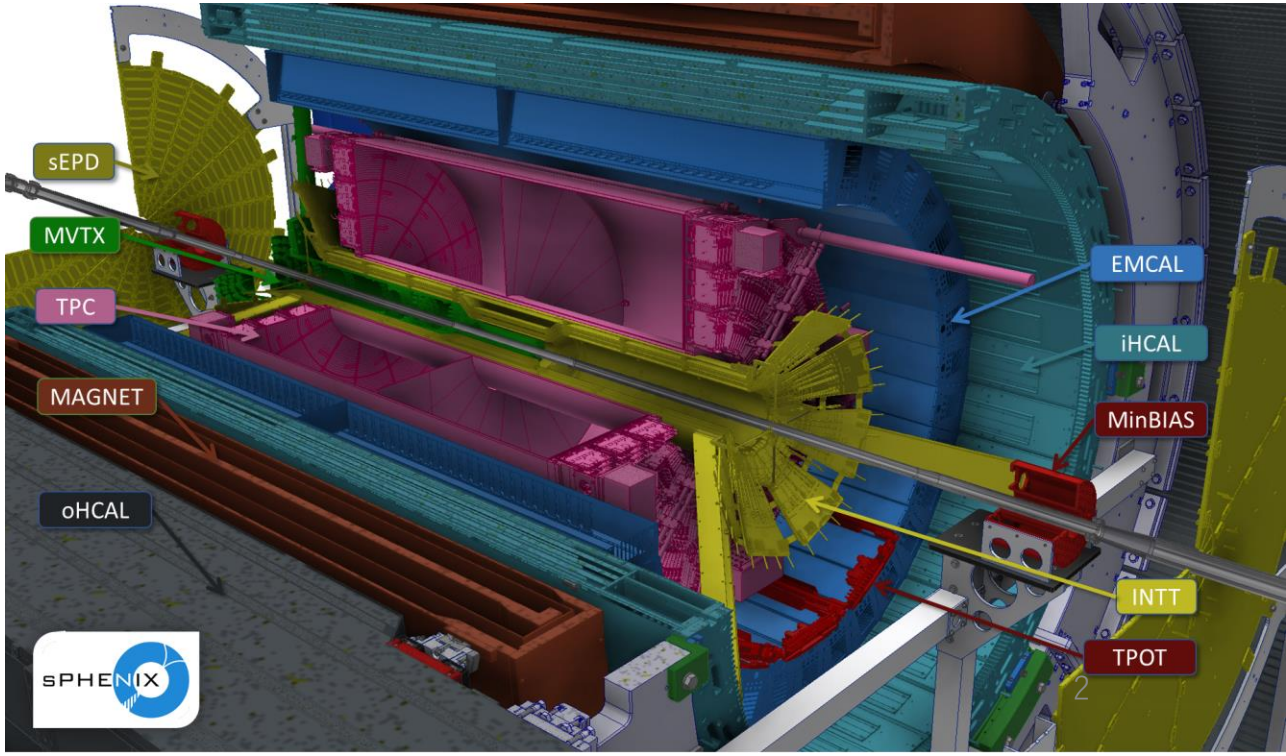
1/15　立教ワークショップ

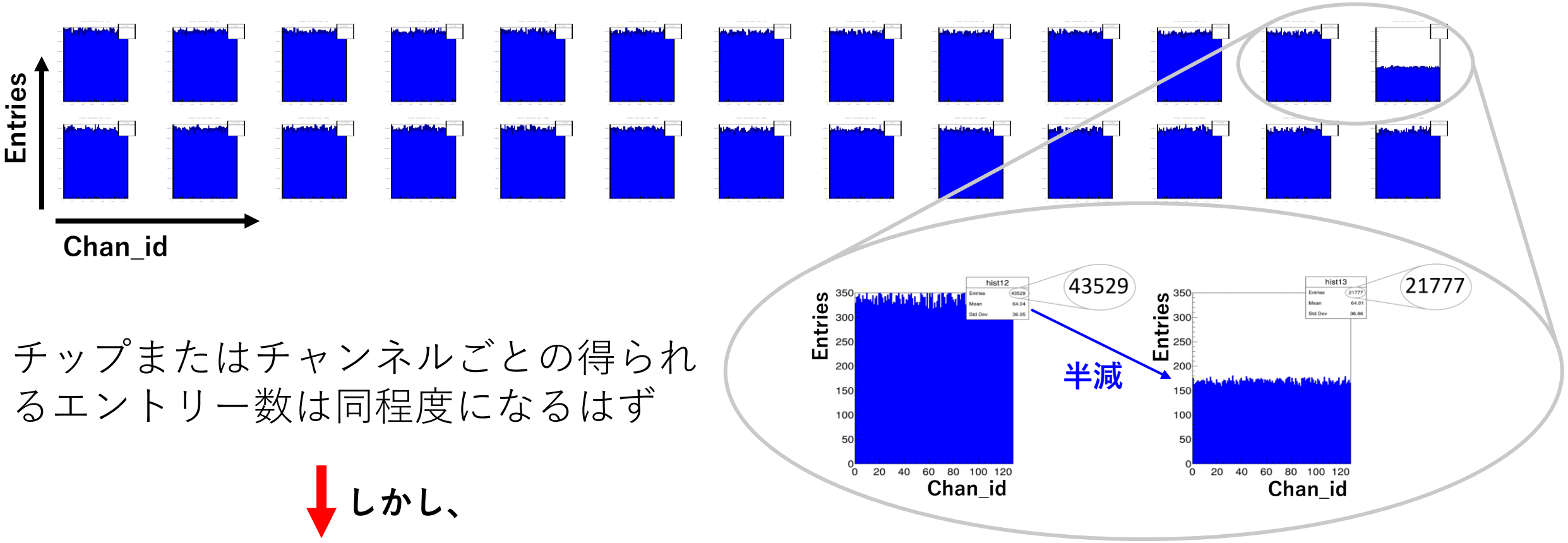
立教大学4年　原田

概要

1. 卒論テーマについて（ハーフエントリー問題）
2. 理研でのノイズデータの解析
3. INTT実機のノイズデータの解析
4. ハーフエントリーチップの特定



1. 卒論テーマについて（ハーフエントリー問題）



チップまたはチャンネルごとの得られるエントリー数は同程度になるはず

しかし、

正常なチップに対して、エントリー数が半減してしまうようなチップが存在した。
これをハーフエントリー問題という。

1. 卒論テーマについて（ハーフエントリー問題）

転送方法に着目

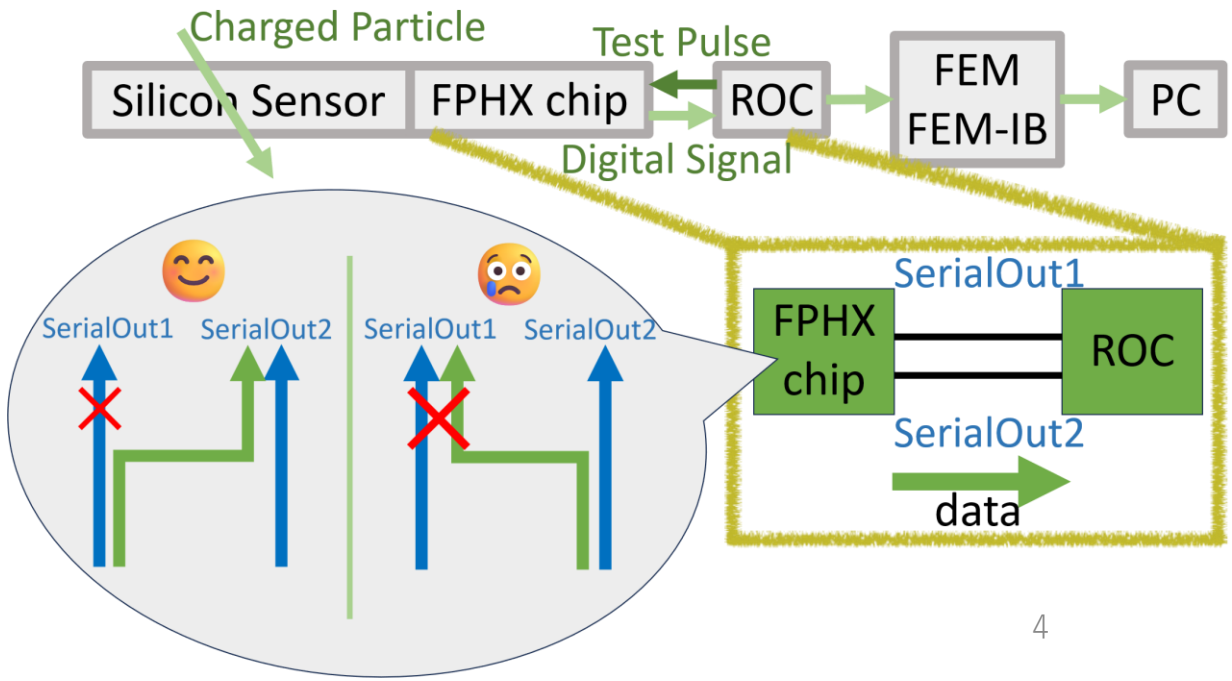
- FPHX-chipから後段のROCまでは2本のデータ転送ラインがある

仮説

- FPHX-chipには、実装された稼働モードをパラメータ制御するDigital Controlという機能があり、それによって失われていたデータを転送できる

検証結果

- 理研に存在するハーフエントリーを持つチップのエントリー数の復活が見られた



1. 卒論テーマについて（ハーフエントリー問題）

ハーフエントリーチップ表

Felix Server	0	0	3	5	7	7
pid	3001	3001	3004	3006	3008	3008
module	6	7	13	3	0	1
chip id	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	15	21, 23, 25	16	1, 2, 3, 4, 5, 7, 9, 11, 13, 14, 15, 16, 17, 18, 20, 22, 24, 26	1, 2

