ANALYSIS OF THE EMPTY CELL RUNS INCLUDING COMPARISON WITH SIMULATION

$$\gamma p \rightarrow \rho p \rightarrow \pi^{+}\pi^{-}p$$

NATHALY SANTIESTEBAN

MAYO 2021

Data Selection

Part 1

Runs: 30333, 30334, 30336, 30337, 30564, 30728, 40903, 41386, 41615, 51011, 51013, 51556

Reconstruction Plugin

- Location: /w/halld-scifs17exp/halld2/home/nathaly/test/halld_recon/src/plugins/src-test/2pi1p
- Generates a root file with the information of all candidates: *Position, Momentum, CDC, FDC, TOF .. information
- The most relevant characteristics are described in the following couple of slides

1. Hypothesis Selection

- Only 3 Charged Tracks and No Showers events:

```
if (ch_tracks.size()==3) return NOERROR;
if ( showers.size() !=0 ) return NOERROR;
```

- Particle Hypothesis are determined by the GetHypotheses function:

Definition:

```
//Recursive function for determining possible particle assignments
// -- J. Pybus
void DEventProcessor_2pi1p::GetHypotheses(vector<const DChargedTrack *> &tracks,
                                          map<Particle_t, int> &particles,
                                          map<Particle_t, vector<const DChargedTrackHypothesis*> > &assignmentHypothesis,
                                          vector<map<Particle_t, vector<const DChargedTrackHypothesis*> > > &hypothesisList
                                          ) const
{

  const DChargedTrack * firstTrack = tracks.front();
  vector<const DChargedTrack *> otherTracks(tracks);
  otherTracks.erase(otherTracks.begin());

  map<Particle_t, int>::iterator partIt;

  for (partIt = particles.begin(); partIt != particles.end(); partIt++){

     if (partIt->second > 0){
        Particle_t particle = partIt->first;
        const DChargedTrackHypothesis *hyp=NULL;

        if ((hyp = firstTrack->Get_Hypothesis(particle)) != NULL){

                double prob = TMath::Prob(hyp->Get_ChiSq(),hyp->Get_NDF());
                if (prob < 1E-4) continue;
                map<Particle_t, vector<const DChargedTrackHypothesis*> > newHypothesis = assignmentHypothesis;

                if (assignmentHypothesis.find(particle) == assignmentHypothesis.end()){
                        vector<const DChargedTrackHypothesis*> newVector;
                        newHypothesis[particle] = newVector;
                }

                newHypothesis[particle].push_back(hyp);

                if (otherTracks.empty()){
                        hypothesisList.push_back(newHypothesis);
                } else {
                        map<Particle_t, int> otherParticles(particles);
                        otherParticles[particle]--;
                        GetHypotheses(otherTracks, otherParticles, newHypothesis, hypothesisList);
                }
        }
     }
  }
} // end GetHypotheses
```

```
GetHypotheses(ch_tracks,targetParticles,emptyHypothesis,hypothesisList);
```

```
vector<const DChargedTrack*>ch_tracks;
loop->Get(ch_tracks);
```

```
map<Particle_t, int> targetParticles = {
        {Proton,1},
        {PiPlus,1},
        {PiMinus,1}
};
```

Hypothesis accepted if probability > 1E4
based in the timing information.

2. Vertex Fit

- For each Hypothesis the Vertex Fit is performed:

Only includes the charged tracks hypothesis.

```
//---------------------------------
// Kinematic fit
//---------------------------------
DKinFitUtils_GlueX *dKinFitUtils = new DKinFitUtils_GlueX(loop);
DKinFitter *dKinFitter = new DKinFitter(dKinFitUtils);

dKinFitter->Reset_NewFit();

set<shared_ptr<DKinFitParticle>> FinalParticles, NoParticles;

shared_ptr<DKinFitParticle>myProton=dKinFitUtils->Make_DetectedParticle(proton_track);
shared_ptr<DKinFitParticle>myPiMinus=dKinFitUtils->Make_DetectedParticle(pi_min_track);
shared_ptr<DKinFitParticle>myPiPlus=dKinFitUtils->Make_DetectedParticle(pi_plus_track);

FinalParticles.insert(myProton);
FinalParticles.insert(myPiMinus);
FinalParticles.insert(myPiPlus);

//  Production Vertex constraint
set<shared_ptr<DKinFitParticle>> locFullConstrainParticles;
locFullConstrainParticles.insert(myPiPlus);
locFullConstrainParticles.insert(myPiMinus);
locFullConstrainParticles.insert(myProton);

shared_ptr<DKinFitConstraint_Vertex> locProductionVertexConstraint =  dKinFitUtils->Make_VertexConstraint(locFullConstrainParticles, NoParticles, proton_track->position());
dKinFitter->Add_Constraint(locProductionVertexConstraint);
// PERFORM THE KINEMATIC FIT
dKinFitter->Fit_Reaction();
//GET THE FIT RESULTS
double CL = dKinFitter->Get_ConfidenceLevel();
```

Example of PiMiums tracking information (same apply for the other hypothesis):

```
const DChargedTrackHypothesis *hyp_pi_min   = thisHyp[PiMinus][0];
const DTrackTimeBased *pi_min_track = hyp_pi_min->Get_TrackTimeBased();
```

Final Events Selection

Section 2

Runs: 30333, 30334, 30336, 30337, 30564, 30728, 40903, 41386, 41615, 51011, 51013, 51556

Macro can be found in:
/w/halld-scifs17exp/halld2/home/nathaly/test/scripts/read_2pi1p_candidates.C

1. Reconstructed candidates selection

```
TLorentzVector hyp1pim(pX_piminuskinfit[0],pY_piminuskinfit[0],pZ_piminuskinfit[0], E_piminuskinfit[0]);
TLorentzVector hyp1pip(pX_pipluskinfit[0],pY_pipluskinfit[0],pZ_pipluskinfit[0], E_pipluskinfit[0]);
TLorentzVector hyp1p(pX_protonkinfit[0],pY_protonkinfit[0],pZ_protonkinfit[0], E_protonkinfit[0]);
TLorentzVector hyp1rho = hyp1pim + hyp1pip;
```

- Mass of the reconstructed rho: $0.6 < m_{\rho} < 1.$ [GeV]

- Coplanarity between ρ and p: $160 < \Delta\phi(\rho - p) < 200$ deg

- Reconstructed Energies in the range that are expected for $E_{\gamma} > 7GeV$
  $E_{\rho} + E_{p} > 7GeV$

2. Photon information

- $E_\gamma > 7$ GeV (only events above that energy were taken to select the in-time and off-time photons).
- $|dE| < 1$ GeV, where $dE = E_\gamma + m_p - E_p - E_{\pi^+} - E_{\pi^-}$
- Final Candidates:

```
TLorentzVector pPim(pX_pim,pY_pim,pZ_pim,E_pim)
TLorentzVector pPip(pX_pip,pY_pip,pZ_pip,E_pip)
TLorentzVector pP(pX_p,pY_p,pZ_p,E_p)
TLorentzVector Ep(0,0,bmE[j],bmE[j])
TLorentzVector rho = pPim + pPip
 s =  (pPim+pPip+pP)*(pPim+pPip+pP)
 t  =  -(Ep-rho)*(Ep-rho)
 u  = -(Ep-pP)*(Ep-pP);
```

- In-time Photons $\Delta t = t - \left(t_{RF} + \frac{Z_{vtx} - Z_{Center}}{29.9792458} \right) < 2ns$

  t : Time from the vertex fit

  t_{RF} : Time from the beam to the center of the target:

```
vector<const DBeamPhoton*> beam_ph;
loop->Get(beam_ph);
beam_ph[ii]->time();
```

