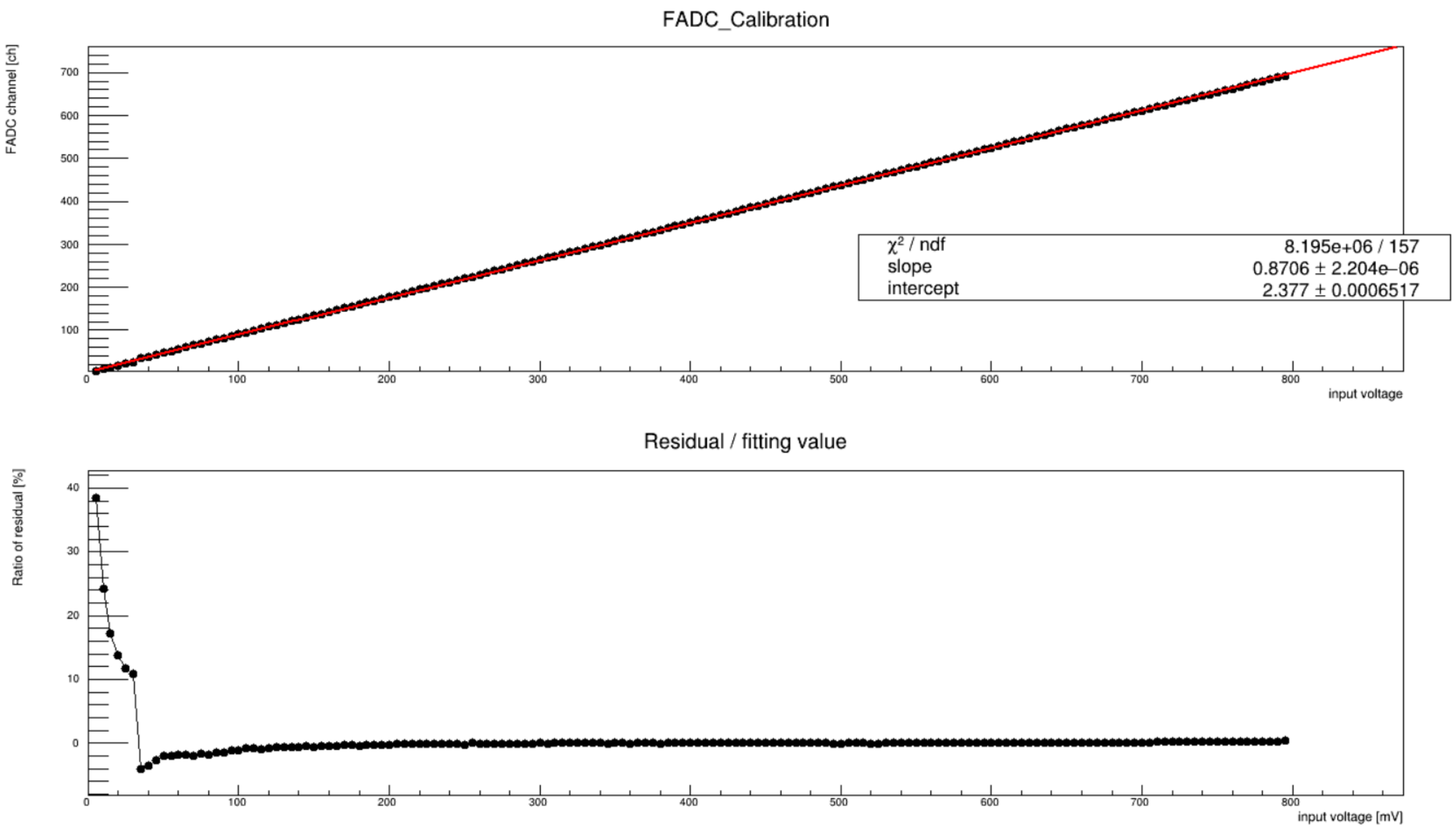
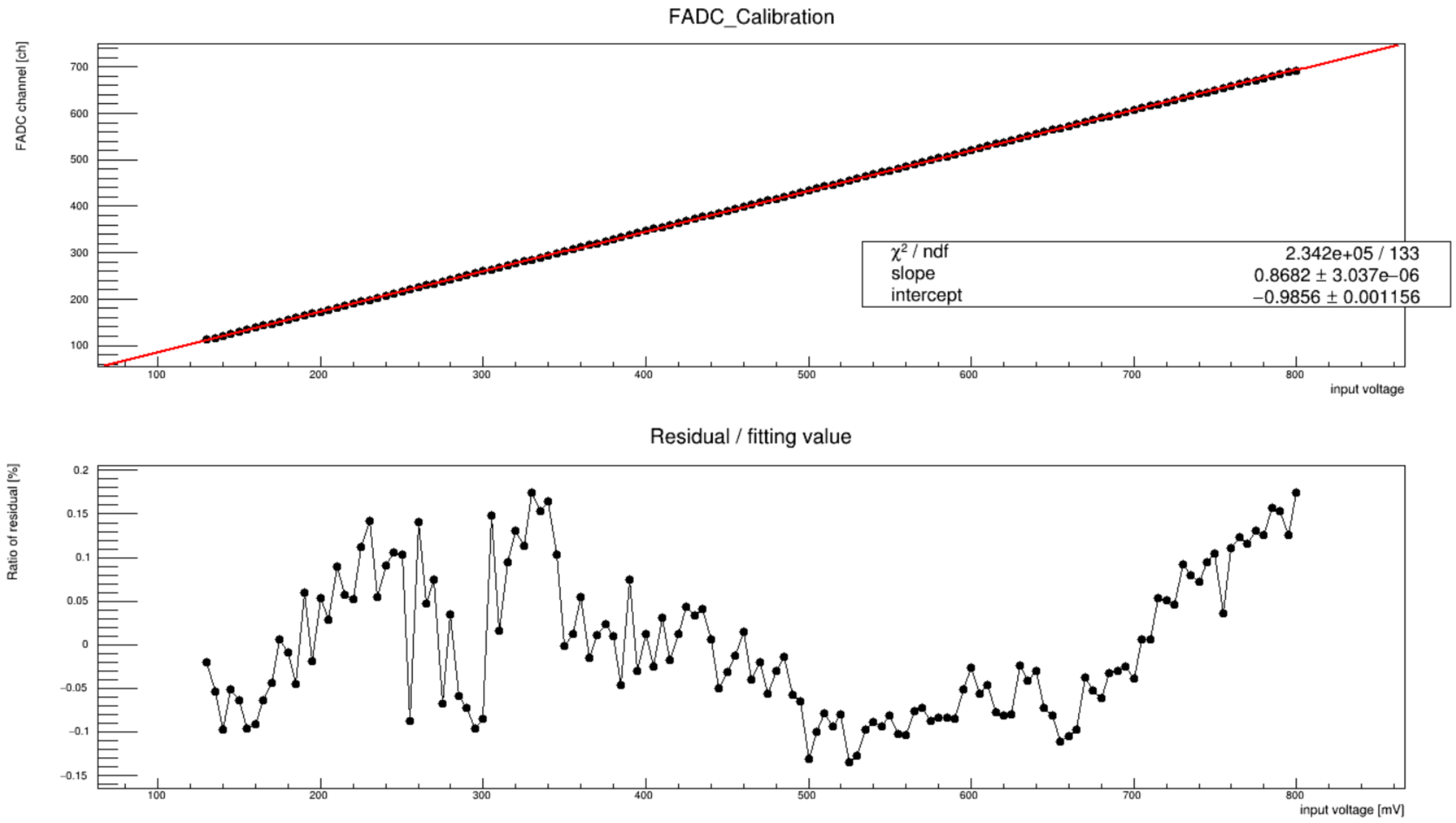
FADC test

The whole



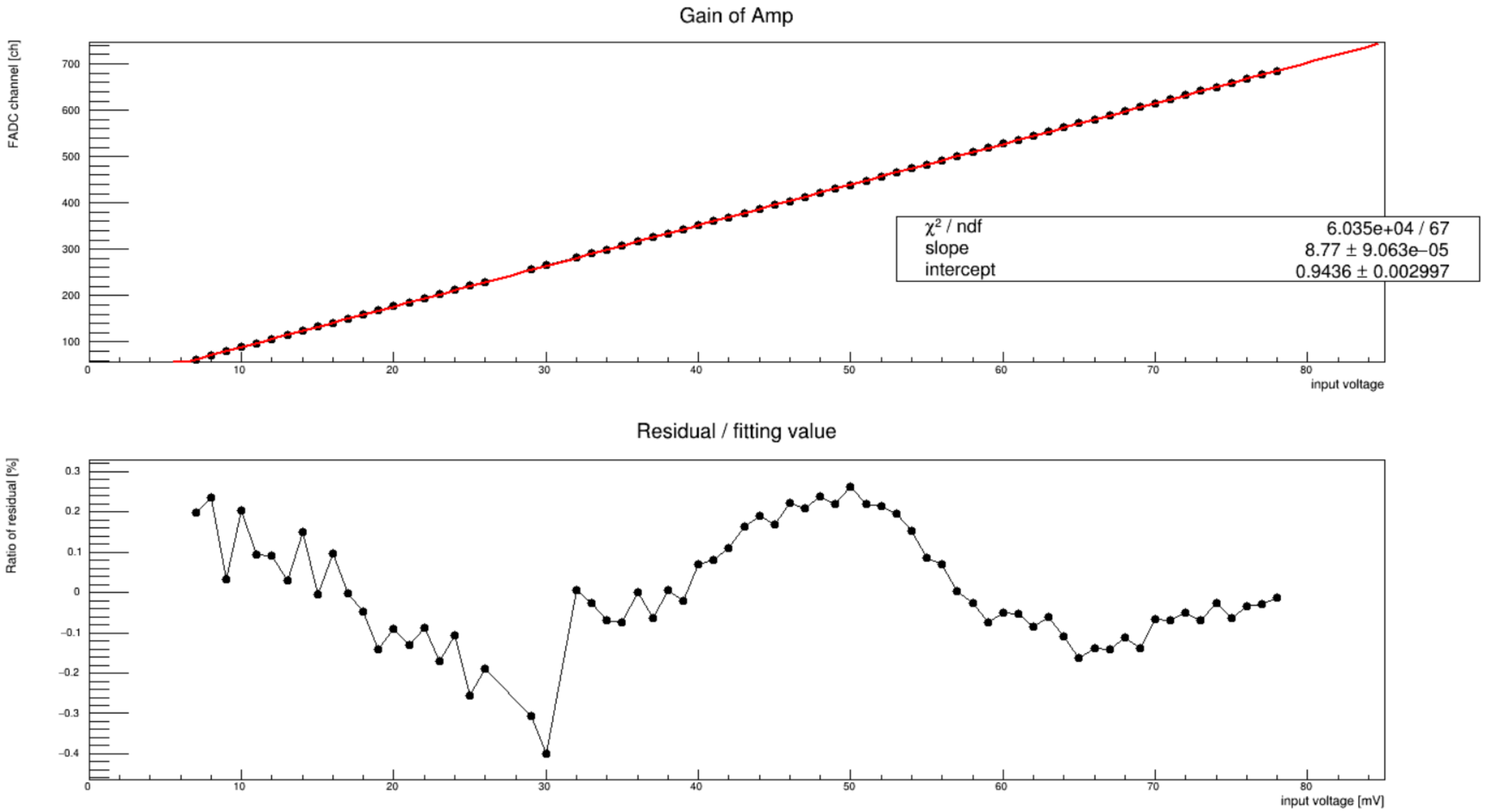
FADC_Calibration
FADC channel [ch]
700
600
500
400
300
200
100
0
100
200
300
400
500
600
700
800
input voltage
χ² / ndf 8.195e+06 / 157
slope 0.8706 ± 2.204e−06
intercept 2.377 ± 0.0006517
Residual / fitting value
Ratio of residual [%]
40
30
20
10
0
input voltage [mV]

