RD53b
S-Curve Fitting

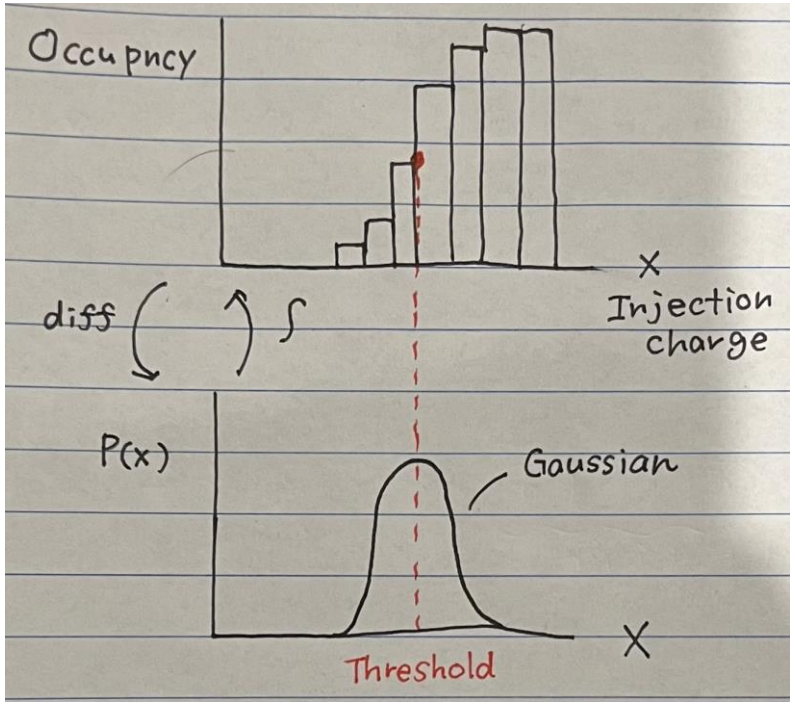
September 19th ,2022

LBNL M2 Taisei Kumakura

Calculation method

- Try to **calculate** threshold and noise by using s-cueve distribution **without fitting**

- Idea is
  - ⓪**Differential equation** of s-curve distribution must be gaussian
  - ①Get discrete derivative from occupancy distribution.
    - $dn(i) = n(i+1) - n(i)$
  - ②Expectation of discrete derivative(=mean of gaussian) must be threshold
    - $Threshold\ E = \sum_{i=0}^{N} x \cdot p(x)$
    - $Noise\ \sigma = \sqrt{Varaiance} = \sqrt{\sum_{i=0}^{N}\left(x^2 \cdot p(x)\right) - threshold^2}$
    - parameter
      - x: injection charge = $(i + 0.5) \times Vcalstep$
      - p(x):probability to get hits in derivative = dn(i)/n_injections



- What I did this week
  - update Std_analysis.cpp for calculation
  - and git push to yarr_devel

Updated code

- I added a code to use calculation method in Scurve_Fitter in Std_analysis.cpp
  - **calc_sucrve function** is come from another code called calc_scurve.cpp and it is included at the biggening of the code.

```
std::chrono::high_resolution_clock::time_point start;
std::chrono::high_resolution_clock::time_point end;
start = std::chrono::high_resolution_clock::now();
if (reverse) {
  calc_scurve(&x[0],histos[ident]->getData(),injections,vcalMax,vcalMin,vcalBins, par);
  //lmcurve(n_par, par, vcalBins, &x[0], histos[ident]->getData(), reverseScurveFct, &control, &status);
} else {
  calc_scurve(&x[0],histos[ident]->getData(),injections,vcalMax,vcalMin,vcalBins, par);
  //lmcurve(n_par, par, vcalBins, &x[0], histos[ident]->getData(), scurveFct, &control, &status);
}
```

->Try threshold scan using actual module

How calculation method work

tuning threshold=2000[e]

- step_size=10
- Average of 5times threshold scan of **HPK quad module**
  - Threshold and noise value are **almost same** as fitting results
  - Scan time and analysis time are **almost same** because fitting needs only few seconds
  - **Number of failed fit was decreased!**
    - Number of failed fit = ccalMax>Thre>vcalMin, vcalMax-vcalMin>Noise>0, fabs(par[2]-/injections-1)<0.1, Chi2>???

Fitting

V1.1 HPK	chip1	chip2	chip3	Chip4
Threshold[e]	1833±0.9	1853±19.5	1846±2.3	1958±46.1
Noise[e]	220.4±0.5	220±0.4	221.8±0.4	323.4±0.5
Failed Fit	612±12	327±11	824±13	2148±21
Scan Time[s]	97.699			
Analysis Time[s]	0.713			

Calculation

V1.1 HPK	chip1	chip2	chip3	Chip4
Threshold[e]	1834±22.4	1844±1.5	1845±1.6	1962±1
Noise[e]	226.6±0.5	225±0	229±0	341.2±0.5
Failed Fit	254±5.1	151±10	280±5.3	1404±10
Scan Time[s]	95.991			
Analysis Time[s]	0.797			

How calculation method work

tuning threshold=2000[e]