  Z_{vtx} : Z position from the vertex fit

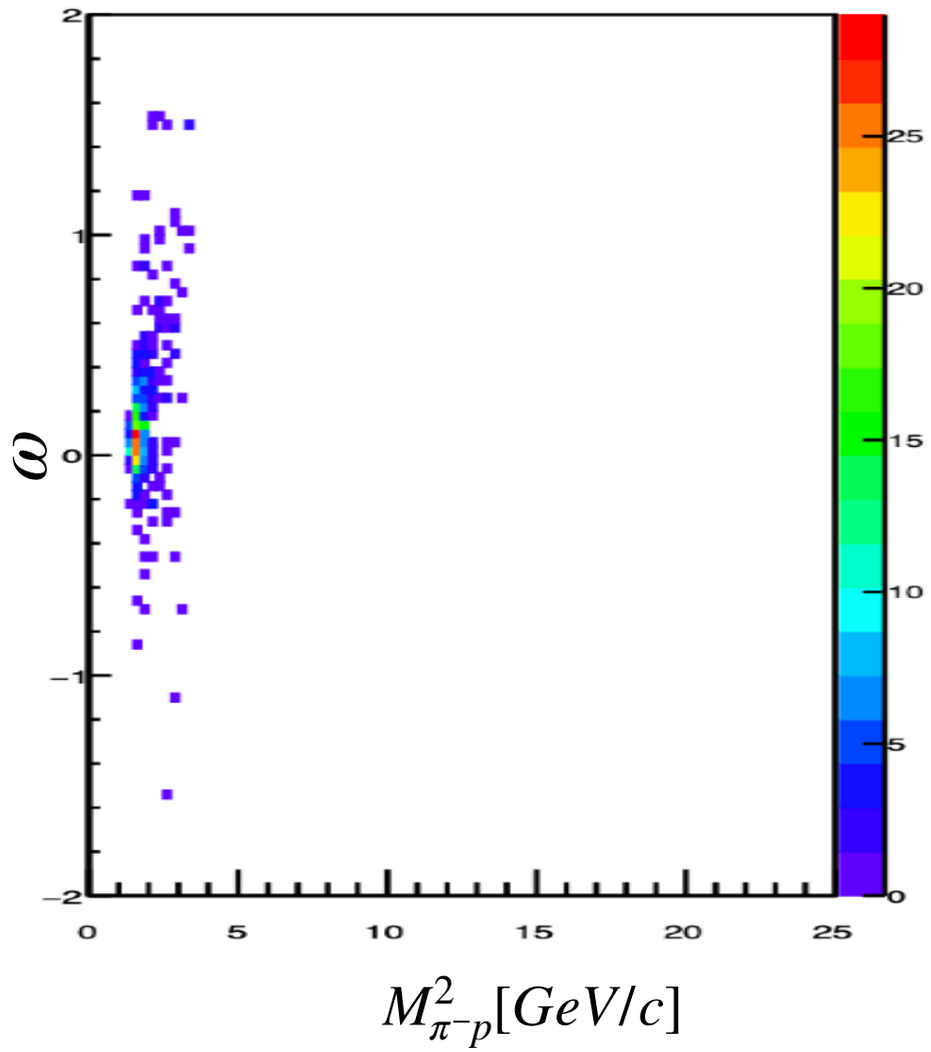
  Z_{center} : Z position from the center of the target (65 cm)

- $t > -1GeV^2, u > -1GeV^2$

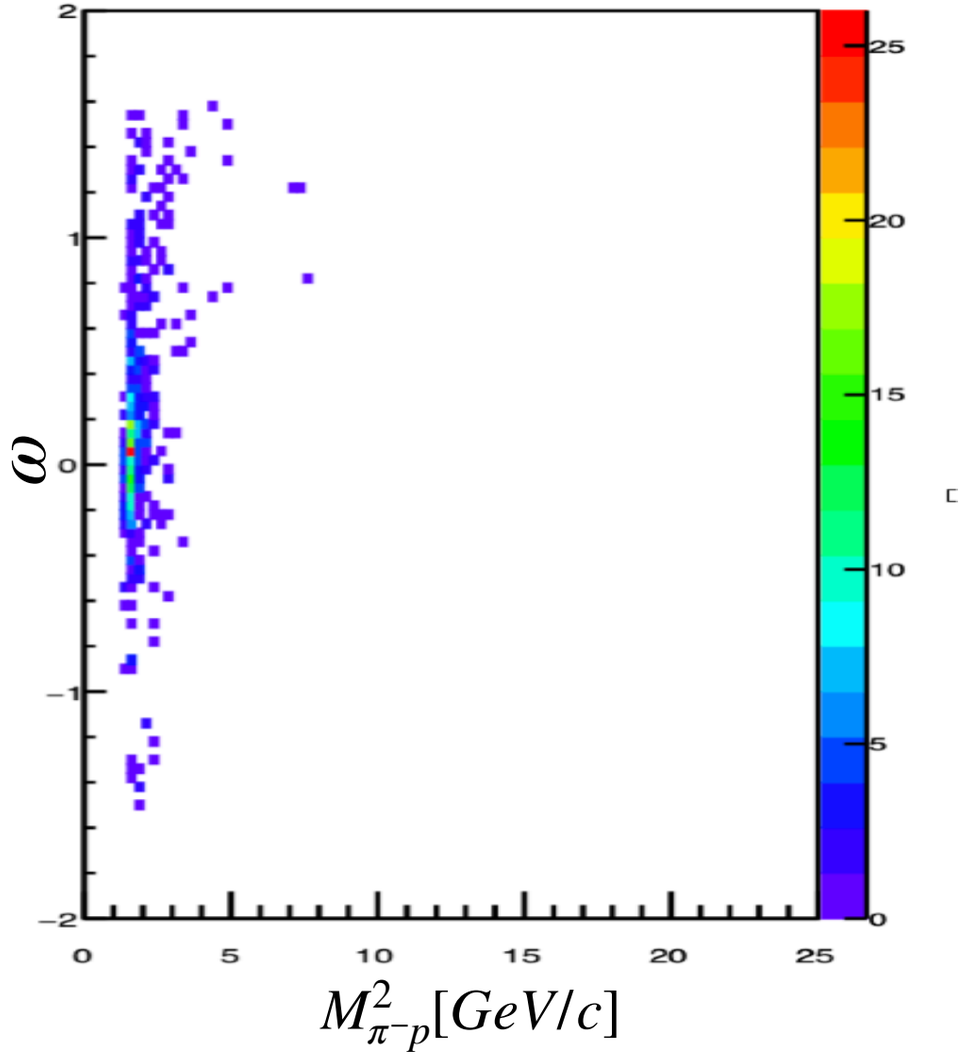
2. Miss-reconstructed PiPlus and Proton

- The miss-reconstructed Pi+ and Protons are suppressed by using: $m^2_{\pi^- p} > 5\ \mathrm{GeV}^2$

Example of simulated events reconstructed by purposely exchange of PiPlus <-> Proton

