Checks of the ToF-dEdx in 10th reduction

General remark: It seems there is some strange problem of the "ToFTrack" class in 10th reduction. What we have:
1) The raw ADC values in the ToFPMT class, you derive them doing a loop like:

```
for (Int_t ipmt=0; ipmt<pam_event->GetToFLevel2()->npmt() ; ipmt++){
ToFPMT *tofpmt = pam_event->GetToFLevel2()->GetToFPMT(ipmt);
ipm = tofpmt->pmt_id;
adcraw[ipm] = tofpmt->adc ;  // raw ADC values
```

2) The ToFTrkVar class: ToFTrkVar* toftrack = pamtrack->GetToFTrack();
You derive the dEdx for each PMT doing a loop like

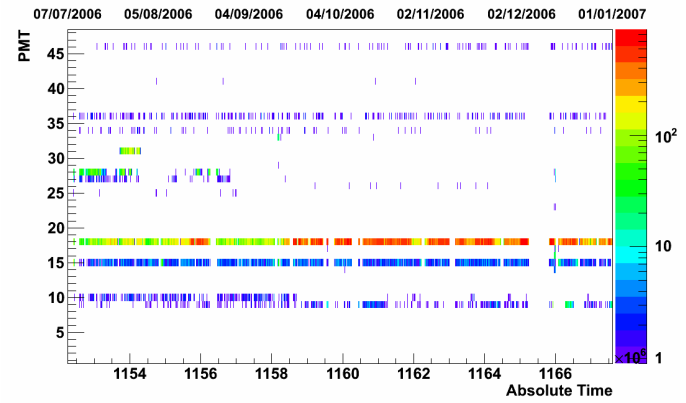
```
for (Int_t ipmt=0; ipmt<toftrack->npmtadc; ipmt++){
      Int_t pmtadc = toftrack->pmtadc[ipmt];
      Float_t dEdx = toftrack->dedx[ipmt];
```

In the 10th reduction I found problems with this ToFTrkVar class: As a check I looked at the paddle penetrated by the track using "GetpaddleIdOfTrack":

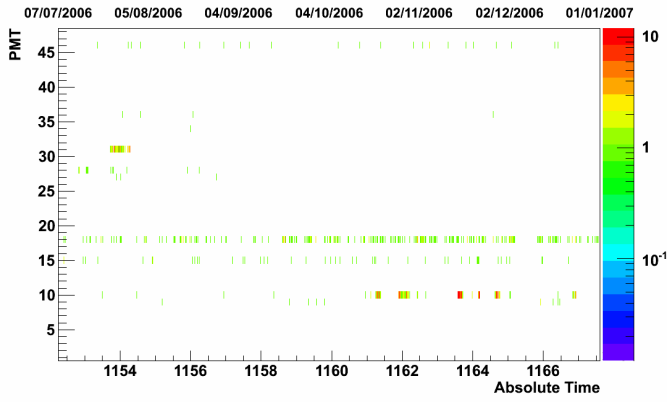
```
for (Int_t  ilay=0; ilay<6; ilay++)
paddleidoftrack[ilay] = pam_event->GetToFLevel2()->GetPaddleIdOfTrack(xv[ilay], yv[ilay], ilay, 0.0) ;
```

In each of the hitted paddles I checked the raw ADC values from the ToFPMT class. If the ADC value was "good" (less than 4095), this means the PMT had seen something, and it should show up in the ToFTrkVar class.
If this is NOT the case, I flag the PMT. Selecting either protons or helium and then plotting each PMT flag vs. the time for the 10th reduction:

Protons 2006 10th

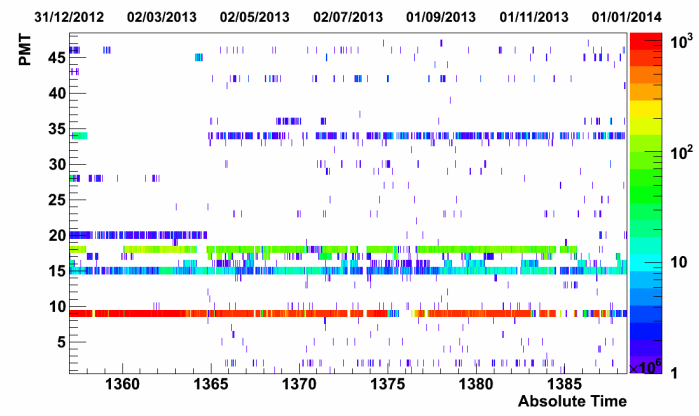


Helium 2006 10th

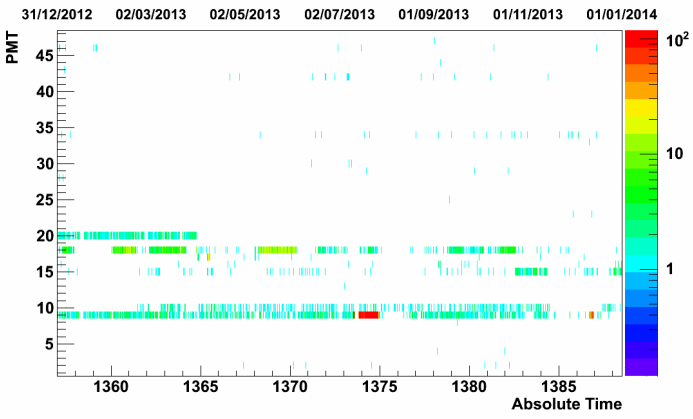


So here in the proton selection there are PMTs like 15, 18, 28 which are missing quite often, but some periods are not as bad as others... In the 9th reduction this picture looked totally different, where the rate of missed PMTs was practically the same for all PMTs. In 2013 the picture looks like this

Protons 2013 10th



Helium 2013 10th



I did NOT debug DarthVader to find out the differences in the code between 9th and 10th reduction, for my patch I will use "GetpaddleIdOfTrack" to define the hitted paddle and take the raw ADC values from the ToFPMT class)

Calculation of the ToF dEdx in the 10th reduction

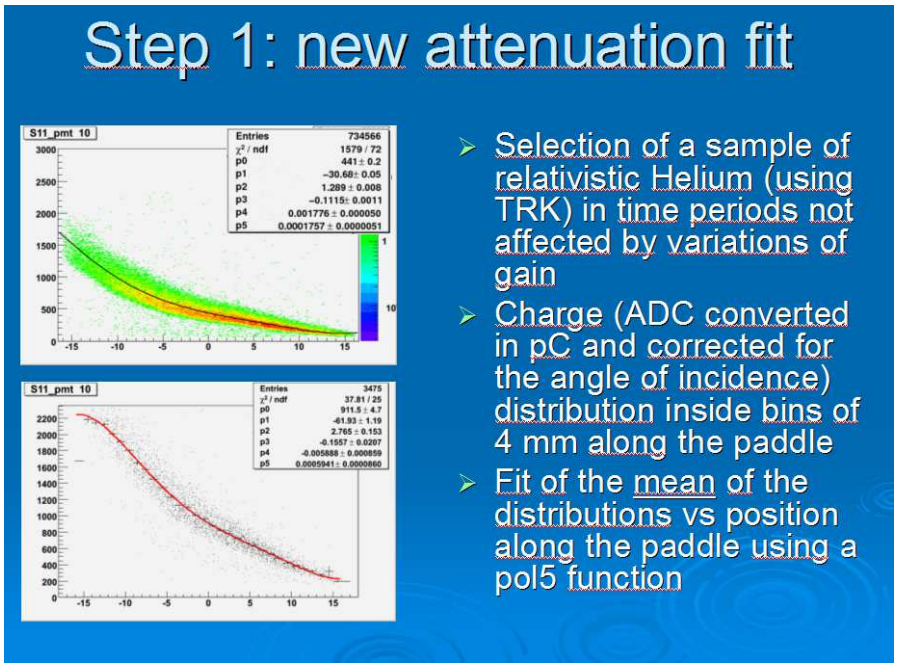
The calculation of the dEdx is done in /DarthVader/ToFLevel2/src/ToFLevel2.cpp
where a “ToFdEdx” class is defined. Most of the functions are exactly the same as in the **ToFNaNuclei** package. I recommend to look again in Rita Carbone’s talks (I used some of the figures to explain the methods):
http://pamela.roma2.infn.it/technical-docs/18th_Pamela_sw_meeting/Carbone-nuclei-18th_SWMeeting.ppt
http://pamela.roma2.infn.it/technical-docs/19th_Pamela_sw_meeting/Carbone_Nuclei_ToFCal_Lindau2009.ppt

In /DarthVader/ToFLevel2/src/ToFCore.cpp there is a call to “tofdedx->Process” and then the dEdx values are filled into the data.

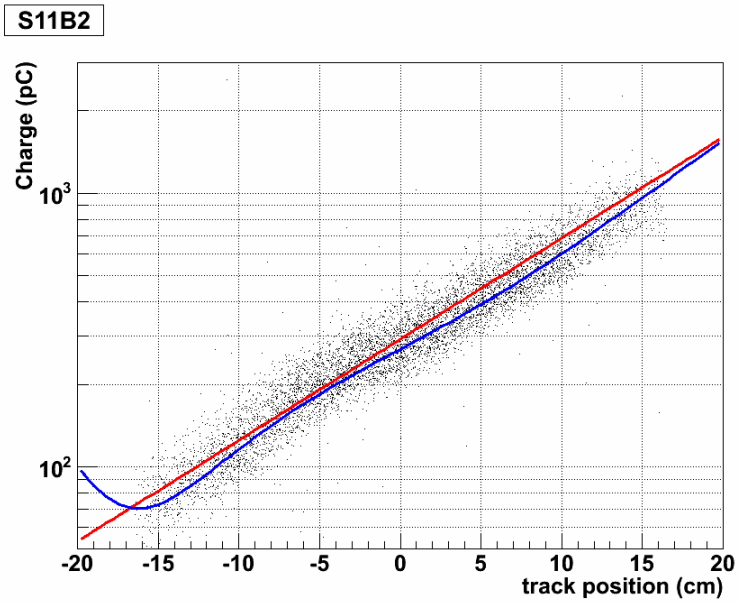
How the calculation of the dEdx works:

In the very first step the raw ADC value is converted to charge in picoCoulomb using a calibration function derived by Napoli.
The incident angle is calculated from the track positions in S11 and S32 and is taken constant in all scintillator layers.

Now the first step: Attenuation in the paddle



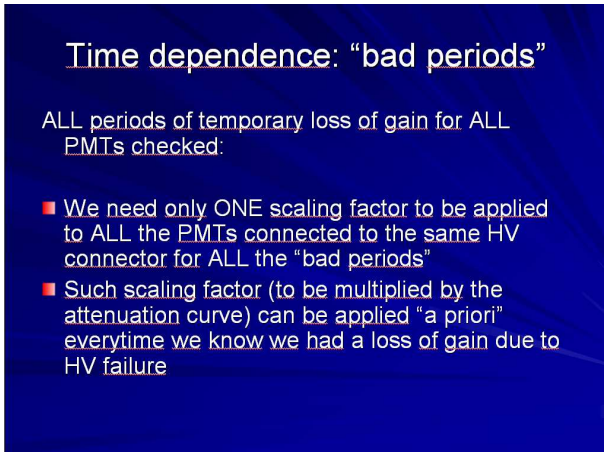
Here the attenuation is a "pol5" fit derived for relivistic helium events, while I use a double exponential fit for the ToF timing routines.
As an example the fits for PMT S11B2 for july 2006, Rita's fit in blue, mine in red:



The “pol5” is better to describe the variations in the paddle, but often has oscillations at the edges like in this case.

In the ToFNaNuclei pacckage there was only one attenuation used for the whole 8th reduction period, now these attenuation fits are done in time intervals: From 1 month intervals in the beginning to 3 - 4 months in 2014. The function "ReadParAtt" reads the text files with the parameters of the fit.

HV-failure periods: The periods with HV-failures are put in the database with the failure times of the specific connectors (hardcoded in ToFNaNuclei): In this period all four PMTs of the connector are treated with a correction factor.



