SOUTH DAKOTA MINES

Particle ID in LANL test beam analysis - Status

Diana Leon
South Dakota School of Mines and Technology

Jun 20, 2023

SOUTH DAKOTA MINES

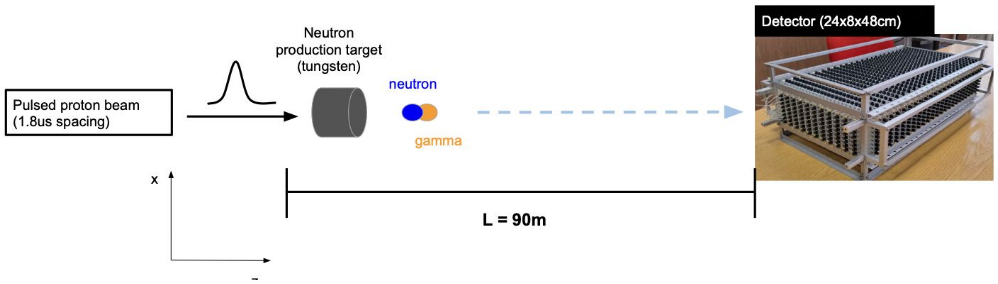
OUTLINE

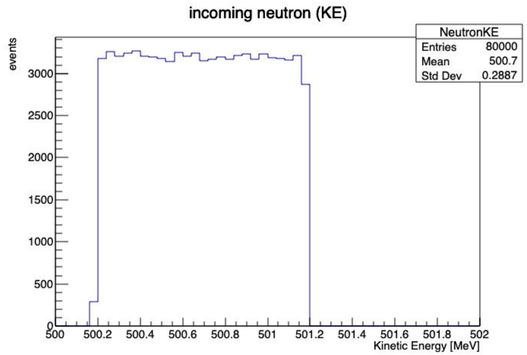
- Event reconstruction
- Event selection topology
- dE/dl calculation using linear fit

Neutron sample for PID application with SuperFGD prototype

Goal: Develop a PID using neutron MC simulation in the SuperFGD prototype detector (24x8x48cm):

- Beam placed at 90 meters from the detector
- Kinetic energy between 500-501 MeV (for tools development)





Neutron sample for PID application with SuperFGD prototype

1. Event reconstruction

2D hits → **3D hits** → **Clusters** → **Vertices**

2D hits
- PE cut on hits
- Event register more than 3 hits

3D hits
- 3D view matching of time - clustered hits
- 3 hits sharing the same XYZ coordinates

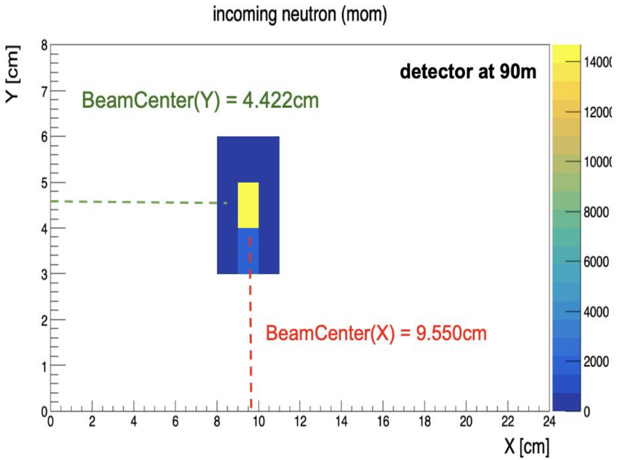
Clusters
- DBSCAN clustering of voxels
- Maximum distance of 1.8 cm between voxels in the same cluster

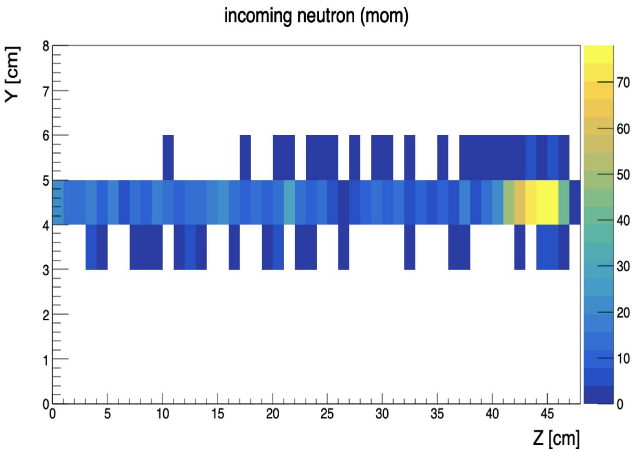
Vertices
- Vertex finding with voxel earliest in Z

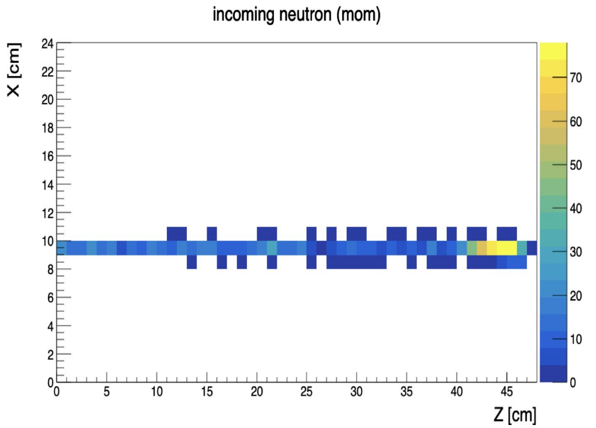
Neutron sample for PID application with SuperFGD prototype