From 110 mV to 800 mV



Amp7 test

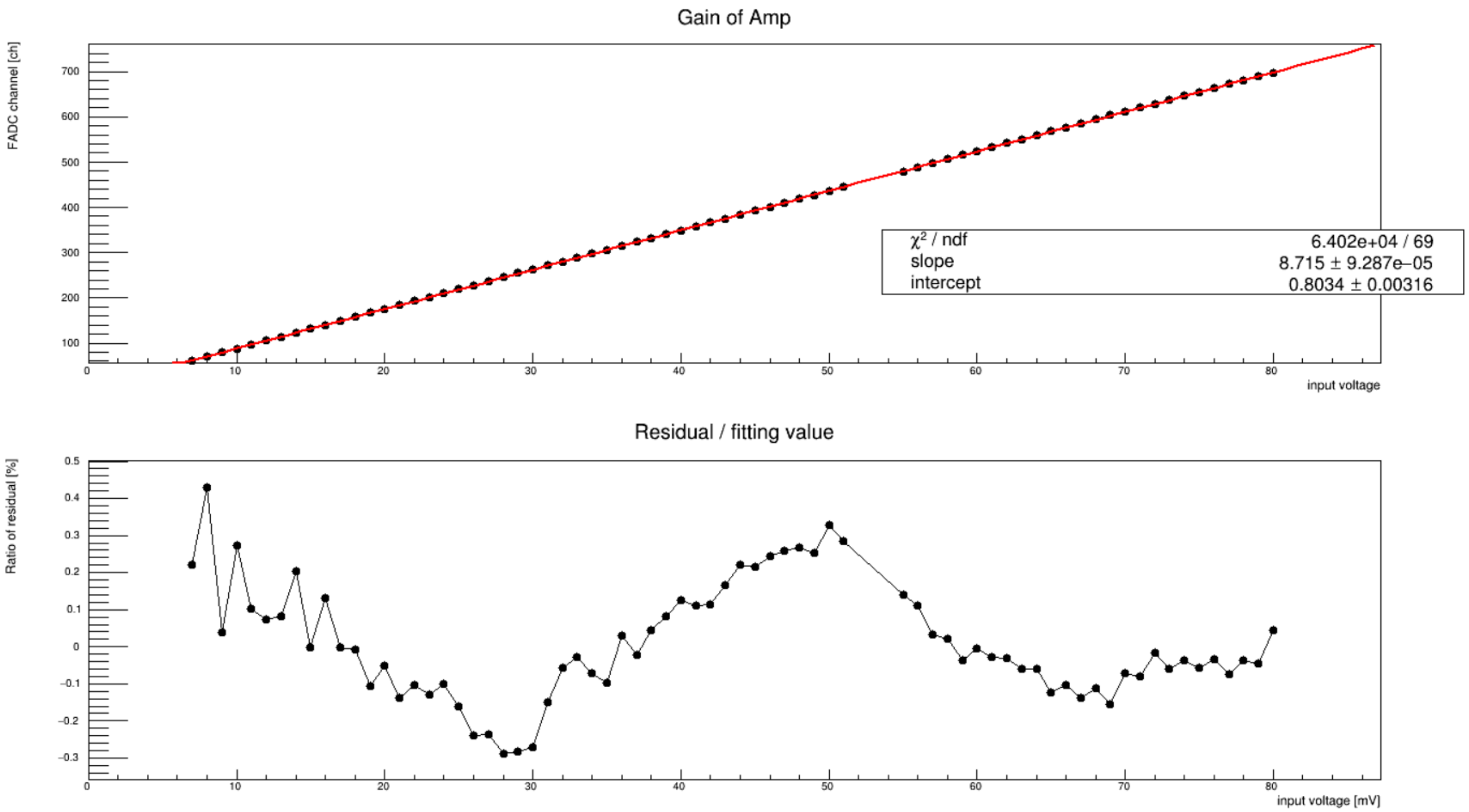
7 mV ~ 80 mV



Gain of Amp
FADC channel [ch]
input voltage
χ^2 / ndf 6.035e+04 / 67
slope 8.77 ± 9.063e−05
intercept 0.9436 ± 0.002997
Residual / fitting value
Ratio of residual [%]
input voltage [mV]

Amp8 test

7 mV ~ 80 mV



Gain of Amp
FADC channel [ch]
700
600
500
400
300
200
100
0
10
20
30
40
50
60
70
80
input voltage
χ² / ndf 6.402e+04 / 69
slope 8.715 ± 9.287e−05
intercept 0.8034 ± 0.00316
Residual / fitting value
Ratio of residual [%]
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
input voltage [mV]

Conclusion

- Considering the result of FADC calibration, gain of Amp7 is about 10.10 (= 8.77 mV / 0.8682 mV)
- Considering the result of FADC calibration, gain of Amp8 is about 10.03 (= 8.715 mV / 0.8682 mV)

Gain of MPPC

Fitting Function

- Assumption

- The number of photon from thermoemission is smaller than the number of photon from LED

- 실제 신호

- LED에서 발생되는 photon 중 일부만이 MPPC에 도달하므로 Poisson distribution을 따른다.
- Avalanche 에 의해 만들어지는 신호는 Gaussian distributio을 따른다.
- 따라서, 두 함수의 Convolution으로 표현된다.

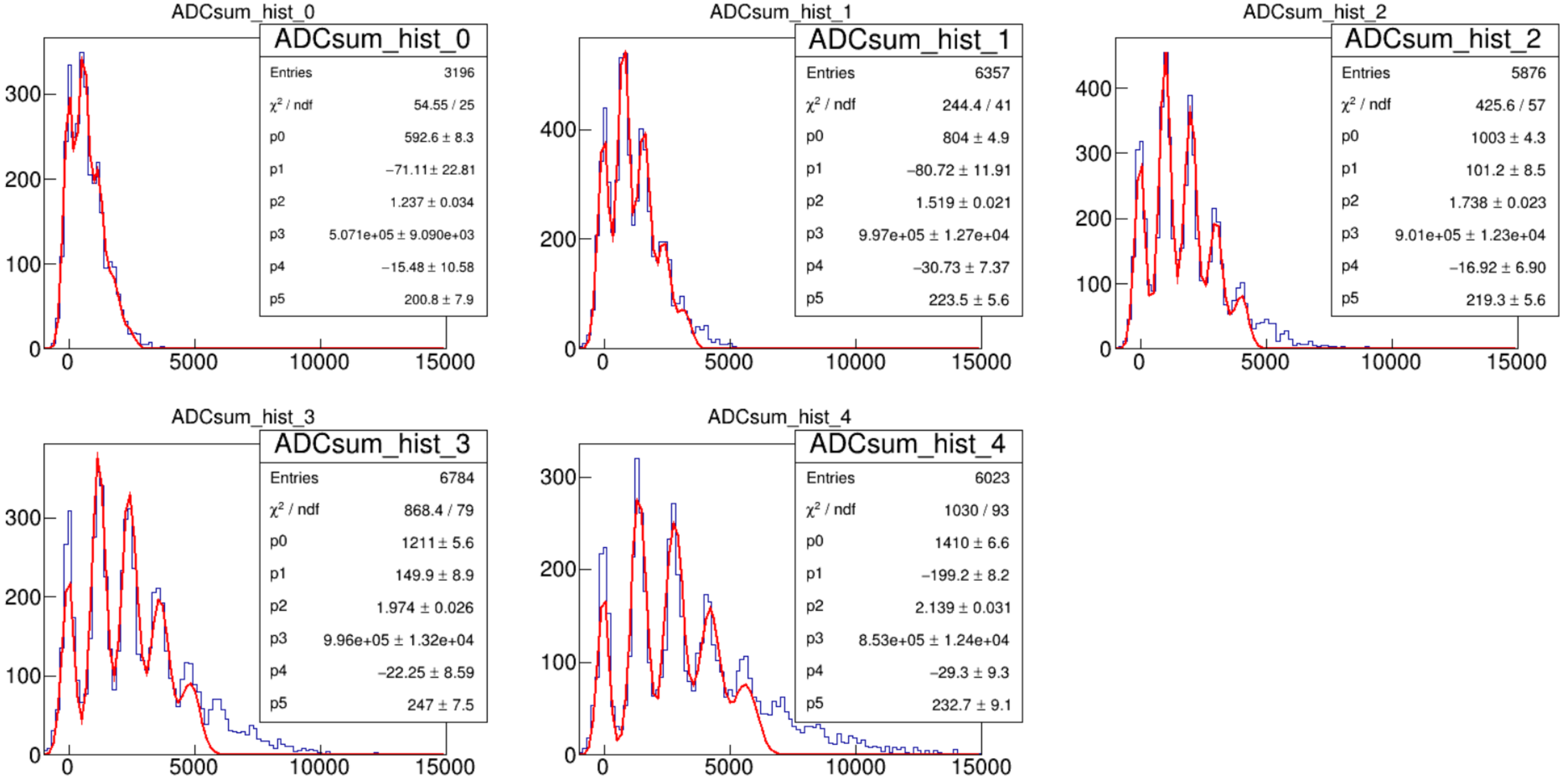
- Noise 신호

- 가정에 의해 pedestal을 만드는 noise 신호만 고려하면 되고, 이 경우 Gaussian distribution을 따른다.

