PMT Calibration measurement Manual(日本語版)

1．Gian

2．Linearity

3．2D relative scan

4．2D absolute PDE scan

5．2D Gain map

6．Re-start Latafa PC

Rev1.　June 20, 2020

0. はじめに

このマニュアルは 2D-scan　box を用いた測定方法の詳細を記載する。元になるのは、Lasse が記載した「DEgg calibration Manual」を元に、それを日本語化し、より詳細に測定や解析方法を記載したものである。

主な使用機器

Laser: Hamamatsu c10196 : 400nm pulse width 60ps

Filter: 固定　1% + 回転式 0.1%、1%, 5%, 10%, 50%, 100%

Oscilloscope: R&S: RTO-1044

Power Supply: KEYSIGHT E3631A + HV board

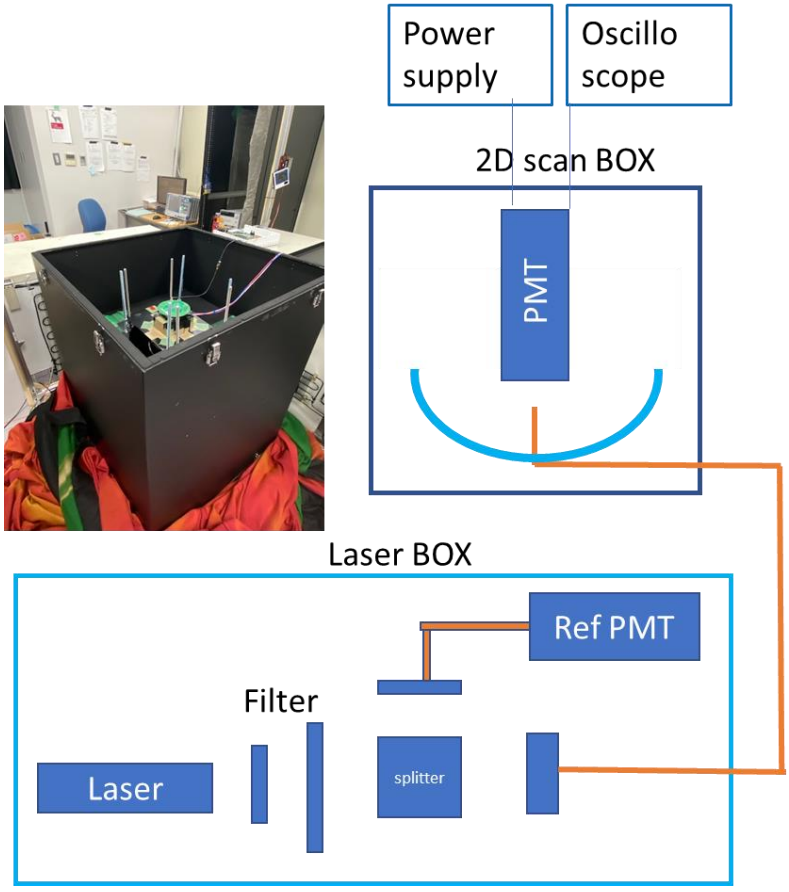
Helmholtz coil

PC: Latafa

この中に、データ収集用と解析用のコードがある。

データ収集は Jupiter notebook　解析はターミナルから python コードを動かす。

設備構成図



注）レーザ BOX を開けて作業するときには、レーザ電源を切る事。開けて作業が必要な時は、レーザ用のサングラスをつける事。

１．PMT　Gain 測定マニュアル

１)セッティング

以下、作業については、チェックシートに記載すること。

① PMT にソケットをつける。

② PMT を固定治具につける。写真にあるように、固定治具とガラス面の差が 13mm になるようにする。HV ボードをつけ、２D スキャンボックスに据え付ける。水平を確認する。

③ 電源ケーブルと、モニタ用の同軸ケーブルを接続する。オシロを ON
オシロの CH2 に reference の PMT、CH3 に測定用の PMT の信号、CH4 にレーザの外部トリガ信号を入れる

④ 蓋をして、カーテンをかける。ヘルムフォルツコイルの電源を入れる。　X：18V,　Y:12V
これは地磁気の影響をキャンセルするためである。

⑤ 電源を ON にして、＋26V 側（制御電源）を５V にする。この時電流はほぼゼロ。
次に 6V 側（高圧供給側）を、0.1V 単位で上がるようにして、ゆっくり上げる。こちらは高圧になり、PMT にはこれの 400 倍の電圧がかかる。3.6V（1440V)。電流値(35mA ぐらい)をシートに記載

⑥ Reference　PMT を使う時は、1560V にリミットをセットして、HV OUT にして、ゆっくりあげて、1550V にする。

⑦ レーザ BOX の箱をあけ、フィルタが 0.1%(回転式)＋1%になっている事を確認。(1%が抜いている時がある。）　箱を閉める。

⑧ オシロを設定する。
水平軸を―100ns～300nS をカバーするように。Resolution は 400ps
CH3 は２0mV/div で、後でレーザによる光信号で、調整する。
トリガは CH4 の立ち下がり

⑨ レーザのスイッチを ON にする。繰り返しは 100Hz。強度のダイヤルを 5.8（元のマニュアル値）
ただし、オシロで見て、弱い時には、12 ぐらいまであげる。電灯を消して、オシロの画面を見る。

２）Gain 測定

①Jupiter Notebook で次を開く。(PC 上に開いている場合あり)

/home/icecube/measurements/daq/notebooks/gain_calibration.ipynb

自分用に名前を変更して保存。(元のは標準のため)

②上から順番に実行。2 番目ので、オシロからレスポンスがある事を確認

SPE レベルだと、１０回に１回ぐらいしかパルス波形がでず、後はノイズ波形

4 番目のでコードで、シングルの波形がとれている事を確認する。波形が大きすぎたり、小さすぎたりする時は v/div を調整する。通常は 20mV/div

③テスト Run

HV vs Gain scan の所の PATH（データ保存用）を変更

/home/icecube/measurements/daq/data/今回のデータ用デレクトリ/PMT と日付からの名前/

テストは degg_sq0987_20210517r r をつける

Vctr：Test RUN 3.2、3.7、4.2 の 3 点で

10７ゲインは、PMT によって、3.5V～4.5V の範囲でばらつくので、どのあたりかのめぼしをつけるテスト RUN を 1 回目にして、2 回目に詳細 RUN をする。テスト RUN はファイル名に r をつける

