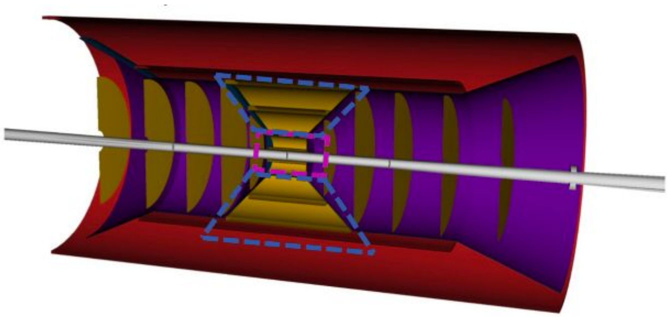
Issues with Noise Implementation in the ePIC Silicon Tracker Layers

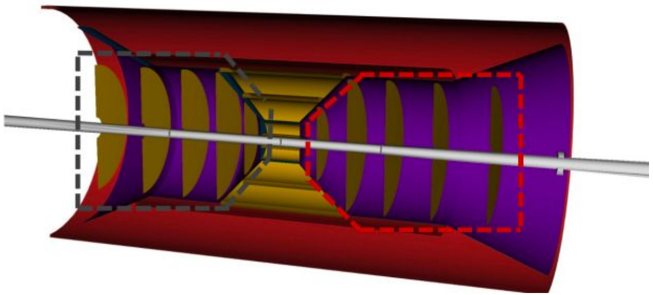
Current Estimation of Noise Hit Count

Sampled fake-hit rate: **FHR < 5 x 10^{-7} per event per pixel**
Fake hits/event/collection: **FHR x total pixels**