$38\ \text{chips}/2912\ \text{chips} = \mathbf{1.3}\%$

1. 卒論テーマについて（ハーフエントリー問題）

Date/Time	Run#	Run Type	Mag	Link	Active Felix List	DAC 0	L1 Delay	n_coll	open time	Modebit	Comments
2024/10/22 14:40	55060	pedestal	Off		All Felix	12	114	100	60	intt_streamingmoc	Digicon=7 DAC = 8,10,15,30,35,50,100,210 (40 min)
2024/10/22 15:25	55062	pedestal	Off		All Felix	12	114	100	60	intt_streamingmoc	Digicon=2 DAC=8,10,15,30,35,50,100,210 (40 min)
2024/10/22/ 16:02	55064	pedestal	Off		All Felix	12	114	100	60	intt_streamingmoc	Digicon=10 DAC=8,10,15,30,35,50,100,210 (40 min)

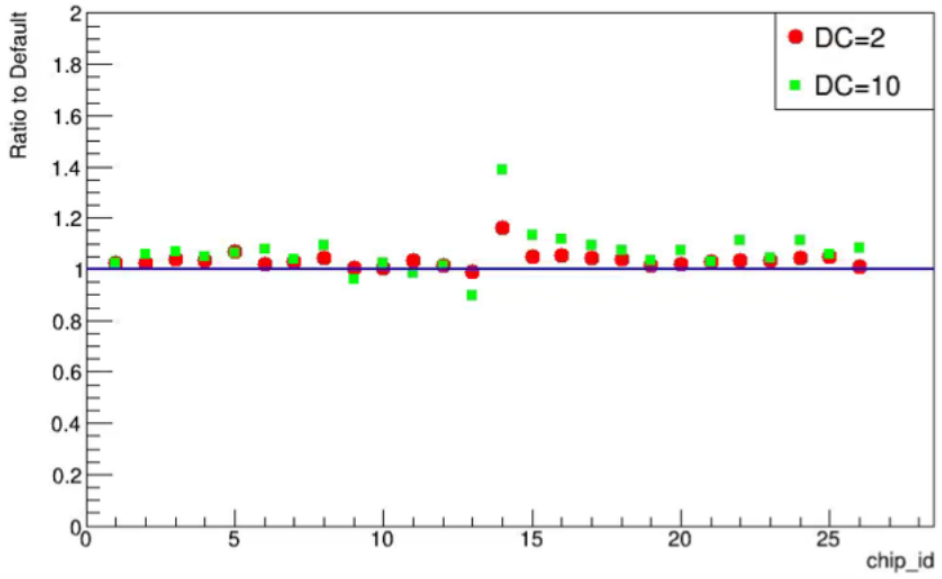


図 46: チップ番号に対するデフォルトとのエントリー数の比

(a) pid=3008、module=0

(b) DC=2(赤)、DC=10(緑)

表 4: ハーフエントリーチップ表

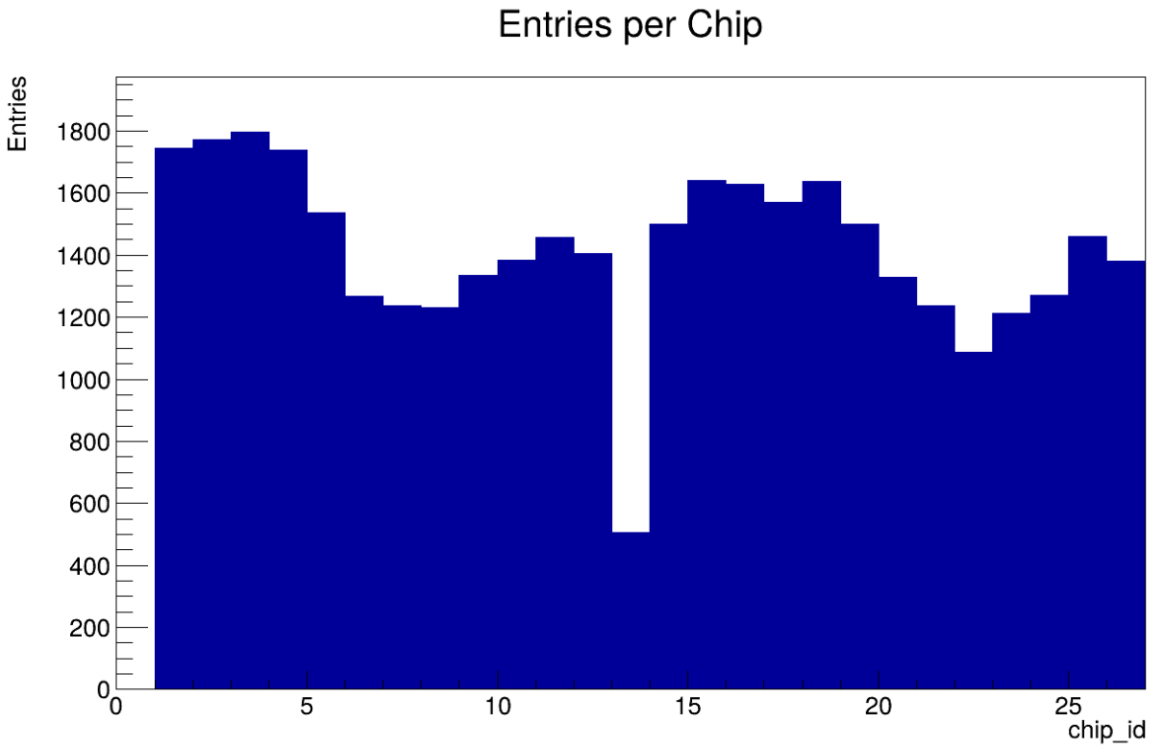
(a) 図 46 に含まれるハーフエントリーチップを表示

Felix Server	7
pid	3008
module	0
chip id	1, 2, 3, 4, 5, 7, 9, 11, 13, 14, 15, 16, 17, 18, 20, 22, 24, 26