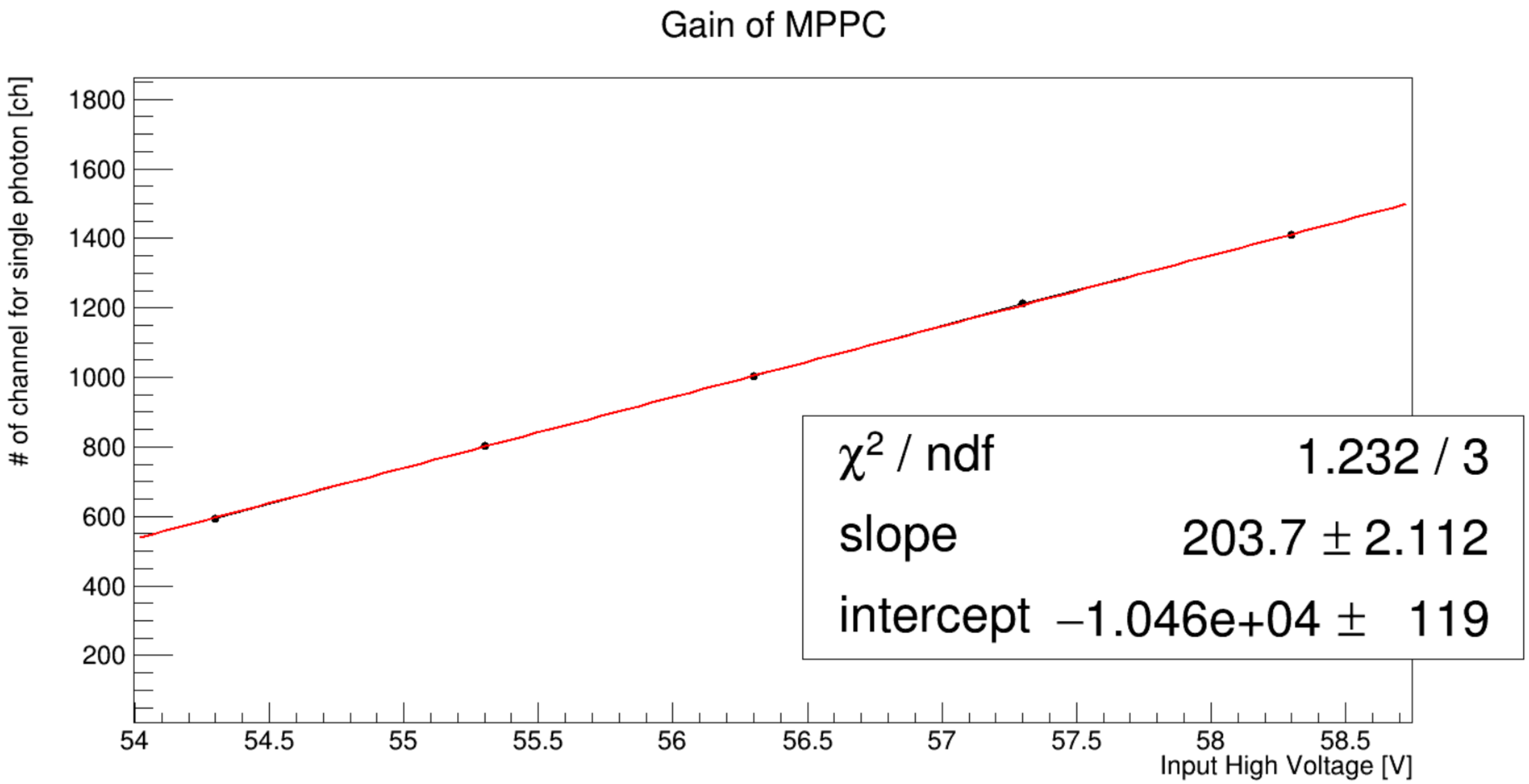
Fitting Function

```
par[0] : single photon ADC sum
par[1] : sigma for single photon
par[2] : parameter for Poisson distribution(mean and variation of Poission distribution)
par[3] : normalization factor
*/
double pedestal_mean = -5.493;
double pedestal_sigma = 260.3;

double fitting_function = TMath::Poisson(0, par[2]) * TMath::Gaus(x[0], pedestal_mean, pedestal_sigma, 1);
double fitting_function = 0;
for (int i = 0; i < 5; i++)
{
    double n = (double)i;
    fitting_function += TMath::Poisson(n, par[2]) * TMath::Gaus(x[0], pedestal_mean + n * par[0], TMath::Sqrt(pedestal_sigma * pedestal_sigma + n * par[1] * par[1]), 1);
}

fitting_function = par[3] * fitting_function;

return fitting_function;
```

Fitting results



Gain



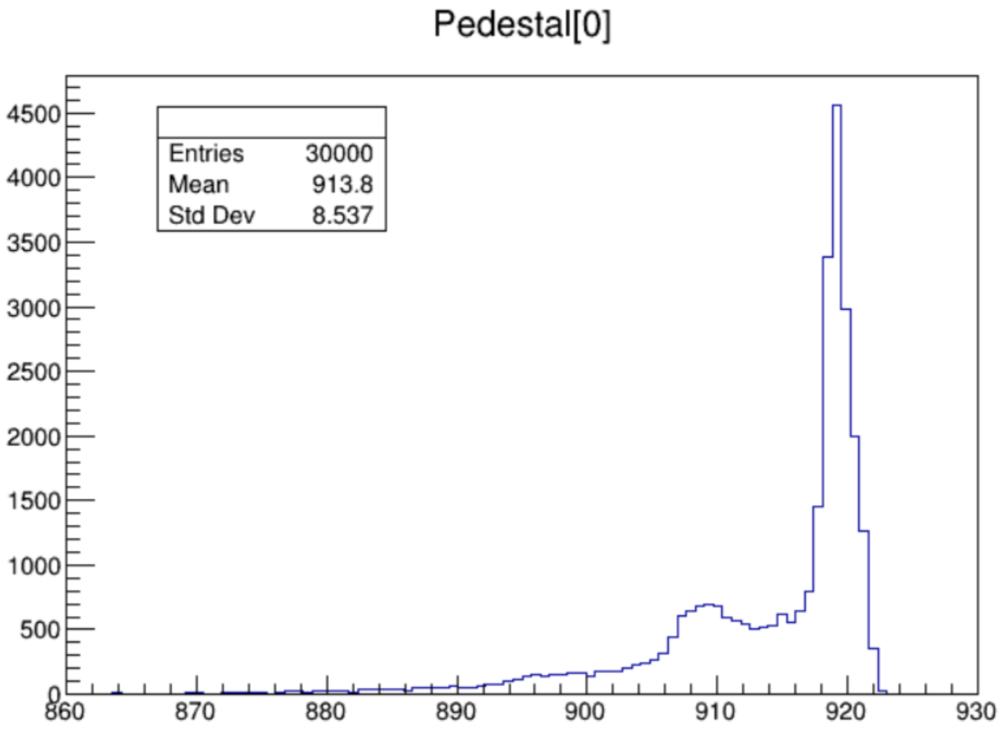
Gain of MPPC
# of channel for single photon [ch]
1800
1600
1400
1200
1000
800
600
400
200
54
54.5
55
55.5
56
56.5
57
57.5
58
58.5
Input High Voltage [V]
χ² / ndf 1.232 / 3
slope 203.7 ± 2.112
intercept −1.046e+04 ± 119