- step_size=20
- Average of 5times threshold scan of **HPK quad module**
- **big step_size increase failed fit**
  - **but Both step_size=10 or 20 is fine**

Fitting

V1.1 HPK	chip1	chip2	chip3	Chip4
Threshold[e]	1829±1.1	1840±2.3	1841±0.8	1976±1.4
Noise[e]	217±0	217.2±0.4	219±0	210±0.4
Failed Fit	693±9	359±10	943±14	3707±25
Scan Time[s]	52.347			
Analysis Time[s]	0.771			

Calculation

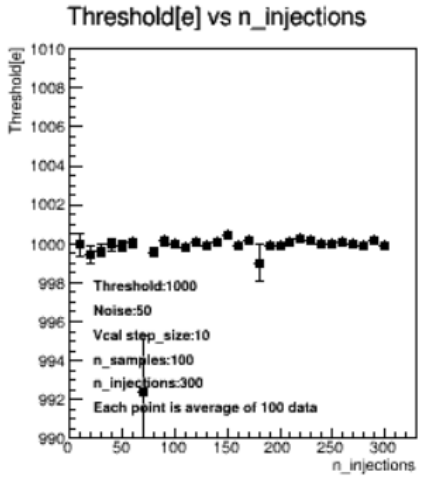
V1.1 HPK	chip1	chip2	chip3	Chip4
Threshold[e]	1836±17	1841±24	1838±0.5	1951±0.7
Noise[e]	231±0	228±0.4	232±0.4	350±0
Failed Fit	284±5	146±5	320±11	1478±11
Scan Time[s]	51.805			
Analysis Time[s]	0.790			

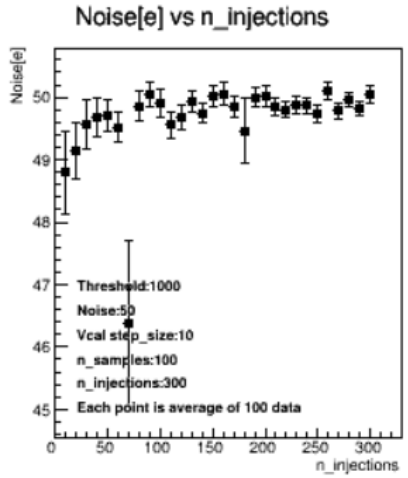
- **This is my last week to stay USA**
- **thanks for teaching carefully this 10 month!**

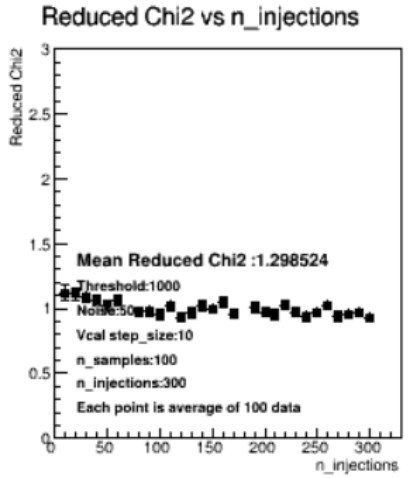
完成品(scurve_test.json_chi2＝1

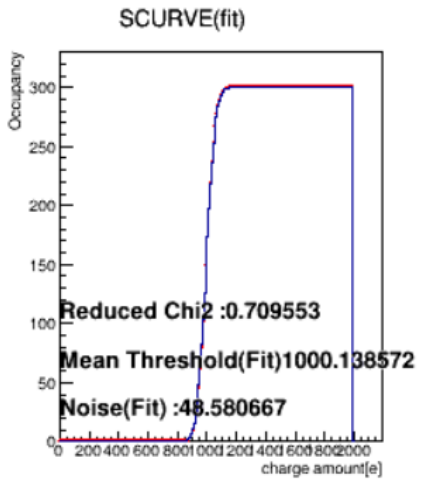
- 先週のアドバイス：Occupancyの縦軸を100に揃えないようにした
- 結果：理想的な場合では、Chi2がすべてのn_injectionで1に近くなった
  - →このChi2定義の方が有意義だと思うが、Timonからは反対されてて議論中

