Effect of electronics drift on tracker performance

Edward Overton

24 September 2015

1 Introduction

The purpose of this document is to give some indication of the effect of electronics drift in the tracker readout. From this an allowable variation can be set and used. The parameters used in this study are shown in table 1. Note that these studies were done for a low gain VLPC, which is most susceptible to drift effects. Most VLPC modules in the tracker have a significantly higher gain and will be much less effected.

VLPC Gain	6.0 adc counts
VLPC singles probability	3.5%
Pedestal Offset	30.0 adc counts
Pedestal Width	1.6 adc counts
Light Yield	11.23 photo-electron
Cluster photo-electron cut	2.0

Table 1: Parameters used in model.

The noise rate, light yield and cluster finding probability, were studied against changes in the pedestal location and gain of the cassette. These results are presented in figure 1.

2 Noise Rates

The 2 p.e. noise rate for a single channel is shown in the upper two plots. Note that to convert this into a duplet, the following conversion is required:

$$P_D = \frac{2}{3}(P_h N_C)^2 \tag{1}$$

where P_D is the duplet probability, P_h is the 2 p.e. noise probability and N_C is the number of channels in a tracker plane.

If the pedestal drifts by a single ADC count then the noise single channel noise probability will rise to 0.1%, or 3.0% probability of forming a duplet in a station.