2. Defining a FV: reduce the beam uncorrelated and secondary neutron background

- Using the incoming neutron information

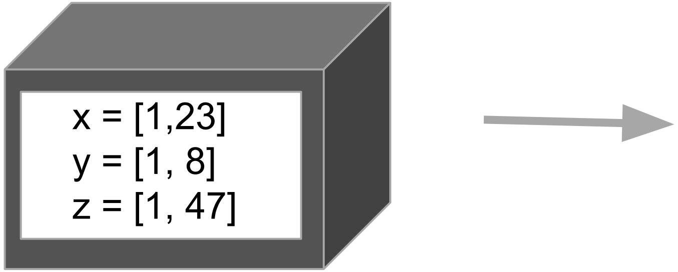


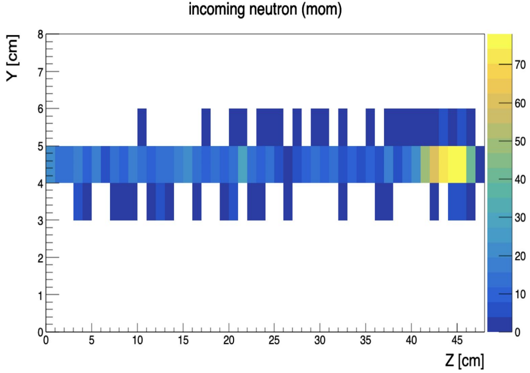


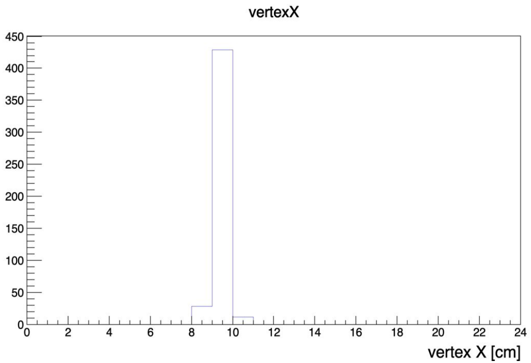


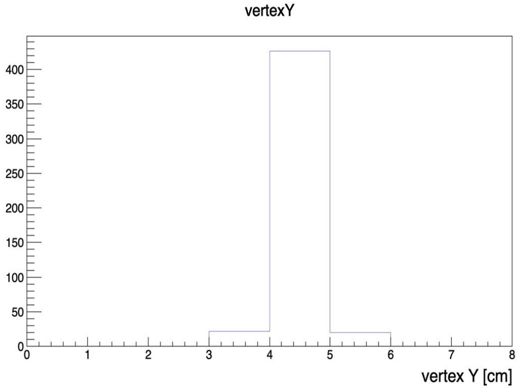
Neutron sample for PID application with SuperFGD prototype

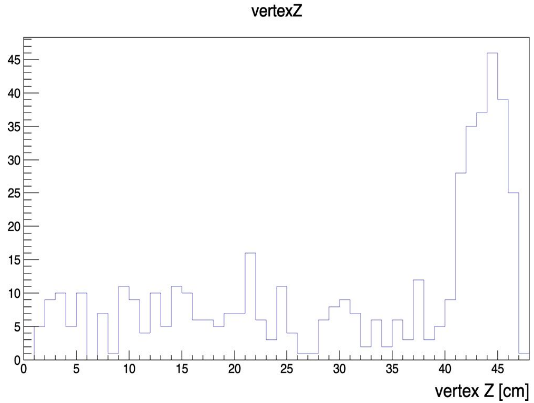
3. Containment function





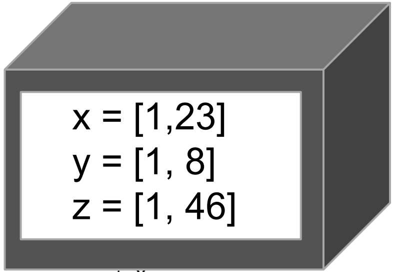


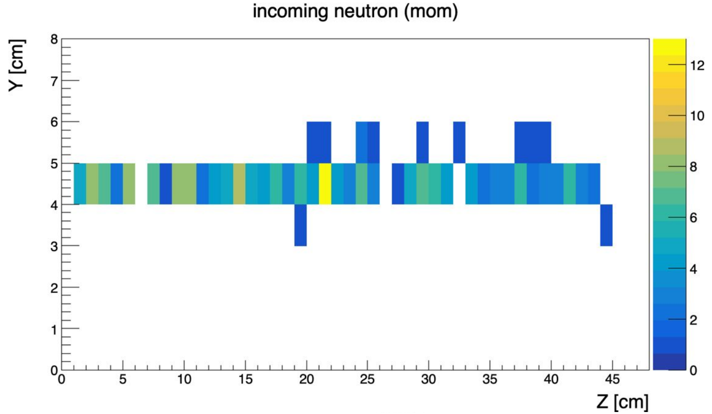


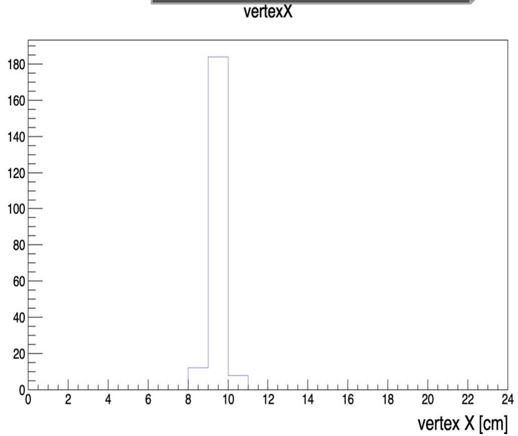


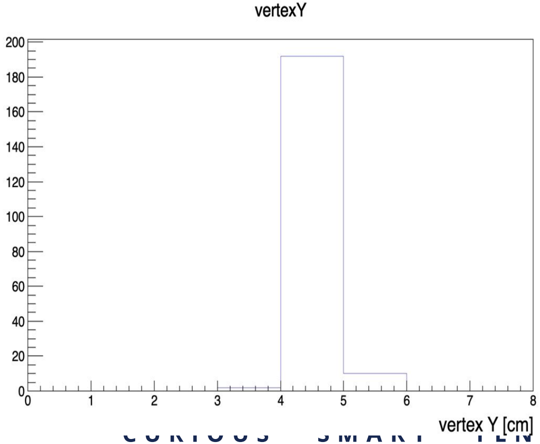
Neutron sample for PID application with SuperFGD prototype

