UNIVERSIDAD SAN FRANCISCO DE QUITO

COLEGIO DE CIENCIAS E INGENIERÍA

Optimizing Analysis Software for Single Wire Proportional Counters

David Alberto Hervas Aguilar

César Zambrano, PhD., Director de Tesis
Beatrice Mandelli, PhD., Supervisora en CERN

Tesis de grado presentada como requisito
para la obtención del título de Licenciado en Física

Quito, Abril 2015

Universidad San Francisco de Quito

Colegio de Ciencias e Ingeniería

HOJA DE APROBACIÓN DE TESIS

Optimizing Analysis Software for Single Wire Proportional Counters

David Hervas Aguilar

César Zambrano, PhD. ....
Decano de la Escuela de Ciencias
Colegio de Ciencias e Ingeniería
Director de la Tesis

Edgar Carrera, PhD. ....
Miembro del Comité de Tesis

Carlos Montúfar, PhD. ....
Miembro del Comité de Tesis

Alessandro Veltri, PhD. ....
Miembro del Comité de Tesis

Darío Niebieskikwiat, PhD. ....
Director Dep. Física

Quito, Mayo 2015

© DERECHOS DE AUTOR

Firma:

____

Nombre: David Alberto Hervas Aguilar

C.I.: 171217438-0

Fecha: 11 de mayo del 2015

Acknowledgements

First and foremost I would like to recognize the invaluable support from USFQ along all my studies and the connections this institution has helped me forge. This thesis would have not been possible without Edgar Carrera's continuos support to send annual groups to CERN and of course without CERN's magnificent Non-Member State Summer Student Programme. I would like to thank my supervisor Beatrice Mandelli along with Roberto Guida and Mar Capeans for the knowledge I gained, their advise, the final review of this manuscript and a great summer at CERN. Specifically, I would like to express my sincere thanks to Beatrice Mandelli for providing the data and original scripts that made this analysis possible and for her daily counsel and teachings. Finally, I am very grateful for Cézar Zambrano's guidance throughout the elaboration of this manuscript and for his valuable time as my thesis director and as chair of the science branch of the Polytechnic School at USFQ.

DEDICATION

I would like to dedicate my thesis to my Mom and Dad, for their unconditional support through my five years at USFQ, three career changes and two years at the University of Illinois.

For their love, their enthusiasm, their knowledge and their never ending care.

For the Sister who always made me laugh in times of need.

Resumen

Cuando radiación ionizante atraviesa las cámaras de gas en un detector de hilo único, los átomos de gas se separan en iones y electrones. Mediante un campo eléctrico fuerte y localizado cerca del hilo una avalancha de electrones se crea y puede ser capturada. La corriente que se produce en el hilo es proporcional a la energía original de la partícula detectada. No obstante, existen varios factores que puede contribuir al envejecimiento del detector. Estos se manifiestan en una pérdida de la ganancia causada por la deposición de contaminantes en el hilo. Este estudio consiste de técnicas de análisis de datos originales que se aplican para procesar grandes cantidades de datos producidos por dos detectores de hilo único corriendo simultáneamente. Varios factores de envejecimiento se analizan y se corrige los efectos causados por fluctuaciones ambientales. Una serie de scripts filtra datos, empareja datos y realiza correcciones y gráficos usando las extensas librerías de ROOT creadas en CERN.

Abstract

When ionizing radiation passes through gas chambers in single wire detectors gas atoms separate into ions and electrons. By applying a strong localized electric field near the single wire an avalanche of electrons is created and it can be collected. The current produced in the wire is then proportional to the energy of the particle detected. Nevertheless, many factors can contribute to detector aging effects which are visible in a loss of gain caused by deposition of contaminants on the collecting wire. This study consists on novel data analysis techniques used to process large amounts of data produced by two simultaneously running single wire detectors. Aging effects are analyzed while environmental fluctuations are corrected for. A series of scripts carry out data filtering, data matching, corrections, and finally trend plotting by using ROOT's extensive libraries developed at CERN.

Contents

1. Introduction

The work that follows consists of a the complex interaction of scripts written in order to analyze the aging effects of two simultaneously running single wire detectors. The bulk of the of the programing and analysis was conducted in the PH-DT-DI (Physics - Detector Technologies - Detector Infrastructure) department at The European Organization for Nuclear Research (CERN) in Switzerland and at Universidad San Francisco de Quito in Ecuador, to process a year of data collected by Dr. Beatrice Mandelli of the PH-DT-DI department at CERN. For this purpose ROOT, "an object oriented framework for large scale data analysis" developed at CERN [1], was used. The programing was conducted by myself under Dr. Mandelli's supervision and was jumpstarted by some of her existing scripts. Concordantly, some useful segments of Dr. Mandelli's code where included in mine (for more details refer to References). Subsequently, my thesis director in Ecuador, Dr. César Zambrano, guided me through the polishing of my analysis and the writing of this work. This thorough analysis consists of three main parts which complement each other. After a thorough theoretical framework of particle detectors and specifically Gaseous Ionizing detectors working in the Proportional Counter Regime (of which a Single Wire detector is the most basic example) the theory, script methodology, results and analysis of the Spectra Generation, Environmental Effects and Integrated Charge Sections are presented.

The Spectra Generation consists of grouping data into histograms which plot the the number particle detection events at each energy bin (also knows as spectra). The script gathers the data from different files created during the measurement period and generates 4 histograms for each day of the year. Using carefully designed fits to maximize fit accuracy and convergence many parameters are extracted and plotted against time but only the Gain, Resolution and Fit Means trends are analyzed. Here aging effects start to become clear.

The following section consists of the Environmental Corrections of the trends that were generated from the spectra. Specifically, pressure and temperature fluctuations can have a considerable effect on the detector due to the sensibility of its gas to these parameters. The script written for this Section reduces the dependence on these parameters and then supplementary scripts carry out a thorough analysis of the aging process while completely isolating temperature and pressure dependencies.

Finally, the concept of Integrated charge is introduced to briefly analyze the time trends produced in the Spectra Generation cection from a different angle. Instead of plotting the gain against time it is plotted against charge "accumulated" in the detector. Detecting particles generate small amounts of current that pass through the detector; therefore by integrating this current in time a total or "accumulated" charge in the detector is calculated.

2. Theoretical Framework

2.1. Particle Detectors

Particle detectors are the window into the world of the subatomic. They can range from a simple pocket dosimeter to the enormous ATLAS detector with 25 meters in diameter and 46 meters in length [2]. Particle detectors fundamentally rely on the interactions of particles with matter, hence the variety of particle detectors is as broad as these interactions and the types of particles. Effectively they detect, track, time, identify, and/or measure the energy, amongst many other functions, of incoming particles therefore becoming the most important tool in the field of High Energy Physics. It is important to point out some of the subtleties of particle detectors.

- Detector Efficiency: The number of particles detected divided by the total number of particles hitting the detector is known as the intrinsic efficiency of the detector (the absolute efficiency is given by number of particles detected divided by the total number particles produced by the source) . Detectors (Refer to Figure 2.1) have a given dead time in which no particles can be detected after any detection event which limits the efficiency of the detector[3].

- Saturation: A detector may become saturated if the rate of incoming particles surpasses the rate of particle detection.

- Detector Resolution: The resolution of a detector is given by the ability to discern two particles of different energies as such. While a detector with low resolution might see these two particles as having the same energy, one with a higher resolution will be able to distinguish them. If two peaks are separated by a greater distance than their full width at half maximum (FWHM) the the peaks are said to be resolved. Equation 2.1 gives the relative resolution at energy E, where $\triangle E$ is the FWHM[3] .

$$Resolution = \frac{\triangle E}{E} \tag{2.1}$$

Depending on the functionality of the detector, particle detectors are typically characterized within the following groups.

- Calorimeters: “A calorimeter measures the energy a particle loses as it passes through. It is usually designed to stop entirely or “absorb” most of the particles coming from a collision, forcing them to deposit all of their energy within the detector.”[4]

- Tracking Detectors: “Tracking devices reveal the paths of electrically charged particles as they pass through and interact with suitable substances. Most tracking devices do not make particle tracks directly visible, but record tiny electrical signals that particles trigger as they move through the device.”[4]

- Triggers: "A trigger is a system that uses simple criteria to rapidly decide which events in a particle detector to keep when only a small fraction of the total can be recorded. Trigger systems are necessary due to real-world limitations in data storage capacity and rates."[5]

When the energy of a particle is measured a spectrum is formed. The spectrum is a simple plot of the number of particles detected per specific energy. Radioactive isotopes may have many decay modes and emit radiation at different energies. These specific energies are called spectral lines. However, this nomenclature is actually misleading. All the detected particles from one spectral line actually do not fall within one exact energy, but within a Gaussian distributed range. The "line" refers to the mean of the distribution which ideally corresponds to the peak. Furthermore, the FWHM of the distribution is the width of the "line" and is related to the resolution of the detector by Equation 2.1. Typically a detector will not measure the energy of a particle directly, but it must be calculated depending on the output of the detector. For example the detector used in this experiment outputs ADC (analog to digital) counts generated by the internal circuitry of the detector. These counts then can be converted to energy by calibrating the detector using radiation sources of known energy.

2.1.1. Types of Particle Detectors

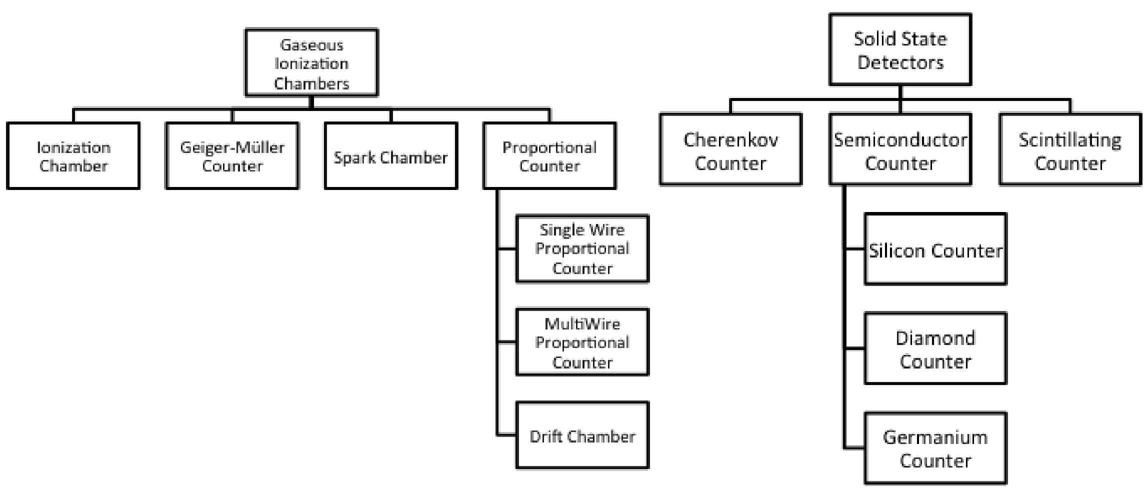

Figure 2.1: Hierarchy of particle detectors. Some gaseous detectors served as tracking systems (as well as calorimeters) in the past but were replaced by the significantly faster silicon detectors [6, 7].

2.1.2. Ionizing Gas Chambers

Gaseous ionizing detectors were the first electrical particle detectors and continue to be in widespread use today as radiation monitors. Their basic operation principle consists on the ionization of gas molecules and atoms. As ionizing radiation passes through this medium, ions and electrons are formed. Meanwhile, an electric field is applied to collect the generated electrons; thus a current is produced in the anode of the detector [3]. Gases are used as ionization media because of the greater mobility of ions and electrons through them. A mixture of mostly noble gases is used in the chamber and the specifications of these mixtures naturally affect the current measured. A basic layout of a cylindrical gaseous ionizing detector is shown in Figure 2.2.

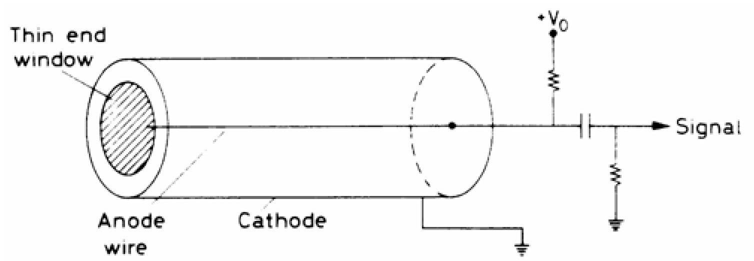

Figure 2.2: Cylindrical gaseous ionizing detector typical layout [3].

The simplicity of this design, the ease of operation, and low cost are the main reasons why these detectors are still in use today. The electric field produced by the wire on which a voltage V_0 is applied is given by:

$$E = \frac{1}{r}\frac{V_0}{ln\left(\frac{b}{a}\right)} \tag{2.2}$$

Where r is the distance from the wire, b the inside radius of the cylinder and a the radius of the central (anode) wire. This electric field is localized around the wire, which will become of importance in further discussions. As one can expect increasing the voltage applied to the detector will change the magnitude of the electric field and therefore have a major effect in the number of ions collected [3]. Many regimes can be identified as a function of increasing applied voltage. These are shown in Figure 2.3. This basic design coupled with all these working voltage regions encompass all the Gaseous Ionization Chamber detectors described in Figure 2.1.

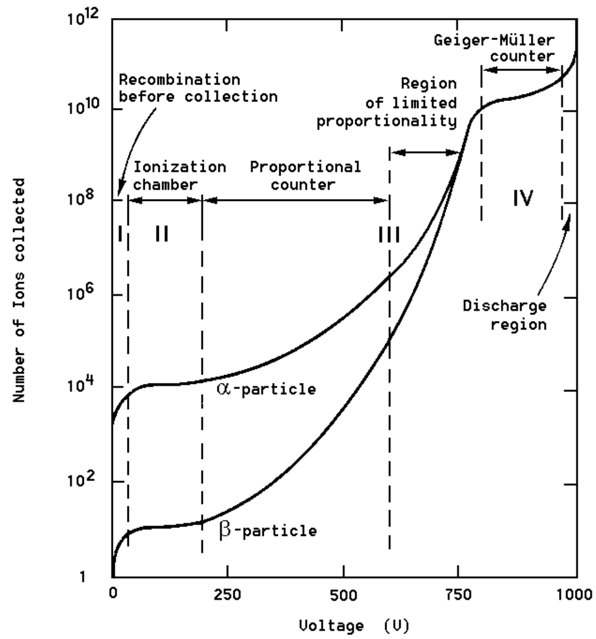

Figure 2.3: Different opereting regimes of ionization gas chambers [3].

If the voltage is too low the electron-ion pairs will recombine because of their own attraction forces. By passing this limit the ionization chamber regime is reached. At this working point ion-electron pairs are collected with no intermediate effects. For this reason the generated signal will be very low and is only used with strong radiation sources. Increasing the voltage beyond this limit results in the proportional counter regime. This is the area of interest since this study centers data analysis of single wire gaseous ionizing detectors working in the proportional counter regime. At this voltage range the traveling electrons have enough energy to hit other gaseous particles in their path and cause further ionizations. This effect is referred to as an ionization avalanche. The total amount of ion-electron pairs produced is still proportional to the original ionization events caused by radiation; therefore, the current measured is proportional to the energy of the detected particle. The final voltage regions are the Geiger-Muller regime and then discharge region which corresponds to the working voltage of the Spark Chamber [3].

2.2. Single Wire Detectors

Single wire detectors are the simplest type of proportional counter. Having a design identical to that portrayed in Figure 2.2 they operate under the principles described below.

2.2.1. Ionization of Gases

Gas molecules and atoms can be readily ionized by ionizing radiation. There are many mechanisms through which this can occur. When a charged particle interacts with matter it can lose energy in one of two reactions: excitation or ionization. Excitation occurs when a charged particle transfers its energy to an atom as in Equation 2.3.

$$X + p \rightarrow X^* + p \tag{2.3}$$

Where p is the charged particle and X is the atom to be excited [3]. Although, no electrons are generated the exited atom may ionize other atoms in further reactions. Furthermore, direct ionizations may occur if the energy of the charged particle is high enough. Equation 2.4 demonstrates the ionization process.

$$X + p \rightarrow X^+ + p + e- \tag{2.4}$$

This is known as a primary ionization. If the energy transferred to the emitted electron is high enough, it can participate in the ionization of further atoms, that is creating secondary ionizations. As noted for the excitation reaction, excited atoms may cause further ionizations too. The Penning Effect is an example of such a phenomenon. Certain atoms are not able to de-excite immediately (through emmision of a photon) and can collide against other atoms thus starting an ionization reaction. A classical example is that of the interaction between different noble gasses. Finally, the positive atoms formed in an ionization reaction can combine with neutral atoms of the same type and form a molecular ion releasing an electron in the process [3].

2.2.2. Fill Gas Choice

For the detector to work the electron ion pairs must remain intact till they are collected. Recombination and electron attachment come into play in this scenario. As explained in Section 2.1.2 while referencing Figure 2.3 if the working voltage of the detector is too low, recombination of the electron and ion will occur and a photon will be emitted.

$$X^+ + e^- \rightarrow X + h\nu \tag{2.5}$$

Electron attachment is a similar process where the freed electron is captured by an atom with a high electron affinity [3].

$$X + e^- \rightarrow X^- + h\nu \tag{2.6}$$

It is evident that gasses with low electron affinities much be used, such as the noble gasses Ar, He and Ne which have negative electron affinities. Additionally it is important to monitor the levels of gasses with high electron affinities. Molecular oxygen in the air is a particular contaminant that must be monitored at all time. Since O_2 has a high electron affinity it can disrupt the operation of the single wire detector significantly[3].

The electric field applied in the detector forces ions and electrons to accelerate in opposite directions. Due to collisions with other atoms this acceleration is capped and a maximum average velocity is achieved. The drift velocity is defined as the average speed that is attained in this process. Also note that particles have their own random thermal velocities given by

$$v = \sqrt{\frac{8k_BT}{\pi m}} \tag{2.7}$$

Where k_B is the Boltzman constant, T the temperature and m the mass of the particle. These velocities are much higher than the corresponding drift velocities of the electrons and ions. On the other hand the drift velocity u depends on the mobility of the charge μ and the electric field strength E [3].

$$u = \mu E \tag{2.8}$$

In turn the mobility is related to the diffusion constant D by the Einstein relation in ideal gasses:

$$\frac{D}{\mu} = \frac{k_BT}{e} \tag{2.9}$$

Where e is the charge of the electron. And D is given by the expression in Equation 2.10.

$$D = \frac{2}{3\sqrt{\pi}}\frac{1}{p\sigma_0}\sqrt{\frac{(k_BT)^3}{m}} \tag{2.10}$$

Where p is the pressure and σ_0 is the total cross Section for collision with a gas molecule [3]. Finally by joining Equations 2.8 - 2.10 a final expression for drift velocity is achieved.

$$u = \frac{2}{3\sqrt{\pi}}\frac{e}{p\sigma_0}\sqrt{\frac{k_BT}{m}}E \tag{2.11}$$

The drift velocities for electrons will be much higher than for positive ions since they are lighter. Also note the environmental dependancies (temperature and pressure) of Equation 2.11. Higher drift velocities are desired to avoid electron attachment and recombination. The addition of certain polyatomic gasses such as CO_2, CH_4, or CF_4 leads to larger electron drift velocities [8] and are therefore readily used in single wire detectors.

2.2.3. Electron Avalanche

As mentioned in Section 2.2.1 secondary ionizations may occur if the energy of the incoming particle is high enough. Hence if the primary ionization electrons gain enough energy from being accelerated by the electric field secondary ionizations are caused. If this effect continues an electron avalanche will occur. Since the electric field is highly localized along the central wire of the detector, ionization avalanches only occur within a few radii of this wire [3]. As explained in the previous Section since electrons have higher drift velocities than positive ions the electron avalanche will form the particular drop shape represented in Figure 2.2.3.

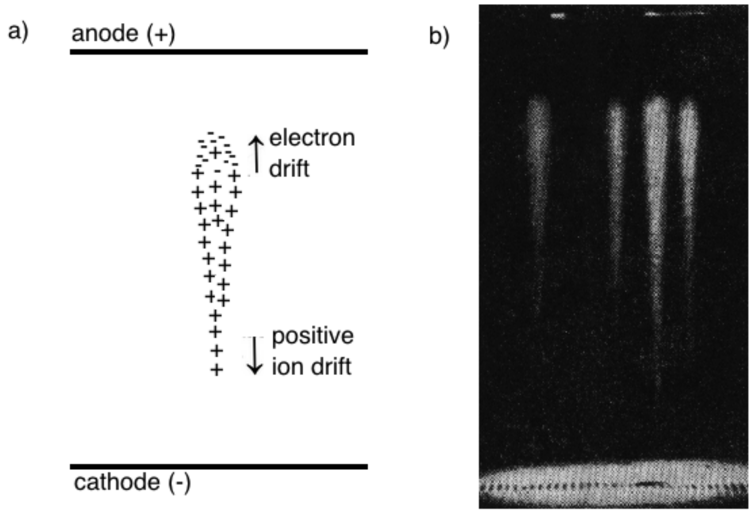

Figure 2.4: a) Symbolical representation of electron avalanche with liquid drop shape. b) Actual photograph of electron avalanche formation [9].

To further characterize the electron avalanche, α is defined as the mean free path of an electron for a secondary ionizing collision. Therefore, $1/\alpha$ is the probability of an ionization per unit path length, Hence if there are n electrons, there will be dn new electrons created in the path dx [3].

$$dn = n\alpha\,(x)\,dx \tag{2.12}$$

Since the electric field in Equation 2.2 is non-uniform α is a function of x. Integrating the expression in Equation 2.12 and defining n_0 as the number of primary ionization electrons then an avalanche multiplication factor M can be calculated.

$$M \equiv \frac{n}{n_0} = exp\left(\int_{r_1}^{r_2} \alpha\,(x)\,dx\right) \tag{2.13}$$

Where r_1 and r_2 are the initial and final points in the path. This multiplication factor is known as the gas gain or gain, a fundamental property of all proportional counters and specifically of the studied single wire detectors [3]. Recall that the current produced in the anode of the detector is directly proportional to the energy of the incoming particles. Therefore, an increase in gain will cause an increase in measured energy. Note that the energy of the particle has remained constant, therefore it is vital to calibrate a detector with respect to its given gain.

2.3. Single Wire Detector Aging

As with any particle detector, single wire gaseous detectors have an apparent aging process. The root of this aging occurs by the deterioration of the anode wire which causes the detector signal to worsen. Consequently, the measured energy spectrum is affected. It is by studying this effect that the aging effect can be understood. As a single wire detector ages the initially Gaussian peak formed by measuring the energy of incoming particles starts to spread out. This aging process eventually forms what appears to be a second peak in the energy spectrum as in Figure 2.5. Note that no new particles are impaling the detector at a different energy, just that detector itself is not measuring correctly the energy of the particles emitted by the source. This spreading out of the peak has three main effects that are of interest in this discussion.

1. Reduction of gain. The appearance of a second peak lowers the mean of the peak measured as a whole. The individual means of the peaks measured separately are also lowered. The loss of gain is reflected as a loss of measured energy (peak mean).

2. Change of overall peak width (when both peaks are measured as one). The FWHM varies; this change along with the drop in energy affects the resolution (refer to Equation 2.1) of the detector.

3. Fitting problems. This is a practical effect of the aging effect. In general a code for fitting this peak should be able to fit it at any point in the aging process. However the smooth transition between one peak and two leads to coding challenges.

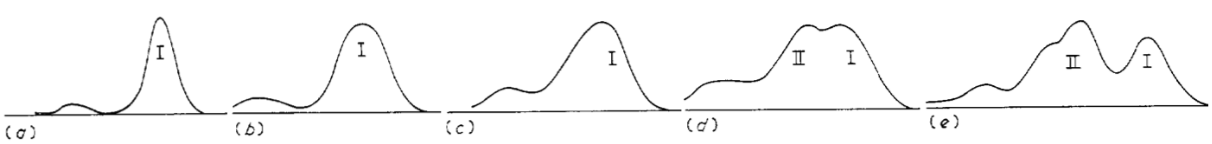

Figure 2.5: "Five successive pulse-height distributions as they develop during irradiation with a 5.9 KeV source of a proportional counter filled with argon + 10 % methane, demonstrating the ageing effect" [10].

In general there are three major causes that alter measurements made by this type of detector. First, variation of environmental parameters such as temperature and pressure cause an immediate effect on the measurements. This does not contribute to aging but rather to instantaneous measurements. The environmental fluctuations should be corrected for since they can be considerable. Pressure changes can change the density of medium inside the ionizing gas chamber in the detector effectively increasing the amount of particles that can be ionized. The same is true for temperature fluctuations. Second, the construction of the detector has a major impact on its performance. Leaks can alter the gas mixture sufficiently and therefore alter the current measured. Oxygen is a particular contaminant that is of interest so its levels are always monitored within the chamber. Finally, aging itself alters measurents. The deposition of contaminants on the central wire is the most important cause for aging. Contaminants can come in through leaks or be produced inside the detector itself by the outgassing of the materials used to build it.

2.3.1. Deposition on Anode

Many studies [10, 11, 12, 13] have concluded that aging effects in single wire detectors occur mainly because of deposits in the anode wire. This is discussed in detail in [10]. Two single wire detectors that allow a free exchange of gas between them were run simultaneously forming what is referred to as a twin counter. One of the anodes is irradiated continuously while the other is not. While applying the same voltage to both anodes it was discovered that the anode being irradiated demonstrated the aging effect while the anode that was not irradiated did not. Since they share the same gas, differences in temperature and pressure or the composition of the gas should affect both equally; hence, the difference must be in the anode itself. When the irradiated anode was annealed (heat treated) the original single gaussian shape was obtained. This eliminated the possibility of permanent damage to the anode such as that of sputtering (when atoms of a target material are ejected when bombarded by energetic particles [14]). The conclusion was that there must be deposition on the anode wire mostly composed of negative ions. Within this particular study it was found that deposition on an anode wire with a radius of 12.5 μm amounted to 1-3 μm resulting in a loss of gain by a factor of 2. Since the drift velocity is directly proportional to the electric field E as given by Equation 2.8 and in turn E is highly dependent on the anode wire's radius a (Equation 2.2) a slight change in a due to contaminant buildup with affect the gain. This was also demonstrated during a characterization of ATLAS's radiation trackers (also single wire detectors), where actual deposition is depicted.

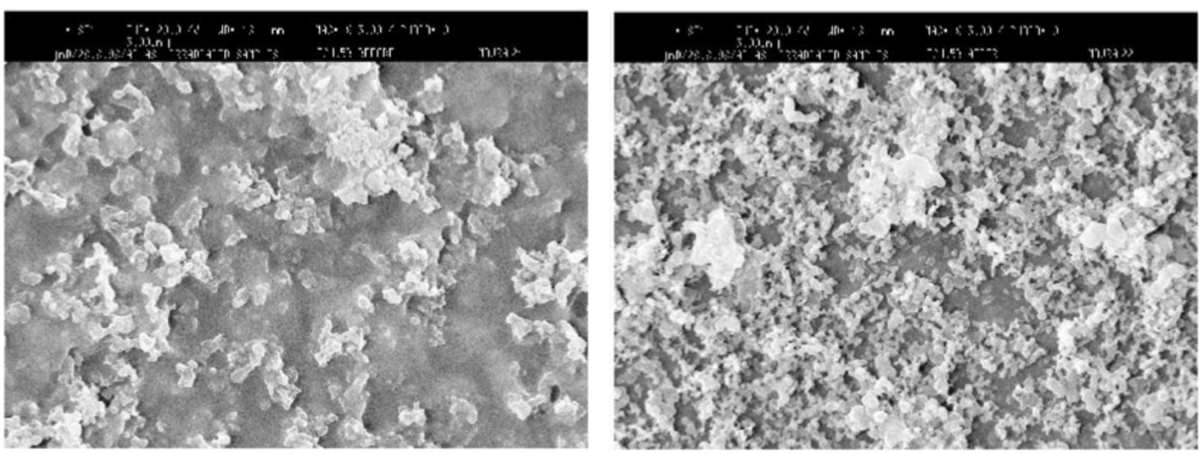

Figure 2.6: Micro-photograph of the cathode surface before irradiation 25x20μm (left). Micro-photograph of the cathode surface after irradiation 25x20μm (right) [11].

Furthermore, several other studies [15, 16] have characterized the aging of single wire detectors with respect to the materials that were used to build them. This has answered the question with respect to what materials are being deposited on the anode wire. For example in [15] a comparison between a detector built with a stainless steel box and one made with fiberglass is made.

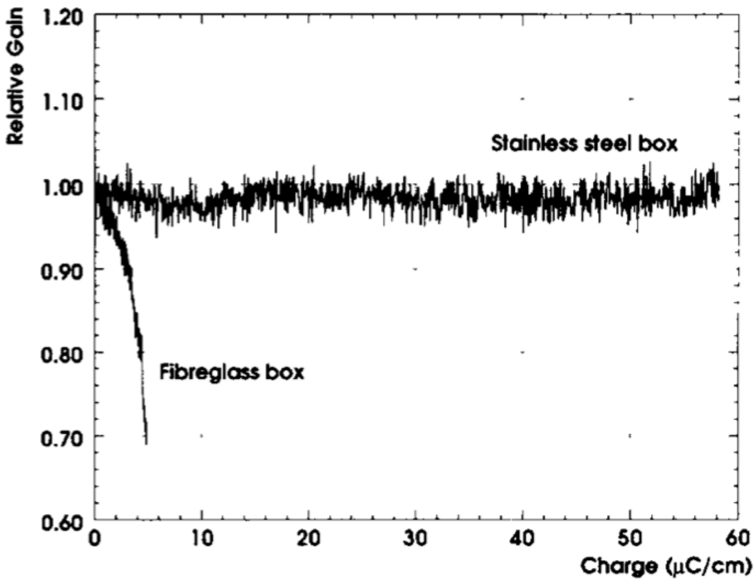

Figure 2.7: Gain dependence on charge measured for the same MSGC plate assembled in a clean, stainless steel box, and in a fibreglass box with rubber 0-rings and Araldit epoxy. [15]

In Figure 2.7 it is evident that the relative gain (instantaneous gain divided by initial gain) drops rapidly for the fiberglass box due to increased deposition on the anode caused by the materials that were used to build it. The x-axis in this Figure is integrated charge, a concept that will be analyzed in detail in Section 7. In short, it is the charge that has been accumulated in the detector by integrating the current over time. In [10] the deposits on the wire were mainly found to be made of hydrocarbons. This is also confirmed in [15] in the fiberglass box case where the deposits originate from the outgassing of the glues used to build the detector. In this very same study different glues are compared since the least outgassing glue is desired. At higher temperatures glues tend to outgass more so the detectors were tested at room temperatures and above.

