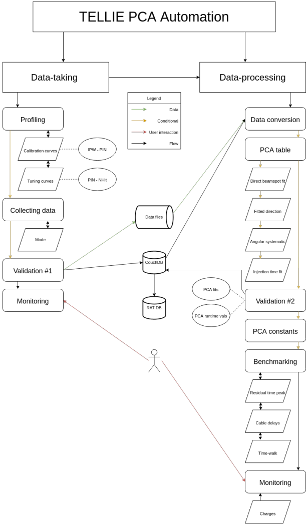
TELLIE PCA Automation: Data processing

Overview
November 21, 2021

Michal Rigan
mrigan@snolab.ca

University of Sussex

Overview



TELLIE PCA Automation

Data-taking

Data-processing

Legend

Data

Conditional

User interaction

Flow

Profiling

Calibration curves

IPW - PIN

Tuning curves

PIN - NHit

Collecting data

Mode

Validation #1

Monitoring

Data files

CouchDB

RAT DB

Data conversion

PCA table

Direct beamspot fit

Fitted direction

Angular systematic

Injection time fit

PCA fits

PCA runtime vals

Validation #2

PCA constants

Benchmarking

Residual time peak

Cable delays

Time-walk

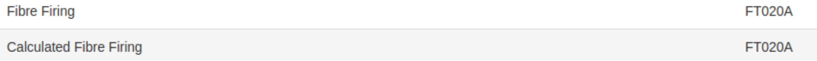
Monitoring

Charges

Validation #1

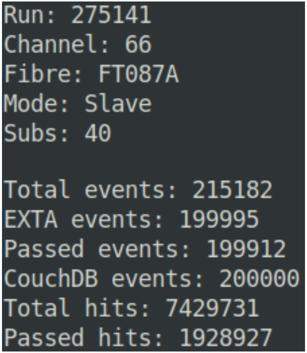
Goal: ensure that the data (just taken) is valid and can be used for PCA.

Validation #1: correct fibre



Fibre Firing	FT020A
Calculated Fibre Firing	FT020A

Validation #1: Stats - events

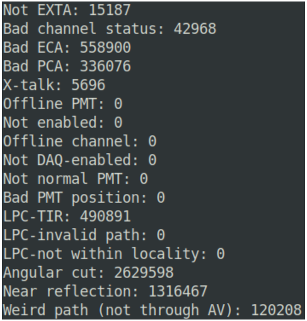


```
Run: 275141
Channel: 66
Fibre: FT087A
Mode: Slave
Subs: 40

Total events: 215182
EXTA events: 199995
Passed events: 199912
CouchDB events: 200000
Total hits: 7429731
Passed hits: 1928927
```

(missing EXTA recovery ?)

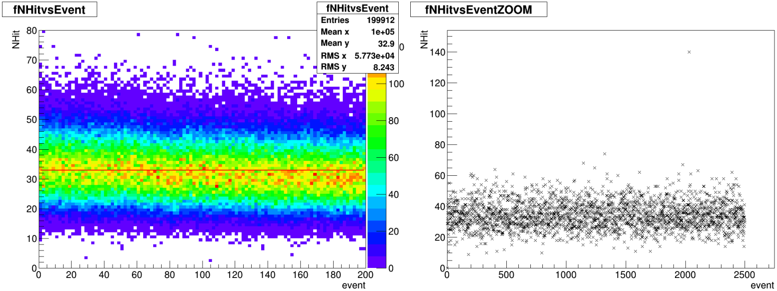
Validation #1: Stats - cuts



```
Not EXTA: 15187
Bad channel status: 42968
Bad ECA: 558900
Bad PCA: 336076
X-talk: 5696
Offline PMT: 0
Not enabled: 0
Offline channel: 0
Not DAQ-enabled: 0
Not normal PMT: 0
Bad PMT position: 0
LPC-TIR: 490891
LPC-invalid path: 0
LPC-not within locality: 0
Angular cut: 2629598
Near reflection: 1316467
Weird path (not through AV): 120208
```