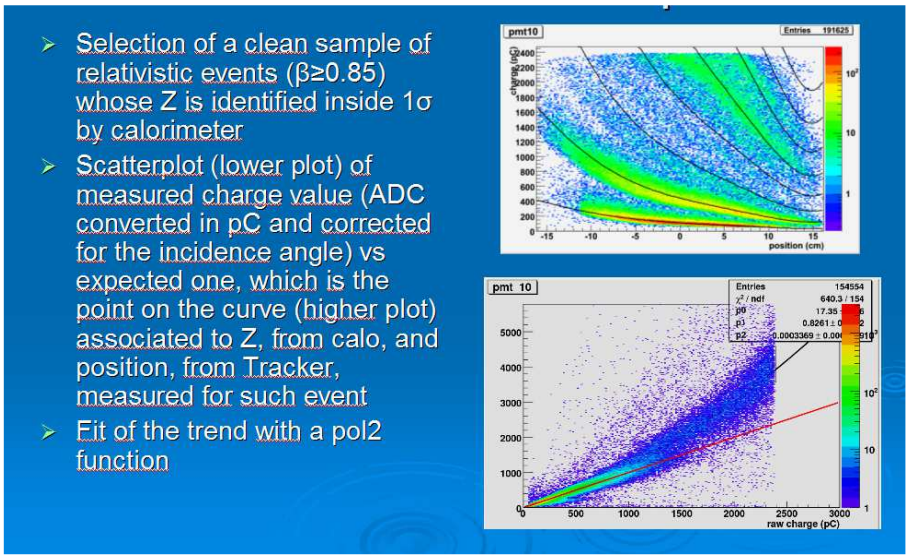
Snippet of the code in ToFLevel2.cpp:

```
Double_t correction = 1.;

if(Aconn==1 && (ii==0 || ii==20 || ii==22 || ii==24)){
  correction = 1.675;
}
else if(Bconn==1 && (ii==6 || ii==12 || ii==26 || ii==34)){
  correction = 2.482;
}
else if(Cconn==1 && (ii==4 || ii==14 || ii==28 || ii==32)){
  correction = 1.464;
}
else if(Dconn==1 && (ii==2 || ii==8 || ii==10 || ii==30)){
  correction = 1.995;
}
else if(Econn==1 && (ii==42 || ii==43 || ii==44 || ii==47)){
  correction = 1.273;
}
else if(Fconn==1 && (ii==7 || ii==19 || ii==23 || ii==27)){
  correction = 1.565;
```

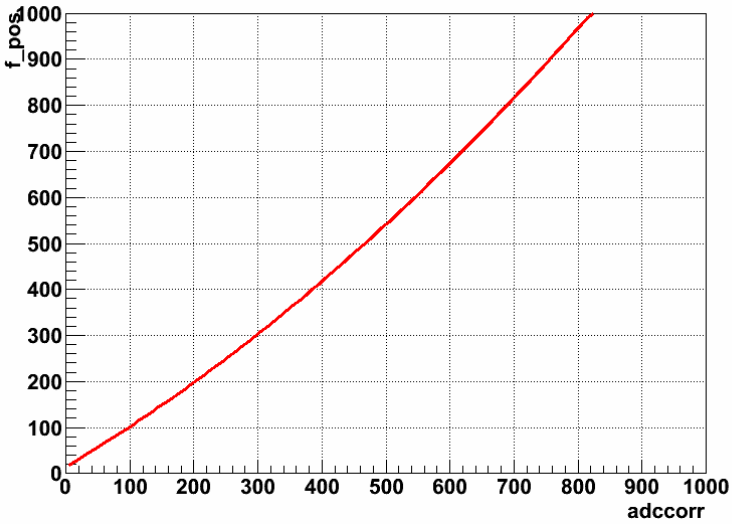
I found that a period in 2006 was missing in the database. (In my analysis for a patch I found some PMTs which did not behave like the others for this connector, but made changes to the original code only in one case).

Next step: Correction for nonlinearity of the PMT



The function "ReadParPos" reads the file "desaturation_position.txt" which contains parameters for "pol3" functions for each PMT. Parameters are exactly the same as in ToFNaNuclei.

Here an example:
X-axis is "adccorr" which is the raw charge in pC corrected with cos(theta), the y-axis is the value of the function "f_pos".
One sees the nonlinear behaviour of the PMT:

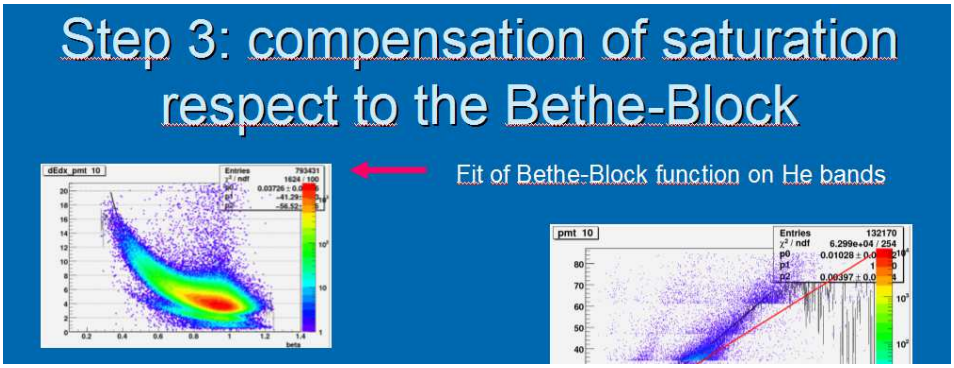


Next step:
Definition of a first preliminary dEdx using the value of the "f_pos" function above, the value of the attenuation function according to the position in the paddle, and then scaled to 4 mip (dEdx = 4.*f_pos/attenuation), this value is called "adclin"
This is in principle the same method I use to get 8th-reduction-dEdx, since at low light yield the value of the nonlinearity function "f_pos" is the same as the PMT signal.

Next step:
If there is no beta, the value of "adclin" is used as the dEdx for this PMT, and the further calculations are **not** done:

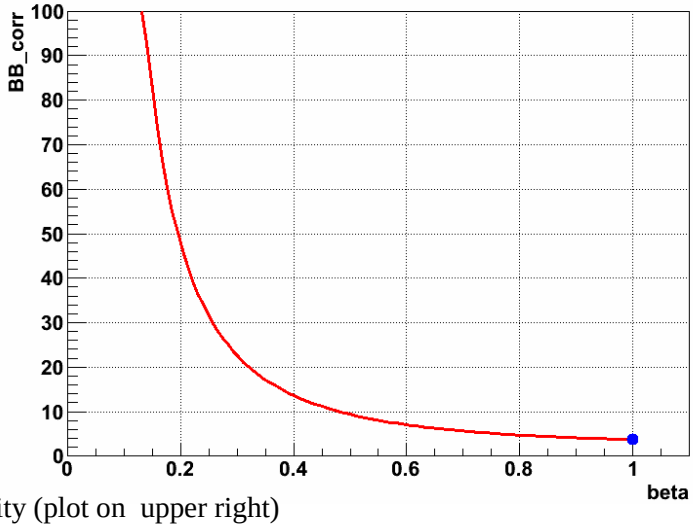
```
//
if ( betamean > 99. ){
  //      eDEDXpmt.AddAt((Float_t)adclin,ii);
  eDEDXpmt->AddAt((Float_t)adclin,ii);
  //      printf(" AAPMT IS %i dedx is %f vector is %f \n",ii,adclin,eDEDXpmt[ii]);
  if ( debug ) printf(" %i betamean > 99 \n",ii);
  continue;
};
//
```

This is a bug, since for the next steps of the dEdx calculations the beta is NOT needed! The beta **was** needed in ToFNaNuclei to calculate the charge:

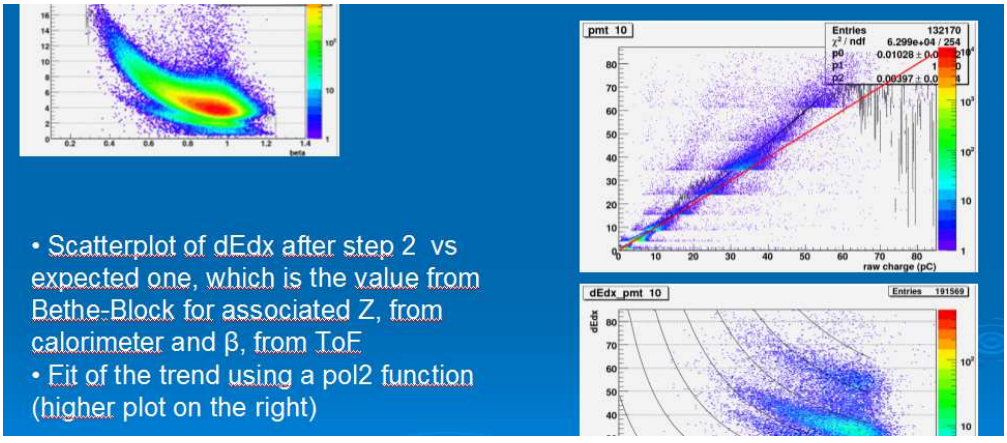


For each PMT there is a curve "f_BB" if beta<1 (see red curve on the right) plus fix value for beta > 1 (blue dot on the right)

This curve was then used in ToFNaNuclei to calculate a charge.
But in ToFLevel2.cpp this is not used, so the cut on good beta makes no sense, but leads to some problems (see below)

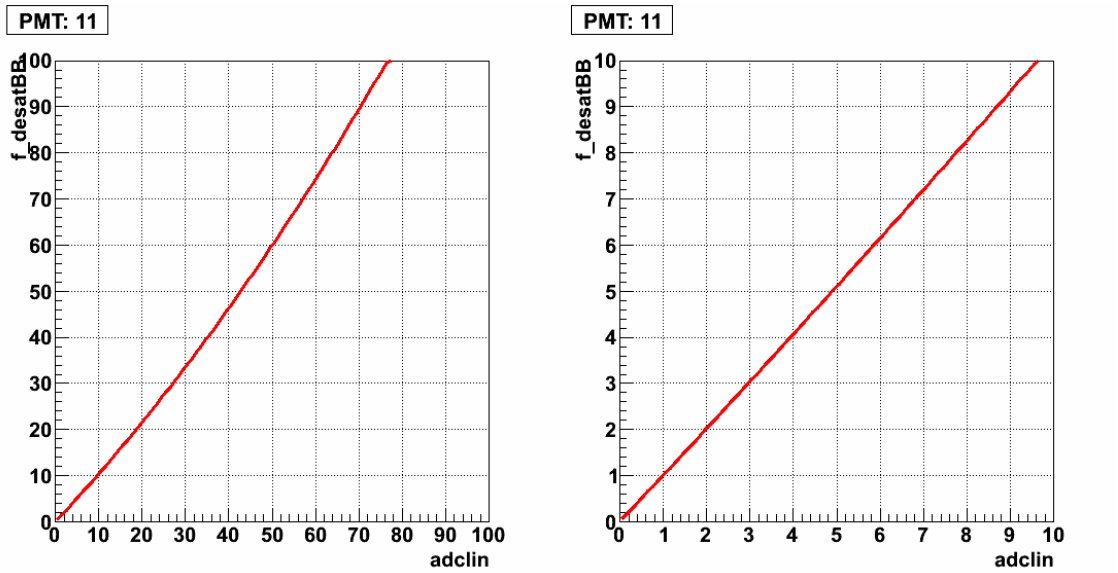


In the next step there is another correction due to nonlinearity (plot on upper right)



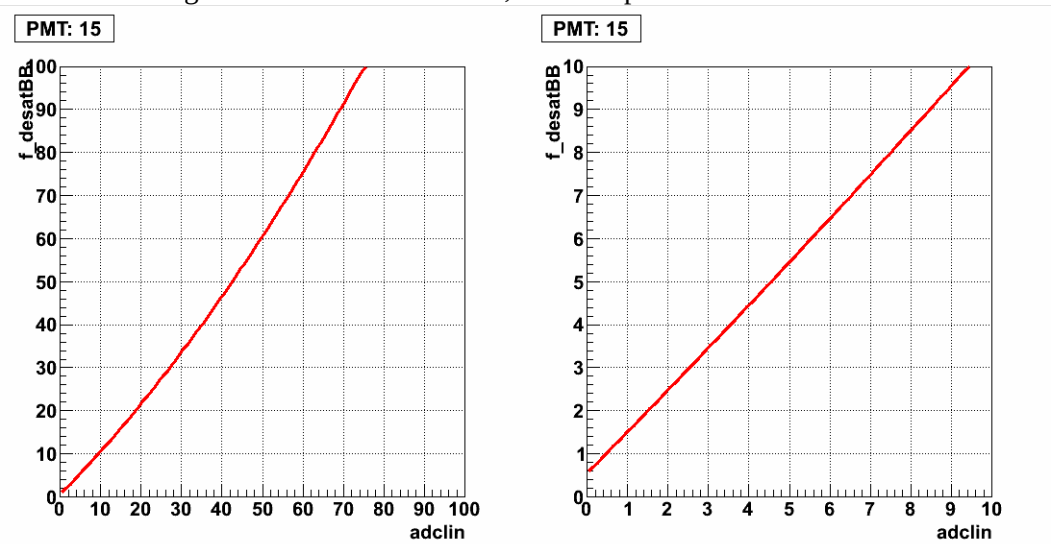
In the function “ReadParDesatBB” the file “desaturation_beta.txt" is read, it contains parameters for a ”pol2” fit for each PMT (exactly the same as in ToFNaNuclei). The input is “adclin” (so “x” is in dEdx) and the output is then a corrected dEdx value.

