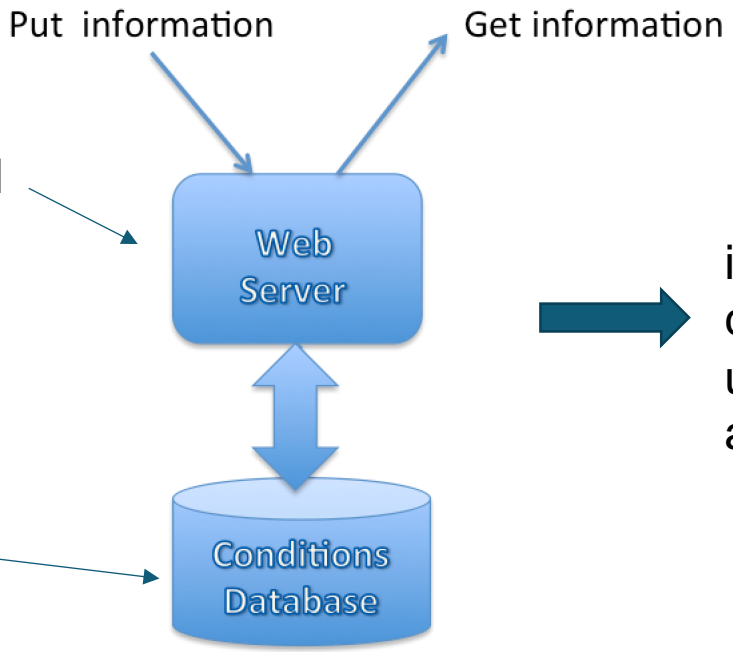

Brookhaven
National Laboratory
U.S. DEPARTMENT OF
ENERGY
Closer Look at
Calibration DBs
Lino Gerlach, Paul Laycock
08.06.2022
@BrookhavenLab

Introduction

- Still gathering example uses cases of conditions data
- Identified ‘Calibration Database’ (thanks Tingjun for providing the material) https://wiki.dunescience.org/wiki/ProtoDUNE-SP_Calibration_Database



Don’t have access to source code of API

Don’t have access to underlying DB

investigate dunesw source code & http service to understand how the data is accessed & used

Fermilab conditions database

- 2 kinds of calibration, 4 tables in DB (there are more):
  - Electron lifetime
  - dQ/dx YZ
  - dQ/dx X
  - dQ/dx normalization

} Space charge effect ('SCE')

AFAIK, the other tables are not accessed

DB content also stored here:
/dune/data/users/wwu/protodune/database/

DUNE Conditions Database

Tables in namespace: pdunesp

Go to namespace: pdunesp Go

Name	Snapshots	Plot data
pdunesp.adcgain	0	plot
pdunesp.channel_status	0	plot
pdunesp.distcorrnorm	13	plot
pdunesp.distcorrx	36	plot
pdunesp.distcorryz	61	plot
pdunesp.gain	0	plot
pdunesp.lifetime_purmon	3	plot
pdunesp.pedestals	0	plot
pdunesp.wwu_test	2	plot

Couldn't get the plotting to work ☹

dunesw accesses data via http service:

DBWeb query: https://dbdata0vm.fnal.gov:9443/dune_con_prod/app/get?table=pdunesp.lifetime_purmon&type=data&tag=v1.1&t0=1539711086&t1=1539883886&columns=center,low,high
Got 3 rows from database
run: 5387 ; subrun: 1 ; event: 3
evttime: 1539797486
fLifetime: 17518.348506 [us]

from logfiles when running reconstruction on raw data

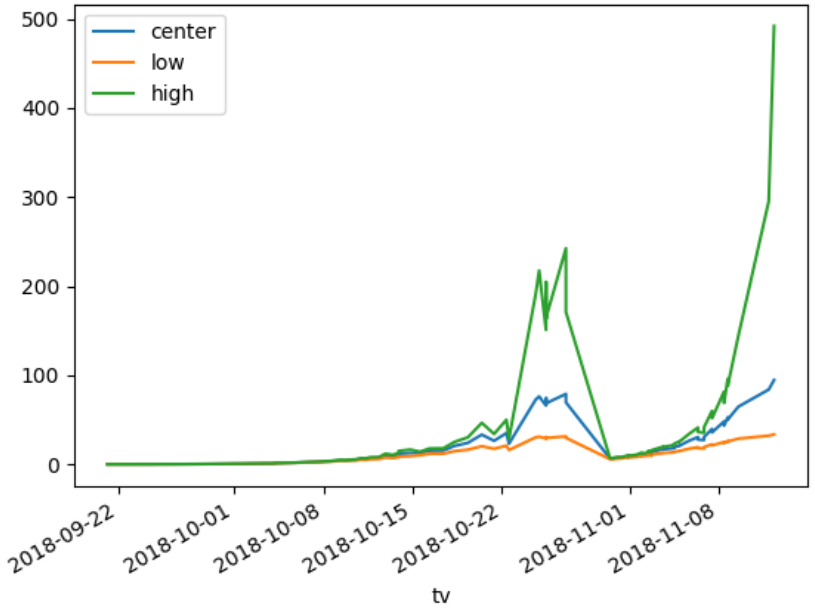
Content of the lifetime calibration DB

(More precisely: what the http service returns)