本番は r をつけない

2 回目：どのあたりかのめぼしがたったら、その近くで、0.1V 刻みで 5 点とる

例えば 4.0V より少し下そうなら、〔3.7、3.8、3.9、4.0、4.1〕

④新しいものは慣らしの 1 時間がいるので、Burn-in が 60＊60 がはいる

⑤RUN させる。データ取得は、慣らし 1 時間＋1 時間＝2 時間かかる

⑥終了後は、レーザを OFF、HV とコイルの電源を OFF

３）. Analysis

①PC で terminal を開く。Dir は

/home/icecube/measurements/analysis/workspace/gain/

②database の新しいデータ領域を作る

$ create_new_degg_entry 〔Enter〕

PMT の番号と日付、HV ボードの型番を問い合わせてくるので入力する

テスト run は、日付の後に r を入れる。 20210517r

本番は r をつけない

③ヒストグラムを作るコードを電圧ごとに走らせる。 PMT 番号__日付(20202205) 電圧を入れる。

(例)

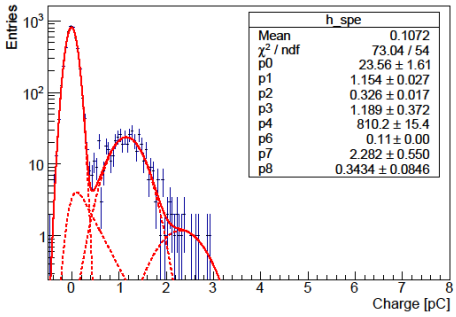
python run.rev5.py degg_sq0244_20201105 3.5V

python run.rev5.py degg_sq0244_20201105 3.6V

注：最後に Vが付く、テスト run は日付の後に r がつく

④ヒストグラムの図を確認。ターミナルから

Evince figs/gain curve PMT 名_日付 電圧 hspe.pdf



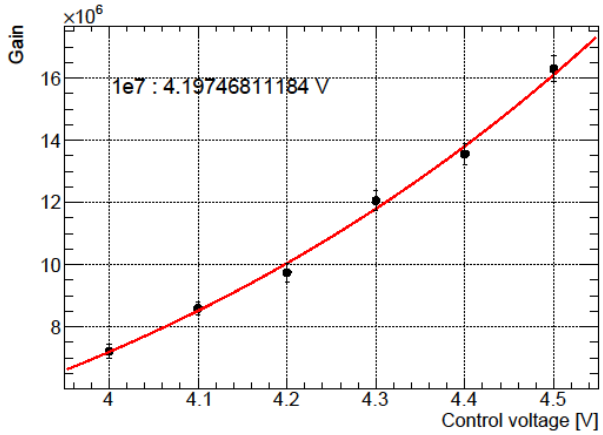
このように、ゼロ近くで鋭い高い山、その右で２番目の山があること

PC のファイル（Windows のエクスローラ相当）で、figs のデレクトリ内にあるのをクッリックしても見ることができる。

　この図の２番目のピークが２－３と大きくなりすぎているとレーザが強すぎる

⑤ゲインカーブの図を表示する。ターミナルより、ＰＭＴ番号、日付、電圧を入れて走らせる

python get_gain_curve.py degg_sq0244_date 4.0 4.1 4.2 4.3 4.4 4.5



これより、１X10^{7}のゲインを得る電圧が求められる

テスト run のフィアルは削除する。

４）．コードの内容

4.1）　測定コード：gain_calibration.ipynb

　各測定電圧で、signal_line（CH3）：メイン PMT　ref_line（CH2）：referencePMT の波形を NLOOP＝5000 個の取る。データは時間と電圧

データ格納

　/home/icecube/measurements/daq/data/今回のデータ用デレクトリ/PMT と日付からの名前/

例：/home/icecube/measurements/daq/data/high_bright/degg_sq0975_20201211/signal_line

　　/ref_line

　一つの電圧で、5000 波形ずつ取る。wf_3.5V_c00000.npy ~ c49999.npy

　　16kB x 5000 = 80MB　Signal と Ref で　160MB　電圧 6 点測ると、960MB～**1 GB**

ファイルサイズが大きいので、定期的に削除すること。波形データは、USB-HDD に移して、本体側は削除する

4.2)　解析コード

run.py

rev5 は ReferencePMT を含めない

これは次からなる

① auto_charge_calc_rev5.py　：波形よりチャージを計算

② auto_make_charge_dist_rev5.py :チャージデータを root のファイルにする

③ spe_fitting.py　　：SPE のヒストグラムとフィッテングの図を作成

① 　python3 charge_calc_auto_gate.py

wfana.get_charges(path_sig, label, 80e-9, -100e-9, 60e-9, 5000)

で、(波形ファイルをよびだし、積分開始時間、ベース設定のペデスト時間、積分期間、波形数)を設定している。この区間があっているか確認。これで、波形ごとのチャージを計算

その電圧での、チャージ 5000 個のデータファイルを作る

データは、次に収納される

Charges/charges_PMT 名_電圧.dat

Charges/charges__ref_PMT 名_電圧.dat

② 　auto_make_charge_dist_rev5.py

① で作ったチャージのリストのデータ呼び出し、ヒストグラムの root のファイルにする

ファイルは、次になる

analysis/workspace/gain/PMT 名/rooftiles/dists_電圧.root

③ 　spe_fitting.py

フィッテングをする。root で書かれている

$$SPE_{fit} = A_{pde}\exp\left(-\frac{(x-0)^2}{2\sigma_{ped}^2}\right) + A_{exp}erf\,(x)\exp(-x/\tau) + A_{SPE}\exp\left(-\frac{(x-\mu_{SPE})^2}{2\sigma_{SPE}^2}\right) + A_{2PE}\exp\left(-\frac{(x-\mu_{2PE})^2}{2\sigma_{2PE}^2}\right)$$

ファイルは、figs/gain にあり、電圧の数だけできる。ファイルサイズは 30kB 程度と小さい

PMT 名_日付 電圧 hspe.pdf

‘get_gain_curve.py

ゲインカーブの図を表示。

ファイルは gain_curve_PMT 名.pdf

チェックシート

PMT _____ **HV board** _____ **HV module** _____

Data directory name _____

Your name _____

Gain calibration Date _____ Time _____ Room temperature _____

Settings

- ☐ Helmholtz coil X 18.0 V Y 12.0 V
- ■ Low voltage _____
- ■ Current when $V_{\mathrm{ctrl}} = 3.5$ V _____
- ☐ Filter 1 % + 0.1 %
- ☐ Intensity 5.8
- ☐ Frequency 100 Hz
- ☐ Resolution 500 ps
- ■ Horizontal range (MUST include −100 ns – ∼ 150 ns) _____
- ☐ Trigger low enough
- ■ Scan control voltages ( ) ( ) ( ) ( )

Analyses

Gain scan

- ■ Gate start time _____
- ■ V_{ctrl} @1e7 gain _____

V_{ctrl} @1e7 gain

- ■ SPE mean charge _____
- ■ SPE peak height _____

Linearity & afterpulse Date _____ Time _____ Room temperature _____

Settings

- ☐ Helmholtz coil X 18.0 V Y 12.0 V
- ■ Low voltage _____
- ■ Control voltage (MUST V_{ctrl} @1e7 gain) _____
- ■ Current _____
- ☐ Intensity 14
- ☐ Frequency 10 Hz
- ☐ Resolution 1 ns
- ☐ Horizontal range −20 μs – 20 μs