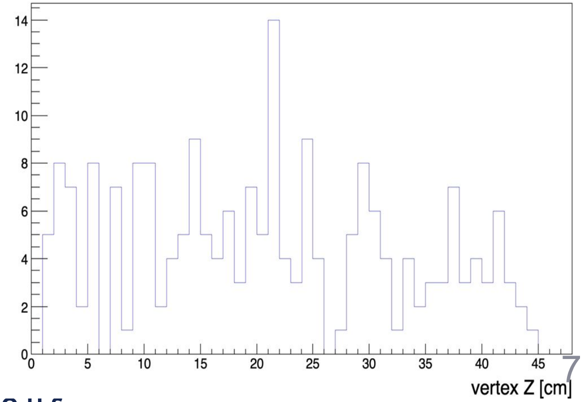
3. Containment function







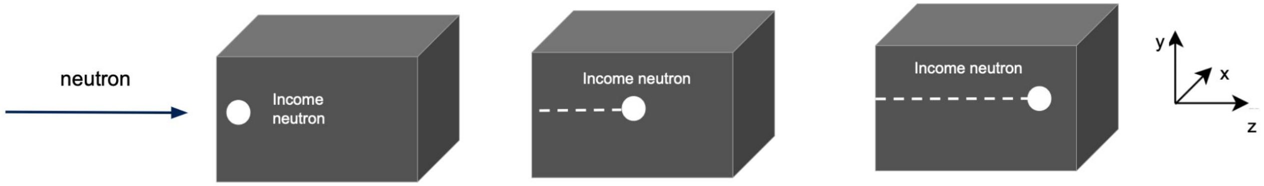


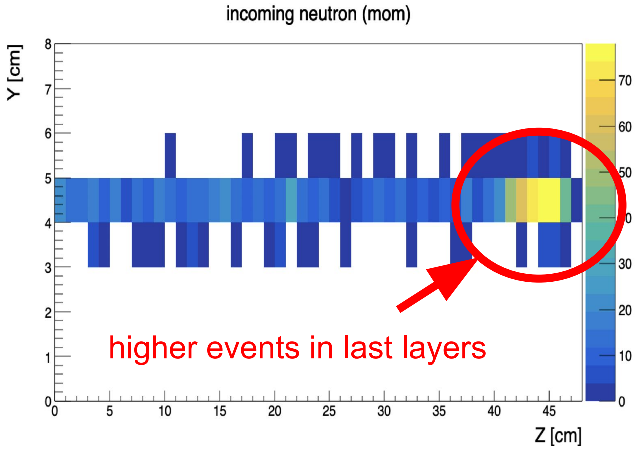


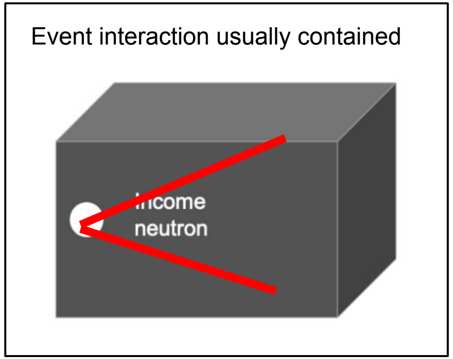
SOUTH DAKOTA MINES

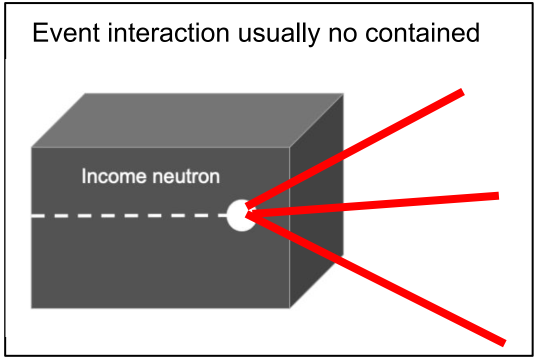
Neutron sample for PID application with SuperFGD prototype

2. Defining a FV





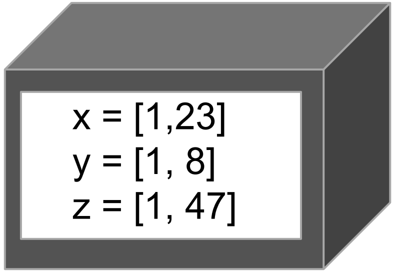




Neutron sample for PID application with SuperFGD prototype

3. Containment function:

For the moment, the events are requested to be contained in:



4. Events topology

- Single cluster
- Single DBSCAN cluster of voxel
- 3-8 voxels in single cluster
- linearity > 0.70
- cluster width < 1.4 and max-vox-line < 1.2
- vertex FV (1.5cm radius around the beam center

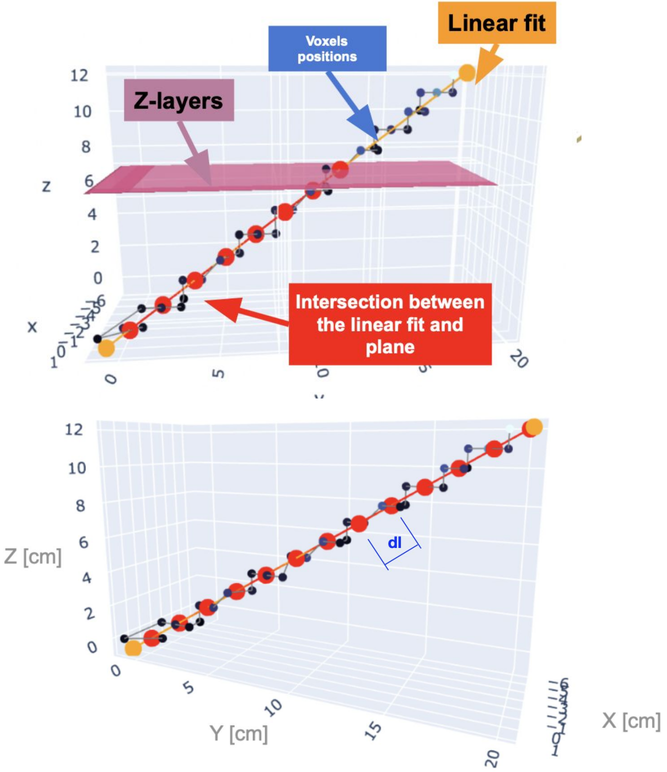
```
//----------------------------------------------------------------------------

// Defining global variables
bool batch = true;              // do you want to show the canvas (plots)
bool IsMC = true;               // is it MC (true or false)?
bool IsTwenty = false;          // Beam data from 20m location from beam to detector
bool IsNinety = true;           // Beam data from 90m location from beam to detector
bool IsRotated = false;         // Detector rotated by 180
bool IsCalibrated = false;      // MC sample
bool IsAttenuated = true;       // Attenuated
bool IsTimeCorrected = false; // TimeCorrected

double MaxClustDist = 1.8;
bool LinearityCut = true;
float Linearity = 0.70;

bool ClusterWidthCut = true;
float ClusterWidth = 1.4;

bool FVCut = true;
float FV = 1.5;

float mass = 939.56;

int MinVoxel = 3;
int MaxVoxel = 8; // same as voxel range selection. analize if we need to increase!
bool MaxVoxLineCut = true;
float MaxVoxLine = 1.2;
int HitMaxZ = 48;

int PE = 20;
int PE1 = 20;
int PE2 = 30;
int PE3 = 20;
int PE4 = 40;
int PE5 = 60;
int PE6 = 60;
int PE7 = 50;
```

Neutron sample for PID application with SuperFGD prototype

5. dE/dl calculation using linear fit

We proceed to calculate the energy deposition per length (dE/dl) using linear fit:

1. The linear fit is calculated using the principal vector (calculated by PCA)
2. Slicing the detector along z-axis with planes (z-layers)
3. We calculate the length dl per event using intersection between the linear fit and planes (red points)
4. We estimate the energy deposition in the specific length



Neutron sample for PID application with SuperFGD prototype

Explicit calculation of length and energy deposition

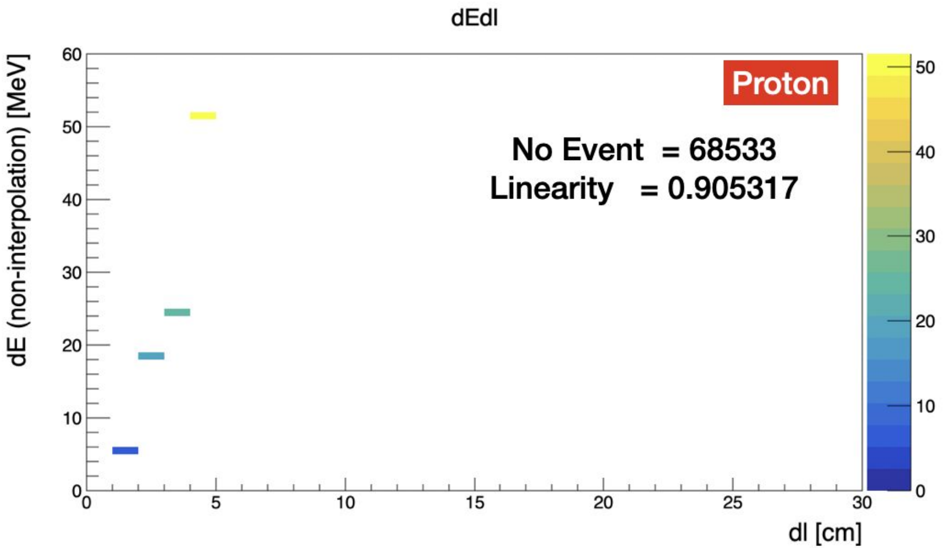
Length using intersection points

```
// Calculate the distance between consecutive rows
std::vector<double> df_points_energy;
std::vector<double> df_points_length;

double temp_distance = 0;
int interpolation_counter = 0;

for (std::size_t i = 0; i < points_intersection.size() - 1; ++i)
{
    double distance = euclideanDistanceFinal(points_intersection[i], points_intersection[i + 1]);

    temp_distance += distance;

    df_points_length.push_back(temp_distance);
    df_points_energy.push_back(df_energy[i]);

    dEdl->Fill(temp_distance, df_energy[i], df_energy[i]);

    //std::cout << "dl = " << temp_distance << " Edep (no interpolation) = " << df_energy[i] << std::endl;

    interpolation_counter += 1;
}
```

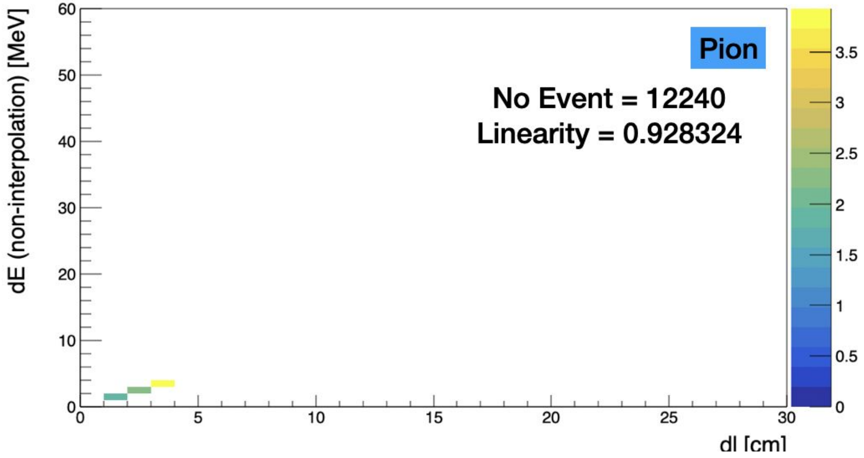
Deposited energy between z-layers

```
// Sum energies with respect to the insersections
double minElement = *std::min_element(z_points_intersection.begin(), z_points_intersection.end());
double maxElement = *std::max_element(z_points_intersection.begin(), z_points_intersection.end());

std::vector<double> df_energy;

for (int i = minElement; i < maxElement; ++i)
{
    double sum_energyPE = 0.0;
    for (int j = 0; j < z.size(); ++j)
    {
        if (i <= z[j] && z[j] < i + 1)
        {
            sum_energyPE += edep[j];
        }
    }
    df_energy.push_back(sum_energyPE);
}
```

Neutron sample for PID application with SuperFGD prototype



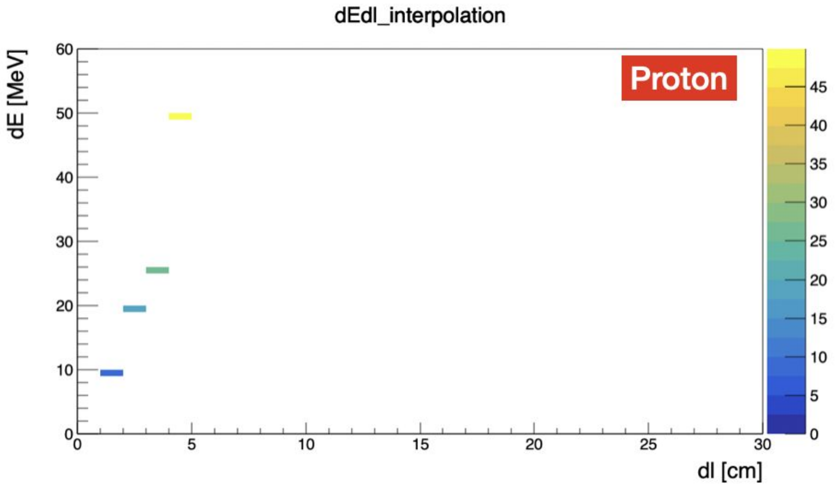

```
dl = 1.14173 Edep (no interpolation) = 5.46262
dl = 2.28346 Edep (no interpolation) = 18.8775
dl = 3.42519 Edep (no interpolation) = 24.037
dl = 4.56692 Edep (no interpolation) = 51.4993
```