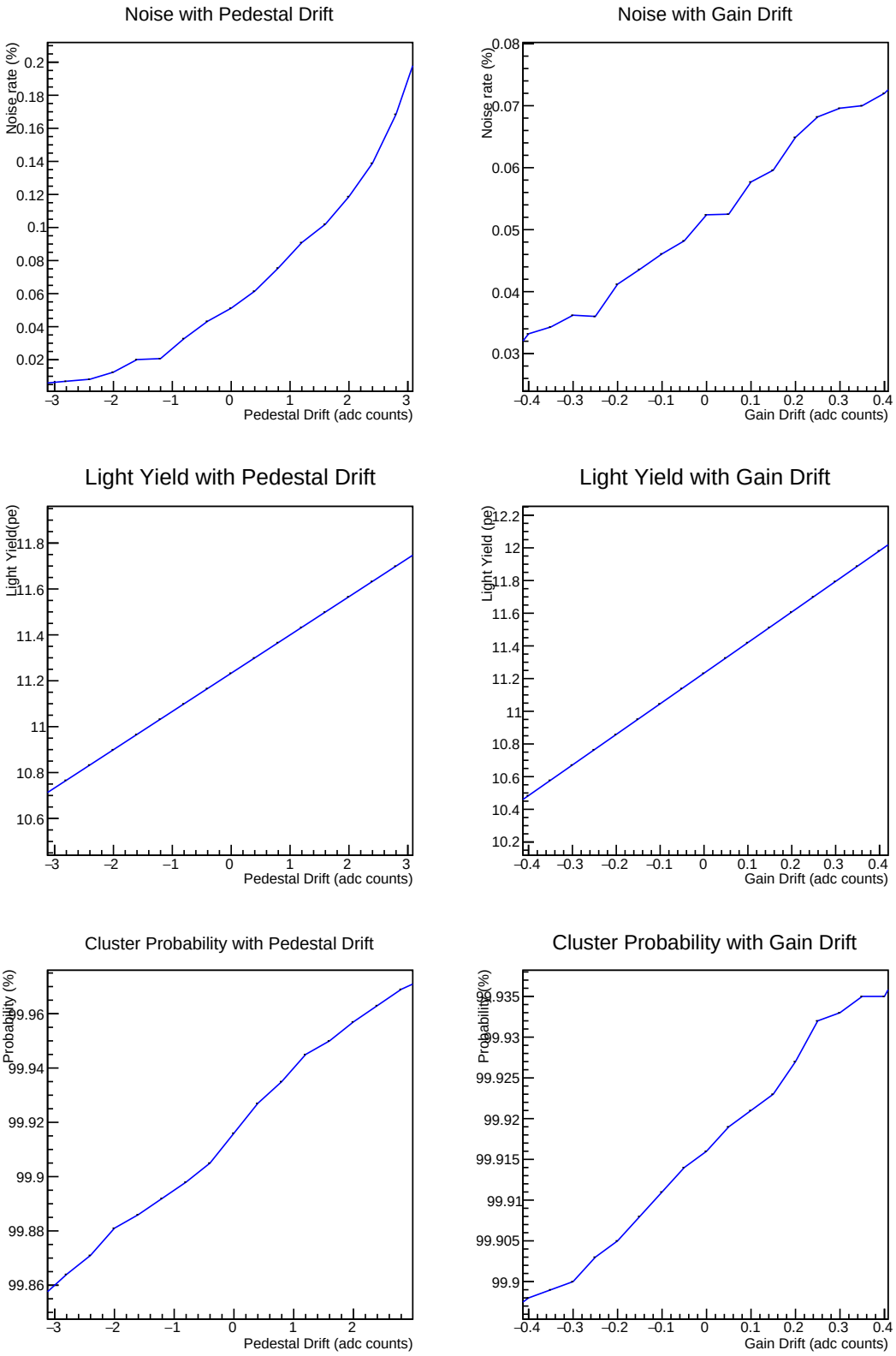

Figure 1: Effect of electronics drift on a number of tracker parameters.

The effect of a change in gain, of up to 0.4 counts is notably less than the pedestal drift.

3 Light Yield

The measured light yield from the detector is probably the most sensitive to electronics drift, which is expected since both the pedestal and gain are used in the light yield calculation.

In the case of a 1 count pedestal drift the light yield is changed by approximately 0.2 p.e. A change in gain of 0.4 counts causes a change in perceived light yield of 0.6 p.e., which is very large.

Since the tracker is primarily designed for tracking, it is possible these drifts are tolerable. However if the light yield is to be studied then less than 0.05 counts of pedestal drift and less than 0.01 count in gain drift are required to achieve ≈ 0.05 p.e. accuracy.

4 Cluster Finding

Arguably the most important metric of the tracker is the efficiency at which clusters can be found. In order to maintain a 99.9% probability of cluster finding then a pedestal drift of less than 0.5 counts is required. Again the gain change of less than 0.3 counts is sufficient.

Note that at 99.8% probability of finding a 2 p.e. cluster in a plane relates to a 99.4% triplet finding probability and a 0.6% probability of finding a duplet. The probability of finding neither of these is 0.001%.

The probabilities for finding duplet/triplets are given below:

$$P_{\text{Triplet}} = P_c^3 \tag{2}$$

$$P_{\text{Duplet}} = 3P_c^2(1 - P_c) \tag{3}$$

where P_c is the probability of forming a cluster, P_{Triplet} is the triplet forming probability and P_{Duplet} is the duplet forming probability.

5 Tolerance

From the previous sections, table 2 has been prepared to indicate tolerances.

Parameter	Cluster Tolerance	Light Yield Tolerance
Gain Drift	0.3	0.01
Pedestal Drift	0.5	0.05

Table 2: Tolerances for cluster finding and light yield estimation, given in adc counts.

6 Soaktest Stability

A two week soak test of the detector was conducted, for which the detector was read out continuously. A single channel was fitted for pedestal and gain and is displayed in figure 2. For this study channel 100 was chosen, since the gain of this cassette ensures that the pedestal and first photo-electron fitting procedure is reliable.

The study indicates that the pedestal can vary by a substantial amount (0.8 counts), however the gain is very robust. A flat straight line was fitted through the gain, which had an agreeable chi-square (188/250).

From this data, it is sensible to assume that measured gain of the devices is very stable, however the pedestal may move by a substantial amount. This is possible to account for with regular calibrations, which may be derived solely from the pedestal peak.