Analyses

- ■

２．Linearity measurement and Pre-, After- and Late pulses

１)セッティング

①PMT にソケットをつけ、HV ボードをつける

②固定治具につけ、２D スキャンボックスに据え付ける。水平を確認する。

③電源ケーブルと、モニタ用の同軸ケーブルを接続する。オシロを ON

オシロの CH3 に PMT の信号、CH4 にレーザの外部トリガ信号を入れる

④ 蓋をして、カーテンをかける。

⑤ 電源を ON にして、２６V 側を５V にする。この時電流はゼロ。

次に 5V 側を、0.1V 単位で上がるようにして、ゆっくり上げる。$1X10^7$ のゲインの電圧にする。

⑥ レーザ BOX の箱をあけ、最初は固定の 1%、回転式は、0.1%になっている事を確認。　箱を閉める。

⑦オシロを設定する。

水平軸を―20μs～+20μS をカバーするように。Resolution は 1ns。

CH3　通常：大強度のパルスも見えるように、1V/div

ただし、1%+0.1%では 1V/div ではほとんど見えないので、下げる

0.1%(20mV/div)、　1%(100mV/div)、５%(500mV/div)　以降 1V/div

Fine の時：アフターパルスやドループでベースラインが上がるのも見るため、50mV/div 程度。

後でレーザによる光信号で、調整する。

ただし、現状の Linearity ではアフターの部分はいらないので、fine は不要。

1%前段フィルタをつけているときは fine なし

1%前段フィルタなしで、After pulse も見る場合は、通常と fine を行う

トリガは CH4 の立ち下がり

⑧レーザのスイッチを ON にする。繰り返しは 10Hz。強度のダイヤルを 14（元のマニュアル値）

⑨ヘルムフォルツコイルの電源を入れる。　X：18V,　Y:12V

２)データ取得

この測定ではフィルタを手動で変えて光の強度を変えて、データを取得することを繰り返す

①Jupiter Notebook で次を開く。(PC 上に開いている場合あり)

/home/icecube/measurements/daq/notebooks/linearity_and_afterpulse_meas.ipynb

自分用に名前を変更して保存。(元のは標準のため)

②上から順番に実行。2 番目ので、オシロからレスポンスがある事を確認

５番目の # test のコードで、シングルの波形がとれている事を確認する。波形が大きすぎ、小さすぎる時は v/div を調整する。

Communication test with a LV control で電源がリモートで動作することを確認

③Linearity & Afterpulse measurements の所の PATH（データ保存用）を変更

/home/icecube/measurements/daq/data/linearity/今回のデータ用デレクトリ/PMT と日付 HVB からの名前/

例：degg_20201106_kp9995-E-B

③LABEL＝’f0.1”

LABEL=’f0.1_fine’

以下続いている。測定するところだけコメントをはずす。

0.1 の時は、フィルタを手動変更する。**変更の時にはレーザを OFF すること**

④新しいものは慣らしの 1 時間がいるので、Burn-in が 60＊60 がはいる

④ RUN させる。データ取得は、10 分/filter 程度

以下次のパターンで繰り返す。

１％前段フィルタ付きの時

LABEL＝’f0.1%‘　　：フィルタ　0.1%　CH3　20mV/div

LABEL＝’f1.0%‘　　：フィルタ　1.0%　CH3　100mV/div

LABEL＝’f5.0%‘　　：フィルタ　5.0%　CH3　1 V/div

LABEL＝’f10.0%‘　　：フィルタ　10.0%　CH3　1 V/div

LABEL＝’f50.0%‘　　：フィルタ　50.0%　CH3　1 V/div

LABEL＝’f100.0%‘　　：フィルタ　100%　CH3　1 V/div

1%前段フィルタ無で、Afterpulse も測定するときは、Fine も行う

ファイル名に r をつける

LABEL＝’f0.1%‘　　：フィルタ　0.1%　CH3　1 V/div

LABEL＝’f0.1%_fine‘　：フィルタ　0.1%　CH3　50mV/div

LABEL＝’f1.0%‘　　：フィルタ　1.0%　CH3　1 V/div

LABEL＝’f1.0%_fine‘　：フィルタ　1.0%　CH3　50mV/div

LABEL＝’f5.0%‘　　：フィルタ　5.0%　CH3　1 V/div

LABEL＝’f5.0%_fine‘　：フィルタ　5.0%　CH3　50mV/div

LABEL＝’f10.0%‘　　：フィルタ　10.0%　CH3　1 V/div

LABEL＝’f10.0%_fine‘　：フィルタ　10.0%　CH3　50mV/div

LABEL＝’f50.0%‘　　：フィルタ　50.0%　CH3　1 V/div

LABEL＝’f50.0%_fine‘　：フィルタ　50.0%　CH3　50mV/div

LABEL＝’f100.0%‘　　：フィルタ　100%　CH3　1 V/div

LABEL＝’f100.0%_fine‘　：フィルタ　100%　CH3　50mV/div

⑦ 終了後は、レーザを OFF、HV とコイルの電源を OFF

注）1%との組合せでは、fine は不要

さらに大強度を測定するには固定の１％フィルターをはずす。

0.1%は前回の 1%+10%とほぼ等価

ファイル名が重ならないように Sq0886r　にようにする

After パルスが出るので調べるなら Fine も測定する

３）Analysis

① terminal を開き、dir を次にする

/home/icecube/measurements/analysis/workspace/linearity/

次を走らせ、各コンフィグでのデータのヒストグラムを作成する。Dir 名は前項の③

python run.py Dirname f0.1

python run.py Dirname f1.0

python run.py Dirname f5.0

python run.py Dirname f10.0

python run.py Dirname f50.0

python run.py Dirname f100.0

例　python run.py degg_sq0921_20201106_kp9995-E-B f5.0

1 回、 1 分程度

なお、このプログラムでは Fine のファイルは処理しない

② linearity (peak current, pe/ns) and (main_pulse_charge, pe)をプロットする

python3 plot_linearity_pre_dvt.py dirname gain_dirname Vctrl

Vctrl は 1e7 Gain で図っていれば諸略可能。ここで dirname は前項の③でよい、

次の gain_dirname は gain 測定した時のデータがあるもの。場所は、Workspace/jsonfiles　内にあるファイル名だが、.json は不要