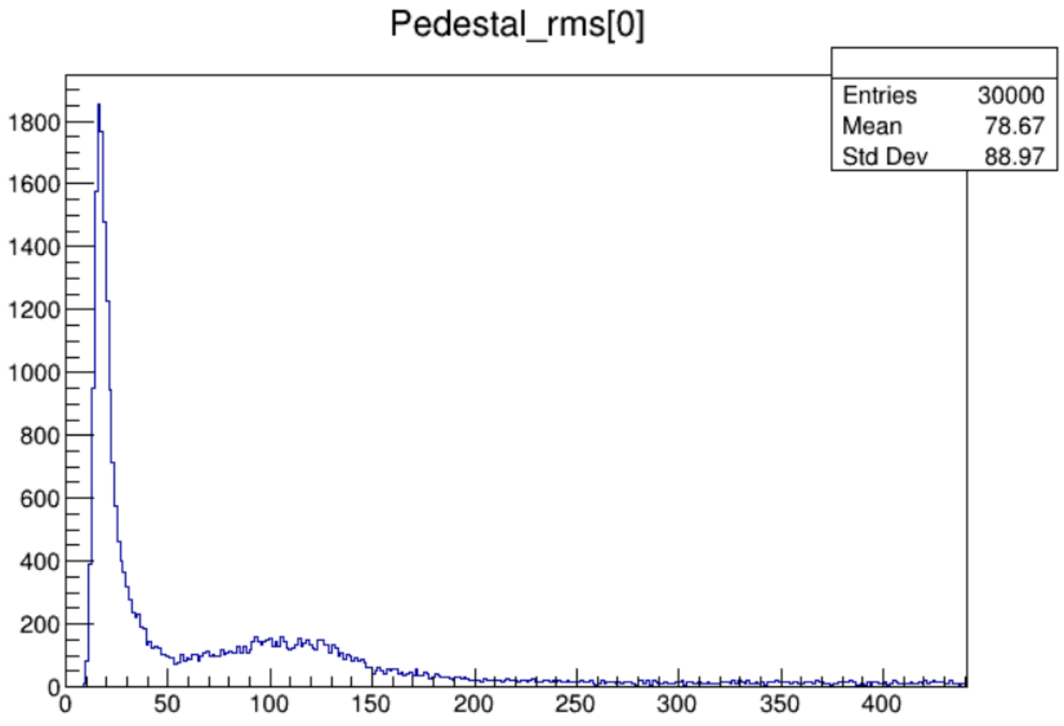
Random trigger

Pedestal 정하기

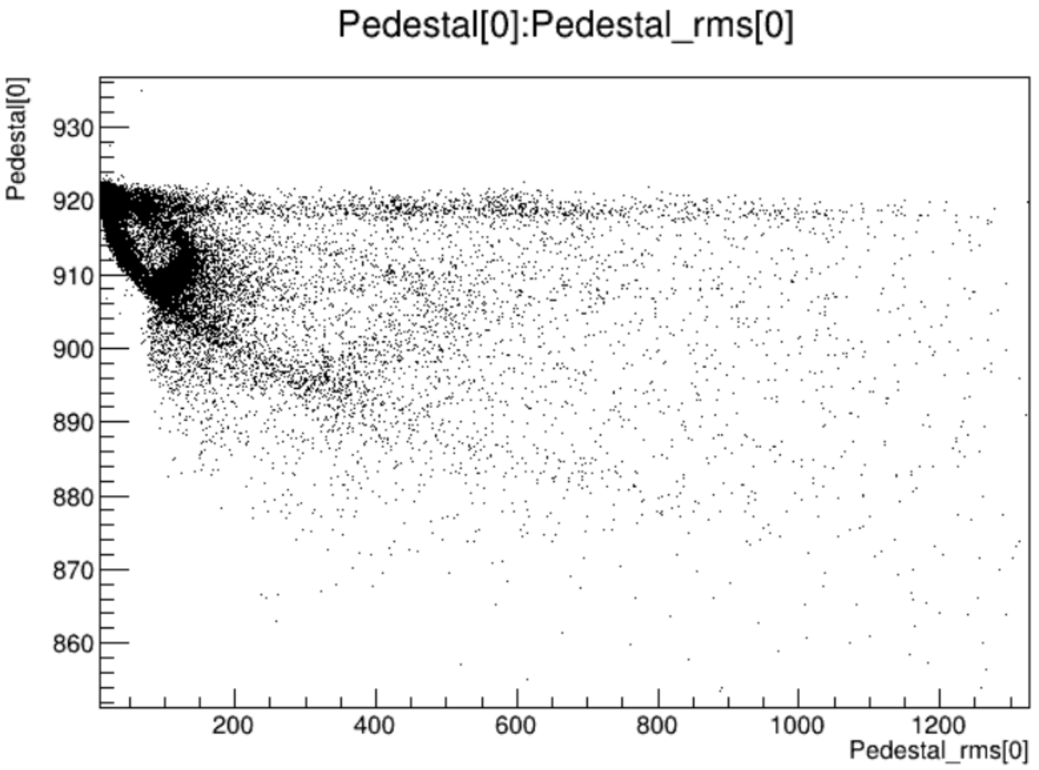
- Random trigger의 경우, FADC로 측정된 앞 구간이 pedestal로 사용할 수 있을 정도로 편평하다고 할 수 없다.
- 따라서 앞에서 100 point의 평균과 표준편차를 계산하고, 표준편차의 값이 작을 경우에 대해서만 pedestal을 계산하는데 사용한다.

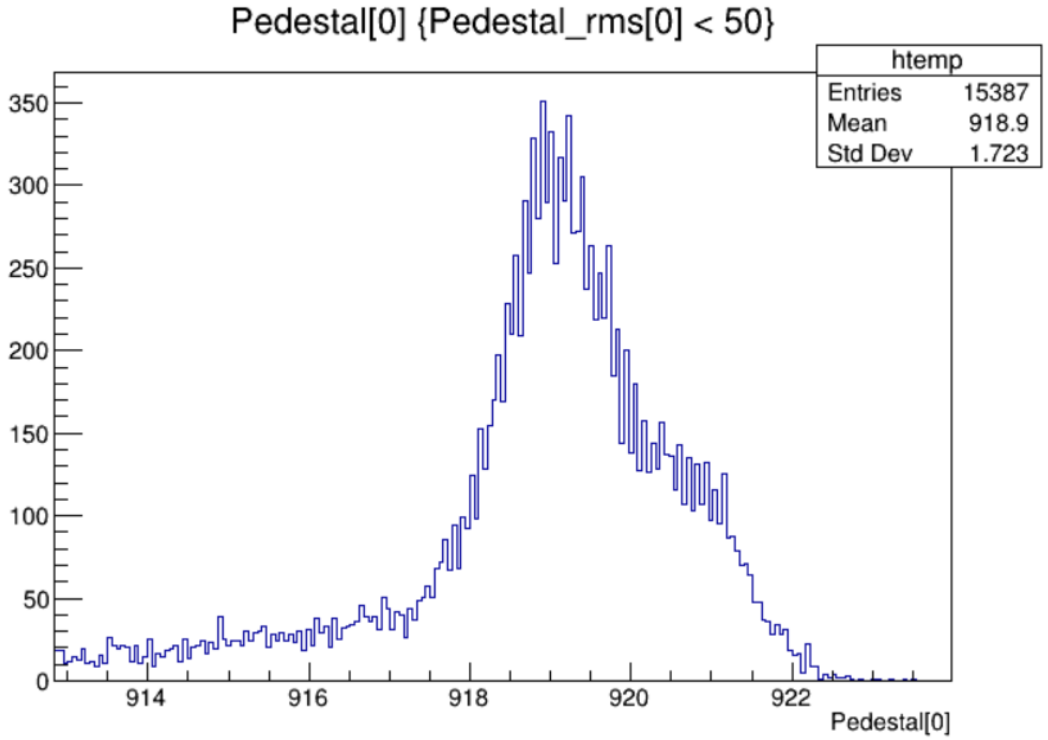
Pedestal and standard deviation



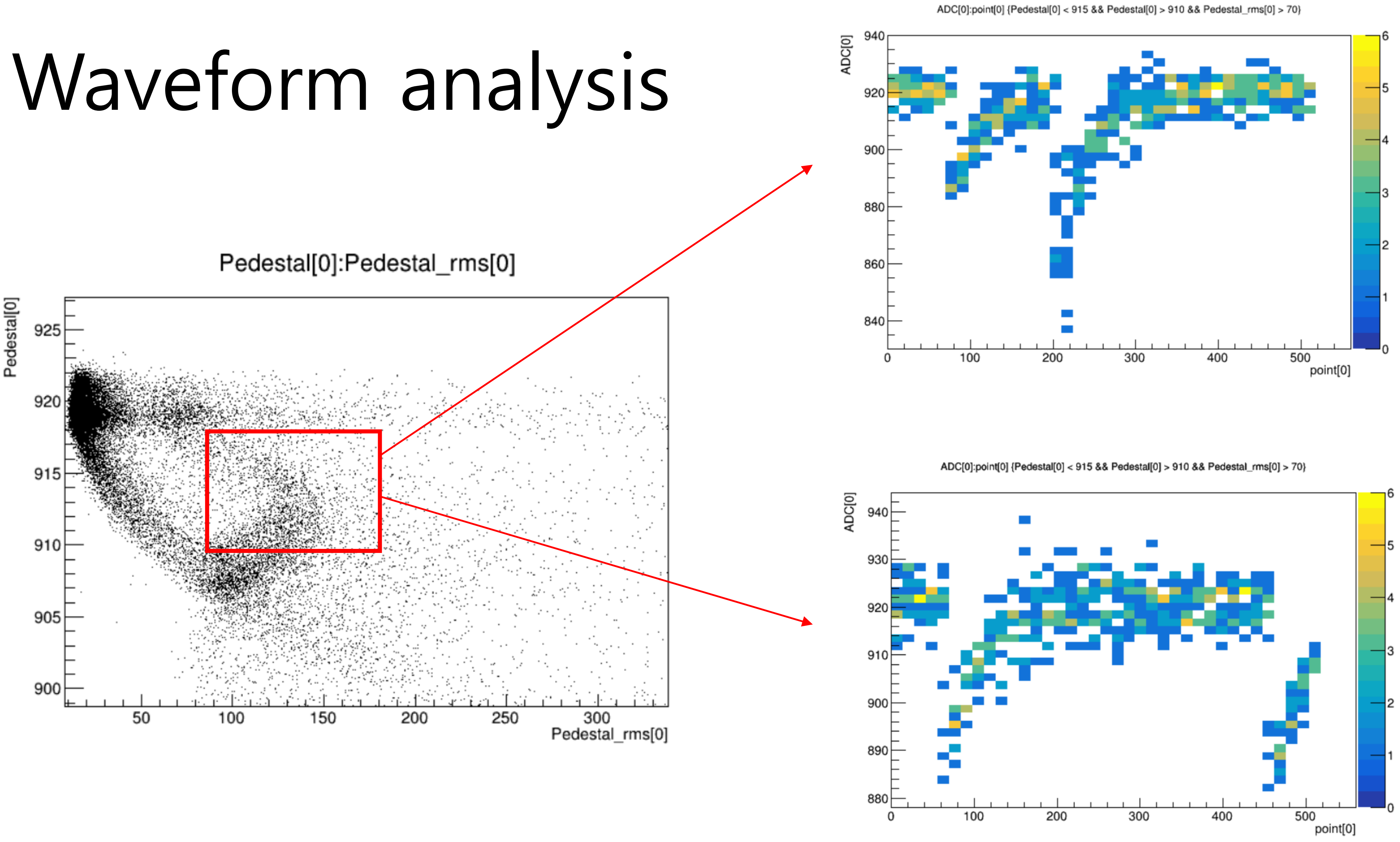


Relation

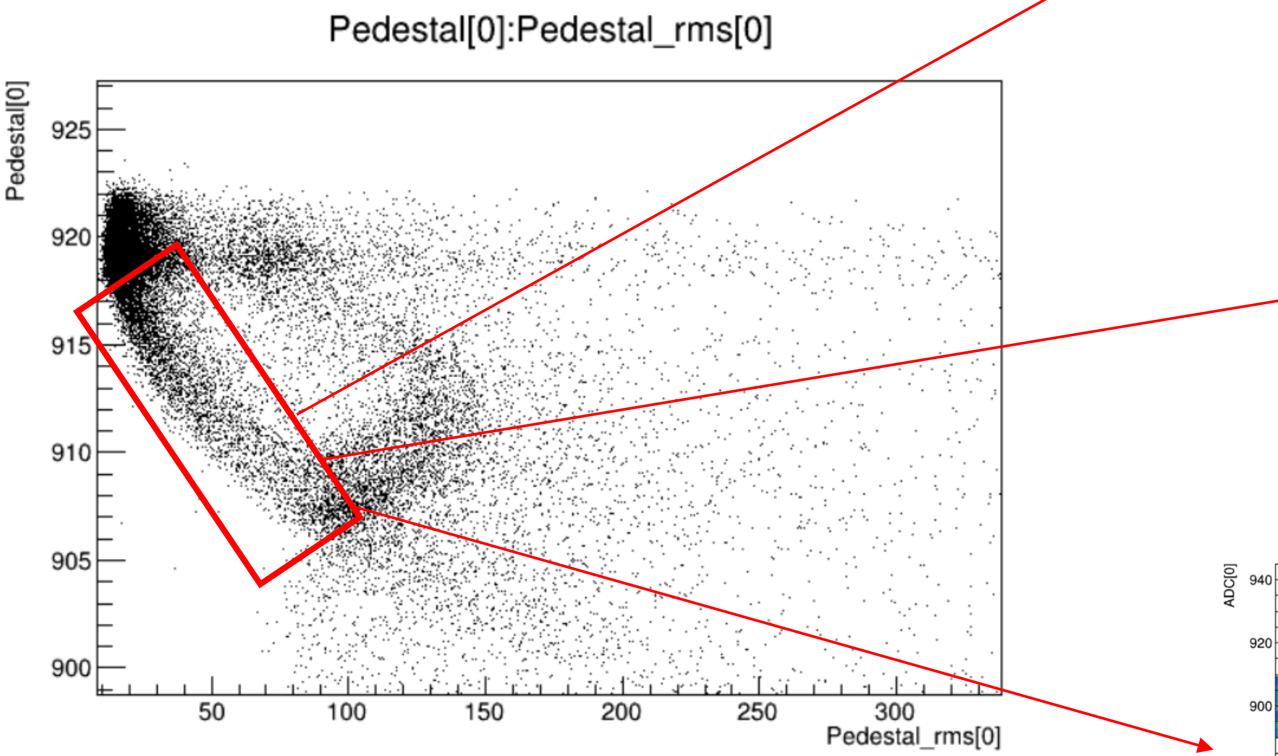

Pedestal[0]:Pedestal_rms[0]
Pedestal[0]
930
920
910
900
890
880
870
860
200
400
600
800
1000
1200
Pedestal_rms[0]