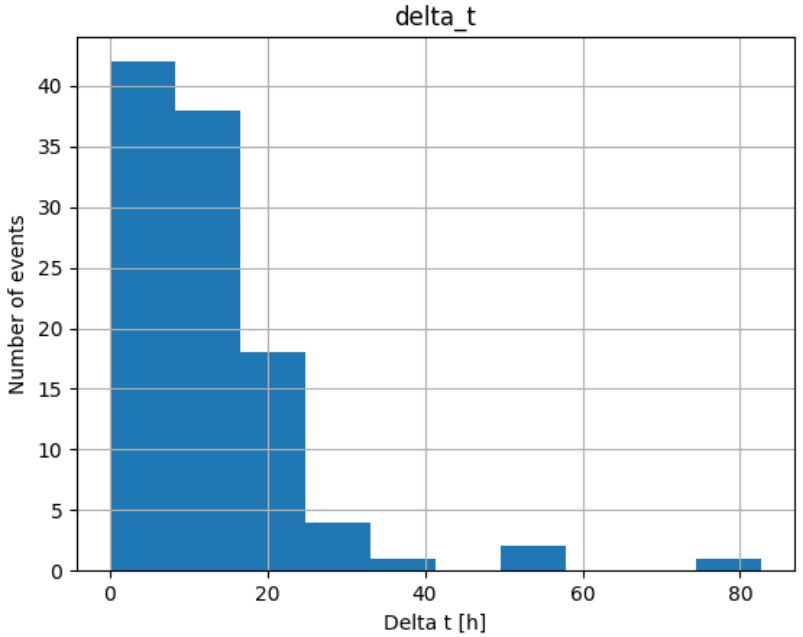
the kind of info we are looking for

- Lifetime calibration: can specify start- and end-time stamp
  - Pick start and end of Run-1, tag: ‘v1.1’
  - Return 107 rows (4 float cols: timestamp, low, center, high)

Values vs time



Duration between two entries

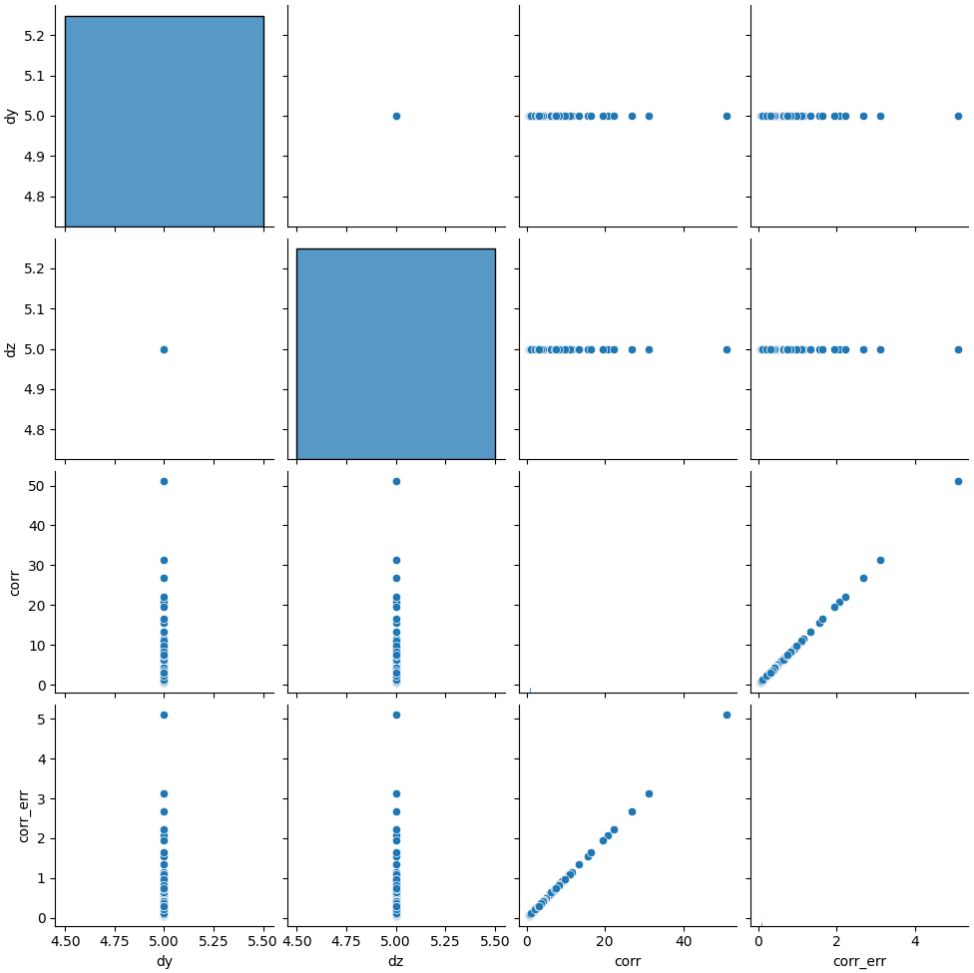


Content of dQ/dx Calibrations

- Indexed by run number & 'channel' (combination is unique)
  - Consider Run 1: 92 run numbers contain values
  - Requiring a run number that does not contain values, returns the last run number that does contain values