例：python3 plot_linearity_pre_dvt.py degg_sq0921_20201106_kp9995-E-B

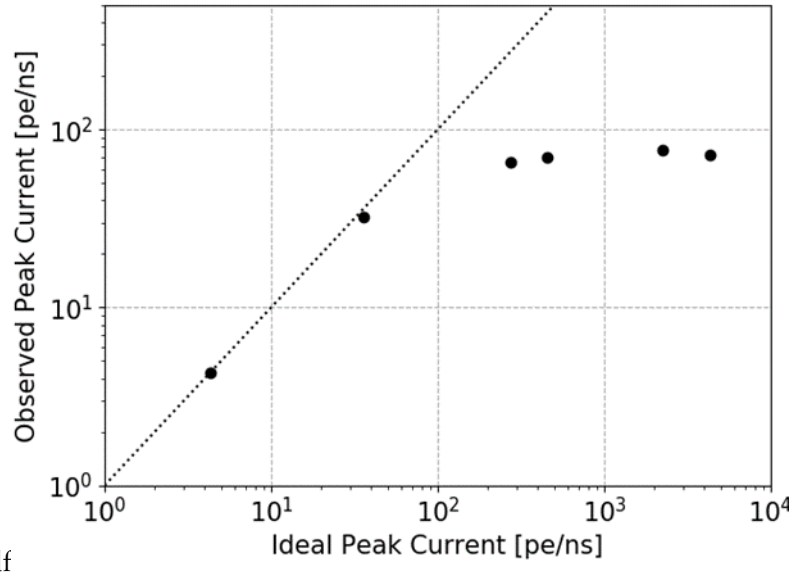
以下のグラフは Lasse のマニュアルの例

画面にグラフが表示される。消さないと次がでない。

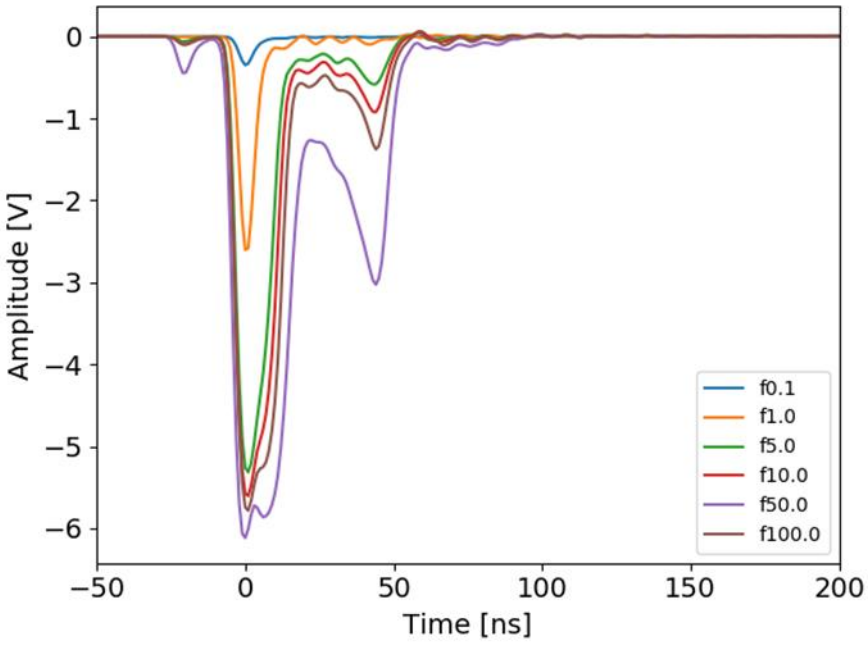
Current linearity plots will be saved as　linearity 配下で

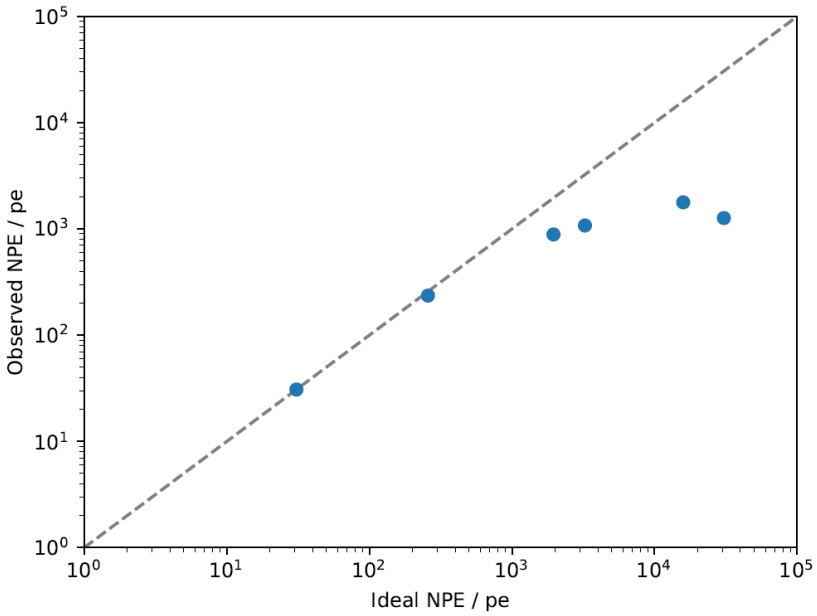
figs/lin_{pmtname}_peak_current.pdf

figs/lin_{pmtname}_waveforms.pdf



figs/lin_{pmtname}_charge.pdf





ただし、現状スプリッターで分離後レーザ光を光ファイバーで輸送して PMT に当てているのでロスが大きく固定 1%付きのデータだけでは、充分な強度がでないので、１%付きと無のをつなぎあわせている。1%無で 0.1%は飽和領域に入らないので、これを基準点と ideal の量をもとめて、２つのデータを元に、一つの図を書かせる。

4)測定コード　linearity_and_afterpulse_meas.ipynb
一つのフィルタ設定で、1000 波形を取る。
データは、/daq/data/linearity/PMT 名/wf_フィルタ名_c00000.npy の形式

一波形で 4000 点のデータ 626ｋB と大きい（サンプリング時間が長いので）
626kB＊1000 個＊12 パターン＝7.5G
解析後、波形の Raw データは削除すること。

５）解析コード
‘run.py の中身
①　charge_calc_auto_gate.py：波形データよりチャージを計算
wafana.get_ave_peak_time で波形のピークになる時間の平均をもとめ、自動で、
積分する開始 gate の時間を決めている。それをメイン、Late、after で決めている。
Main_gate_start=ピーク時間－12.5ns　　-15ns～35ns
‘ped_main_gate_start=main の開始より 1 μ s 前
‘pre_gate_start=ピーク時間-30ns　　-35ns～-15ns
‘ped_pre_gate_start=上記より 1μ s 前
‘late_gate_start=ピーク時間＋35ns　　35ns～300ns
‘afp_gate_start=ピーク時間＋300ns　　0.3μs～10.3μS
wafana.get_charges で波形ごとの main,　pre, late, after のチャージを計算
ファイル
/charges/charges_PMT 名_フィルタラベル.dat
　この中に、c_main, c_pre, c_late, c_afp が入る
/waveforms/wf_PMT 名_フィルタラベル.dat
　平均した波形データ、ゲートの開始と幅の情報がある
② make_dist.py
‘①で作ったファイルを開き、main,　pre, late, after のチャージのヒストグラムを作る
例‘h_main: X_min :-200,　X_Max:12000　X_bin : 24400
ファイルは rootfiles/dists_PMT 名_フィルタラベル_.root