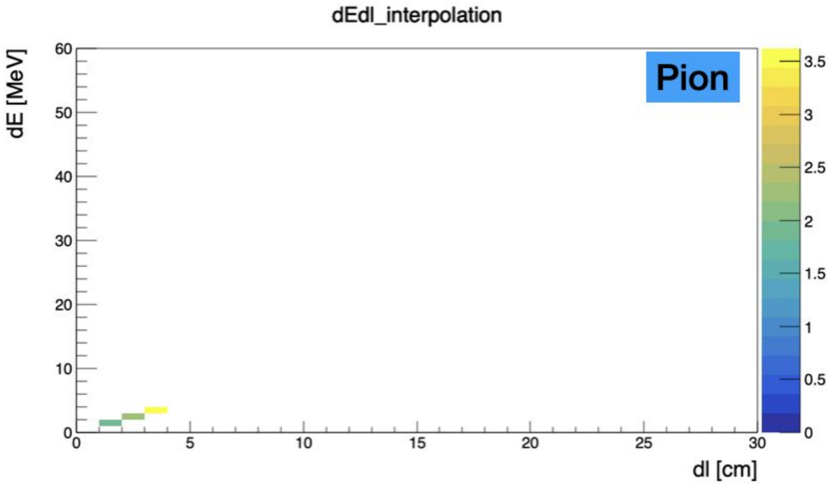

```
dl = 1.23804 Edep (no interpolation) = 1.86716
dl = 2.47609 Edep (no interpolation) = 2.17353
dl = 3.71413 Edep (no interpolation) = 3.92719
```

Neutron sample for PID application with SuperFGD prototype

Applying interpolation



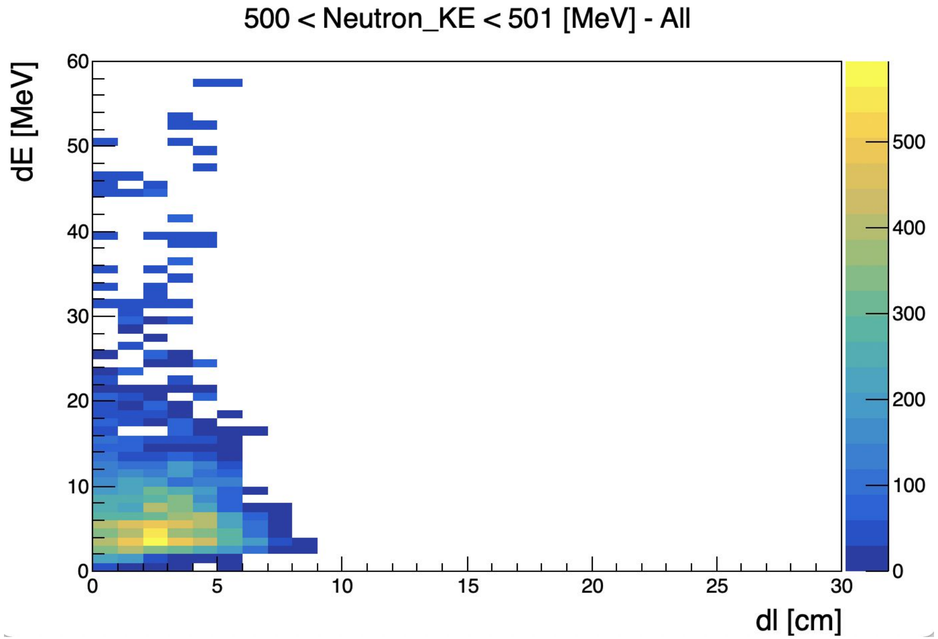

```
dl = 1.5 Edep (interpolation) = 9.67215
dl = 2.5 Edep (interpolation) = 19.856
dl = 3.5 Edep (interpolation) = 25.8365
dl = 4.5 Edep (interpolation) = 49.8897
```




```
dl = 1.5 Edep (interpolation) = 1.93199
dl = 2.5 Edep (interpolation) = 2.2074
dl = 3.5 Edep (interpolation) = 3.62388
```

Neutron sample for PID application with SuperFGD prototype

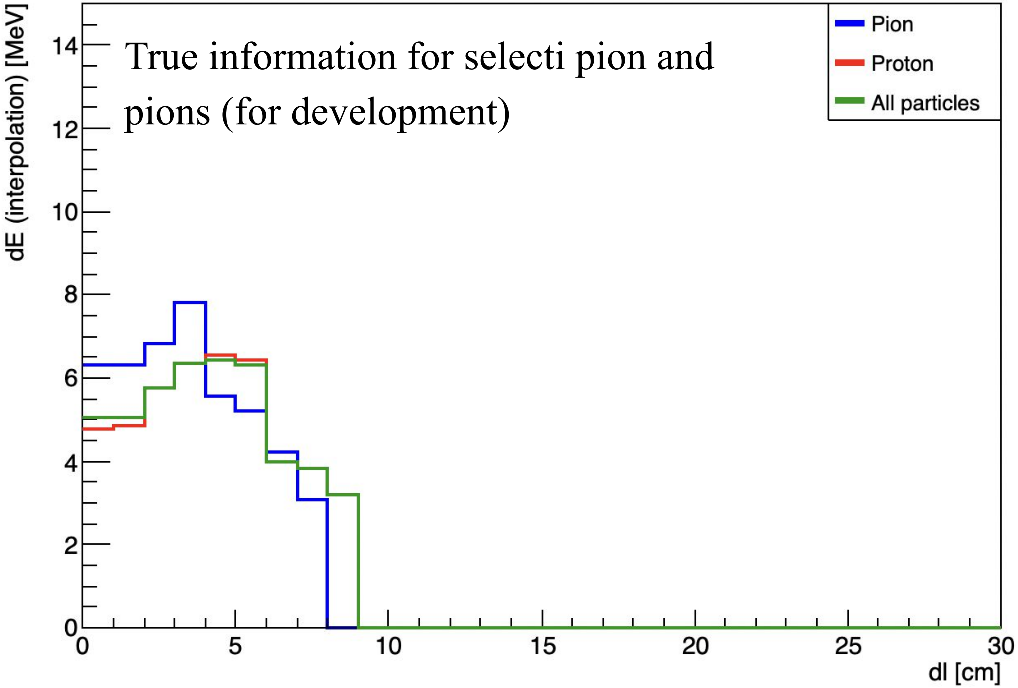
Calculation of dE/dl distribution



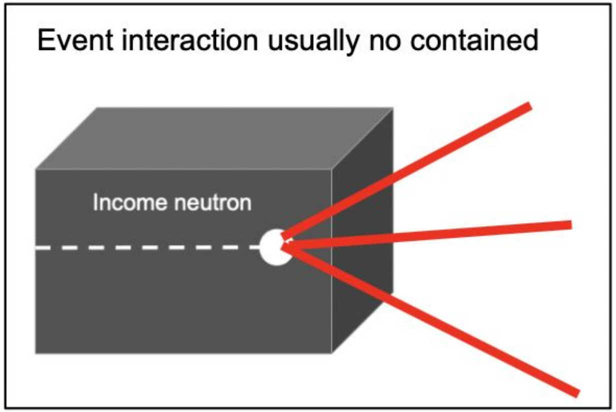
Events amount in %

```
Protons is: 88.6853
Pions is: 4.31034
Neutrons is: 0
Gammas is: 0
Others is: 1.93966
Alphas is: 1.2931
Deuterons is: 1.07759
He3s is: 0.431034
Pi_Zeros is: 0
Electrons is: 1.61638
Positrons is: 0.646552
Muons is: 0
AntiMuons is: 0
```