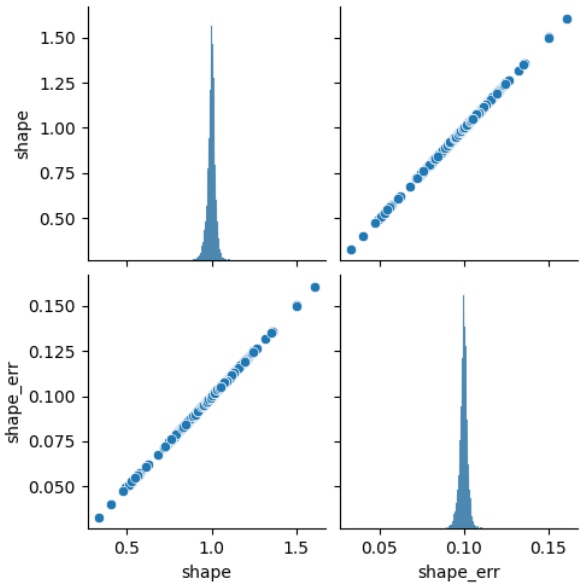
- distcorrx: 39,728 rows, 6 cols: channel, tv, x, dx, shape, shape_err
  - varying number of channels per run (431 or 432)
  - values for all 92 runs
- distcorryz: 6,197,659 rows, 8 cols: channel, tv, y, dy, z, dz, corr, corr_err
  - varying number of channels per run (99717 to 100080)
  - only has values for 62 runs
- distcorrnorm: 276 rows, 4 cols: channel, tv, norm, norm_err
  - 3 channels (0,1,2) per run
  - values for all 92 run numbers

Redundant data in calib DB - constants

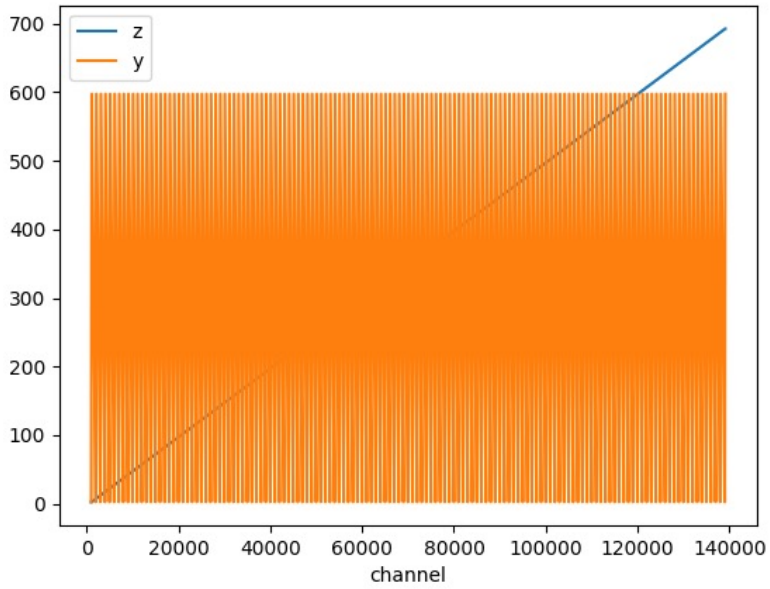


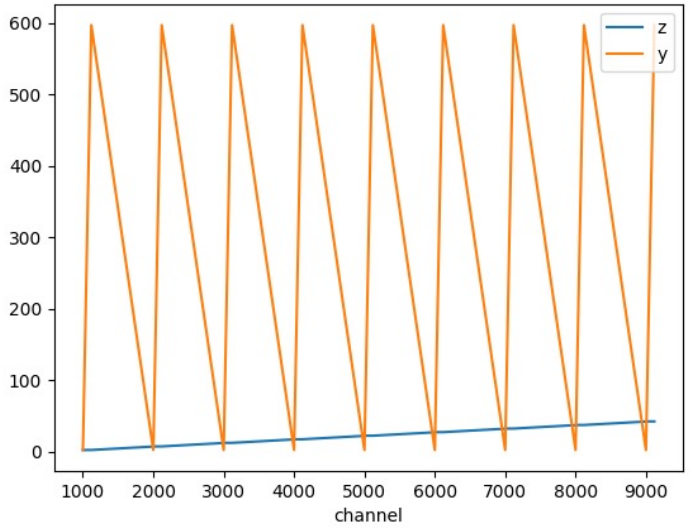
Partly redundant data:

- dx, dy, dz always 5.0
- shape_error = 0.1*shape
- corr_error = 0.1*corr



Redundant data in calib DB – Indexing





- Correction factor for each space point in cart. coordinates
  - Evenly spaced grid (dx=dy=dz=5 cm), parametrized by single index
  - Can be reduced to single index + corr (8->3 columns)

Electron Lifetime vs Space Charge Calib

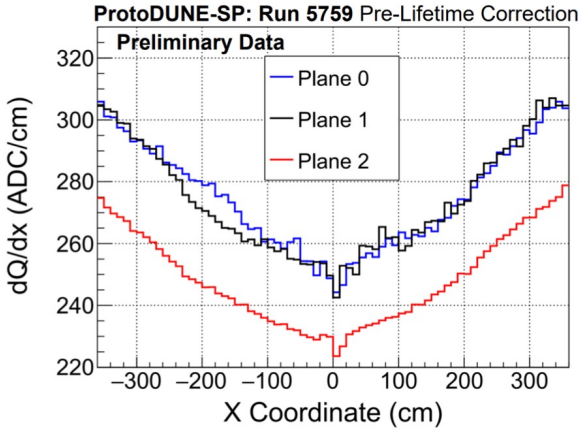
Why only ~100 rows for lifetime but ~6 Mio for SCE?

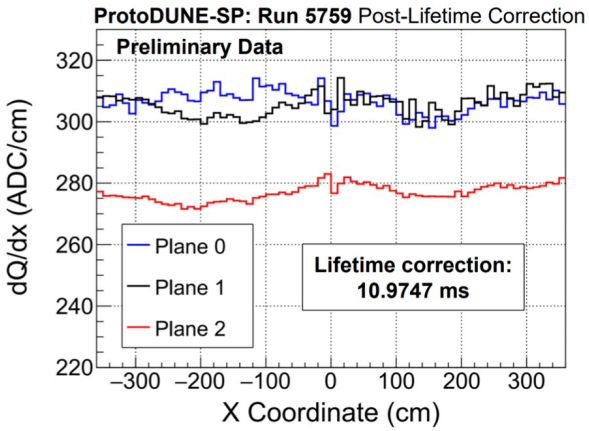
Lifetime calibration

- assume exp. func.:

$$Q(t) = Q_0 \exp(-(t_{hit} - t_0/\tau))$$

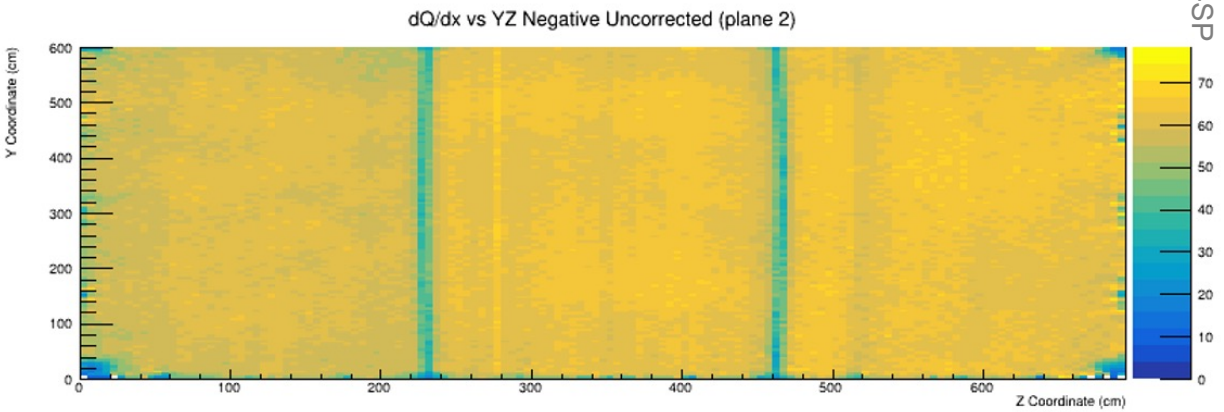
- Only pass param τ





SCE calibration

- dQ/dx from cosmic μ
- 5x5 cm^2 grid in YZ
- 5cm grid in X



from Mitchell mote (DRA meeting, 19.05.21)

Conclusion

- Conditions DB at Fermilab is used for calibration
  - Two basic kinds: electron lifetime & SCE (the latter is split into 3 tables)
  - Accessed via HTTP service
- The accessed data volume varies. For Run 1:
  - Min: Lifetime: 4 float cols, 107 rows
  - Max: dQ/dx in YZ-plane: 8 float cols, >6 Mio rows
    - Can probably be reduced

Thank you for your attention!

Questions?