Plot_linearity_pre_dvt.py
1） Json ファイルから、gain の数字を決める。デフォルトは 10^7
2） 0.1%フィルタに相当する透過度を table から求める
‘analysis/tables/twheel_filter_nd/filter_transp_2019Feb.tab
3） フィルタごとのゲート幅の情報を得る
4） Plot_and_fit_current
・平均した波形をフィルタ値ごとに表示
・ピーク電流は、観測した電流にフィルタ透過率の比をかけている。
・Observed Current[poto electorn/ns]=Vpeak[V]/50[Ohm]/(Gain *1.6E-19/10E-9)

透過率のテーブル
wheel ND filter transparency table（フィルタ％と透過率　0.1%を 1 として）

0.1　1
1.0　8.36434
5.0　63.4507
10.0　105.418
50.0　515.069
100.0　993.169

チャージは、NPE（Number of Photo Electron）を観測値と理想値で記載

コードは、アフターパルスやレイトパルスも図れるようにしているが、計算出力やグラフは出していない。
ただし、現状スプリッターで分離後レーザ光を光ファイバーで輸送して PMT に当てているのでロスが大きく固定 1%付きのデータだけでは、充分な強度がでないので、１%付きと無のをつなぎあわせている。1%無で 0.1%は飽和領域に入らないので、これを基準点と ideal の量をもとめて、２つのデータを元に、一つの図を書かせる。
これは別のコードを作り行っている

2.1　Linearity 解析プログラム

1)概要

Latafa　ＰＣの解析プログラムでは次の制限があるので、自分の PC に波形データを取り込んで解析する

・1%フィルタつきと無の 2 ケースの範囲で測定するが、それを結合して一つのグラフにできない

・アフタパルスの解析が出来ていない。

コード：Linearity_analysis_standard.ipynb

(1) 1%付きの 6 個のフィルタの波形データを読み、積分する。積分区間は 80ns－125ns
平均ＰＥとピーク電流を求める。平均波形の表示
Ideal は 1%を基準にする。0.1%は弱くバラツキが大きいため。透過率の比で ideal を求める

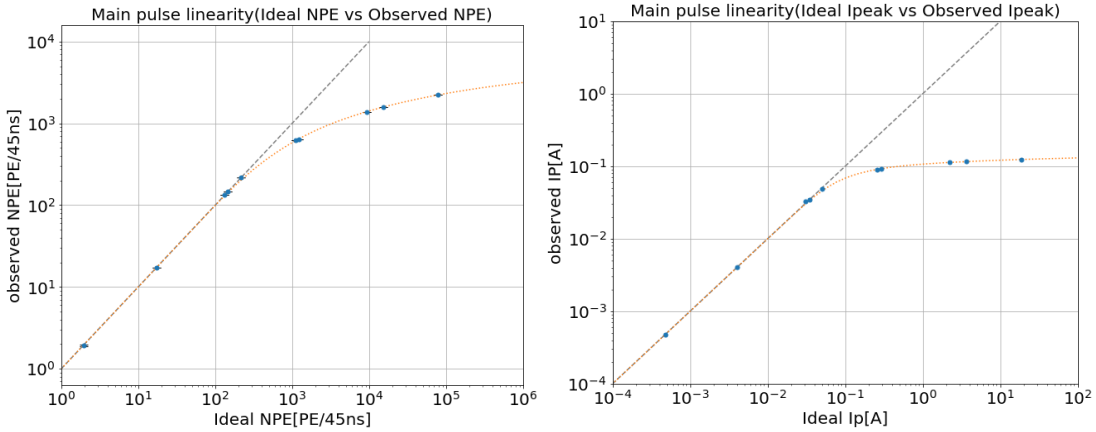
(2) 1%無の 6 個のフィルタの波形データを読み、メイン平均ＰＥとピーク電流を求める。
平均波形の表示。Ideal は 0.1%を基準にする。強度としては 1%固定＋10%に相当する。

(3) メインの 2 つのファイルから、idealNPE－obsNPE のグラフを書く
近似曲線を引く（通常のリニアリティの出力）

(4) ピーク電流の ideal－obs のグラフを書く

フィッテイング関数は次

(1./x + (1./p0) * np.log(1. + (x/p1)**3) /np.log(1. + (x/p2)**0.5)) ** -1



３．　2D uniformity scan(Relative)マニュアル

相対強度分布を測定する。各スキャンポイントでの波形を 200 個とって、チャージを計算し、2 次元強度マップを作る

１）セットアップ
・本測定時に PMT を据え付ける時には、高さ(11mm と 55mm)と水平に注意する事。これがずれると特にエッジ付近が均一にならない。また HV ソケットの切欠きのあるの側を常に同じ方向（grappa 室側）にすること。第一アノードの方向を同じにするためである。
・フィルタは、１%+0.1%（MPE レベル）（～600mV の波高さになるように調整する）
・レーザ：繰り返し　100Hz　強度　8.5　（600mV になるくらい）
・
２）測定
次のデレクトリにあるコードを jupyter notebook で走らせる
/home/icecube/measurements/daq/notebooks/2d_cathode_scan.ipynb
・label にある PMT 名を変更する。例「degg_sq0510_20201126_kp9995-E-B」
・Path が正しいか確認する
・データ点は　Zenith＝64、　azimuth＝72　　Singnal-wave　200　ref-wave　10
・測定点が 64X72　＝4608 点あり、1 点あたり 10 秒かかるので、測定に 13 時間、初期安定化時間 3 時間で、あわせて 16 時間ぐらいかかる
・Ref の Zen は一か所のみで、ファイル名を 999 にしている。
・データは、
Measurements/daq/data/cathod_sca/{label}の下に
Signal_line/azi(番号 0～71)_zen（番号 0～63）_c（番号 0～199）_zenith_equidist.npy
に一回ごとの波形がはいる。同一地点で 200 個のファイルができる。
Reerence データは
Ref_line/ azi(番号 0～71)_zen999_c（番号 0～9）_zenith_equidist.npy
メッシュが 4600 点と多いが、1 波形の時間が短く、MPE レベルで 1 点 200 個なので、全体のデータサイズは 56MB とそれほど大きくない。