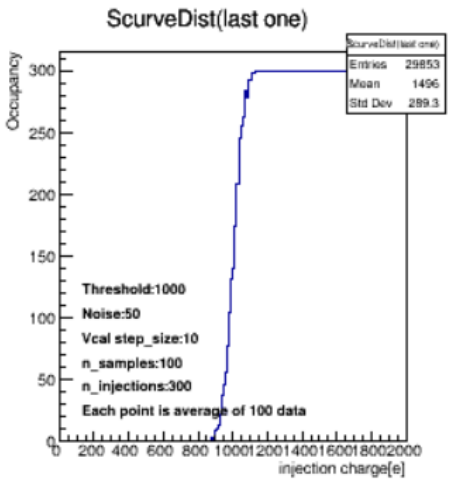
反対の理由を明確に

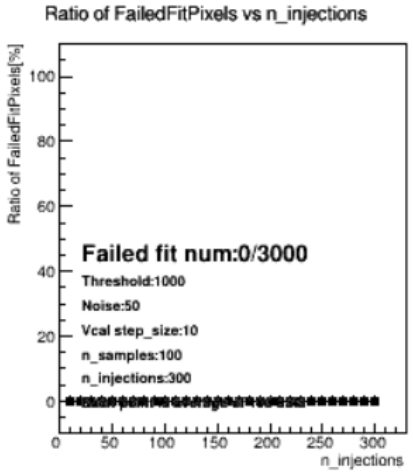






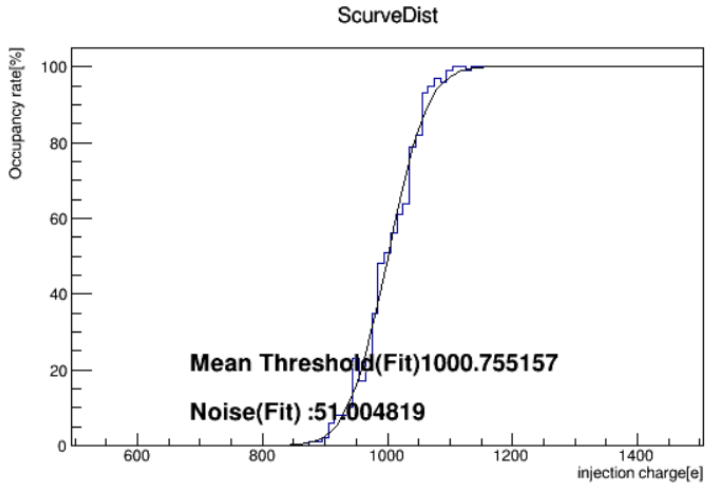




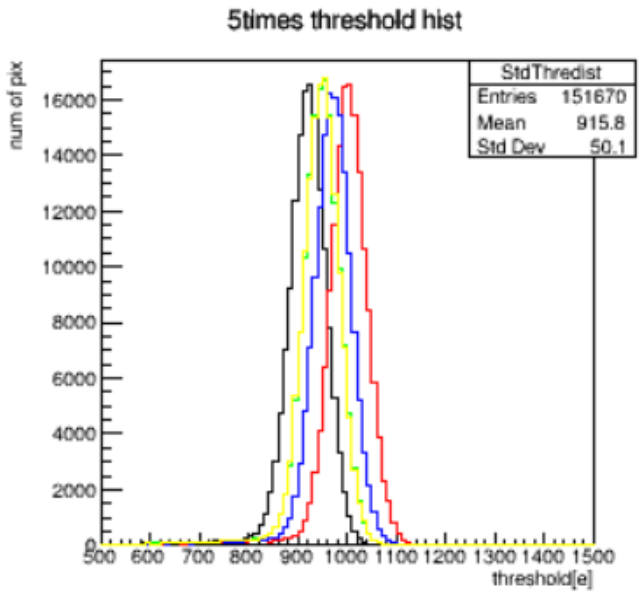


S-curve fitting method

- we usually use **s-curve fitting** to know threshold and noise
  - ideal s-curve: $P_{hit}(x) = \frac{1}{2}\left(2 - E_r f_c\left(\frac{x - threshold}{\sqrt{2} * noise}\right)\right) * 100$
  - Error function $E_r f_c = \frac{2}{\sqrt{\pi}} \int_x^{\infty} exp(-t^2)\, dt$



- But this fitting method has issues
  - **①Unstable results(=change results in every scan)**
  - ②Long calculation time



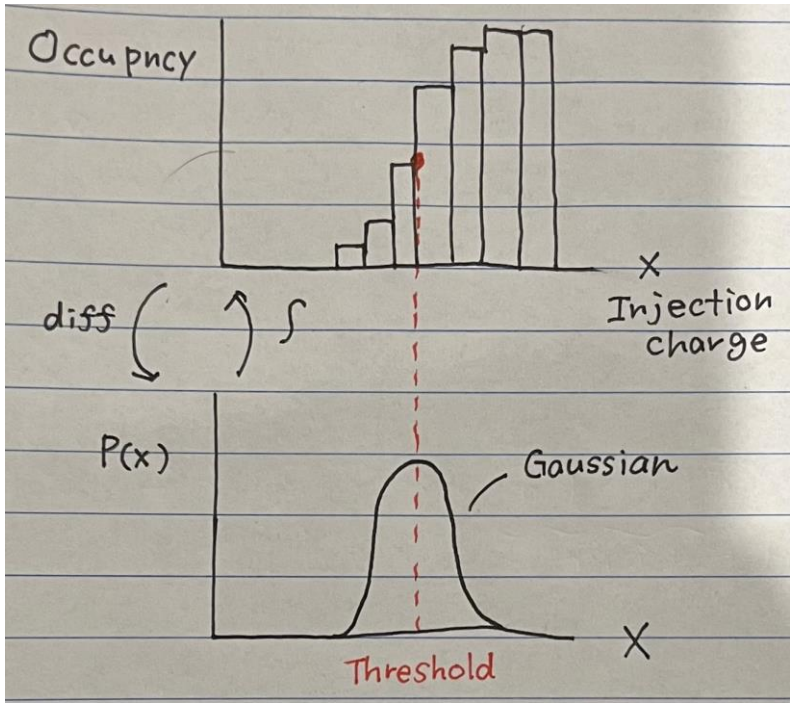
[TEST]
Run std-threshold scan **5times in same setting and same data** and compare threshold value in pixel by pixel
mean of 5times scan=953[e]
stddev of 5times scan=**32[e]**
module threshold has up to **83[e]** difference at max.