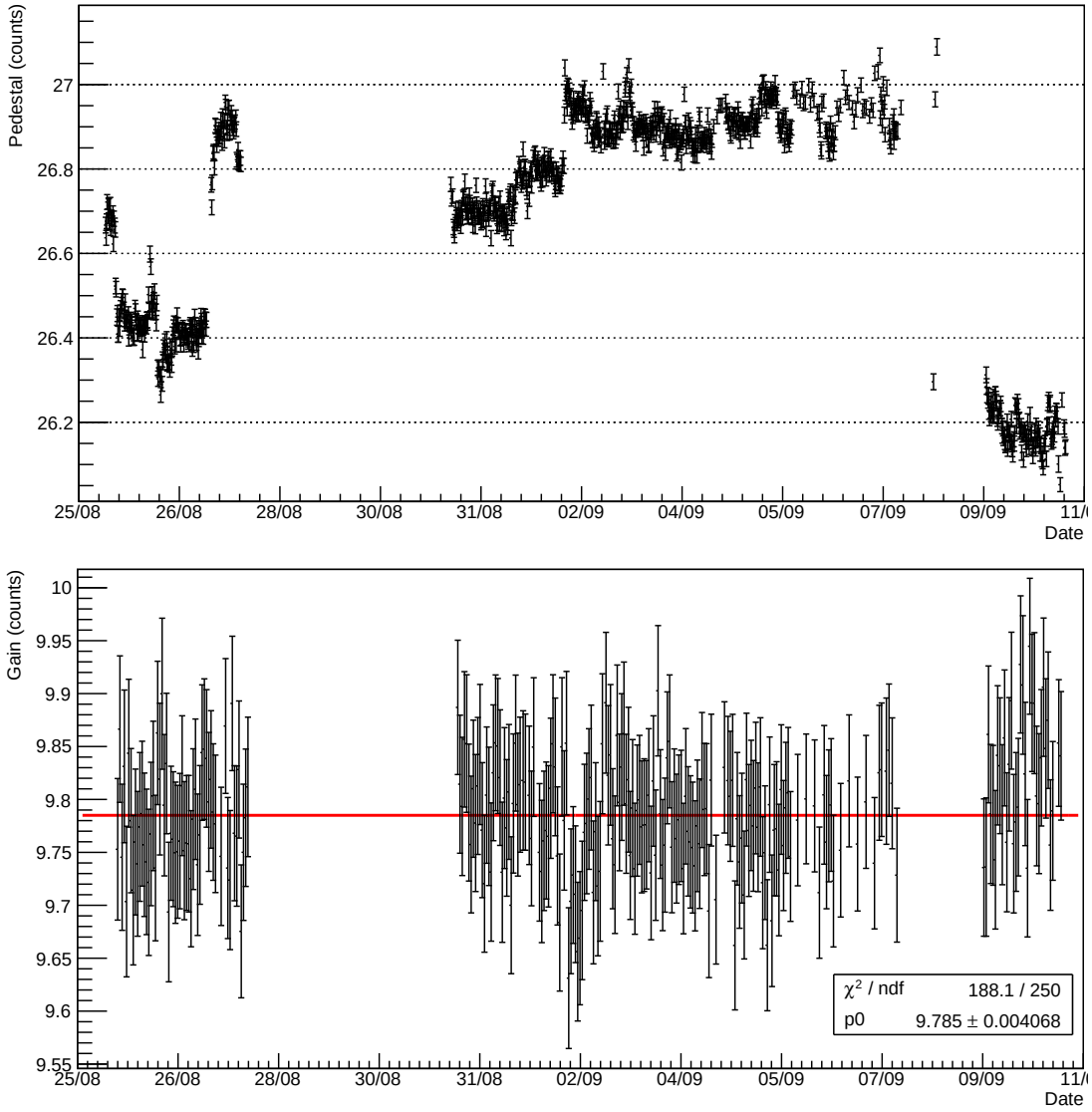

Figure 2: Stability of the pedestal and gain using data collected during the tracker soak test of the front end electronics.