３）モータ系のトラブル対応
終了すると、ゼロ点に戻るはず。
戻らない時は、jupyter の下にある個別コマンドで中心にもどす
ゼロ点に戻せない時は手で、モータの近くの棒を回して戻す。
その際には、AC プラグを抜いて、モータの電源を切る事。切らないとトルクがかかったままで、手で動かせない。Zenith は箱の上からのぞいて回せる。Azimuth は、横の箱を上げてのけてから、手で回す。このコードでは逆回転ができないので手動で戻す。これはかなり大変。
プラスに進めて、一周まわしてゼロに合わせないこと。次にスキャンさせると 2 周回ることになり、

ケーブルが軸にとぐろ巻いて巻き、ケーブルが切れるリスクが大。
測定が終了したら、del motorcontl をすること。これをしないで次に走らせると Kernel が死ぬ
死んだ時は、jupyter を一旦落として、再度立ち上げる必要がある

モータコントロールが異常になった時。
・ＰＣの電源とモータドライバーの電源を落とす
・ＰＣを再立ち上げする。Boot するときにキーを押してとめて、Linux のバージョンが２つでるが、下の方にする。これをしないと動かない Ver で立ち上がる
Linux ratafia 2.6.32-696.e16.x86_64
その後で、dir を
/usr/src/interface/gpg7400/x86_64/linux/drivers
にする
su
PW
source insmtn.sh
exit
その後で、jupyter を立ち上げる

４）Analysis

dir を次にする/home/icecube/measurements/analysis/workspace/cathordscan/
ここにある README に従って行う
$ ./run.sh {pmtname}
$ cd table_generator
$ ./bin/make_uniformity_table {pmtname}
$ cd scripts
$ python fishfingers_custard.py {pmtname}

./table_generator/scripts/figs/cemap_{pmtname}.pdf will be created

① run.sh の中身
#!/bin/sh
label=$1　　：PMTname がくる
echo ${label}
python3 charge_calc.py ${label}
unlink charges/use_this_pmt
ln -s /home/icecube/measurements/analysis/workspace/cathode_scan/charges/${label}

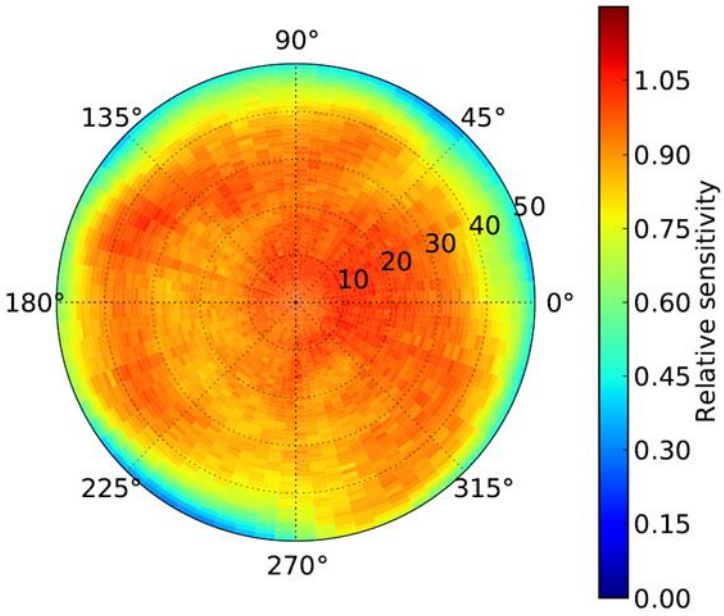
/home/icecube/measurements/analysis/workspace/cathode_scan/charges/use_this_pmt
root -l -q -b plot_charge_dist_w_refPMT.C
mv ./fit_res/fit_results_use_this_pmt.dat ./fit_res/fit_results_${label}.dat
mv ./fit_res/fit_results_ref_use_this_pmt.dat ./fit_res/fit_results_ref_${label}.dat

② charge_calc.py ${label}
これが Charege を計算する。AZI と ZEN の最大 BIN 数を　72 と 64 に固定しているので、元を変える場合はこちらも変える
wfana.py を読みだしている。その中の get_Charges_cathorde_scan で計算させている
この中で測定した波形ごとに呼びだして、波形ごとにチャージを計算している。
その結果は、
'cahord_scan/charges/PMT 名/charges_azi 番号_zen 番号.dat'
にあり、テキストで、200 個のデータがある
その後で、一つの点ごとに、チャージ(pC 単位)の平均値と STD を求めて
'cahord_scan/ave_charges.dat で出力する。これは毎回書き換えられるので、PMT ごとに保存されない

③ root -l -q -b plot_charge_dist_w_refPMT.C
ここで RefencePMT のデータを使っている
ヒストグラムを作り、Gauss フィットしている
正方形の図を記載する

④ informality table が作られる

⑤ Fishfinger_custerd.py
上記テーブルから丸い図にして表示



４．　2D Photon Detection Efficiency scan マニュアル

概要

MPE レベルで、場所ごとに Photon Detection Efficiency を決定する。場所ごとの gain calibration が必要となる。絶対値が calibration された reference　PMT で測定された値と、メインの PMT で測定された photon 数を比較する

1)Setting

・40PE レベル　（400mV~600mV ぐらい）

・CH4：レーザからの外部トリガ、CH3：sgnal_line　CH2: reference line

・フィルターは、１%+5%（MPE レベル）（400~600mV の波高さになるように調整する）

・レーザ：繰り返し　100Hz　強度　10 程度

・Reference PMT。HV=1500V

・Helmholtz コイルを使う（18V、12V）

2)測定

次のデレクトリにあるコードを jupyter notebook で走らせる

/home/icecube/lasse/notebooks/2d_mpe_pde_scan/2d_mpe_scan.ipynb

元のマニュアルの場所から次に変更（ここでは Motor 動かない）

/home/icecube/measurement/daq/notebooks/2d_mpe_pde_scan/2d_mpe_scan.ipynb

ただし、analysis は Lasse の dir で良い

・Path の PMT 名と日付を変える

・Vc を 1X10^{7} ゲインの値にする

・各点 200 個の波形を記録する。メッシュは 64X72 ＝4608 点。

・メイン PMT と Ref PMT の両方のデータを取る

測定には、1 時間＋42 時間 24 分＝43 時間半　ほぼ 2 日かかる。

終了したら、ゼロ点にもどっているか確認する。戻っていない時は手で回して戻す

メッシュの削減は可能。解析コードもメッシュ固定でなく、測定の角度データで分析するので、短い時間で傾向を知りたいときは、メッシュをそれぞれ半分にすると 1/4 になり、11 時間程度で完了する

