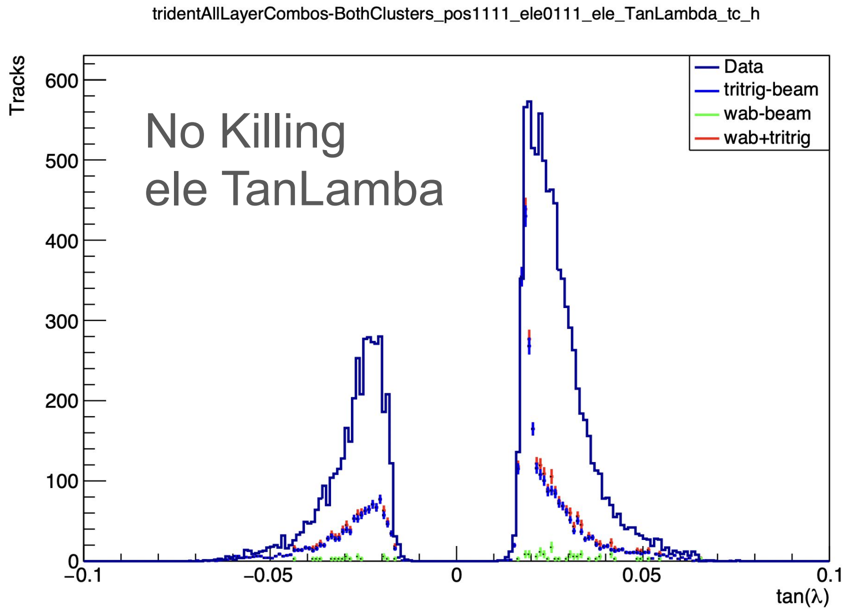
	Total Evts	f(L1L1)	f(L1L2)	f(L2L1)	f(L2L2)
run 7800	319509	0.663	0.081	0.228	0.029
Killed MC	323966	0.687	0.072	0.219	0.022
Nominal MC	329019	0.782	0.035	0.176	0.007

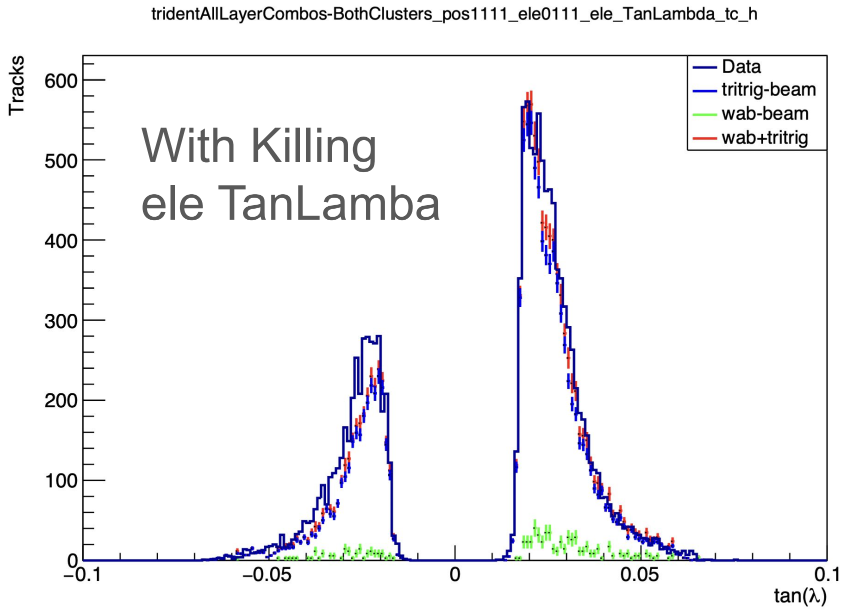
↑ mostly cWABs

This counts up classes of events how they would be classified at the module level…e.g. if a track had only a stereo or axial hit on a lay, that layer would be counted as being missed.
These event-category fractions after killing look good! As good or better than slope-based killing using GBL.