Applying the calibration

- ProtoDUNE Data reconstruction is split into two steps
  - Both apply electron lifetime correction
  - Only second step applies XYZ corrections

```
# lifetime database
services.LifetimeCalibService.IsMC: false
services.LifetimeCalibService.UseCondbLifetime: true
services.LifetimeCalibService.LifetimeDBTag: "v1.1"

# xyzt database
services.XYZCalibService.IsMC: false
services.XYZCalibService.UseCondbXYZCorr: false
#services.XYZCalibService.YZCorrDBTag: "prod4"
#services.XYZCalibService.XCorrDBTag: "prod4"
#services.XYZCalibService.NormCorrDBTag: "prod4"
```

from protoDUNE_SP_keepup_decoder_reco_stage1.fcl

```
# lifetime database
services.LifetimeCalibService.IsMC: false
services.LifetimeCalibService.UseCondbLifetime: true
services.LifetimeCalibService.LifetimeDBTag: "v1.1"

# xyzt database
services.XYZCalibService.IsMC: false
services.XYZCalibService.UseCondbXYZCorr: true
services.XYZCalibService.YZCorrDBTag: "prod4"
services.XYZCalibService.XCorrDBTag: "prod4"
services.XYZCalibService.NormCorrDBTag: "prod4"
```

from protoDUNE_SP_keepup_decoder_reco_stage2.fcl

Reading data from lifetime calib table

All code shown here is from dunecalib/dunecalib/Calib/LifetimeCalibProtoDUNE.cxx

- Initialize & configure a 'Table' object (from nutools)

```
nutools::dbi::Table LifetimePurMonTable;
LifetimePurMonTable.SetDetector("pdunesp");
LifetimePurMonTable.SetTableName("lifetime_purmon");
LifetimePurMonTable.SetTableType(nutools::dbi::kConditionsTable);
LifetimePurMonTable.SetDataTypeMask(nutools::dbi::kDataOnly);
```

```
// nutools includes
#include "nuevdb/IFDatabase/Table.h"
```

- Load data from database via .Load() method

```
if (!fLifetimeFileName.empty())
  readOk = LifetimePurMonTable.LoadFromCSV(fLifetimeFileName);
else
  readOk = LifetimePurMonTable.Load();

if (! readOk) {
 mf::LogError("LifetimeCalibProtoDUNE") << "Load from lifetime calib database table failed.";
```

- If activated, interpolate between last & next values:

```
// do linear interpolation using previous one and following one
// refer: https://en.wikipedia.org/wiki/Linear_interpolation
interpolate_lifetime.center = ( loaded_center[t0_idx]*(loaded_tv[t1_idx]-fCurrentTS) + loaded_center[t1_idx]*(fCurrentTS-loaded_tv[t0_idx]) ) / (loaded_tv[t1_idx]-loaded_tv[t0_idx]);
interpolate_lifetime.low = ( loaded_low[t0_idx]*(loaded_tv[t1_idx]-fCurrentTS) + loaded_low[t1_idx]*(fCurrentTS-loaded_tv[t0_idx]) ) / (loaded_tv[t1_idx]-loaded_tv[t0_idx]);
interpolate_lifetime.high = ( loaded_high[t0_idx]*(loaded_tv[t1_idx]-fCurrentTS) + loaded_high[t1_idx]*(fCurrentTS-loaded_tv[t0_idx]) ) / (loaded_tv[t1_idx]-loaded_tv[t0_idx]);
```

- Else, use latest previous entry in DB

Tracking down actual DB access

https://internal.dunescience.org/doxygen/Table_8cpp_source.html

- Table::Load() calls Table::LoadConditionsTable()

```
bool Table::Load()
{
  if (Util::RunningOnGrid()) {
    fConnectionTimeout = 1800;
  }
  if (fTableType==kConditionsTable)
    return LoadConditionsTable();
  else if (fTableType==kUnstructuredConditionsTable)
    return LoadUnstructuredConditionsTable();
  else
    return LoadNonConditionsTable();
}
```

- Table::LoadConditionsTable constructs search string for the web query:

```
std::stringstream myss;

myss << fWSURL << "get?table=" << Schema() << "." << Name() << "&";

if (fDataTypeMask > kNone) {
  myss << "type=";

  if ((fDataTypeMask & kMCOnly)) myss << "mc";
  if ((fDataTypeMask & kDataOnly)) myss << "data";

  myss << "&";
}
```

- Table::GetDataFromWebService() calls getDataWithTimeout()

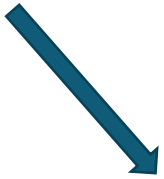
```
bool Table::GetDataFromWebService(Dataset& ds, std::string myss)
{
  Tuple tu;
  char ss[1024];
  char ss2[1024];
  int wda_err, err;
  std::vector<int> colMap(fCol.size());
  std::vector<bool> isString(fCol.size());
  std::vector<bool> isKnownField(fCol.size());

  const char* uagent = NULL;

  if(fVerbosity > 0)
    std::cout << "DBWeb query: " << myss << std::endl;

  boost::posix_time::ptime ctt1;
  boost::posix_time::ptime ctt2;

  if (fTimeQueries) {
    ctt1 = boost::posix_time::microsec_clock::local_time();
  }

  ds = getDataWithTimeout(myss.c_str(), uagent,
                          fConnectionTimeout, &wda_err);
```