#	1	2	3	4	5
In time threshold[e]	915±50	998±53	943±50	966±52	943±50
untune	2031	1984	1941	1928	1930

Calculation method

- Try to **calculate** threshold and noise by using s-cueve distribution **without fitting**

- Idea is
  - ⓪**Differential equation** of s-curve distribution must be gaussian
  - ①Get discrete derivative from occupancy distribution.
    - $dn(i) = n(i+1) - n(i)$
  - ②Expectation of discrete derivative(=mean of gaussian) must be threshold
    - $Threshold\ E = \sum_{i=0}^{N} x \cdot p(x)$
    - $Noise\ \sigma = \sqrt{Varaiance} = \sqrt{\sum_{i=0}^{N}\left(x^2 \cdot p(x)\right) - threshold^2}$
    - parameter
      - x: injection charge = $(i + 0.5) \times Vcalstep$
      - p(x):probability to get hits in derivative = dn(i)/n_injections



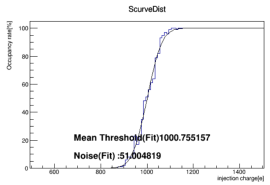
- Compare calculation results with results from fitting about
  - ①**Ideal** s-curve distribution samples
  - ②**Raw** s-curve distribution data from HPK module

Processing Time measurement

Make following **ideal samples** using gaussian->Use both **calculation and fitting method** and compare results

- ①10,000 different s-curve(n_pixels=10,000)
- ②153,600 different s-curve (n_pixels=153,600)
  - threshold:2000[e] / noise:150[e]
  - injection:100 / measurement range:1000~3000[e] / vcval_step:50[e]
  - fitting initialization parameter{threshold, noise, norm}={1000,50,50}

Make scurve samples



Start measuring time

Get threshold and noise by
①Fitting
②Calculation
loop by N_pixels

End measuring time

n_pixel	①10,000	②153,600
Threshold[e] by fitting	1000	1999
Threshold[e] by calculation	1000	2000
Noise[e] by fitting	49.9	150
Noise[e] by calculation	50	149
Time[s] by fitting	0.77	11.85
Time[s] by calculation	0.02	0.23