7 Script

```
#!/bin/env python

import ROOT
import math
import numpy

"""
Script to model the effect of electronics drift
on the detectors:
"""

# Compute the noise rate for a given module:
def DriftModelNoise(pedestal=30, rms=1.70, gain=6.0,\
                    singlesprob=0.035, gain_drift=0.0,\
                    pedestal_drift=0.0, pe_cut = 2.0):

    entries = 1000000

    # Fill a Histogram:
    pedestal_fraction = 1.0 - sum ([math.pow(singlesprob,i)
       for i in[1,2,3,4]])
    ratios = [pedestal_fraction] + [math.pow(singlesprob,i)
       for i in[1,2,3,4]]

    hist = ROOT.TH1D("test","test",256, -0.5, 255.5)
    for i in range(len(ratios)):
        fill_func = ROOT.TF1("fillfunc","gaus", 3)
        fill_func.SetParameters(1,pedestal+pedestal_drift+i*(
           gain+gain_drift), rms)
        hist.FillRandom("fillfunc", int(ratios[i]*entries))

    thresh_bin = hist.FindBin(pedestal+(pe_cut*gain))
    below = hist.Integral(0,thresh_bin)
    above = hist.Integral(thresh_bin+1, 256)

    return above/(below+above)

# Compute the effect on light yield from drift.
def DriftModelLY(pedestal=30, rms=1.70, gain=6.0, gain_drift
   =0.0,\
                pedestal_drift=0.0, light_yield=11.23,
```

```
                    pe_cut=2.0):

    entries = 100000

    # Generate hits:
    hist = ROOT.TH1D("test","test",256, -0.5, 255.5)
    rand = ROOT.TRandom3()

    for e in range(entries):
        fill_value = rand.Poisson(light_yield)*(gain+
           gain_drift)+\
                     rand.Gaus(pedestal + pedestal_drift, rms
                        )
        hist.Fill(fill_value)

    # Get mean and convert into a light yield:
    ly = (hist.GetMean()-pedestal)/gain

    # Compute probability of counting clusters.
    thresh_bin = hist.FindBin(pedestal+(pe_cut*gain))
    below = hist.Integral(0,thresh_bin)
    above = hist.Integral(thresh_bin+1, 256)

    return ly, above/(below+above)


if __name__== "__main__":

    # Uncomment below to compare to a reference.
    ref_noise = 0 #DriftModelNoise()
    ref_ly, ref_prob = 0,0 #DriftModelLY()

    tg_n_pd = ROOT.TGraph()
    tg_l_pd = ROOT.TGraph()
    tg_p_pd = ROOT.TGraph()
    tg_n_gd = ROOT.TGraph()
    tg_l_gd = ROOT.TGraph()
    tg_p_gd = ROOT.TGraph()

    for pd in numpy.arange(-4.0, 4.0, +0.4):
        tg_n_pd.SetPoint(tg_n_pd.GetN(), pd,\
                         100.*(DriftModelNoise(pedestal_drift
                            =pd) - ref_noise))
```

```
    ly, prob = DriftModelLY(pedestal_drift=pd)
    tg_l_pd.SetPoint(tg_l_pd.GetN(), pd, ly - ref_ly)
    tg_p_pd.SetPoint(tg_p_pd.GetN(), pd, 100.*(prob -
       ref_prob))

for gd in numpy.arange(-0.5, 0.5, +0.05):
    tg_n_gd.SetPoint(tg_n_gd.GetN(), gd,\
                     100.*(DriftModelNoise(gain_drift=gd)
                         - ref_noise))
    ly, prob = DriftModelLY(gain_drift=gd)
    tg_l_gd.SetPoint(tg_l_gd.GetN(), gd, ly - ref_ly)
    tg_p_gd.SetPoint(tg_p_gd.GetN(), gd, 100.*(prob -
       ref_prob))

c = ROOT.TCanvas("drift","drift", 600,900)
c.Divide(2,3)

# Left hand plots:
c.cd(1)
tg_n_pd.Draw("alp")
tg_n_pd.SetTitle("Noise with Pedestal Drift; Pedestal
   Drift (adc counts); Noise rate (%)")
tg_n_pd.GetYaxis().SetTitleOffset(1.4)
tg_n_pd.SetLineColor(ROOT.kBlue)
c.cd(3)
tg_l_pd.Draw("alp")
tg_l_pd.SetTitle("Light Yield with Pedestal Drift;
   Pedestal Drift (adc counts); Light Yield(pe)")
tg_l_pd.GetYaxis().SetTitleOffset(1.4)
tg_l_pd.SetLineColor(ROOT.kBlue)
c.cd(5)
tg_p_pd.Draw("alp")
tg_p_pd.SetTitle("Cluster Probability with Pedestal Drift
   ; Pedestal Drift (adc counts); Probability (%)")
tg_p_pd.GetYaxis().SetTitleOffset(1.5)
tg_p_pd.SetLineColor(ROOT.kBlue)

c.cd(2)
tg_n_gd.Draw("alp")
tg_n_gd.SetTitle("Noise with Gain Drift; Gain Drift (adc
   counts); Noise rate (%) ")
tg_n_gd.GetYaxis().SetTitleOffset(1.4)
tg_n_gd.SetLineColor(ROOT.kBlue)
```

```
c.cd(4)
tg_l_gd.Draw("alp")
tg_l_gd.SetTitle("Light Yield with Gain Drift; Gain Drift
    (adc counts); Light Yield (pe)")
tg_l_gd.GetYaxis().SetTitleOffset(1.4)
tg_l_gd.SetLineColor(ROOT.kBlue)
c.cd(6)
tg_p_gd.Draw("alp")
tg_p_gd.SetTitle("Cluster Probability with Gain Drift;
   Gain Drift (adc counts); Probability (%)")
tg_p_gd.GetYaxis().SetTitleOffset(1.5)
tg_p_gd.SetLineColor(ROOT.kBlue)

raw_input("Done, Press enter to exit")
```