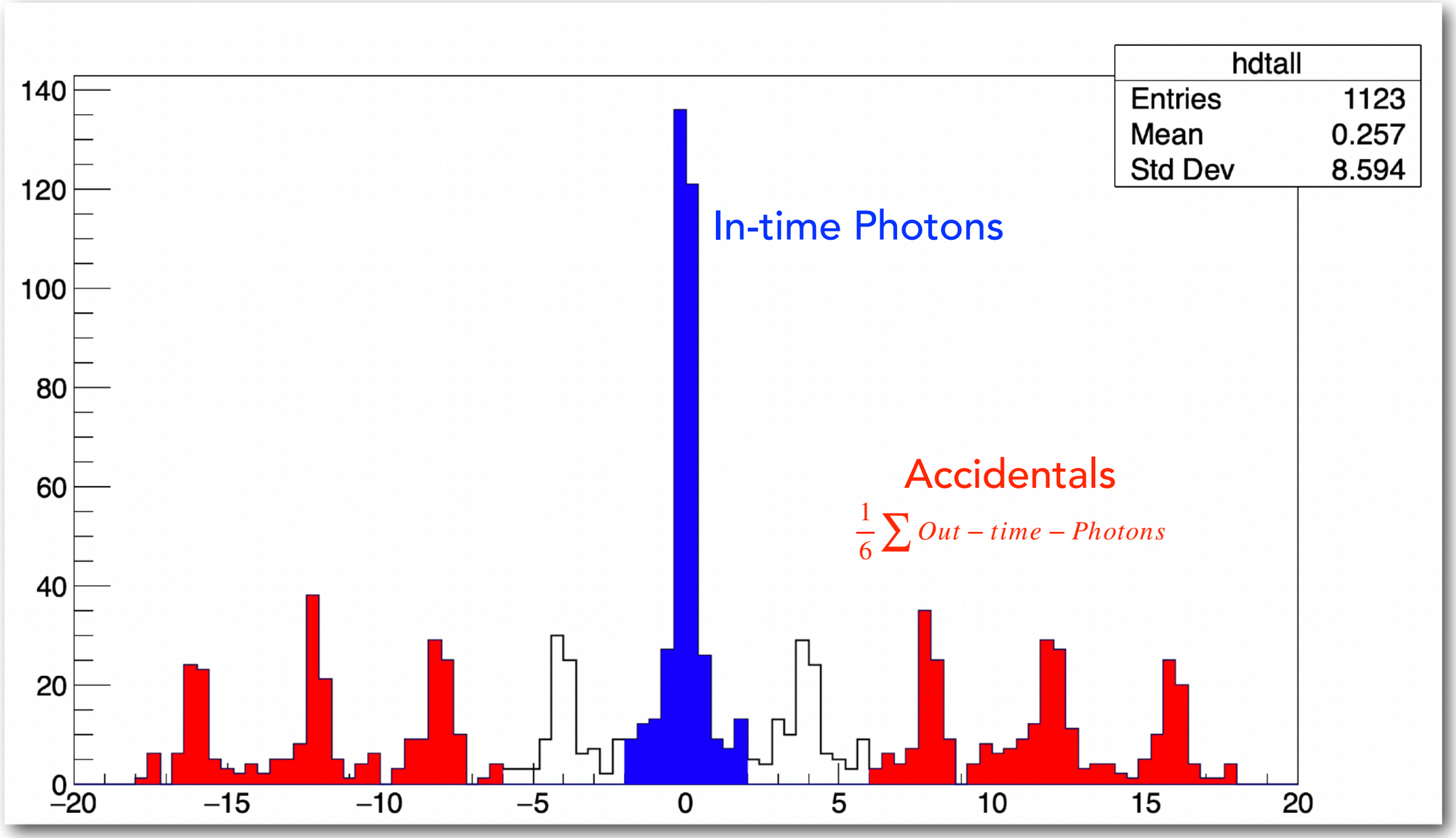


ω is the angle in the Van Hove Plots



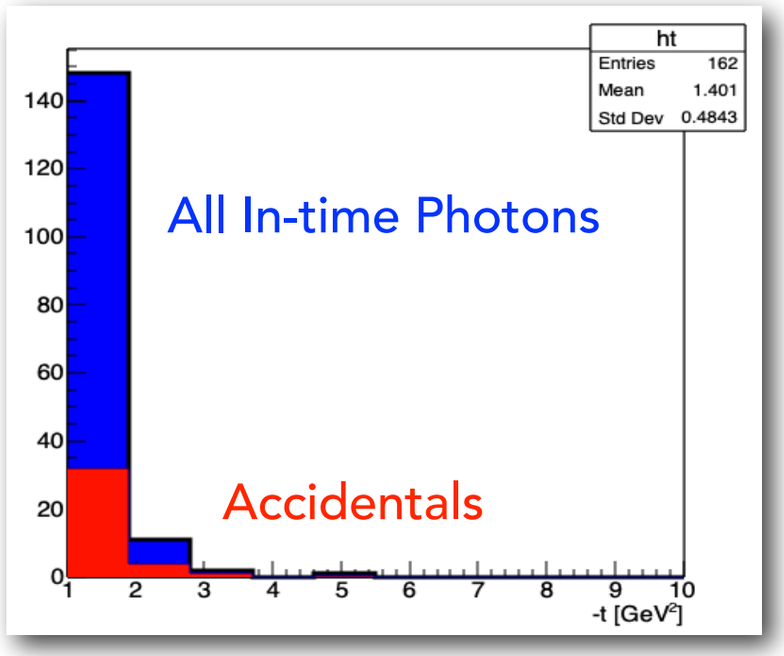
3. Accidentals Subtraction

Of all variables that required the beam energy, the accidentals are subtracted:

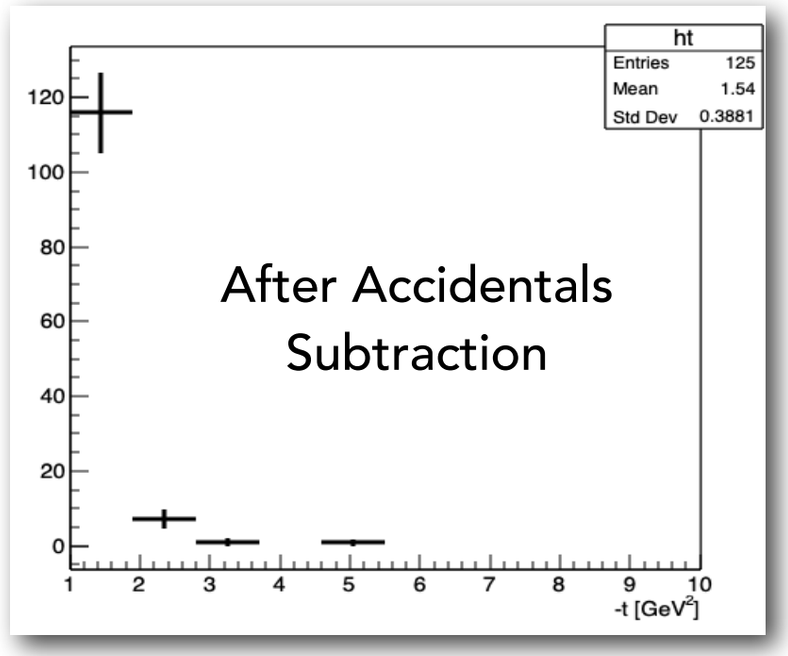


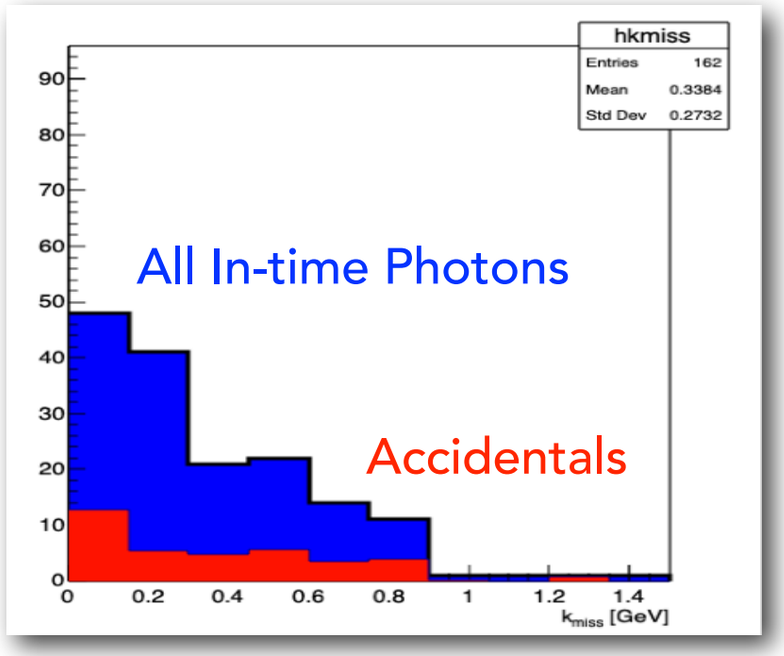
Only in-time photons are selected

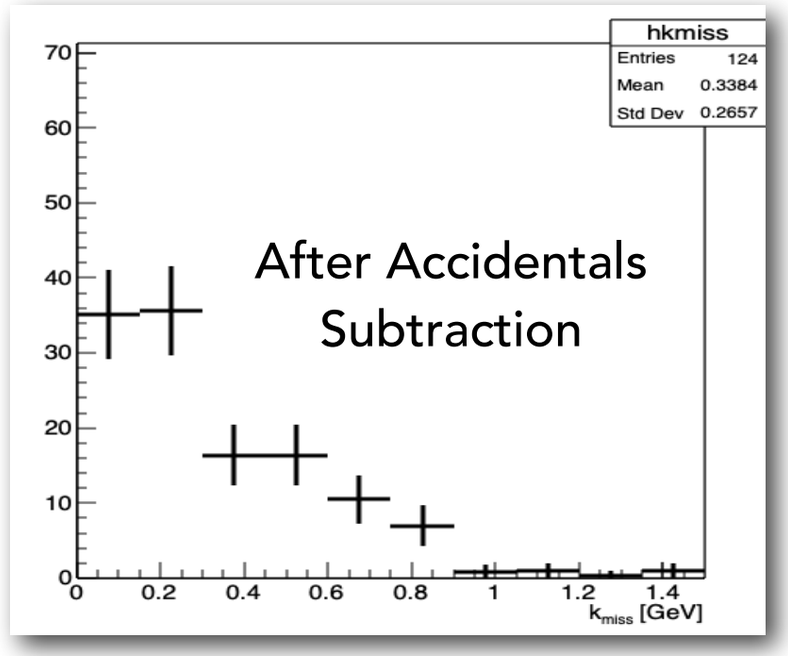
Example:



After Accidental Subtraction

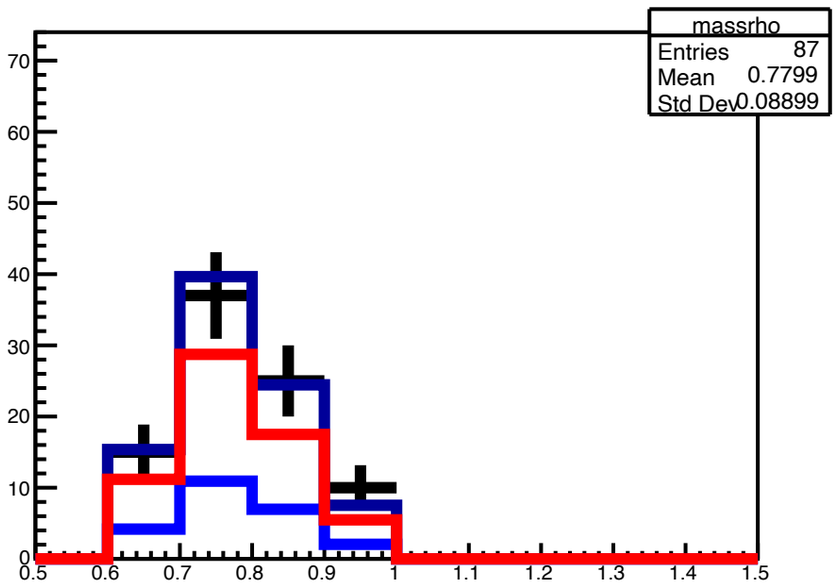






4. Comparing data with simulation

Note: The simulation is area normalized to match the data



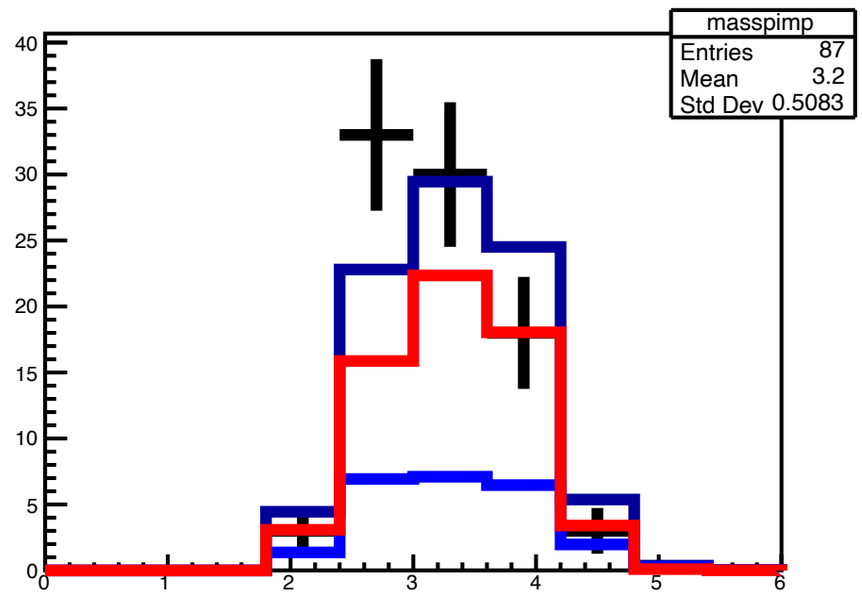
$M_{\pi^+\pi^-}[GeV]$

+ **Data**

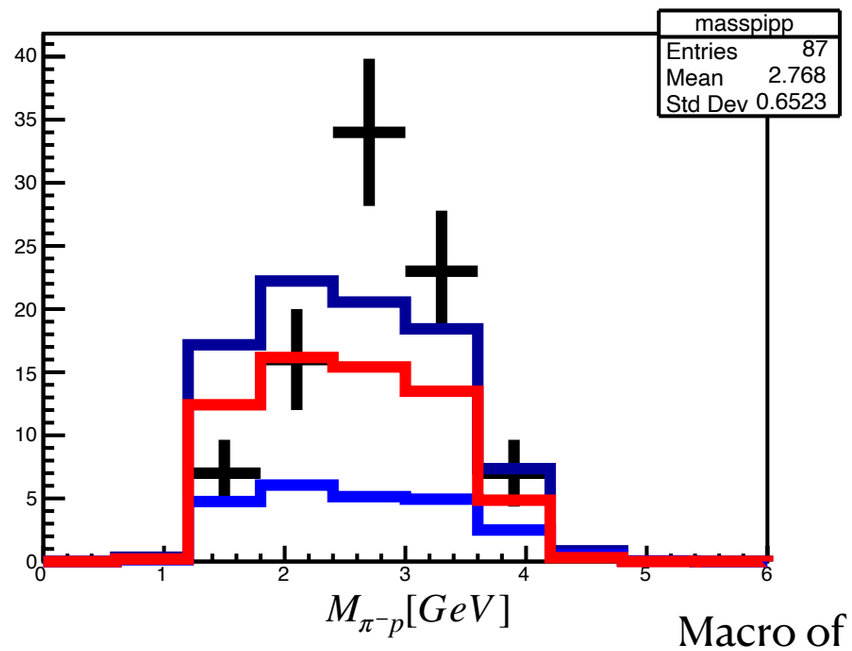
MF+SRC Simulation

MF Simulation

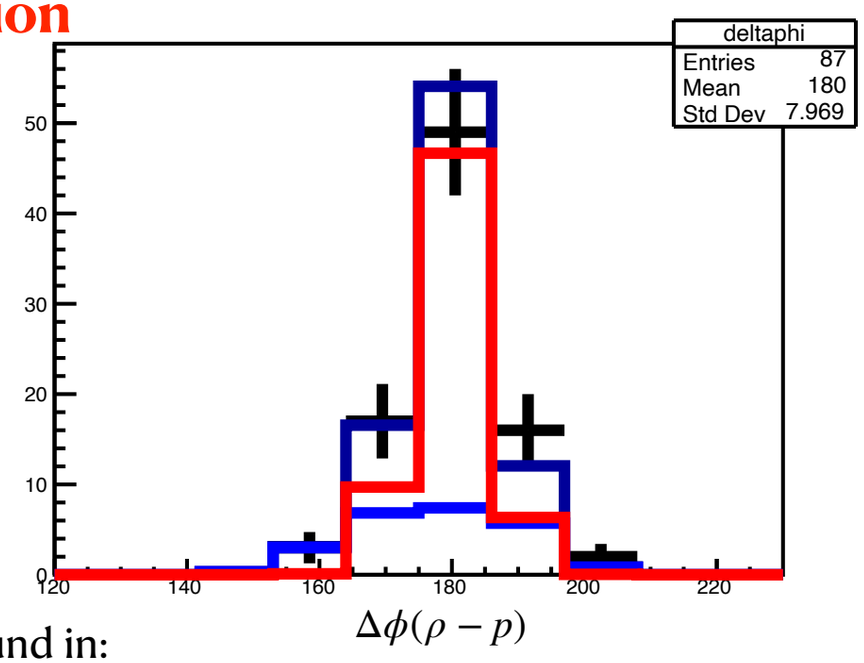
SRC Simulation



$M_{\pi^-p}[GeV]$



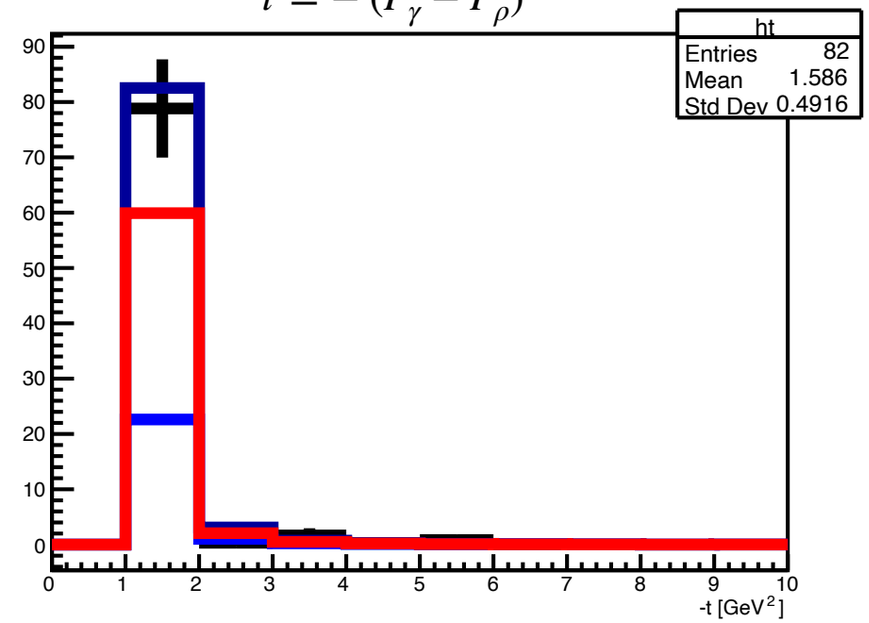
$M_{\pi^-p}[GeV]$



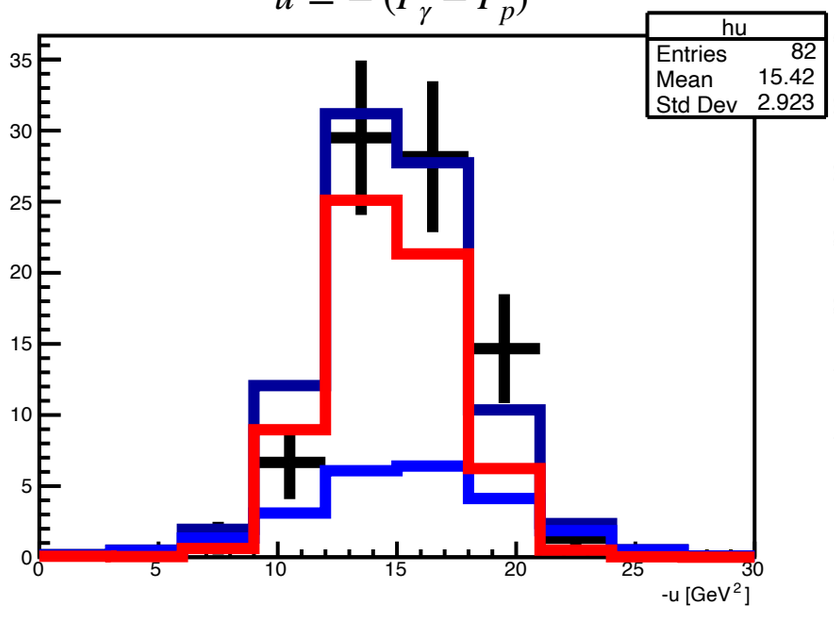
$\Delta\phi(\rho - p)$