2.4. Analysis Tools: ROOT

ROOT is an object oriented framework that was designed to process large amounts of data efficiently. It provides the user with a vast array of objects and fast access to their attributes ideal for the high volume data processing needed in high energy physics. In fact, it is the most widespread software in use in the field [17]. ROOT's native language is C++ which makes it relatively easy for new users to learn. Many of the objects designed in this framework are essential to the analysis that follows. The main advantage of using ROOT for this study is the way it can treat clusters of data as objects; therefore providing a complete set of methods that can be implemented or attributes that can be extracted for each object (data cluster). It provided the histogram construct, fit capabilities and graphic visualization tools needed for the creation of the scripts presented in Section 4.

Section 5 deals mainly with the spectra generation and fits. The spectra corresponds to histograms created by using ROOT. With the provided capabilities the histogram objects packaged six hours of data each and extracting statistical information about this data was as easy as accessing the object's attributes. For example the histogram mean could be extracted with one line of code. Fits can be efficiently created and are also treated as objects; therefore the fit parameters can be extracted with ease as well.

3. Experimental Setup and Measurements

To observe the aging effects in single wire detectors two of these detectors were run at the same time over the course of a year. Except for the periods with CF_4 concentration changes the mixture of gasses in the chambers is composed of 45% Ar, 15% CO_2 and 40% CF_4 [1]. One of them is irradiated by a source at a constant position while in the second the detector is moved to different positions throughout the year.

3.1. Timeline

3.1.1. Single Wire 1 (SW1)

1. 04/09/13: Start of Irradiation. Fixed position.
2. 07/01/14: Unexpected gain drop
3. 28/04/14: CF_4 gas mixture changes start
4. 28/05/14: CF_4 gas mixture changes end
5. 14/07/14: End of Irradiation

3.1.2. Single Wire 2 (SW2)

1. 04/09/13: Start of Irradiation. Position 3.
2. 01/10/13: Move to position 4.
3. 01/11/13: New SW2. Position 3.
4. 21/11/13: Soldering on pin connector.
5. 07/01/14: Position 3 (bottom).

[1] The reason this 45% Ar, 15% CO_2 and 40% CF_4 mixture was chosen is that this single wire detector will be used to monitor the gas mixture sent to a detector that uses exactly this gas mixture composition. The standard gas mixture used is composed of 70% Ar and 30% CO_2.

6. 27/02/14: New SW2 position 4.
7. 28/04/14: CF_4 gas mixture changes start
8. 28/05/14: CF_4 gas mixture changes end
9. 14/07/14: End of Irradiation

3.2. Radiation Source

- ^{55}Fe source producing 5.9 KeV photons.
- Activity (measured in becquerels: [Bq] = 1 decay per second): 1 MBq

4. Script File Map

Raw data from the detectors was collected continuously for a year alongside many environmental parameters (such as temperature, pressure and oxygen levels within the detectors). This constituted a challenge for the analysis that follows since problems with the data acquisition software arose and all this data had to be carefully matched. The following series of scripts were created to carry out this process (Refer to Figure 4). The general idea is to group clumps of data into histograms from which different trends are to be extracted through carefully designed fits. These trends, such as FWHM, RMS, peak position or gain, fit means, integral and integral noise are plotted as a function of time or integrated charge to analyze how aging affects them. Raw environmental data is included to correct for fluctuations in temperature and pressure. Finally, supplementary scripts calculate specific relations between different parameters to give a better understanding of the aging effects.

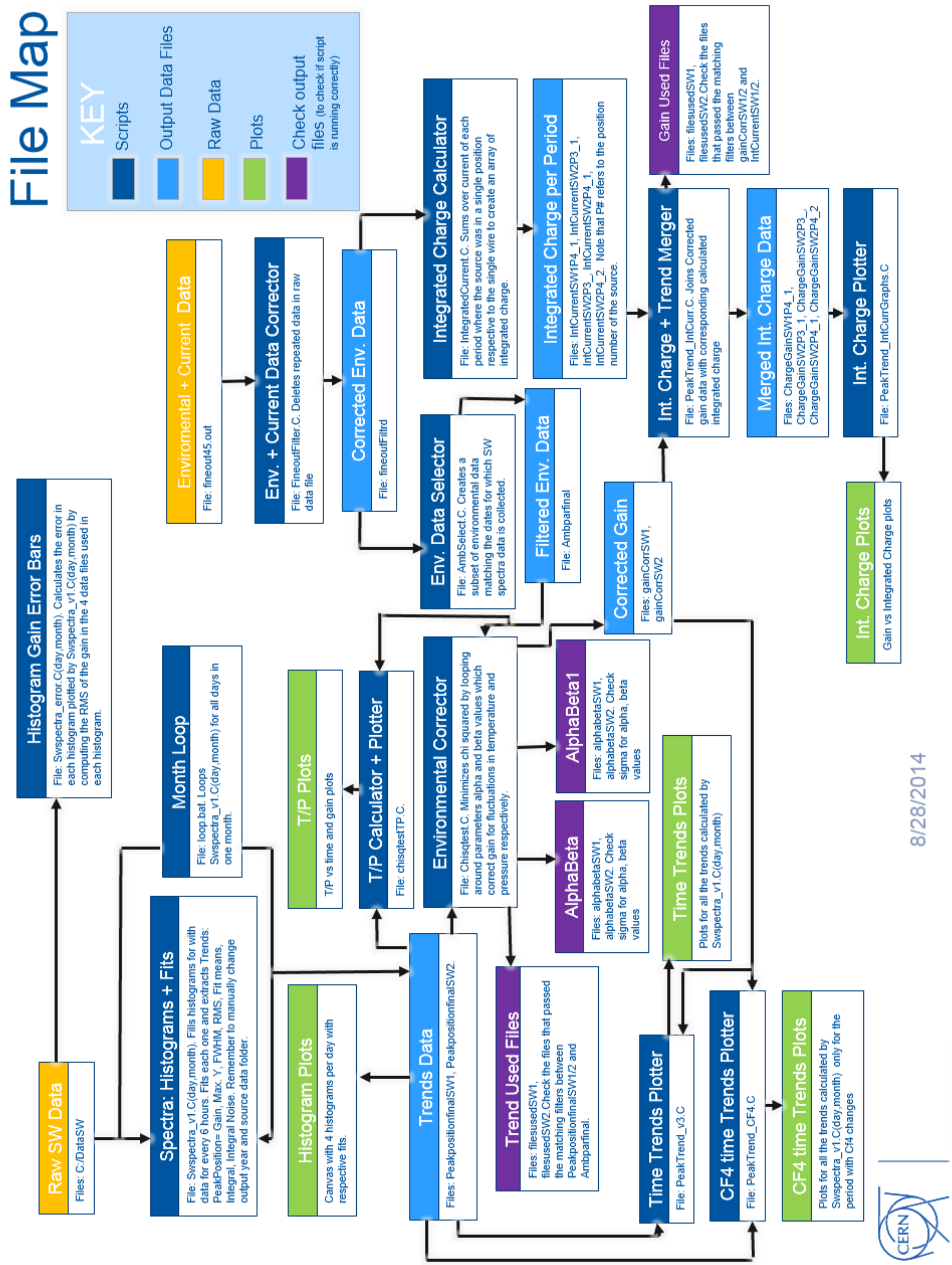

Figure 4.1: File and script map

5. Spectra Generation

The energy spectrum is of great importance to understand the detector performance. Depending on the radioactive source strength and its distance from the detector itself the spectra (with a statistically significant number of counts) can be generated within seconds to hours of irradiation. However the quality of these measurements are only as good as the detector itself. In this Section the effects of aging in the detector will start to become apparent through a systematic generation of statistically significant histograms, their fits and the evolution of fit parameters over time.

5.1. Introduction

In order to understand the following methodology a few terms should be defined:

- Histogram: a graphical representation of the Single Wire data distribution. Effectively the number of counts per energy value in the detector; also known as the Spectrum.
- Bins: The energy range (represented as ADC counts: refer to x-axis of Figure 5.2) is divided into subranges called bins. The histogram is the representation of particle counts incoming with an energy that falls within this energy subrange (bin).
- Binning: The size of the bin or number of bins for a given range. Increasing bin size effectively decreases resolution but also eliminates variability within that bin (only a computational artifact).
- Pedestrial: An artificial signal (created by the electronics in the setup) that is present throughout most of the data. This signal is purposefully inserted at a lower energy (than the detected particles) in the spectrum to serve as a marker to detect electronic shifts. If the electronics themselves are failing (ADC shifts can occur due to these malfunctions) the real signal can be extrapolated by subtracting the signal coming from electronics

5.2. Scripts: Spectra: Histograms and Fits (SWspectra_v1.C)

This script constitutes the first step in processing the Raw SW Data. Through a series of techniques it groups data into histograms and produces their corresponding fits. For every histogram created a series of parameters is collected and stored as Trend Data in the PeakpositionfinalSW1.txt and PeakpositionfinalSW2.txt files. The collected parameters are as follows: Peak Position or Gain (histogram mean), Fit Means (two Gaussian are used to fit the histogram, refer to Section 5.2.3), and Full Width at Half Maximum (FWHM), Root Mean Squared (RMS), Maximum, Integral, and Integral Noise of the histogram. This script by itself only runs the data collected from one day at a time [2].

[2] There is a supplementary script that runs this script for multiple days at a time. Refer to Section 5.2.5

5.2.1. Grouping Data into Histograms

Since data is being collected continuously, an appropriate choice of data sampling and grouping is needed. The quantity of data grouped into a histogram has to be sufficient to provide the correct statistics for the time slot that that group corresponds to. Nevertheless it can't be too large since parameters can fluctuate too much within that time period. In fact, a script was created just for this purpose. Data is sampled every six hours resulting in four histograms a day each containing one hour of data. This is enough to be statistically significant; however, the question that arises is that if environmental (among other) fluctuations can significantly change measurements during the course of an hour. The output of this script shows that the error ranges within 1% of the measured parameters and therefore the sampling size and rate are appropriate.

5.2.2. SW Loops: Histogram Arrays

SWspectra_1.C runs all the data (mean, max, RMS, FWHM, integral...) collection code from the histograms (h0_1[d][m], h6_1[d][m], h12_1[d][m], h18_1[d][m], h0_2[d][m], h6_2[d][m], h12_2[d][m], h18_2[d][m]) in one loop for each single wire instead of running each histogram independently. This has a slight disadvantage in that parameters (such as fit parameters) can not be set specifically for each histogram but rather have to work in general for each day. Ultimately a general approach is what we are looking for since we want to run this data once over the course of a year; nevertheless, it makes individual corrections more cumbersome to achieve. To carry out the loops for the single wires a single array of histograms was created containing all the already filled (h0_1[d][m], ...) histograms: testarray. Thus data collection operations are conducted on the individual elements of this array looping over its indexes.

5.2.3. Fits and Methodology

The fit methodology is of great importance for the purposes of this study. Effectively during the creation of the scripts, major differences in output data files was observed when fit methodology was changed. Ideally the observed spectra should be Gaussian; nevertheless in reality this spectrum can become asymmetric over time eventually resulting in the appearance of a second peak. The wire diameter is changing due to aging; therefore, different electric fields are present in the two sides of the single wire. The second peak is a direct result of this. Each histogram is thus fit as the sum of two Gaussians. This method was effective for clear double peak spectrum but failed when the spectra was mostly a single Gaussian peak. Therefore a guided two Gaussian fit method was developed as shown in equation 5.1.

$$f(x) = a_1 e^{-\frac{(x-b_1)^2}{2c_1^2}} + a_2 e^{-\frac{(x-b_2)^2}{2c_2^2}} \tag{5.1}$$

Where a_1, b_1, c_1, a_2, b_2, and c_2 are the fit parameters. It is important to understand what the effects of a guided fit may be. If no guidelines (the parameters in equation 5.1) are set the fits will fail for an unacceptable percentage of cases. However if not enough liberty is given the fits wont resemble the data. Therefore it is important to have a careful balance between what is set and what is left to fit calculation.

Due to aging the peaks (however small) will drift towards opposite sides of the histogram mean. Therefore the Gaussian fits were forced to be on opposite sides of the histogram mean by setting the corresponding range for the parameters b_1 and b_2: $Min(Range) < b_1 < Mean(histogram)$, $Mean(histogram) < b_2 < Max(Range)$. Here the Range is that calculated by the algorithm explained in Section 5.2.4. This constraint alone increased the fit rate from around 50% to around 80%. Furthermore more evident fit limits were set, such as matching the maximum of the fit with the maximum of the histogram.

5.2.4. Peak Rage for Fits

Originally, the range for which all the data was collected from the histograms was fixed. This made some fits fail when the peak moved out of range and it creates the need for an algorithm to find where the main peak is. The main method is located in lines 273-317 in the code included in Appendix 9.6. This method consists on setting a threshold y-value on line 268 (needs to be above the noise). If values are found above this value they are considered part of the main peak, unless they are part of the pedestrial. Hence, the first two points where the threshold is passed are discarded since they are part of the pedestrial. The third and last points where the threshold is passed is considered the limits of the main peak. If no pedestrial is present (not typical) line 305 must be activated. Now the first and last instance where the threshold is passed are considered the limits of the main peak. Notice that however necessary the presence of the pedestrial, it complicates the programing and flow of the data processing.

A few bins are added on each side of the range to recapture the data missed by the threshold setting. Since a higher threshold will result in a higher percentage of fits working an optimal setting of threshold and additional bins (on each side) must be searched for depending on the noise in the data and the chosen binning. Note: FWHM calculation uses same algorithm but with the half maximum as a threshold. Due to the subtleties of this methodology the FWHM only has the accuracy of the size of the bin, thus the calculated FWHM values are somewhat clumped into bins as demonstrated in Figure 5.1.

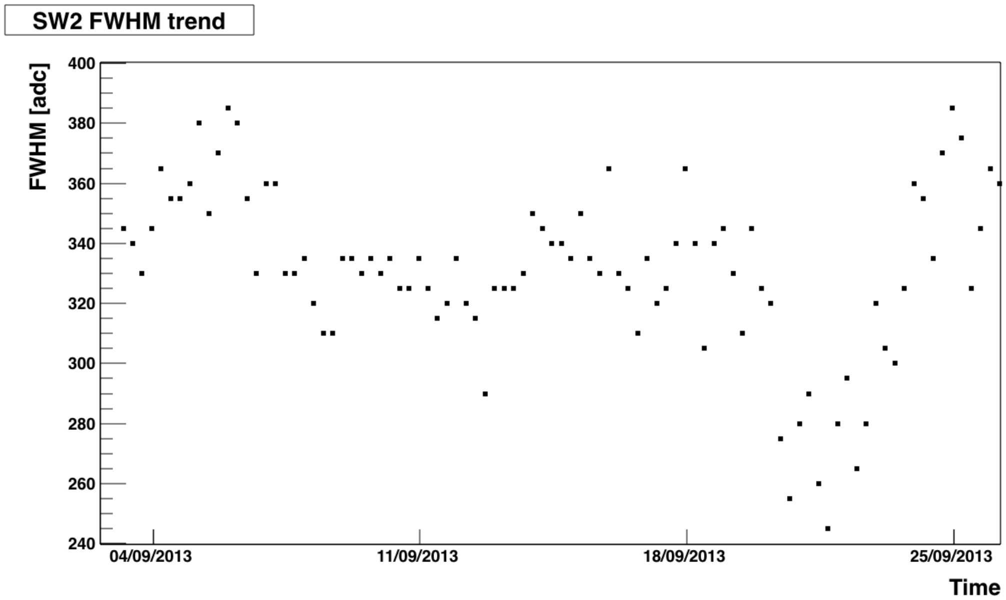

Figure 5.1: FWHM time trend over first month of detector operation of SW2. Values of FWHM can only be multiples of 5 due to the set bin size of 5 ADC counts.

5.2.5. Month Loop

Recall that the SWspectra_1.C(d,m) script just generates histograms for one specific day of data. To generate the histograms for the entire year this code must be run iterativelty by changing the date in each iteration. However since this code can not run inside another root script it must run inside the Unix Shell Bash. The month loop is a very simple bash script that runs this code in the described manner for each month. Hence, instead of running the code for each day of the year individually the bash script is run for each month of the year drastically saving user time [18].

5.3. Results and Anlysis

5.3.1. Spectra

The methodology presented in Section 5.2.3 is exemplified by Figures 5.2 and 5.3. Here both discussed situations are shown for which the fits were effective. Figure 5.2 shows no sign of aging since just one gaussian peak is observed. The fits had to be fine tuned so a high percentage of single peak spectra like this one could be fit.

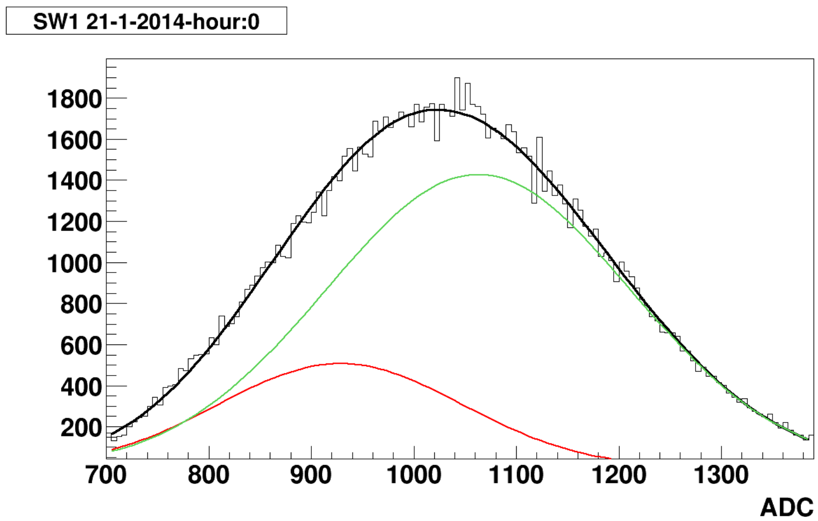

Figure 5.2: Two Gaussian fits applied to a mostly Gaussian peak. Note the positions of the peaks with respect to the mean of the histogram. The sum of both Gaussian peaks is shown in black.

On the other hand, Figure 5.3 shows clear signs of aging. The fits worked exceptionally well for these cases since there is a clear double peak.

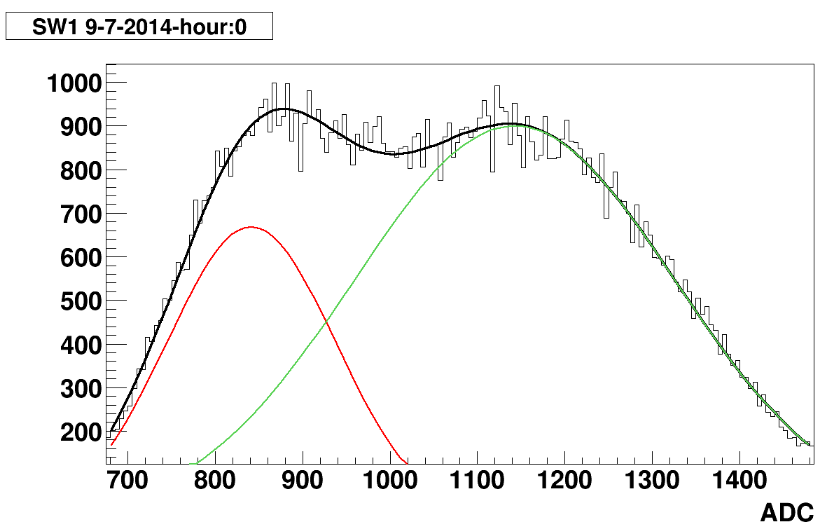

Figure 5.3: Same as in Figure 5.2 but with a clear double peak histogram.

To compare to the illustration in Figure 2.5 a series of histograms are set side by side in Figure 5.3.1. It is clear that the aging effect is manifested in this set of 5 histograms evenly selected over the course of a month.

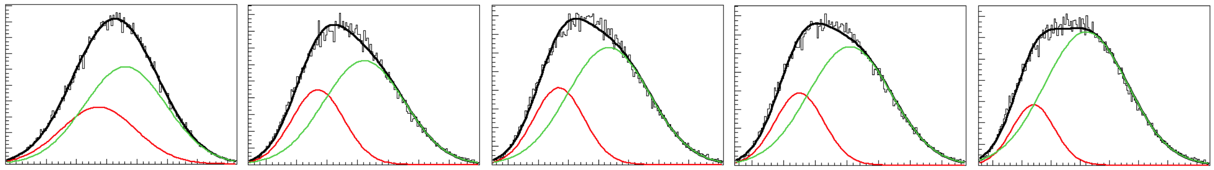

Figure 5.4: Five pulse-height distributions as they develop durring irradiation of SW2 spread out over a month (between the dates 1 and 2 in Section 3.1.2).

With the techniques described in Section 5.2.3 the code was able to fit around 95% of all histograms. This is a vast improvement over just fitting the histograms with the unconstrained sum of two gaussians fit given in Equation 5.1. In fact the latter only produced a fit rate of 50%.

Even though the aging effect is not as dramatic as in Figure 2.5 the gradual transition to two peaks is observed. This becomes evident by analyzing carefully the trend of the individual peak means as follows.

5.3.2. Fit Means

The yearlong fit means trend can be constructed by extracting the means from the two Gaussian calculated through the guided fit described in Section 5.2.3. These are presented for SW1 and SW2. Each of the means are normalized with respect to their original values where no aging is present.

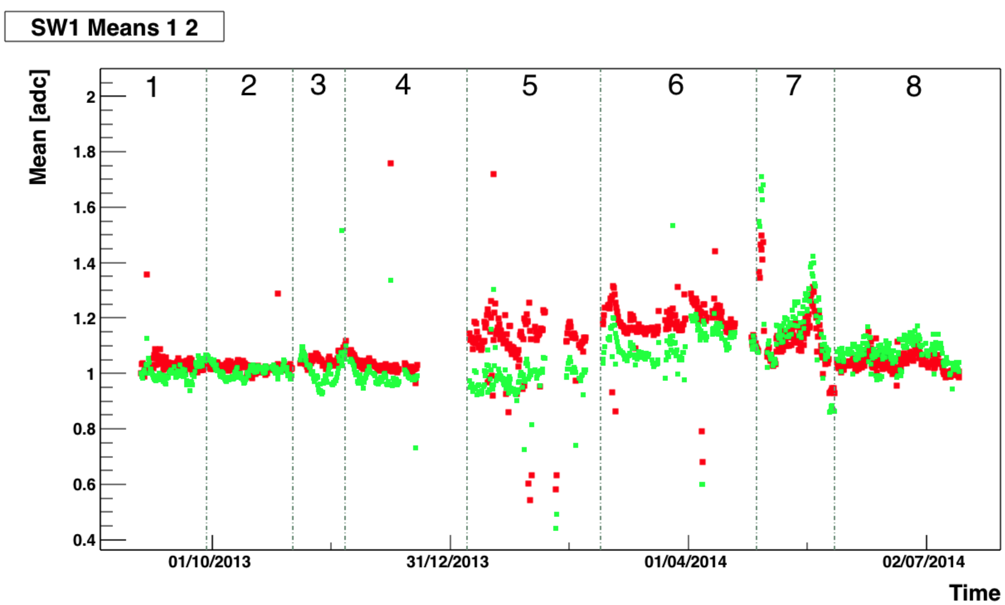

Figure 5.5: Trends of the normalized means of the two Gaussian fits applied to each histogram over one year of detector operation of SW1. The dashed lines mark the dates corresponding to those in Section 3.1.2 for SW2 for this plot and all that follow.

In this case we are interested in the in the trend for the first two periods of SW1 compered with that of SW2. In general this will be the case for most of this study because either more complex variables come into play during some portions of the year (like the month of CF_4 changes) or there is not a clear trend. In SW1 the means hover around each other depending on how the fit converged. However for the first two periods in SW2 there is a separation of the means over time.

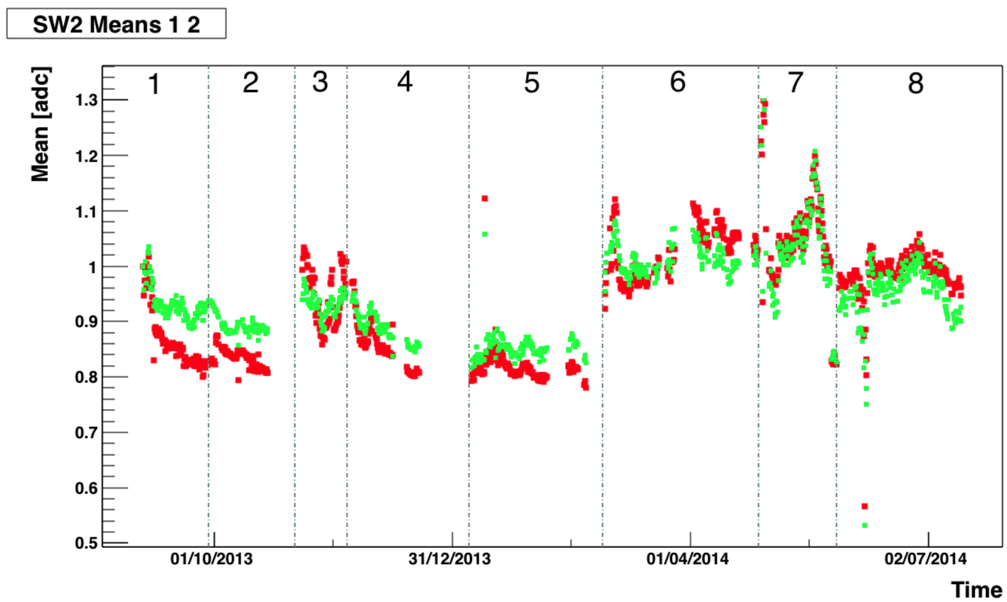

Figure 5.6: Trends of the normalized means of the two Gaussian fits applied to each histogram over one year of detector operation of SW2. The colors used for the different means match those in Figure 5.3.1.

Zooming into the first period of the year this becomes even more apparent.

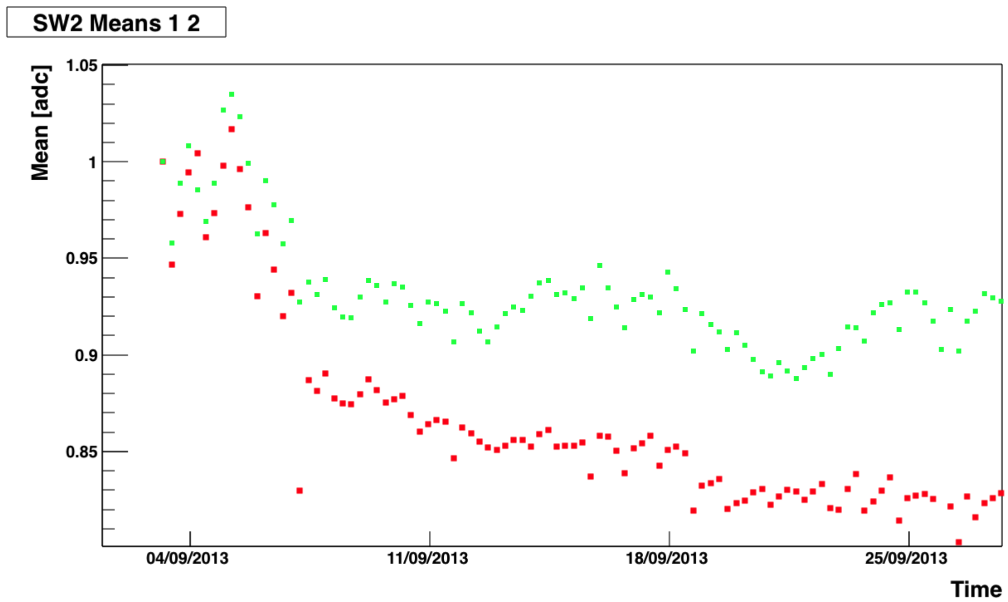

Figure 5.7: Trends of the means of the two Gaussian fits applied to each histogram over one aging period of SW2. This data corresponds to the period between the dates 1 and 2 in Section 3.1.2.

To quantify this difference the relative separation between the peaks can be calculated.