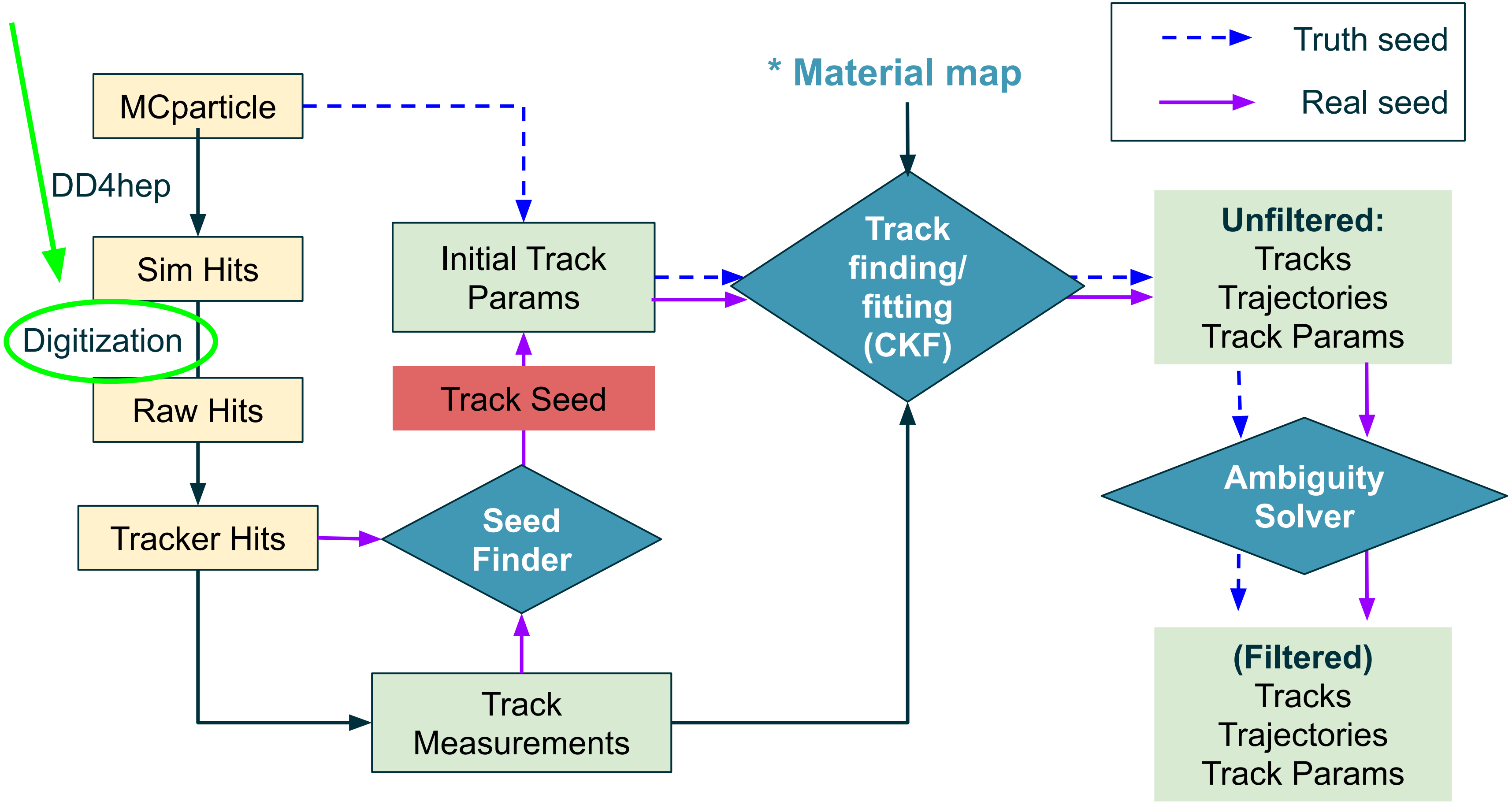
Pixels sizes: **20x20μm²**

	Inner Barrel	Outer Barrel	Endcaps
Est. total pixels	8.65E+08	7.83E+09	1.18E+10
Fake hits/event	4.33E+02	3.92E+03	5.91E+03





Noise Implementation



Step 1: Set up the workflow with pre-generated cell ID

1. SiliconTrackerDigi.cc

```
std::vector<std::uint64_t> disk_noisehits = {8615967849238777709,⋯
for (const auto& disk_noisehit : disk_noisehits) {⋯

std::vector<std::uint64_t> btrk_noisehits = {12628316263719227,⋯
for (const auto& btrk_noisehit : btrk_noisehits) {⋯

std::vector<std::uint64_t> bvtx_noisehits = {18301502869779853855,⋯
for (const auto& bvtx_noisehit : bvtx_noisehits) {⋯
```

2. SiliconTrackerDigiConfig.h

```
bool bvtxnoise = false;
bool btrknoise = false;
bool disknoise = false;
```

3. SiliconTrackerDigi_factory.h

```
ParameterRef<bool> m_bvtxnoise {this, "bvtxnoise", config().bvtxnoise};
ParameterRef<bool> m_btrknoise {this, "btrknoise", config().btrknoise};
ParameterRef<bool> m_disknoise {this, "disknoise", config().disknoise};
```

4. BVTX, BTRK, ECTRK.cc

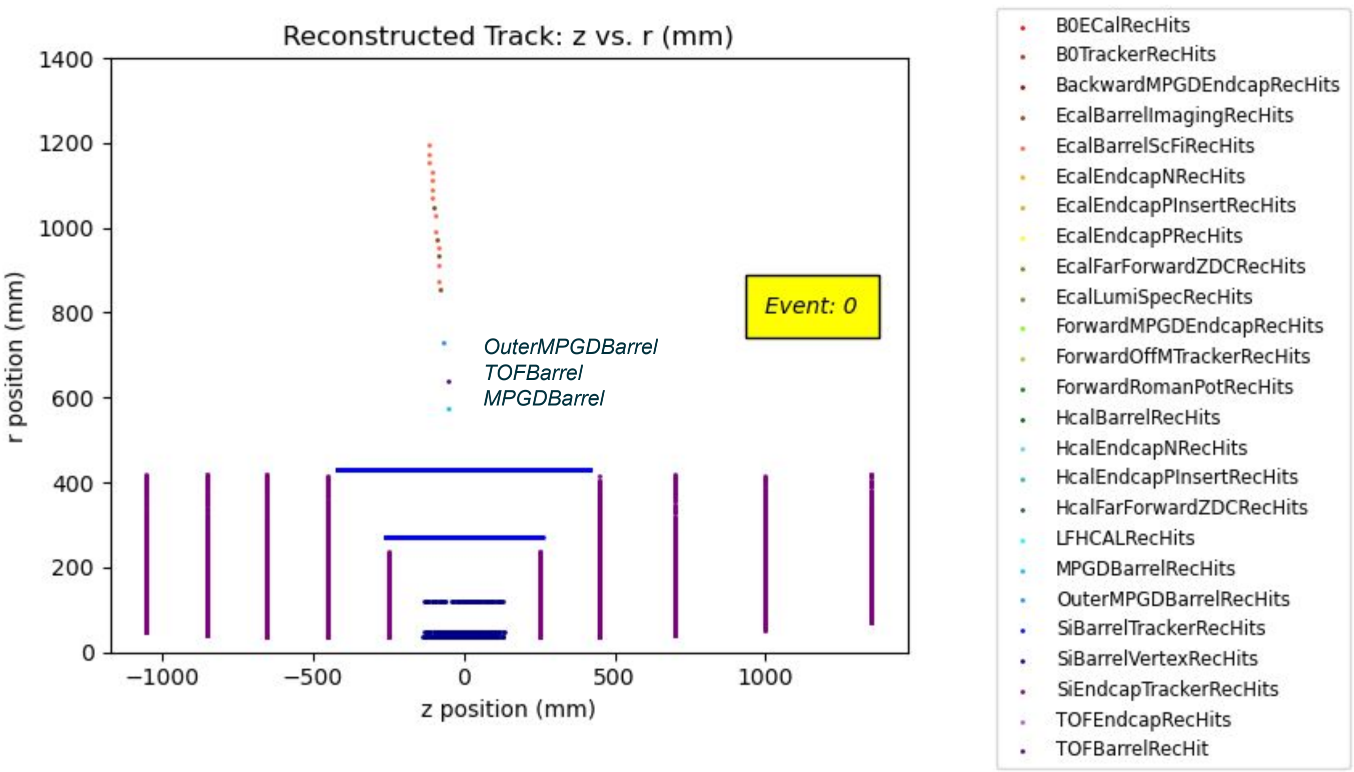
```
.btrknoise = true,
.bvtxnoise = true,
.disknoise = true,
```