What about ‘LinCalib’?

```
-bash-4.2$ ls dunecalib/dunecalib/Calib
CMakeLists.txt            LifetimeCalib.h            LinCalib.h             XYZCalib.h
getLinConstsProtoDUNE.cc  LifetimeCalibProtoDUNE.cxx LinCalibProtoDUNE.cxx XYZCalibProtoDUNE.cxx
getXYZCalibProtoDUNE.cc   LifetimeCalibProtoDUNE.h   LinCalibProtoDUNE.h   XYZCalibProtoDUNE.h
```

Not used in any .fcl file

Table does not exist



```
t.SetDetector("pdunesp");
t.SetTableName("linconsts");
t.SetTableType(nutools::dbi::kConditionsTable);
t.SetDataTypeMask(nutools::dbi::kDataOnly);
if (fIsMC)
  t.SetDataTypeMask(nutools::dbi::kMCOnly);

int statusIdx = t.AddCol("status","int");
int gainIdx   = t.AddCol("gain","float");
int offsetIdx = t.AddCol("offset","float");
int shapeIdx  = t.AddCol("shape","float");
int chi2Idx   = t.AddCol("chi2","float");
int adcLowIdx = t.AddCol("adc_low","int");
int adcHiIdx  = t.AddCol("adc_high","int");
```

Which columns are accessed?

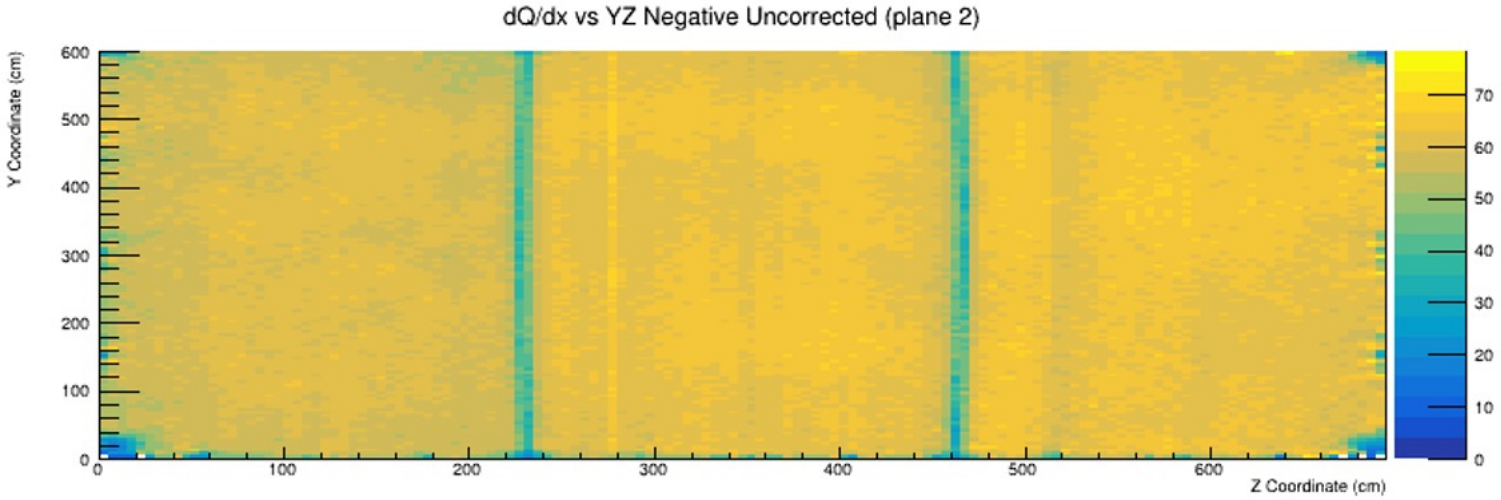
```
YZCorrTable.SetDetector("pdunesp");
YZCorrTable.SetTableName("distcorryz");
YZCorrTable.SetTableType(nutools::dbi::kConditionsTable);
YZCorrTable.SetDataTypeMask(nutools::dbi::kDataOnly);
if (fIsMC)
  YZCorrTable.SetDataTypeMask(nutools::dbi::kMCOnly);

int corrIdx = YZCorrTable.AddCol("corr","double");
int corrErrIdx = YZCorrTable.AddCol("corr_err","double");
int yIdx = YZCorrTable.AddCol("y","double");
//  int dyIdx = YZCorrTable.AddCol("dy","double");
int zIdx = YZCorrTable.AddCol("z","double");
//  int dzIdx = YZCorrTable.AddCol("dz","double");
```

```
int shapeIdx = XCorrTable.AddCol("shape","double");
int shapeErrIdx = XCorrTable.AddCol("shape_err","double");
int xIdx = XCorrTable.AddCol("x","double");
int dxIdx = XCorrTable.AddCol("dx","double");
```

```
int normIdx = NormCorrTable.AddCol("norm","double");
int normErrIdx = NormCorrTable.AddCol("norm_err","double");
```