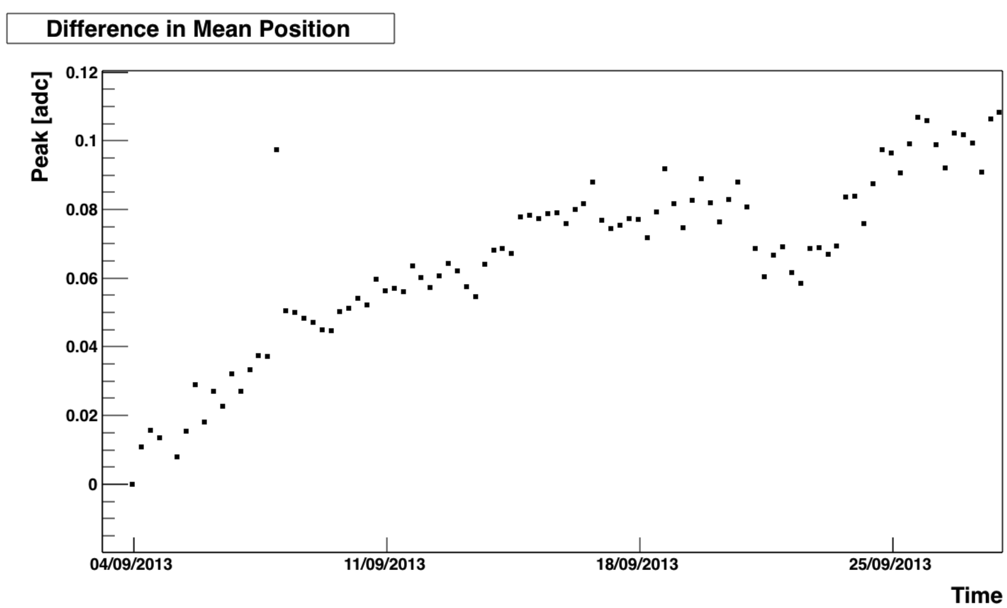

Figure 5.8: The distance between the original peak and the emergent peak (the two fit means) for the period decribed in Figure 5.7. Since the means in Figure 5.7 are normalized to the initial value of the period this plot represents the fractional difference between the two peaks.

The separation grows rapidly at the beginning and then this rate tappers off. This is a clear sign that the aging effect rate decreases over time. As stated in Section 2.3.1 the aging effect occurs due to deposition in the wire. Since the rate of the aging effect decreases then the initial deposits have a greater effect than the ones that follow. Setting all the constants equal to 1 except a and eliminating the r dependence in Equation 2.2 results in:

$$f(a) = \frac{1}{\ln\left(\frac{1}{a}\right)} \tag{5.2}$$

Where a is the now variable radius of the anode. The derivative of $f(a)$ provides insight into the sensitivity of the electric field with respect to this value.

$$\frac{d}{da} f(a) = \frac{1}{a \ln^2(a)} \tag{5.3}$$

Since we set b (the inner radius in the cylinder of the detector) to 1, an approximate value for a would be $a = 0.001$ given the standard specifications of this type of detector. Henceforth we can study Equation 5.3 in this region.

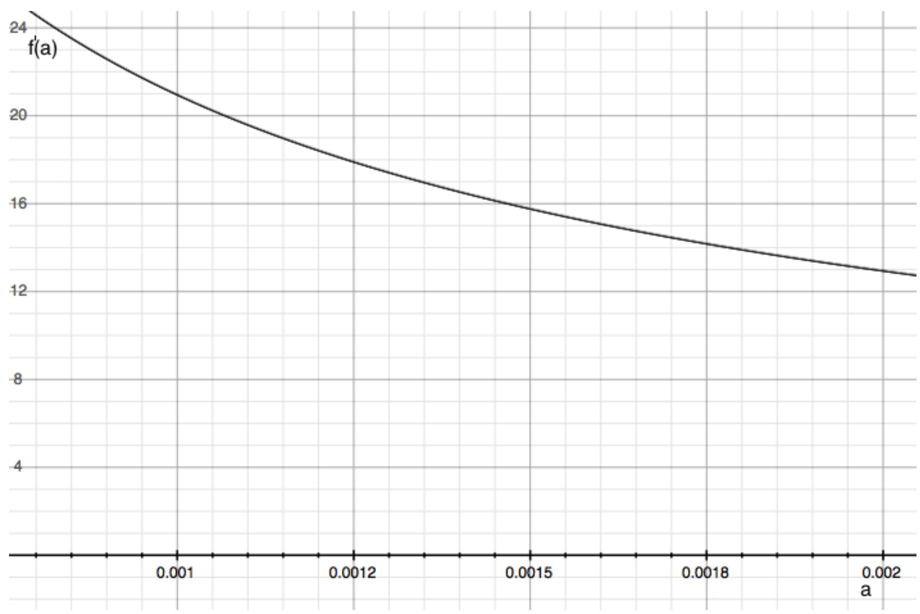

Figure 5.9: A graph of Equation 5.3 within the range defined by standard single wire detector specifications.

The derivative of $f(a)$ is decreasing, since the electric field changes more dramatically at lower values of a. This explains why initial deposits have larger effects than further ones thus slowing down the aging effect.

5.3.3. Relative Gain

As opposed to extracting the means of the two peaks from the fits themselves as in the previous section the mean of the peak as a whole is gathered from the actual data within the histogram.

This mean is what could be converted to the energy of the incoming particles. However, this is not of interest in this discussion. Since the energy of the particles is constant the change in this mean will reflect a change in the gain of the detector. Thus the relative gain is defined as:

$$Relative\,Gain = \frac{instantaneous\,Gain}{Initial\,Gain} = \frac{instantaneous\,Peak\,Mean}{Initial\,Peak\,Mean} \tag{5.4}$$

The Relative gain is now plotted for the entire year for SW1 and SW2 in Figures 5.10 and 5.11 respectively.

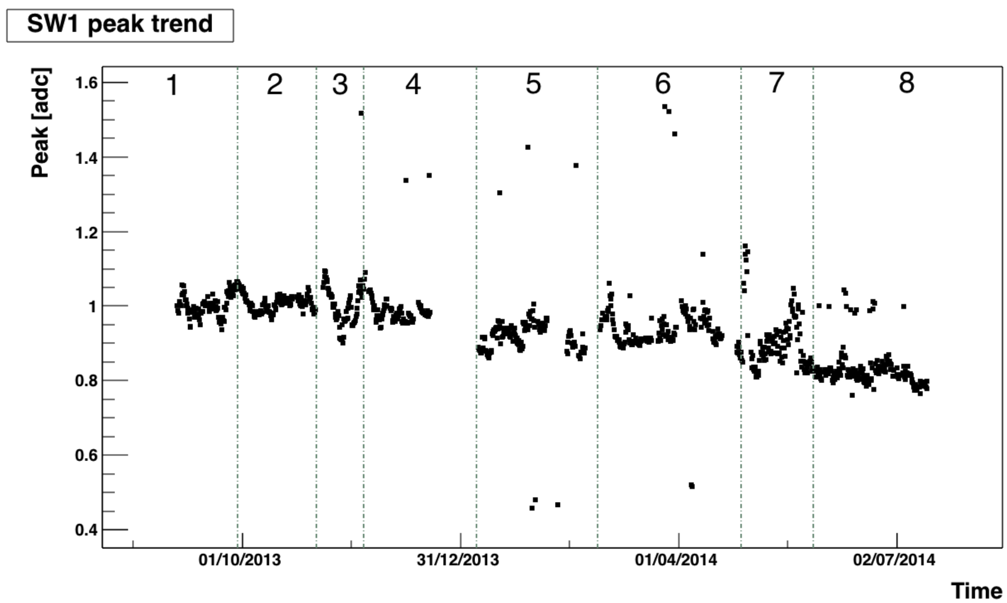

Figure 5.10: Gain time trend over one year of detector operation of SW1

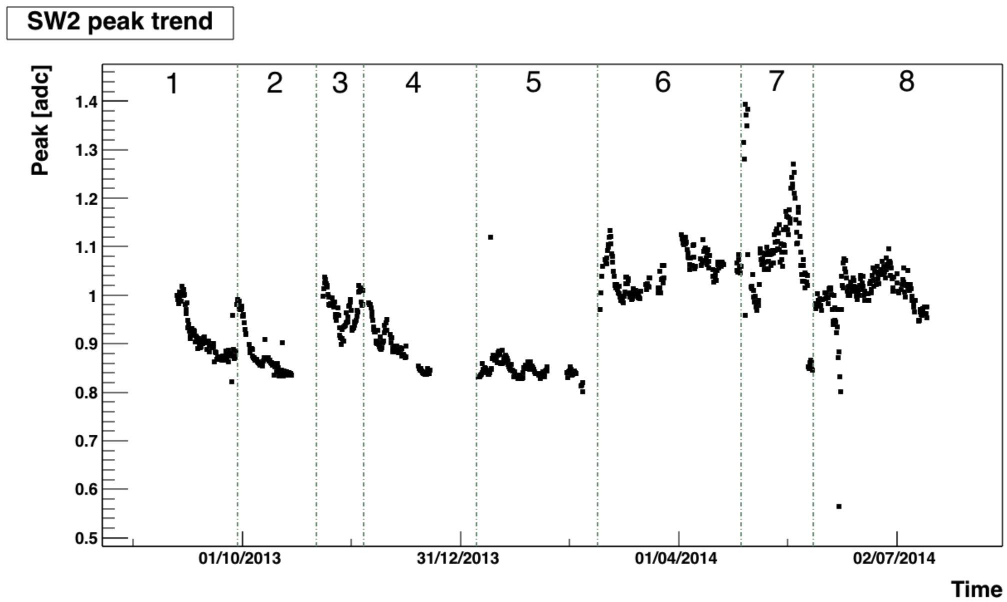

Figure 5.11: Gain time trend over one year of detector operation of SW2

Again the focus will be on the first 2 periods of this trend. As expected the gain follows the same trend as the individual means in the previous Section. SW2 undergoes an aging process during the first month of irradiation. Then the detector is moved to a different position (refer to the timeline in Section 3.1.2). The relative gain shoots up after this change. This occurs because the deposits in the anode are not evenly spread out [10] and this spread depends on the position of the source. Since the detector was moved, a section of the wire with less deposits must be responsible for the recovery of the gain almost to its initial value. This followed by a similar aging period. In 01/11/13 (the start of period 3) the wire was swapped for a new one and the original gain was completely recovered as observed.

5.3.4. Resolution

The resolution is calculated by plugging in the instantaneous peak mean from the previous Section and the FWHM calculated as explained in Section 5.2.4 into Equation 2.1. This is done for the entire year of data to produce the plots for SW1 and SW2. To ilustrate this Figure 5.12 follows.

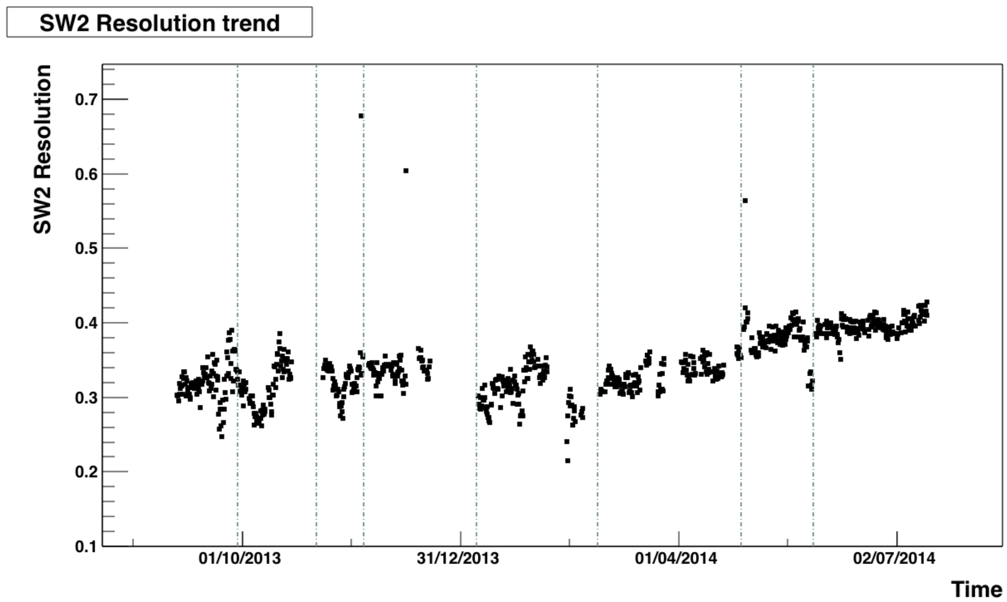

Figure 5.12: Resolution time trend over one year of detector operation of SW2

Since there are two changing variables coming into play for the resolution this trend is harder to analyze. The two initial periods that in SW2 that exhibit the aging effect in previous sections show a distinct pattern in the resolution trend. There resolution first decreases and then increases for each aging period.

5.4. Conclusions

Massive amounts of data require careful packaging and processing. By grouping the particle count data in 6 hour packages, 4 spectra or histograms can be generated per day. It is important to remark on why the data was chosen to be analyzed at this level. For example one histogram could have been generated every day or every 10 minutes, so why was one generated every 6 hours? This relates to the variance of the different variables at play during each chosen histogram period. As for environmental conditions in a laboratory setting, they vary very little and an average over 6 hours is sufficient. As for the aging process, it has been observed to occur in the order of a month. Therefore with the chosen data analysis level, around 120 data points are produced for study for the periods that present aging. This value is sufficient while not being computationally overwhelming. By taking advantage of ROOT's ability to process data in histograms (which are treated as objects), statistical parameters such as the histogram mean (which is converted to relative gain) can be accessed easily by calling one of the histograms attributes. Each one of these histograms are fit by a function resulting from the sum of two gaussians. Nevertheless this fit must be guided for the fits to work for the entire year of operation. Since aging effects are present the fit must be able to handle single peak histograms and double peak histograms as aging causes the single peak to separate over time. This increase in separation becomes evident by studying the means of the individual peaks produced by the fit and plotting their difference over time. Furthermore the aging effect is more pronounced at the beginning of each period due to the dependence of the electric field on the changing anode radius due to deposits. These deposits are the cause of the aging effect. The relative gain gives further insight into this process. Every time the anode is changed or the detector is moved the detector recovers its initial gain (or a large fraction of it for the later). A "V" shaped pattern in resolution emerges in the periods with aging effects. Unfortunately all the effects are studied just for the initial 2 periods of SW2 which exhibits clear aging periods and SW1 which does not for that time frame. A more complex analysis is required for the rest of the year since other variables come into play.

6. Environmental Effects

As seen in Equation 2.11 temperature and pressure fluctuations can significantly affect instantaneous measurements in the detector. Thus to produce accurate and reliable results the effects of these fluctuations must be removed. This Section contains different techniques for doing so.

6.1. Scripts: Environmental Corrector (chisqtest.C)

Fluctuations in temperature and pressure can induce significant changes in the behavior of the detector as explained before. Therefore, environmental corrections are very important since they eliminate the temperature and pressure dependence of the gain. This script cycles through two parameters to minimize the standard deviation of the gain in periods with no aging where gain variance is caused in part by these environmental fluctuations. The corrected gain is then printed alongside the uncorrected gain and the FWHM.

6.1.1. Data Matching

Since raw data is coming from 2 sources, the spectra data has to be correctly matched to the environmental data. This is done to some extent independently of this script. The raw environmental data contains many repeated sequences caused by errors in data acquisition. The Environmental and Current Data Corrector (FineoutFilter.C) cleans up these errors. The corrected data then passes to the Environmental Data Selector (AmbSelect.C), a script that samples the environmental data every six hours to match the spectra data sampling. Nevertheless, chisqtest.C does a final check in lines 238-248 (Appendix 9.2). This final check runs through all the lines in the environmental data file for each of the lines in the spectra data file. When it finds the corresponding line in the environmental data file the values are pasted together and passed to the environmental correction section. If no match is found for a line in the spectra file this data is discarded. Note that the environmental data file does not need to be in chronological order for this to work.

6.1.2. Methodology

A correction term for temperature and one for pressure is introduced in the calculation of the gain as shown in Equation 6.1.

$$G_i = G_0 \left(\frac{T_i}{T_0}\right)^{\alpha} \left(\frac{P_0}{P_i}\right)^{\beta} \tag{6.1}$$

Here G_0 is the uncorrected gain, T_0 and P_0 the average temperature and pressure respectively over the correction period, α and β are the correction factors, and G_i, T_i and P_i the instantaneous corrected gain, temperature and pressure respectively.

The methodology for the correction is a minimization of the standard deviation of gain over a period where no aging effects are present (where the gain is constant except for the fluctuations induced by temperature and pressure). To reduce the standard deviation the gain was calculated for a wide range of α and β values over the selected period. The value to be minimized is then:

$$\sum_i \left(G_i - \overline{G_i}\right)^2 \tag{6.2}$$

Where G_i is the corrected gain from equation 6.1 and $\overline{G_i}$ is the average corrected gain over the selected period. With this methodology an appropriate period is chosen for each single wire detector (SW1 and SW2) and values of α and β are looped over in a rage from 0 to 5 in steps of 0.1.

6.1.3. Executing File

1. The standard deviation is supposed to be minimized in a period where the gain is constant except for environmental factors (no aging effects). Therefore by referring to the Time Trend Plots for the Gain (of the desired single wire to be analyzed) chose a period of "constant" gain.

2. Input this period in the period selector by entering the start date in lines 253-255 and the end date in lines 263-265 (Appendix 9.2).

3. Run the file in ROOT. A prompt will appear asking for SW1 or SW2. Chose the one corresponding to the period entered.

4. Two lines will be printed on the console. The first one is the average temperature and average pressure of the entire period.

5. An output file is created. The last line signals the selected period. 1 signals a date within the period and 0 a date outside the period.

6.2. Results

For SW1 the period 1/10/13 - 1/11/13 was selected and the α and β values obtained are:

$$G_i = G_0 \left(\frac{T_i}{T_0}\right)^{1.8} \left(\frac{P_0}{P_i}\right)^{3.4} \tag{6.3}$$

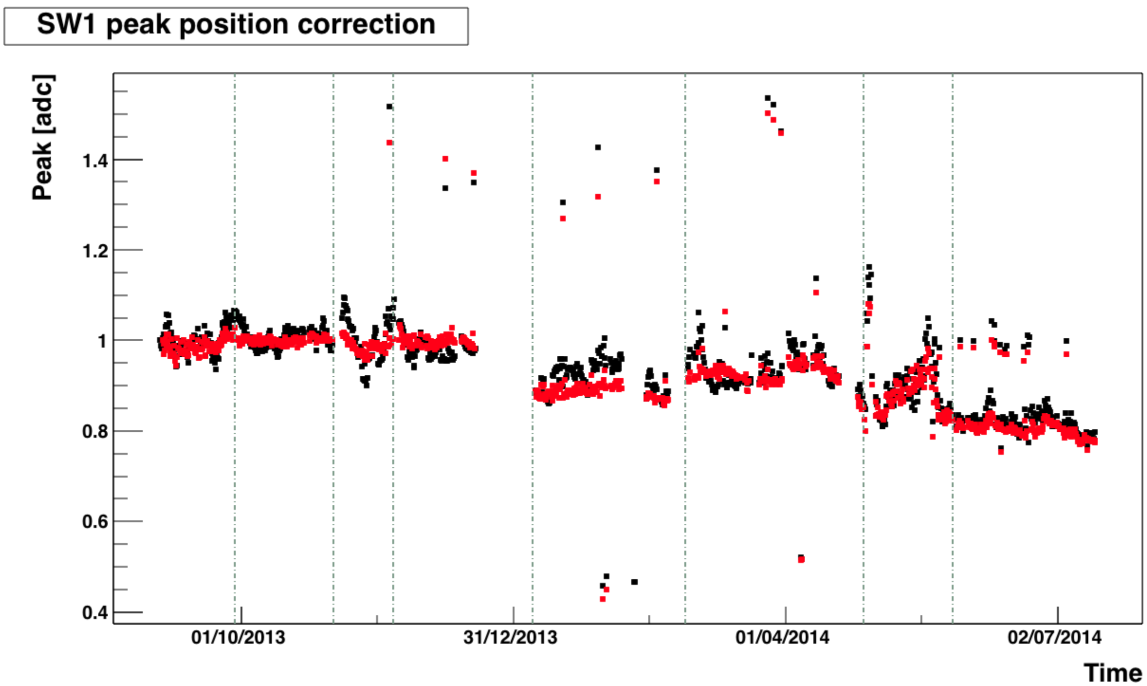

Figure 6.1: Corrected and uncorrected normalized gain for one year of detector operation of SW1. Corrected gain is plotted in red.

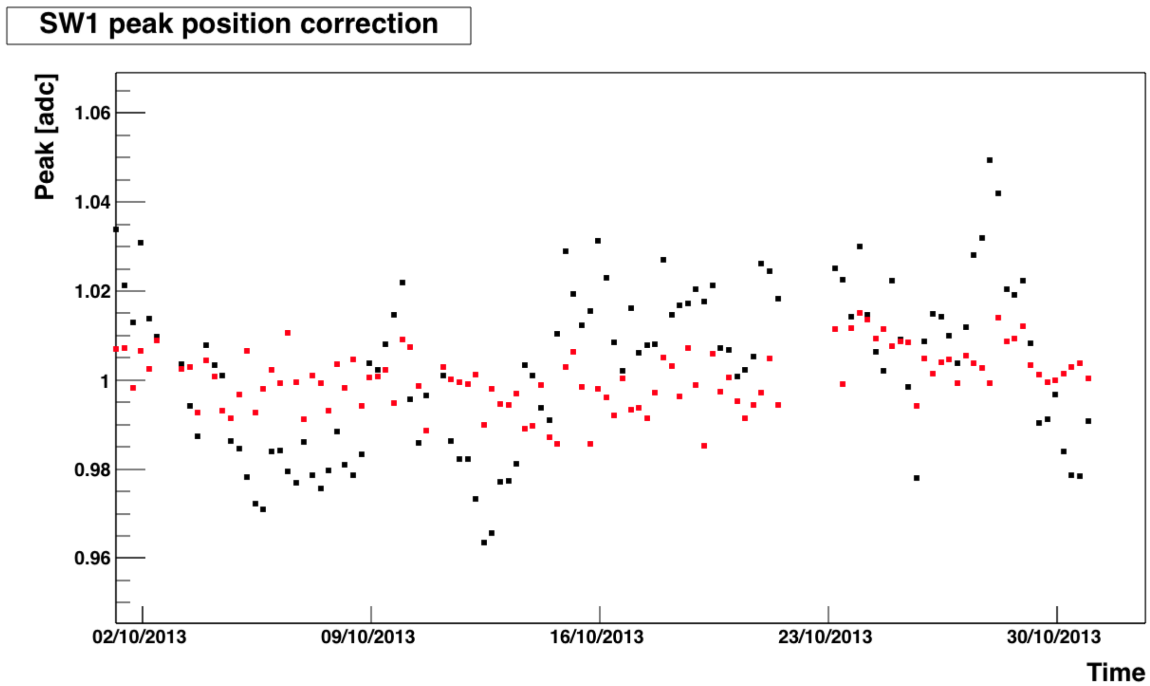

Figure 6.2: Corrected and uncorrected normalized gain the second month of irradiation for SW1. Corrected gain is plotted in red.

As it is shown by Figure 6.1 and the zoomed in version in Figure 6.2 the gain fluctuations are significantly reduced over the entire year and especially for the period from which these calculations were based. Note that even though the corrections are calculated from a subset of data they are applied to all of it.

For SW2 the period 7/1/14 - 27/2/14 was selected and the α and β values obtained are:

$$G_i = G_0 \left(\frac{P_0}{P_i}\right)^{0.8} \tag{6.4}$$

Note the absence of the temperature related correction factor. The script calculated $\alpha = 0$.

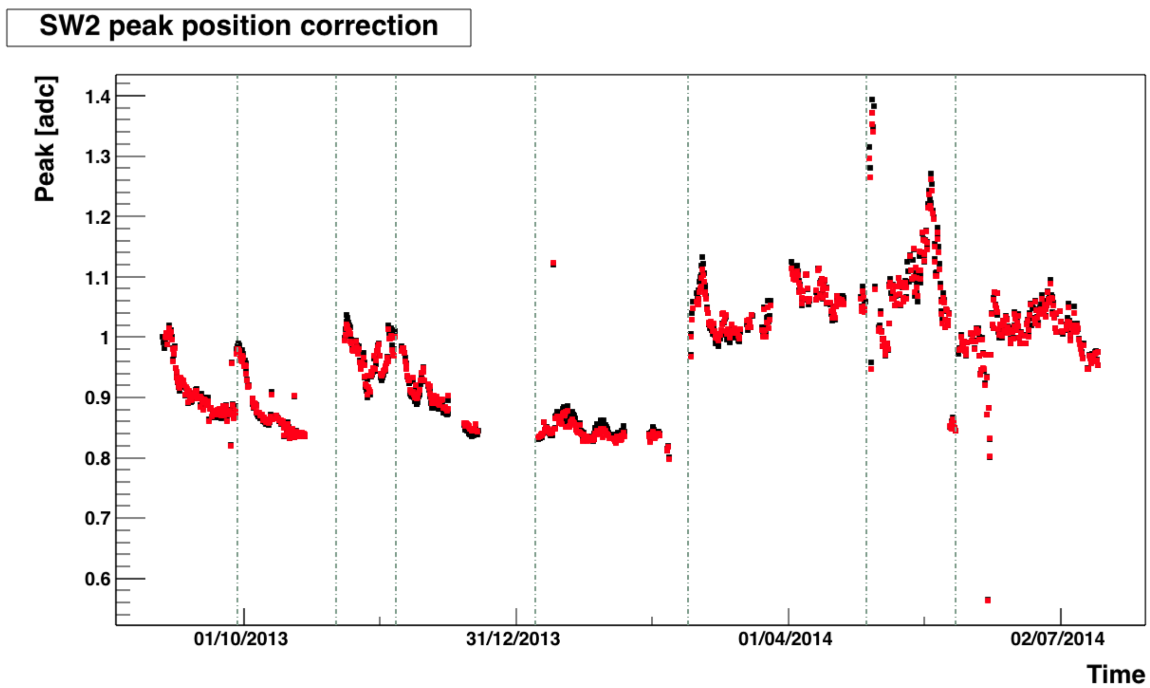

Figure 6.3: Corrected and uncorrected normalized gain for one year of detector operation of SW2. Corrected gain is plotted in red.

6.3. Analysis

In this Section the terms peak position, peak mean and gain are used interchangeably. The peak position refers to the peak mean in the spectra. Recall that these values are then directly proportional to the relative gain or simply gain as in Equation 5.4.

6.3.1. Townsend Coefficient

The Townsend coefficient and the gain are strictly correlated and it is interesting to study the effect of temperature and pressure fluctuations on the gain. The effect of temperature and pressure on the gain is modeled by the following equation.

$$G_i = G_0 e^{B\left(\frac{T_i}{P_i} - \frac{T_0}{P_0}\right)} \tag{6.5}$$

Where G_0 is the uncorrected gain, T_0 and P_0 the average temperature and pressure, G_i, T_i and P_i the instantaneous corrected gain, temperature and pressure respectively and B is the so called Townsend coefficient. In laboratory conditions temperature and pressure have small fluctuations, therefore to first order Equation 6.5 can be approximated with the following:

$$G_i = G_0 \left(1 - B + B\left(\frac{T_i}{T_0}\frac{P_0}{P_i}\right)\right) \tag{6.6}$$

Therefore to study the direct effect to temperature and pressure variations on gain we generate plots of gain vs T/P. For this purpose we start selecting periods without aging. In this way the effect of T and P variations on gain can be understood independently. Here SW2 is of specific interest (refer to Figure 6.2).

There are two periods where the gain is constant (no aging effects): the fifth time subdivision and the final time subdivision. The peak position can now be plotted against T/P for these two periods.

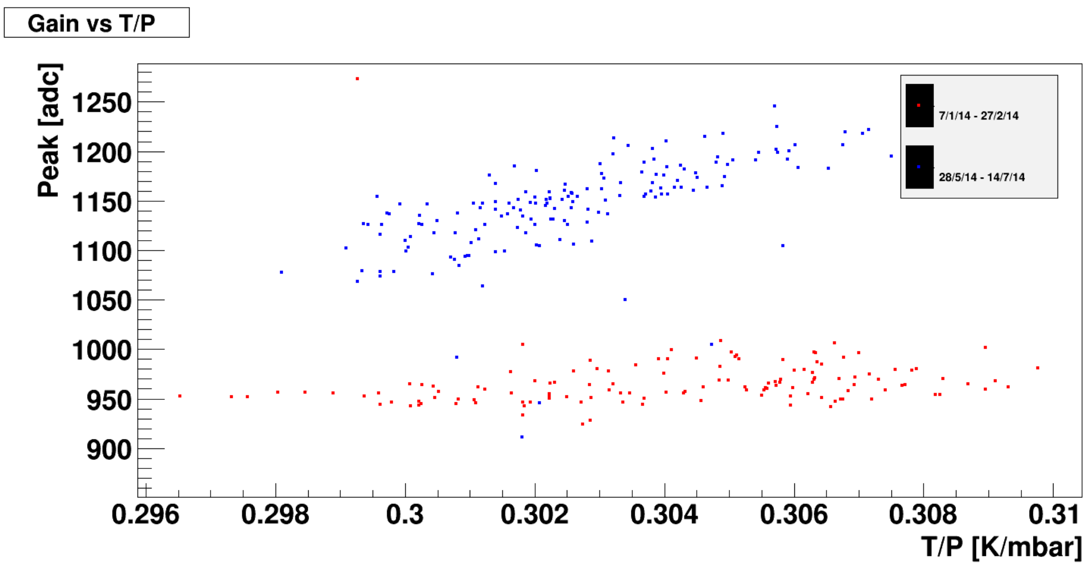

Figure 6.4: Peak Position (Uncorrected peak mean) vs Temperature/Pressure for the two periods in SW2 with no aging effects.

A similar plot but with three periods of constant gain is included for SW1 for completeness.