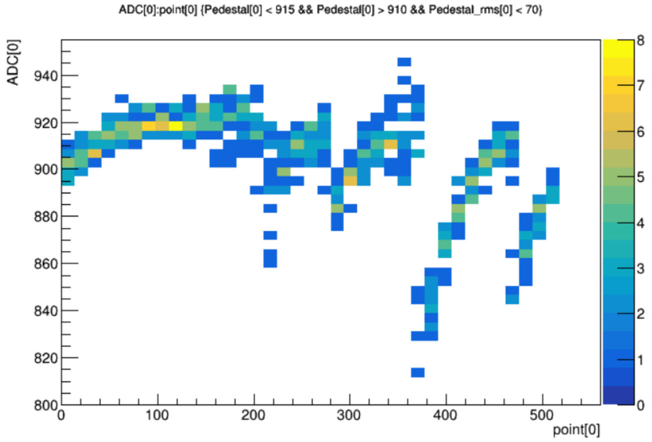

Pedestal[0] {Pedestal_rms[0] < 50}
htemp
Entries 15387
Mean 918.9
Std Dev 1.723
350
300
250
200
150
100
50
0
914
916
918
920
922
Pedestal[0]

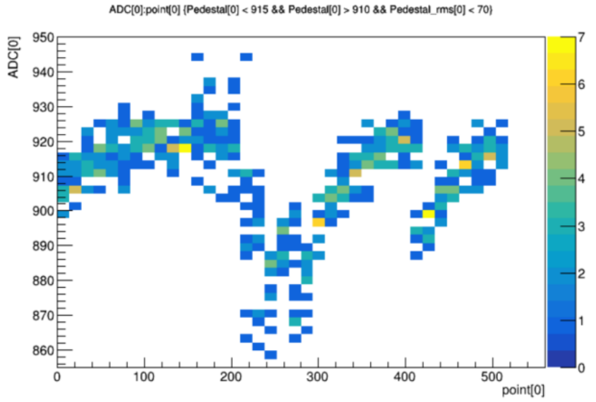
Waveform analysis

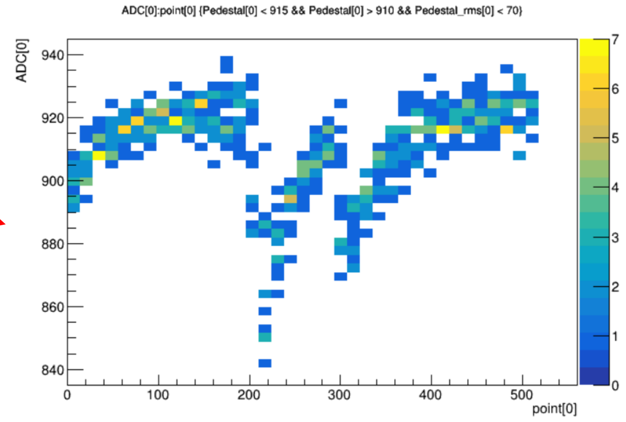


Waveform analysis

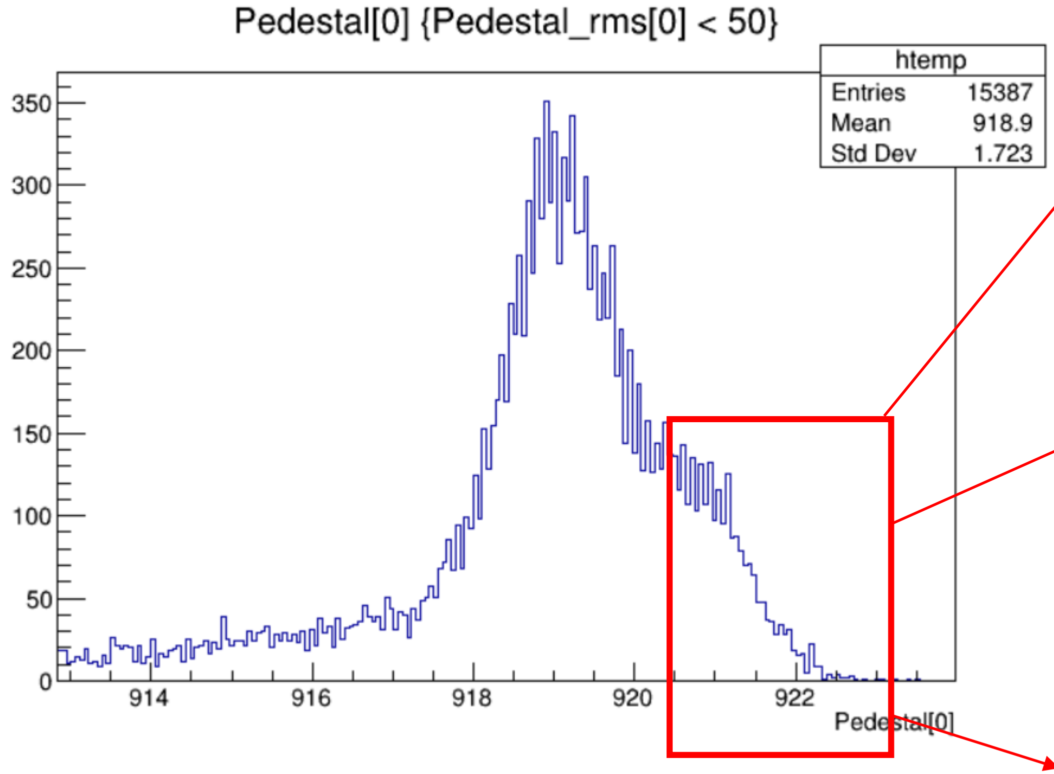


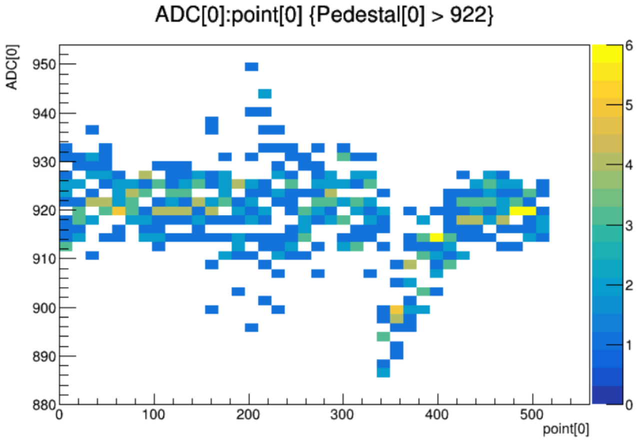


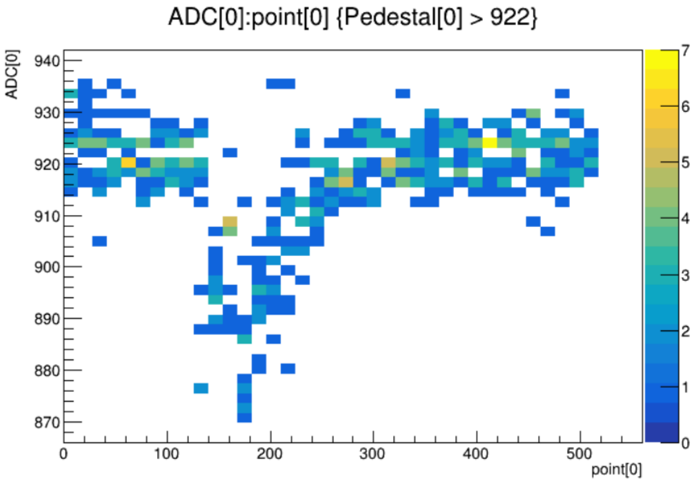


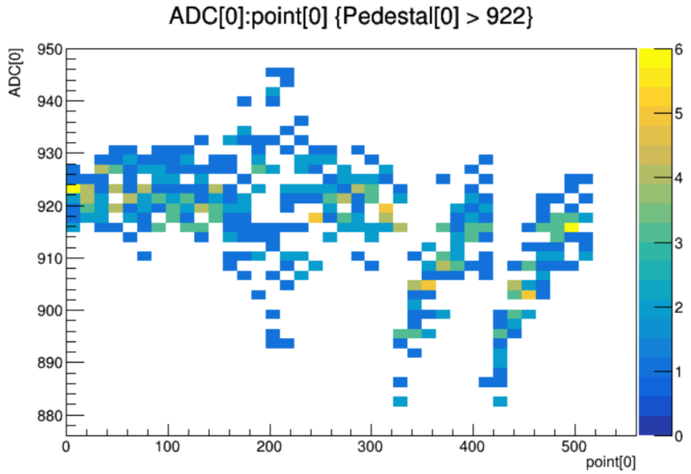


Waveform analysis



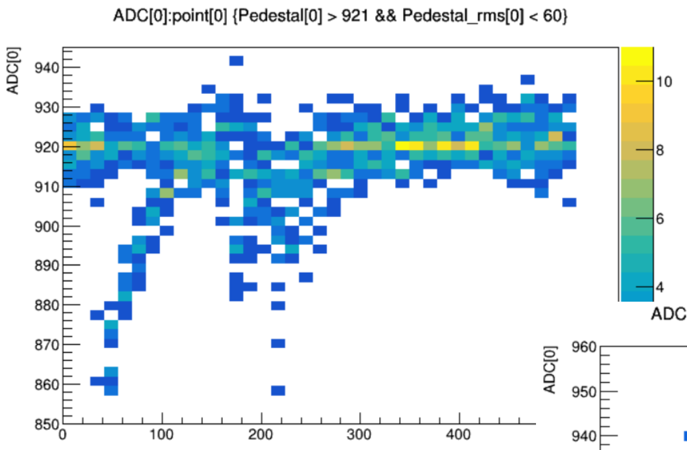


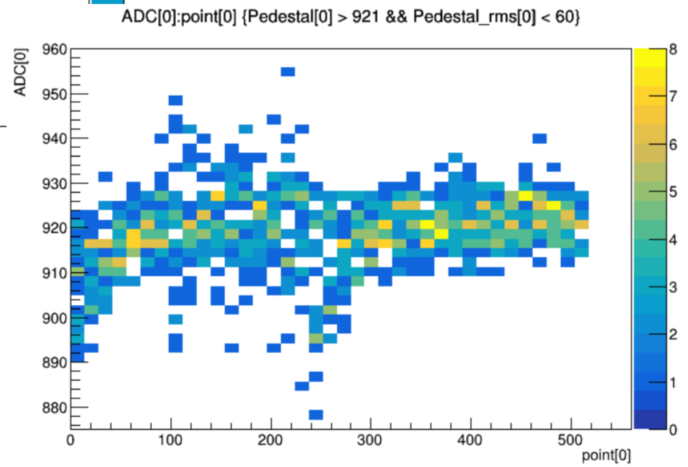


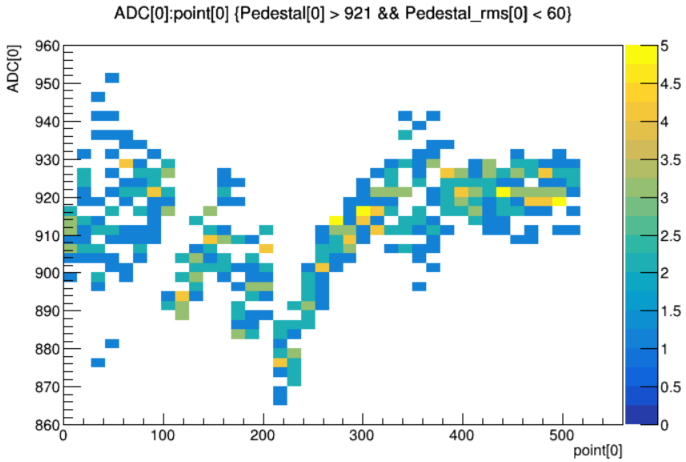


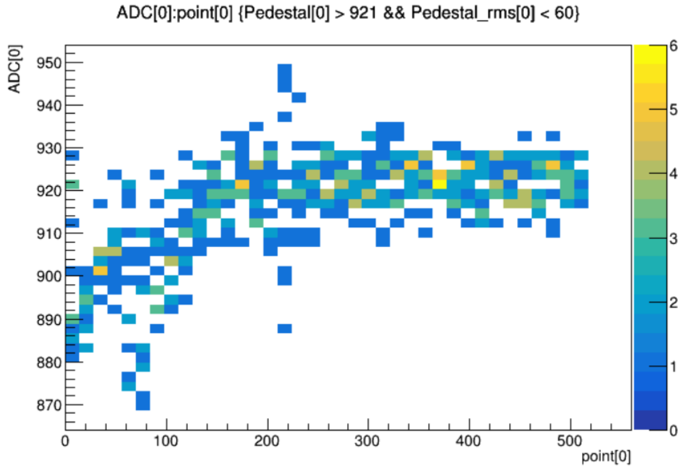
Waveform analysis

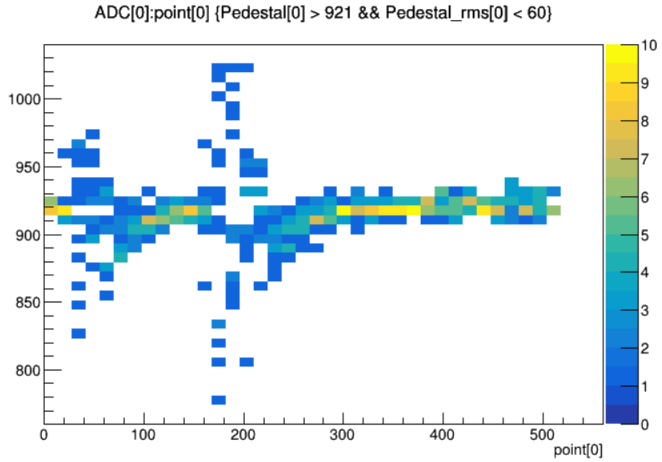
- 앞쪽에 신호가 있어, overshoot에 의한 것일 수도 있다고 생각하여, Pedestal을 정하는 영역을 뒷부분으로 설정하여 waveform을 확인하였으나, 그렇지 않다는 것을 확인하였다.