Macro of the plots can be found in:
/w/halld-scifs17exp/halld2/home/nathaly/test/scripts/plots_data.C

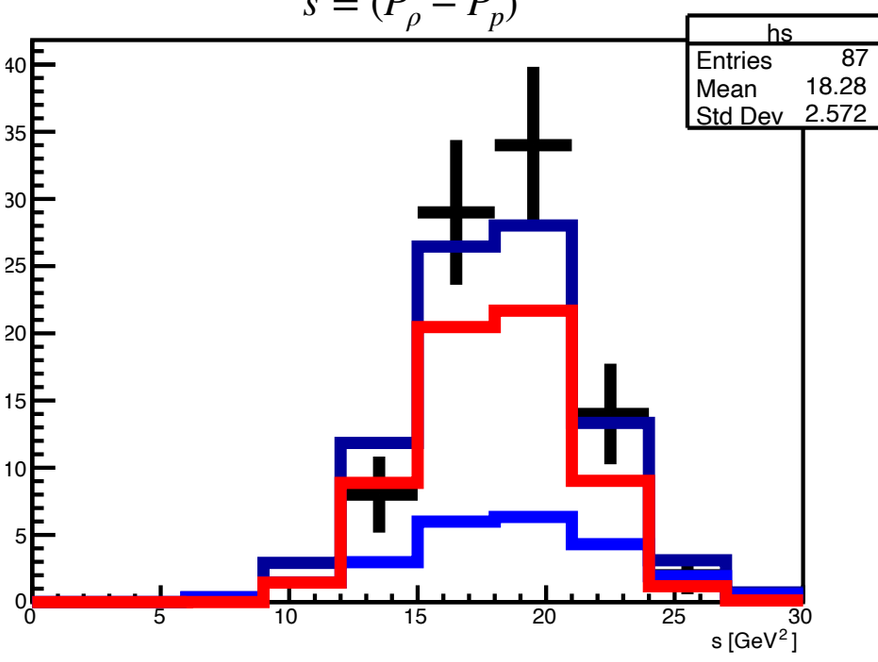
$t = -(P_\gamma - P_\rho)^2$



$u = -(P_\gamma - P_p)^2$



$s = (P_\rho - P_p)^2$



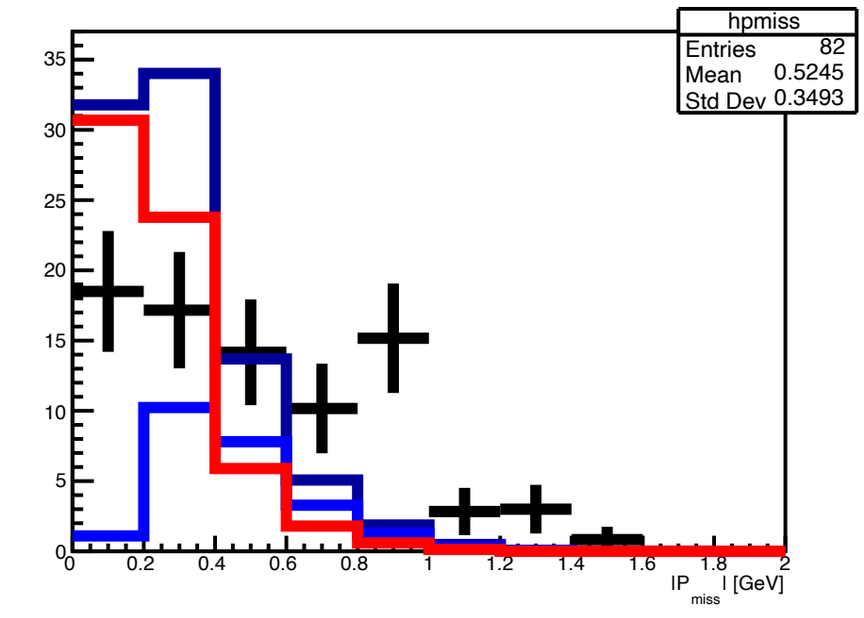
+ **Data**

MF+SRC Simulation

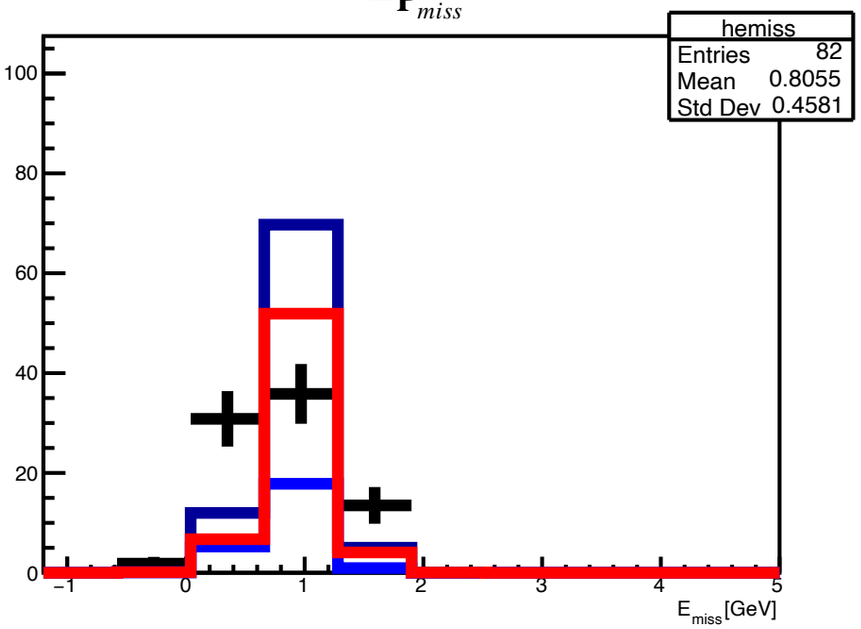
MF Simulation

SRC Simulation

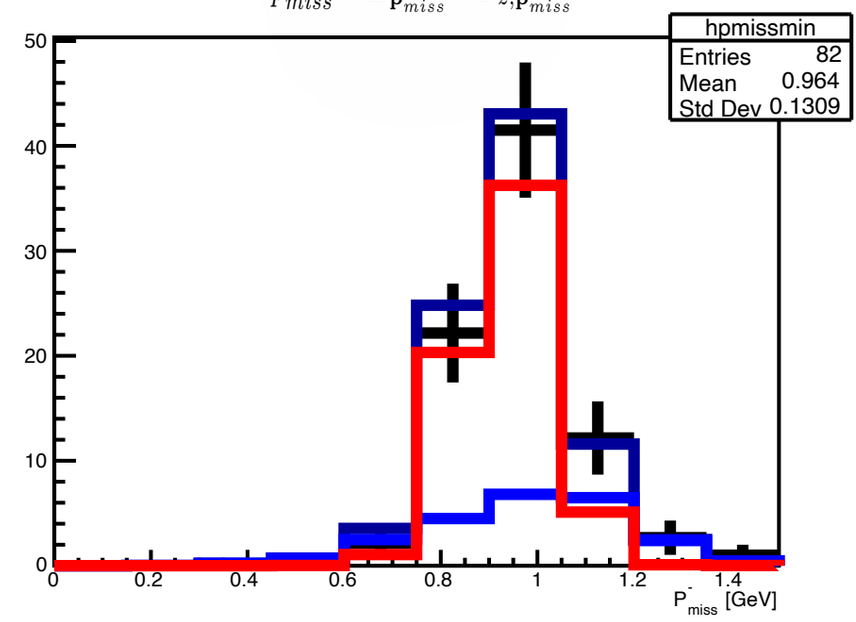
$\mathbf{p}_{miss} = \mathbf{p}_p + \mathbf{p}_{\pi^+} + \mathbf{p}_{\pi^-} - \mathbf{p}_\gamma$