pos1111_ele0111…a mostly inefficiency combo

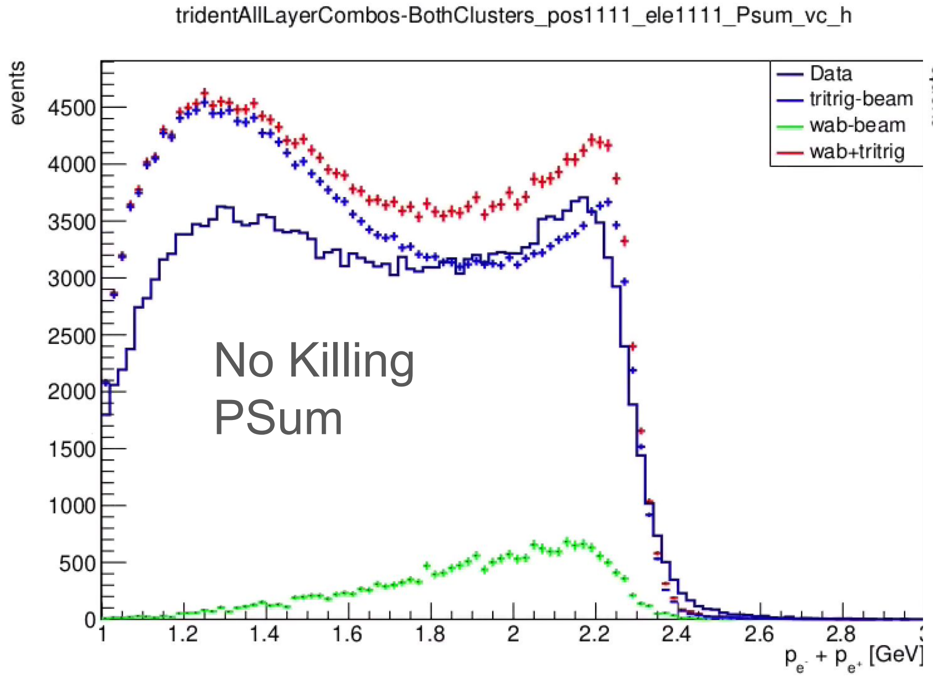
I show this to give an example of what cluster killing gets you…