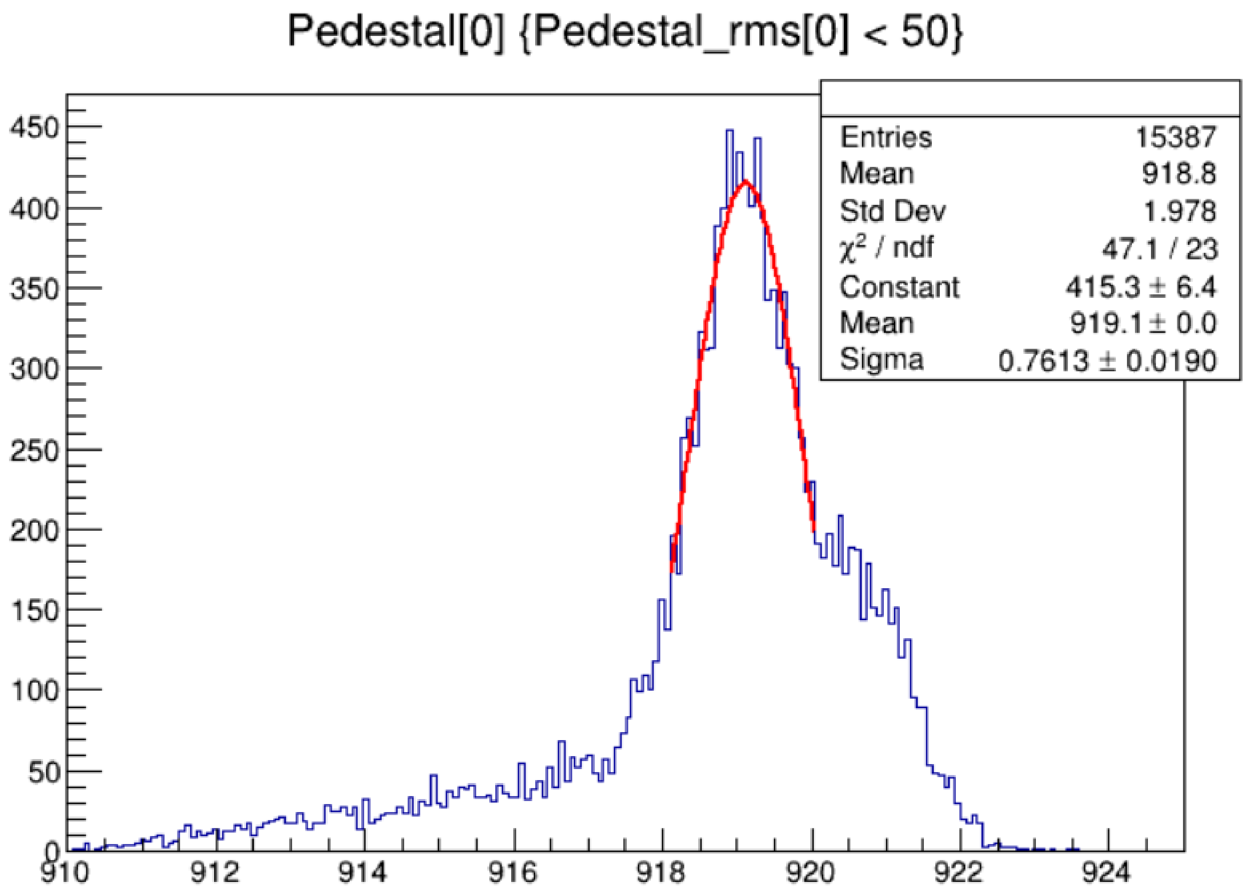




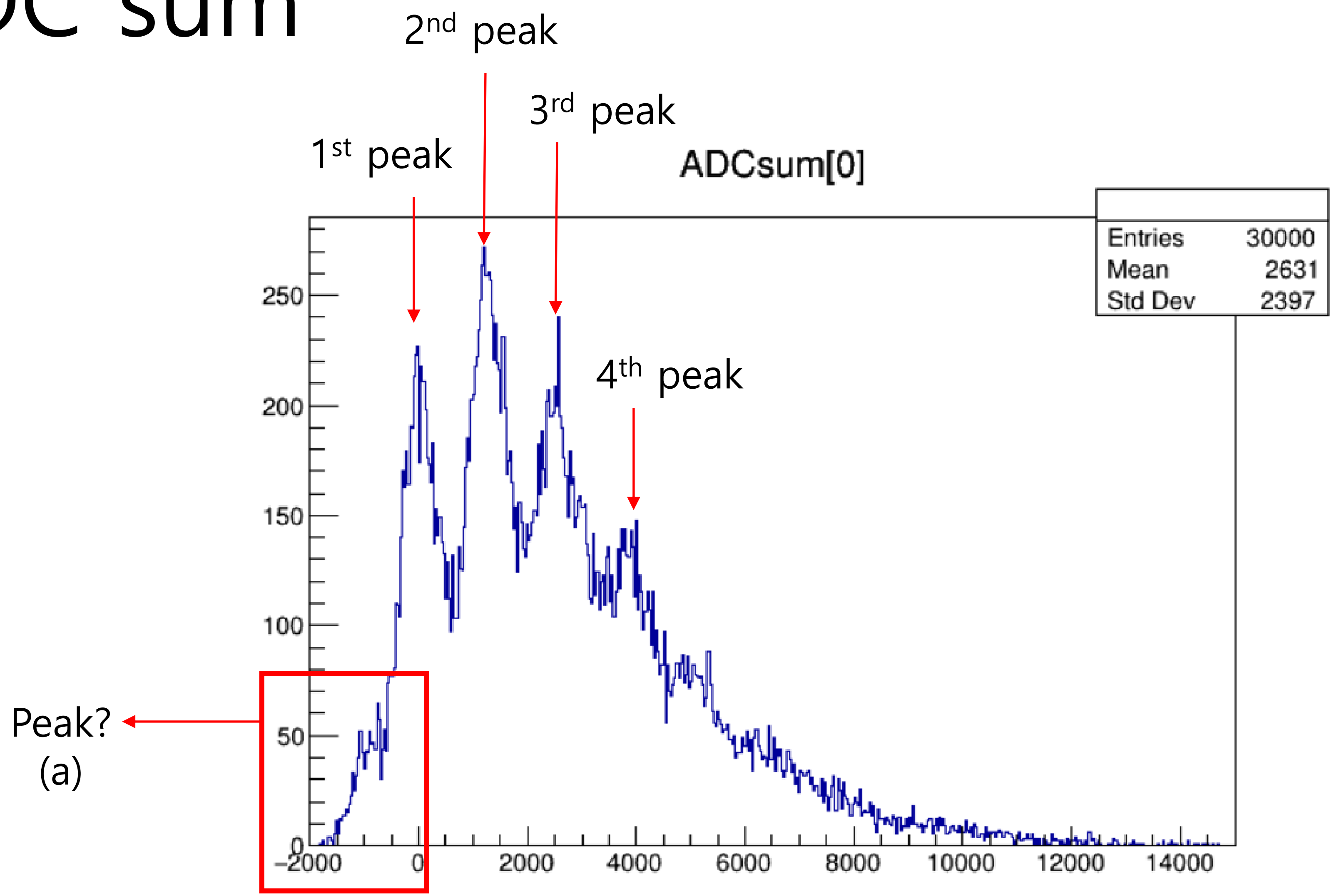




Pedestal

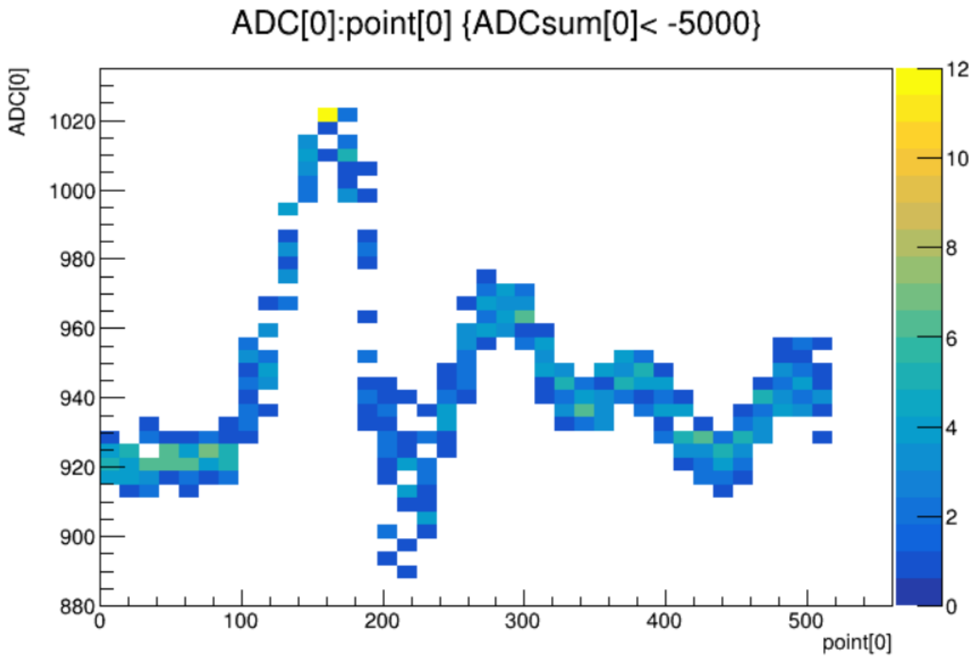

Pedestal[0] {Pedestal_rms[0] < 50}
Entries 15387
Mean 918.8
Std Dev 1.978
χ2 / ndf 47.1 / 23
Constant 415.3 ± 6.4
Mean 919.1 ± 0.0
Sigma 0.7613 ± 0.0190
0
50
100
150
200
250
300
350
400
450
910
912
914
916
918
920
922
924

ADC sum

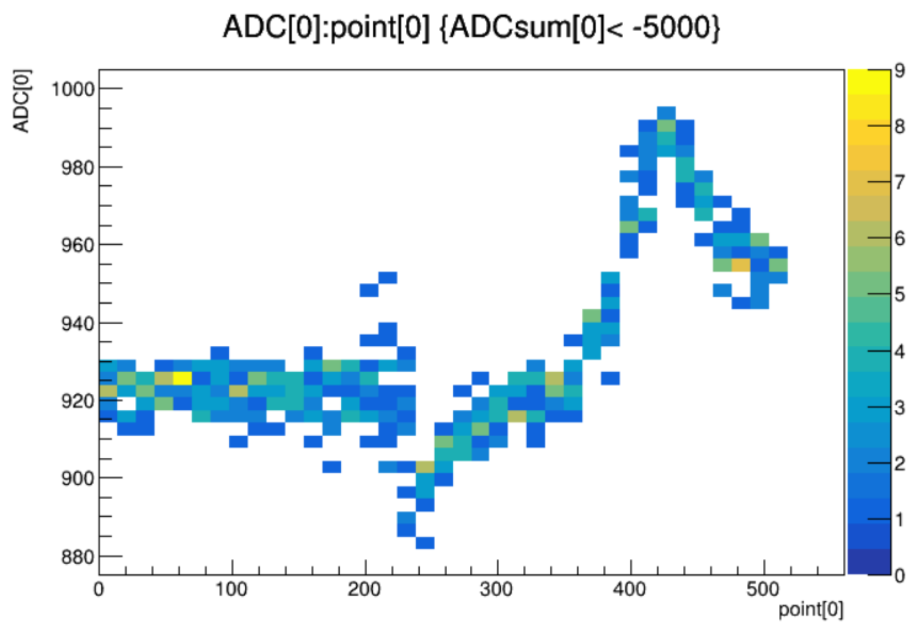