Neutron sample for PID application with SuperFGD prototype

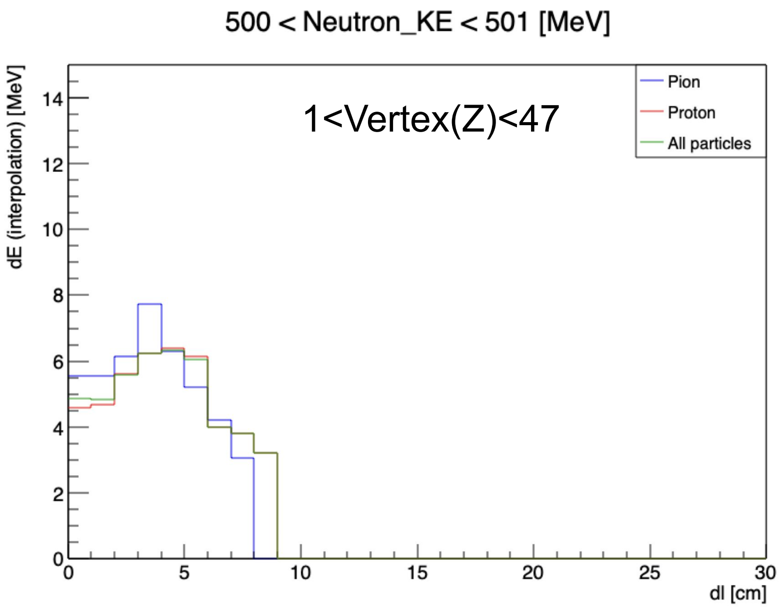


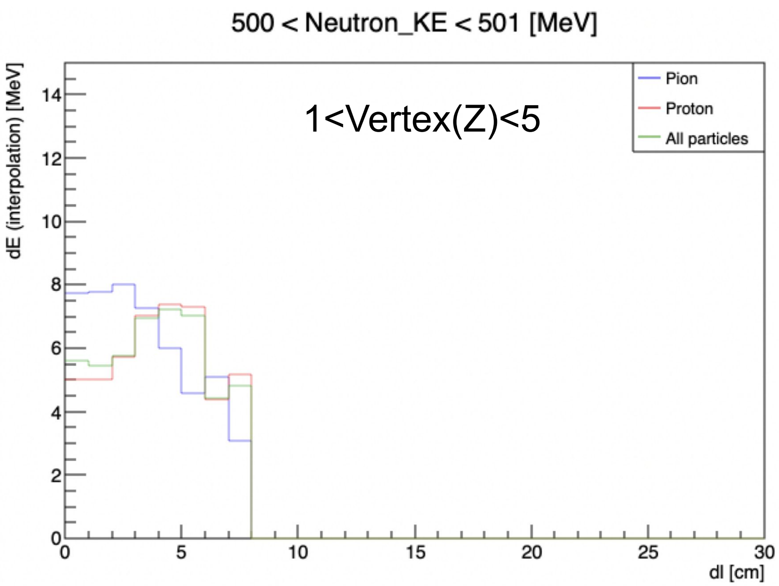
Why is not clear the bragg peak?



Neutron sample for PID application with SuperFGD prototype

dE/dl distributions applying a vertex(z) cuts

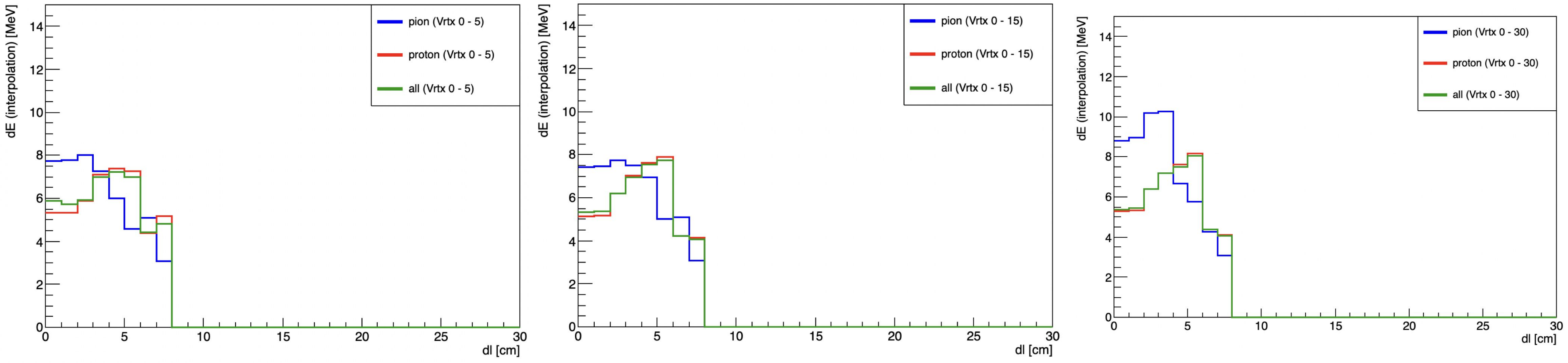




SOUTH DAKOTA MINES

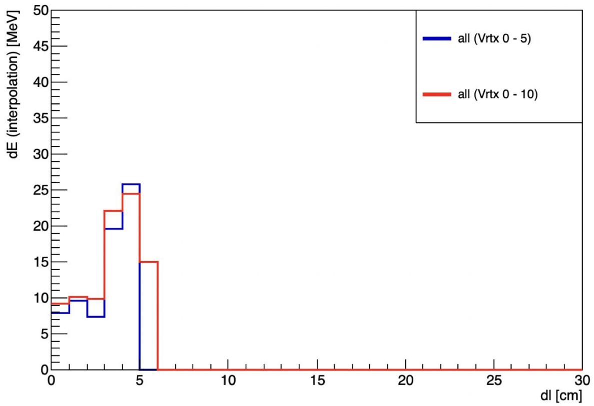
Neutron sample for PID application with SuperFGD prototype