データ量　15.8K/（一波形）X2CHX200 個X64X72～30G

raw データは、外付け Disk に移す（ただし、かなり時間かかる)、または消す

3)測定コード

2d uniformity scan　とほとんど同じ

data　ディレクトリの変更と Vc の変更、メッシュ数を確認する

4)Analysis

dir を次にする

/home/icecube/lasse/analysis/2d_mpe_absolute_pde/

例
[icecube@ratafia ~]$ cd lasse/analysis/2d_mpe_absolute_pde/
[icecube@ratafia 2d_mpe_absolute_pde]$ python3 build_charge_dict.py --data /home/icecube/lasse/notebooks/2d_mpe_pde_scan/data/sq0866_degg_full_run_20210319 --plot_dir data/sq0866/data0319 --plot_n_waveforms 10 --signal_line_signal_gate 375 625 --signal_line_baseline_gate 50 325 --reference_line_signal_gate 250 500 --reference_line_baseline_gate 50 200
画面上は次のとおり
Plotting 10 random waveforms for signal and reference line
Plotting signal line waveforms: 100%|##| 10/10 [00:02<00:00, 3.71plot/s]
Plotting reference line waveforms: 100%|##| 10/10 [00:01<00:00, 5.98plot/s]
Looping over scan points: 100%|##| 4608/4608 [1:04:05<00:00, 1.20scan point/s]
Dumping the charge dictionary to /home/icecube/lasse/notebooks/2d_mpe_pde_scan/data/sq0866_degg_full_run_20210319/f/s]
これは 40 分~ 1 時間かかる。
Plot_dir を PMT 番号と日付を指定してあげないと、前回の結果に上書きしてしまうので、注意すること。
パルスの位置とゲートは次の通り。-100ns～300ns　0.4ns/1K_sample 前提

	ピーク	積分範囲	ゲート
Signal	+90ns	50ns~150ns	375~625
refe	+40ns	0 ns～100ns	250~500

random_waveforms　に波形データがあり、積分区間（赤おび）内に波形がきていることを確認する。

次に Map を書かせる。例
[icecube@ratafia 2d_mpe_absolute_pde]$ python3 mpe_absolute_pde_scan_morii.py --datadir data/sq0866/ --plot_dir data/sq0866/data0319 --beamline_factor 0.585
以下画面
Loading charge dictionary...
Done
Preparing charge dictionary
Testing gain map function at (0, 0):　10490000.0
Plotting the used gain map...

Building dictionary...
Making absolute gain plots
Making relative gain plots
Done
Plotting 1D PDE vs Zenith...
Done
Plotting 1D PDE vs Zenith in band mode...
Done
Plotting 2D absolute PDE maps...
Done
Plotting 2D relative PDE maps...
Done

beam line factor は 2021 年 2 月から 0.585 を用いる
QE はデフォルトで OK,　ref の gian も defalt にした
dir でエラーがでるときは、最初と最後に/を入れてみる、入れない場合を行う。これで引っかかる時がある

ディスクデータのクリーンナップ
　ファイルサイズが 30G と大きいので、解析が終われば、消す。以下に示している pickle ファイルに波形ごとに積分したチャージのデータが残るので、それで十分である。もし波形データも残したいのなら、波形データは USB のＨＤＤに移す。ただし、ファイルの個数が多いので、時間がかかりすぎるので、各点ごとの 200 ファイルを一つにまとめ、4608Ｘ 2 (Sig と ref)に減らしてコピーするスクリプトを動かす
　Jupyter にある“2d_mpe_data_convert”を使う
保存先は、
media/HDJA=UT/daq/data/2d_mpe_absolute_pde/sq0XXX　（PMT 名）
　実施時間は、４ ６分。終わったら、本体側ＨＤＤ側のファイルをディレクトリごと削除する
lasse/notebooks/2d_mpe_scan/data/該当するＰＭＴ番号

解析コード
build_charge_dict.py
・波形を読み取る
・要求があれば、サンプルで波形のプロット（waveform_ana.plot_waveform）
をする。それはランダムに選ぶ。収納先は指定先の random_waveforms
　積分範囲のゲートの位置の確認になる
・各点で、Singnal と Reference の波形の数だけ積分を実施

waveform_ana.integrate_waveform で実施
次の形で場所とチャージの辞書を作る。各点に 200 個のチャージがある
charge_dict[(azimuth, zenith)]['signal'] = np.array(signal_charges)
charge_dict[(azimuth, zenith)]['reference'] = np.array(reference_charges)
charge.dict の pickle ファイルが保存される

解析 2
mpe_absolute_pde_scan.py
次の option がある
--datadir "The top level directory the charge dictionary is stored in" 必要
--plot_dir “The directory the plots shall be stored in' 必要
--gain_signal_pmt The gain of the PMT in the signal line 不要（デフォルト）
(default 1.049E7) Use this, if a constant gain over the whole photocathode should be assumed"
--gain_scan_dict_path "Path to a dictionary of the 2D gain scan. Use this, if a position dependent gain of the PMT should be used"" 2d_gain_map をしていれば、そこで、gain_dict.pckl ファイルを呼ぶ。dir でなくファイル名 していなければ、不要
--gain_interpolation action='store_true',Set this flag if the gain map should be interpolated. The scipy RectSphereBivariateSpline will be used""") 無でデフォルトでやる
--zoomed_plot_rmax"The maximum zenith angle in degrees in the zoomed plots (default 45)"不要
--gain_reference_pmt "The gain of the PMT in the reference line (default 5.73E6)) 最新に変更
'--beamline_factor“The average number of photons at the end of the signal beamline for each photon at the end of the reference baseline (default 0.758)" 変更必要
--reference_pmt_pde"The absolute photon detection efficiency of the reference PMT (default 0.263)") 浜ホトカタログ値
--rotate_pde_plot "Rotate the PDE plots X degrees counterclockwise （default=0.0） 不要
--rescale_pde_to_center', "If set, rescale all azimuth slices, such that the center value matches the mean" action='store_true 不要

Main
charge_dict.pckl を開く
gain map を作る （build_gain_map_function）：gain map は利用しないケースがほとんど。
各点での gain の値をだす
要求あれば間を Interpolate(補間)する。しないとそのまま
各点のチャージの合計値を出す
以下の式で PDE を求める

$$PDE_{DEgg}(\Phi,\Theta) = PDE_{ref} \cdot \frac{1}{f_{beamline}} \cdot \frac{\sum charges_{DEgg}(\Phi,\Theta)\,/\,gain_{DEgg}(\Phi,\Theta)}{\sum charges_{ref}\,/\,gain_{ref}}$$