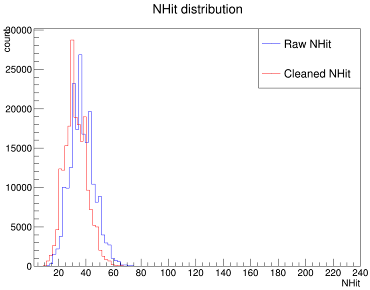
threshold on valid events / hits

Validation #1: NHit

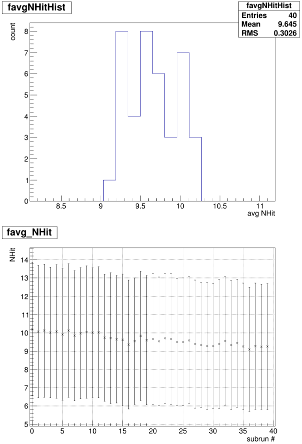


fibre intensity, stability

Validation #1: NHit

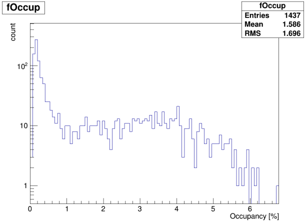


Raw

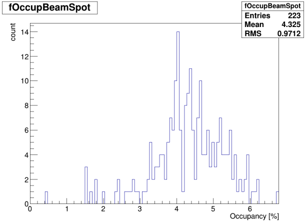


Valid hits

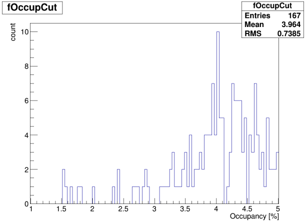
Validation #1: Occupancy



All

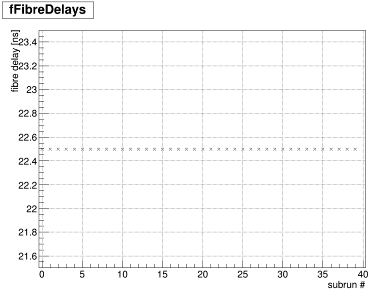


Beamspot

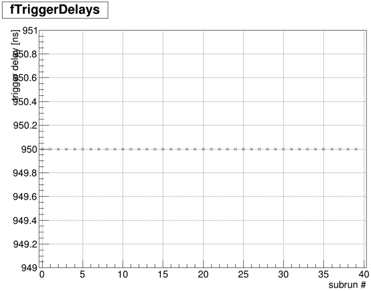


Cuts

Validation #1: Delays

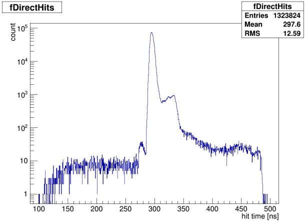


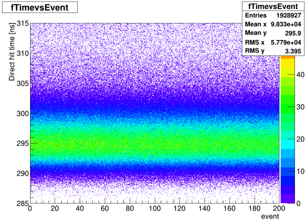
Fibre delay

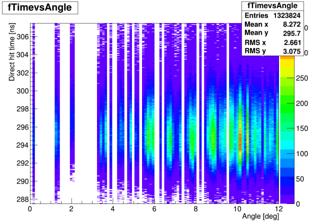


Trigger delay

Validation #1: Time distribution

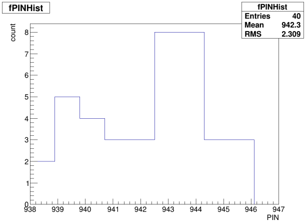


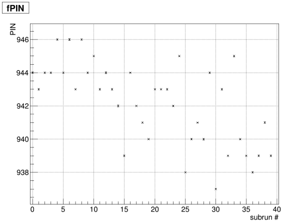


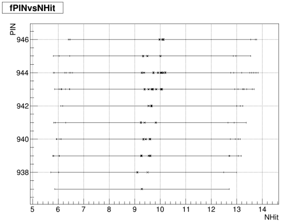


Direct hit time dist | Direct hit time f. evs | Direct hit time f. angle

Validation #1: PIN

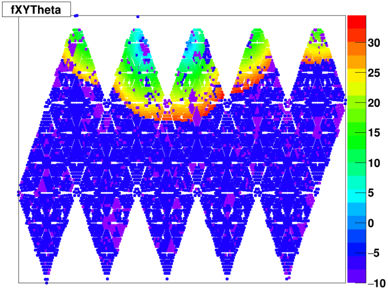