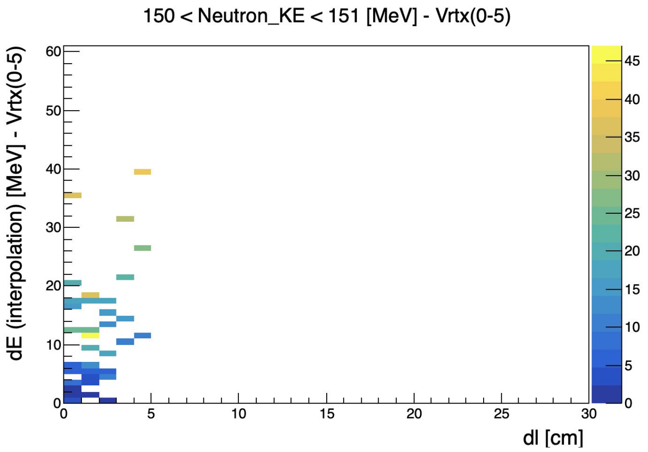
- Beam placed at 90 meters from the detector
- Kinetic energy between 500-501 MeV
- dE/dl distributions under different cuts

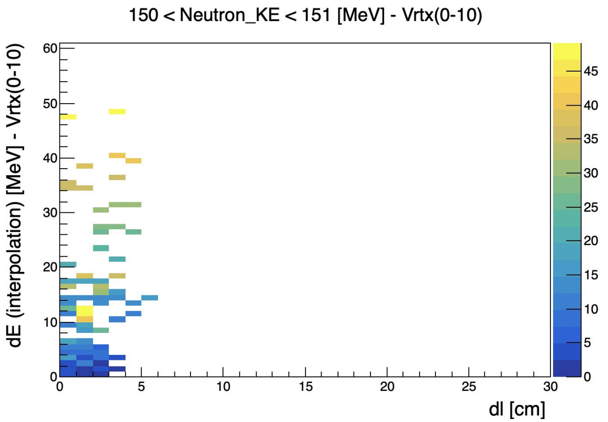


Neutron sample for PID application with SuperFGD prototype

- Beam placed at 90 meters from the detector
- Kinetic energy between 150-151 MeV
- dE/dl distributions under two vertex cuts

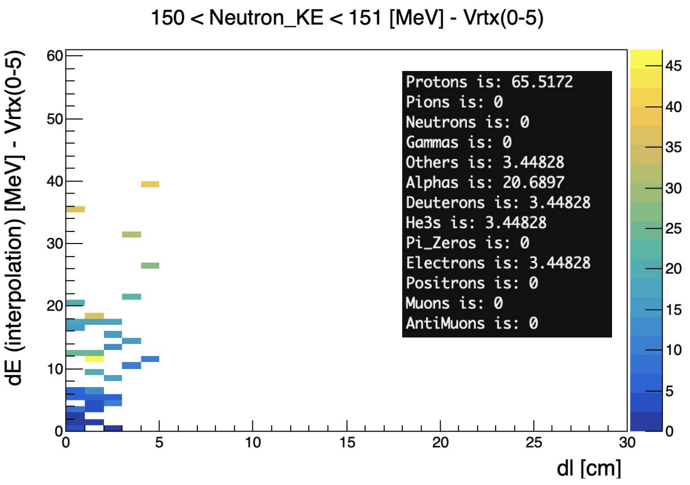


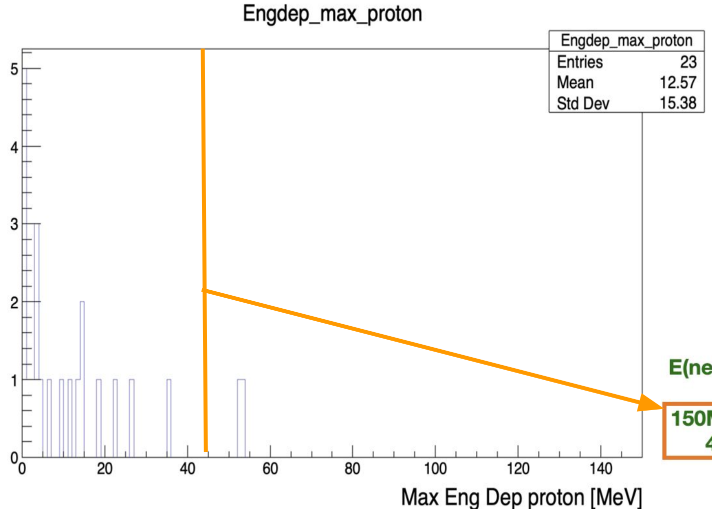




Neutron sample for PID application with SuperFGD prototype

- Beam placed at 90 meters from the detector
- Kinetic energy between 150-151 MeV
- dE/dl distribution for 0<vertex(z)<5
- Proton = 65% and Pion = 0%





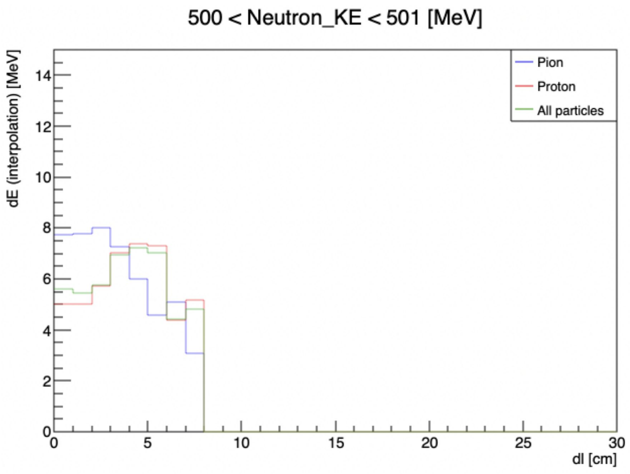
E(neutron)*C(12) = 150MeV*0.284 = 42.6 MeV

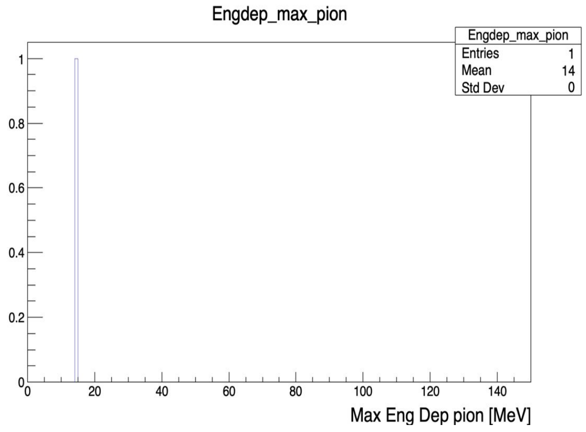
TABLE 9.4. Maximum Fraction of Energy Lost, Q_{max}/E_n from Eq. (9.3), by Neutron in Single Elastic Collision with Various Nuclei