2nd peak
3rd peak
1st peak
ADCsum[0]
Entries 30000
Mean 2631
Std Dev 2397
4th peak
250
200
150
100
50
0
Peak?
(a)
−2000
0
2000
4000
6000
8000
10000
12000
14000

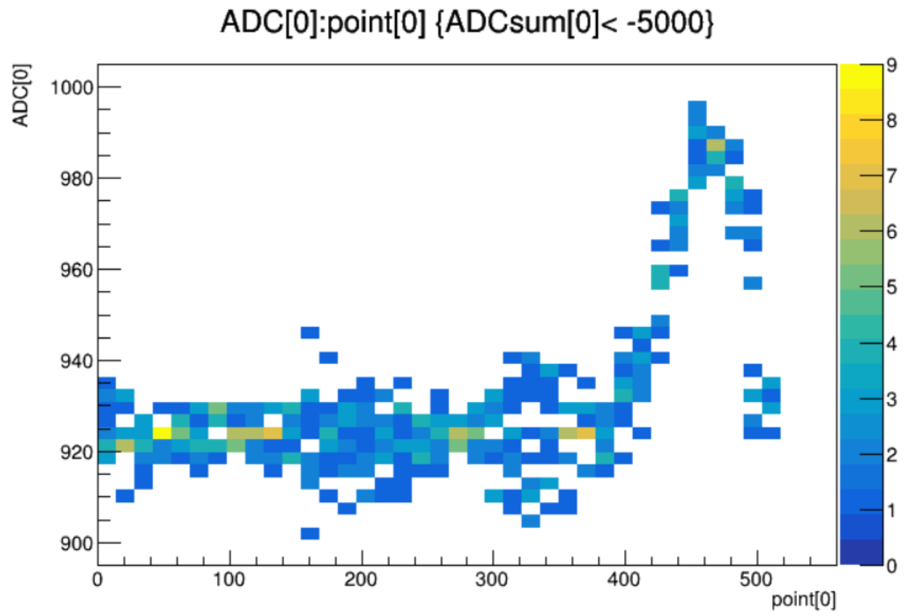
Waveform(a)



ADC[0]:point[0] {ADCsum[0]< -5000}
ADC[0]
point[0]

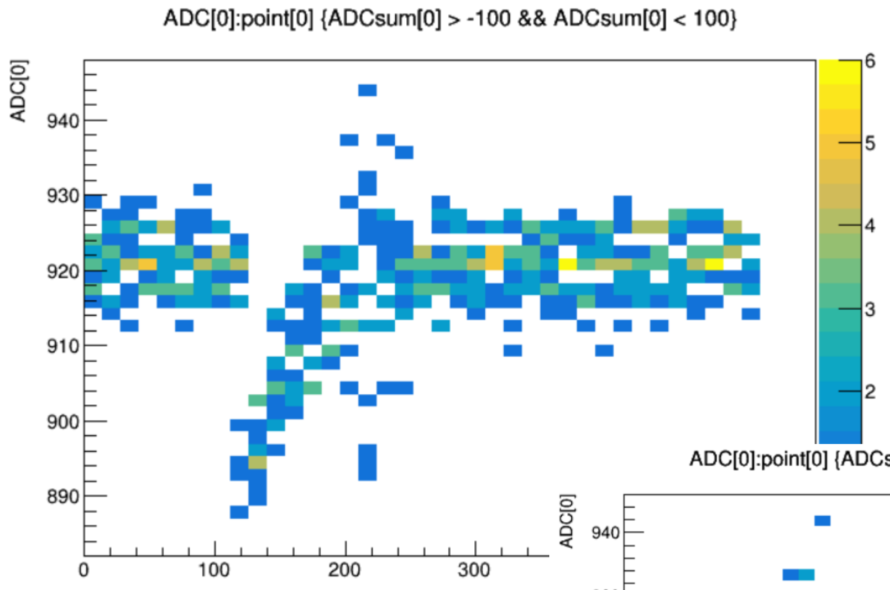


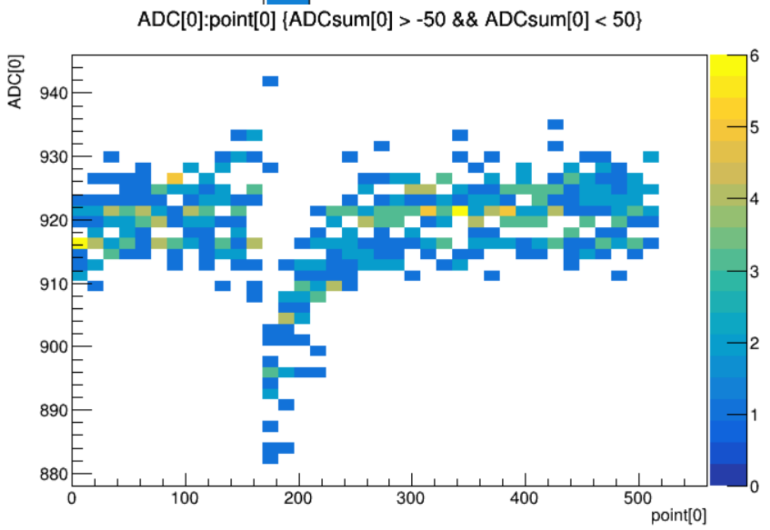
ADC[0]:point[0] {ADCsum[0]< -5000}
ADC[0]
point[0]