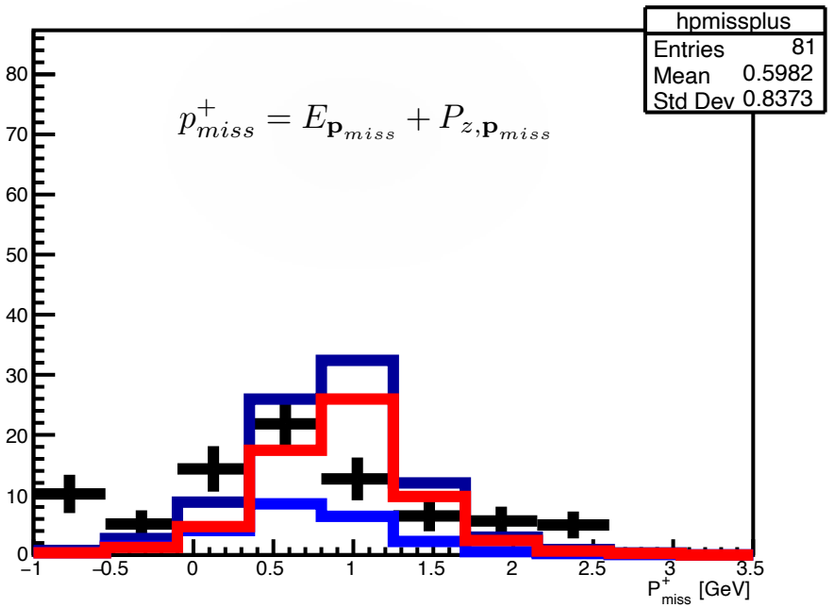


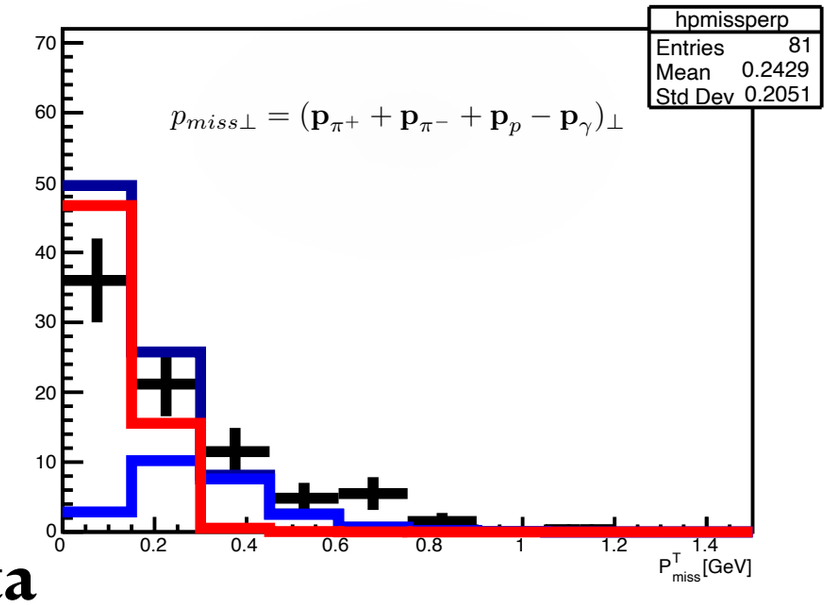
$E_{\mathbf{P}_{miss}}$



$p^-_{miss} = E_{\mathbf{p}_{miss}} - P_{z,\mathbf{p}_{miss}}$





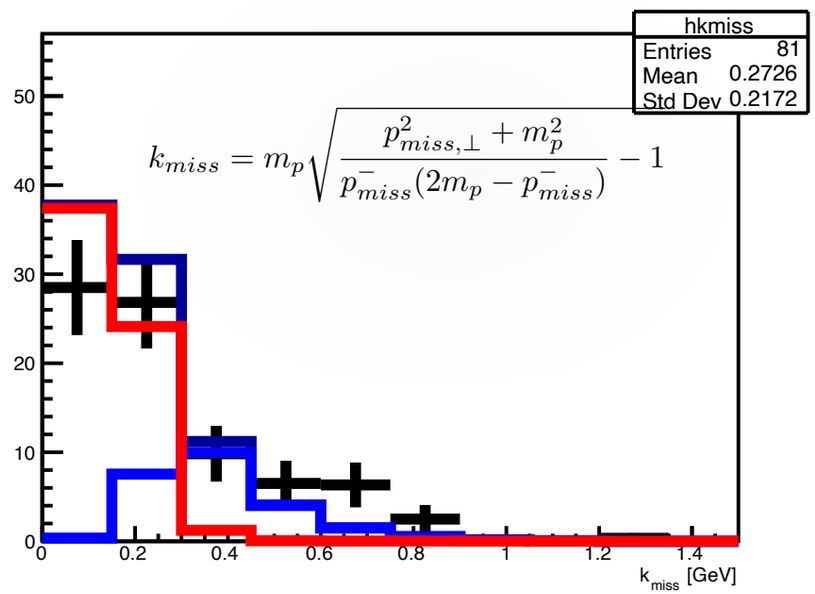


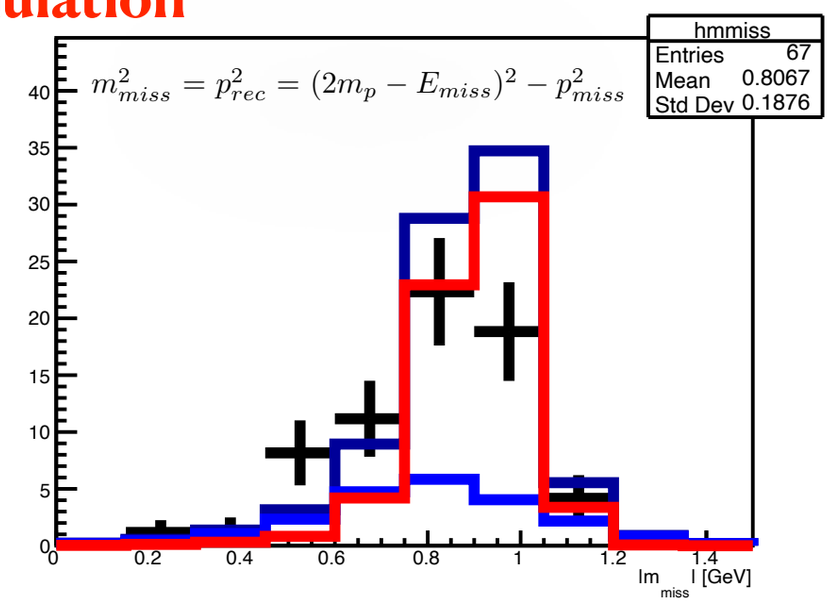
+ **Data**

MF+SRC Simulation

MF Simulation

SRC Simulation





Simulation

$$\gamma p \rightarrow \rho p \rightarrow \pi^{+}\pi^{-}p$$

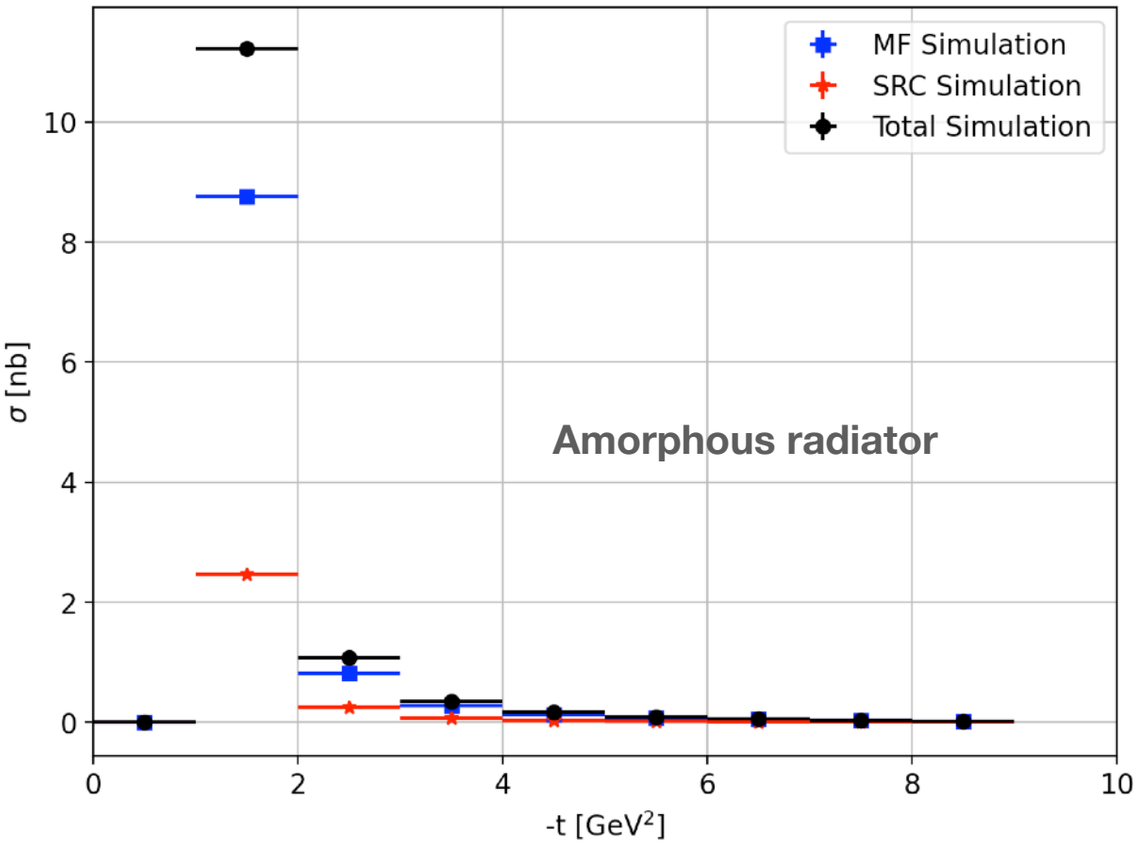
SIMULATION

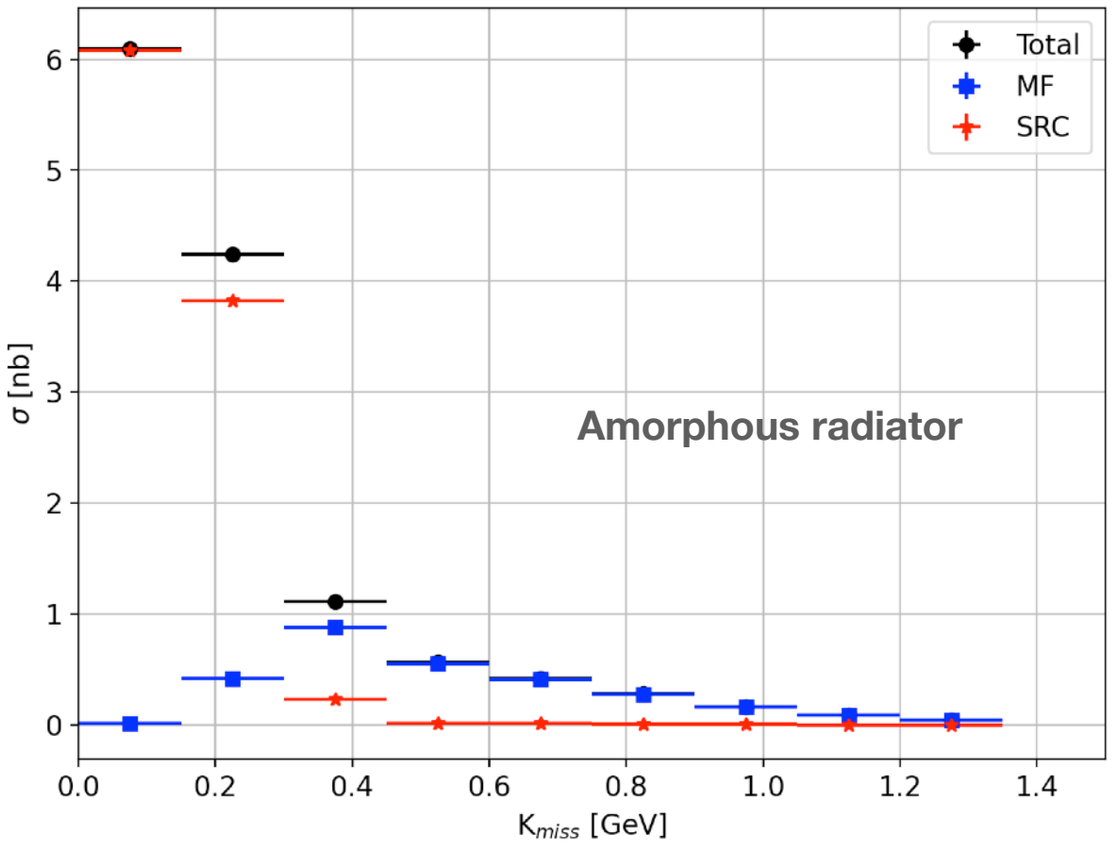
- Generated for ^{4}He