Inside the for-loop:

1. Add noise cellID’s into the cellhit map for detectors with **noise==true** (BVTX, BTRK, ECTRK)
2. Define energy to be anything above the threshold
3. Define a random timestamp

```
for (const auto& bvtx_noisehit : bvtx_noisehits) {

        // time smearing
        double time_smearing = m_gauss();
        double result_time = 1 + time_smearing; //replaced sim_hit.getTime() with 1, a random number
        auto hit_time_stamp = (std::int32_t) (result_time * 1e3); //do I have to change the name of hit_time_stamp be
        //no, if defined under different blocks, they don't clash.

        if (m_cfg.bvtxnoise == true) { //add another if statement
        //std::cout << "it's a noise hit!" << std::endl;

            if (cell_hit_map.count(bvtx_noisehit) == 0) { //replaced sim_hit.getCellID() with noise_hits[noise_hit] to
                // This cell doesn't have hits
                cell_hit_map[bvtx_noisehit] = {
                    bvtx_noisehit,
                    (std::int32_t) std::llround(5.4 * 1e6), //*1e6 to convert from KeV it to be GeV
                    hit_time_stamp  // ns->ps
                };
            } else {
                // There is previous values in the cell
                auto& hit = cell_hit_map[bvtx_noisehit];
                debug("  Hit already exists in cell ID={}, prev. hit time: {}", bvtx_noisehit, hit.getTimeStamp());

                // keep earliest time for hit
                auto time_stamp = hit.getTimeStamp(); //where is getTimeStamp defined?
                hit.setTimeStamp(std::min(hit_time_stamp, hit.getTimeStamp()));

                // sum deposited energy
                auto charge = hit.getCharge();
                hit.setCharge(charge + (std::int32_t) std::llround(5.4 * 1e6));
            }
        }
}
```

Successful reconstruction of manually inputted cellID's



Description:
Reconstructed hit position plot for one single-muon event. Includes manually inputted SVT hits, whose cellID's were copied from a separate reconstruction file.

Step 2: Generate random 64-bit Cell ID’s efficiently

Inner Barrels: system:8,layer:4,module:12,sensor:2,x:32:-16,y:-16

Outer Barrels: system:8,layer:4,module:12,sensor:2,x:32:-12,y:-20

Endcap disks: system:8,layer:4,module:12,sensor:2,x:32:-16,z:-16

(from the detector xml files)

Collect. name	sys_id	layer	module	sensor	*seg_x	*seg_y/z	# est. noise hits
IB	31	1,2,4	1-128	1	***~146*	*~13500*	433
OB	59, 60	1	1-44, 1-69	1	*ongoing study*	*ongoing study*	3920
Disks	68-70, 77-79	1	1-36	1	*ongoing study*	*ongoing study*	5910

*ATM, we know the segment ranges for IB. Included are number of pixels
**stave width translates to at most 146 pixels (depends on layer), and they are stored in 32-bits

Current Question:

We are trying to find an efficient way to provide these information to the digi file and generate noise hits that are **uniformly distributed** across the SVT layers. For loop is possible, but a more automated process is desirable.

Incorporating noise hits is also important for other detectors (i.e. calorimeters), so a unified solution in that sense would be ideal.