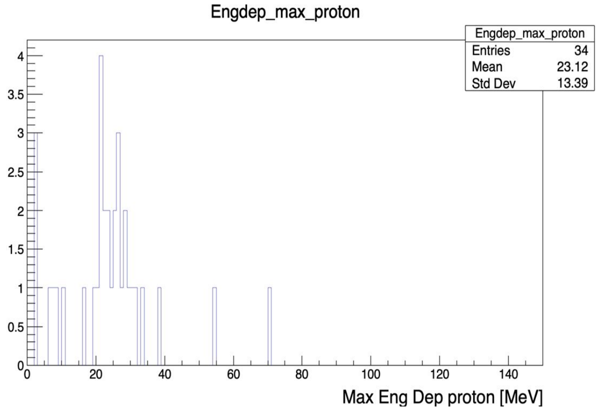
Nucleus	Q_{max}/E_n
$^{1}_{1}H$	1.000
$^{2}_{1}H$	0.889
$^{4}_{2}He$	0.640
$^{9}_{4}Be$	0.360
$^{12}_{6}C$	0.284
$^{16}_{8}O$	0.221
$^{56}_{26}Fe$	0.069
$^{118}_{50}Sn$	0.033
$^{238}_{92}U$	0.017

Neutron sample for PID application with SuperFGD prototype

- Beam placed at 90 meters from the detector
- Kinetic energy between 500-501 MeV
- dE/dl distribution for 0<vertex(z)<5
-



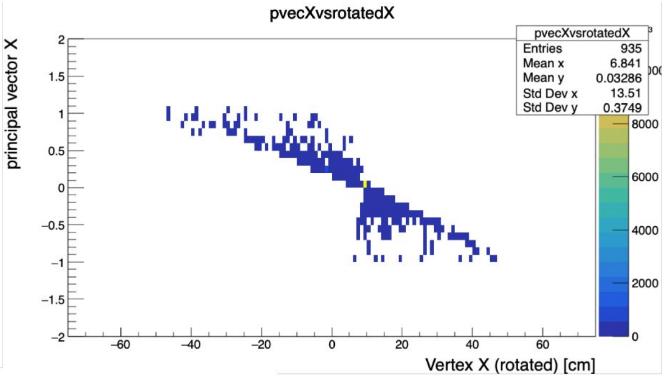


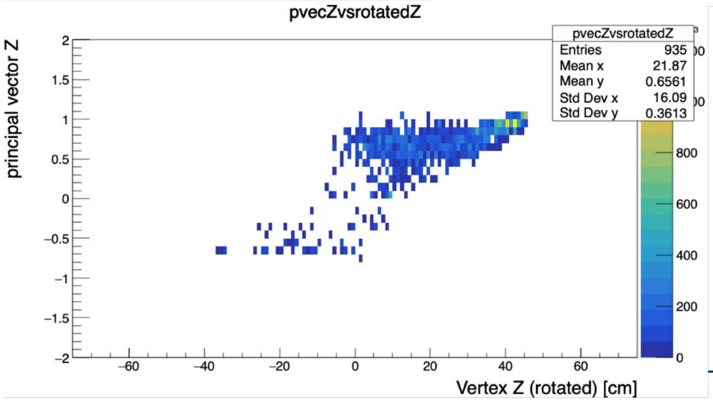


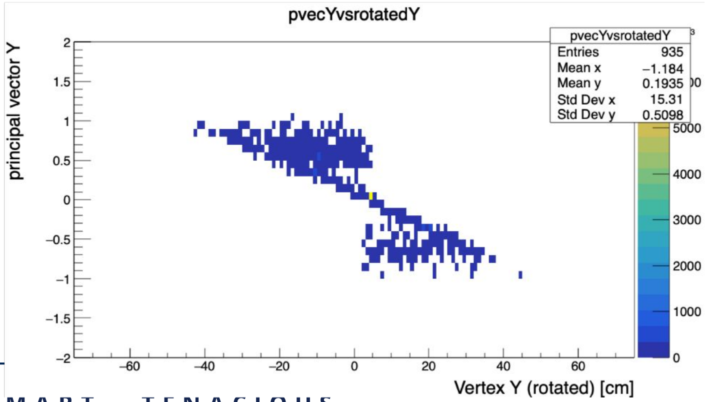
SOUTH DAKOTA MINES

Backup

Rotation


pvecXvsrotatedX
pvecXvsrotatedX
Entries 935
Mean x 6.841
Mean y 0.03286
Std Dev x 13.51
Std Dev y 0.3749
principal vector X
Vertex X (rotated) [cm]
8000
6000
4000
2000
0


pvecZvsrotatedZ
pvecZvsrotatedZ
Entries 935
Mean x 21.87
Mean y 0.6561
Std Dev x 16.09
Std Dev y 0.3613
principal vector Z
Vertex Z (rotated) [cm]
800
600
400
200
0


pvecYvsrotatedY
pvecYvsrotatedY
Entries 935
Mean x −1.184
Mean y 0.1935
Std Dev x 15.31
Std Dev y 0.5098
principal vector Y
Vertex Y (rotated) [cm]
5000
4000
3000
2000
1000
0

Rotation (angle correction function)

```
double calculateAngle(const std::vector<double> &vectorA, const std::vector<double> &vectorB)
{
    if (vectorA.size() != 3 || vectorB.size() != 3)
    {
        std::cout << "Error: Vectors must have three components." << std::endl;
        return 0.0;
    }

    double dotProduct = 0.0;
    double magnitudeA = 0.0;
    double magnitudeB = 0.0;

    for (int i = 0; i < 3; ++i)
    {
        dotProduct += vectorA[i] * vectorB[i]; // Calculate the dot product
        magnitudeA += vectorA[i] * vectorA[i]; // Sum the squares of components of vector A
        magnitudeB += vectorB[i] * vectorB[i]; // Sum the squares of components of vector B
    }

    magnitudeA = std::sqrt(magnitudeA); // Calculate the magnitude of vector A
    magnitudeB = std::sqrt(magnitudeB); // Calculate the magnitude of vector B

    double angle = std::atan2(magnitudeA * std::sin(std::acos(dotProduct / (magnitudeA * magnitudeB))), dotProduct); // Calculate the angle in radians

    return angle;
}
```

SOUTH DAKOTA MINES

Rotation