Typical “f_desatBB” function: Example PMT #11, left figure x-axis from 0 - 100, right figure x-axis from 0 -10



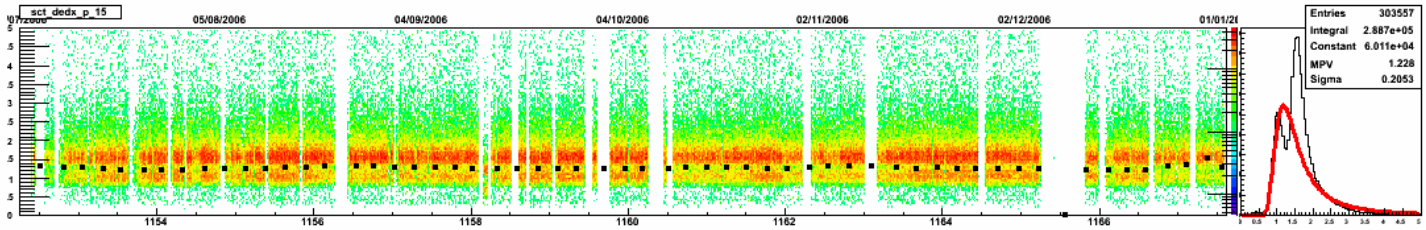
In my understanding this is a nonlinearity for high dEdx due to further detector effects like Birks’ saturation of the scintillator etc.. For this PMT the behaviour for small “adclin” is quite linear as one would expect.

However, some PMTs have a big offset at low dEdx values, for example PMT #15:



So if “adclin” is for example “1”, the corrected value would be ca. 1.5

If one selects protons and looks at the response of PMT #15 vs time, one sees a double band structure:



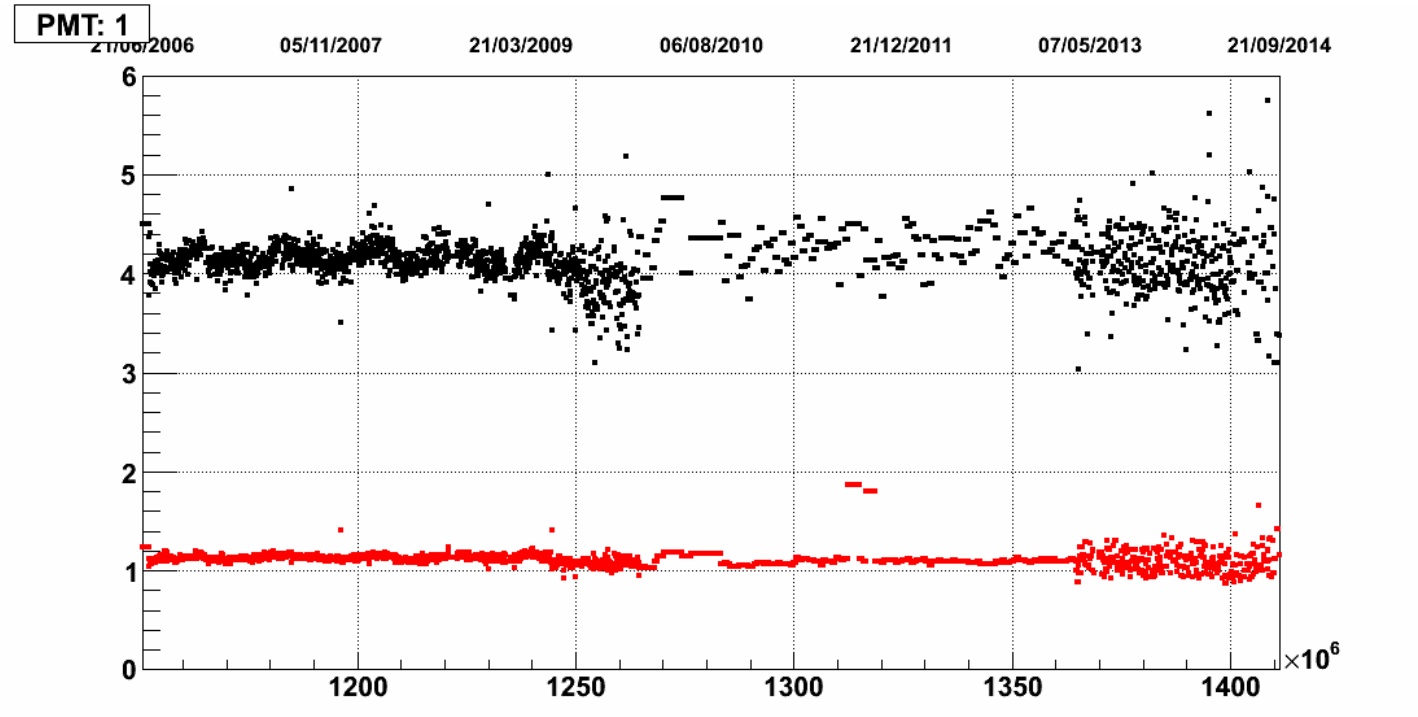
If one selects only good beta, only the upper band survives, because in this case always the “f_desatBB” function is used. As said before, if beta is not good, the code is exiting and the simple “adclin” is used as a dEdx.

In principle the calculation of the dEdx for this PMT is finished now. If one plots the dEdx for each PMTs vs time for protons or helium (or carbon), one will see if/how the dEdx varies a little bit with the time. Also the dEdx values will be probably be not exactly at 1 mip for protons or 4 mip for helium.

Therefore next step is a second order correction: In a text file the mean values for protons and helium for each PMT are stored for each day. For the 10th reduction there are two “biscale” files with values for protons and helium, here the snippet from the database:

biscale_tofdedx_x10thred.txt",1,1364790400,203," ToF dE/dx II order correction parameters - Rita"
biscale_tofdedx_x10thred_corrected2014.txt",1364790400,4294967295,203," ToF dE/dx II order correction parameters - EM

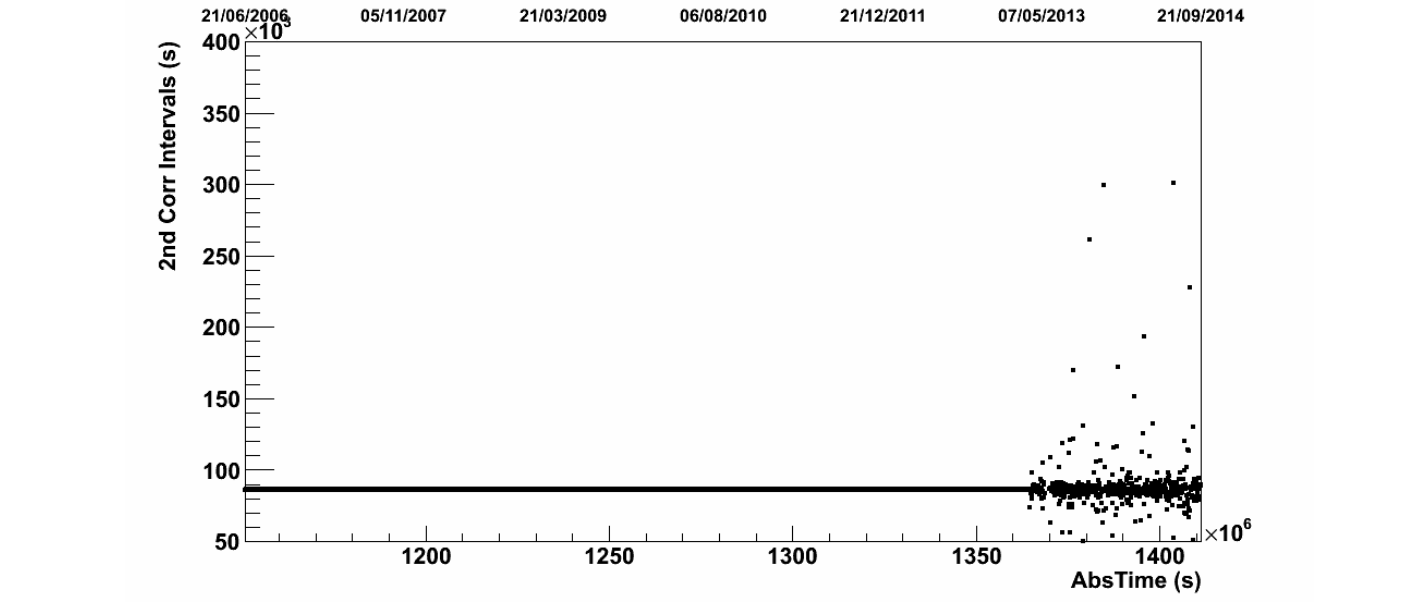
Plotting the values for PMT #1 vs. time, black dots helium, red dots protons:



To me it seems that there was a change in the method to derive the correction values in spring 2010, and then again a change when Emiliano did the correction in april 2013.

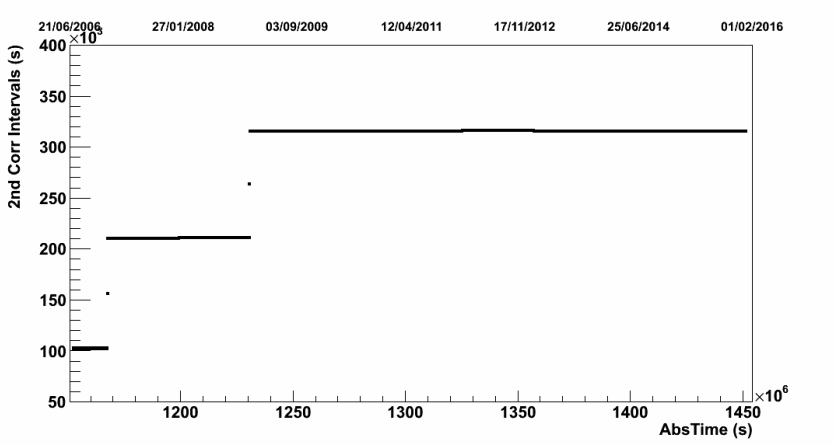
In any case, with the values for protons and helium one can derive the intercept and slope of a linear function (setting protons to 1 MIP and Helium to 4 MIP), then the final dEdx value is: dEdx = inter + slope * dEdx_1st_order
Snippet of the code in ToFCore.cpp:
*t_tof->dedx.AddAt((inter_dedx[pmt_id]+slope_dedx[pmt_id]*tofdedx->GetdEdx_pmt(pmt_id)),t_tof->npmtadc);// RC new dE/dx II order correction*

The time intervals are constant “one day” for Rita's correction and then for Emiliano’s values –not clear to me why- some variations:



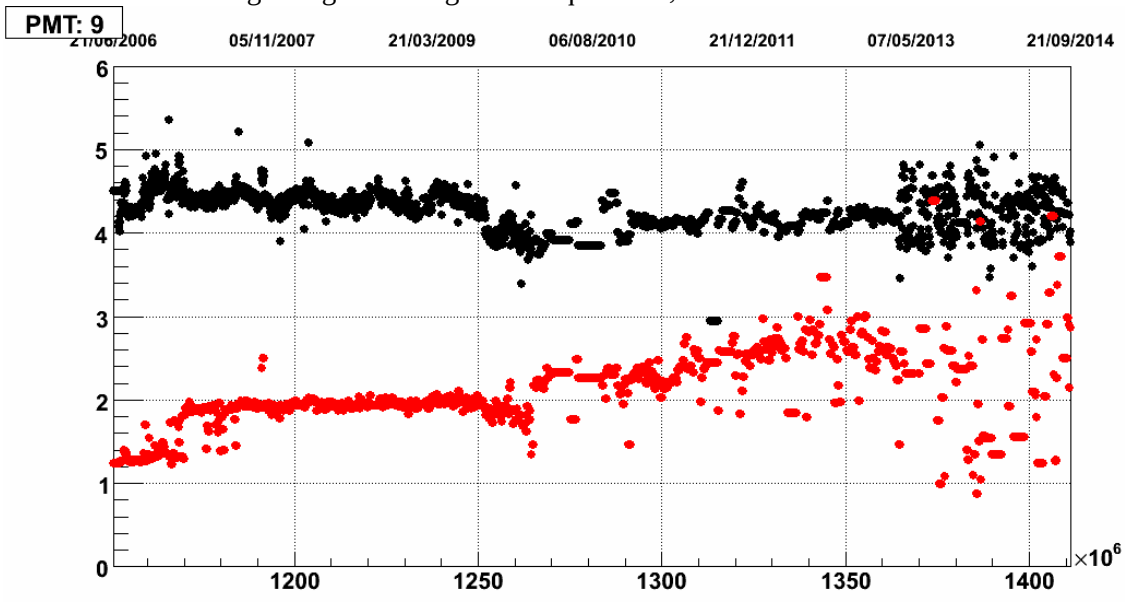
For a comparison I show the time intervals I use for my „ToFPatch“ 2nd-order correction:

I use ~100000 sec at the beginning and then Change to ~ 200000 sec and ~300000 sec later.

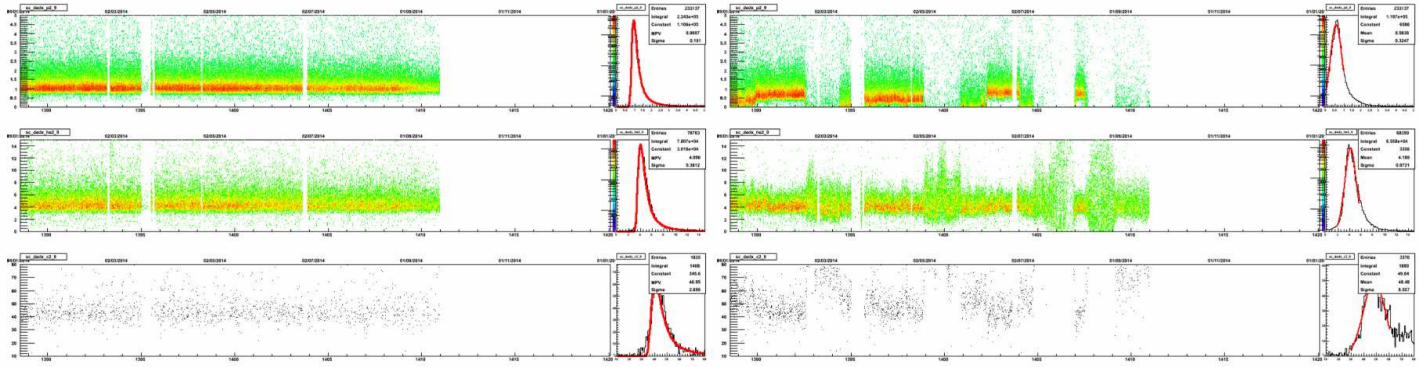


While PMT 1 looked more or less reasonable, a lot of PMTs show a strange behaviour:

For example PMT #9: In the beginning of the flight looks quite OK, but at the end....

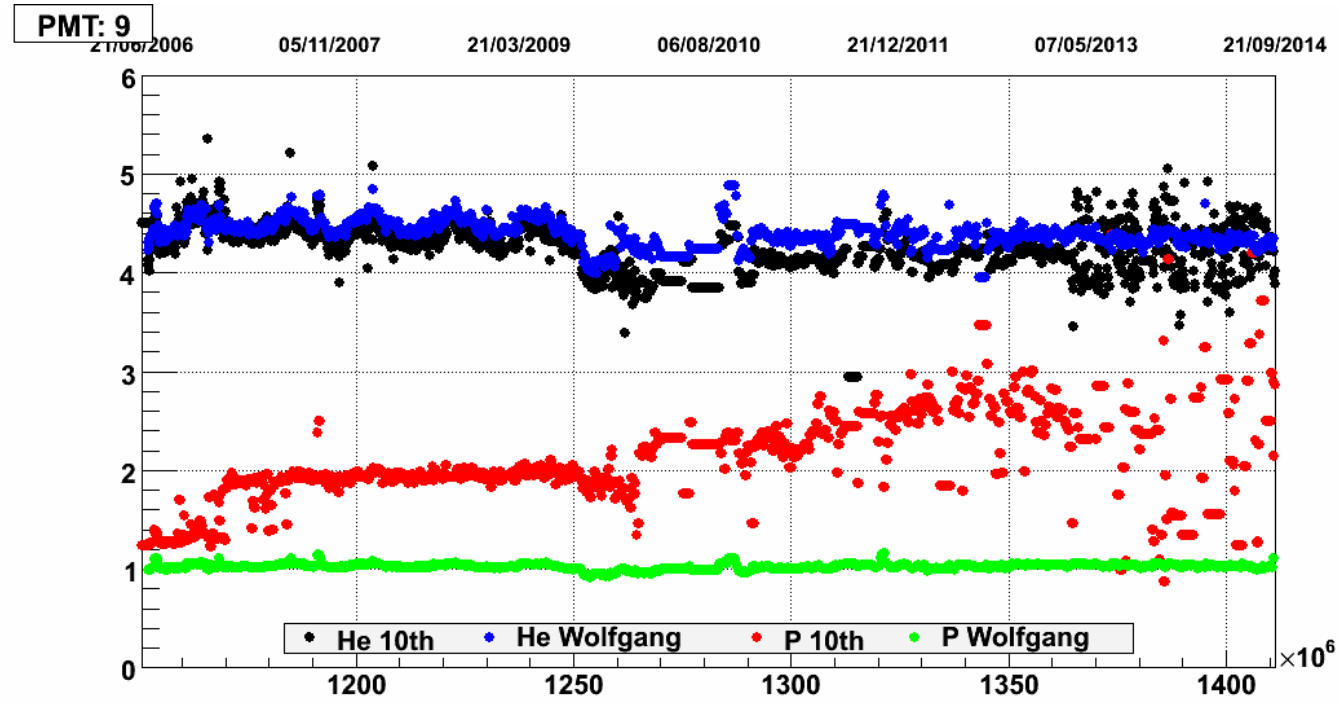


This second order correction has serious effects: In the next plot on the left side there are the results for PMT #9 in 2014 **without** the second order correction, on the right side **with** the second oder correction. From top to bottom protons, helium, and carbon.



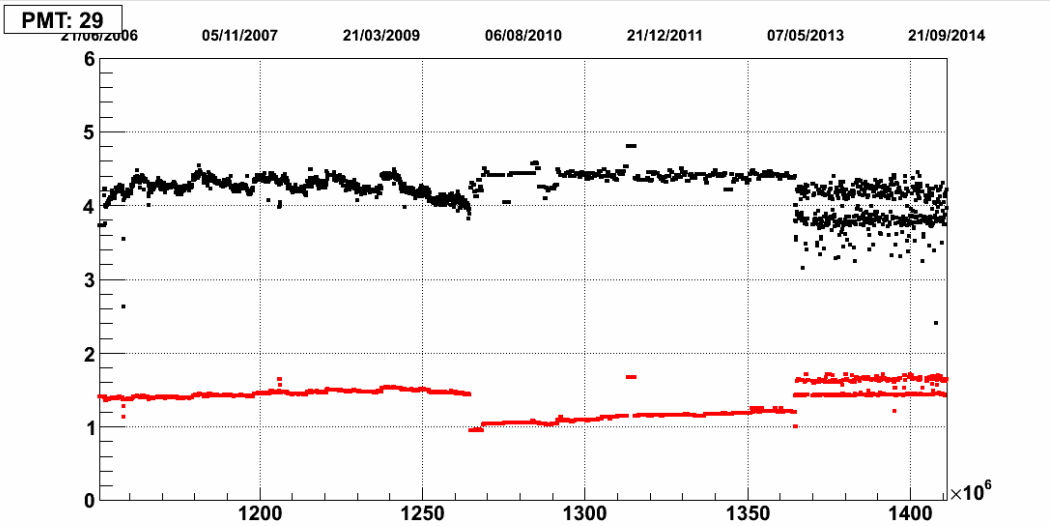
As one can see, the results without the second order correction look quite OK, while on the right side especially the protons look very bad, with huge fluctuations, some periods where the protons are missing at all..

If I run the first order procedures explained above (using time intervals as in my "ToFPatch" shown above) and then do the 2nd order correction by myself, I get the following results (blue and green points):

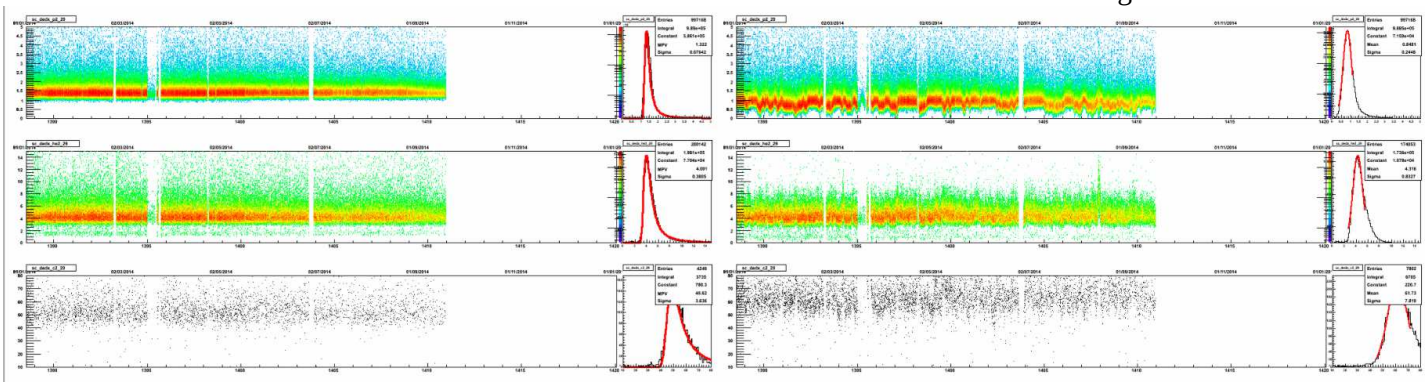


Why the current 10th-reduction correction shows this behaviour I cannot say since I don't know how exactly it was done... It would be normal to have always a systematic difference between the two corrections (look at the helium up to 2009 in the upper figure) since the selection criteria for protons and helium might differ (I select events with rigidities > 2 GV)

Another example:
PMT #29: For the10th red a sharp step in spring 2010, and a double band structure for protons and helium at the end:

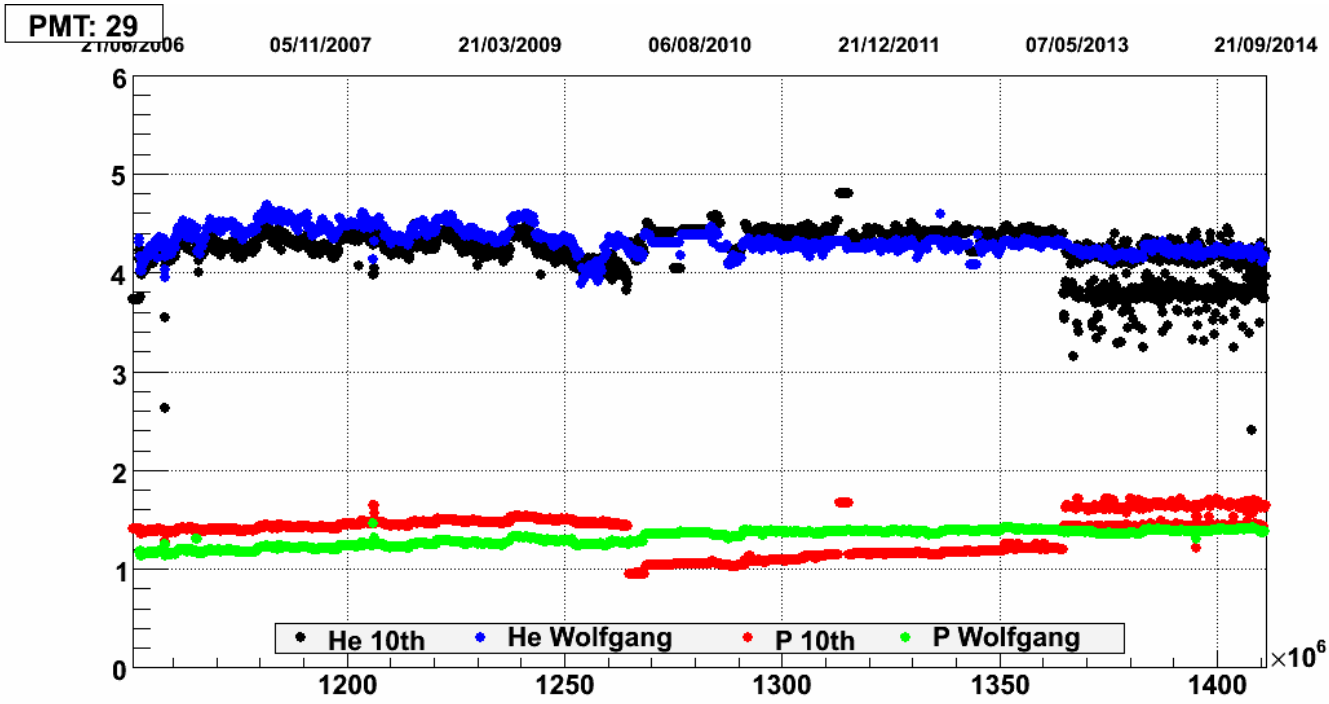


For this PMT in 2014 the results without the correction on the left and with correction on the right:

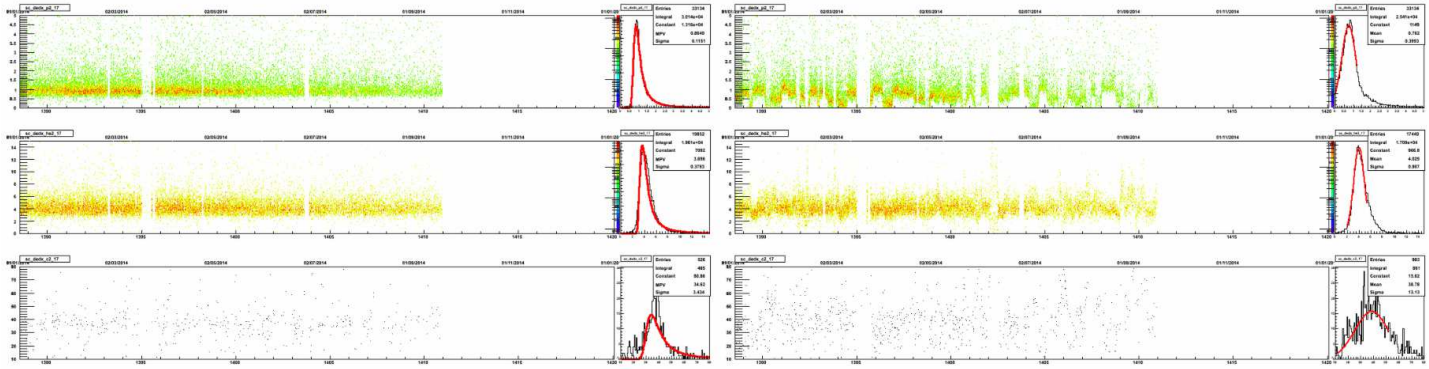


Also here the second order correction makes things worse especially for protons...

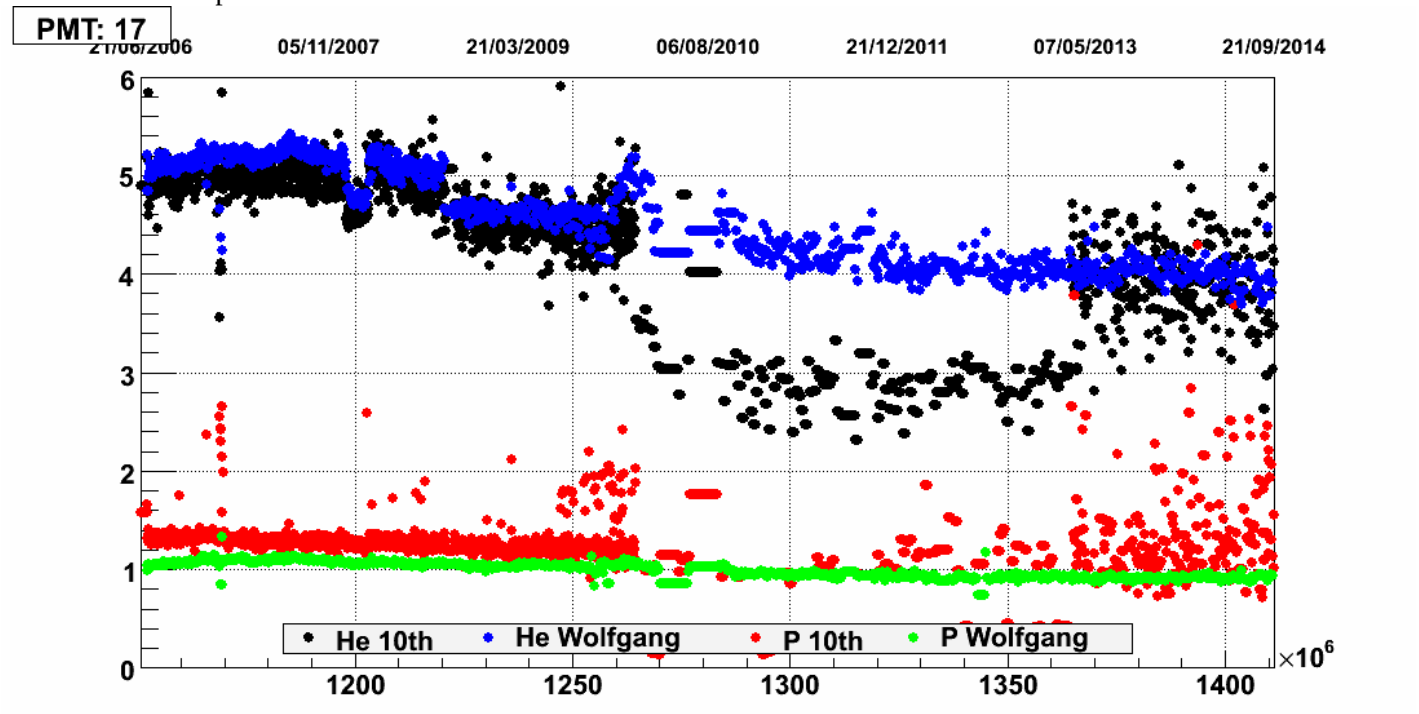
The correction for this PMT looks quite OK (no “steps” or double bands) in my analysis:



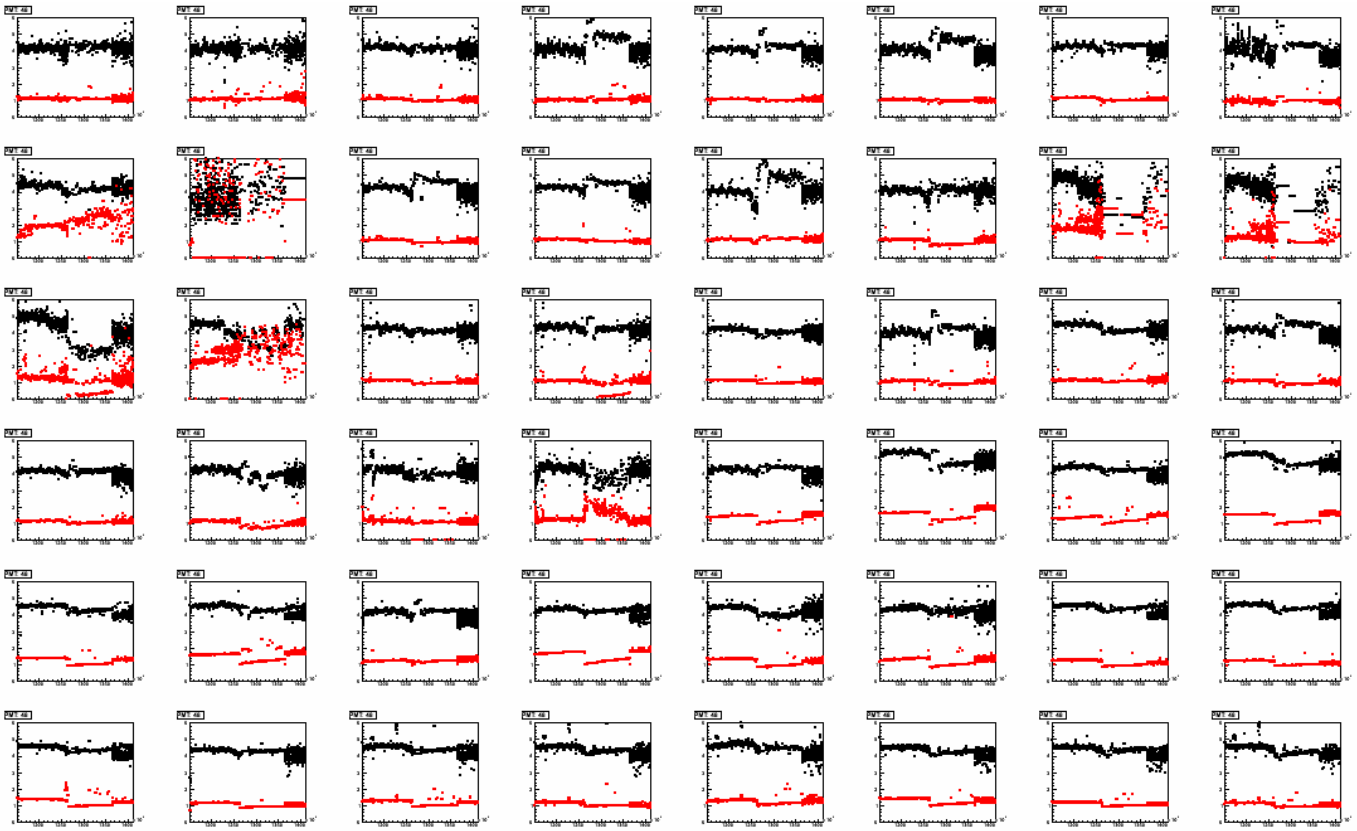
Practically all PMTs look –at least more or less- worse after the correction than before.. A last example: PMT #17 in 2014:



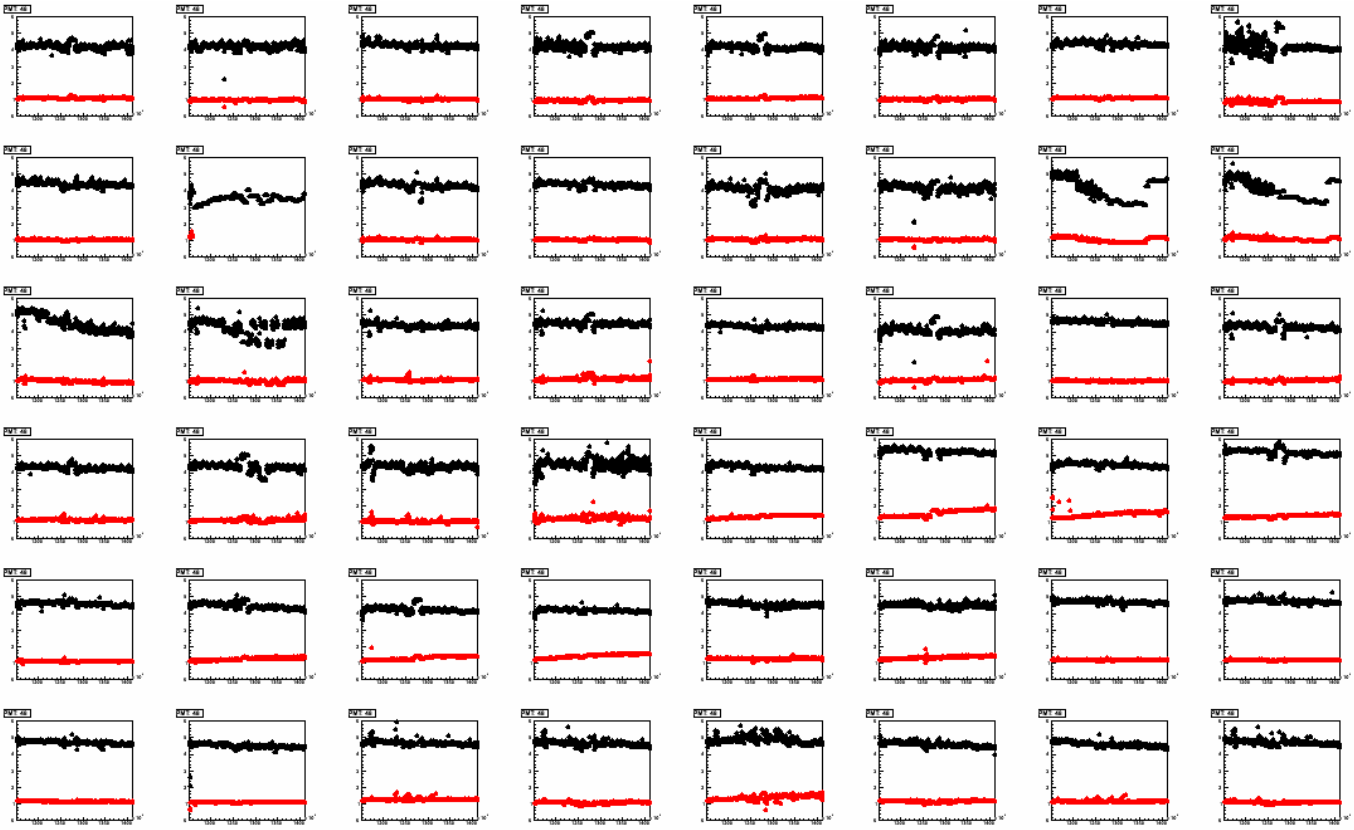
Also here the comparison of the second order corrrection:



As a summary the “biscale” corrections for the current 10th red all 48 PMTs in one plot:



My own corrections look like:



(I do some special treatment to PMTS 15 & 16 which have very low statistics after 2009, and also the PMT 10 is treated specially after its failure 2006)

So what I did after these findings:
I started to write a “patch” similar to the “ToFPatch” package, for the moment named “ToFdEdx_patch”. I used parts of the "ToFPatch" code and then inserted parts of the "ToFLevel2" and some lines of the "ToFCore" code. What is does:

1. Initialize:
It reads the attenuation files (the appropriate time intervals and names are taken from the database but now stored in a text file)
The other calibration files are read, they are valid for the whole flighttime as for the 10th reduction
A second order correction text file is read (1050 time intervals up to september 2014)

2. Process:
It checks
a) the time interval for the attenuation correction file
b) time interval limits for 2nd order correction
c) if we are in a HV-failure period (time limits hardcoded)

If we have a track, we calculate theta angle and derive the hitted paddles.
We loop over the PMTs and if a PMT is inside a hitted paddle, we proceed like explained above:
First attenuation correction, then “desaturation” function, then “f_desatBB” function.

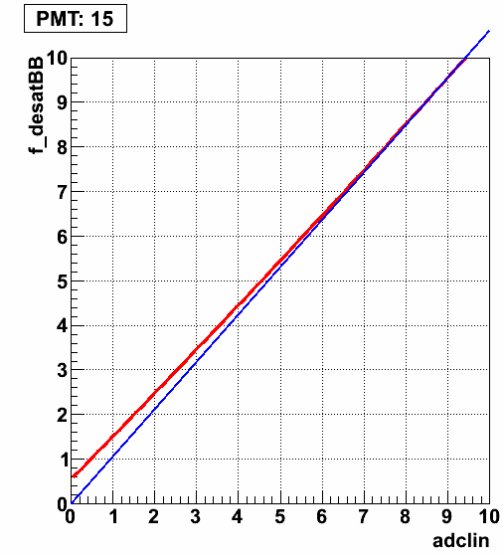
My changes:

- To derive the dEdx, beta is not neccessary. So the cut on beta is deleted. If we want to derive the charge, then we need beta, but only then.

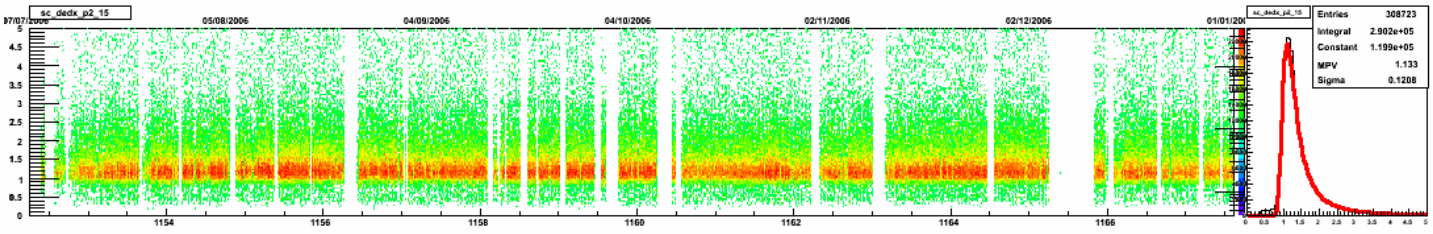
- The “f_desatBB” correction which causes a double band structure for some PMTs:

In my understanding this offset from zero for small dEdx is due to a bad fit and not physical: One mip should be one mip... Only a few PMTs have this strange fitcurve.
As a workaround I calculate the fit value at adclin=10 and then draw a line starting at zero to this point, see the blue line in the right figure (the red line is the original curve):

As said, this workaround is done only for a small number of PMTs.



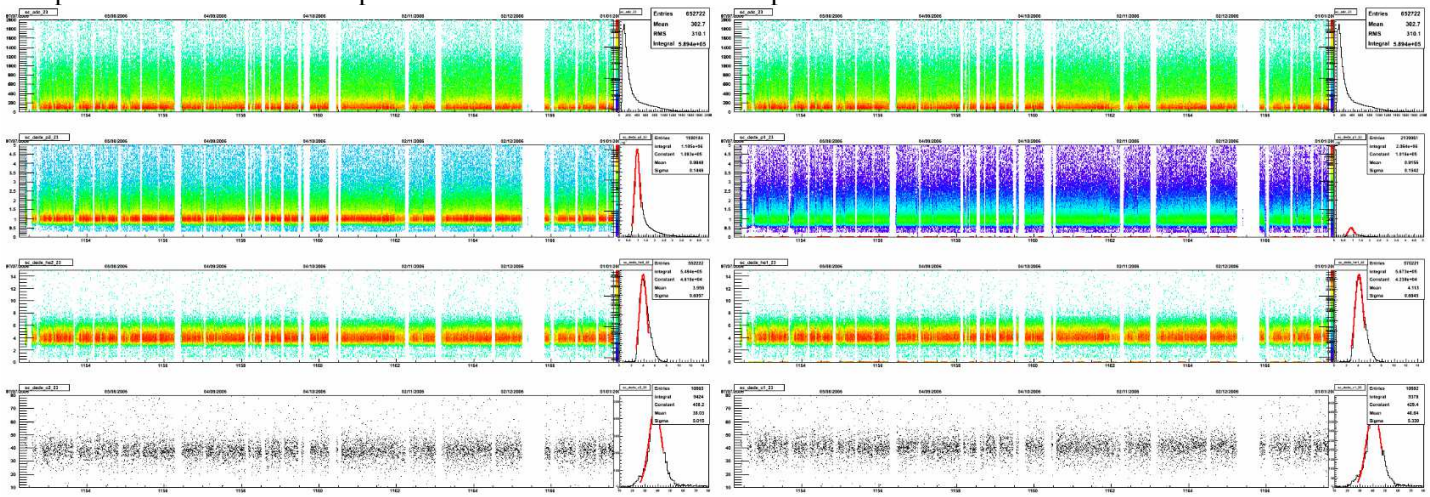
For example, the double band structure for protons in PMT #15 is gone now:



In the following figures I compare the output of my ToFdEdx_patch (with my 2nd-order correction) with the dEdx of the 10th reduction: On the left side there is the ToFdEdx_patch, on the right side the 10th reduction.
In the top row there are the raw ADC signals for helium, in the second row the dEdx for protons, then the dEdx for helium and the dEdx for carbon.

Some examples from 2006:

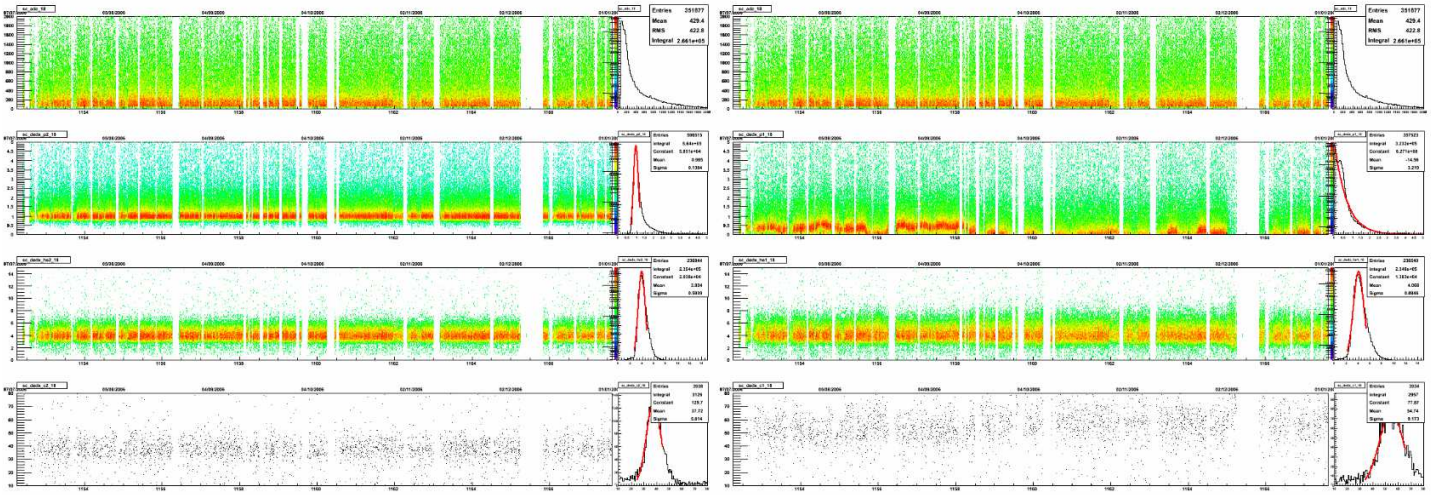
The problem of the zeroes for protons is now much better... example PMT #23



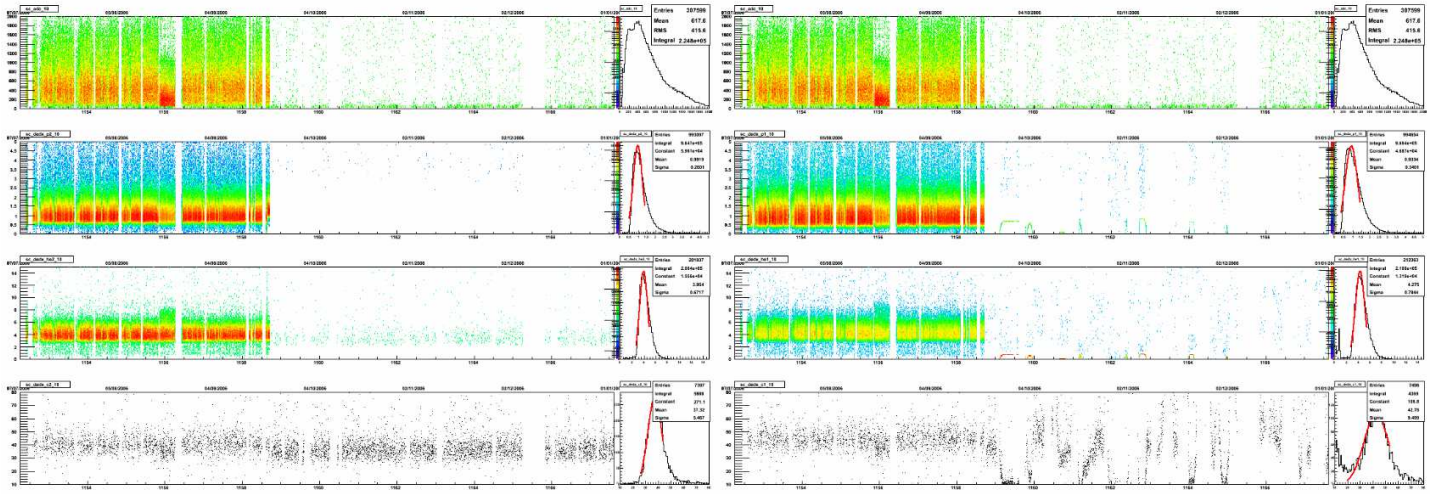
Comment: It is still not clear to me what is the reason for these large number of zeroes for protons in the 10th reduction.
In "ToFCore" there is some check if the paddle was hitted, so I would expect to see signals, strange...
The second order correction for this PMT in 2006 looks reasonable.