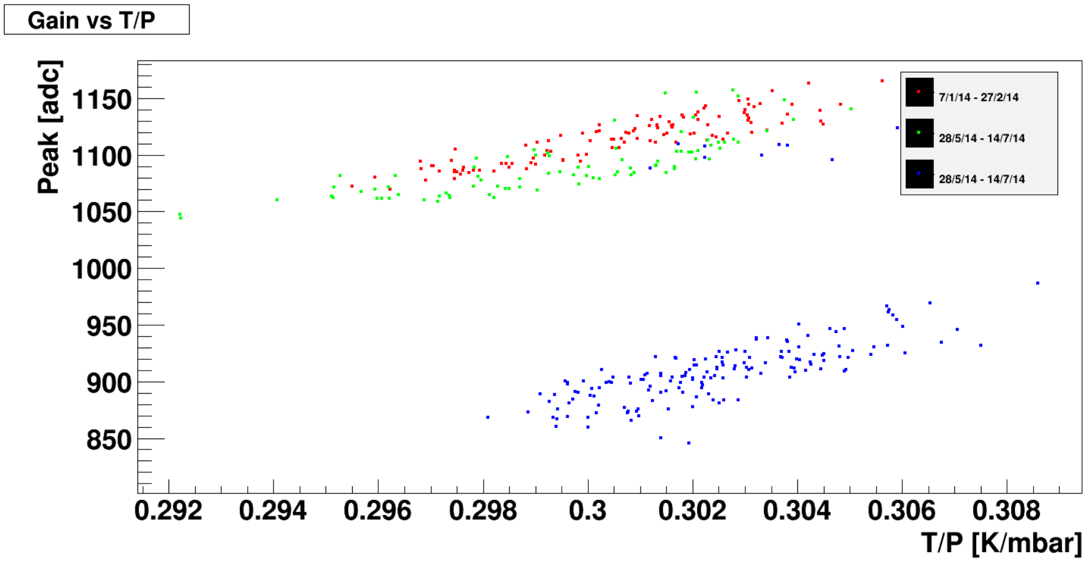

Figure 6.5: Peak Position (Uncorrected peak mean) vs Temperature/Pressure for the two periods in SW1 with no aging effects.

The slope of each one of these clusters will correspond to the Townsend coefficient. It is evident that the approximation made in Equation 6.6 is applicable because of the high linear correlation in each data cluster. Nevertheless, this plot does not explain how this relation changes over time. It is useful to plot these two parameters against time, this three dimensional plot is represented in the Figure 6.7 using a color scale to represent the z-axis. For ease of comparison to the original gain trend over a period with no apparent aging effects, Figure 6.8 is included.

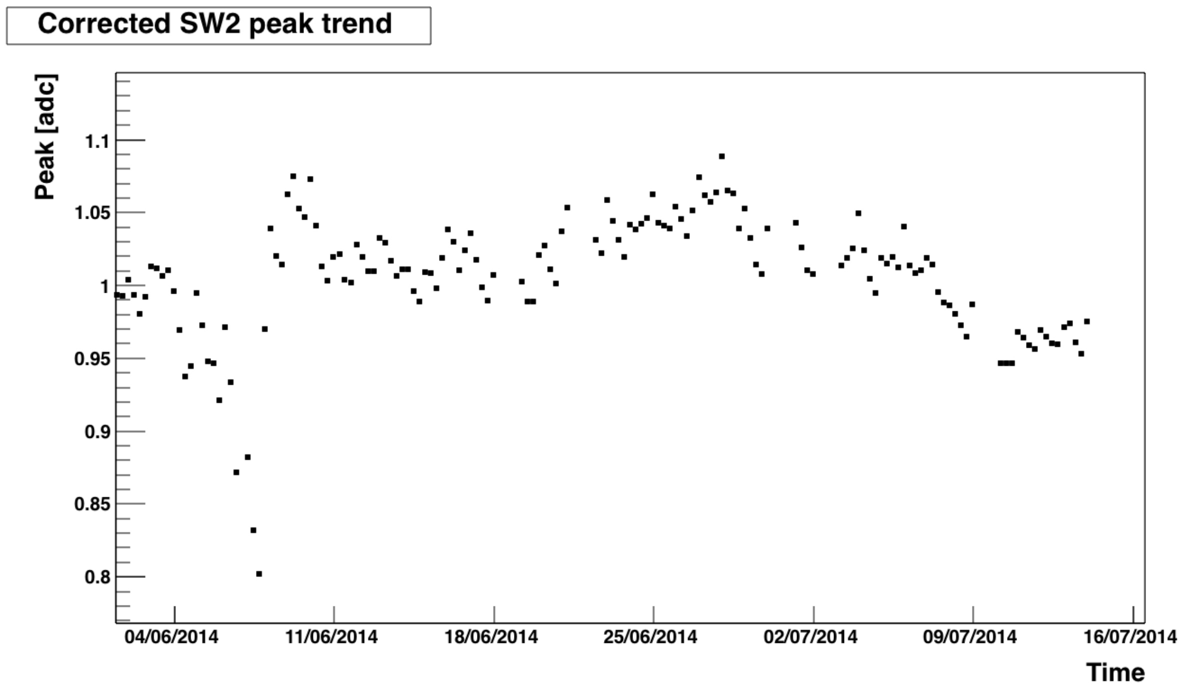

Figure 6.6: Corrected normalized gain for the period between the dates 8 and 9 in subSection 3.1.2 for SW2.

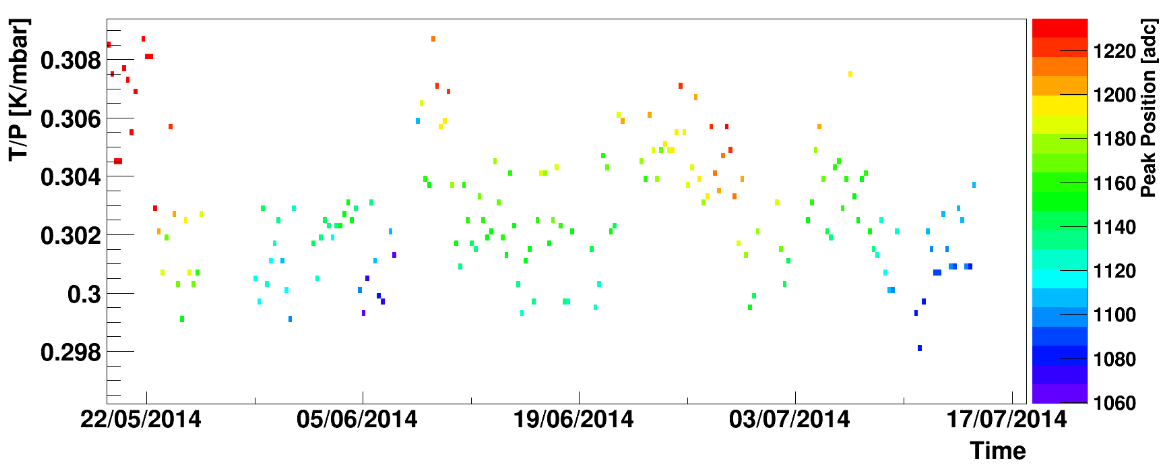

Figure 6.7: Peak Position (uncorrected peak mean (gain) (non normalized)) vs Time and T/P for the period described in Figure 6.6 in SW2.

This plot has to be read in different ways. Here horizontal color stripes can be observed throughout the whole time period. Going vertically one read off the data that is being represented in Figure 6.7: as T/P increases the peak position increases linearly. However, this plot adds another dimension to Figure 6.6, demonstrating that this trend holds constant over time. The opposite can also be done. Periods where aging is present can be plotted in the same fashion as in Figure 6.7. For ease of comparison to the original gain trend over the aging periods, Figure 6.8 is included.

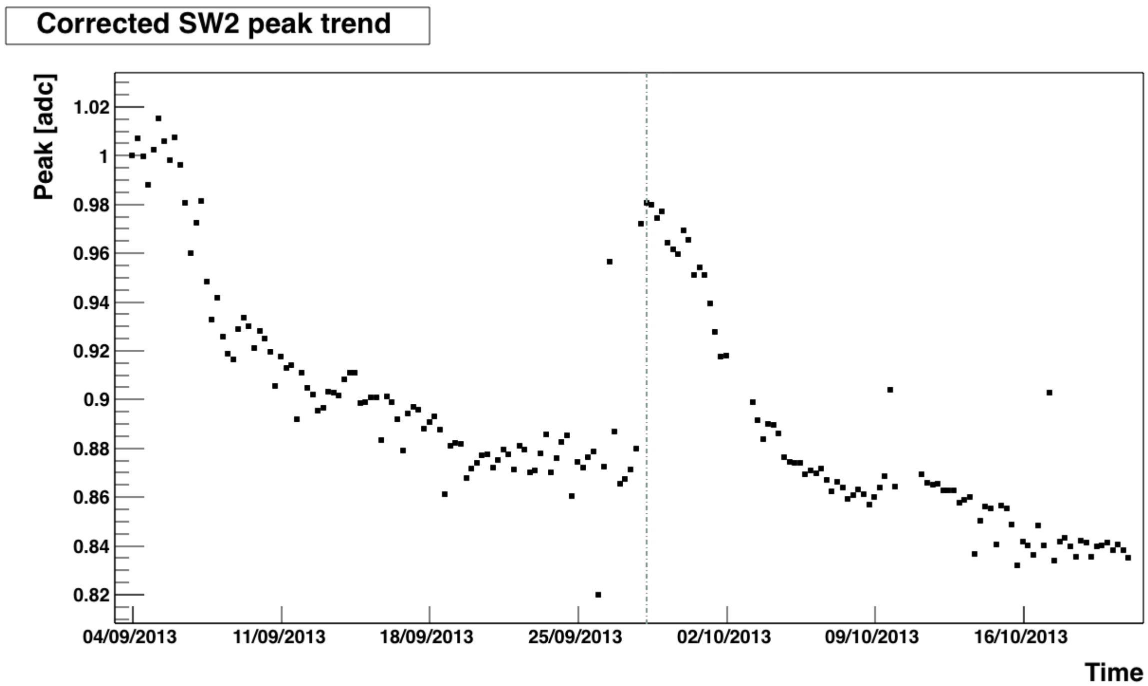

Figure 6.8: Corrected normalized gain for the periods between the dates 1 and 2 and 2 and 3 in subSection 3.1.2 for SW2.

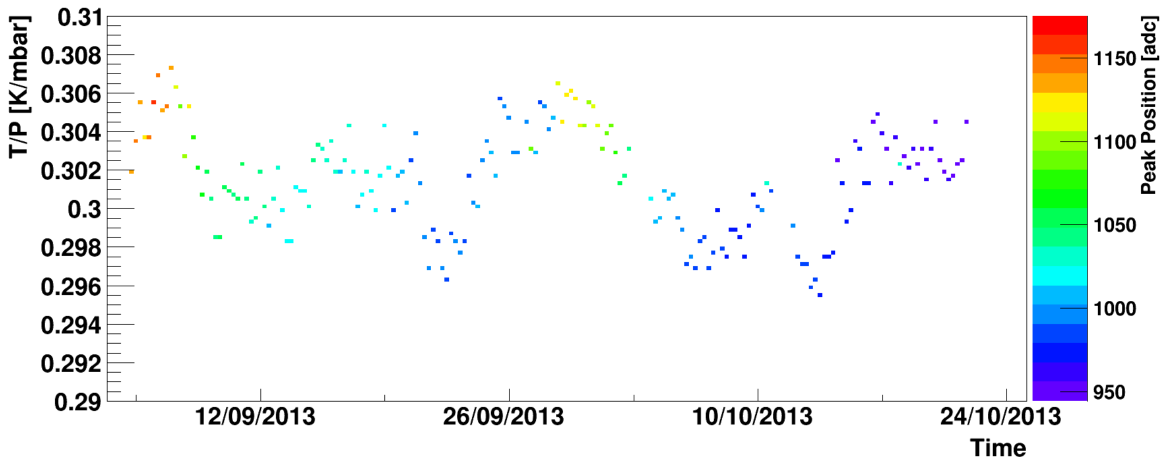

Figure 6.9: Peak Position (uncorrected peak mean (gain) (non normalized)) vs Time and T/P for the periods described in Figure 6.8.

In contrast with Figure 6.7 in Figure 6.9 vertical color stripes are present. By choosing a constant value of T/P and reading horizontally one can observe the decrease of the peak position with respect to time repeated for both periods. This tells us that at any value for T/P the aging trend holds.

6.4. Conclusions

Single wire detectors are highly sensitive to environmental fluctuations such as changes in temperature and pressure. This is due to operation mechanism which uses the gases as an ionizing medium for detection of particles. The gas itself is what is sensitive to these environmental fluctuations. Nevertheless, the effects caused by these can be easily removed in their majority. First of all the fluctuations in a period where the gain seems to be constant can be eliminated in their majority by applying Equations 6.3 and 6.4 respectively. These equations were calculated by searching for the equations that minimize the standard deviation for a period with seemingly constant gain. When no aging effects are present the relation between gain and T/P can be compared to the theoretical model given in Equation 6.6. These are in close agreement since Figures 6.5 and 6.4 have clusters (corresponding to constant gain periods) that have high linear correlations. Finally by adding a time dimension into these plots two very interesting plots are created: Figures 6.7 and 6.9. These completely isolate environmental parameters by analyzing gain trends at specific values of T/P. Since the same trend is found for all values of T/P the conclusions with respect to aging periods or non-aging periods in SW2 hold strongly.

7. Integrated Charge

7.1. Introduction

Integrated charge refers to the accumulation of charge over time in the detector caused by current collection. It effectively describes the usage of a detector in a more absolute way than time. It is of interest to understand how much current a detector can handle before aging settles in rather that how much time has passed. For example, given two identical detectors, if detector 1 is used with a strong radioactive source and detector 2 with a weaker one, detector 1 will show aging first in time. Nevertheless these two detectors will show aging at the same integrated charge.

7.2. Results and Analysis

In contrast with Figure 5.10 the gain is plotted against integrated charge in 7.1.

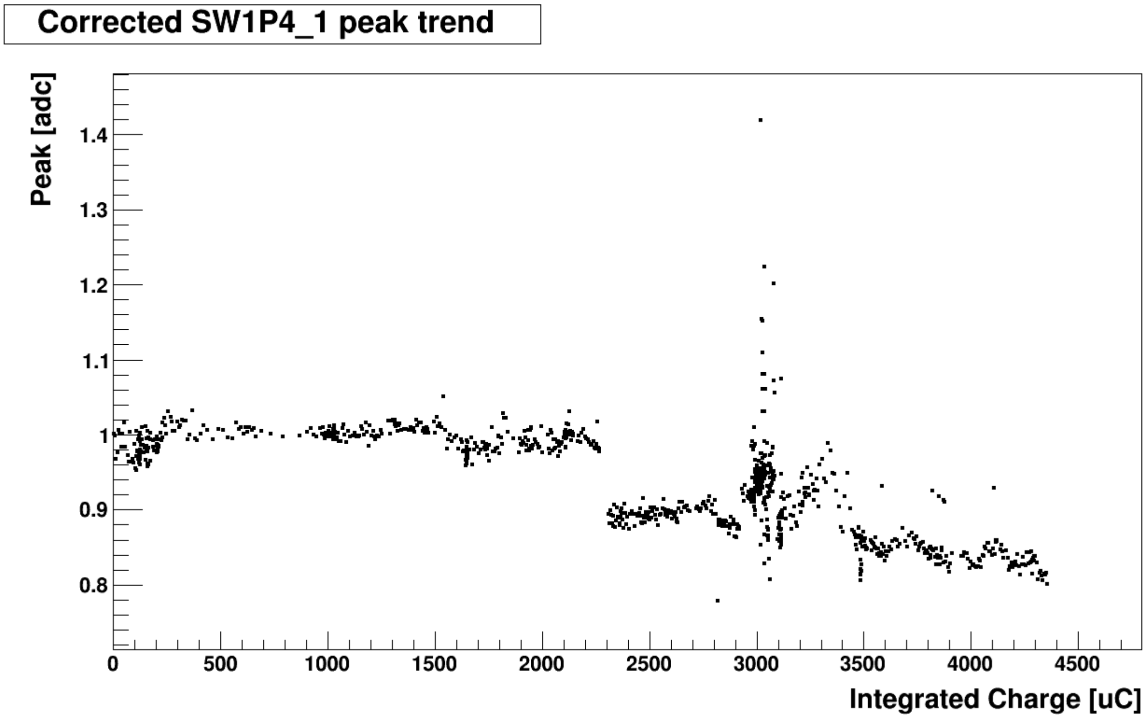

Figure 7.1: Corrected gain vs Integrated Charge for one year of detector operation of SW1

While the y values remain the same, there is a clear distortion in the x-axis since it now represents integrated charge which is a function of time. There is no clear advantage in doing this yet, however by doing the same for periods with aging these advantages become clear. Calculating the integrated charge for the heavily analyzed first two aging periods in SW2 results in Figures 7.2 and 7.3.

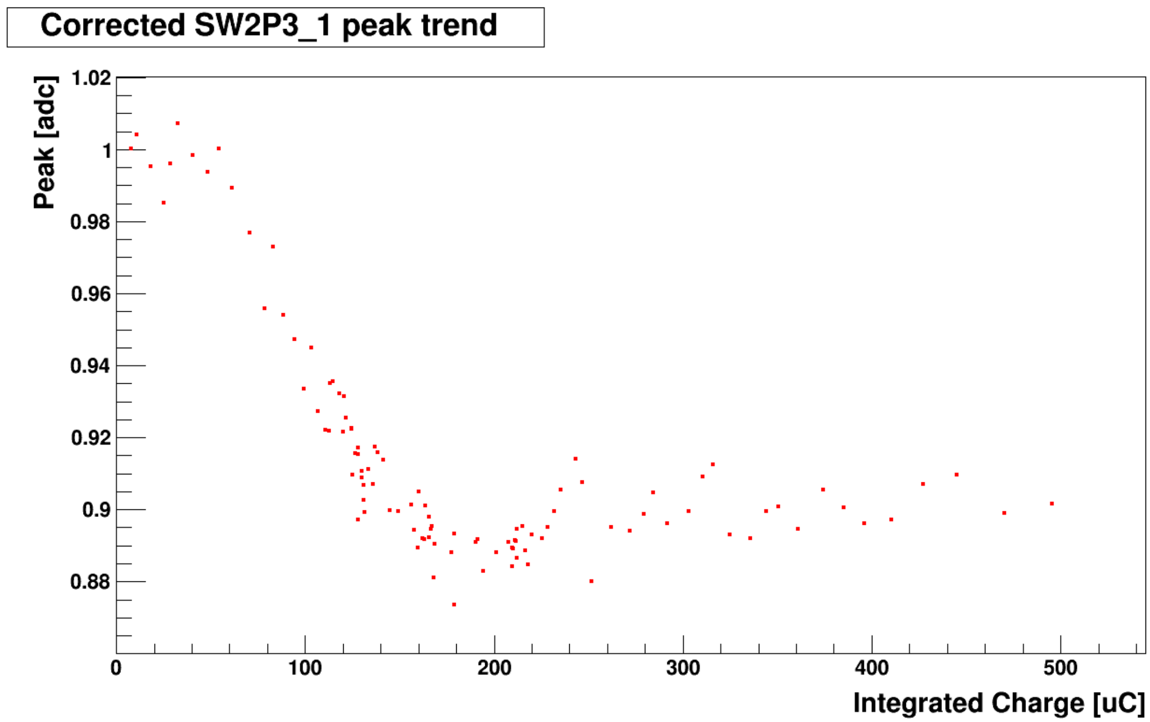

Figure 7.2: Corrected gain vs Integrated Charge for the period between the dates 1 and 2 in subSection3.1.2 for SW2 operation in position 3.

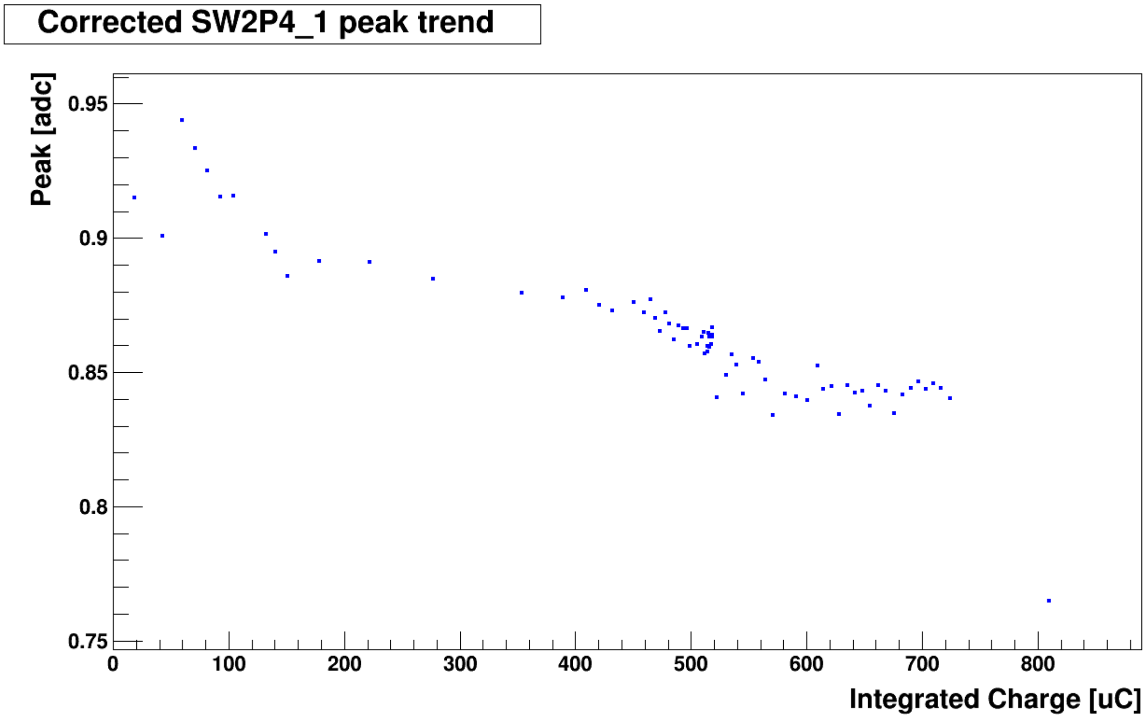

Figure 7.3: Corrected gain vs Integrated Charge for the period between the dates 2 and 3 in subSection 3.1.2 for SW2 operation in position 4.

These two Figures correspond to the periods in Figure 6.8. The difference is dramatic. When the gain is plotted against time the aging process settles in 16 days for the first period (detector in position 3) and 20 days for the second period (detector in position 4). Does this mean the aging process lasts 25% more in position 4? Not exactly. To compute this an absolute comparison parameter is needed. This is where the integrated charge comes in. Referring to Figures 7.2 and 7.3, the 16 days in the first period corresponds to an integrated charge of 200 μC while in period the 20 days correspond to an integrated charge of 600 μC, a 300% increase.

Dividing these integrated charges by the corresponding number of days an average current for each period is obtained: 0.14 nA for the first period and 0.35 nA for the second. Therefore, there are around 2.5 times more particles counted when the detector is in position 4 since the current generated in the wire is proportional to the incoming number of particles.

7.3. Conclusions

Integrated charge is a great tool to characterize the aging in single wire detectors. It is a parameter that allows comparison of aging effects between detectors subject to different radiation sources or a detector thats moved with respect to a single radiation source. Additionally the integrated charge provides information about the current in the wire and the total number of particles detected.

8. General Conclusions

Single wire detectors work on the principle of gas particle ionization and electron avalanche production to detect various types of ionizing radiation. Since they work in the proportional counter regime the energy of the detected particle can be easily extrapolated from the measured current. However, many factors can contribute to a change in the measured current. The gas mixture itself is an essential component of this. Environmental fluctuations, especially in temperature and pressure, can affect the density of the gas mixture in the detector changing the mean free path of collected electrons. These environmental parameters are corrected by using a simple minimization of the standard deviation of the gain fluctuations. By applying these corrections, the detector performances are well understood. Furthermore by carefully analyzing the effects of environmental T/P fluctuations on the gain the precise effects were understood over time. The most important outcome of these tests is the characterization of single wire detectors in terms of aging and environmental effects. The gain increases as T/P increases (giving a positive Townsend coefficient) in a consistent manner over time and that at stable T/P values aging effects are clearly observed. The single wire detector aging effects are due to the deposition of contaminants along the different lengths of the wire. The effects are clearly visible by moving the detector relative to the radioactive source as to expose wire sections that have not been irradiated before since the gas gain is restored. Supplementary to this time trend analysis one can plot all the variables with respect to integrated charge giving a more absolute perspective of the aging in the detectors.