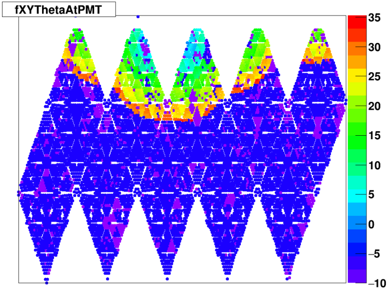


PIN to NHit (tuning)

Validation #1: Angles



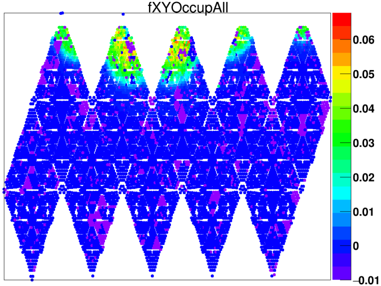
angle: fibre - PMT



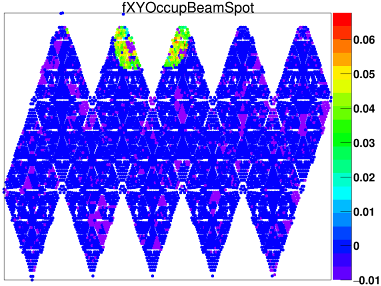
angle: light - PMT bucket

Some PMTs don't see light that should - DB with problematic PMTs?

Validation #1: More flat maps



Occupancy (all)



Occupancy (beamspot)

Validation #1

Work: need checks/thresholds based on the plots above to flag run good/bad.
If bad - retry.
If bad repeatedly - skip.

PCA table

Goal: fit for parameters required for the extraction of PCA constants.

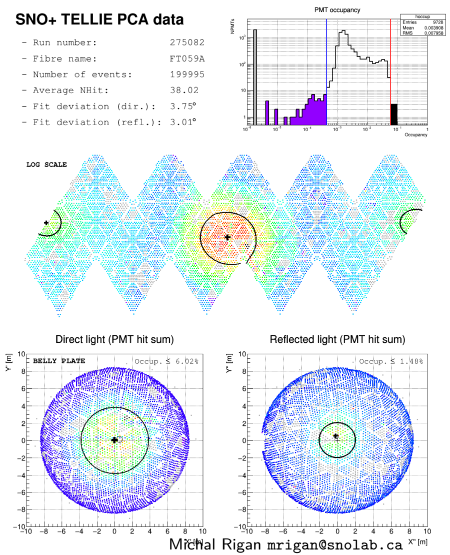
PCA table: Beam spot + direction

Goal: fit the direct light beamspot, extract the direction from the fibre position (assume correct from DB), to the center of the beamspot.

The process is based on splitting PMTs into faces (triangles) and fitting based on hits.

May be improved using dynamic beamspot / NHit instead.