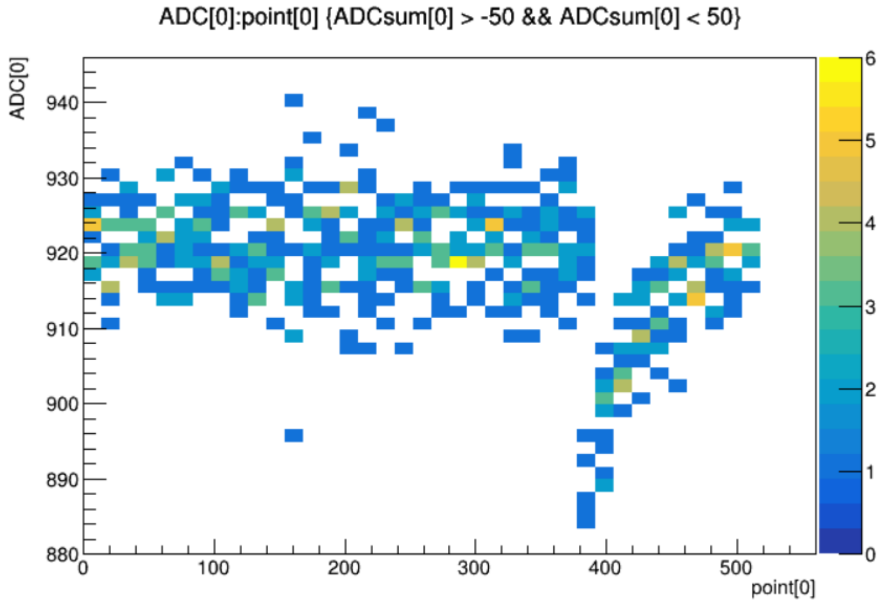


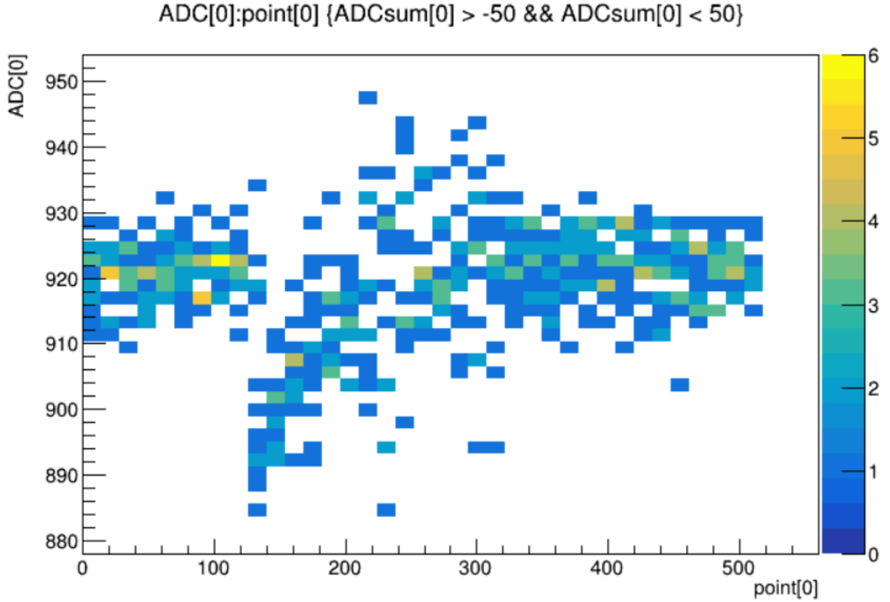
ADC[0]:point[0] {ADCsum[0]< -5000}
ADC[0]
point[0]

Waveform

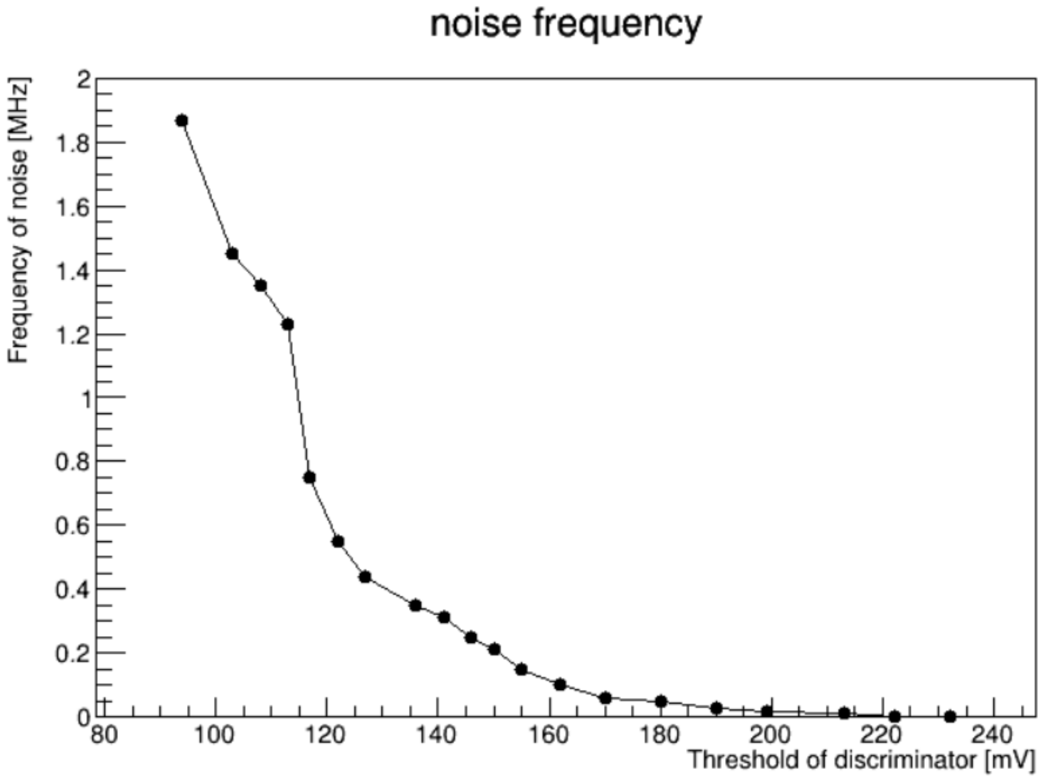

ADC[0]:point[0] {ADCsum[0] > -100 && ADCsum[0] < 100}
ADC[0]

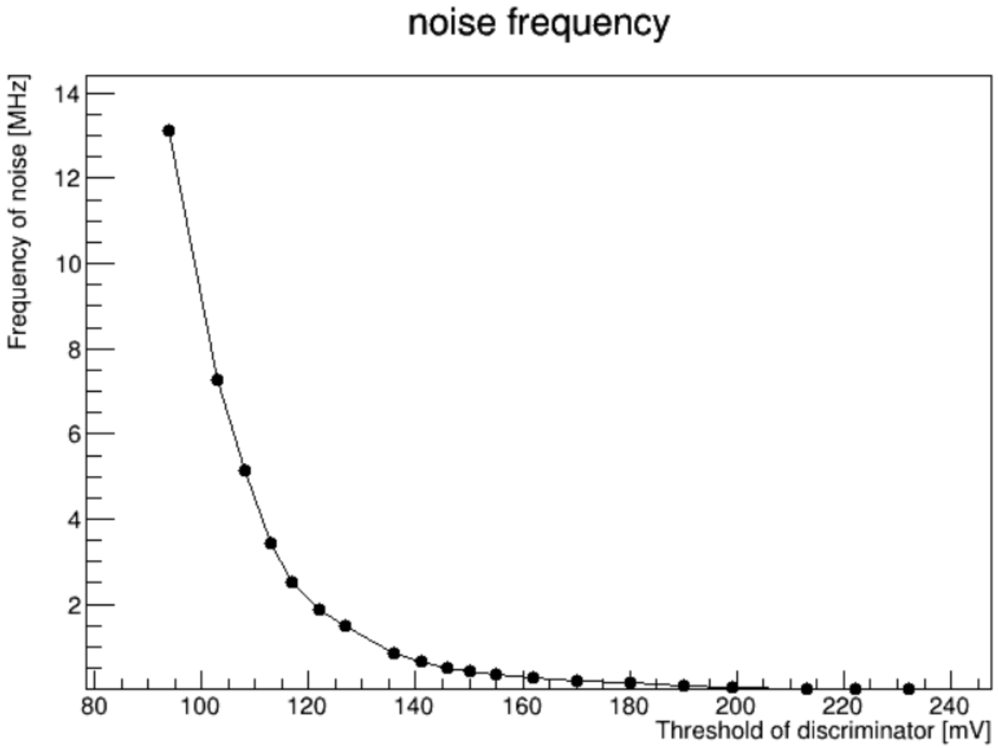

ADC[0]:point[0] {ADCsum[0] > -50 && ADCsum[0] < 50}
ADC[0]
point[0]


ADC[0]:point[0] {ADCsum[0] > -50 && ADCsum[0] < 50}
ADC[0]
point[0]


ADC[0]:point[0] {ADCsum[0] > -50 && ADCsum[0] < 50}
ADC[0]
point[0]

Noise Frequency





문제점

- Noise를 신호에서 제외시킨 뒤 count 해보기.
- Discriminator가 신호를 만들 때, 어느 정도의 시간이 지난 뒤에 다음 신호를 만들 수 있을 텐데, 그 시간 확인해서 coding에 반영하기