9. Apendix

9.1. AmbSelect.C

```
//trend of enviromental parameter, current, corrected current, peak position
// plot for only ONE single wire -> choose SW from DataPeak line! (or use other script: PeakTrend_v?.C)
// output of enviromental parameter in AmbParCorr.dat

#include <string>
#include <stdlib>
#include <fstream>
#include <iostream>


void AmbSelect(){

    //----------------------------------------

        gROOT->SetStyle("Plain");
        // background is no longer mouse-dropping white
        gStyle->SetCanvasColor(kWhite);
        // blue to red false color palette. Use 9 for b/w
        gStyle->SetPalette(1,0);
        // turn off canvas borders
        gStyle->SetCanvasBorderMode(0);
        gStyle->SetPadBorderMode(0);
        // What precision to put numbers if plotted with "TEXT"
        gStyle->SetPaintTextFormat("5.2f");

        // For publishing:
        gStyle->SetLineWidth(1.5);
        gStyle->SetTextSize(1.1);
        gStyle->SetLabelSize(0.03,"xy");
        gStyle->SetTitleSize(0.04,"xy");
        gStyle->SetTitleOffset(1.1,"x");
        gStyle->SetTitleOffset(0.9,"y");
        gStyle->SetPadTopMargin(0.1);
        gStyle->SetPadRightMargin(0.05);
        gStyle->SetPadBottomMargin(0.1);
        gStyle->SetPadLeftMargin(0.1);

    //---------------------------------------


    //open file
    //fstream check("C:/root/macros/SingleWire/OutputandPlots/check.out",ios::out);
    fstream AmbParCorr("C:/root/macros/SingleWire/OutputandPlots/AmbParCorrfinal.dat",ios::out);
    fstream DataCurrent("C:/root/macros/SingleWire/OutputandPlots/FilterdFineout.dat",ios::in); //merged file of current and enviromental parameters
    fstream DataPeak("C:/root/macros/SingleWire/OutputandPlots/peakpositionfinal_SW1.txt",ios::in); //remember to choose SW1 or SW2!!!

    //file variables
    int xcheck = 1;
    float a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a17,a18,a19;
    float a1new;
    float mean1, mean2, maximumX, FirstValue, integral, integralnoise;
    float convYearPeak, convMonthPeak, convDayPeak, convHourPeak, convMinutePeak, convSecondPeak, a6fPeak, a12fPeak, a15fPeak; //Bea
    float a1f[100000],a2f[100000],a3f[100000],a4f[100000],a5f[100000],a6f[100000],a7f[100000],a8f[100000],a9f[100000],a10f[100000];
    float a11f[100000],a12f[100000],a13f[100000],a14f[100000],a15f[100000],a15f_test[100000],a16f[100000],a17f[100000],a18f[100000],a19f[100000];
    float a1newf[100000];
    float CorrFact[100000];
    float CorrCurr[100000];
    float PeakADC[100000], PeakADCtmp;
    float P0 = 970.; //mbar
    float T0 = 293.; //K

    //time variables
    char cday[30],cmonth[30],cyear[30],chour[30],cmin[30],csec[30];
    int day,month,year,hour,min,sec;
    TDatime t;
    UInt_t ctime;
    int convYear[100000], convMonth[100000], convDay[100000]; //boh
    int convHour[100000], convMinute[100000], convSecond[100000]; //boh
    int pTime;
    float Time, Timef[100000], Timeg[100000];

    //plot variables
    TCanvas *c[100];
    TGraph *gr[100];
    TLegend *Legend;

    char ctitle[30];
    char titlegif[60];
```

```
    char title[60],title2[60],titleday[60];

    //?
    int iplt;
    int i=0;
    int oldHour = 0;

    while(!DataCurrent.eof())
    {
        DataCurrent >> a1new >> a6 >> a7 >> a9 >> a12 >> a15 >> a17 >> a18;

        Timef[i] = a1new;
        t.Set(a1new);
        convYear[i] = t.GetYear(); convMonth[i] = t.GetMonth(); convDay[i] = t.GetDay();
        convHour[i] = t.GetHour(); convMinute[i] = t.GetMinute(); convSecond[i] = t.GetSecond();
        /*      if (i<10) { */
        /*        cout << a1new << "\t" << convYear[i] << "\t" << convMonth[i] << "\t" << convDay[i] << endl; */
        /*        cout << convHour[i] << "\t" << convMinute[i] << "\t" << convSecond[i] << endl; */
        /*      } */
        //T1
        a6f[i] = a6;
        //T2
        a7f[i] =  a7;
        //I (mA)
        a9f[i] =  a9 * 1000.;
        //dewp
        a12f[i] =  7.5*(a12*1000)-90.;
        //Pabs
        a15f[i] =  ((0.075*(1000*a17))-0.3)*1000.;      //Prel
        a17f[i] =  3.125*(a15*1000)-37.5;
        //O2
        a18f[i] = 6.25*(a18*1000)-25;
        //CorrectionFactor
        CorrFact[i]= ((273.0+a7f[i])/T0)*(P0/a15f[i]);
        //CorrectionFactor
        CorrCurr[i]= a9f[i] * CorrFact[i] *  CorrFact[i];
        if (convHour[i] != oldHour && convHour[i]%6==0) //boh
        {
            AmbParCorr << convYear[i] << "\t" << convMonth[i] << "\t" << convDay[i] << "\t" << //boh
            convHour[i] << "\t" << convMinute[i] << "\t" << convSecond[i] << "\t" <<  a6f[i] << "\t" << a15f[i] << "\t" << a18f[i] << "\t" << a12f[i] << endl;
            oldHour = convHour[i];

        }
        i++;
    }

    int npt = i-1;

    //***Start Plots***

    //a6->T1
    iplt=1;
    sprintf(title, "plt[%d]",iplt);
    sprintf(title2, " plt %d",iplt);
    c[iplt] = new TCanvas(title,title2,10,10,1300,800);

    c[iplt]->SetFillColor(0);
    c[iplt]->GetFrame()->SetBorderSize(0);
    gr[iplt] = new TGraph(npt, Timef, a6f);

    gr[iplt]->SetMarkerSize(0.75);
    gr[iplt]->SetMarkerStyle(21);
    gr[iplt]->SetMarkerColor(1);

    gr[iplt]->GetXaxis()->SetLimits(0.9999*(Timef[1]),1.0001*(Timef[npt-1]));

    cout << "time" << Timef[npt-1] << endl;

    gr[iplt]->SetTitle(title);
    gr[iplt]->GetXaxis()->SetTitle("");
    gr[iplt]->GetYaxis()->SetTitle("T1 (C)");

    gr[iplt]->GetXaxis()->SetNdivisions(605);
    gr[iplt]->GetXaxis()->SetTimeDisplay(1);
    gr[iplt]->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
    gr[iplt]->Draw("AP");
    //c[iplt]->Print("/Applications/root/macros/swc256/jpeg/T1.jpg");

    //a7->T2
    iplt=2;
    sprintf(title, "plt[%d]",iplt);
    sprintf(title2, " plt %d",iplt);
    c[iplt] = new TCanvas(title,title2,10,10,1300,800);
```

```
	c[iplt]->SetFillColor(0);
	c[iplt]->GetFrame()->SetBorderSize(0);
	gr[iplt] = new TGraph(npt, Timef, a7f);

	gr[iplt]->SetMarkerSize(0.75);
	gr[iplt]->SetMarkerStyle(21);
	gr[iplt]->SetMarkerColor(1);

	gr[iplt]->GetXaxis()->SetLimits(0.9999*(Timef[1]),1.0001*(Timef[npt-1]));

	gr[iplt]->SetTitle(title);
	gr[iplt]->GetXaxis()->SetTitle("");
	gr[iplt]->GetYaxis()->SetTitle("T2 (C)");

	gr[iplt]->GetXaxis()->SetNdivisions(605);
	gr[iplt]->GetXaxis()->SetTimeDisplay(1);
	gr[iplt]->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
	gr[iplt]->Draw("AP");
	//c[iplt]->Print("/Applications/root/macros/swc256/jpeg/T2.jpg");

	//a9->I
	iplt=3;
	sprintf(title, "plt[%d]",iplt);
	sprintf(title2, " plt %d",iplt);
	c[iplt] = new TCanvas(title,title2,10,10,1300,800);

	c[iplt]->SetFillColor(0);
	c[iplt]->GetFrame()->SetBorderSize(0);
	gr[iplt] = new TGraph(npt, Timef, a9f);

	gr[iplt]->SetMarkerSize(0.75);
	gr[iplt]->SetMarkerStyle(21);
	gr[iplt]->SetMarkerColor(1);

	gr[iplt]->GetXaxis()->SetLimits(0.9999*(Timef[1]),1.0001*(Timef[npt-1]));

	gr[iplt]->SetTitle(title);
	gr[iplt]->GetXaxis()->SetTitle("");
	gr[iplt]->GetYaxis()->SetTitle("I (mA)");

	gr[iplt]->GetXaxis()->SetNdivisions(605);
	gr[iplt]->GetXaxis()->SetTimeDisplay(1);
	gr[iplt]->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
	gr[iplt]->Draw("AP");
	//c[iplt]->Print("/Applications/root/macros/swc256/jpeg/Current.jpg");

	//a12->dewp
	iplt=4;
	sprintf(title, "plt[%d]",iplt);
	sprintf(title2, " plt %d",iplt);
	c[iplt] = new TCanvas(title,title2,10,10,1300,800);

	c[iplt]->SetFillColor(0);
	c[iplt]->GetFrame()->SetBorderSize(0);
	gr[iplt] = new TGraph(npt, Timef, a12f);

	gr[iplt]->SetMarkerSize(0.75);
	gr[iplt]->SetMarkerStyle(21);
	gr[iplt]->SetMarkerColor(1);

	gr[iplt]->GetXaxis()->SetLimits(0.9999*(Timef[1]),1.0001*(Timef[npt-1]));

	gr[iplt]->SetTitle(title);
	gr[iplt]->GetXaxis()->SetTitle("");
	gr[iplt]->GetYaxis()->SetTitle("dewpoint (C)");

	gr[iplt]->GetXaxis()->SetNdivisions(605);
	gr[iplt]->GetXaxis()->SetTimeDisplay(1);
	gr[iplt]->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
	gr[iplt]->Draw("AP");
	//c[iplt]->Print("/Applications/root/macros/swc256/jpeg/dewpoint.jpg");

	//a15->Pabs
	iplt=5;
	sprintf(title, "plt[%d]",iplt);
	sprintf(title2, " plt %d",iplt);
	c[iplt] = new TCanvas(title,title2,10,10,1300,800);

	c[iplt]->SetFillColor(0);
	c[iplt]->GetFrame()->SetBorderSize(0);
	gr[iplt] = new TGraph(npt, Timef, a15f);

	gr[iplt]->SetMarkerSize(0.75);
	gr[iplt]->SetMarkerStyle(21);
```

```
    gr[iplt]->SetMarkerColor(1);

    gr[iplt]->GetXaxis()->SetLimits(0.9999*(Timef[1]),1.0001*(Timef[npt-1]));

    gr[iplt]->SetTitle(title);
    gr[iplt]->GetXaxis()->SetTitle("");
    gr[iplt]->GetYaxis()->SetTitle("Pabs (mbar)");

    gr[iplt]->GetXaxis()->SetNdivisions(605);
    gr[iplt]->GetXaxis()->SetTimeDisplay(1);
    gr[iplt]->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
    gr[iplt]->Draw("AP");
    //c[iplt]->Print("/Applications/root/macros/swc256/jpeg/Pabs.jpg");

    //a17->Prel
    iplt=6;
    sprintf(title, "plt[%d]",iplt);
    sprintf(title2, " plt %d",iplt);
    c[iplt] = new TCanvas(title,title2,10,10,1300,800);

    c[iplt]->SetFillColor(0);
    c[iplt]->GetFrame()->SetBorderSize(0);
    gr[iplt] = new TGraph(npt, Timef, a17f);

    gr[iplt]->SetMarkerSize(0.75);
    gr[iplt]->SetMarkerStyle(21);
    gr[iplt]->SetMarkerColor(1);

    gr[iplt]->GetXaxis()->SetLimits(0.9999*(Timef[1]),1.0001*(Timef[npt-1]));

    gr[iplt]->SetTitle(title);
    gr[iplt]->GetXaxis()->SetTitle("");
    gr[iplt]->GetYaxis()->SetTitle("Prel (mbar)");

    gr[iplt]->GetXaxis()->SetNdivisions(605);
    gr[iplt]->GetXaxis()->SetTimeDisplay(1);
    gr[iplt]->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
    gr[iplt]->Draw("AP");
    //c[iplt]->Print("/Applications/root/macros/swc256/jpeg/Prel.jpg");

    //a18->O2
    iplt=7;
    sprintf(title, "plt[%d]",iplt);
    sprintf(title2, " plt %d",iplt);
    c[iplt] = new TCanvas(title,title2,10,10,1300,800);

    c[iplt]->SetFillColor(0);
    c[iplt]->GetFrame()->SetBorderSize(0);
    gr[iplt] = new TGraph(npt, Timef, a18f);

    gr[iplt]->SetMarkerSize(0.75);
    gr[iplt]->SetMarkerStyle(21);
    gr[iplt]->SetMarkerColor(1);

    gr[iplt]->GetXaxis()->SetLimits(0.9999*(Timef[1]),1.0001*(Timef[npt-1]));

    gr[iplt]->SetTitle(title);
    gr[iplt]->GetXaxis()->SetTitle("");
    gr[iplt]->GetYaxis()->SetTitle("O2 (ppm)");

    gr[iplt]->GetXaxis()->SetNdivisions(605);
    gr[iplt]->GetXaxis()->SetTimeDisplay(1);
    gr[iplt]->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
    gr[iplt]->Draw("AP");
    //c[iplt]->Print("/Applications/root/macros/swc256/jpeg/O2.jpg");

    //CorrFact
    iplt=10;
    sprintf(title, "plt[%d]",iplt);
    sprintf(title2, " plt %d",iplt);
    c[iplt] = new TCanvas(title,title2,10,10,1300,800);

    c[iplt]->SetFillColor(0);
    c[iplt]->GetFrame()->SetBorderSize(0);
    gr[iplt] = new TGraph(npt, Timef, CorrFact);

    gr[iplt]->SetMarkerSize(0.75);
    gr[iplt]->SetMarkerStyle(21);
    gr[iplt]->SetMarkerColor(1);

    gr[iplt]->GetXaxis()->SetLimits(0.9999*(Timef[1]),1.0001*(Timef[npt-1]));

    gr[iplt]->SetTitle(title);
    gr[iplt]->GetXaxis()->SetTitle("");
    gr[iplt]->GetYaxis()->SetTitle("CorrFact[i]");
```

```
    gr[iplt]->GetXaxis()->SetNdivisions(605);
    gr[iplt]->GetXaxis()->SetTimeDisplay(1);
    gr[iplt]->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
    gr[iplt]->Draw("AP");
    //c[iplt]->Print("/Applications/root/macros/swc256/jpeg/CorrFact.jpg");

    //CorrCurrent
    iplt=11;
    sprintf(title, "plt[%d]",iplt);
    sprintf(title2, " plt %d",iplt);
    c[iplt] = new TCanvas(title,title2,10,10,1300,800);

    c[iplt]->SetFillColor(0);
    c[iplt]->GetFrame()->SetBorderSize(0);
    gr[iplt] = new TGraph(npt, Timef, CorrCurr);

    gr[iplt]->SetMarkerSize(0.75);
    gr[iplt]->SetMarkerStyle(21);
    gr[iplt]->SetMarkerColor(1);

    gr[iplt]->GetXaxis()->SetLimits(0.9999*(Timef[1]),1.0001*(Timef[npt-1]));

    gr[iplt]->SetTitle(title);
    gr[iplt]->GetXaxis()->SetTitle("");
    gr[iplt]->GetYaxis()->SetTitle("Corr Curr (mA)");

    gr[iplt]->GetXaxis()->SetNdivisions(605);
    gr[iplt]->GetXaxis()->SetTimeDisplay(1);
    gr[iplt]->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
    gr[iplt]->Draw("AP");
    //c[iplt]->Print("/Applications/root/macros/swc256/jpeg/CorrCurr.jpg");

    //***End Plots***

    //Peak SW1
    int i=0;
    int j=0;
    /*
    AmbParCorr.close();
    fstream AmbParCorr2("/Applications/root/macros/SW/OutputAndPlots/AmbParCorr2.dat",ios::in); //boh
    while(!DataPeak.eof() && !AmbParCorr.eof()){ //Bea

        //DataPeak >> mean1 >> mean2 >> maximumX >> hour >> day >> month >> year; //old settings
        DataPeak >> mean1 >> mean2 >> maximumX >> integral >> integralnoise >> hour >> day >> month >> year; //new settings
        year= 2000 + year; //spostata qui da sotto

        if (i==0) {
                AmbParCorr2 >> convYearPeak >> convMonthPeak >> convDayPeak >> convHourPeak >> convMinutePeak >> convSecondPeak >> a6fPeak >> a12fPeak >> a15fPeak; //Bea
        cout <<     "ambi " << convYearPeak <<"\t"<< convMonthPeak  <<"\t"<< convDayPeak <<"\t"<< convHourPeak <<"\t"<< endl ;
        }

        if (j==1) {

        //q-if (convMonthPeak != month && convDayPeak != day && convHourPeak != hour) {
            while (convMonthPeak != month || convDayPeak != day || convHourPeak != hour) {
                AmbParCorr2 >> convYearPeak >> convMonthPeak >> convDayPeak >> convHourPeak >> convMinutePeak >> convSecondPeak >> a6fPeak >> a12fPeak >> a15fPeak; //Bea
            }

        cout << "peakw " << year <<"\t"<< month <<"\t" << day <<"\t"<< hour << endl ;
        cout << "ambiw " << convYearPeak <<"\t"<< convMonthPeak  <<"\t"<< convDayPeak <<"\t"<< convHourPeak << endl ;
        cout << endl;

        //CorrectionFactor
        CorrFact[i]= ((273.0+a6fPeak)/T0)*(P0/(-3.3*a15fPeak));

        PeakADCtmp = maximumX;
        maximumX = maximumX;// / (CorrFact[i] * CorrFact[i] * CorrFact[i] * CorrFact[i] * CorrFact[i]);

        cout <<  maximumX << "\t" << PeakADCtmp << "\t" << CorrFact[i] << "\t" << a6fPeak << "\t" << a15fPeak << endl;

        //q-}
        //
        }

        //year= 2000 + year; spostata sopra
        TDatime * date = new TDatime(year, month, day, hour, min, sec);
        UInt_t ttt;
        ttt = date->Convert();
```

```
        pTime = (int)ttt;
        Timeg[i] = (float)pTime;

        if (i==0) {
            FirstValue = maximumX; //define the first value for the normalization
            PeakADC[i]=maximumX/FirstValue;
            //cout << i << "\t" << PeakADC[i] << endl;
        }
        if (i>0) {
            PeakADC[i] = maximumX/FirstValue;
        }
        i++;
    }

    int nptP = i-1;

    //PeakPlot SW1
    iplt=12;
    sprintf(title, "plt[%d]",iplt);
    sprintf(title2, " plt %d",iplt);
    c[iplt] = new TCanvas(title,title2,10,10,1300,800);

    c[iplt]->SetFillColor(0);
    c[iplt]->GetFrame()->SetBorderSize(0);
    gr[iplt] = new TGraph(nptP, Timeg, PeakADC);

    gr[iplt]->SetMarkerSize(0.75);
    gr[iplt]->SetMarkerStyle(21);
    gr[iplt]->SetMarkerColor(1);

    gr[iplt]->SetTitle(title);
    gr[iplt]->GetXaxis()->SetTitle("");
    gr[iplt]->GetYaxis()->SetTitle("Peak(adc)");

    gr[iplt]->GetXaxis()->SetNdivisions(605);
    gr[iplt]->GetXaxis()->SetTimeDisplay(1);
    gr[iplt]->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
    gr[iplt]->Draw("AP");


    //Time division lines

    year = 2013;
    month = 10;
    day = 31;
    hour = 0; min = 0; sec = 0;
    TDatime * date = new TDatime(year, month, day, hour, min, sec);
    UInt_t ttt;
    ttt = date->Convert();
    pTime = (int)ttt;
    float Time1 = (float)pTime;

    c[iplt]->Update();
    TLine *l=new TLine(Time1,c[iplt]->GetUymin(),Time1,c[iplt]->GetUymax());
    l->SetLineColor(32);
    l->SetLineWidth(2);
    l->SetLineStyle(5);
    l->Draw();

    //Finish time line division

    c[iplt]->Print("/Applications/root/macros/SW/OutputAndPlots/Peak_adc.jpg");
    */
}
```

9.2. chisqtest.C (Lines 149-205, 287-367 [19])

```
#include <fstream>
bool eof();

void chisqtestTP()
{
//-----------------------------------------

gROOT->SetStyle("Plain");
  // background is no longer mouse-dropping white
  gStyle->SetCanvasColor(kWhite);
  // blue to red false color palette. Use 9 for b/w
  gStyle->SetPalette(1,0);
  // turn off canvas borders
  gStyle->SetCanvasBorderMode(0);
  gStyle->SetPadBorderMode(0);
  // What precision to put numbers if plotted with "TEXT"
  gStyle->SetPaintTextFormat("5.2f");

  // For publishing:
  gStyle->SetLineWidth(1.5);
  gStyle->SetTextSize(1.1);
  gStyle->SetLabelSize(0.05,"xy");
  gStyle->SetTitleSize(0.05,"xy");
  gStyle->SetTitleOffset(1.1,"x");
  gStyle->SetTitleOffset(0.9,"y");
  gStyle->SetPadTopMargin(0.1);
  gStyle->SetPadRightMargin(0.1);
  gStyle->SetPadBottomMargin(0.16);
  gStyle->SetPadLeftMargin(0.12);

//-----------------------------------------

 int SingleWire;
 cout << "SingleWire 1 or 2?    ";
 cin >> SingleWire;
 if (SingleWire==1)
 {
  //fstream alphabeta("C:/root/macros/SingleWire/OutputandPlots/alphabetaSW1.dat", ios::out );
  //fstream filesused("C:/root/macros/SingleWire/OutputandPlots/FilesUsedSW1.dat", ios::out );
  //fstream alphabeta1("C:/root/macros/SingleWire/OutputandPlots/alphabeta1SW1.dat", ios::out );
  //fstream gaincorr("C:/root/macros/SingleWire/OutputandPlots/gaincorrSW1.dat", ios::out );
  //ifstream ftime("/home/daqrpc/daq-1.0.0/Scan_v0/Backup_pccms2/CAENHVWrapper_2_4/RPCtest/ftime_thp.tmp0", ios::in);

  ifstream fgain1("C:/root/macros/SingleWire/OutputandPlots/peakpositionfinal_SW1clean.txt", ios::in);//These two have to be the same file
  ifstream fgainP("C:/root/macros/SingleWire/OutputandPlots/peakpositionfinal_SW1clean.txt", ios::in);//These two have to be the same file
 }
 if (SingleWire==2)
 {
  //fstream alphabeta("C:/root/macros/SingleWire/OutputandPlots/alphabetaSW2.dat", ios::out );
  //fstream filesused("C:/root/macros/SingleWire/OutputandPlots/FilesUsedSW2.dat", ios::out );
  //fstream alphabeta1("C:/root/macros/SingleWire/OutputandPlots/alphabeta1SW2.dat", ios::out );
  //fstream gaincorr("C:/root/macros/SingleWire/OutputandPlots/gaincorrSW2.dat", ios::out );
  //ifstream ftime("/home/daqrpc/daq-1.0.0/Scan_v0/Backup_pccms2/CAENHVWrapper_2_4/RPCtest/ftime_thp.tmp0", ios::in);

  ifstream fgain1("C:/root/macros/SingleWire/OutputandPlots/peakpositionfinal_SW2clean.txt", ios::in);//These two have to be the same file
  ifstream fgainP("C:/root/macros/SingleWire/OutputandPlots/peakpositionfinal_SW2clean.txt", ios::in);//These two have to be the same file
 }
 if (SingleWire!=1 && SingleWire!=2) {cout<< "Selection has to be 1 or 2" << endl;}

  ifstream famb("C:/root/macros/SingleWire/OutputandPlots/AmbParCorrfinal.dat", ios::in);
  //ifstream fhv[50];


  int i;
  //int startday,stopday;
  //float iMax;
  float Avegain1[50];
  //Int_t idx[50], tmpidx;
  int idx;
  char title[256];

  TCanvas *c[50];
  TGraph *grc[50], *grcc[50];
  TGraph *grhv[50];

  float FWHM[10000];
  float gain1[10000],hv[50][10000]; //gain1[50][10000]
  float dt[10000];
  int dt2[10000],dt_month[10000],dt_day[10000],dt_year[10000],dt_hour[10000],dt_min,dt_sec;
```

```
  //Int_t pTime[10000];
  float Time[10000];

  float Tin[10000],Patm[10000], dewP[10000], O2[10000];
  float Tsinc[10000], Psinc[10000];

  float T0=0;
  float P0=0;
  float Tcorr, Pcorr; //RHcorr;
  Int_t idx_T0, idx_P0;
  Int_t alpha, beta, alphaMax, betaMax;
  Int_t best_alpha, best_beta;

  float sigma[50][50]; //alpha, beta, gap
  float minsigma; //alpha, beta, gap
  float gain1_corr[50][50][10000]; //alpha, beta, gap, time
  //float gain1_i2_corr[50][50][10000]; //alpha, beta, gap, time
  float Avegain1_corr[50][50]; //alpha, beta, gap
  float gain1_bestcorr[50][10000]; //best correction

  for(int filli=0; filli<50; filli++)
  {
      for(int fillj=0; fillj<50; fillj++)
      {
          Avegain1_corr[filli][fillj]=0;
          sigma[filli][fillj]=0;
      }
  }


  //Int_t iday_i1, iday_i2;

  //TCanvas *c[50];
  //TGraph *grc[50];

  //float Tmin, TMax, Patmmin, PatmMax;

  //TF1 *fGraph[50];
  //Float_t nP0_reg1, nP0E_reg1;

  //Int_t ITimeZona1, ITimeZona2, ITimeZona3;

  //Tmin= 17;
  //TMax = 25;
  //Patmmin = 930;
  //PatmMax = 1000;
  //iMax = 60;

  alphaMax = 49;
  betaMax = 49;

  i = 0;
  while(!famb.eof())
  {
    famb >> dt_year[i] >> dt_month[i] >> dt_day[i] >> dt_hour[i] >> dt_min >> dt_sec >> Tin[i] >> Patm[i] >> O2[i] >> dewP[i];

    //if (i<150) cout<<  dt_year[i] <<endl;

    //dt[i] = float(dt2[i]);

    //check << i << "\t" << dt[i] << endl;
    /*
    TDatime * date = new TDatime(dt_year[i], dt_month[i], dt_day[i], dt_hour[i], dt_min, dt_sec);
    UInt_t ttt;
    ttt = date->Convert();
    pTime[i] = (int)ttt;
    Time[i] = (float)pTime[i];
    */
    //THP SHOULD THIS BE INCLUDED????????
    Patm[i] = Patm[i] + 4.0; //offset sensore GIF (to be checked!)
    if(Tin[i] < 10)
    {
      Tin[i] = 10 ;
      //RHin[i]= 30 ;
      //Tout[i]= 10 ;
      //RHout[i]= 30;
      Patm[i] = 1000;
    }

    //T0, RH0, P0, ... average value
    if(Tin[i]>10)
    {
      idx_T0++;
      T0 = T0 + Tin[i];
```

```
  }
  if(Patm[i]>900)
  {
   idx_P0++;
   P0 = P0 + Patm[i];
  }

  //fine THP
  //check << dt[i] << " " << dt_month <<" " << dt_day << endl;
  //if(i==0) startday = Time[i];//dt[i];
  //if(i>0) stopday = Time[i-1];//dt[i-1];
  i++;
 }


 T0 = T0 / idx_T0;
 P0 = P0 / idx_P0;
 cout << T0 << "\t" << P0 << endl;

/*
 int days = i;
 check << "start stop day "
  << startday << "\t" << stopday << "\t" << days << "\n";

 for(int k = 0; k <= days; k++)
 {
  sPatm[k] = (((Patm[k]-Patmmin)/(PatmMax-Patmmin))*(TMax-Tmin))+Tmin;
 }

 int i1,i2;

 //  for(int ichn=1;ichn<12;ichn++){ //era <8 comm.10/03/2011
for(int ichn=4;ichn<5;ichn++)
 { //era <8
  i1 = 2*ichn - 1;
  i2 = 2*ichn;

  if(ichn == 9){
   i1 = 17;
   i2 = 20;
  }
*/
  i = 0;
  int j0 = 0; //
  int iday;

  float trash;
  float firstgain;
  int checkhour, checkday, checkmonth, checkyear;
  int yearP, monthP, dayP, hourP, minP, secP;
  float periodstart, periodend, periodday;
  int pass;
  int nused=0;
  idx = 0;

  // *********************************************************************MaiN loop needed for chi squared reduction to find alpha and beta***************************************************
  //I removed the indices [i1] to uncomplicate things... basically extracting this loop from the one it is in (line 221)

  while(!fgain1.eof())
  {
    j0=0;
    fgain1 >> trash >> trash >> gain1[i] >> trash >> trash >> FWHM[i] >> trash >> trash >> trash >> trash >> checkhour >> checkday >> checkmonth >> checkyear;
    checkyear=2000+checkyear;

    //if(i==0) idx = 0;
    if (i==0)
    {
      firstgain=gain1[i];
      gain1[i]=gain1[i]/firstgain;
    }
    if (i>0) gain1[i]=gain1[i]/firstgain;
    //cout<< gain1[i] << endl;
    //if(i>20) break;

    while(checkhour!=dt_hour[j0] || checkday!=dt_day[j0] || checkmonth!=dt_month[j0] || checkyear!=dt_year[j0] )
    {
      j0++;
      if(j0>9999)
      {
        pass=0;
        break;
```

```
            }
            else pass=1;
        }
        if(checkhour==dt_hour[j0] && checkday!=dt_day[j0] && checkmonth==dt_month[j0] && checkyear==dt_year[j0]) pass=1;



        //**************************************************Period selector****************************************************
        yearP = 2013;
        monthP = 10;
        dayP = 1;
        hourP = 0; minP = 0; secP = 0;
        TDatime * dateP = new TDatime(yearP, monthP, dayP, hourP, minP, secP);
        UInt_t ttt;
        ttt = dateP->Convert();
        pTimeP = (int)ttt;
        periodstart = (float)pTimeP;

        yearP = 2013;
        monthP = 11;
        dayP = 1;
        hourP = 0; minP = 0; secP = 0;
        TDatime * dateP = new TDatime(yearP, monthP, dayP, hourP, minP, secP);
        UInt_t ttt;
        ttt = dateP->Convert();
        pTimeP = (int)ttt;
        periodend = (float)pTimeP;

        yearP = checkyear;
        monthP = checkmonth;
        dayP = checkday;
        hourP = 0; minP = 0; secP = 0;
        TDatime * dateP = new TDatime(yearP, monthP, dayP, hourP, minP, secP);
        UInt_t ttt;
        ttt = dateP->Convert();
        pTimeP = (int)ttt;
        periodday = (float)pTimeP;

        if(periodday<periodstart || periodday>periodend) pass=0;

//************************************************************************************************************************
        if (pass==1)
        {
            for(int alpha=0; alpha< alphaMax; alpha++)
            {
                for(int beta=0; beta< betaMax; beta++)
                {
                    Tcorr = pow(((Tin[j0]+273.1)/(T0+273.1)),-0.1*alpha);
                    Pcorr = pow((P0/Patm[j0]),-0.1*beta);
                    gain1_corr[alpha][beta][nused] = Tcorr * Pcorr * gain1[i];
                    //cout<< gain1[i] << "        " << gain1_corr[alpha][beta][i] << endl;
                    if(alpha == 0 && beta == 0) idx++;
                    //Avegain1_corr[alpha][beta]++;
                    Avegain1_corr[alpha][beta] = Avegain1_corr[alpha][beta] + gain1_corr[alpha][beta][nused];
                    //cout << i << "     " << alpha << "     " << beta << "     " << Avegain1_corr[alpha][beta] << "    " << gain1_corr[alpha][beta][nused]<<  "    " << idx << endl;
                }
            }
            //filesused << i << "\t" <<nused << "\t" << checkyear << "\t" << checkmonth << "\t" << checkday << "\t" << checkhour << endl;
            nused++; //i++;
            //j0++;
            //if (i>2) break;
        }
        i++;
    }

    iday = nused - 1;
    //
    //stopday = Time[iday_i1];


    for(int alpha=0; alpha< alphaMax; alpha++)
    {
        for(int beta=0; beta< betaMax; beta++)
        {
            Avegain1_corr[alpha][beta] = Avegain1_corr[alpha][beta] / idx;
            //cout << i << "     " << alpha << "     " << beta << "     " << Avegain1_corr[alpha][beta] << endl;
        }
    }