Results looks same

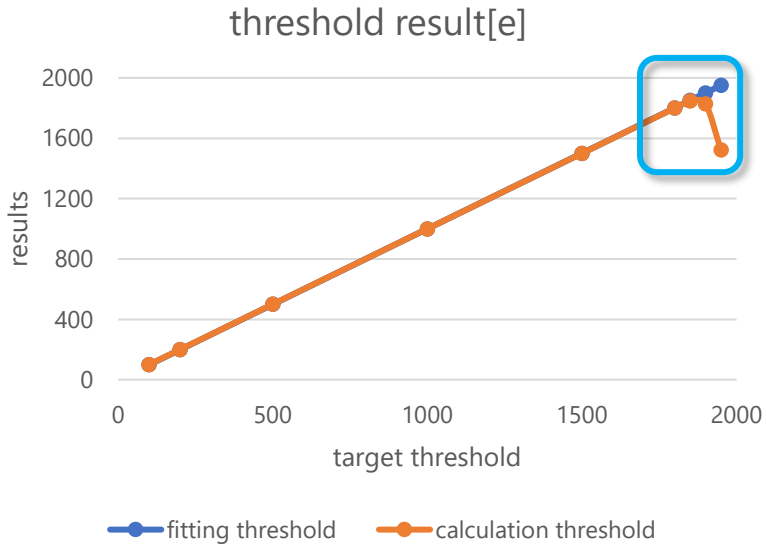
-97%

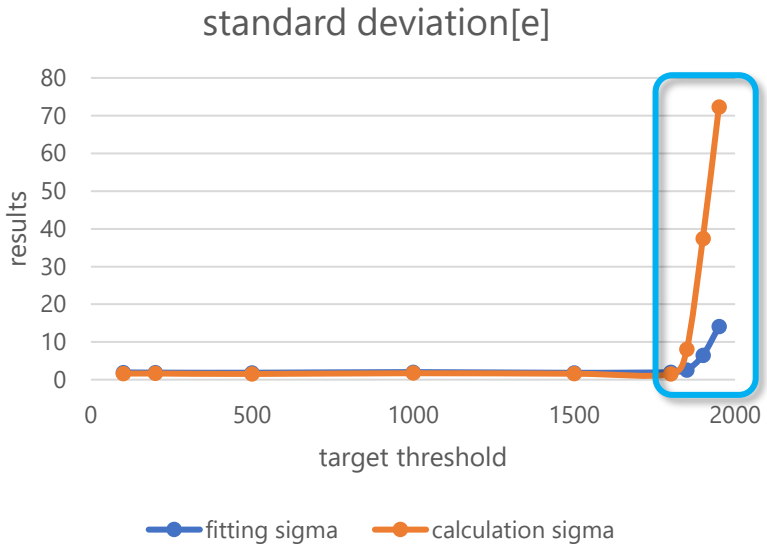
-98%

Calculation method can reduce 98% time

Change target threshold with fixing window

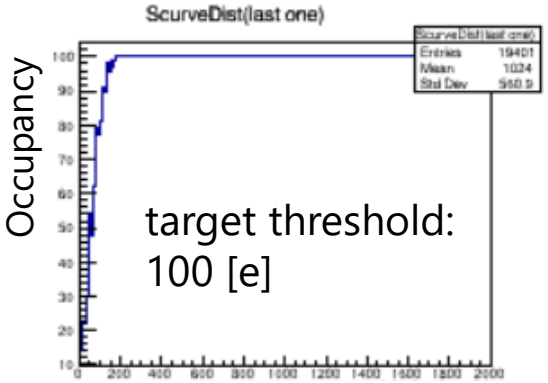
- 10 times measurement to get mean threshold and standard deviation
- Compare threshold results from fitting with one from calculation
  - Threshold: 1000[e] / noise: 50[e] / n_injections: 100 / step_size: 10[e] / window:0~2000[e]

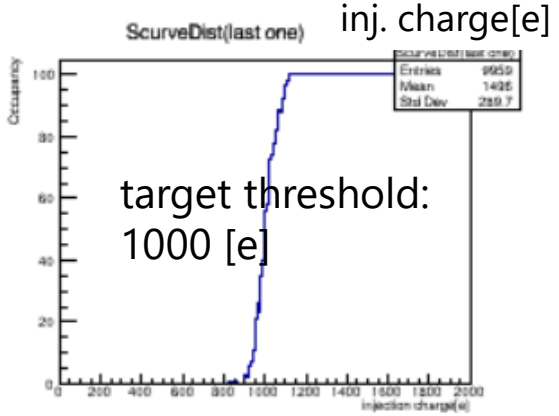


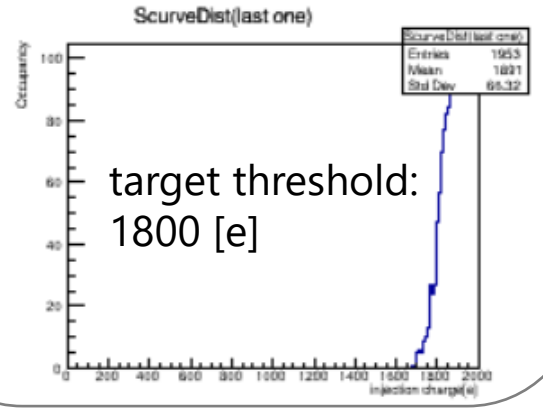


when target threshold is around the right part of window, calculation results get worse

samples



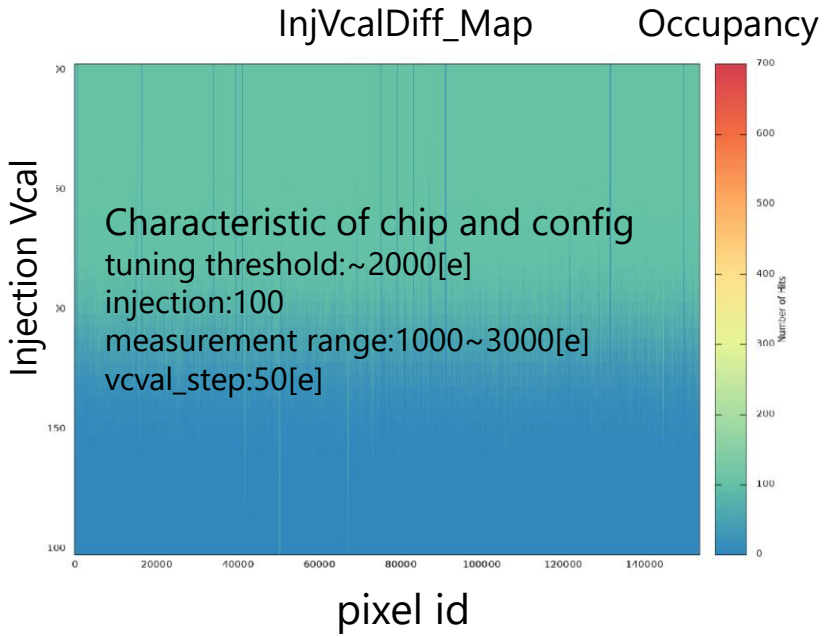




Raw data analysis

- Get real data from InjVcalDiff_Map.json(HPK Quad module chip2)
  - it is **more practical**...some pixels are noisy and have no data
  - -> check the quality of calculation and fitting

- Results



Threshold[e]	Fitting	Calculation
Raw Data	1832±20	1866±56
Ideal Sample	2000±7.5	2000±6.5

Noise[e]	Fitting	Calculation
Raw Data	134.2±4.0	198±13
Ideal Sample	149.5±8.4	150.4±5.9

Time[s]	Fitting	Calculation
Raw Data	25.27	0.14
Ideal Sample	25.25	0.13

Different output
->Because threshold value get smaller than target threshold through tuning process

Different output
->Why??