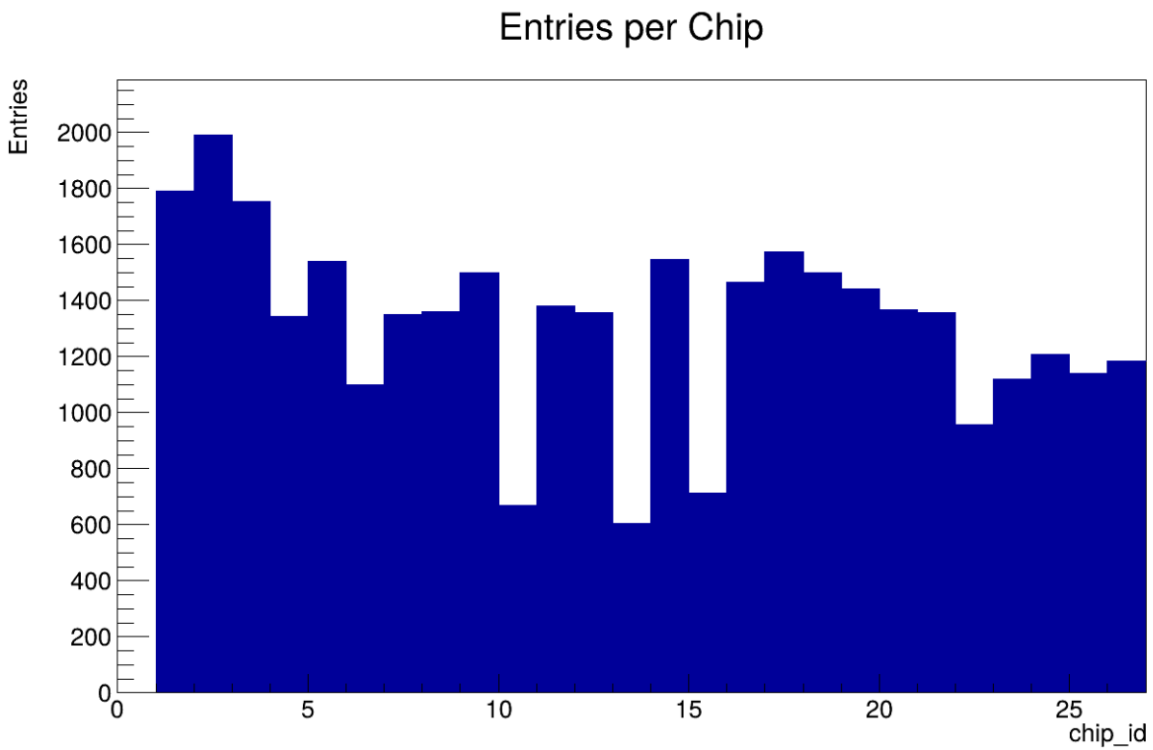
エントリー数の復活は見られなかった

2. 理研でのノイズデータの解析

宇宙線測定（低レート）



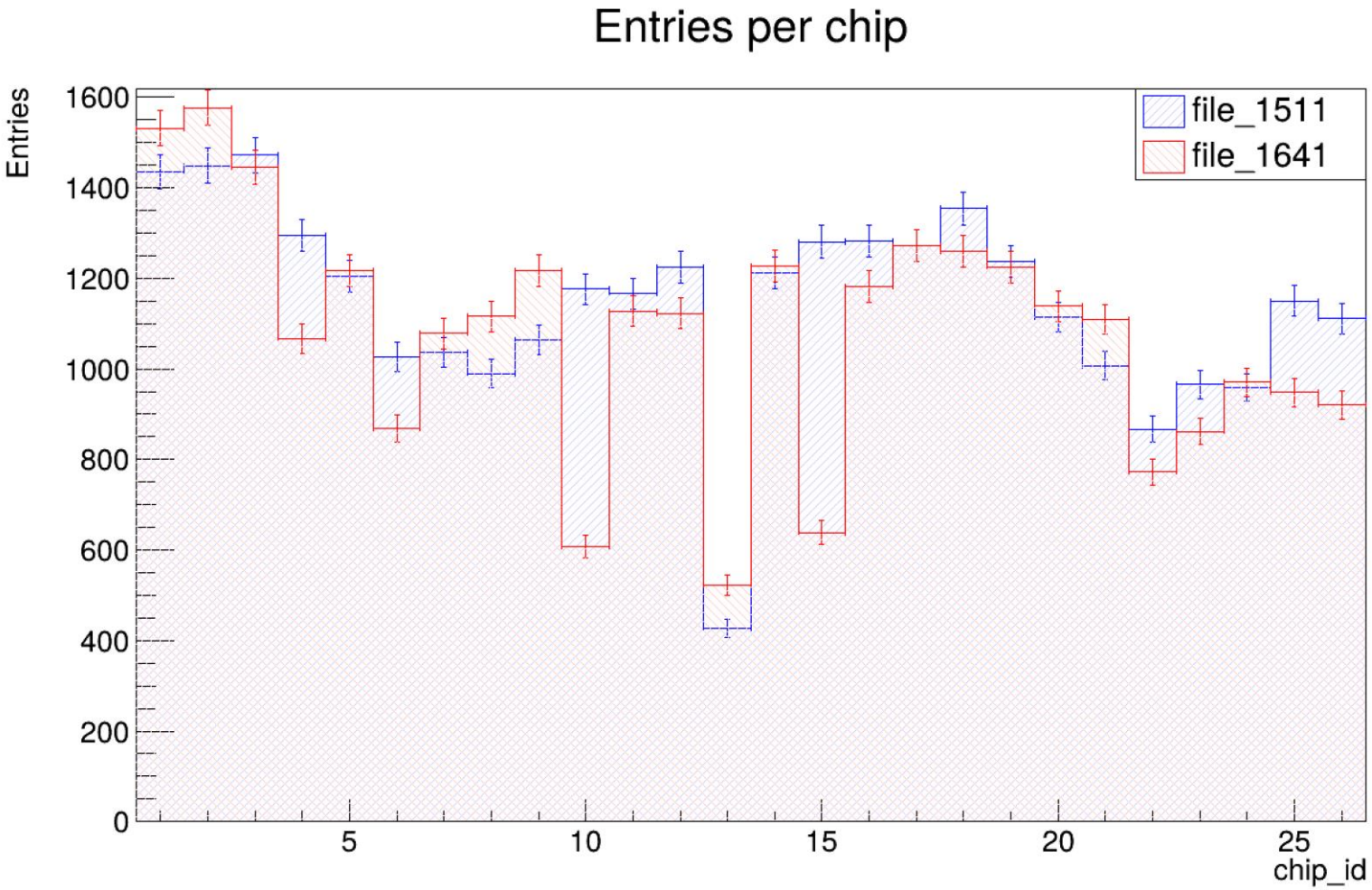
chip_id=13はハーフエントリー



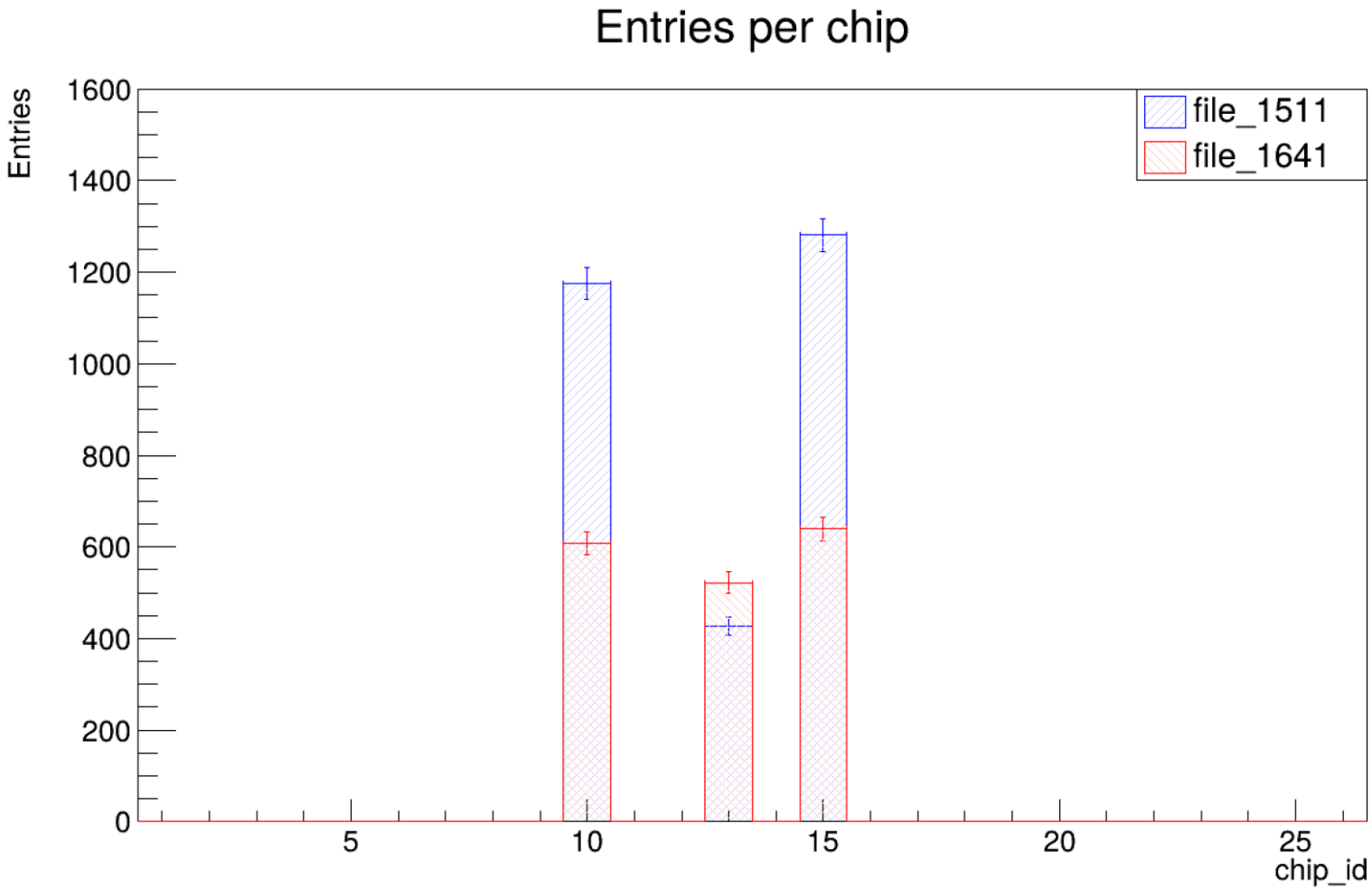
chip_id=10, 15を疑似的にハーフエントリーにしている

重ねると、

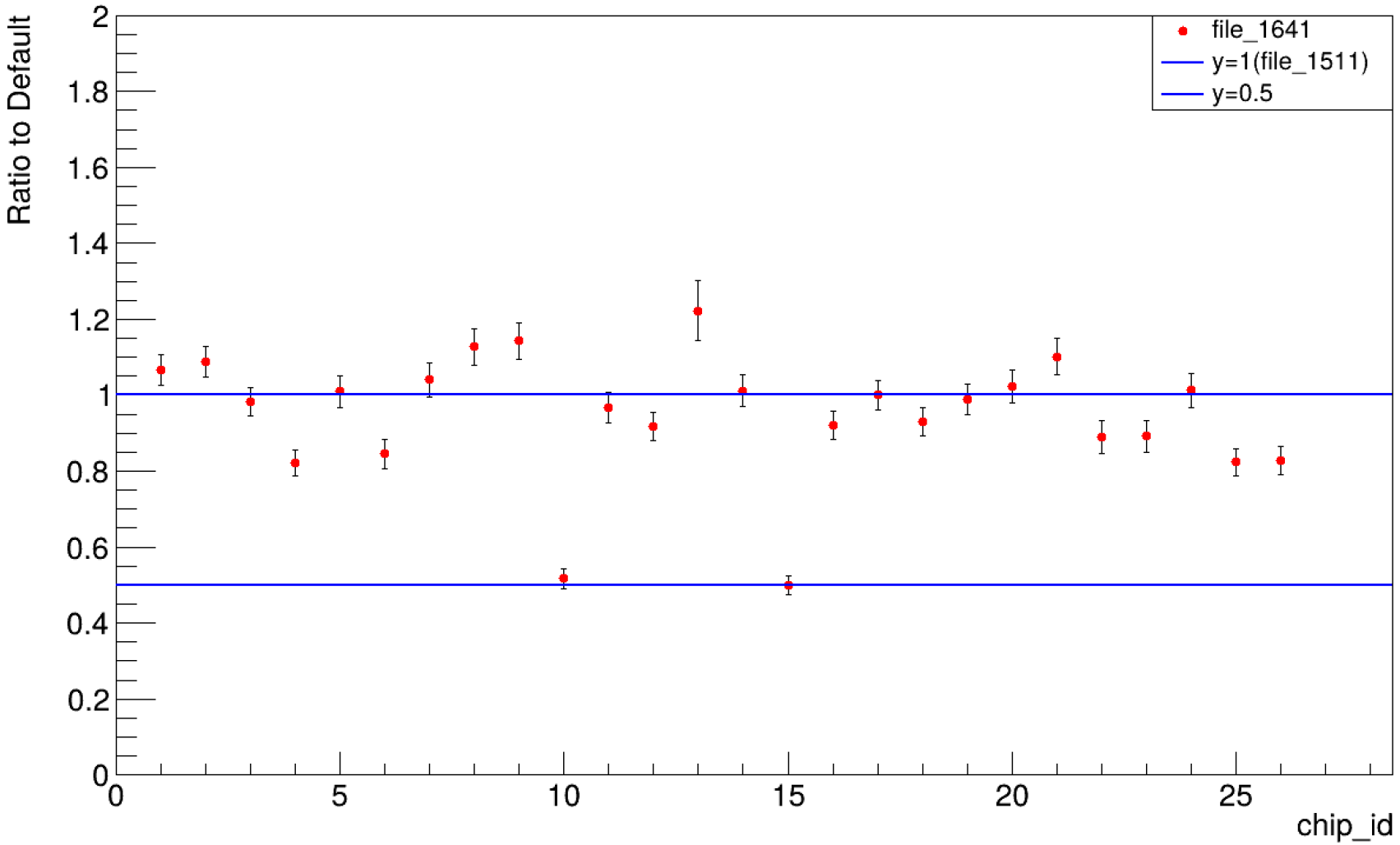
2. 理研でのノイズデータの解析