```

```
    for(int j=0; j < iday; j++)
    {
        for(int alpha=0; alpha< alphaMax; alpha++)
        {
            for(int beta=0; beta< betaMax; beta++)
            {
                if(gain1[j] > 0)
                sigma[alpha][beta] = sigma[alpha][beta] + pow((gain1_corr[alpha][beta][j] - Avegain1_corr[alpha][beta]) , 2);
                //cout << iday << "     " << alpha << "     " << beta << "     " << sigma[alpha][beta] << endl;

            }
        }
    }




    for(int alpha=0; alpha< alphaMax; alpha++)
    {
        for(int beta=0; beta< betaMax; beta++)
        {
            sigma[alpha][beta] = sigma[alpha][beta]/ (iday-1);
            sigma[alpha][beta] = pow(sigma[alpha][beta], 0.5);
            //alphabeta << alpha << "\t" <<  beta << "\t" << sigma[alpha][beta] << endl;

            if(alpha == 0 && beta == 0)
            {
                minsigma = sigma[alpha][beta];
                best_alpha = alpha;
                best_beta = beta;
            }

            //alphabeta1 << alpha << "\t" <<  beta << "\t" << sigma[alpha][beta] << "\t" << minsigma<< "\t" << endl;

            if(sigma[alpha][beta] < minsigma)
            {
                best_alpha = alpha;
                best_beta = beta;
                minsigma = sigma[alpha][beta];

            }
        }
    }
    cout << best_alpha << "      " << best_beta <<endl;

    float gain_corrP;
    int iP=0;
    float gainP[10000];
    float gainTP[10000], gainTP1[10000], gainTP2[10000], gainTP3[10000], gainTPall[10000];
    float TP[10000], TP1[10000], TP2[10000], TP3[10000], TPall[10000];
    float timeTP[10000];
    nused=0;
    int selectedperiod;
    int tp=0;
    int tp1=0;
    int tp2=0;
    int tp3=0;
    int year, month, day, hour, min, sec;
    TH2F *TPPeak;
    TPPeak = new TH2F("TPpeak","",50,0.29,0.31,50,850,1400);
    TH2F *TPPeakall;
    TPPeakall = new TH2F("TPpeak","",25,0.29,0.31,25,850,1400);
    TH2F *TPtimeG;
    TPtimeG = new TH2F("TPG","",1620,1375000000,1410000000,100,0.29,0.31);


    //Periods for gain vs T/P plots

  float Time1[7];
  year = 2013;
  month = 10;
  day = 1;
  hour = 0; min = 0; sec = 0;
  TDatime * date = new TDatime(year, month, day, hour, min, sec);
  UInt_t ttt;
  ttt = date->Convert();
  pTime = (int)ttt;
  Time1[0] = (float)pTime;

  year = 2013;
  month = 11;
  day = 1;
```

```
  hour = 0; min = 0; sec = 0;
  TDatime * date = new TDatime(year, month, day, hour, min, sec);
  UInt_t ttt;
  ttt = date->Convert();
  pTime = (int)ttt;
  Time1[1] = (float)pTime;

  year = 2013;
  month = 11;
  day = 21;
  hour = 0; min = 0; sec = 0;
  TDatime * date = new TDatime(year, month, day, hour, min, sec);
  UInt_t ttt;
  ttt = date->Convert();
  pTime = (int)ttt;
  Time1[2] = (float)pTime;

  year = 2014;
  month = 1;
  day = 7;
  hour = 0; min = 0; sec = 0;
  TDatime * date = new TDatime(year, month, day, hour, min, sec);
  UInt_t ttt;
  ttt = date->Convert();
  pTime = (int)ttt;
  Time1[3] = (float)pTime;

  year = 2014;
  month = 2;
  day = 27;
  hour = 0; min = 0; sec = 0;
  TDatime * date = new TDatime(year, month, day, hour, min, sec);
  UInt_t ttt;
  ttt = date->Convert();
  pTime = (int)ttt;
  Time1[4] = (float)pTime;

  year = 2014;
  month = 4;
  day = 28;
  hour = 0; min = 0; sec = 0;
  TDatime * date = new TDatime(year, month, day, hour, min, sec);
  UInt_t ttt;
  ttt = date->Convert();
  pTime = (int)ttt;
  Time1[5] = (float)pTime;

  year = 2014;
  month = 5;
  day = 28;
  hour = 0; min = 0; sec = 0;
  TDatime * date = new TDatime(year, month, day, hour, min, sec);
  UInt_t ttt;
  ttt = date->Convert();
  pTime = (int)ttt;
  Time1[6] = (float)pTime;

  //*****************************************************************************
    //Print corrected gain alongside uncorrected.
    while(!fgainP.eof())
    {
        j0=0;
        fgainP >> trash >> trash >> gainP[iP] >> trash >> trash >> FWHM[iP] >> trash >> trash >> trash >> trash >> checkhour >> checkday >> checkmonth >> checkyear;
        checkyear=2000+checkyear;
        while(checkhour!=dt_hour[j0] || checkday!=dt_day[j0] || checkmonth!=dt_month[j0] || checkyear!=dt_year[j0] )
        {
            j0++;
            if(j0>9999)
            {
                pass=0;
                break;
            }
            else pass=1;
        }
        if(checkhour==dt_hour[j0] && checkday!=dt_day[j0] && checkmonth==dt_month[j0] && checkyear==dt_year[j0]) pass=1;

        yearP = checkyear;
        monthP = checkmonth;
        dayP = checkday;
        hourP = 0; minP = 0; secP = 0;
        TDatime * dateP = new TDatime(yearP, monthP, dayP, hourP, minP, secP);
        UInt_t ttt;
```

```
        ttt = dateP->Convert();
        pTimeP = (int)ttt;
        periodday = (float)pTimeP;

        if (pass==1)
        {
            TPall[nused] = (Tin[j0]+273.1)/Patm[j0];
            gainTPall[nused] = gainP[iP];
            TPPeakall->Fill(TPall[nused], gainTPall[nused]);
            if(periodday>periodstart && periodday<periodend)
            {
                selectedperiod=1;
            }
            else selectedperiod=0;
            // peakpos vs T/P plot
            if(periodday>Time1[0] && periodday<Time1[1])
            //if(periodday>Time1[6])
            {
                TP1[tp1] = (Tin[j0]+273.1)/Patm[j0];
                gainTP1[tp1] = gainP[iP];
                tp1++;
                TP[tp] = (Tin[j0]+273.1)/Patm[j0];
                gainTP[tp] = gainP[iP];
                timeTP[tp] = periodday;
                TPPeak->Fill(TP[tp], gainTP[tp]);
                int gn=0;
                while(gn<gainTP[tp])
                {
                    TPtimeG->Fill(timeTP[tp],TP[tp]);
                    gn++;
                }
                tp++;
            }

            if(periodday>Time1[2] && periodday<Time1[3])
            {
                TP2[tp2] = (Tin[j0]+273.1)/Patm[j0];
                gainTP2[tp2] = gainP[iP];
                tp2++;
                TP[tp] = (Tin[j0]+273.1)/Patm[j0];
                gainTP[tp] = gainP[iP];
                timeTP[tp] = periodday;
                TPPeak->Fill(TP[tp], gainTP[tp]);
                int gn=0;
                while(gn<gainTP[tp])
                {
                    TPtimeG->Fill(timeTP[tp],TP[tp]);
                    gn++;
                }
                tp++;
            }

            if(periodday>Time1[6])
            {
                TP3[tp3] = (Tin[j0]+273.1)/Patm[j0];
                gainTP3[tp3] = gainP[iP];
                tp3++;
                TP[tp] = (Tin[j0]+273.1)/Patm[j0];
                gainTP[tp] = gainP[iP];
                timeTP[tp] = periodday;
                TPPeak->Fill(TP[tp], gainTP[tp]);
                int gn=0;
                while(gn<gainTP[tp])
                {
                    TPtimeG->Fill(timeTP[tp],TP[tp]);
                    gn++;
                }
                tp++;
            }

            //
            Tcorr = pow(((Tin[j0]+273.1)/(T0+273.1)),-0.1*best_alpha);
            Pcorr = pow((P0/Patm[j0]),-0.1*best_beta);
            gain_corrP = Tcorr * Pcorr * gainP[iP];

            //gaincorr << nused << "\t" << checkyear << "\t" << checkmonth << "\t" << checkday << "\t" << checkhour << "\t" << gainP[iP] << "\t" << gain_corrP << "\t" << FWHM[iP] << "\t" << selectedperiod <<endl;
            nused++;
        }
        iP++;
    }
    cout << tp << "     " << tp1 << "     " << tp2 << "     " << tp3 << endl;


    //****************************************************************END MAIN
```

```
LOOP***********************************************************************************************

 //Canvas1 = new TCanvas("Gain vs T/P","Gain vs T/P",10,10,1300,800);
 TPplot1 = new TGraph(tp1-1,TP1, gainTP1);
 TPplot1->SetMarkerSize(0.6);
 TPplot1->SetMarkerStyle(21);
 TPplot1->SetMarkerColor(2);
 TPplot1->SetTitle("7/1/14 - 27/2/14");

 TPplot2 = new TGraph(tp2-1,TP2, gainTP2);
 TPplot2->SetMarkerSize(0.6);
 TPplot2->SetMarkerStyle(21);
 TPplot2->SetMarkerColor(3);
 TPplot2->SetTitle("28/5/14 - 14/7/14");

 TPplot3 = new TGraph(tp3-1,TP3, gainTP3);
 TPplot3->SetMarkerSize(0.6);
 TPplot3->SetMarkerStyle(21);
 TPplot3->SetMarkerColor(4);
 TPplot3->SetTitle("28/5/14 - 14/7/14");

Canvas1 = new TCanvas("Gain vs T/P","Gain vs T/P",10,10,1600,800);
TPplot = new TMultiGraph("Gain vs T/P", "Gain vs T/P");
TPplot->Add(TPplot1);
TPplot->Add(TPplot2);
TPplot->Add(TPplot3);
TPplot->Draw("AP");
TPplot->SetTitle("Gain vs T/P ");
TPplot->GetXaxis()->SetTitle("T/P [K/mbar]");
TPplot->GetYaxis()->SetTitle("Peak [adc]");
TPplot->Draw("AP");
Canvas1->BuildLegend(0.75);
/*
Canvas2 = new TCanvas("T/P vs time","T/P vs time",10,10,1300,800);
TPTime = new TGraph(tp-1, timeTP, TP);
 //Peak position
TPTime->SetMarkerSize(0.6);
TPTime->SetMarkerStyle(21);
TPTime->SetMarkerColor(1);
TPTime->SetTitle("T/P vs time");
TPTime->GetXaxis()->SetTitle("Time");
TPTime->GetYaxis()->SetTitle("T/P [K/mbar]");
//TPTime->GetXaxis()->SetNdivisions(605);
//TPTime->GetXaxis()->SetTimeDisplay(1);
//TPTime->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
TPTime->SetMinimum(0.28);
TPTime->SetMaximum(0.32);
TPTime->Draw("AP");
*/
Canvas3 = new TCanvas("T/P vs time & gain","T/P vs time & gain",10,10,1300,800);
TPTimeG = new TGraph2D(tp-1, timeTP, TP, gainTP);
 //Peak position
TPTimeG->SetMarkerSize(0.6);
TPTimeG->SetMarkerStyle(21);
TPTimeG->SetMarkerColor(1);
TPTimeG->SetTitle("T/P vs time");
TPTimeG->GetXaxis()->SetTitle("Time");
TPTimeG->GetYaxis()->SetTitle("T/P [K/mbar]");
TPTimeG->GetZaxis()->SetTitle("Peak [adc]");
TPTimeG->GetXaxis()->SetNdivisions(605);
TPTimeG->GetYaxis()->SetNdivisions(605);
TPTimeG->GetXaxis()->SetTimeDisplay(1);
TPTimeG->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
//TPTimeG->SetMinimum(0.28);
//TPTimeG->SetMaximum(0.32);
TPTimeG->Draw("P");

Canvas4 = new TCanvas("TP vs peak all","TP vs peak all",10,10,1300,800);
gStyle->SetOptStat(000000000);
//gStyle->SetPalette()
TPPeakall -> Draw("BOX");

Canvas5 = new TCanvas("TP vs peak","TP vs peak",10,10,1300,800);
gStyle->SetOptStat(000000000);
//gStyle->SetPalette()
TPPeak -> Draw("BOX");

Canvas6 = new TCanvas("TP vs time C","TP vs time C",10,10,1300,800);
gStyle->SetOptStat(000000000);
//gStyle->SetPalette()
TPtimeG -> Draw("COLZ");

 }
```

```
//  exit(0);
```

9.3. FineoutFilter.C

```
//trend of enviromental parameter, current, corrected current, peak position
// plot for only ONE single wire -> choose SW from DataPeak line! (or use other script: PeakTrend_v?.C)
// output of enviromental parameter in AmbParCorr.dat

#include <string>
#include <stdlib>
#include <fstream>
#include <iostream>


void FineoutFilter()
{

    //open file

    fstream filtered("C:/root/macros/SingleWire/OutputandPlots/FilterdFineout.dat",ios::out);
    fstream DataCurrent("C:/root/macros/SingleWire/fineout45.out",ios::in); //merged file of current and enviromental parameters
    int i=0;
    int j=0;
    int r=0;
    int f=0;
    int pass=0;
    int time[100000];
    float a6[100000], a7[100000], a9[100000], a12[100000], a15[100000], a17[100000], a18[100000];
    int time1[100000];
    float a61[100000], a71[100000], a91[100000], a121[100000], a151[100000], a171[100000], a181[100000];
    while(!DataCurrent.eof())
    {
        DataCurrent >> time[i] >> a6[i] >> a7[i] >> a9[i] >> a12[i] >> a15[i] >> a17[i] >> a18[i];
        //filtered <<  time[i] << "\t" << a6[i] << "\t" << a7[i] << "\t" << a12[i] << "\t"<< a12[i] << "\t" << a12[i] <<  "\t" << a15[i]  << "\t" << a17[i] << "\t" << a18[i] << endl;
        i++;
    }
    j=i;
    while(j>=0)
    {
        //cout << j << endl;
        for(r=1; r<=j; r++)
        {
            //cout << r << endl;
            if(time[j] != time[j-r]) pass=1;
            else
            {
                pass=0;
                break;
            }
        }
        if(pass == 1)
        {
            time1[f] = time[j];
            a61[f] = a6[j];
            a71[f] = a7[j];
            a91[f] = a9[j];
            a121[f] = a12[j];
            a151[f] = a15[j];
            a171[f] = a17[j];
            a181[f] = a18[j];
            f++;
        }
        j--;
    }
    int k = f;
    while(k>=0)
    {
        filtered <<  time1[k] << "\t" << a61[k] << "\t" << a71[k] << "\t" << a91[k] << "\t" << a121[k] <<  "\t" << a151[k]  << "\t" << a171[k] << "\t" << a181[k] << endl;
        k--;
    }



}
```

9.4. IntegratedCurrent.C

```
//trend of enviromental parameter, current, corrected current, peak position
// plot for only ONE single wire -> choose SW from DataPeak line! (or use other script: PeakTrend_v?.C)
// output of enviromental parameter in AmbParCorr.dat

#include <string>
#include <stdlib>
#include <fstream>
#include <iostream>


void IntegratedCurrent(){

    //open file
    //fstream check("C:/root/macros/SingleWire/OutputandPlots/check.out",ios::out);
    fstream IntCurrentAll("C:/root/macros/SingleWire/OutputandPlots/IntCurrentAll.dat",ios::out);
    fstream IntCurrentP("C:/root/macros/SingleWire/OutputandPlots/IntCurrentSW2P4_1.dat",ios::out);
    fstream DataCurrent("C:/root/macros/SingleWire/OutputandPlots/FilterdFineout.dat",ios::in); //merged file of current and enviromental parameters
    //fstream DataPeak("C:/root/macros/SingleWire/OutputandPlots/peakpositionfinal_SW1.txt",ios::in); //remember to choose SW1 or SW2!!!

    //file variables
    float a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a17,a18,a19;
    float a1new;
    float mean1, mean2, maximumX, FirstValue, integral, integralnoise;
    float convYearPeak, convMonthPeak, convDayPeak, convHourPeak, convMinutePeak, convSecondPeak, a6fPeak, a12fPeak, a15fPeak; //Bea
    float a1f[100000],a2f[100000],a3f[100000],a4f[100000],a5f[100000],a6f[100000],a7f[100000],a8f[100000],a9f[100000],a10f[100000];
    float a11f[100000],a12f[100000],a13f[100000],a14f[100000],a15f[100000],a15f_test[100000],a16f[100000],a17f[100000],a18f[100000],a19f[100000];
    float IntCurr[100000], IntCurrA[100000];
    float a1newf[100000];
    float P0 = 970.; //mbar
    float T0 = 293.; //K

    //time variables
    char cday[30],cmonth[30],cyear[30],chour[30],cmin[30],csec[30];
    int day,month,year,hour,min,sec;
    TDatime t; //boh
    UInt_t ctime; //boh
    int convYear[100000], convMonth[100000], convDay[100000]; //boh
    int convHour[100000], convMinute[100000], convSecond[100000]; //boh
    int pTime;
    float Time, Timef[100000], Timeg[100000];

    //plot variables
    TCanvas *c[100];
    TGraph *gr[100];
    TLegend *Legend;

    char ctitle[30];
    char titlegif[60];
    char title[60],title2[60],titleday[60];

    //?
    int iplt;
    int i=0;
    int oldHour = 0; //boh

    while(!DataCurrent.eof())
    {
        DataCurrent >> a1new >> a6 >> a7 >> a9 >> a12 >> a15 >> a17 >> a18;

        Timef[i] = a1new;
        t.Set(a1new); //boh
        convYear[i] = t.GetYear(); convMonth[i] = t.GetMonth(); convDay[i] = t.GetDay(); //boh
        convHour[i] = t.GetHour(); convMinute[i] = t.GetMinute(); convSecond[i] = t.GetSecond(); //boh
        /*      if (i<10) { */
        /*        cout << a1new << "\t" << convYear[i] << "\t" << convMonth[i] << "\t" << convDay[i] << endl; */
        /*        cout << convHour[i] << "\t" << convMinute[i] << "\t" << convSecond[i] << endl; */
        /*      } */
        //T1
        a6f[i] = a6;
        //T2
        a7f[i] =  a7;
        //I (nA)
        a9f[i] =  -1*a9 * 1000. - 0.2; //(0.2 nA dark current)
        //dewp
        a12f[i] =  7.5*(a12*1000)-90.;
        //Pabs
        a15f[i] =  ((0.075*(1000*a17))-0.3)*1000.; //giusto che sia a17: invertito in file creato da labview
```

```
        //Prel
        a17f[i] =  3.125*(a15*1000)-37.5;
        //O2
        a18f[i] = 6.25*(a18*1000)-25;
        i++;
    }
    int k=0;
    int k1=0;
    float periodstart, periodend, periodday;
    int yearP, monthP, dayP, hourP, minP, secP;

    //**************************************************Period selector****************************************************

        yearP = 2013;

        monthP = 10;

        dayP = 1;

        hourP = 0; minP = 0; secP = 0;

        TDatime * dateP = new TDatime(yearP, monthP, dayP, hourP, minP, secP);

        UInt_t ttt;

        ttt = dateP->Convert();

        pTimeP = (int)ttt;

        periodstart = (float)pTimeP;



        yearP = 2013;

        monthP = 11;

        dayP = 1;

        hourP = 0; minP = 0; secP = 0;

        TDatime * dateP = new TDatime(yearP, monthP, dayP, hourP, minP, secP);

        UInt_t ttt;

        ttt = dateP->Convert();

        pTimeP = (int)ttt;

        periodend = (float)pTimeP;

//**********************************************************************************************************************
    while(k<i)
    {

        yearP = convYear[k];

        monthP = convMonth[k];

        dayP = convDay[k];

        hourP = convHour[k]; minP = 0; secP = 0;

        TDatime * dateP = new TDatime(yearP, monthP, dayP, hourP, minP, secP);

        UInt_t ttt;

        ttt = dateP->Convert();

        pTimeP = (int)ttt;

        periodday = (float)pTimeP;



        if(periodday>periodstart && periodday<periodend)
        {
            if(k1==0) IntCurr[k1] = a9f[k];
            else IntCurr[k1] = a9f[k] + IntCurr[k1-1];
            if (convHour[k] != oldHour && convHour[k]%6==0)
            {
                IntCurrentP << convYear[k] << "\t" << convMonth[k] << "\t" << convDay[k] << "\t" << convHour[k] << "\t"<< IntCurr[k1] << endl;
```

```
                oldHour = convHour[k];
            }
            k1++;
        }

        if(k==0) IntCurrA[k] = a9f[k];
        else IntCurrA[k] = a9f[k] + IntCurrA[k-1];
        IntCurrentAll << convYear[k] << "\t" << convMonth[k] << "\t" << convDay[k] << "\t" << convHour[k] << "\t"<< IntCurrA[k] << "\t" << a9f[k] << endl;

        k++;
    }
}
```