Calculation time is much smaller than fitting!
-~99%

Fitting Parameter sweeping

- Fit to Raw data(HPK module) with various initializing parameters
  - threshold:2000[e] / noise:150[e] /injection:100 / measurement range:1000~3000[e] / vcval_step:50[e]

Threshold[e]±stddev

Threshold[e]/noise[e]	100	500	800	1000	1500
20	- Fitting	-	-	1854±117	1861±73
50	- Failed	-	1969±85	1866±60	1861±73
100	-	1960±13	1866±59	1865±60	1861±73
200	1941±29	1866±60	1866±60	1866±60	1861±73

Time[s]->**no tendency?**

Threshold[e]/noise[e]	100	500	800	1000	1500
20	-	-	-	25.5	14.9
50	-	-	26.4	18.2	14.5
100	-	12.3	36.2	19.7	14.3
200	19.4	45.1	26.1	17.8	16.7

Noise[e]±stddev

Threshold[e]/noise[e]	100	500	800	1000	1500
20	- Fitting	-	-	194±25	195±25
50	- Failed	-	213±91	196±16	196±22
100	-	200±18	197±14	194±28	195±18
200	198±17	196±14	197±13	196±16	197±14

FailedFitPixel(threshold<1000e or threshold>3000e)

Threshold[e]/noise[e]	100	500	800	1000	1500
20	- Fitting	-	-	162	160
50	- Failed	-	146804	2243	160
100	-	2240	2240	2242	163
200	135642	2444	2244	2244	166

Threshold scan in HPK module

- Failed Fit Definition
  - After tuning 2000[e] and use same config
  - data(6567~)
  - Mean and stddev of 5times measurement
  - Almost same result…

素で
Failed Fitが 0 になる？
→クライテリアの確認
Chi2のばらつきが大きいモジュールで確認
結局何の時間が変わったのか
FiledFitがなくなることが最大のメリット

Vcal range:0~3000[e]
Vcal step: 50[e]

V1.1 HPK chip1	Fitting	Calculation
Data	6577	6573
Threshold[e]	1846±14	1847±18
Noise[e]	185±0.4	187±0.4
Failed Fit	54	51
Scan Time[s]	154.4	154.5
Analysis Time[s]	0.8	0.6

Vcal range:0~3000[e]
Vcal step: 100[e]

V1.1 HPK chip1	Fitting	Calculation
Data	6582	6588
Threshold[e]	1847±19	1847±17
Noise[e]	185±0.4	187±14
Failed Fit	55	50
Scan Time[s]	79.1	79.0
Analysis Time[s]	0.75	0.77

What is Difference?

- The way to store data is different

- ①My way
  - Push all pixel data from VcalDiff_Map.json to **vector x,y**(x=Vcal, y=occupancy)
  - **Read x, y data from vector** and fitting altenately

- ②StdAnalysis way
  - Push 1 pixel data from threshold scan to float(?) x,y
  - use x, y data and fit alternately. after that, we update x and y for next pixel

- Pushing data to vector and reading data from vector affect processing time a lot?

- なぜGITにpushできないのか
  - egroupにsubscribeしてなかったから
- ✓測定時間がAnalysisに来るのかScanに来るのか確認する→どっちも見る必要がある？
  - scanはscandone-scanstart：各ピクセルの計算はscanの中で行われている。
  - analusisはalldone-processor done。おそらく全ピクセルデータからMeanやFailedFitを出している。

- ✓10秒程度の短縮じゃほとんどスキャン時間に聞いてこない
  - Time per fit distを見ると大体1ピクセルあたり10us
  - Calculationなら1ピクセルあたり2usのはずだからちょっと早くなる
  - 153600ピクセルの処理だとCalculationで0.3秒、Fittingで1.5秒しかかからない→インジェクションとかの時間に比べたら些細。全然変わらない。

- 実際のHPK測定データもスキャンでやるんじゃなくてベクターに詰めてからやってみる
- ✓Fitting Failedの条件を確認する
  - ccalMax>Thre>vcalMin, vcalMax-vcalMin>Noise>0, fabs(par[2]-/injections-1)<0.1, Chi2>???
  - おそらくFailedの50はもともと閾値が小さくチューンされたものとかが含まれている
  - Chi2の条件を戻して一回Fitの結果を求めてみる
- ✓Chi2のせいでfailed fitが減ってたんじゃないか
  - Chi2のクライテリアを加えてFittingしてみる→FailedFitは2くらいしか変わらず。