Noise Folder in DD4hep

Link:https://github.com/eic/DD4hep/tree/4234279b462b521d215ace43c36b920e8c1fe719/DDDigi/src/noise

Git Location: DD4hep/DDDigi/src/noise

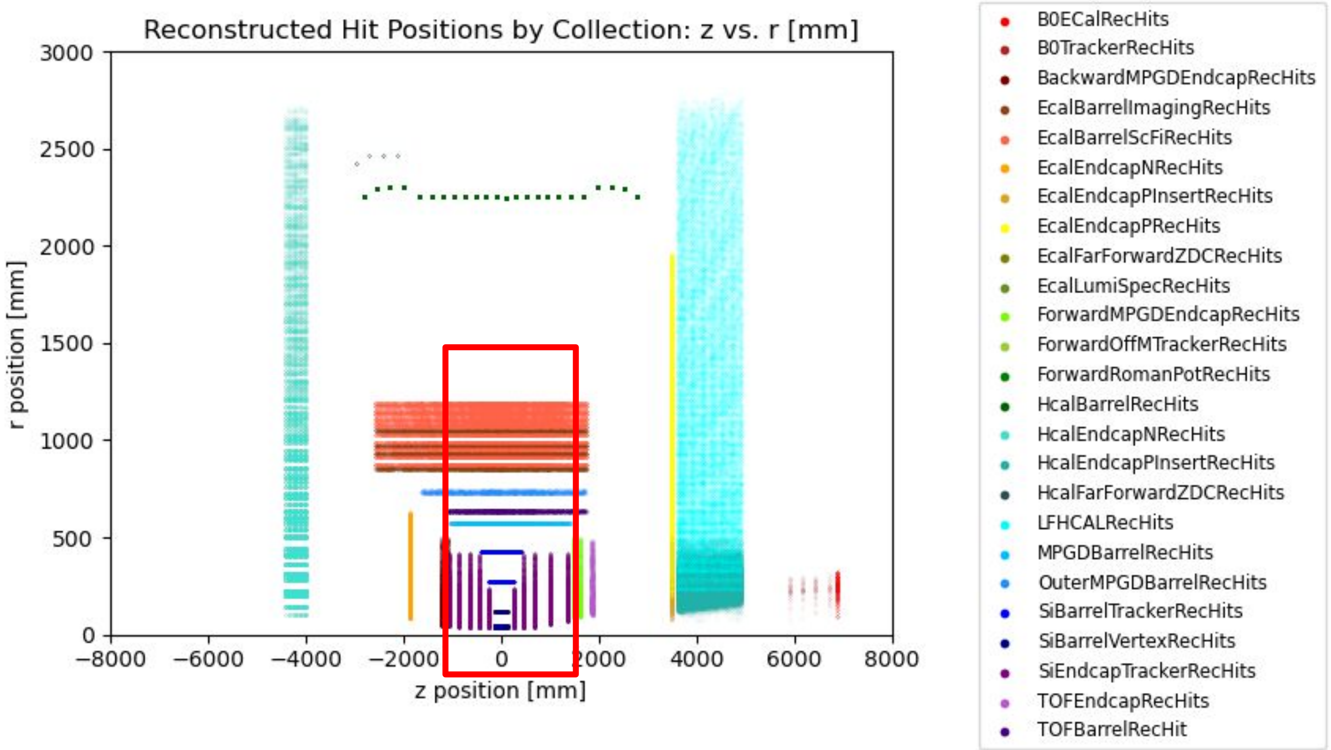
- DigiExponentialNoise.cpp
- DigiGaussianNoise.cpp
- DigiLandauNoise.cpp
- DigiPoissonNoise.cpp
- DigiRandomNoise.cpp
- DigiSignalProcessorSequence.cpp
- DigiSubdetectorSequence.cpp
- DigiUniformNoise.cpp
- FalphaNoise.cpp

```
// Framework include files
#include <DD4hep/InstanceCount.h>
#include <DDDigi/DigiRandomGenerator.h>
#include <DDDigi/noise/DigiUniformNoise.h>

using namespace dd4hep::digi;

/// Standard constructor
DigiUniformNoise::DigiUniformNoise(const DigiKernel& krnl, const std::string& nam)
  : DigiSignalProcessor(krnl, nam)
{
  declareProperty("minimum", m_min);
  declareProperty("maximum", m_max);
  InstanceCount::increment(this);
}

/// Default destructor
DigiUniformNoise::~DigiUniformNoise() {
  InstanceCount::decrement(this);
}

/// Callback to read event uniformnoise
double DigiUniformNoise::operator()(DigiCellContext& context)  const  {
  return context.context.randomGenerator().uniform(m_min,m_max);
}
```