PCA table: Beam spot + direction



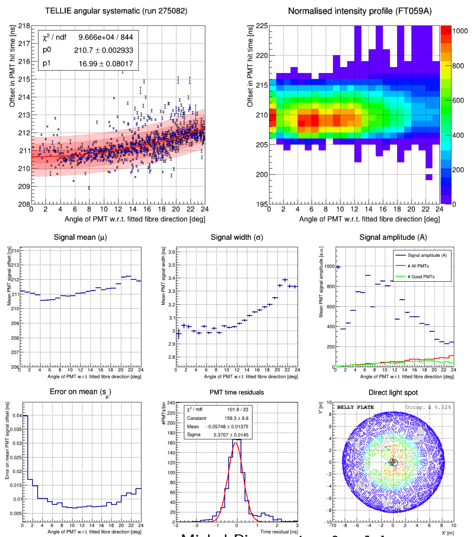
SNO+ TELLIE PCA data

- Run number:	275082
- Fibre name:	FT059A
- Number of events:	199995
- Average NHit:	38.02
- Fit deviation (dir.):	3.75°
- Fit deviation (refl.):	3.01°

PCA table: Angular systematic

Goal: evaluate the effect of modal dispersion for each fibre. Fit the distribution.
The angular range is wider here, since the effect is more pronounced at higher angles.
Requires previous PCA constants.

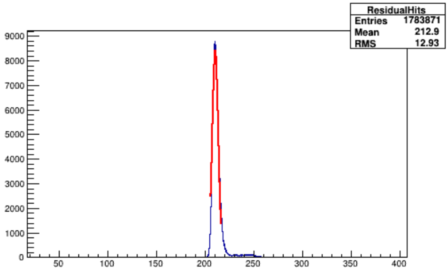
PCA table: Angular systematic

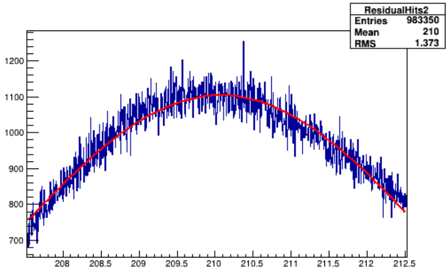


PCA table: Injection time

Goal: fit the injection time =
(hit time - bucket time - flight time - angular correction).
Needed for PCA extraction.
Requires previous PCA constants

PCA table: Angular systematic





Validation #2

Goal: check new pca table values - reasonable? Compare to previous set(s). Will include:

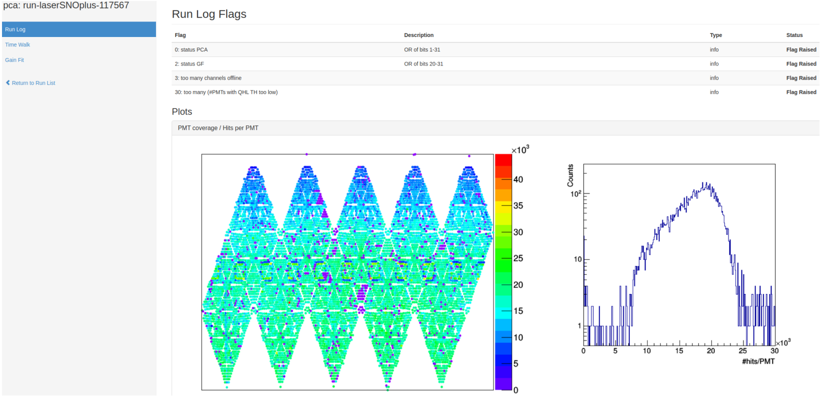
- Time of flight correction
- Bucket time correction
- Angular systematic correction
- Fibre direction deviation
- Emission time
- Run-time values: IPW, delays...

PCA constants

Goal: PCAProc that extracts the PCA values.
Idea: split LB and TELLIE processor.
(LB processor not used/tested for scintillator.)