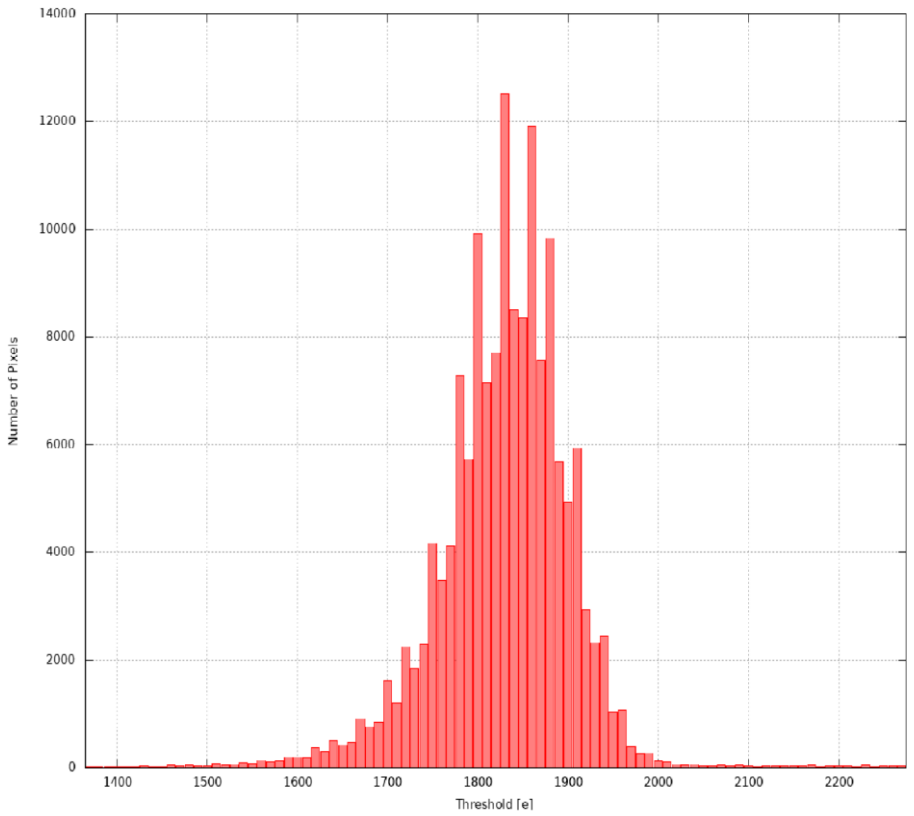
- ✓Fittilg Failedがもっと多いデータを取ってきて確認する
  - 他のチップを見てみる→4チップ同時に読み出したらFailedFitも閾値もバラバラになった。
    - 消費電力が大きい状況程ばらつきが増えるのか？

  - 昔の5回の測定データは何のモジュール？→閾値1000だから3DFBK

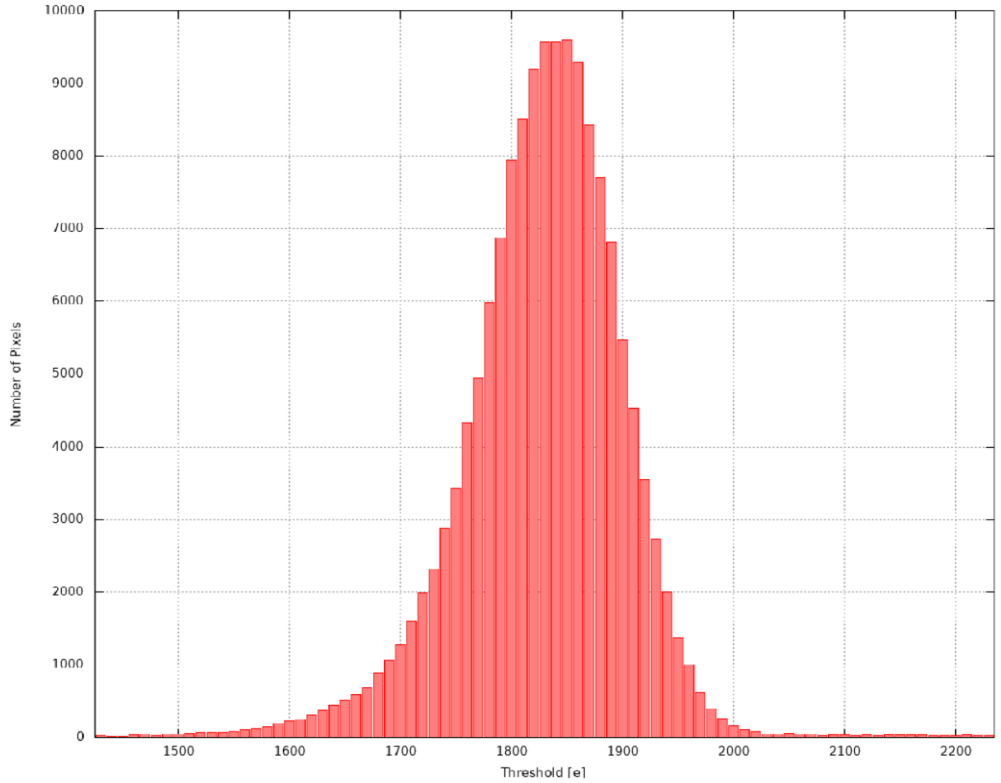
Threshold dist

- あ

Calculation



Fitting



4チップ全ての読み出し

- 5回の平均を調べよう→FailedPixelが安定すればばらつきも減るのでは？
- バラつきが同じくらいなんだとしたら、きっと3DFBKの問題

V1.1 HPK	chip1	chip2	chip3	Chip4
Data	6611			
Threshold[e]	1855	1864	1857	2006
Noise[e]	224	224	226	329
Failed Fit	783	388	1034	3480
Scan Time[s]	98715			
Analysis Time[s]	738		fit with chi2	