2. 理研でのノイズデータの解析



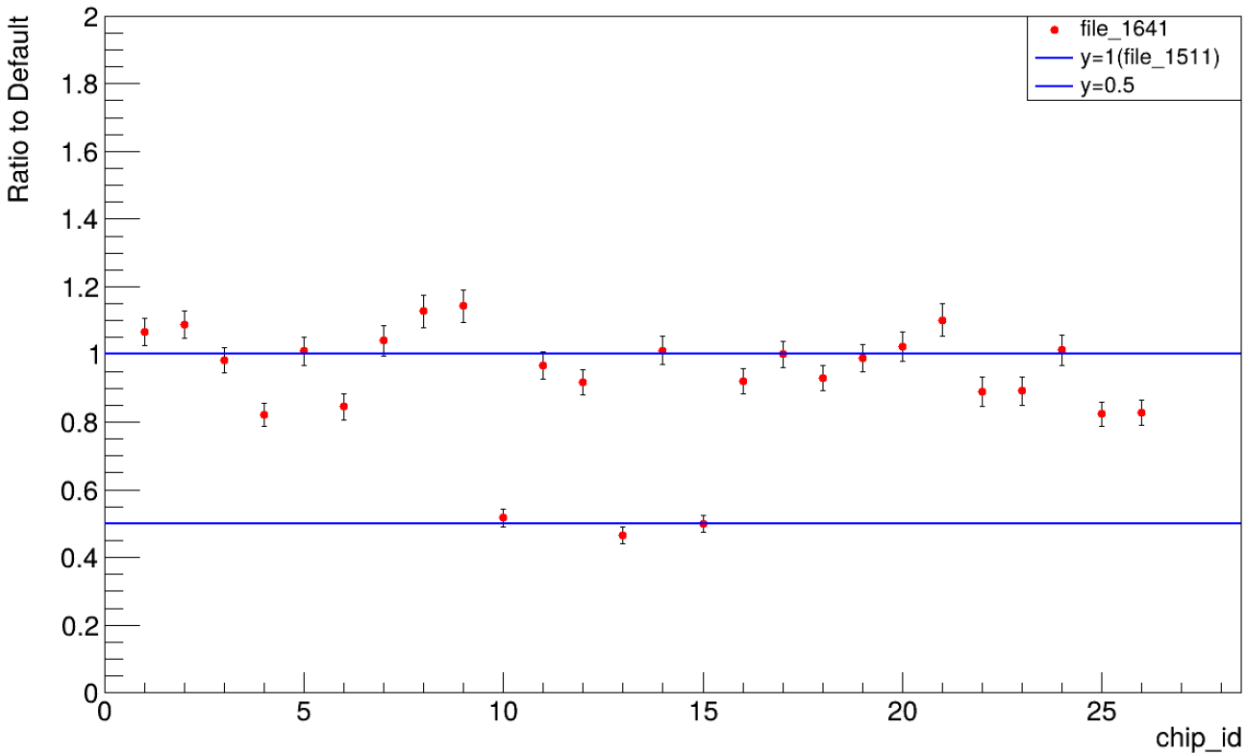
2. 理研でのノイズデータの解析



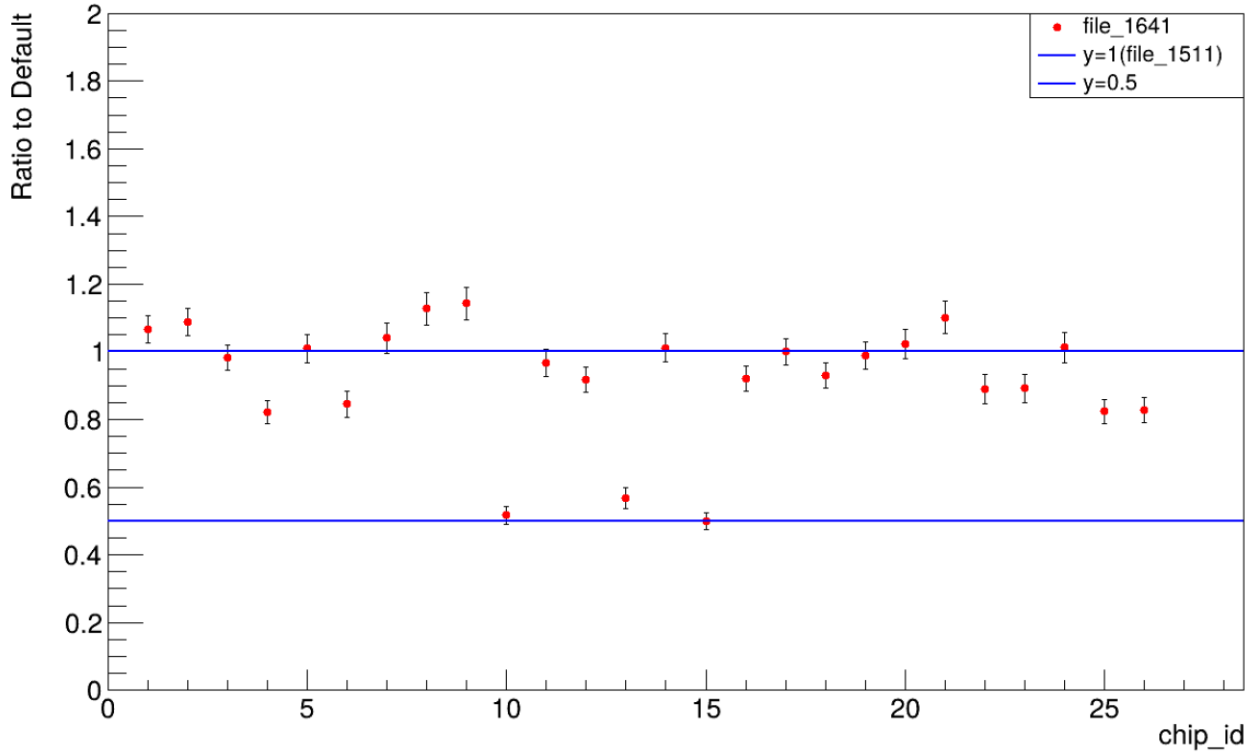
chip_id = 10 : $\mathbf{0.52 \pm 0.03}$

chip_id = 15 : $\mathbf{0.50 \pm 0.02}$

2. 理研でのノイズデータの解析

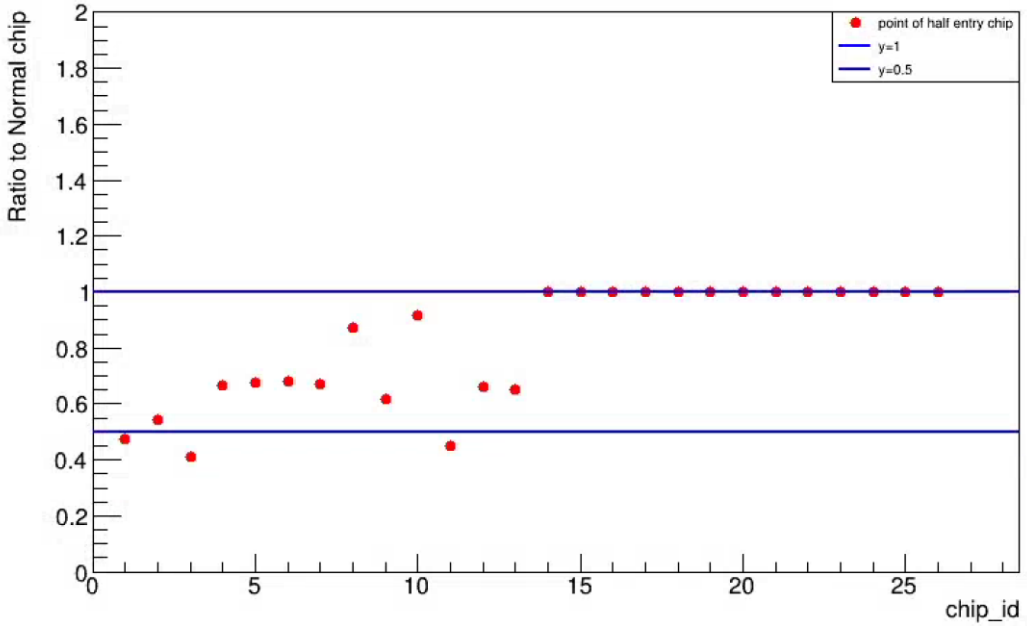