- Two components Mean Field (MF) and Short Range Correlation (SRC)
- Uses hdgeant4
- All events in the simulation was from the center of the target.
- The simulation events have been selected using the same logic than the data. (Requirements for the event selection)

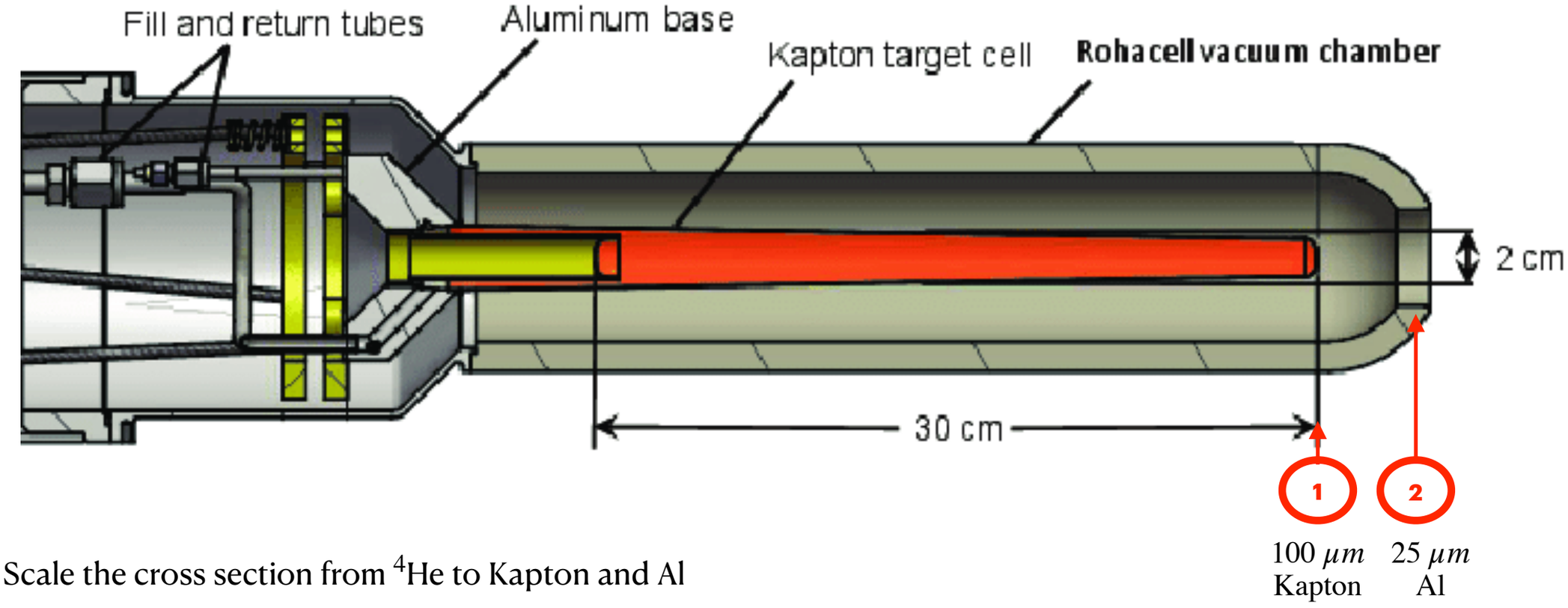
$$\sigma = \frac{N}{\epsilon \mathscr{L}}$$

σ: Cross section
ϵ: Efficiencies
$\mathscr{L}$: Luminosity

Used an amorphous radiator, since the data was taken with an amorphous radiator.