中心で recale するか？　するならその分だけ変更：通常はしない

PDE 値を’pde_dict.pckl’に保存する

それを元に次の図を作成する。実際使うのは黄色の図

plot_2d_gain_map

gain_map_absolute

'gain_map_absolute_zoomed',

'gain_map_relative',

'gain_map_relative_zoomed',

plot_pde_vs_zenith

'pde_vs_zenith

'pde_vs_zenith_zoomed',

plot_pde_vs_zenith

'pde_vs_zenith_band',

'pde_vs_zenith_band_zoomed'

plot_2d_pde_map

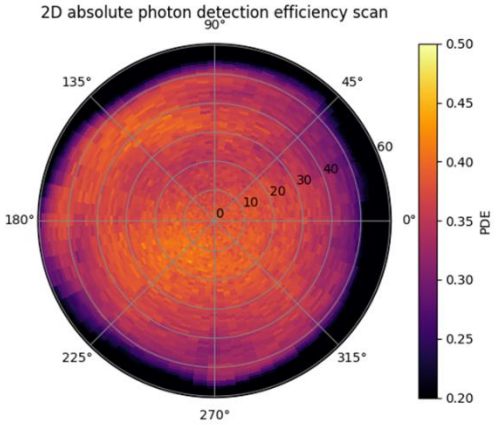
'2d_relative_pde_map',

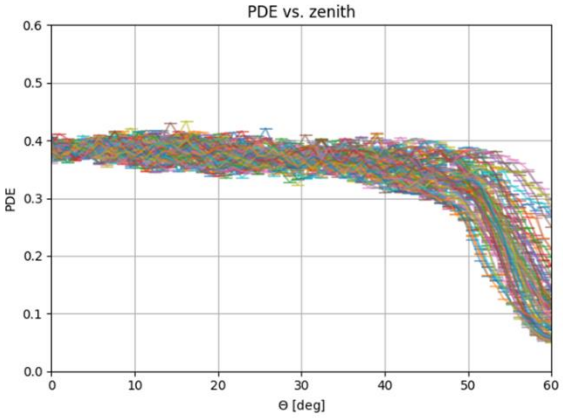
'2d_relative_pde_map_zoomed'

'2d_absolute_pde_map',

‘2d_absolute_pde_map_zoom’

カラーマップの範囲は固定なので、低い時は真っ黒になる。デフォルトの PDE は 0.2〜0.5





５．DEgg 2D Gain Map

目的

Photocathode　の場所によるゲインの変動を見る。

各スキャンポイントで 10000 波形、SPE レベル

セッティング

フィルタは 1%+0.1%　レーザ強度は　~12 ぐらい　（SPE レベル）

Vc は 1 E7 ゲイン

ヘルムフォルツコイル必要

測定コード

/home/icecube/lasse/notebooks/2d_gain_scan/gain_calibration.ipynb

元のマニュアルの場所から変更（ここでは Motor 動かない）

/home/icecube/measurement/daq/notebooks/2d_gain_calibration.ipynb

測定点：Zenith　10、　Azimuth　12　合計　120 点　　10000Waveform/点

一時間のアイドリングタイム

各点に行くと、1 分の待ち

各点ごとに、10000 個の波形ファイルを作成

CH3 で Signal-line のデータを取っている

CH2 で reference-PMT を読むルーチンはあるが、使っていない。

16 分/ポイント X12X10　＝32 時間！

データ容量　18G　（16kB/波形 X10000X120 ）

単純にモータで動かしながら各点の SPE レベルの波形を集めるだけのプログラム

Analysis

/home/icecube/lasse/analysis/2d_gain_calibration/make_gain_maps_2d.py

子プログラム　waveform_analysis

Gain_vs_lv_scan

実行方法と必要な引数

python3 make_gain_maps --datadir

/home/icecube/lasse/notebooks/2d_gain_scan/data/sq0975_degg_2d_scan_4.0V_fine_20210108/ --plot_dir {plot の dir} –plot_n_waveforms 10000

注　‘で囲まない、はじめと終わりに/を入れる

plot の dir は存在している dir にすること

ゲート幅がデフォルト（500－650）とだとずれる時がある。（400―550）がよい。

ベースも 50－450 でなく、50－400

dir を間違えていると main で‘list index out of range’のエラーがでる

・各点での SPE のスペクトル（ヒストグラム）を出し、SPE のピークを求め、その点での gain を計算する。

1） 各点ごとに波形データを読みチャージを計算する。各点ごとに波形数（10000 個）繰り返す
　　データを plot_dir+charge_dir.pckl　にバイナリで書き込む

2） Combined SPE distribution　の図を書く
　２D の絶対値のゲインマップ
　２D の相対ゲインのマップを作製
　ゲインの角度分布グラフ

結果は--plot_dir で指定した場所に

absolute_2d_gain_map.png　：絶対ゲインマップの図

relative_2d_gain_map.png：相対的なゲインマップの図

combined_spe_distributions.png：全部のＳＰＥカーブの図

gain_vs_zenith.png : Zenith の角度とゲインのグラフ

SPE__Spectra　このディレクトリィ内には、各点での SPE カーブとフィッテングの図がある

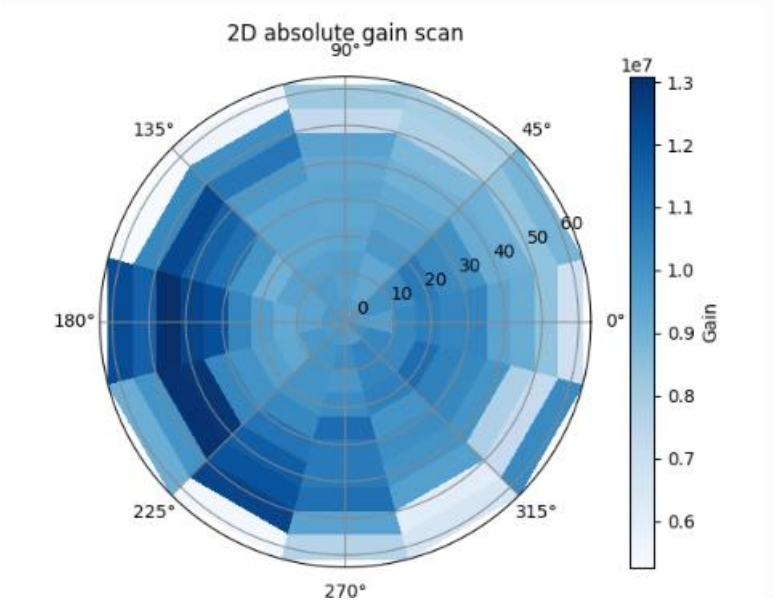
charged_dict.pckl

gain_dict.pckl

Reference PMT や前の sig/ref の比は使っていない

Gain map 出力例

SPE カーブのフィッテングに失敗した点は、真っ白で空白に表示される。



６．Latafa　PC を再立ち上げした時の立上げ手順

電源を入れると同時に F1 キーを押す。その後で、メッセージに従って、どれかのキーを押すと
　Linux のバージョンが２つでるが、下の方にする。これをしないとモータが動かない
　　Linux ratafia 2.6.32-696.e16.x86_64
　IceCube で立ち上げる(PW： 5 x x xxxxx)

その後で、dir を 2 つ上にあげて
/usr/src/interface/gpg7400/x86_64/linux/drivers
にする
su
PW（5xxxxxx）
source insmtn.sh
exit
その後で、jupyter を立ち上げる