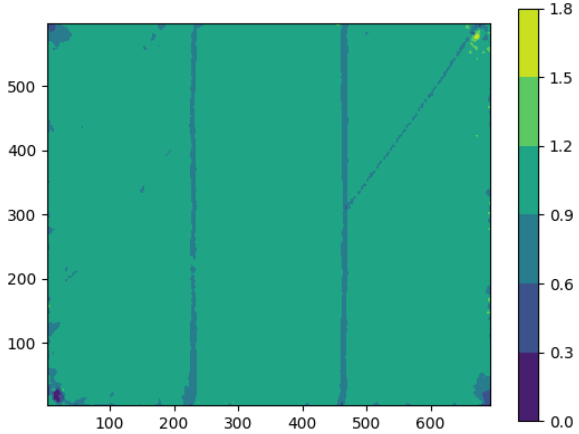
```
int centerIdx = LifetimePurMonTable.AddCol("center", "double");
int lowIdx = LifetimePurMonTable.AddCol("low", "double");
int highIdx = LifetimePurMonTable.AddCol("high", "double");
```

dQ/dx Corrections (Space Charge Effect)



From Mitchell mote (DRA meeting, 19.05.21)

- Observe charge distributions from cosmic muons
  - Derive inverse functions (for ZY-planes, in X, and norm.)



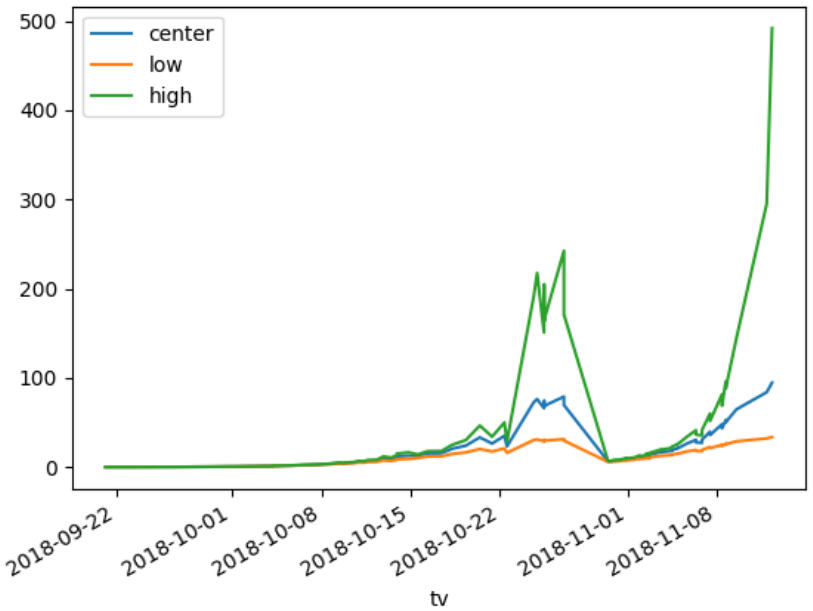
Inverse correction from database
In 1st YZ plane (run 5842)

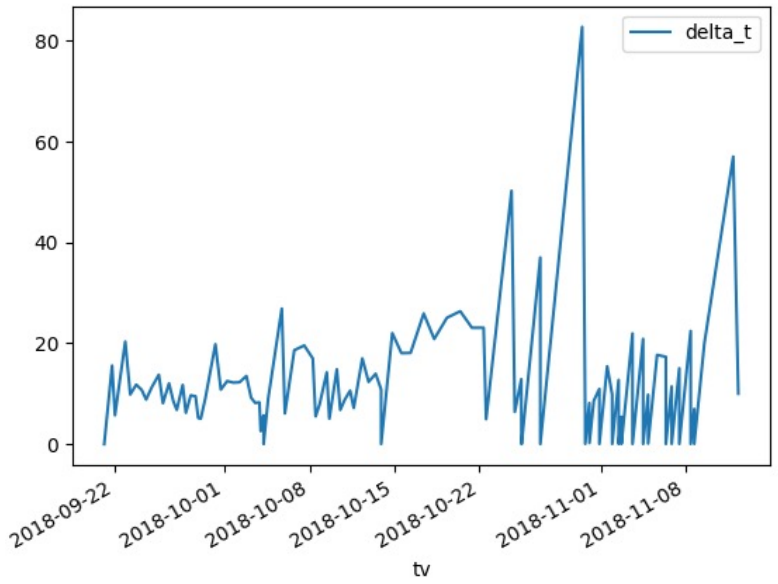
Channel must be == 1:

```
if (chan != 1) {
  mf::LogError("LifetimeCalibProtoDUNE") << "Channel numuber in lifetime calib table is not  1.  This should never be the case!";
  return false;
}
```

```
if (!fInterpolate) {
  LifetimePurMonTable.SetMinTSVld(fCurrentTS); // only load one previous lifetime
  LifetimePurMonTable.SetMaxTSVld(fCurrentTS);
}
else {
  // load lifetime IOV: runtime +- 1 days for interpolation
  LifetimePurMonTable.SetMinTSVld(fCurrentTS-1*86400.);
  LifetimePurMonTable.SetMaxTSVld(fCurrentTS+1*86400.);
}
```