Scale the cross section from ^{4}He to Kapton and Al

Scale it by :

$\rho(^4\text{He}) = 0.117\ \text{g/cm}^3$

$\rho(\text{Kapton}) = 2.7\ \text{g/cm}^3$

$\rho(\text{Al}) = 1.42\ \text{g/cm}^3$

$(0.2*\rho(\text{Al})+0.8*\rho(\text{Kapton}))/\ \rho(^4\text{He}) = 14.32$

$$N_{ev} = \sigma \cdot flux \cdot thickness \cdot 14.32$$

Flux

Total Flux: 1.55E+12 γ on target for the empty cell run
Data is normalize by the Total Flux and then scaled by the
Expected flux 2E7 γ/s

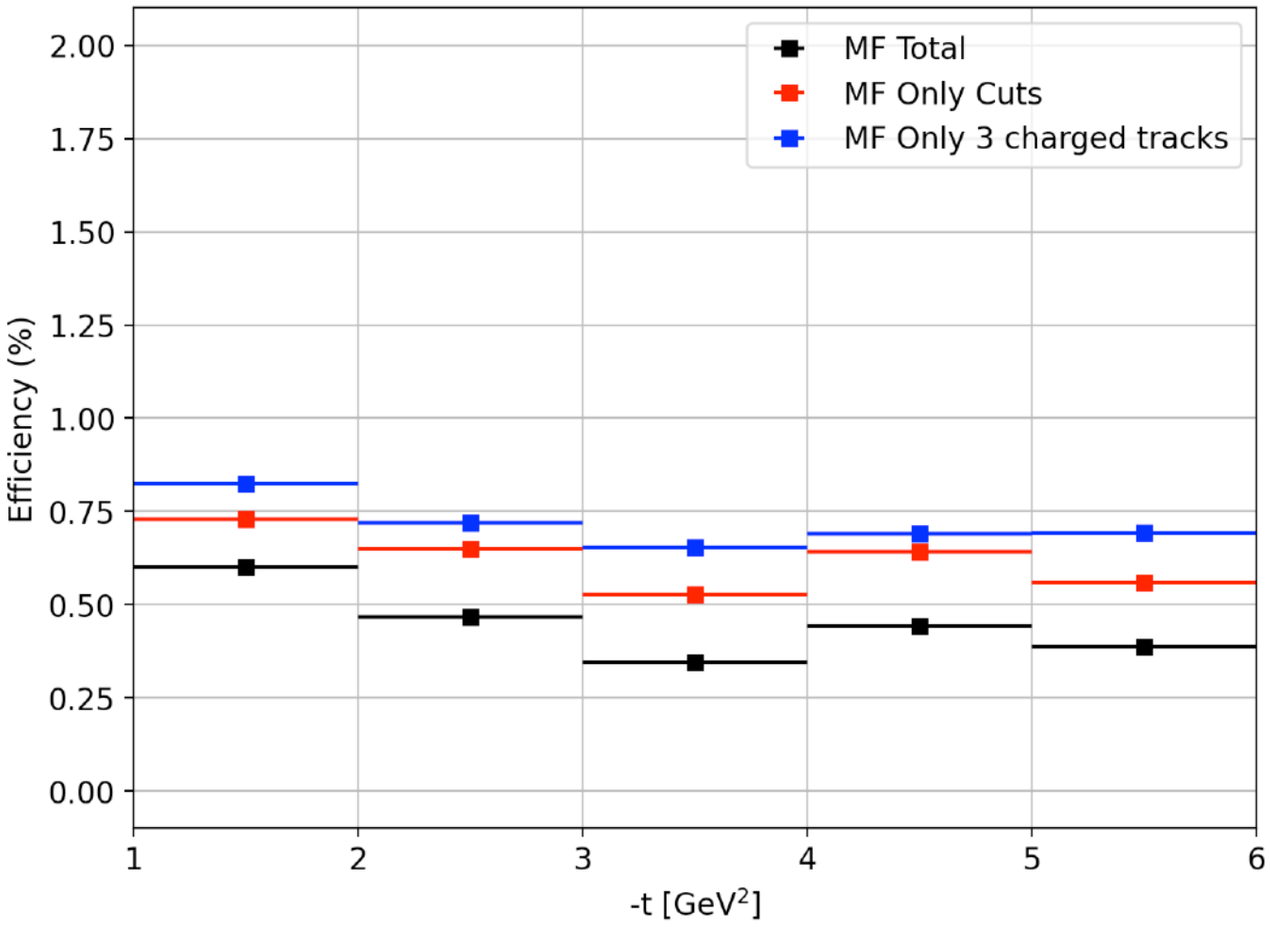
Target Thickness

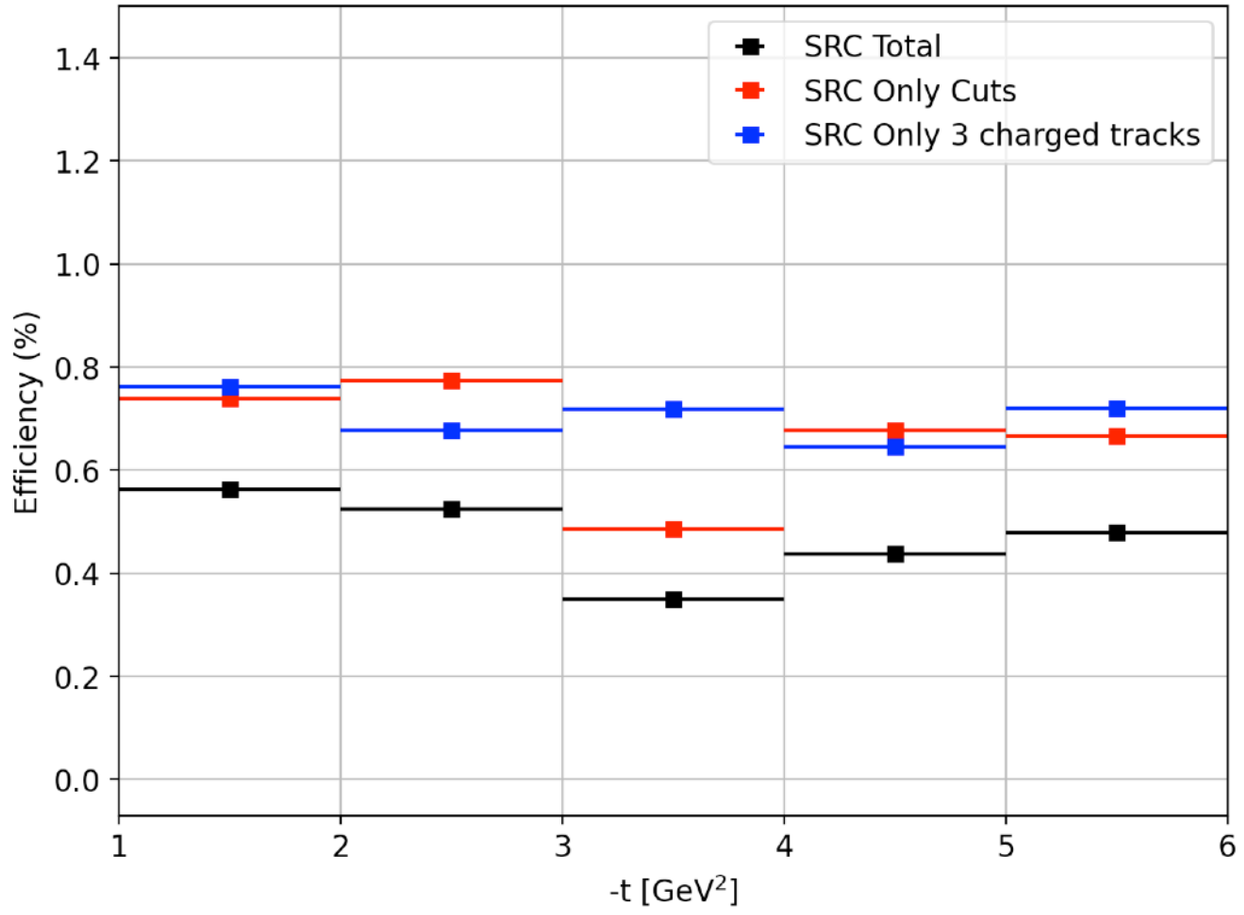
	Thickness [cm] / % X0	Atoms/cm2
4He	30 / 4	5.68E+23

Efficiencies

$$\epsilon = \frac{N_{reconstructed}}{N_{thrown}}$$

Monte-Carlo based: Assumes the simulation reproduces the data.





Rate Estimates

Comparison data with simulation