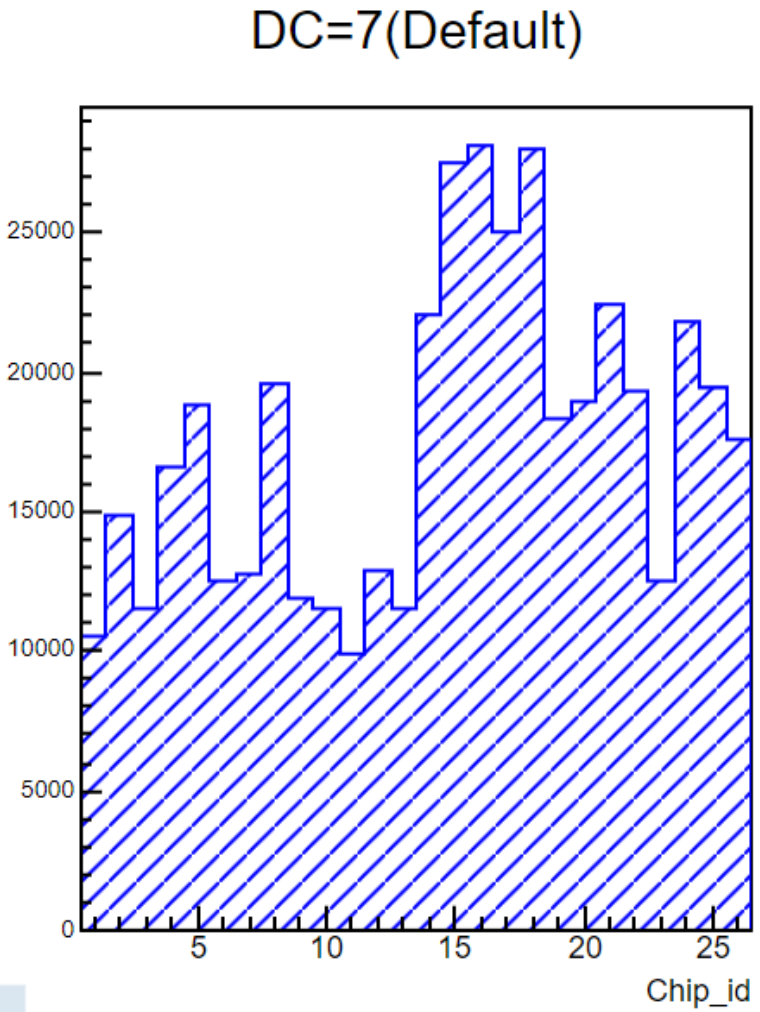
chip_id = 13 : $\mathbf{0.46 \pm 0.02}$
chip_id = 12と比較



chip_id = 13 : $\mathbf{0.57 \pm 0.03}$
chip_id = 12と比較

3. INTT実機のノイズデータの解析



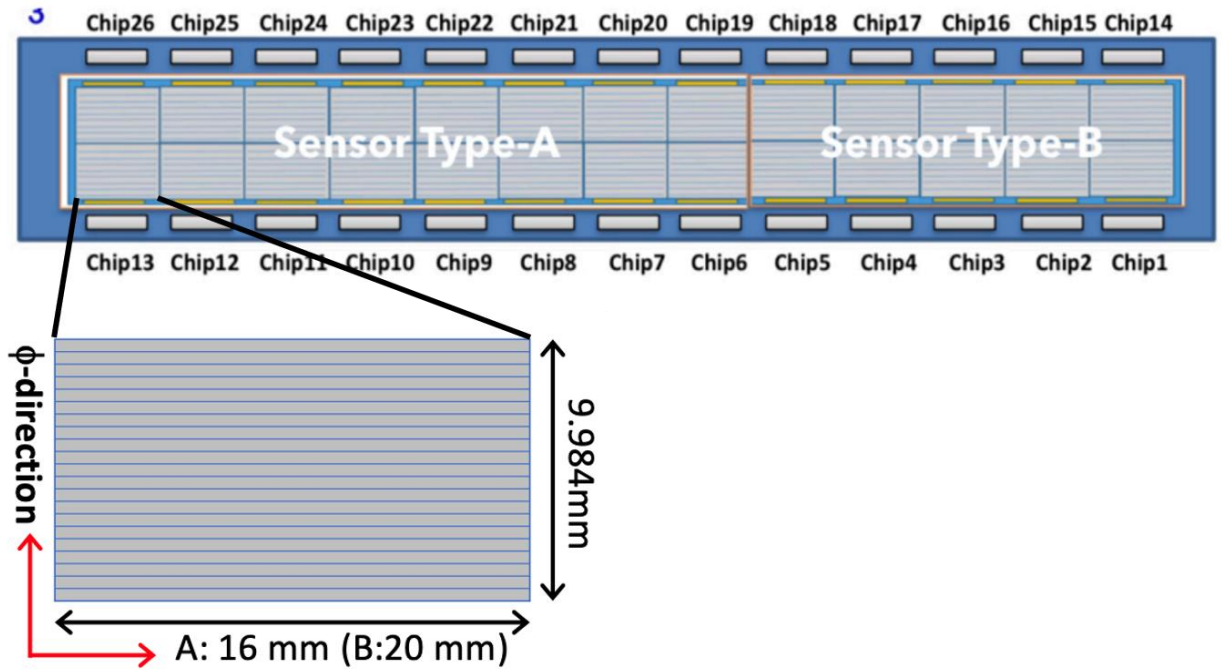


INTT	0
pid	3001
module	6
chip id	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13