Backup

Full 10k Reconstructed Events

*different xy range and color coding as the slide before



Step 2: Generate random 64-bit Cell ID’s efficiently

Inner Barrels: system:8,layer:4,module:12,sensor:2,x:32:-16,y:-16

Outer Barrels: system:8,layer:4,module:12,sensor:2,x:32:-12,y:-20

Endcap disks: system:8,layer:4,module:12,sensor:2,x:32:-16,z:-16

(from the detector xml files)

Collect. name	sys_id	layer	module	sensor	seg_x (L0,1,2,3,4)	seg_y/z	# est. noise hits
IB	31	1,2,4	1-128	1	**(0,43) or (65492, 65535)** **(0,57) or (65478, 65535)** **(0,146) or (65386, 65535)**	(0,6749) or (58786,65535)	433
OB	59, 60	1	1-44, 1-69	1	*ongoing study*	*ongoing study*	3920
Disks	68-70, 77-79	1	1-36	1	*ongoing study*	*ongoing study*	5910

Segment_x Calculations

Credit: Barak Schmookler

1. Calculate # pixels along xy of staves by dividing widths of staves by 20um

```
VertexBarrelStave1_width = 2*(36*1000 um)*tan(180 * (pi/180) / 128) = 1767.500791842632 um   = 88.375 pixels
VertexBarrelStave2_width = 2*(48*1000 um)*tan(180 * (pi/180) / 128) = 2356.6677224568425 um  = 117.833 pixels
VertexBarrelStave2_width = 2*(120*1000 um)*tan(180 * (pi/180) / 128) = 5891.6693061421065 um = 294.58 pixels
```

2. Divide (1) by 2 because starting segmentation value jump from 0 to 65535 (smallest->largest 64bit #) at middle of each stave

Segment_y Calculations

```
<constant name="VertexBarrel_length"                    value="270.0*mm"/>
```

1. Calculate # pixels along z: 270000um/20um = 13500 pixels
2. Divide (2) by 2 because segmentation values jump because starting segmentation value jump from 0 to 65535 (smallest->largest 64bit #) at z=0

Comparing segment y vs. global z

```
---------------------
Z bit value = 0
Current cell ID = 17289503
Current cell Position : {x , y , z} = {35.308270 , -7.023252 , 0.000000} mm
Current cell Position : {r , z} = {36.000000 , 0.000000} mm
Current cell ID converted from Position = 17289503
---------------------

Warning in <TGeoMatrix::dtor>: Registered matrix component0_placement was removed
---------------------
Z bit value = 1
Current cell ID = 281474994000159
Current cell Position : {x , y , z} = {35.308270 , -7.023252 , -0.020000} mm
Current cell Position : {r , z} = {36.000000 , -0.020000} mm
Current cell ID converted from Position = 281474994000159
---------------------
```