V1.1 HPK	chip1	chip2	chip3	Chip4
Data	6613			
Threshold[e]	1834	1845	1850	1982
Noise[e]	221	221	222	324
Failed Fit	622	346	841	3171
Scan Time[s]	97699			
Analysis Time[s]	713		fit w/o chi2	

V1.1 HPK	chip1	chip2	chip3	Chip4
Data	6614			
Threshold[e]	1835	1844	1847	1963
Noise[e]	227	225	229	341
Failed Fit	259	169	288	1410
Scan Time[s]	95991			
Analysis Time[s]	797		Calc w/o chi2	

- git push origin calc_scrve branch

```
-bash-4.2$ git push origin calc_scurve
Username for 'https://gitlab.cern.ch': tkumakura
Password for 'https://tkumakura@gitlab.cern.ch':
fatal: Authentication failed for 'https://gitlab.cern.ch/YARR/YARR.git/'
-bash-4.2$ git push origin calc_scurve
Username for 'https://gitlab.cern.ch': tkumakur
Password for 'https://tkumakur@gitlab.cern.ch':
Counting objects: 11, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.29 KiB | 0 bytes/s, done.
Total 7 (delta 4), reused 0 (delta 0)
remote:
remote: To create a merge request for calc_scurve, visit:
remote:   https://gitlab.cern.ch/YARR/YARR7-/merge_requests/new?merge_request%5Bsource_branch%5D=calc_scurve
remote:
To https://gitlab.cern.ch/YARR/YARR.git
 * [new branch]      calc_scurve -> calc_scurve
-bash-4.2$
```

```
-bash-4.2$ git log
commit 8c1d69bba95b0792565eadb5ecb0c119d1563d57
Author: Taisei Kumakura <tkumakura@pigwhale.dhcp.lbl.gov>
Date:   Fri Sep 16 17:34:57 2022 -0700

    calculation method by taisei
```

変更点

```
+void calc_scurve(const double *x, const double *hitdata, int injections, int vcalMax, int vcalMin, int VcalBins, double
+  double T=0,N=0;  //T=threshold, N=noise
+  int step_size =10*(vcalMax-vcalMin)/VcalBins;
+
+  //time measurement start
+  clock_t start =clock();
+
+  // [Calculation]Start
+  double x_v=0,p=0,T_ele=0,Variance=0;
+  for (unsigned j=0; j<VcalBins-1; j++) {
+    x_v = j*step_size+vcalMin*10;
:
@@ -784,12 +787,12 @@ void ScurveFitter::processHistogram(HistogramBase *h) {
                     }

                     double chi2= status.fnorm/(double)(vcalBins - n_par);
-
                     if (par[0] > vcalMin && par[0] < vcalMax && par[1] > 0 && par[1] < (vcalMax-vcalMin) && par[1] >= 0
-                         && chi2 < 2.5 && chi2 > 1e-6
-                         && fabs((par[2] - par[3])/injections - 1) < 0.1) {  // Add new criteria: difference between
-                       FrontEndCfg *feCfg = dynamic_cast<FrontEndCfg*>(bookie->getFe(id));
-                       thrMap[outerIdent]->setBin(bin, feCfg->toCharge(par[0], useScap, useLcap));
+                       && fabs((par[2] - par[3])/injections - 1) < 0.1) {
+                       //chi2 requirement is not needed because chi2 is meaningless value @ taiei
+                       // Add new criteria: difference between 100% baseline and 0% baseline should agree with number o
+                       FrontEndCfg *feCfg = dynamic_cast<FrontEndCfg*>(bookie->getFe(id));
+                       thrMap[outerIdent]->setBin(bin, feCfg->toCharge(par[0], useScap, useLcap));
                        // Reudce effect of vcal offset on this, don't want to probe at low vcal
                        sigMap[outerIdent]->setBin(bin, feCfg->toCharge(par[0]+par[1], useScap, useLcap)-feCfg->toCharg
                        chiDist[outerIdent]->fill(status.fnorm/(double)status.nfev);
```

- ✓②一つのモジュールだけ動かしたときと4つのモジュールを動かしたときのthreshold, noise, failed fitを比べる
  - 同時に動かしたほうが安定しない気がする
  - →というか供給電源が変わるから、スレッショルドの値が小さくなる。おそらくプリアンプが大きくなったおかげで、小さいインジェクションに対しても大きな反応が来る→thresholdが小さくなる

- ✓①step sizeを大きくして測定する
  - 大きいほうが安定しなそう。
  - 50でも同じような結果が出る？
- マージリクエストする
  - →Angiraに聞く→MTGまでに完成させる

- ③ｐのグラフを確認する。→きれいなガウシアンなら、Ｍｅａｎを取るだけでいいんじゃないか

- ③なぜ５nsなのかをしっかり検証する
  - HPKのディレイをしっかり測定して結果をまとめる

- ④Chi2の議論をする
  - 実際のコードのsigmaの定義を変更しに行く