確実にハーフエントリーが見えるとは言えない。
→もう少しほかのランのデータを見てみる

4. ハーフエントリーチップの特定

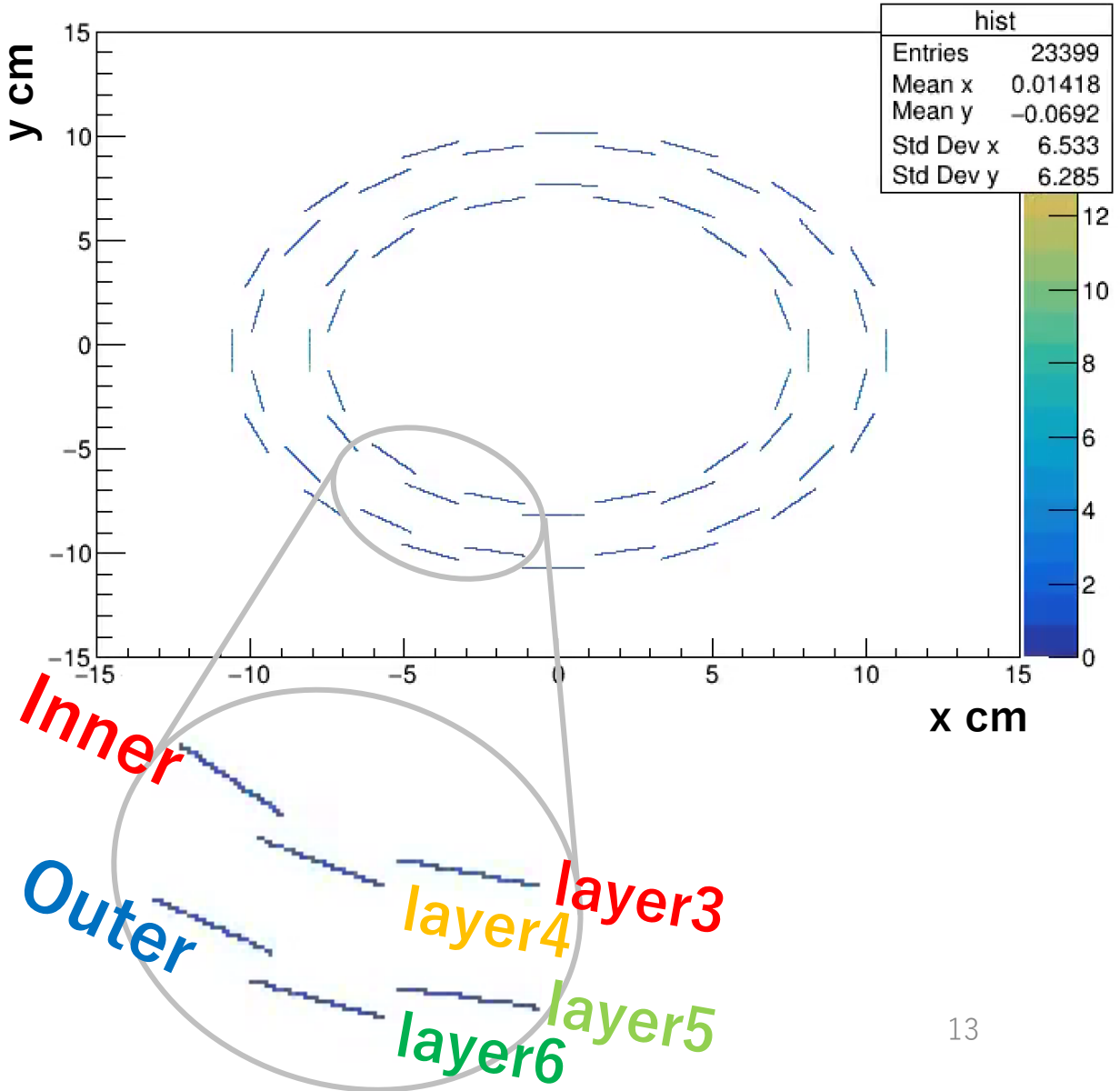
チップタイプ依存



Chip type-B ＞ Chip type-A

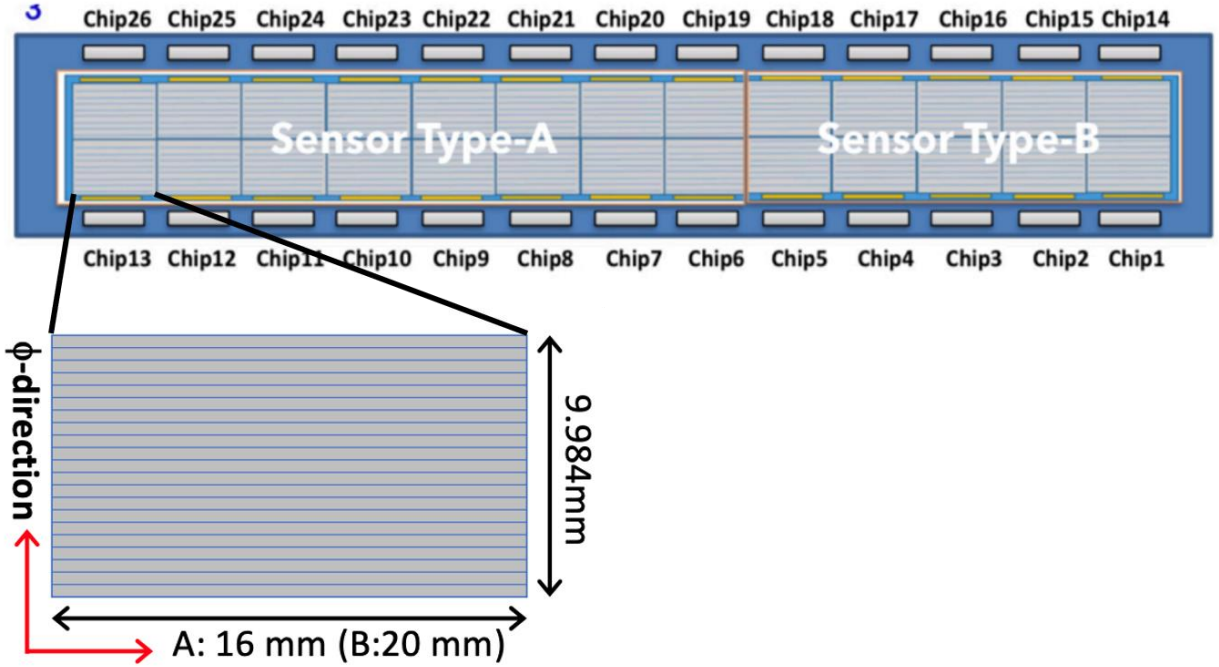
Inner layer ＞ Outer layer

レイヤー依存

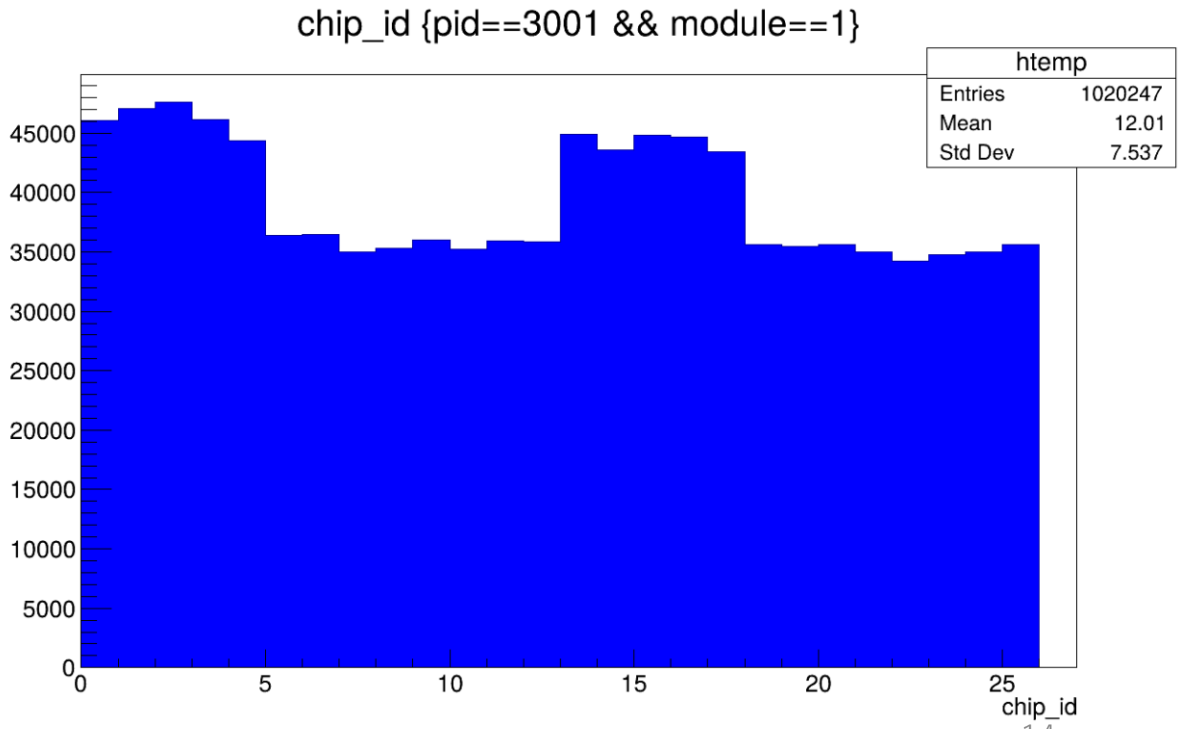


4. ハーフエントリーチップの特定