Credit: Barak Schmookler

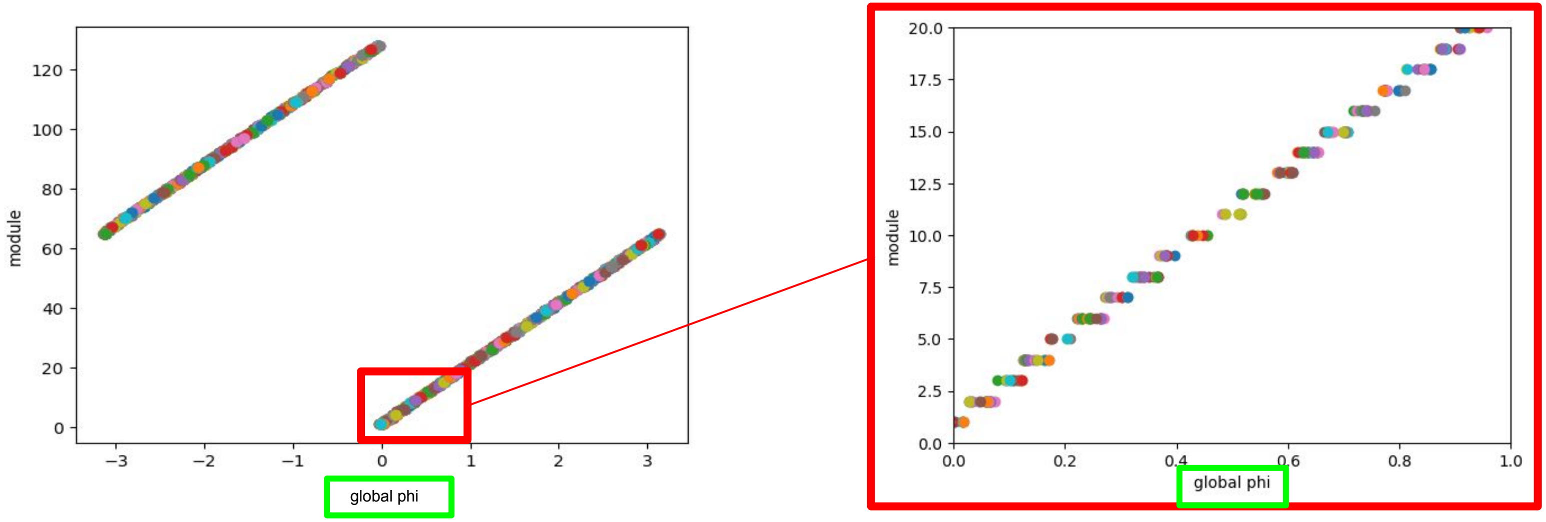
Comparing segment y vs. global z

```
Warning in <TGeoMatrix::dtor>: Registered matrix component0_placement was removed
---------------------
Z bit value = 65534
Current cell ID = 18446181123773419807
Current cell Position : {x , y , z} = {35.308270 , -7.023252 , 0.040000} mm
Current cell Position : {r , z} = {36.000000 , 0.040000} mm
Current cell ID converted from Position = 18446181123773419807
---------------------

Warning in <TGeoMatrix::dtor>: Registered matrix component0_placement was removed
---------------------
Z bit value = 65535
Current cell ID = 18446462598750130463
Current cell Position : {x , y , z} = {35.308270 , -7.023252  0.020000} mm
Current cell Position : {r , z} = {36.000000 , 0.020000} mm
Current cell ID converted from Position = 18446462598750130463
```

Credit: Barak Schmookler

Example Plots for Inner Barrels:

global phi vs. module

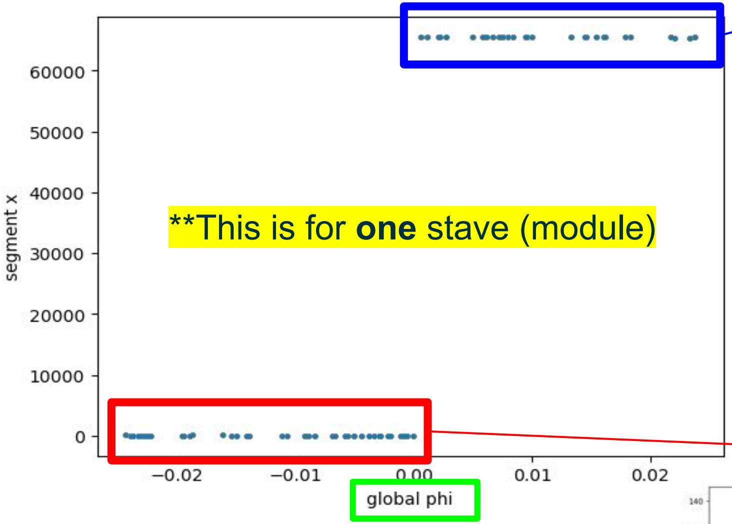


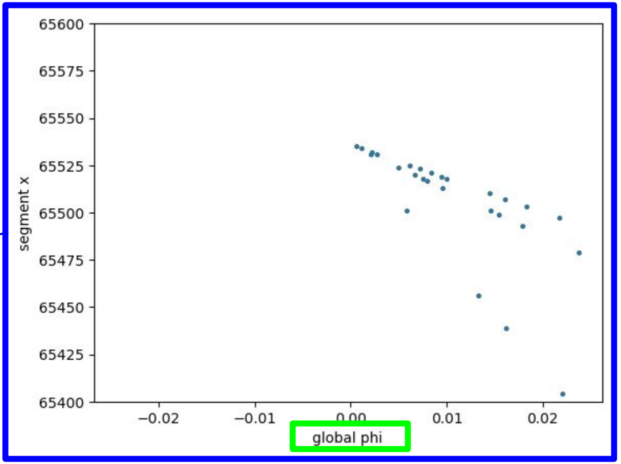
global_phi = np.arctan2(y,x)