9.5. PeakTend_v3.C

```
//trend of the peak position

#include <string>
#include <stdlib>
#include <fstream>
#include <iostream>
#include <TGaxis.h>
#include "TCanvas.h"

void PeakTrend_v3(){

  //----------------------------------------

  gROOT->SetStyle("Plain");
  // background is no longer mouse-dropping white
  gStyle->SetCanvasColor(kWhite);
  // blue to red false color palette. Use 9 for b/w
  gStyle->SetPalette(1,0);
  // turn off canvas borders
  gStyle->SetCanvasBorderMode(0);
  gStyle->SetPadBorderMode(0);
  // What precision to put numbers if plotted with "TEXT"
  gStyle->SetPaintTextFormat("5.2f");

  // For publishing:
  gStyle->SetLineWidth(1.5);
  gStyle->SetTextSize(1.1);
  gStyle->SetLabelSize(0.03,"xy");
  gStyle->SetTitleSize(0.04,"xy");
  gStyle->SetTitleOffset(1.1,"x");
  gStyle->SetTitleOffset(0.9,"y");
  gStyle->SetPadTopMargin(0.1);
  gStyle->SetPadRightMargin(0.05);
  gStyle->SetPadBottomMargin(0.1);
  gStyle->SetPadLeftMargin(0.1);

  //---------------------------------------

  //Open File
  fstream DataPeakSW1("C:/root/macros/SingleWire/OutputandPlots/peakpositionfinal_SW1clean.txt",ios::in);
  fstream DataPeakSW2("C:/root/macros/SingleWire/OutputandPlots/peakpositionfinal_SW2clean.txt",ios::in);
  fstream gaincorrSW1("C:/root/macros/SingleWire/OutputandPlots/gaincorrSW1.dat",ios::in);
  fstream gaincorrSW2("C:/root/macros/SingleWire/OutputandPlots/gaincorrSW2.dat",ios::in);

  //fstream DataPeakSW2("/Applications/root/macros/SW/OutputAndPlots/peakpositionRMS_SW2.txt",ios::in);
  //fstream AmbParCorr2("/Applications/root/macros/SW/OutputAndPlots/AmbParCorr2.dat",ios::in); //boh

  //Variables
  float mean1, mean2, maximumX, maximumXcorr, FirstValue, FirstValue1, FirstValue2, FirstValuec, integral, integralnoise, RMS, maximumY, fit1RMS, fit2RMS, FWHM;
  float convYearPeak, convMonthPeak, convDayPeak, convHourPeak, convMinutePeak, convSecondPeak, a6fPeak, a12fPeak, a15fPeak;
  float CorrFact[100000];
  float PeakADC1[100000], PeakADC1corr[100000], PeakADCtmp, maximumXSW1[100000], mean1SW1[100000], mean2SW1[100000], FWHMSW1[100000], fit1RMSSW1[100000], fit2RMSSW1[100000], resolutionSW1[100000], integralSW1[100000], RMSSW1[100000];
  float PeakADC2[100000], PeakADC2corr[100000], PeakADCtmp2, maximumXSW2[100000], mean1SW2[100000], mean2SW2[100000], FWHMSW2[100000], fit1RMSSW2[100000], fit2RMSSW2[100000], resolutionSW2[100000], integralSW2[100000], RMSSW2[100000];
  float P0 = 970.; //mbar
  float T0 = 293.; //K
  int day,month,year,hour,min,sec;
  int yearC, monthC, dayC, hourC;
  float Timeg1[100000], Timeg2[100000], TimegC[100000], TimegC2[100000];
  float trash;

  /******* SW1 *******/

  int i=0;
  int iC=0;
  int j=0;

  //Peak SW1
 while(!gaincorrSW1.eof()) //while(!DataPeakSW1.eof() && !AmbParCorr2.eof())
 {
    gaincorrSW1 >> trash >> yearC >> monthC >> dayC >> hourC >> trash >> maximumXcorr >> trash >> trash;

    //Time conversion
    min=0;
    sec=0;
    TDatime * date = new TDatime(yearC, monthC, dayC, hourC, min, sec);
    UInt_t ttt;
    ttt = date->Convert();
    pTime = (int)ttt;
    TimegC[iC] = (float)pTime;
```

```


    //Normalization for peak position
    if (iC==0)
    {
      FirstValuec = maximumXcorr; //define the first value for the normalization
      PeakADC1corr[iC]=maximumXcorr/FirstValuec;
    }
    if (iC>0)
    {
      PeakADC1corr[iC] = maximumXcorr/FirstValuec;
    }
    iC++;
 }
  int nptPC = iC-1;

  iC=0;
  //SW2 corrected peak possition

 while(!gaincorrSW2.eof()) //while(!DataPeakSW1.eof() && !AmbParCorr2.eof())
 {
    gaincorrSW2 >> trash >> yearC >> monthC >> dayC >> hourC >> trash >> maximumXcorr >> trash >> trash;

    //Time conversion
    min=0;
    sec=0;
    TDatime * date = new TDatime(yearC, monthC, dayC, hourC, min, sec);
    UInt_t ttt;
    ttt = date->Convert();
    pTime = (int)ttt;
    TimegC2[iC] = (float)pTime;


    //Normalization for peak position
    if (iC==0)
    {
      FirstValuec = maximumXcorr; //define the first value for the normalization
      PeakADC2corr[iC]=maximumXcorr/FirstValuec;
    }
    if (iC>0)
    {
      PeakADC2corr[iC] = maximumXcorr/FirstValuec;
    }
    iC++;
 }
  int nptPC2 = iC-1;

 while(!DataPeakSW1.eof()) //while(!DataPeakSW1.eof() && !AmbParCorr2.eof())
  { //Bea

    DataPeakSW1 >> mean1 >> mean2 >> maximumX >> maximumY >> RMS >> FWHM >> fit1RMS >> fit2RMS >> integral >> integralnoise >> hour >> day >> month >> year;
    year= 2000 + year;

    //Time conversion
    min=0;
    sec=0;
    TDatime * date = new TDatime(year, month, day, hour, min, sec);
    UInt_t ttt;
    ttt = date->Convert();
    pTime = (int)ttt;
    Timeg1[i] = (float)pTime;

    //Variables for plots


    //mean1SW1[i] = mean1;
    //mean2SW1[i] = mean2;
    FWHMSW1[i] = FWHM;
    if (fit1RMS>=0) fit1RMSSW1[i] = fit1RMS;
    if (fit1RMS<0) fit1RMSSW1[i] = -1*fit1RMS;
    if (fit2RMS>=0) fit2RMSSW1[i] = fit2RMS;
    if (fit2RMS<0) fit2RMSSW1[i] = -1*fit2RMS;
    resolutionSW1[i] =FWHM/maximumX;
    integralSW1[i] = integral;
    RMSSW1[i] = RMS;

    //Normalization for peak position, mean1, mean 2
    if (i==0)
    {
      FirstValue = maximumX; //define the first value for the normalization
      PeakADC1[i]=maximumX/FirstValue;
      FirstValue1 = mean1; //define the first value for the normalization
      mean1SW1[i]=mean1/FirstValue1;
      FirstValue2 = mean2; //define the first value for the normalization
```

```
      mean2SW1[i]=mean2/FirstValue2;
    }
    if (i>0)
    {
      PeakADC1[i] = maximumX/FirstValue;
      mean1SW1[i] = mean1/FirstValue1;
      mean2SW1[i] = mean2/FirstValue2;
    }
    //PeakADC1[i] = maximumX;
    i++;
  }

  int nptP1 = i-1;

  //Graphs SW1:
  //peak position

  Canvas1 = new TCanvas("SW1 Peak Position Trend","SW1 peak trend",10,10,1300,800);

  GraphSW1 = new TGraph(nptP1, Timeg1, PeakADC1);
  //Peak position
  GraphSW1->SetMarkerSize(0.6);
  GraphSW1->SetMarkerStyle(21);
  GraphSW1->SetMarkerColor(1);
  GraphSW1->SetTitle("SW1 peak trend");
  GraphSW1->GetXaxis()->SetTitle("Time");
  GraphSW1->GetYaxis()->SetTitle("Peak [adc]");
  GraphSW1->GetXaxis()->SetNdivisions(605);
  GraphSW1->GetXaxis()->SetTimeDisplay(1);
  GraphSW1->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
  GraphSW1->Draw("AP");


   //*********************Time division lines for SW1****************************
  float Time1[7];
  year = 2013;
  month = 10;
  day = 1;
  hour = 0; min = 0; sec = 0;
  TDatime * date = new TDatime(year, month, day, hour, min, sec);
  UInt_t ttt;
  ttt = date->Convert();
  pTime = (int)ttt;
  Time1[0] = (float)pTime;

  year = 2013;
  month = 11;
  day = 1;
  hour = 0; min = 0; sec = 0;
  TDatime * date = new TDatime(year, month, day, hour, min, sec);
  UInt_t ttt;
  ttt = date->Convert();
  pTime = (int)ttt;
  Time1[1] = (float)pTime;

  year = 2013;
  month = 11;
  day = 21;
  hour = 0; min = 0; sec = 0;
  TDatime * date = new TDatime(year, month, day, hour, min, sec);
  UInt_t ttt;
  ttt = date->Convert();
  pTime = (int)ttt;
  Time1[2] = (float)pTime;

  year = 2014;
  month = 1;
  day = 7;
  hour = 0; min = 0; sec = 0;
  TDatime * date = new TDatime(year, month, day, hour, min, sec);
  UInt_t ttt;
  ttt = date->Convert();
  pTime = (int)ttt;
  Time1[3] = (float)pTime;

  year = 2014;
  month = 2;
  day = 27;
  hour = 0; min = 0; sec = 0;
  TDatime * date = new TDatime(year, month, day, hour, min, sec);
  UInt_t ttt;
  ttt = date->Convert();
  pTime = (int)ttt;
  Time1[4] = (float)pTime;
```

```
  year = 2014;
  month = 4;
  day = 28;
  hour = 0; min = 0; sec = 0;
  TDatime * date = new TDatime(year, month, day, hour, min, sec);
  UInt_t ttt;
  ttt = date->Convert();
  pTime = (int)ttt;
  Time1[5] = (float)pTime;

  year = 2014;
  month = 5;
  day = 28;
  hour = 0; min = 0; sec = 0;
  TDatime * date = new TDatime(year, month, day, hour, min, sec);
  UInt_t ttt;
  ttt = date->Convert();
  pTime = (int)ttt;
  Time1[6] = (float)pTime;

  //********************Time division lines for SW1 end***************************
  //Canvas1 lines
  Canvas1->Update();
  for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines],Canvas1->GetUymin(),Time1[lines],Canvas1->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }

  Canvas1corr = new TCanvas("Corrected SW1 Peak Position Trend","Corrected SW1 peak trend",10,10,1300,800);
  GraphSW1corr = new TGraph(nptPC, TimegC, PeakADC1corr);
  //Peak position
  GraphSW1corr->SetMarkerSize(0.6);
  GraphSW1corr->SetMarkerStyle(21);
  GraphSW1corr->SetMarkerColor(2);
  GraphSW1corr->SetTitle("Corrected SW1 peak trend ");
  GraphSW1corr->GetXaxis()->SetTitle("Time");
  GraphSW1corr->GetYaxis()->SetTitle("Peak [adc]");
  GraphSW1corr->GetXaxis()->SetNdivisions(605);
  GraphSW1corr->GetXaxis()->SetTimeDisplay(1);
  GraphSW1corr->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
  GraphSW1corr->Draw("AP");

  //Canvascorr1 lines
  Canvas1corr->Update();
  for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines],Canvas1corr->GetUymin(),Time1[lines],Canvas1corr->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }

 CanvasSW1both= new TCanvas("Corrected SW1 Peak Position Trend vs Uncorrected","Corrected SW1 peak trend vs Uncorrected",10,10,1300,800);
 Corr1Graph = new TMultiGraph("SW1 peak position correction", "SW1 peak position correction");
 Corr1Graph->Add(GraphSW1);
 Corr1Graph->Add(GraphSW1corr);
 Corr1Graph->Draw("AP");
 Corr1Graph->GetXaxis()->SetTitle("Time");
 Corr1Graph->GetYaxis()->SetTitle("Peak [adc]");
 Corr1Graph->GetXaxis()->SetNdivisions(605);
 Corr1Graph->GetXaxis()->SetTimeDisplay(1);
 Corr1Graph->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
 //MeansGraph->Draw("AP");

  CanvasSW1both->Update();
  for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines], CanvasSW1both->GetUymin(),Time1[lines], CanvasSW1both->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }

  Canvas2 = new TCanvas("SW1 Mean1 & Mean2 Trend","SW1 Mean1 & Mean2 Trend",10,10,1300,800);
  Canvas2->Divide(1,2);

  GraphMean1SW1 = new TGraph(nptP1, Timeg1, mean1SW1);
```

```
  GraphMean2SW1 = new TGraph(nptP1, Timeg1, mean2SW1);

   //mean1
  Canvas2->Update();
  GraphMean1SW1->SetMarkerSize(0.75);
  GraphMean1SW1->SetMarkerStyle(21);
  GraphMean1SW1->SetMarkerColor(2);
  GraphMean1SW1->SetTitle("SW1 Mean1 Trend");
  GraphMean1SW1->GetXaxis()->SetTitle("Time");
  GraphMean1SW1->GetYaxis()->SetTitle("mean [adc]");
  GraphMean1SW1->GetXaxis()->SetNdivisions(605);
  GraphMean1SW1->GetXaxis()->SetTimeDisplay(1);
  GraphMean1SW1->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");

  //mean2
  GraphMean2SW1->SetMarkerSize(0.6);
  GraphMean2SW1->SetMarkerStyle(21);
  GraphMean2SW1->SetMarkerColor(3);
  GraphMean2SW1->SetTitle("SW1 Mean2 Trend");
  GraphMean2SW1->GetXaxis()->SetTitle("Time");
  GraphMean2SW1->GetYaxis()->SetTitle("mean [adc]");
  GraphMean2SW1->GetXaxis()->SetNdivisions(605);
  GraphMean2SW1->GetXaxis()->SetTimeDisplay(1);
  GraphMean2SW1->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");

 Canvas2->cd(1);
 GraphMean1SW1->Draw("AP");

 Canvas2->cd(2);
 GraphMean2SW1->Draw("AP");

//Canvas2 lines
  Canvas2->Update();
  for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines],Canvas2->cd(1)->GetUymin(),Time1[lines],Canvas2->cd(1)->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }
  for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines],Canvas2->cd(2)->GetUymin(),Time1[lines],Canvas2->cd(2)->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }

/*
 Canvas7 = new TCanvas("SW1 Means","SW1 Means",10,10,1300,800);
 MeansGraph = new TMultiGraph("SW1 Means 1 2", "SW1 Means 1 2");
 MeansGraph->Add(GraphMean1SW1);
 MeansGraph->Add(GraphMean2SW1);
 MeansGraph->Draw("AP");
 MeansGraph->GetXaxis()->SetTitle("Time");
 MeansGraph->GetYaxis()->SetTitle("Mean [adc]");
 MeansGraph->GetXaxis()->SetNdivisions(605);
 MeansGraph->GetXaxis()->SetTimeDisplay(1);
 MeansGraph->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
 //MeansGraph->Draw("AP");

 //Canvas7 lines
  Canvas7->Update();
  for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines],Canvas7->GetUymin(),Time1[lines],Canvas7->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }


 //FWHM
 Canvas3 = new TCanvas("SW1 FWHM Trend","SW1 FWHM trend",10,10,1300,800);

  GraphFWHMSW1 = new TGraph(nptP1, Timeg1, FWHMSW1);
  //Peak position
  GraphFWHMSW1->SetMarkerSize(0.6);
  GraphFWHMSW1->SetMarkerStyle(21);
  GraphFWHMSW1->SetMarkerColor(1);
```

```
 GraphFWHMSW1->SetTitle("SW1 FWHM trend");
 GraphFWHMSW1->GetXaxis()->SetTitle("Time");
 GraphFWHMSW1->GetYaxis()->SetTitle("FWHM [adc]");
 GraphFWHMSW1->GetXaxis()->SetNdivisions(605);
 GraphFWHMSW1->GetXaxis()->SetTimeDisplay(1);
 GraphFWHMSW1->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
 GraphFWHMSW1->SetMinimum(0.1);
 GraphFWHMSW1->Draw("AP");

  //Canvas3 lines
 Canvas3->Update();
 for(int lines=0; lines<7; lines++)
 {
   TLine *templine = new TLine(Time1[lines],Canvas3->GetUymin(),Time1[lines],Canvas3->GetUymax());
   templine->SetLineColor(32);
   templine->SetLineWidth(2);
   templine->SetLineStyle(5);
   templine->Draw();
 }
 */
  //Resolution
Canvas4 = new TCanvas("SW1 Resolution Trend","SW1 Resolution trend",10,10,1300,800);

 GraphResolutionSW1 = new TGraph(nptP1, Timeg1, resolutionSW1);
 //Peak position
 GraphResolutionSW1->SetMarkerSize(0.6);
 GraphResolutionSW1->SetMarkerStyle(21);
 GraphResolutionSW1->SetMarkerColor(1);
 GraphResolutionSW1->SetTitle("SW1 Resolution trend");
 GraphResolutionSW1->GetXaxis()->SetTitle("Time");
 GraphResolutionSW1->GetYaxis()->SetTitle("SW1 Resolution");
 GraphResolutionSW1->GetXaxis()->SetNdivisions(605);
 GraphResolutionSW1->GetXaxis()->SetTimeDisplay(1);
 GraphResolutionSW1->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
 GraphResolutionSW1->SetMinimum(0.1);
 GraphResolutionSW1->Draw("AP");

  //Canvas4 lines
 Canvas4->Update();
 for(int lines=0; lines<7; lines++)
 {
   TLine *templine = new TLine(Time1[lines],Canvas4->GetUymin(),Time1[lines],Canvas4->GetUymax());
   templine->SetLineColor(32);
   templine->SetLineWidth(2);
   templine->SetLineStyle(5);
   templine->Draw();
 }
 /*
 //RMS
 Canvas5 = new TCanvas("SW1 fit1RMS & fit2RMS Trend","SW1 fit1RMS & fit2RMS Trend",10,10,1300,800);
 Canvas5->Divide(1,2);

 Graphfit1RMSSW1 = new TGraph(nptP1, Timeg1, fit1RMSSW1);
 Graphfit2RMSSW1 = new TGraph(nptP1, Timeg1, fit2RMSSW1);

  //fit1RMS
 Canvas5->Update();
 Graphfit1RMSSW1->SetMarkerSize(0.75);
 Graphfit1RMSSW1->SetMarkerStyle(21);
 Graphfit1RMSSW1->SetMarkerColor(2);
 Graphfit1RMSSW1->SetTitle("SW1 fit1RMS Trend");
 Graphfit1RMSSW1->GetXaxis()->SetTitle("Time");
 Graphfit1RMSSW1->GetYaxis()->SetTitle("fit1RMS [adc]");
 Graphfit1RMSSW1->GetXaxis()->SetNdivisions(605);
 Graphfit1RMSSW1->GetXaxis()->SetTimeDisplay(1);
 Graphfit1RMSSW1->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
 Graphfit1RMSSW1->SetMaximum(300);

 //fit2RMS
 Graphfit2RMSSW1->SetMarkerSize(0.6);
 Graphfit2RMSSW1->SetMarkerStyle(21);
 Graphfit2RMSSW1->SetMarkerColor(3);
 Graphfit2RMSSW1->SetTitle("SW1 fit2RMS Trend");
 Graphfit2RMSSW1->GetXaxis()->SetTitle("Time");
 Graphfit2RMSSW1->GetYaxis()->SetTitle("fit2RMS [adc]");
 Graphfit2RMSSW1->GetXaxis()->SetNdivisions(605);
 Graphfit2RMSSW1->GetXaxis()->SetTimeDisplay(1);
 Graphfit2RMSSW1->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
 Graphfit2RMSSW1->SetMaximum(300);

Canvas5->cd(1);
Graphfit1RMSSW1->Draw("AP");

Canvas5->cd(2);
Graphfit2RMSSW1->Draw("AP");
```

```

  //Canvas5 lines
  Canvas5->Update();
  for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines],Canvas5->cd(1)->GetUymin(),Time1[lines],Canvas5->cd(1)->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }
   for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines],Canvas5->cd(2)->GetUymin(),Time1[lines],Canvas5->cd(2)->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }

  //Integral
 Canvas6 = new TCanvas("SW1 Integral Trend","SW1 Integral trend",10,10,1300,800);

  GraphIntSW1 = new TGraph(nptP1, Timeg1, integralSW1);
  GraphIntSW1->SetMarkerSize(0.6);
  GraphIntSW1->SetMarkerStyle(21);
  GraphIntSW1->SetMarkerColor(1);
  GraphIntSW1->SetTitle("SW1 Integral trend");
  GraphIntSW1->GetXaxis()->SetTitle("Time");
  GraphIntSW1->GetYaxis()->SetTitle("Integral");
  GraphIntSW1->GetXaxis()->SetNdivisions(605);
  GraphIntSW1->GetXaxis()->SetTimeDisplay(1);
  GraphIntSW1->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
  GraphIntSW1->SetMaximum(1000000);
  GraphIntSW1->SetMinimum(100000);
  GraphIntSW1->Draw("AP");

  //Canvas6 lines
  Canvas6->Update();
  for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines],Canvas6->GetUymin(),Time1[lines],Canvas6->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }

  */
  /*
  //Scale GraphSWint to the pad coordinates
  Float_t rightmax = GraphSW1RMS->GetHistogram()->GetMaximum();
  Float_t scale = gPad->GetUymax()/rightmax;
  GraphSW1RMS->SetLineColor(kRed);
  for (int i=0;i<GraphSW1RMS->GetN();i++) {
    GraphSW1RMS->GetY()[i] *= scale; //equivalent to scale
  }
  GraphSW1RMS->Draw("P");

  //draw an axis on the right side
  TGaxis *axis = new TGaxis(gPad->GetUxmax(),gPad->GetUymin(),
               gPad->GetUxmax(), gPad->GetUymax(),0,rightmax,510,"+L");
  axis->SetLineColor(kRed);
  axis->SetLabelColor(kRed);
  axis->Draw();
  */



  /******* SW2 *******/

  int k = 0;

  //Peak SW2
  while(!DataPeakSW2.eof()){

    DataPeakSW2 >> mean1 >> mean2 >> maximumX >> maximumY >> RMS >> FWHM >> fit1RMS >> fit2RMS >> integral >> integralnoise >> hour >> day >> month >> year;
    year= 2000 + year; //spostata qui da sotto


    //Time conversion
    TDatime * date = new TDatime(year, month, day, hour, min, sec);
```

```
    UInt_t ttt;
    ttt = date->Convert();
    pTime = (int)ttt;
    Timeg2[k] = (float)pTime;

    FWHMSW2[k] = FWHM;
    if (fit1RMS>=0) fit1RMSSW2[k] = fit1RMS;
    if (fit1RMS<0) fit1RMSSW2[k] = -1*fit1RMS;
    if (fit2RMS>=0) fit2RMSSW2[k] = fit2RMS;
    if (fit2RMS<0) fit2RMSSW2[k] = -1*fit2RMS;
    resolutionSW2[k] =FWHM/maximumX;
    integralSW2[k] = integral;
    RMSSW2[k] = RMS;

     //Normalization for peak position, mean1, mean 2
    if (k==0)
    {
      FirstValue = maximumX; //define the first value for the normalization
      PeakADC2[k]=maximumX/FirstValue;
      FirstValue1 = mean1; //define the first value for the normalization
      mean1SW2[k]=mean1/FirstValue1;
      FirstValue2 = mean2; //define the first value for the normalization
      mean2SW2[k]=mean2/FirstValue2;
    }
    if (k>0)
    {
      PeakADC2[k] = maximumX/FirstValue;
      mean1SW2[k] = mean1/FirstValue1;
      mean2SW2[k] = mean2/FirstValue2;
    }
    k++;
  }

  int nptP2 = k-1;
  //for(int t=0; t<957; t++)
      //cout <<

  //Graphs for SW

  //peak position

  Canvas8 = new TCanvas("SW2 Peak Position Trend","SW2 peak trend",10,10,1300,800);

  GraphSW2 = new TGraph(nptP2, Timeg2, PeakADC2);
  //Peak position
  GraphSW2->SetMarkerSize(0.6);
  GraphSW2->SetMarkerStyle(21);
  GraphSW2->SetMarkerColor(1);
  GraphSW2->SetTitle("SW2 peak trend");
  GraphSW2->GetXaxis()->SetTitle("Time");
  GraphSW2->GetYaxis()->SetTitle("Peak [adc]");
  GraphSW2->GetXaxis()->SetNdivisions(605);
  GraphSW2->GetXaxis()->SetTimeDisplay(1);
  GraphSW2->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
  GraphSW2->Draw("AP");
  //Canvas 8 lines
  Canvas8->Update();
  for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines],Canvas8->GetUymin(),Time1[lines],Canvas8->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }

  Canvas8corr = new TCanvas("Corrected SW2 Peak Position Trend","Corrected SW2 peak trend",10,10,1300,800);
  GraphSW2corr = new TGraph(nptPC2, TimegC2, PeakADC2corr);
  //Peak position
  GraphSW2corr->SetMarkerSize(0.6);
  GraphSW2corr->SetMarkerStyle(21);
  GraphSW2corr->SetMarkerColor(2);
  GraphSW2corr->SetTitle("Corrected SW2 peak trend ");
  GraphSW2corr->GetXaxis()->SetTitle("Time");
  GraphSW2corr->GetYaxis()->SetTitle("Peak [adc]");
  GraphSW2corr->GetXaxis()->SetNdivisions(605);
  GraphSW2corr->GetXaxis()->SetTimeDisplay(1);
  GraphSW2corr->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
  GraphSW2corr->Draw("AP");



  //Canvascorr1 lines
  Canvas8corr->Update();
  for(int lines=0; lines<7; lines++)
```

```
  {
    TLine *templine = new TLine(Time1[lines],Canvas8corr->GetUymin(),Time1[lines],Canvas8corr->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }

 CanvasSW2both= new TCanvas("Corrected SW2 Peak Position Trend vs Uncorrected","Corrected SW2 peak trend vs Uncorrected",10,10,1300,800);
 corr2Graph = new TMultiGraph("SW2 peak position correction", "SW2 peak position correction");
 corr2Graph->Add(GraphSW2);
 corr2Graph->Add(GraphSW2corr);
 corr2Graph->Draw("AP");
 corr2Graph->GetXaxis()->SetTitle("Time");
 corr2Graph->GetYaxis()->SetTitle("Peak [adc]");
 corr2Graph->GetXaxis()->SetNdivisions(605);
 corr2Graph->GetXaxis()->SetTimeDisplay(1);
 corr2Graph->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
 //MeansGraph->Draw("AP");

  CanvasSW2both->Update();
  for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines], CanvasSW2both->GetUymin(),Time1[lines], CanvasSW2both->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }

  Canvas9 = new TCanvas("SW2 Mean1 & Mean2 Trend","SW2 Mean1 & Mean2 Trend",10,10,1300,800);
  Canvas9->Divide(1,2);

  GraphMean1SW2 = new TGraph(nptP2, Timeg2, mean1SW2);
  GraphMean2SW2 = new TGraph(nptP2, Timeg2, mean2SW2);

   //mean1
  Canvas9->Update();
  GraphMean1SW2->SetMarkerSize(0.75);
  GraphMean1SW2->SetMarkerStyle(21);
  GraphMean1SW2->SetMarkerColor(2);
  GraphMean1SW2->SetTitle("SW2 Mean1 Trend");
  GraphMean1SW2->GetXaxis()->SetTitle("Time");
  GraphMean1SW2->GetYaxis()->SetTitle("mean [adc]");
  GraphMean1SW2->GetXaxis()->SetNdivisions(605);
  GraphMean1SW2->GetXaxis()->SetTimeDisplay(1);
  GraphMean1SW2->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
  GraphMean1SW2->SetMaximum(1.3);
  GraphMean1SW2->SetMinimum(0.5);

  //mean2
  GraphMean2SW2->SetMarkerSize(0.6);
  GraphMean2SW2->SetMarkerStyle(21);
  GraphMean2SW2->SetMarkerColor(3);
  GraphMean2SW2->SetTitle("SW2 Mean2 Trend");
  GraphMean2SW2->GetXaxis()->SetTitle("Time");
  GraphMean2SW2->GetYaxis()->SetTitle("mean [adc]");
  GraphMean2SW2->GetXaxis()->SetNdivisions(605);
  GraphMean2SW2->GetXaxis()->SetTimeDisplay(1);
  GraphMean2SW2->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
  GraphMean2SW2->SetMaximum(1.3);
  GraphMean2SW2->SetMinimum(0.5);

 Canvas9->cd(1);
 GraphMean1SW2->Draw("AP");

 Canvas9->cd(2);
 GraphMean2SW2->Draw("AP");

   //Canvas9 lines
  Canvas9->Update();
  for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines],Canvas9->cd(1)->GetUymin(),Time1[lines],Canvas9->cd(1)->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }
   for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new
```

```
TLine(Time1[lines],Canvas9->cd(2)->GetUymin(),Time1[lines],Canvas9->cd(2)->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }

/*
 Canvas10 = new TCanvas("SW2 Means","SW2 Means",10,10,1300,800);
 MeansGraph2 = new TMultiGraph("SW2 Means 1 2", "SW2 Means 1 2");
 MeansGraph2->Add(GraphMean1SW2);
 MeansGraph2->Add(GraphMean2SW2);
 MeansGraph2->Draw("AP");
 MeansGraph2->GetXaxis()->SetTitle("Time");
 MeansGraph2->GetYaxis()->SetTitle("Mean [adc]");
 MeansGraph2->GetXaxis()->SetNdivisions(605);
 MeansGraph2->GetXaxis()->SetTimeDisplay(1);
 MeansGraph2->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
 //Canvas 10 lines
  Canvas10->Update();
  for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines],Canvas10->GetUymin(),Time1[lines],Canvas10->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }


 //FWHM
 Canvas11 = new TCanvas("SW2 FWHM Trend","SW2 FWHM trend",10,10,1300,800);

  GraphFWHMSW2 = new TGraph(nptP2, Timeg2, FWHMSW2);
  //Peak position
  GraphFWHMSW2->SetMarkerSize(0.6);
  GraphFWHMSW2->SetMarkerStyle(21);
  GraphFWHMSW2->SetMarkerColor(1);
  GraphFWHMSW2->SetTitle("SW2 FWHM trend");
  GraphFWHMSW2->GetXaxis()->SetTitle("Time");
  GraphFWHMSW2->GetYaxis()->SetTitle("FWHM [adc]");
  GraphFWHMSW2->GetXaxis()->SetNdivisions(605);
  GraphFWHMSW2->GetXaxis()->SetTimeDisplay(1);
  GraphFWHMSW2->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
  GraphFWHMSW2->SetMinimum(100);
  GraphFWHMSW2->Draw("AP");
   //Canvas 11 lines
  Canvas11->Update();
  for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines],Canvas11->GetUymin(),Time1[lines],Canvas11->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }
  */
   //Resolution
 Canvas12 = new TCanvas("SW2 Resolution Trend","SW2 Resolution trend",10,10,1300,800);

  GraphResolutionSW2 = new TGraph(nptP2, Timeg2, resolutionSW2);
  //Peak position
  GraphResolutionSW2->SetMarkerSize(0.6);
  GraphResolutionSW2->SetMarkerStyle(21);
  GraphResolutionSW2->SetMarkerColor(1);
  GraphResolutionSW2->SetTitle("SW2 Resolution trend");
  GraphResolutionSW2->GetXaxis()->SetTitle("Time");
  GraphResolutionSW2->GetYaxis()->SetTitle("SW2 Resolution");
  GraphResolutionSW2->GetXaxis()->SetNdivisions(605);
  GraphResolutionSW2->GetXaxis()->SetTimeDisplay(1);
  GraphResolutionSW2->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
  GraphResolutionSW2->SetMinimum(0.1);
  GraphResolutionSW2->Draw("AP");
   //Canvas 12 lines
  Canvas12->Update();
  for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines],Canvas12->GetUymin(),Time1[lines],Canvas12->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }

  /*
```

```
  //RMS
  Canvas13 = new TCanvas("SW2 fit1RMS & fit2RMS Trend","SW2 fit1RMS & fit2RMS Trend",10,10,1300,800);
  Canvas13->Divide(1,2);

  Graphfit1RMSSW2 = new TGraph(nptP2, Timeg2, fit1RMSSW2);
  Graphfit2RMSSW2 = new TGraph(nptP2, Timeg2, fit2RMSSW2);

   //fit1RMS
  Canvas13->Update();
  Graphfit1RMSSW2->SetMarkerSize(0.75);
  Graphfit1RMSSW2->SetMarkerStyle(21);
  Graphfit1RMSSW2->SetMarkerColor(2);
  Graphfit1RMSSW2->SetTitle("SW2 fit1RMS Trend");
  Graphfit1RMSSW2->GetXaxis()->SetTitle("Time");
  Graphfit1RMSSW2->GetYaxis()->SetTitle("fit1RMS [adc]");
  Graphfit1RMSSW2->GetXaxis()->SetNdivisions(605);
  Graphfit1RMSSW2->GetXaxis()->SetTimeDisplay(1);
  Graphfit1RMSSW2->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
  Graphfit1RMSSW2->SetMaximum(300);

  //fit2RMS
  Graphfit2RMSSW2->SetMarkerSize(0.6);
  Graphfit2RMSSW2->SetMarkerStyle(21);
  Graphfit2RMSSW2->SetMarkerColor(3);
  Graphfit2RMSSW2->SetTitle("SW2 fit2RMS Trend");
  Graphfit2RMSSW2->GetXaxis()->SetTitle("Time");
  Graphfit2RMSSW2->GetYaxis()->SetTitle("fit2RMS [adc]");
  Graphfit2RMSSW2->GetXaxis()->SetNdivisions(605);
  Graphfit2RMSSW2->GetXaxis()->SetTimeDisplay(1);
  Graphfit2RMSSW2->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
  Graphfit2RMSSW2->SetMaximum(300);

 Canvas13->cd(1);
 Graphfit1RMSSW2->Draw("AP");

 Canvas13->cd(2);
 Graphfit2RMSSW2->Draw("AP");

   //Canvas13 lines
  Canvas13->Update();
  for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines],Canvas13->cd(1)->GetUymin(),Time1[lines],Canvas13->cd(1)->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }
   for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines],Canvas13->cd(2)->GetUymin(),Time1[lines],Canvas13->cd(2)->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }

  //Integral
 Canvas14 = new TCanvas("SW2 Integral Trend","SW2 Integral trend",10,10,1300,800);

  GraphIntSW2 = new TGraph(nptP2, Timeg2, integralSW2);
  GraphIntSW2->SetMarkerSize(0.6);
  GraphIntSW2->SetMarkerStyle(21);
  GraphIntSW2->SetMarkerColor(1);
  GraphIntSW2->SetTitle("SW2 Integral trend");
  GraphIntSW2->GetXaxis()->SetTitle("Time");
  GraphIntSW2->GetYaxis()->SetTitle("Integral");
  GraphIntSW2->GetXaxis()->SetNdivisions(605);
  GraphIntSW2->GetXaxis()->SetTimeDisplay(1);
  GraphIntSW2->GetXaxis()->SetTimeFormat("%d\/%m\/%Y %F1970-01-01 00:00:00");
  GraphIntSW2->SetMaximum(400000);
  GraphIntSW2->Draw("AP");
   //Canvas 14 lines
  Canvas14->Update();
  for(int lines=0; lines<7; lines++)
  {
    TLine *templine = new TLine(Time1[lines],Canvas14->GetUymin(),Time1[lines],Canvas14->GetUymax());
    templine->SetLineColor(32);
    templine->SetLineWidth(2);
    templine->SetLineStyle(5);
    templine->Draw();
  }
   */
```

```
/*
 //Scale GraphSW2RMS to the pad coordinates
  Float_t rightmax = GraphSW2RMS->GetHistogram()->GetMaximum();
  Float_t scale = gPad->GetUymax()/rightmax;
  GraphSW2RMS->SetLineColor(kRed);
  for (int i=0;i<GraphSW2RMS->GetN();i++) {
    GraphSW2RMS->GetY()[i] *= scale; //equivalent to scale
  }
  GraphSW2RMS->Draw("P");

 //draw an axis on the right side
  TGaxis *axis = new TGaxis(gPad->GetUxmax(),gPad->GetUymin(),
                gPad->GetUxmax(), gPad->GetUymax(),0,rightmax,510,"+L");
  axis->SetLineColor(kRed);
  axis->SetLabelColor(kRed);
  axis->Draw();
*/
}
```

9.6. SWspectra_v1.C (Lines 69-239 [20])

```
//Difference with provabea3.C: adding the integral of peak in peakposition_SW#.txt file
//Add also RMS

#include <string>
#include <cstring>
#include <stdlib>
#include <fstream>
#include <iostream>

using namespace std;

void SWspectra_v1(int d, int m){

    //----------------------------------------

    gROOT->SetStyle("Plain");
    // background is no longer mouse-dropping white
    gStyle->SetCanvasColor(kWhite);
    // blue to red false color palette. Use 9 for b/w
    gStyle->SetPalette(1,0);
    // turn off canvas borders
    gStyle->SetCanvasBorderMode(0);
    gStyle->SetPadBorderMode(0);
    // What precision to put numbers if plotted with "TEXT"
    gStyle->SetPaintTextFormat("5.2f");

    // For publishing:
    gStyle->SetLineWidth(1.5);
    gStyle->SetTextSize(1.1);
    gStyle->SetLabelSize(0.05,"xy");
    gStyle->SetTitleSize(0.05,"xy");
    gStyle->SetTitleOffset(1.1,"x");
    gStyle->SetTitleOffset(0.9,"y");
    gStyle->SetPadTopMargin(0.1);
    gStyle->SetPadRightMargin(0.1);
    gStyle->SetPadBottomMargin(0.16);
    gStyle->SetPadLeftMargin(0.12);

    //---------------------------------------



    fstream peakposition1("C:/root/macros/SingleWire/OutputandPlots/peakpositiontest_SW1.txt",ios::out|ios::app); //|ios::app appende i dati al file
    fstream peakposition2("C:/root/macros/SingleWire/OutputandPlots/peakpositiontest_SW2.txt",ios::out|ios::app); //|ios::app appende i dati al file
    char title[600],title2[600],title3[600],title4[600];
    char date[60];

    //const char *dirname="/Users/beatricemandelli/Desktop/SWprova/";
    const char *dirname="C:/DataSW/SW01-SW02c/";
    const char *ext=".dat";
    char namefile[100];
    char cday[30],cmonth[30],cyear[30],chour[30],cmin[30],csec[30];
    int year, month, day, hour, min, hour_0, hour_6, hour_12, hour_18;
    int if0, if6, if12, if18;
    float a1, a2, a3;
    double mean1, mean2, mean1_6, mean2_6, test,test1;
    TString fname, fname2, fnameDir;
    int HnumBin=600; //only multiples 0f 150 seem to work well

    //TH1F *h0[31][12], *h6[31][12] ,*h12[31][12], *h18[31][12];
    TH1F *h0_1[32][12], *h6_1[32][12], *h12_1[32][12], *h18_1[32][12], *h0_2[32][12], *h6_2[32][12], *h12_2[32][12], *h18_2[32][12];
    TCanvas *peakCanvas1 = new TCanvas ("peakCanvas1","Peaks SW1",1000,750);
    peakCanvas1->Divide(2,2);
    gStyle->SetOptStat(000000000);
    TCanvas *peakCanvas2 = new TCanvas ("peakCanvas2","Peaks SW2",1000,750);
    peakCanvas2->Divide(2,2);


    TSystemDirectory dir(dirname, dirname); //TSystemDirectory(const char* dirname, const char* path)
    TList *files = dir.GetListOfFiles();
    //cout << "files" << endl;
    //files -> Print();


    if (files) {
        TSystemFile *file;
        //TString fname;
        TIter next(files);
        while ((file=(TSystemFile*)next()))
        {
            fname = file->GetName();