ランのデータから、ハーフエントリーを持つチップの分類をするコードを作成

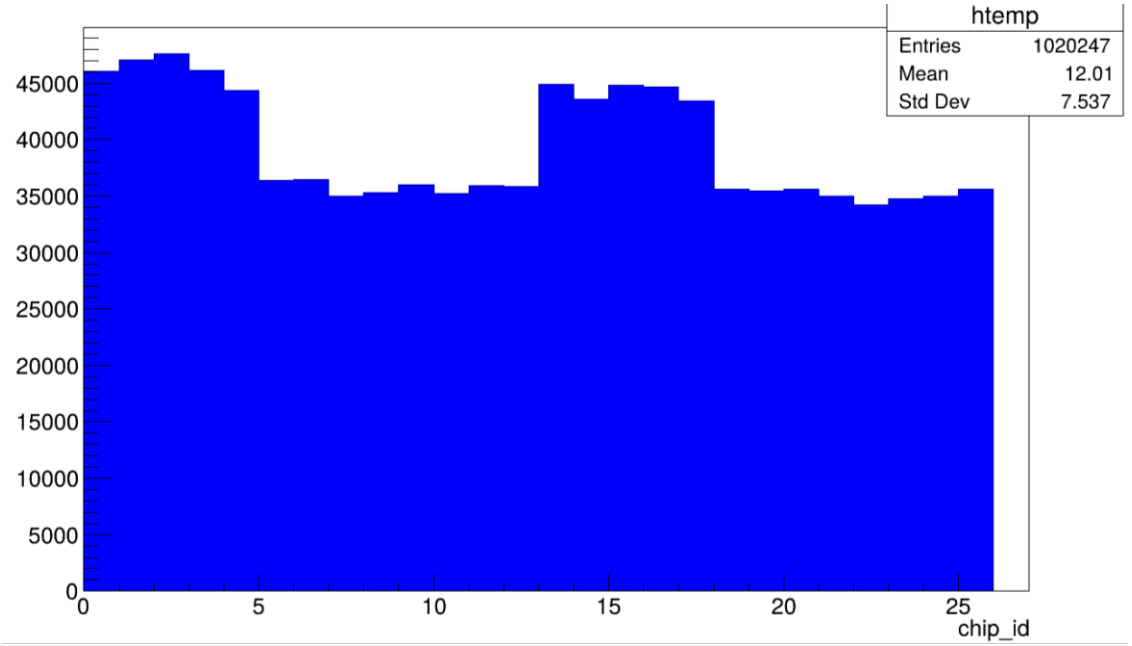


ラダーごとでハーフエントリーの選別を考え、
考慮するのは、**チップタイプ依存性**のみ

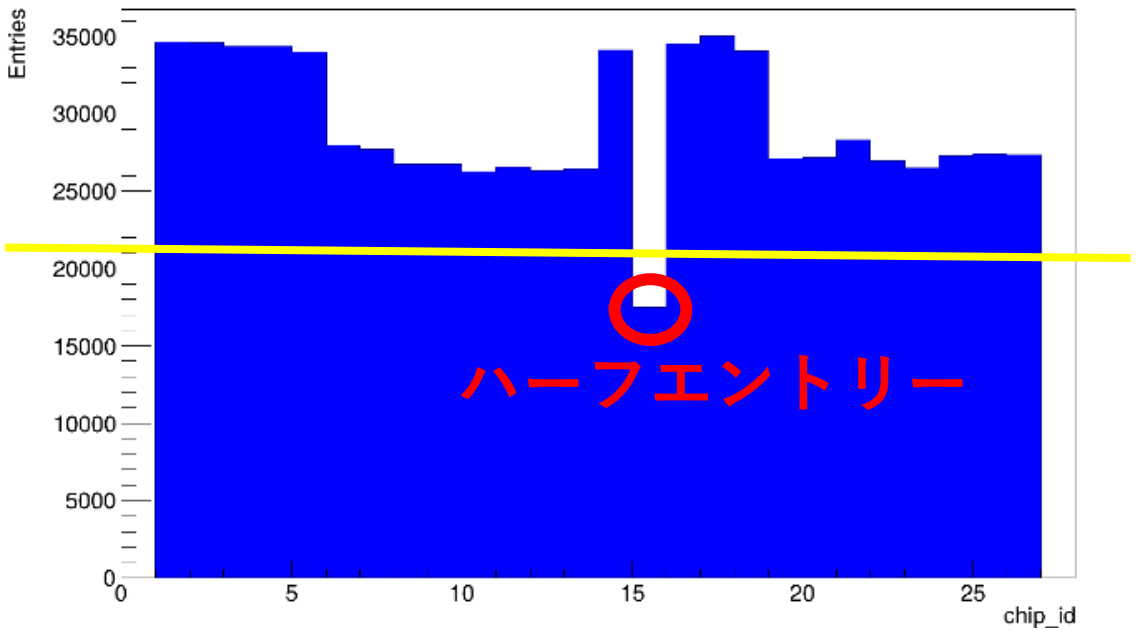


4. ハーフエントリーチップの特定

アルゴリズムの概要



chip_id vs Entries
(pid=3001 & module=1)



chip_id vs Entries
(pid=3001 & module=7)