Monitoring

Goal: revive old PCA monitoring page.



pca: run-laserSNOplus-117567

Run Log

Time Walk

Gain Fit

Return to Run List

Run Log Flags

Flag	Description	Type	Status
0: status PCA	OR of bits 1-31	info	**Flag Raised**
2: status GF	OR of bits 20-31	info	**Flag Raised**
3: too many channels offline		info	**Flag Raised**
30: too many (#PMTs with QHL TH too low)		info	**Flag Raised**

Plots

PMT coverage / Hits per PMT

Open questions

- tagging of events outside Orca
- when (from) to apply new constants
- DB with electronic issues for PMTs
- ensure recent ECA
- discrete vs continuous mode
- continuous data-taking: run-type changes, run rollovers, breakdowns...
- data conversion - now not needed
- server to run this on, with access to data (snug1 / snug2)
- unified event selection, hit cuts

List of cuts: EXTA

EXTA event (macro):

```
/rat/proc/if trigTypeSelector # None of the following
    /rat/procset trigType "N100Low"
    /rat/procset trigType "N100Med"
    /rat/procset trigType "N100High"
    /rat/procset trigType "N20"
    /rat/procset trigType "N20LB"
    /rat/procset trigType "Pedestal"
    /rat/procset trigType "EXT8PulseAsy"
/rat/proc/else
    /rat/proc/if trigTypeSelector
        #Only pure EXTA
        /rat/procset trigType "EXTASY"
```

List of cuts: EXTA

EXTA event (code):

```
trig = ev.GetTrigType();
if (!(trig & 0x8000))
```

List of cuts: PMTs

```
const RAT::DS::CalPMTs& pmts = ev.GetCalPMTs();
for(int iPMT=0;iPMT<pmts.GetNormalCount();iPMT++){
RAT::DS::PMTCal pmt = pmts.GetNormalPMT(iPMT);
int pmtID = pmt.GetID();
const RAT::DU::PMTCalStatus& pmtStatus =
RAT::DU::Utility::Get()->GetPMTCalStatus();
const RAT::DU::ChanHWStatus& chs =
RAT::DU::Utility::Get()->GetChanHWStatus();
const RAT::DU::PMTInfo& pmtinfo_loop =
RAT::DU::Utility::Get()->GetPMTInfo();
```

List of cuts: PMTs

```
unsigned int status = pmtStatus.GetHitStatus(pmt);
if(status & (1«pmtStatus.kCHSBit)){continue;}
if(status & (1«pmtStatus.kECABit)){continue;}
if(status & (1«pmtStatus.kPCABit)){continue;}
if(status & (1«pmtStatus.kXTalkBit)){continue;}
if ( !chs.IsTubeOnline(pmtID) ){continue;}
if ( !chs.IsEnabled() ){continue;}
if ( !chs.IsChannelOnline(pmtID) ){continue;}
if ( !chs.IsDAQEnabled(pmtID) ){continue;}
if ( pmtinfo_loop.GetType(pmtID) != 1 ){continue;}
```

(chs cuts are also used to mark offline PMTs)

List of cuts: PMTs

```
const RAT::DU::PMTInfo& pmtinfo =
RAT::DU::Utility::Get()->GetPMTInfo();
TVector3 pmtPos = pmtinfo.GetPosition(pmtID);
if (pmtPos.Mag()==0){continue;}
```

List of cuts: LPC

```
RAT::DU::LightPathCalculator lpc =
RAT::DU::Utility::Get()->GetLightPathCalculator();
lpc.SetELLIEEvent(true);
lpc.CalcByPosition(fibrepos, pmtPos, energy, LOCALITY);
if (lpc.GetTIR() == 1) {continue;}
if (lpc.GetPathValid() == 0) {continue;}
if (lpc.GetResvHit() == 1) {continue;}

if (lpc.GetTotalDist() <= 12000){continue;}
if (lpc.GetDistInInnerAV() <= 9000){continue;}

if ( (theta > 12) || (theta < 0) ) {continue;}
```

List of cuts: Occupancy

```
pmtOccup[iPMT] = (float)fPMTs[iPMT][7][0]/(float)allEvs;
if ( pmtOccup[iPMT] >= 0.01 && pmtOccup[iPMT] <= 0.05){...}
```