Since the results for my ToFdEdx_patch were quite nice, I did NOT spend more time try to run DarthVader and debug the original code...

Example PMT #18: Strange results for protons in 10th reduction, just fine with my patch:

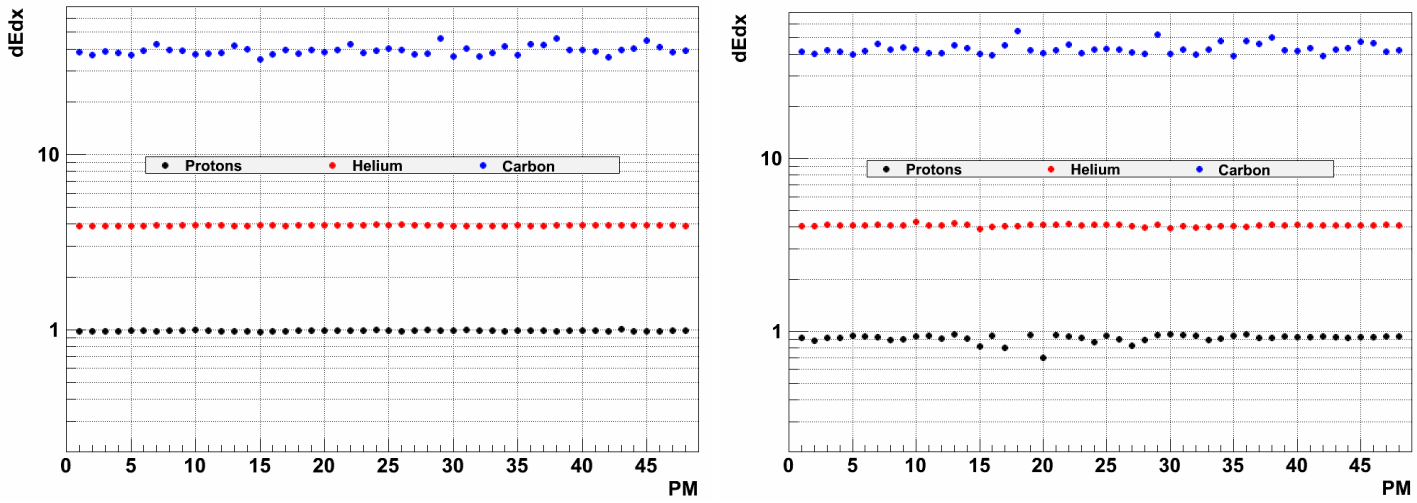


PMT #10 looked very strange in 10th reduction after the failure, with my patch it shows some low efficiency for helium and should be OK for higher charges:



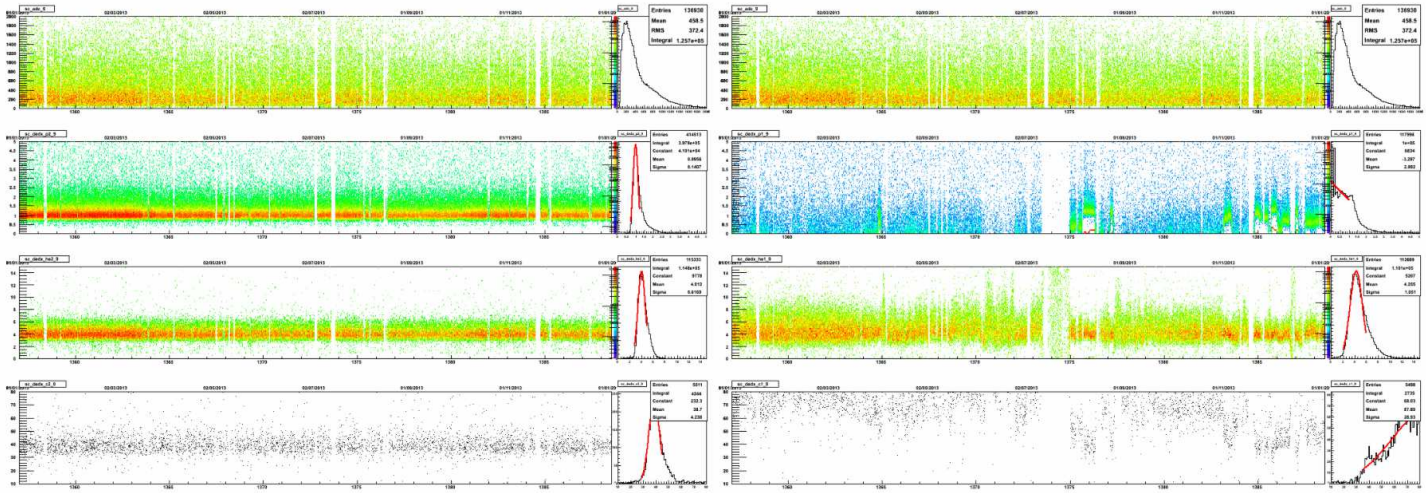
Due to the “biscale” second order correction the peaks for protons and helium should be centered exactly at 1 and 4 respectively. The next figure shows the mean values for protons, helium and carbon for each PMT (x-axis).

Left my patch 2006: Right: 10th reduction 2006

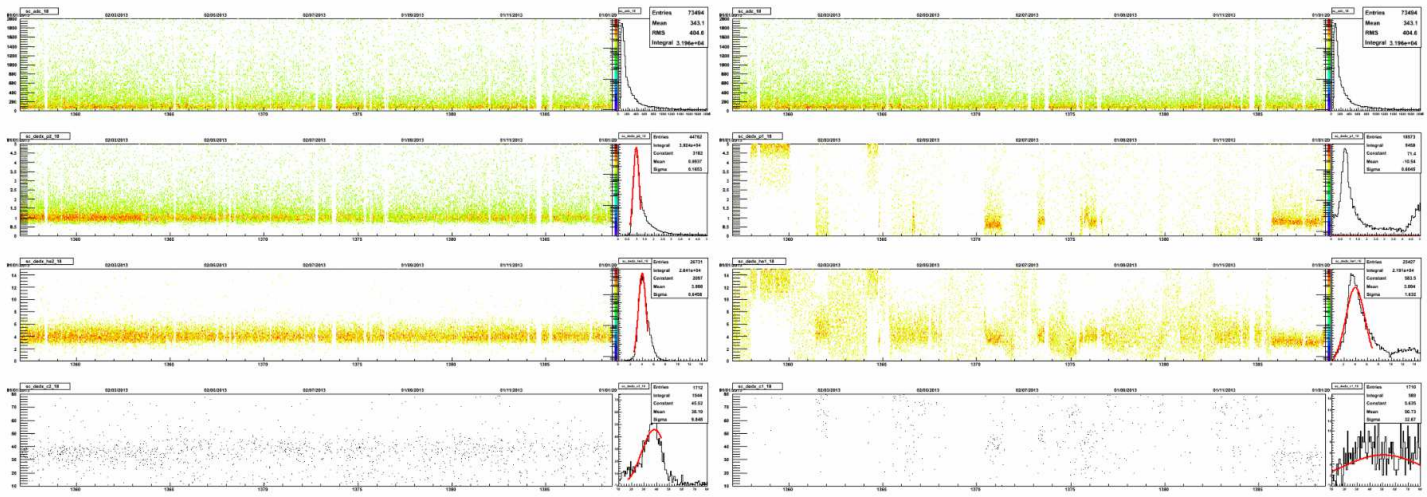


Here the results of the 10th reduction do not look so bad (except for PMTs like #18 for example). Later in the mission the problems got worse... You can also check pages 23 ff. from my talk
http://pamela.roma2.infn.it/technical-docs/9th_Pamela_coll_meeting/Wed30/Menn_ToF_dedx_10th_Problems.ppt

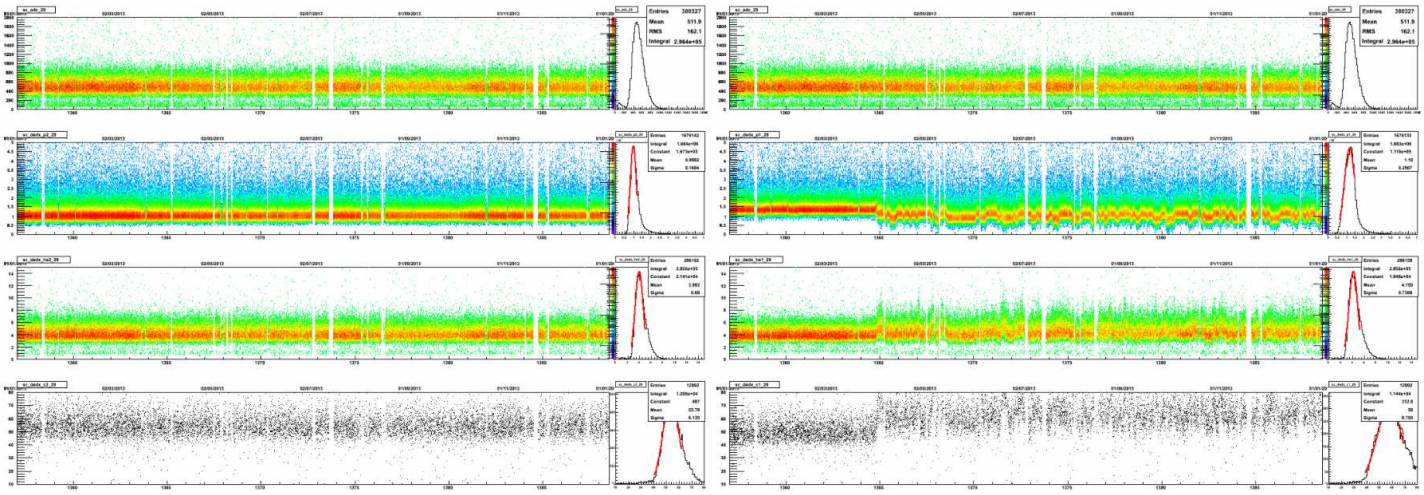
Example PMT #9 in 2013:



PMT #18 in 2013



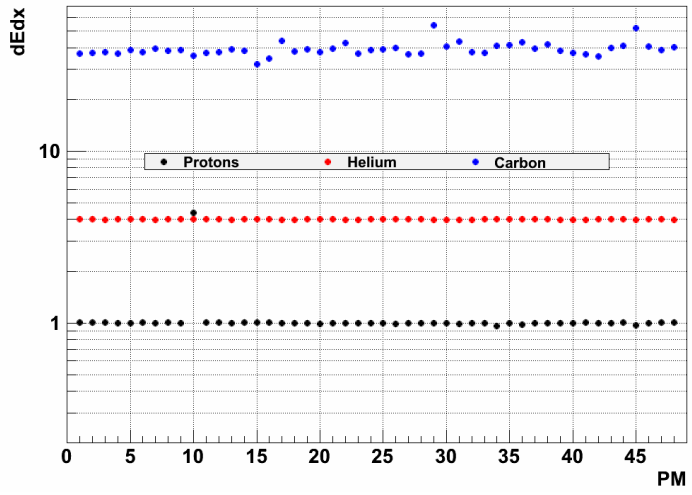
PMT #29 in 2013:



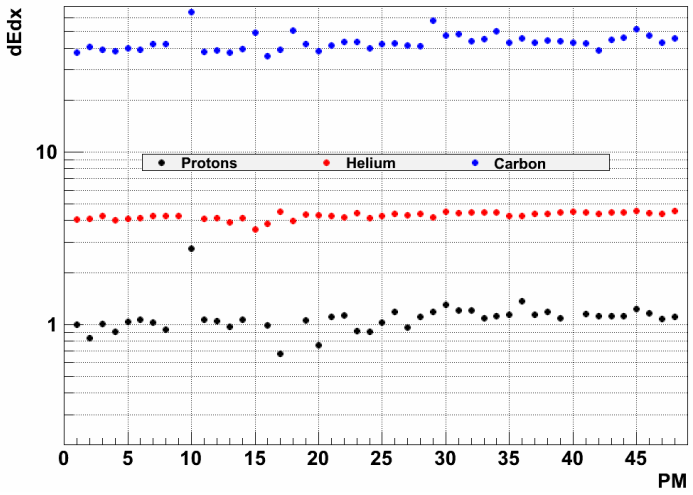
Compared to 10th reduction all PMTs show now a good behaviour. As explained before, in my understanding the most important reason for the strange behaviour in the 10th reduction is the second order correction.

Summary of the peaks of p, He, C for each PMT:

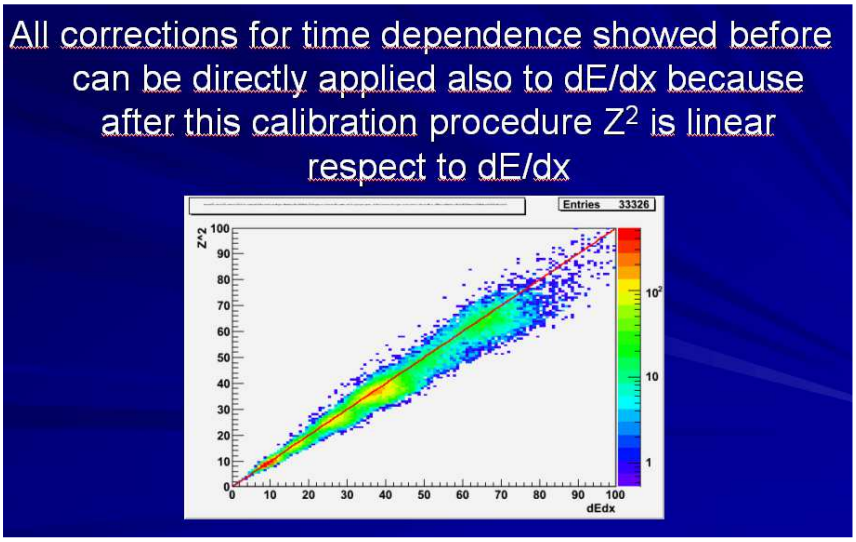
Left: my patch 2013:



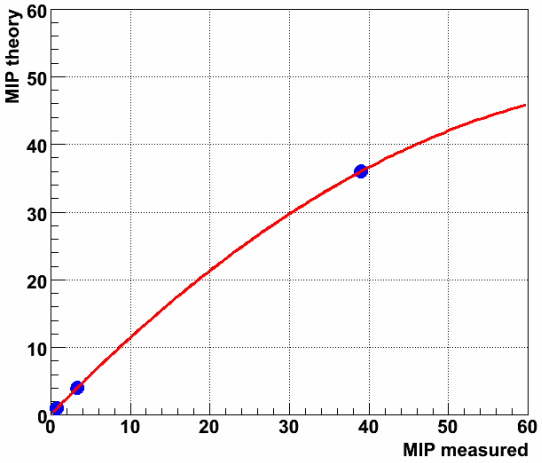
Right: 10th reduction 2013



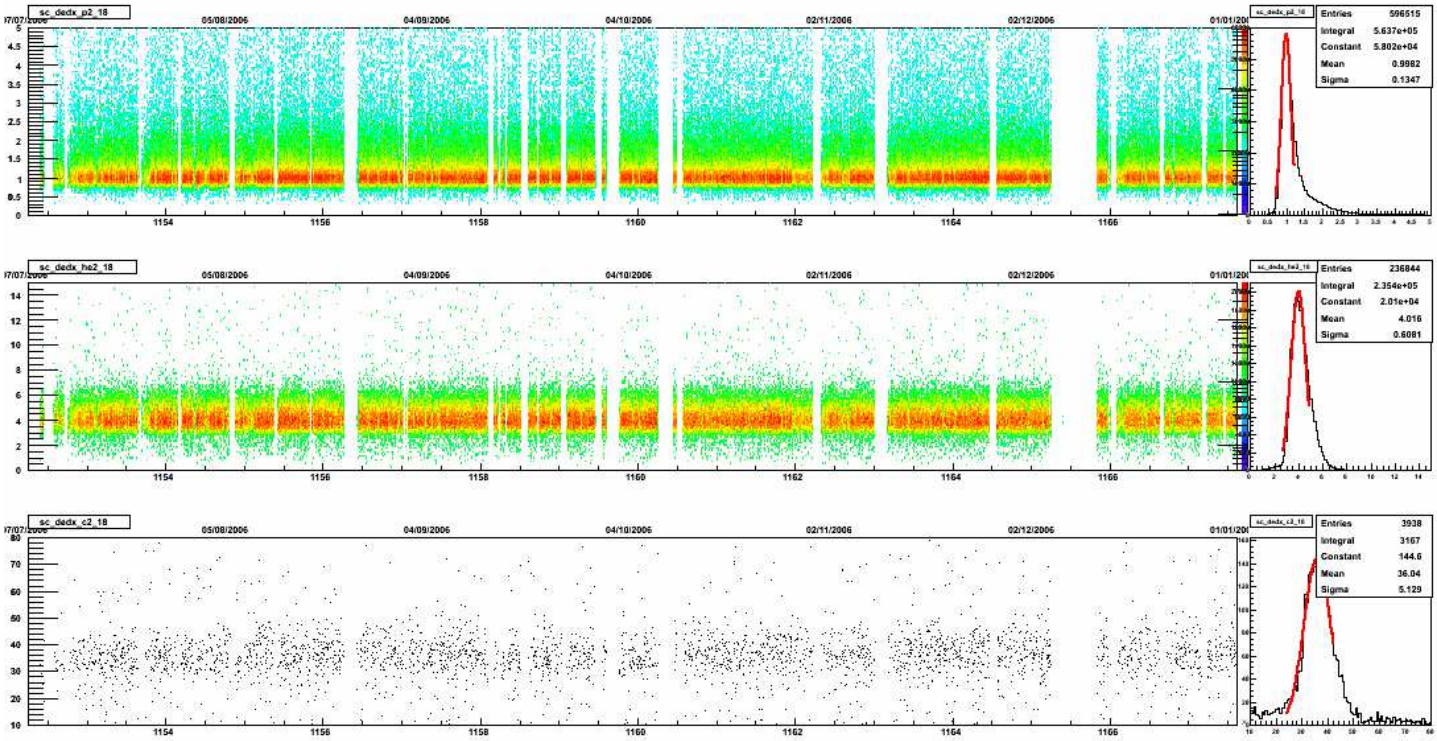
Now the improvement is clearly visible for practically all PMTs. As it can be seen on the left plot, due to the biscale method protons and helium peak exactly at 1 and 4 respectively, but the carbon peaks at 36 only if the detector behaves absolutely linear, which is the case for some PMTs, see Rita's Lindau talk:



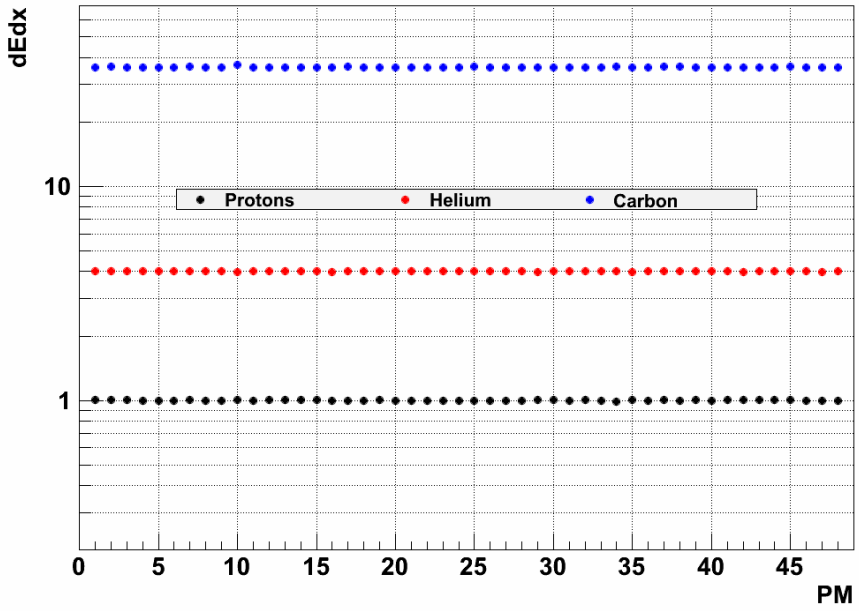
In my data the carbon (rigidity > 2 GV) peaks at ~40 for a lot of PMTs, so there seems to be some non-linearity. In principle one could use the values of the peaks of protons, helium and carbon to define some nonlinear function for the correction:
As a first solution I fitted the carbon distribution (right now for intervals of one year, could be done in smaller intervals) and saved all the values in a new "triscale" second order file.
In the patch routine I have now the peaks for protons, helium and carbon, and can calculate –for example- a "pol2" to fit to the theoretical values 1,4 and 36, as it can be seen on the right figure:



As an example this is the results for PMT #18 in 2006 using the "triscale" calibration (protons, helium, carbon)



So now the carbon is set to 36 as expected. The peaks for all PMTs in 2006 using the "triscale" method:



As said above, right now I have only one value for carbon for each year, one could do finer intervals, though at the end of the flight statistics for some PMTs will be very low.

Remark: I have NOT tested right now how this “pol2” correction works on the other charges beside Z=1,2, and 6. It is reasonable to assume that Z=3,4,5 are OK, but for example oxygen could have an offset. This would need to be tested.

How to run the software:
Insert in your usual analysis script:

```
#include </insert_your_path/ToFdEdx_patch/inc/ToFdEdx_patch.h>

const char* alg = "STD"; // or "EXT" "STD" "EXTF" "NUC" "NUCEXT" "NUCEXTF"

// TRI gives tri-scale calibration using H,He, and C  peaks // BI  gives bi-scale using H and He peaks
// see manual for details
const char* tri_or_bi = "TRI";
....

PamLevel2*  event  = new PamLevel2(dir,filelist,"+AUTO");       // << create pamela event

// Create and initialize ToFdEdx_patch
ToFdEdx_patch *tdedx = new ToFdEdx_patch();
tdedx->InitPar("/insert_your_path/ToFdEdx_patch/GoodParam/","simu");  // for simu data
//tdedx->InitPar("/insert_your_path/ToFdEdx_patch/GoodParam/","flight");  // for flight data

....

// in the event loop

 tdedx->Process(event,alg,tri_or_bi);
// the  dEdx for the 6 ToF layers:
 for (Int_t ii=0; ii<6; ii++) dedx_tof_pl_patch[ii] = tdedx->GetdEdx_layer(ii);
// each PMT
  for (Int_t ipmt=0; ipmt<48; ipmt++) dedx_tof_patch_pmt[ipmt] = tdedx->GetdEdx_pmt(ipmt);
```