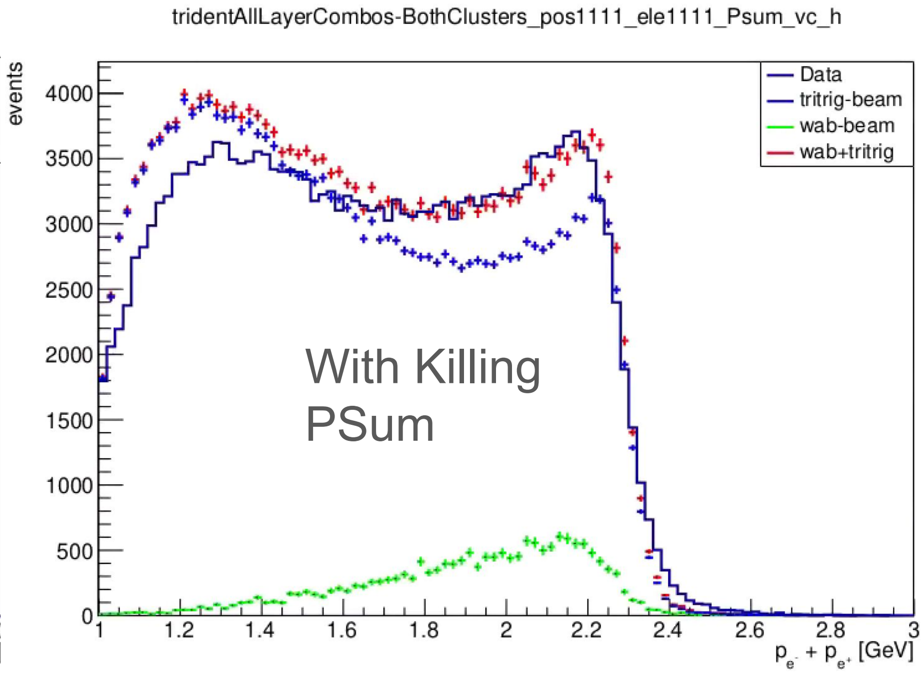




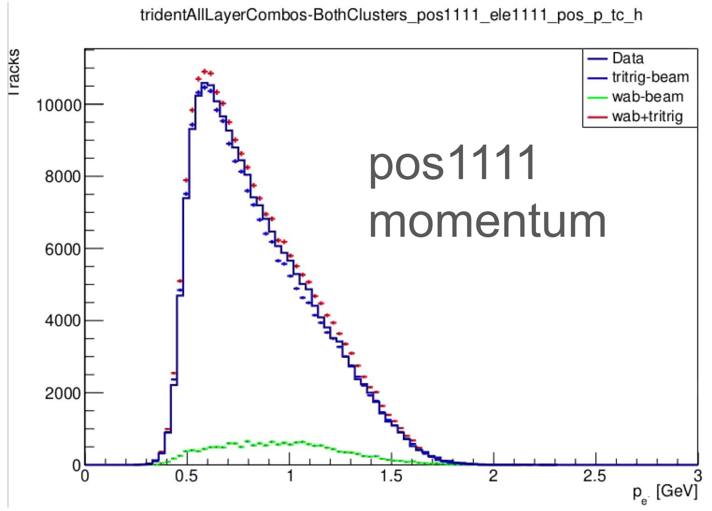
…w/o killing, the MC says this combo is very low rate (which ~ comes from acceptance in L1). Adding killing gets the rate and distribution ~correct

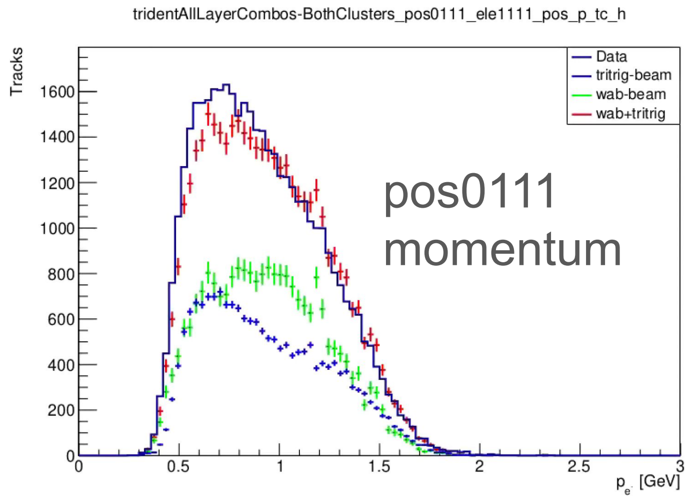
You even need this if requiring L1L1





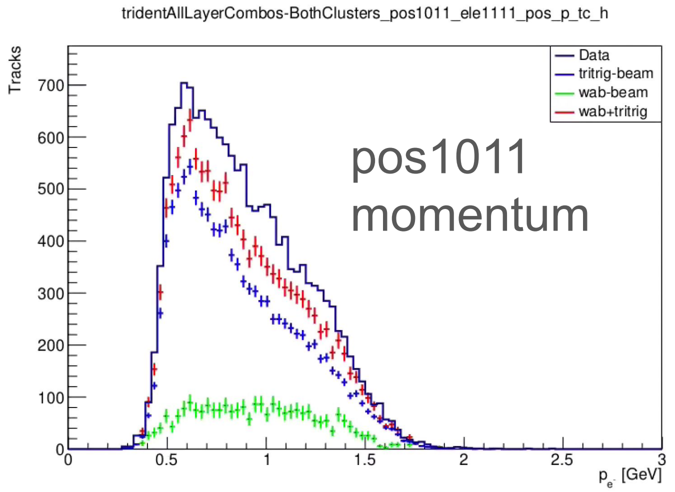
The rate w/o killing is ~15% too high for L1L1 combos…killing ***almost*** fixes it…some issue at low pSum (trigger turn on, maybe? Maybe killing doesn't weigh lowP tracks correctly?)