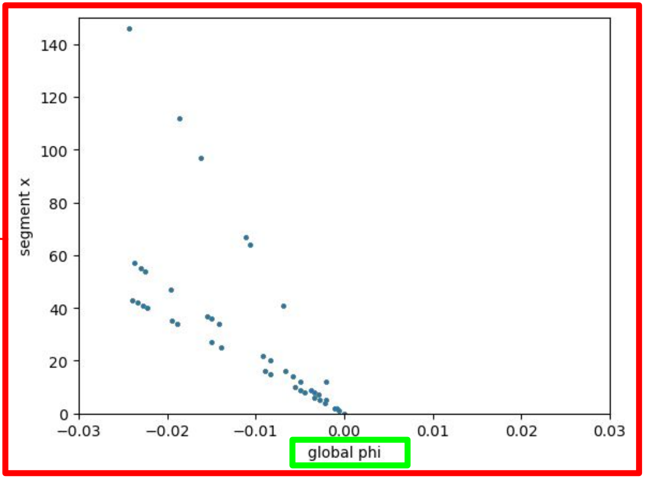
- y=SiBarrelVertxRecHits.position.y
- x= SiBarrelVertxRecHits.position.x

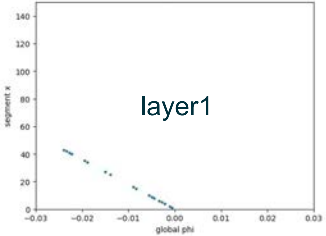
**Each point are actual hits from single-particle simulation rather than noise hits. The values on the y-axis of both plots are extracted from bitwise operations on the cellID variable, and the x-axis as defined in the green box. For the sake of runtime, plots from here include 2000 random events from the 10000 generated events.*