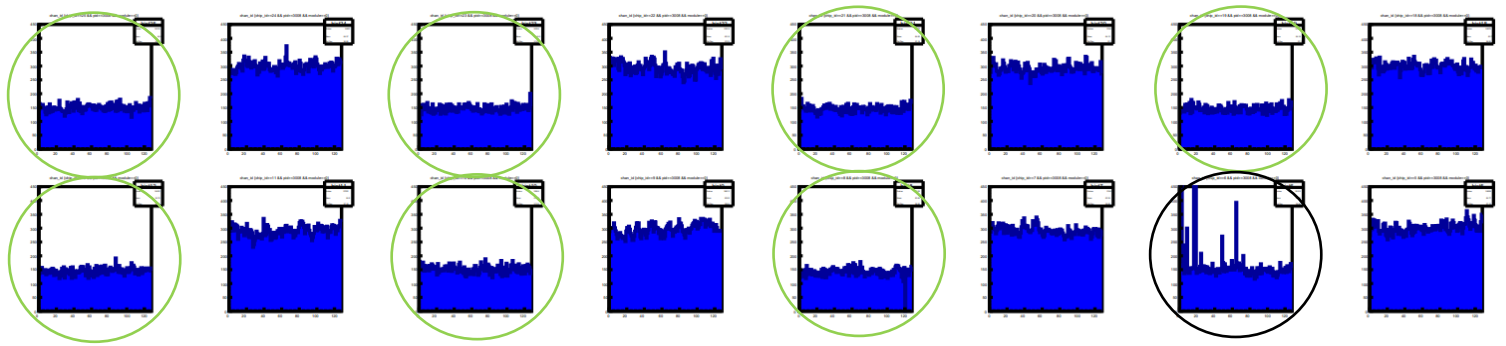
4. ハーフエントリーチップの特定

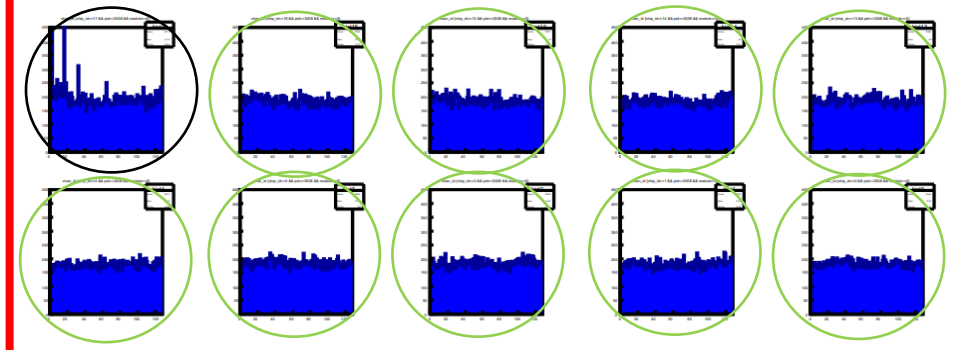
```
// ツリー全体を走査してエントリー数をカウント
Long64_t nentries = tree->GetEntries();
for (Long64_t i = 0; i < nentries; i++) {
  tree->GetEntry(i);

  if (pid >= pid_min && pid <= pid_max &&
      module >= module_min && module <= module_max &&
      chip_id >= chip_id_min && chip_id <= chip_id_max) {
      entries_array[pid - pid_min][module - module_min][chip_id - chip_id_min]++;
  }
}
```

```
for (int p = 0; p <= pid_max - pid_min; ++p) {
  for (int m = 0; m <= module_max - module_min; ++m) {
      if(max_entries_group2_[p][m] * 4.5 > max_entries_group1_[p][m] * 4){
          max_entries_group1[p][m] = max_entries_group2_[p][m] * 5 * 0.6 / 4;
          max_entries_group2[p][m] = max_entries_group2_[p][m] * 0.6;
          //std::cout << p << " , " << m << " , " << max_entries_group1[p][m] << std::endl;
          //std::cout << p << " , " << m << " , " << max_entries_group2[p][m] << std::endl;
      }
      else{
          if(max_entries_group2_[p][m] * 5 < max_entries_group1_[p][m] * 3.6){
              max_entries_group1[p][m] = max_entries_group1_[p][m] * 0.6;
              max_entries_group2[p][m] = max_entries_group1_[p][m] * 4 * 0.6 / 5;
              //std::cout << p << " , " << m << " , " << max_entries_group1[p][m] << std::endl;
              //std::cout << p << " , " << m << " , " << max_entries_group2[p][m] << std::endl;
          }else{
              max_entries_group1[p][m] = max_entries_group1_[p][m] * 0.6;
              max_entries_group2[p][m] = max_entries_group2_[p][m] * 0.6;
              //std::cout << p << " , " << m << " , " << max_entries_group1[p][m] << std::endl;
              //std::cout << p << " , " << m << " , " << max_entries_group2[p][m] << std::endl;
          }
      }
  }
}
```

4. ハーフエントリーチップの特定

ピークを持つチップの対策





```
for (int p = 0; p <= pid_max - pid_min; ++p) {
  for (int m = 0; m <= module_max - module_min; ++m) {
      if(max_entries_group2_[p][m] * 4.5 > max_entries_group1_[p][m] * 4){
          max_entries_group1[p][m] = max_entries_group2_[p][m] * 5 * 0.6 / 4;
          max_entries_group2[p][m] = max_entries_group2_[p][m] * 0.6;
          //std::cout << p << " , " << m << " , " << max_entries_group1[p][m] << std::endl;
          //std::cout << p << " , " << m << " , " << max_entries_group2[p][m] << std::endl;
      }
      else{
          if(max_entries_group2_[p][m] * 5 < max_entries_group1_[p][m] * 3.6){
              max_entries_group1[p][m] = max_entries_group1_[p][m] * 0.6;
              max_entries_group2[p][m] = max_entries_group1_[p][m] * 4 * 0.6 / 5;
              //std::cout << p << " , " << m << " , " << max_entries_group1[p][m] << std::endl;
              //std::cout << p << " , " << m << " , " << max_entries_group2[p][m] << std::endl;
          }else{
              max_entries_group1[p][m] = max_entries_group1_[p][m] * 0.6;
              max_entries_group2[p][m] = max_entries_group2_[p][m] * 0.6;
              //std::cout << p << " , " << m << " , " << max_entries_group1[p][m] << std::endl;
              //std::cout << p << " , " << m << " , " << max_entries_group2[p][m] << std::endl;
          }
      }
  }
}
```

4. ハーフエントリーチップの特定

```
for (int p = 0; p <= pid_max - pid_min; p++) {
  for (int m = 0; m <= module_max - module_min; m++) {
      for (int c = 0; c <= chip_id_max - chip_id_min; c++) {
          if ((c >= 0 && c <= 4) || (c >= 13 && c <= 17)) {
              if (entries_array[p][m][c] < max_entries_group1[p][m] && entries_array[p][m][c] > 5000) {
                  pid = p + pid_min;
                  module = m + module_min;
                  chip_id = c + 1;
                  entries = entries_array[p][m][c];
                  ki->Fill();
                  std::cout << "PID: " << pid << ", Module: " << module
                            << ", chip_id: " << chip_id << std::endl;
                  half_entry_counter++;
              }
          } else {
              if (entries_array[p][m][c] < max_entries_group2[p][m] && entries_array[p][m][c] > 5000) {
                  pid = p + pid_min;
                  module = m + module_min;
                  chip_id = c + 1;
                  entries = entries_array[p][m][c];
                  ki->Fill();
                  std::cout << "PID: " << pid << ", Module: " << module
                            << ", chip_id: " << chip_id << std::endl;
                  half_entry_counter++;
              }
          }
      }
  }
```

4. ハーフエントリーチップの特定

ランのデータから、ハーフエントリーを持つチップの分類をするコードを作成

Felix Server	0	0	3	5	7	7
pid	3001	3001	3004	3006	3008	3008
module	6	7	13	3	0	1
chip id	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	15	21, 23, 25	16	1, 2, 3, 4, 5, 7, 9, 11, 13, 14, 15, 16, 17, 18, 20, 22, 24, 26	1, 2

```
PID: 3001, Module: 6, chip_id: 1
PID: 3001, Module: 6, chip_id: 2
PID: 3001, Module: 6, chip_id: 3
PID: 3001, Module: 6, chip_id: 4
PID: 3001, Module: 6, chip_id: 5
PID: 3001, Module: 6, chip_id: 6
PID: 3001, Module: 6, chip_id: 7
PID: 3001, Module: 6, chip_id: 8
PID: 3001, Module: 6, chip_id: 9
PID: 3001, Module: 6, chip_id: 10
PID: 3001, Module: 6, chip_id: 11
PID: 3001, Module: 6, chip_id: 12
PID: 3001, Module: 6, chip_id: 13
PID: 3001, Module: 7, chip_id: 15
PID: 3004, Module: 13, chip_id: 21
PID: 3004, Module: 13, chip_id: 23
PID: 3004, Module: 13, chip_id: 25
PID: 3006, Module: 3, chip_id: 16
PID: 3008, Module: 0, chip_id: 1
PID: 3008, Module: 0, chip_id: 2
PID: 3008, Module: 0, chip_id: 3
PID: 3008, Module: 0, chip_id: 4
PID: 3008, Module: 0, chip_id: 5
PID: 3008, Module: 0, chip_id: 9
PID: 3008, Module: 0, chip_id: 11
PID: 3008, Module: 0, chip_id: 13
PID: 3008, Module: 0, chip_id: 14
PID: 3008, Module: 0, chip_id: 15
PID: 3008, Module: 0, chip_id: 16
PID: 3008, Module: 0, chip_id: 17
PID: 3008, Module: 0, chip_id: 18
PID: 3008, Module: 0, chip_id: 20
PID: 3008, Module: 0, chip_id: 22
PID: 3008, Module: 0, chip_id: 24
PID: 3008, Module: 0, chip_id: 26
PID: 3008, Module: 1, chip_id: 1
PID: 3008, Module: 1, chip_id: 2
```