```

```
            if (fname!="." && fname!=".." && fname!=".DS_Store") {
                //cout << "fname after  " << fname << endl;

                // miei cambiamenti
                fnameDir="C:/DataSW/SW01-SW02c/"+fname;
                const char *dirname2=fnameDir;
                //cout << "dirname2 " << dirname2 << endl;

                TSystemDirectory dir2(dirname2, dirname2); //TSystemDirectory(const char* dirname, const char* path)
                TList *files2 = dir2.GetListOfFiles();
                //cout << "files2" << endl;
                //files2 -> Print();


                if (files2) {
                    TSystemFile *file2;
                    TIter next2(files2);

                    if0=0;
                    if6=0;
                    if12=0;
                    if18=0;
                    while ((file2=(TSystemFile*)next2())) {

                        fname2 = file2->GetName();          //
                        //cout << "fname2  " <<fname2 << endl;

                        if (!file2->IsDirectory() && fname2.EndsWith(ext)) { //
                            if (fname2[32] =='r'){
                                //prendo file raw
                                //cout << "raw file " << fname2 << endl;
                                sprintf(cyear, "%c%c",fname2[0],fname2[1]);
                                year = atoi(cyear);
                                sprintf(cmonth, "%c%c",fname2[3],fname2[4]);
                                month = atoi(cmonth);
                                sprintf(cday, "%c%c",fname2[6],fname2[7]);
                                day = atoi(cday);
                                sprintf(chour, "%c%c",fname2[9],fname2[10]);
                                hour = atoi(chour);
                                sprintf(cmin, "%c%c",fname2[12],fname2[13]);
                                min = atoi(cmin);
                                //cout << "hour " << hour << endl;

                                //                              if (day == d && month == m) { //aggiunto

                                    //Edited by dhervas
                                        TString  testfname2;
                                        testfname2=fnameDir+"/"+fname2;

                                    //hour 0
                                    if (hour == 0 && if0 <= 4){ //if0<=4

                                        hour_0 = hour;
                                        ifstream datafile;


                                        datafile.open(testfname2,ios::in);

                                        //cout << "testfname2 " << testfname2 << endl;


                                        if (min < 11) { //define title
                                            sprintf(title,"SW1 %d-%d-2014-hour:%d", day, month, hour);
                                            sprintf(title2,"SW2 %d-%d-2014-hour:%d", day, month, hour);
                                            h0_1[d][m] = new TH1F(title,title,HnumBin,0,3000);
                                            h0_2[d][m] = new TH1F(title2,title2,HnumBin,0,3000);

                                        }

                                        while (!datafile.eof()) {
                                            datafile >> a1 >> a2 >> a3; //a1=SW1, a2=SW2, a3=shutter
                                            //cout << "c " << a1 <<endl;
                                            h0_1[d][m]->Fill(a1);
                                            h0_2[d][m]->Fill(a2);
                                            //cout << "fname2 " << fname2 << endl;
                                        }
                                        //cout << if0 << endl;
                                        if0++;
                                    }

                                    //hour 6
                                    if (hour == 6 && if6 <= 4){

                                        hour_6 = hour;
```

```
                                        ifstream datafile6;
                                        datafile6.open(testfname2,ios::in);

                                        if (min < 11) { //define title
                                            sprintf(title,"SW1 %d-%d-2014-hour:%d", day, month, hour);
                                            sprintf(title2,"SW2 %d-%d-2014-hour:%d", day, month, hour);
                                            h6_1[d][m] = new TH1F(title,title,HnumBin,0,3000);
                                            h6_2[d][m] = new TH1F(title2,title2,HnumBin,0,3000);
                                        }

                                        while (!datafile6.eof()) {
                                            datafile6 >> a1 >> a2 >> a3;
                                            //cout << "c2 " << a1 <<endl;
                                            h6_1[d][m]->Fill(a1);
                                            h6_2[d][m]->Fill(a2);
                                        }
                                        if6++;
                                    }

                                    //hour 12
                                    if (hour == 12 && if12 <= 4){

                                        hour_12 = hour;
                                        ifstream datafile12;
                                        datafile12.open(testfname2,ios::in);

                                        if (min < 11) { //define title
                                            sprintf(title,"SW1 %d-%d-2014-hour:%d", day, month, hour);
                                            sprintf(title2,"SW2 %d-%d-2014-hour:%d", day, month, hour);
                                            h12_1[d][m] = new TH1F(title,title,HnumBin,0,3000);
                                            h12_2[d][m] = new TH1F(title2,title2,HnumBin,0,3000);
                                        }

                                        while (!datafile12.eof()) {
                                            datafile12 >> a1 >> a2 >> a3;
                                            h12_1[d][m]->Fill(a1);
                                            h12_2[d][m]->Fill(a2);
                                        }
                                        if12++;
                                    }


                                    //hour 18
                                    if (hour == 18 && if18 <= 4){

                                        hour_18 = hour;
                                        ifstream datafile18;
                                        datafile18.open(testfname2,ios::in);

                                        if (min < 11) { //define title
                                            sprintf(title,"SW1 %d-%d-2014-hour:%d", day, month, hour);
                                            sprintf(title2,"SW2 %d-%d-2014-hour:%d", day, month, hour);
                                            h18_1[d][m] = new TH1F(title,title,HnumBin,0,3000);
                                            h18_2[d][m] = new TH1F(title2,title2,HnumBin,0,3000);
                                        }

                                        while (!datafile18.eof()) {
                                            datafile18 >> a1 >> a2 >> a3;
                                            h18_1[d][m]->Fill(a1);
                                            h18_2[d][m]->Fill(a2);
                                        }
                                        if18++;
                                    }

                                }
                            }
                        }
                    }
                }
            }

        }
    }


    //-----plot hour 0-----
    //SW1

    ///testing histogram arrays
    //TH1F *testhist;
    //int testvar=3;
    //testhist=&testvar;
    //Thf1 does not seem to work as 2d vector
    //TH1F testarray[2][4]={{*h0_1[d][m], *h6_1[d][m], *h12_1[d][m], *h18_1[d][m]}, {*h0_2[d][m], *h6_2[d][m], *h12_2[d][m], *h18_2[d][m]}};
```

```
    TH1F *testarray[8]={*h0_1[d][m], *h6_1[d][m], *h12_1[d][m], *h18_1[d][m], *h0_2[d][m], *h6_2[d][m], *h12_2[d][m], *h18_2[d][m]};
    //TH1F *testarray[1];
    //testarray[0]=*h0_1[d][m];

    ///loop for canvas 1 SW1
    double mean1[4];
    double mean2[4];
    double maximumbinY[4];
    double maximumY[4];
    double maximumX[4];
    double RMS[4];
    double integral[4];
    double integralnoise[4];
    double fit1RMS[4];
    double fit2RMS[4];
    double FWHM[4];
    int rangethreshold=150;

    for (int canvashour=0; canvashour<=3; canvashour++)
    {

        //****************** Start histogram range (without pedestrial) calculation***********
        int bin1range=-1;
        //int nbins = 1000;
        int nbinsr = testarray[canvashour]->GetNbinsX();

        int nabover = 1;

        //Finding bin for first cross -> Skipping pedestrial

        for (int binr=1; binr<=nbinsr; binr++)

        {

            if (testarray[canvashour]->GetBinContent(binr) > rangethreshold)

            {

                bin1range=binr;

                break;

            }

        }
        //Finding bin for second cross -> Skipping pedestrial
        for (int bin2r=bin1range; bin2r<=nbinsr; bin2r++)

        {

            if (testarray[canvashour]->GetBinContent(bin2r) < rangethreshold)

            {

                bin1range=bin2r;

                break;

            }

        }
        //Finding 3rd cross-> first point for range calculation
        for (int bin3r=bin1range; bin3r<=nbinsr; bin3r++)

        {

            if (testarray[canvashour]->GetBinContent(bin3r) > rangethreshold)

            {

                bin1range=bin3r;

                break;

            }

        }
        //int bin1range= testarray[canvashour]->FindFirstBinAbove(rangethreshold); // Activate whe there is no pedestrail in data
        int bin2range= testarray[canvashour]->FindLastBinAbove(rangethreshold);
        //bin1range=bin1range-3;
        //bin2range=bin2range+3;
        double point1range= testarray[canvashour]->GetXaxis()->GetBinCenter(bin1range);
        double point2range= testarray[canvashour]->GetXaxis()->GetBinCenter(bin2range);
```

```
        //histRange = point2range-point1range;

        cout<< point1range << endl << "HEY ITS A MEEEEEEEEEE 1" <<endl;
        cout<< point2range << endl << "HEY ITS A MEEEEEEEEEE 2" <<endl;
        //cout<< histRange << endl << "HEY ITS A MEEEEEEEEEE FHWM" <<endl;

        ///**********End range calculation**********


        peakCanvas1->cd(canvashour+1);
        testarray[canvashour]->GetXaxis()->SetRange(bin1range,bin2range); //40,100 for HnumBin 150
        maximumbinY[canvashour] = testarray[canvashour]->GetMaximumBin();
        maximumY[canvashour] = testarray[canvashour]->GetBinContent(testarray[canvashour]->GetMaximumBin());
        //maximumX[canvashour] =testarray[canvashour]->GetXaxis()->GetBinCenter(maximumbinY[canvashour]);
        double meanhist=testarray[canvashour]->GetMean(1);

        //****************** Start FWHM calculation***********
        int bin1fwhm=-1;
        //int nbins = 1000;
        int nbins = testarray[canvashour]->GetNbinsX();

        int nabove = 1;

        //Finding bin for first cross -> Skipping pedestrial

        for (int bin=1; bin<=nbins; bin++)

        {

            if (testarray[canvashour]->GetBinContent(bin) > maximumY[canvashour]/2)

            {

                bin1fwhm=bin;

                break;

            }

        }
        //Finding bin for second cross -> Skipping pedestrial
        for (int bin2=bin1fwhm; bin2<=nbins; bin2++)

        {

            if (testarray[canvashour]->GetBinContent(bin2) < maximumY[canvashour]/2)

            {

                bin1fwhm=bin2;

                break;

            }

        }
        //Finding 3rd cross-> first point for fwhm calculation
        for (int bin3=bin1fwhm; bin3<=nbins; bin3++)

        {

            if (testarray[canvashour]->GetBinContent(bin3) > maximumY[canvashour]/2)

            {

                bin1fwhm=bin3;

                break;

            }

        }
        //int bin1fwhm= testarray[canvashour]->FindFirstBinAbove(maximumY[canvashour]/2); // Activate whe there is no pedestrail in data
        int bin2fwhm= testarray[canvashour]->FindLastBinAbove(maximumY[canvashour]/2);
        double point1fwhm= testarray[canvashour]->GetXaxis()->GetBinCenter(bin1fwhm);
        double point2fwhm= testarray[canvashour]->GetXaxis()->GetBinCenter(bin2fwhm);
        FWHM[canvashour] = point2fwhm-point1fwhm;

        //cout<< point1fwhm << endl << "HEY ITS A MEEEEEEEEEE 1" <<endl;
        //cout<< point2fwhm << endl << "HEY ITS A MEEEEEEEEEE 2" <<endl;
        //cout<< FWHM[canvashour] << endl << "HEY ITS A MEEEEEEEEEE FHWM" <<endl;
        ///**********End FWHM calculation*********
```

```


        TF1 *f1 = new TF1("f1","gaus(0)+gaus(3)",point1range,point2range);
        //testarray[canvashour]->GetXaxis()->SetRange(40,100);
        f1->SetParameters(maximumY[canvashour],meanhist,50,maximumY[canvashour],meanhist,50);

        //confining the mean of the fits to be on oposing sides of the mean of the histogram

        //with max
        //f1->SetParLimits(1,maximumX[canvashour]-testarray[canvashour]->GetRMS(),maximumX[canvashour]);
        //f1->SetParLimits(4,maximumX[canvashour],maximumX[canvashour]+testarray[canvashour]->GetRMS());
        //with mean

        //double min1boundf1=meanhist-testarray[canvashour]->GetRMS();
        //double max2boundf1=2*meanhist-min1boundf1;


        f1->SetParLimits(1,point1range,meanhist);
        f1->SetParLimits(0,0,maximumY[canvashour]);
        f1->SetParLimits(4,meanhist,point2range);
        f1->SetParLimits(3,0,maximumY[canvashour]);


//f1->SetParLimits(1,testarray[canvashour]->GetMean(1)-testarray[canvashour]->GetRMS(),testarray[canvashour]->GetMean(1));
//f1->SetParLimits(4,testarray[canvashour]->GetMean(1),testarray[canvashour]->GetMean(1)+testarray[canvashour]->GetRMS());
        //
        //testarray[canvashour]->Fit("f1", "R+", "" , 700, 1600);
        testarray[canvashour]->Fit("f1","RB");
        //h0_1[d][m]->Fit("f1", "R+", "" , 700, 1600);//1000, 2000 without pedestrial
        //h0_1[d][m]->SetMaximum(10000);
        TF1 *g1 = new TF1("g1","[0]*exp(-0.5*((x-[1])/[2])**2)",point1range, point2range); //700, 1600  //1000, 2000
        TF1 *g2 = new TF1("g2","[0]*exp(-0.5*((x-[1])/[2])**2)",point1range, point2range); //1000, 2000
        g1->SetParameters(f1->GetParameter(0),f1->GetParameter(1), f1->GetParameter(2) );
        g2->SetParameters(f1->GetParameter(3),f1->GetParameter(4), f1->GetParameter(5) );
        g1->SetLineColor(2);
        g1->SetLineWidth(2);
        g1->Draw("SAME");
        g2->SetLineColor(8);
        g2->SetLineWidth(2);
        g2->Draw("SAME");

        float ciao = f1->GetMaximumX();
        //cout << "ciao " << ciao << endl;
        maximumX[canvashour] =ciao;

        mean1[canvashour] = testarray[canvashour]->GetFunction("f1")->GetParameter(1);
        fit1RMS[canvashour] = testarray[canvashour]->GetFunction("f1")->GetParameter(2);
        //cout << rms1_0 << endl;
        mean2[canvashour] = testarray[canvashour]->GetFunction("f1")->GetParameter(4);
        fit2RMS[canvashour] = testarray[canvashour]->GetFunction("f1")->GetParameter(5);
        //cout << rms2_0 << endl;

        RMS[canvashour] = testarray[canvashour]->GetRMS(); //Get the RMS of the histogram
        integral[canvashour] = testarray[canvashour]->Integral(bin1range,bin2range, "width"); //calculate the integral of the peak
        integralnoise[canvashour] = testarray[canvashour]->Integral(0,bin1range, "width"); //calculate the integral of the noise
        //to see pedestrial uncoment:
        //testarray[canvashour]->GetXaxis()->SetRange();


        peakposition1 << mean1[canvashour] << "\t" << mean2[canvashour] << "\t" << maximumX[canvashour] << "\t" << maximumY[canvashour]<< "\t" << RMS[canvashour] << "\t" << FWHM[canvashour] << "\t" << fit1RMS[canvashour] << "\t" << fit2RMS[canvashour] << "\t" << integral[canvashour] << "\t" << integralnoise[canvashour] << "\t" << canvashour*6 << "\t" << d << "\t" << m << "\t" << "13" << "\t" << endl; //year
        cout << "maxY" << canvashour*6 << "   " << maximumY[canvashour]<< endl;
        cout << "RMS" << canvashour*6 << "   " << RMS[canvashour]<< endl;
        cout << "fit1RMS" << canvashour*6 << "   " << fit1RMS[canvashour]<< endl;
        cout << "fit2RMS" << canvashour*6 << "   " << fit2RMS[canvashour]<< endl;
        cout << "Xvalue" << canvashour*6 << "   " << maximumX[canvashour]<< endl;
        cout << "mean1_" << canvashour*6 << "   " <<  mean1[canvashour]<< endl;
        cout << "mean2_" << canvashour*6 << "   " << mean2[canvashour]<< endl;
        cout << "INTEGRAL" << canvashour*6 << "   " << integral[canvashour]<< endl;
        cout << "INTEGRAL NOISE" << canvashour*6 << "   " << integralnoise[canvashour]<< endl;
    }


    double mean1_2[4];
    double mean2_2[4];
    double maximumbinY_2[4];
    double maximumY_2[4];
```

```
    double maximumX_2[4];
    double RMS_2[4];
    double integral_2[4];
    double integralnoise_2[4];
    double fit1RMS_2[4];
    double fit2RMS_2[4];
    double FWHM_2[4];

    //loop for SW2

    for (int canvashour2=0; canvashour2<=3; canvashour2++)
    {

        //****************** Start histogram range (without pedestrial) calculation***********
        int bin1range=-1;
        //int nbins = 1000;
        int nbinsr = testarray[canvashour2+4]->GetNbinsX();

        int nabover = 1;

        //Finding bin for first cross -> Skipping pedestrial

        for (int binr=1; binr<=nbinsr; binr++)

        {

            if (testarray[canvashour2+4]->GetBinContent(binr) > rangethreshold)

            {

                bin1range=binr;

                break;

            }

        }
        //Finding bin for second cross -> Skipping pedestrial
        for (int bin2r=bin1range; bin2r<=nbinsr; bin2r++)

        {

            if (testarray[canvashour2+4]->GetBinContent(bin2r) < rangethreshold)

            {

                bin1range=bin2r;

                break;

            }

        }
        //Finding 3rd cross-> first point for range calculation
        for (int bin3r=bin1range; bin3r<=nbinsr; bin3r++)

        {

            if (testarray[canvashour2+4]->GetBinContent(bin3r) > rangethreshold)

            {

                bin1range=bin3r;

                break;

            }

        }
        int bin2range= testarray[canvashour2+4]->FindLastBinAbove(rangethreshold);
        bin1range=bin1range-3;
        bin2range=bin2range+3;
        double point1range= testarray[canvashour2+4]->GetXaxis()->GetBinCenter(bin1range);
        double point2range= testarray[canvashour2+4]->GetXaxis()->GetBinCenter(bin2range);
        //histRange = point2range-point1range;

        cout<< point1range << endl << "HEY ITS A MEEEEEEEEEE 1" <<endl;
        cout<< point2range << endl << "HEY ITS A MEEEEEEEEEE 2" <<endl;
        //cout<< histRange << endl << "HEY ITS A MEEEEEEEEEE FHWM" <<endl;

        ///**********End range calculation**********


        peakCanvas2->cd(canvashour2+1);

```

```
        testarray[canvashour2+4]->GetXaxis()->SetRange(bin1range,bin2range); //40,100 for bin 150
        maximumbinY_2[canvashour2] = testarray[canvashour2+4]->GetMaximumBin();
        maximumY_2[canvashour2] = testarray[canvashour2+4]->GetBinContent(testarray[canvashour2+4]->GetMaximumBin());
        //maximumX_2[canvashour2] =testarray[canvashour2+4]->GetXaxis()->GetBinCenter(maximumbinY_2[canvashour2]);
        double meanhist_2=testarray[canvashour2+4]->GetMean(1);
        //****************** Start FWHM calculation***********
        int bin1fwhm=-1;
        //int nbins = 1000;
        int nbins = testarray[canvashour2+4]->GetNbinsX();
        int nabove=1;
        //Finding bin for first cross -> Skipping pedestrial
        for (int bin=1; bin<=nbins; bin++)
        {
            if (testarray[canvashour2+4]->GetBinContent(bin) > maximumY[canvashour2]/2)
            {
                bin1fwhm = bin;
                break;
            }
        }
        //Finding bin for second cross -> Skipping pedestrial
        for (int bin2=bin1fwhm; bin2<=nbins; bin2++)
        {
            if (testarray[canvashour2+4]->GetBinContent(bin2) < maximumY[canvashour2]/2)
            {
                bin1fwhm = bin2;
                break;
            }
        }
        //Finding 3rd cross-> first point for fwhm calculation
        for (int bin3=bin1fwhm; bin3<=nbins; bin3++)
        {
            if (testarray[canvashour2+4]->GetBinContent(bin3) > maximumY[canvashour2]/2)
            {
                bin1fwhm = bin3;
                break;
            }
        }
        int bin2fwhm= testarray[canvashour2+4]->FindLastBinAbove(maximumY[canvashour2]/2);
        double point1fwhm= testarray[canvashour2+4]->GetXaxis()->GetBinCenter(bin1fwhm);
        double point2fwhm= testarray[canvashour2+4]->GetXaxis()->GetBinCenter(bin2fwhm);
        FWHM_2[canvashour2] = point2fwhm - point1fwhm;
        //cout<< point1fwhm << endl << "HEY ITS A MEEEEEEEEEE 1" <<endl;
        //cout<< point2fwhm << endl << "HEY ITS A MEEEEEEEEEE 2" <<endl;
        //cout<< FWHM_2[canvashour2] << endl << "HEY ITS A MEEEEEEEEEE FHWM" <<endl;
        ///**********End FWHM calculation*********
        TF1 *f1 = new TF1("f1","gaus(0)+gaus(3)", point1range, point2range);
        f1->SetParameters(maximumY_2[canvashour2],meanhist_2,50,maximumY_2[canvashour2],meanhist_2,50);
        //confining the mean of the fits to be on oposing sides of the mean of the histogram
        //with max
//f1->SetParLimits(1,maximumX_2[canvashour2]-testarray[canvashour2+4]->GetRMS(),maximumX_2[canvashour2]);
        //f1->SetParLimits(4,maximumX_2[canvashour2],maximumX[canvashour2]+testarray[canvashour2+4]->GetRMS());
```

```
        //with mean
        //double min1boundf1_2 = meanhist_2-testarray[canvashour2+4]->GetRMS();
        //double max2boundf1_2 = 2*meanhist_2-min1boundf1_2;

        f1->SetParLimits(1,point1range,meanhist_2);
        f1->SetParLimits(0,0,maximumY[canvashour2]);
        f1->SetParLimits(4,meanhist_2,point2range);
        f1->SetParLimits(3,0,maximumY[canvashour2]);
        //
        testarray[canvashour2+4]->Fit("f1","RB");
        //h0_1[d][m]->Fit("f1", "R+", "" , 700, 1600);//1000, 2000 without pedestrial
        //testarray[canvashour2+4]->SetMaximum(40000);
        TF1 *g1 = new TF1("g1","[0]*exp(-0.5*((x-[1])/[2])**2)", point1range, point2range); //700, 1600 //1000, 2000
        TF1 *g2 = new TF1("g2","[0]*exp(-0.5*((x-[1])/[2])**2)", point1range, point2range); //1000, 2000
        g1->SetParameters(f1->GetParameter(0),f1->GetParameter(1), f1->GetParameter(2) );
        g2->SetParameters(f1->GetParameter(3),f1->GetParameter(4), f1->GetParameter(5) );
        g1->SetLineColor(2);
        g1->SetLineWidth(2);
        g1->Draw("SAME");
        g2->SetLineColor(8);
        g2->SetLineWidth(2);
        g2->Draw("SAME");


        float ciao2 = f1->GetMaximumX();
        //cout << "cc " << ca << endl;
        maximumX_2[canvashour2] =ciao2;

        mean1_2[canvashour2] = testarray[canvashour2+4]->GetFunction("f1")->GetParameter(1);
        fit1RMS_2[canvashour2] = testarray[canvashour2+4]->GetFunction("f1")->GetParameter(2);
        //cout << rms1_0 << endl;
        mean2_2[canvashour2] = testarray[canvashour2+4]->GetFunction("f1")->GetParameter(4);
        fit2RMS_2[canvashour2] = testarray[canvashour2+4]->GetFunction("f1")->GetParameter(5);
        //cout << rms2_0 << endl;

        //testarray[canvashour2+4]->GetXaxis()->SetRange(40,100); //(40,100)range for peak with pedestrial
        //maximumY_2[canvashour2] = testarray[canvashour2+4]->GetMaximumBin();
        //maximumX_2[canvashour2] =testarray[canvashour2+4]->GetXaxis()->GetBinCenter(maximumbinY_2[canvashour2]);
        RMS_2[canvashour2] = testarray[canvashour2+4]->GetRMS(); //Get the RMS of the histogram
        integral_2[canvashour2] = testarray[canvashour2+4]->Integral(bin1range,bin2range, "width"); //calculate the integral of the peak
        integralnoise_2[canvashour2] = testarray[canvashour2+4]->Integral(0,bin1range, "width"); //calculate the integral of the noise
        //testarray[canvashour2+4]->GetXaxis()->SetRange();


        peakposition2 << mean1_2[canvashour2] << "\t" << mean2_2[canvashour2] << "\t" << maximumX_2[canvashour2] << "\t" << maximumY_2[canvashour2] << "\t" << RMS_2[canvashour2] << "\t"<< FWHM_2[canvashour2] << "\t" << fit1RMS_2[canvashour2] << "\t" << fit2RMS_2[canvashour2] << "\t" << integral_2[canvashour2] << "\t" << integralnoise_2[canvashour2] << "\t" << canvashour2*6 << "\t" << d << "\t" << m << "\t" << "13" << "\t" << endl;
        cout << "maxY" << canvashour2*6 << "   " << maximumY_2[canvashour2]<< endl;
        cout << "RMS" << canvashour2*6 << "   " << RMS_2[canvashour2]<< endl;
        cout << "fit1RMS" << canvashour2*6 << "   " << fit1RMS_2[canvashour2]<< endl;
        cout << "fit2RMS" << canvashour2*6 << "   " << fit2RMS_2[canvashour2]<< endl;
        cout << "Xvalue" << canvashour2*6 << "   " << maximumX_2[canvashour2]<< endl;
        cout << "mean1_" << canvashour2*6 << "   " <<  mean1_2[canvashour2]<< endl;
        cout << "mean2_" << canvashour2*6 << "   " << mean2_2[canvashour2]<< endl;
        cout << "INTEGRAL" << canvashour2*6 << "   " << integral_2[canvashour2]<< endl;
        cout << "INTEGRAL NOISE" << canvashour2*6 << "   " << integralnoise_2[canvashour2]<< endl;
    }




    //save canvas
    sprintf(title,"C:/root/macros/SingleWire/OutputandPlots/SW1PNG/Peaks_SW1_20%d-%d-%d.png", year, m, d);
    peakCanvas1->SaveAs(title);
    sprintf(title2,"C:/root/macros/SingleWire/OutputandPlots/SW1C/Peaks_SW1_20%d-%d-%d.C",year,m,d);
    peakCanvas1->SaveAs(title2);

    sprintf(title3,"C:/root/macros/SingleWire/OutputandPlots/SW2PNG/Peaks_SW2_20%d-%d-%d.png",year,m,d);
    peakCanvas2->SaveAs(title3);
    sprintf(title4,"C:/root/macros/SingleWire/OutputandPlots/SW2C/Peaks_SW2_20%d-%d-%d.C",year,m,d);
    peakCanvas2->SaveAs(title4);

}
```

References

[1] The ROOT Team. (n.d.). ROOT: Data Analysis Framework. Retrieved April 13, 2015, from https://root.cern.ch/drupal/

[2] The ATLAS Experiment. (2011). ATLAS Fact Sheet. Retrieved January 11, 2015, from http://www.atlas.ch/pdf/ATLAS_fact_sheets.pdf

[3] Leo, *W. (*1987*). Techniques for Nuclear and Particle Physics Experiments.* Berlin: Springer-Verlag.

[4] CERN Accelerating science. (n.d.). Retrieved January 11, 2015, from http://home.web.cern.ch/about/how-detector-works

[5] Taking a closer look at LHC - LHC. (n.d.). Retrieved January 11, 2015, from http://www.lhc-closer.es/1/3/13/0

[6] Particle detector. (n.d.). Retrieved April 21, 2015, from http://en.wikipedia.org/wiki/Particle_detector

[7] Haber, C. (Director) (2005, November 16). Tracking with Semiconductor Particle Detectors. Lecture conducted from University of California, Davis.

[8] Particle Data Group. (2010). *Particle Physics Booklet,* Berkeley, Calif.: Lawrence Berkeley National Laboratory ;.

[9] Raether, H. (1964). Electron Avalanches and Breakdown in Gases. *Butterworths Advanced Physics Series.*

[10] Boggende, A., Brinkman, A., & Graaff, W. (1969). Comments on the ageing effect of gas-filled proportional counters. *Journal of Physics E: Scientific Instruments*, (2), 701-705.

[11] Akesson, T. et al. (2003). Aging studies for the ATLAS Transition Radiation Tracker (TRT). *Nuclar Intruments and Methods in Physics Research* A, (515), 166–179.

[12] Kowalski, T., & Mindur, B. (2003). Manifestation of aging effects in gas proportional counters. *Nuclear Instruments and Methods in Physics Research* A, (515), 180-184.

[13] Spielberg, N., & Tsarnas, D. (1975). Counting rate dependent gain shifts in flow proportional counters. *Review of Scientific Instruments*, (46), 1086-1086.

[14] Sputtering. (n.d.). Retrieved February 20, 2015, from http://en.wikipedia.org/wiki/Sputtering

[15] Bouclier, R., Capeans, M., Garabatos, C., Sauli, F., & Silander, K. (1994). Effects of outgassing from some materials on gas chamber ageing. *Nuclar Intruments and Methods in Physics Research A,* (350), 464-469.

[16] Capeans, M. (2003). Aging and materials: Lessons for detectors and gas systems. *Nuclear Instruments and Methods in Physics Research A*, (515), 73-88.

[17] The ROOT Team. (n.d.). ROOT: Data Analysis Framework, About. Retrieved March 23, 2015, from https://root.cern.ch/drupal/content/about

[18] B. Mandelli. (2014). *loop.bat [UNIX shell].*

[19] B. Mandelli. (2014). *chisqtest.C. [ROOT].*

[20] B. Mandelli. (2014). *SWspectra.C. [ROOT].*