Lifetime Calibration Intervals





- Frequency & variation of values don’t seem correlated

Content of XYZ Calibrations

- Indexed by run number & ‘channel’ (combination is unique)
  - Consider Run 1: 92 run numbers contain values
  - Requiring a run number that does not contain values, returns the last run number that does contain values. Example:

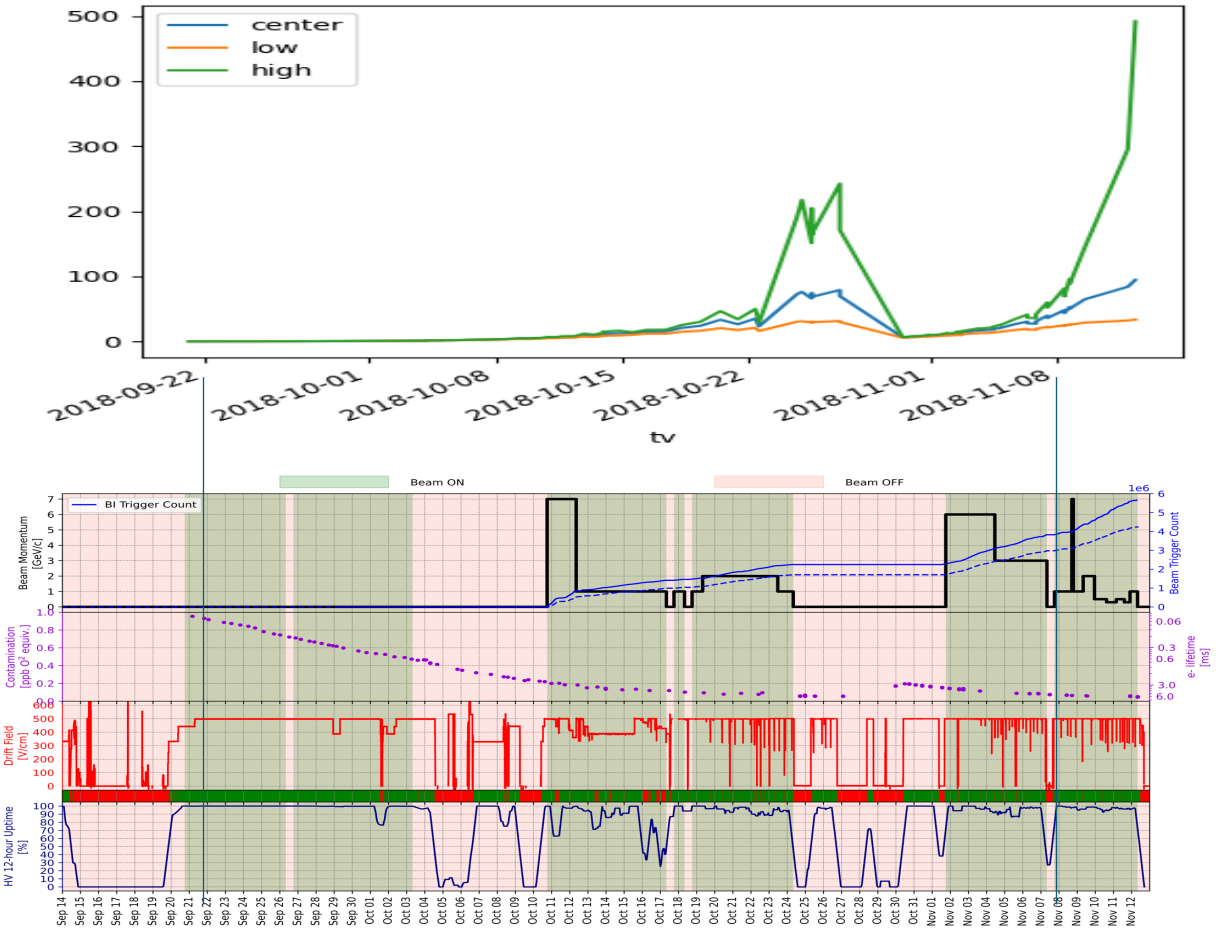
```
macbooklino:fnalcondbanalyzer linogerlach$ curl "https://dbdata0vm.fnal.gov:9443/dune_con_prod/app/ge
t?table=pdunesp.distcorrx&t=5809&columns=x&type=data&"
channel,tv,x
1,5809.000,-357.5
2,5809.000,-352.5
```

```
macbooklino:fnalcondbanalyzer linogerlach$ curl "https://dbdata0vm.fnal.gov:9443/dune_con_prod/app/ge
t?table=pdunesp.distcorrx&t=5808&columns=x&type=data&"
channel,tv,x
1,5797.000,-357.5
2,5797.000,-352.5
```

Data volume details:

- distcorrx: 39,728 rows
  - varying number of channels per run (431 or 432)
  - values for all 92 runs
- distcorryz: 6,197,659 rows
  - varying number of channels per run (99717 to 100080)
  - only has values for 62 runs
- distcorrnorm: 276 rows
  - 3 channels (0,1,2) per run
  - values for all 92 run numbers

Electron Lifetime vs unstable HV



Strong variations in electron lifetime don't seem correlated with periods of unstable voltage