Example Plots for Inner Barrels:
global phi vs. segment x





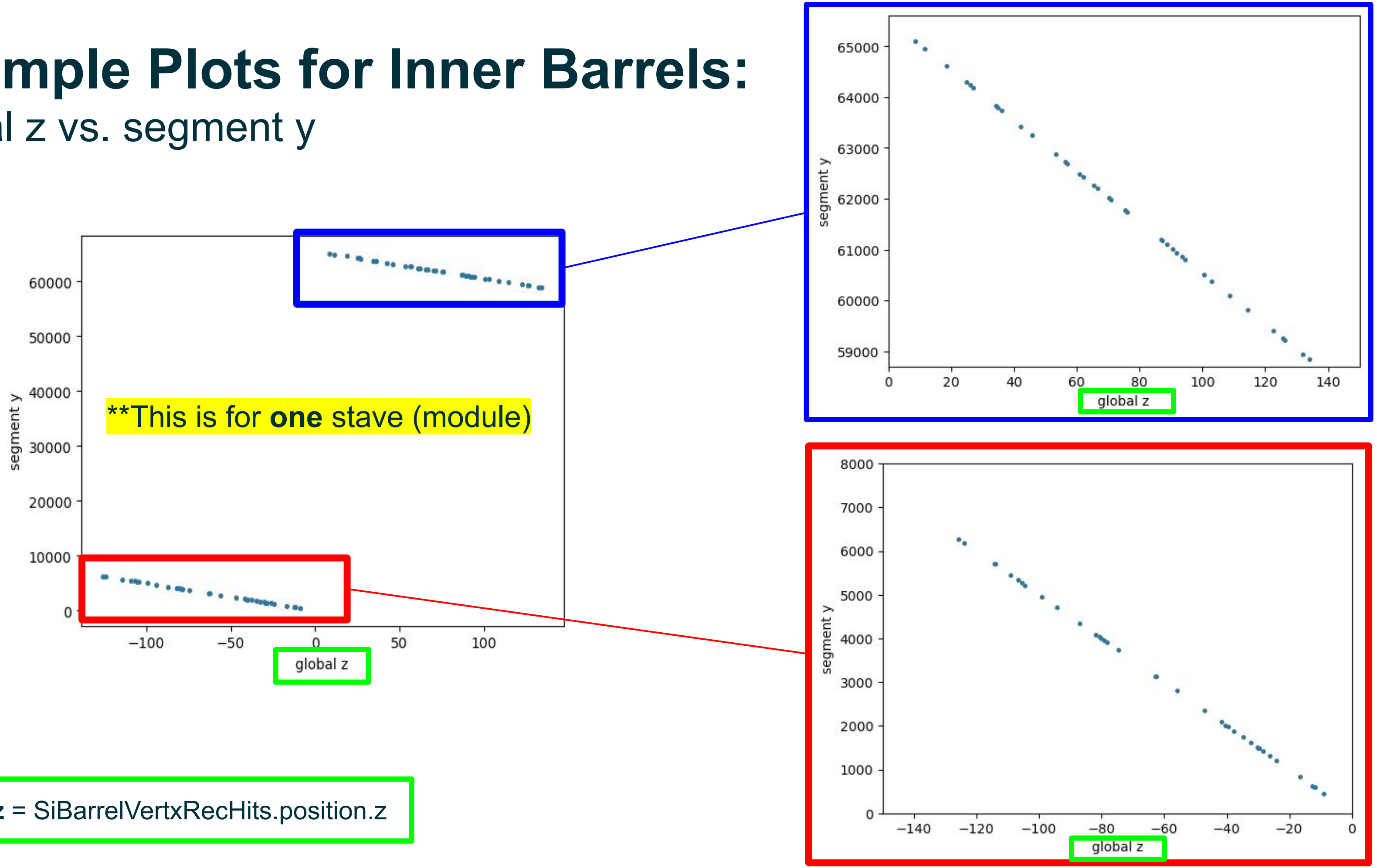




global_phi = np.arctan2(y,x)

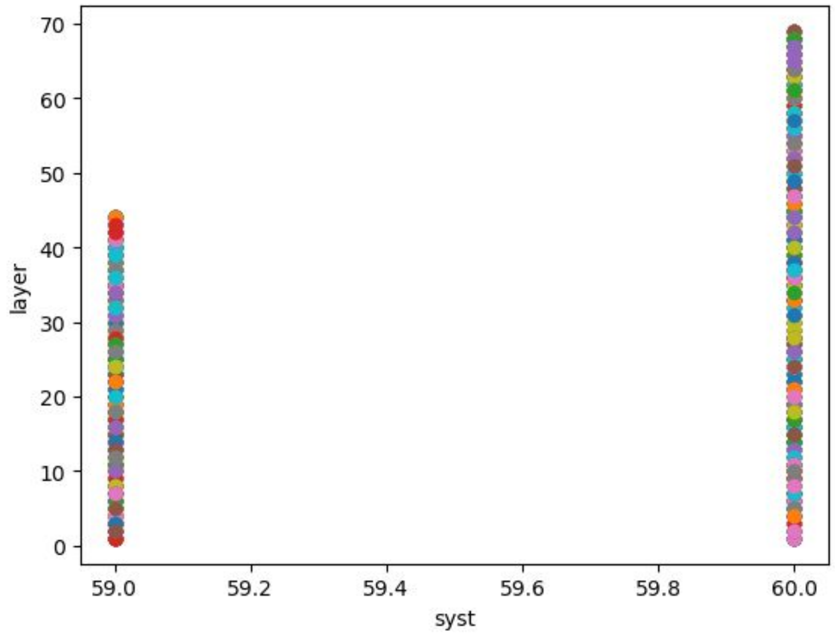
- y=SiBarrelVertxRecHits.position.y
- x= SiBarrelVertxRecHits.position.x

Example Plots for Inner Barrels:
global z vs. segment y

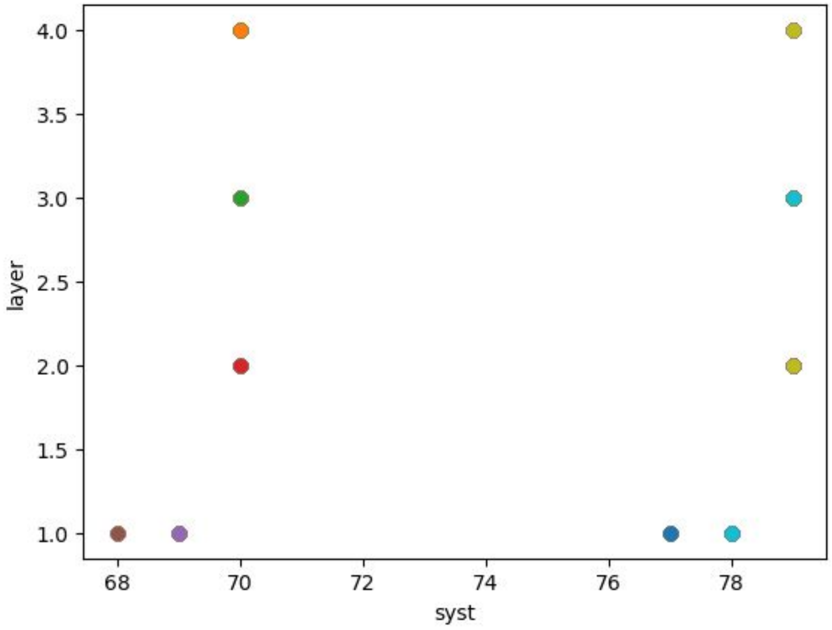


global_z = SiBarrelVertxRecHits.position.z

Example Plots for Outer Barrels and Endcaps (syst vs. layer)



Outer Barrels



End Caps