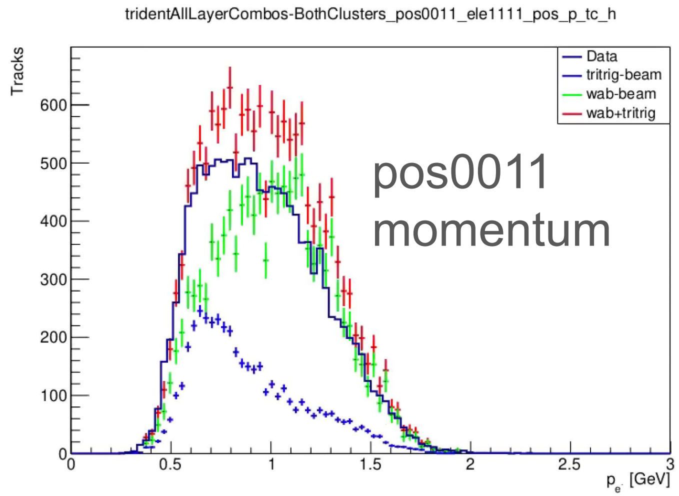




all 4 plots with killing and ele1111

ignore axis title, these are positrons





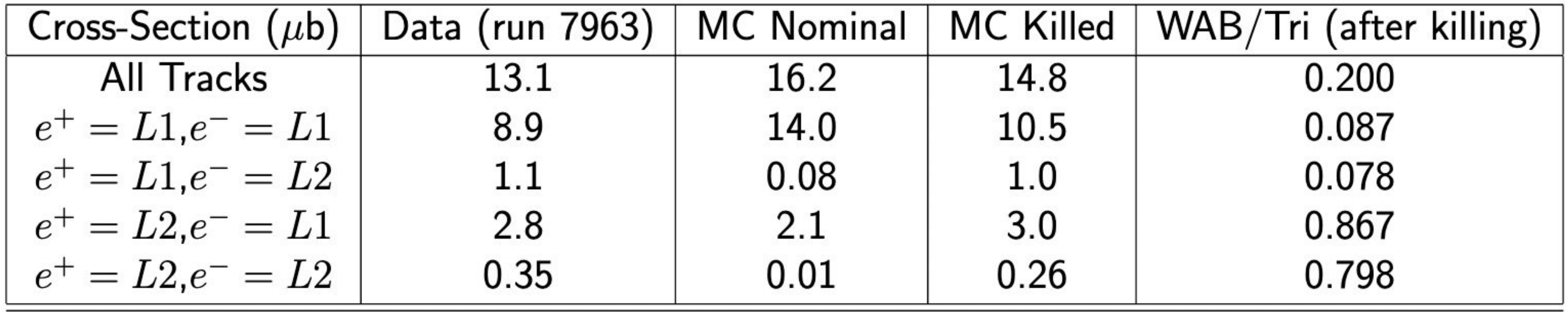
Summary

- The HOT efficiency and cluster killing code (plus required changes to existing code) for KF tracking is written and working. I have a pull request in for this.
- For 2016, sensor inefficiency in the first 2 layers is ~1st order correction…i.e. it’s ~10% for L1L1 efficiency. For L1L2/L2L1/L2L2, it’s ~x2 or higher correction (but in a positive way).
- For the 2016 SIMPs, we should get the MC rereconned ASAP, and that should be the MC used going forward (no more non-killed MC)
- For 2019/2021, once we decide we are ready, we can run this efficiency->constants->testing on that data & nominal MC and re-recon
- In other words…for physics, we should be using cluster-killed MC going forward.

From the 2016 GBL-based note

	Total Rate	f(L1L1)	f(L1L2)	f(L2L1)	f(L2L2)
data	13.1	0.679	0.084	0.214	0.027
Killed MC	14.8	0.709	0.068	0.203	0.018
Nominal MC	16.2	0.864	0.005	0.130	0.001

think these are rates before ~10% scaling



Cross-Section (μb)	Data (run 7963)	MC Nominal	MC Killed	WAB/Tri (after killing)
All Tracks	13.1	16.2	14.8	0.200
$e^+ = L1, e^- = L1$	8.9	14.0	10.5	0.087
$e^+ = L1, e^- = L2$	1.1	0.08	1.0	0.078
$e^+ = L2, e^- = L1$	2.8	2.1	3.0	0.867
$e^+ = L2, e^- = L2$	0.35	0.01	0.26	0.798