

UNIVERSITY COLLEGE LONDON  
University Of London

**Improving Quality Assurance of Proton  
Beam Therapy by advancing pulse signal  
analysis.**

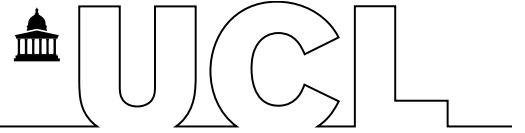
by

Andreas Angelis CHRISTOU

Supervisors: Dr. Simon Jolly and Prof. Ruben Saakyan

DEPARTMENT OF PHYSICS AND ASTRONOMY  
MASTERS THESIS

March 2017  
Gower St, London, WC1E 6BT



**Submission of coursework for Physics and Astronomy course PHASM201/PHAS3400**

Please sign, date and return this form with your coursework by the **specified deadline** to the Departmental Office.

**DECLARATION OF OWNERSHIP**


I confirm that I have read and understood the guidelines on plagiarism, that I understand the meaning of plagiarism and that I may be penalised for submitting work that has been plagiarised.

I confirm that all work will also be submitted electronically and that this can be checked using the JISC detection service, Turnitin®.

I understand that the work cannot be assessed unless both hard copy and electronic versions of the work are handed in.

I declare that all material presented in the accompanying work is entirely my own work except where explicitly and individually indicated and that all sources used in its preparation and all quotations are clearly cited.

Should this statement prove to be untrue, I recognise the right of the Board of Examiners to recommend what action should be taken in line with UCL's regulations.

Signed 

Print Name Andreas Angelis Christou

Dated 27/03/2017

Title	Date Received	Examiner	Examiner's Signature	MARK

## *Acknowledgements*

First and foremost, thank you to my supervisors, particularly Simon for listening to me rabbit on and asking an endless number of questions. Thank you to everyone in the PBT group for your knowledge and patience. Thank you to my family for the support. Thank you to the university gang for making sure I wrote this thesis and tolerating my random moods. Finally, thank you to the squad for the late night sessions enduring my insults and idiosyncracies throughout the past couple years, kept me going the whole way through.

## *Abstract*

The improvement of the quality assurance (QA) procedures in Proton Beam Therapy will lead to significant benefits in the treatment room. Foremost amongst these will be the reduction in time taken to prepare the beam for treatment which will in turn lead to a greater volume of patients being treated daily. This project aims to improve the QA by aiding in the development of fits that can be used for precise determination of the energy resolution of the beam, and also aiding in the automation of data processing/analysis tasks that will need to be carried out before the energy resolution can be determined. The results of this project have yielded a fit with  $\chi^2$  values that have a range of 2 – 5 and the project has also yielded scripts that automatically sort input data into sets that can be used for analysis and those that cannot be used.

# List of Figures

1.1	This graphic displays the various volumes that are of importance in conventional radiotherapy using X-Rays. As can be seen the irradiated volume is actually much larger than the Gross Tumour Volume which leads to many healthy cells being irradiated. (Taken from [7]) . . . . .	10
1.2	This figure depicts dose depositions of different particles as a function of the WEPL(Water equivalent path length)[17] . . . . .	11
2.1	This figure depicts the energy loss of multiple charged particles in air as calculated using Eq. 2.1. [22] . . . . .	13
2.2	This graphic displays a typical dose deposition pulse for a proton beam. The Bragg Peak can clearly be seen at the end of the signal. (Taken from [24]) . . . . .	14
2.3	This figure depicts the relative dominance of the photon interaction modes that take place with matter[25]. . . . .	15
2.4	This graphic displays a plot of the Bragg peak of three beams of varying energies travelling through polyethelene. As can be seen, an increase in energy leads to a greater penetration depth. However, the greater energy also leads to a larger amount of fragmentation taking place when compared with ionisation energy loss[23]. . . . .	16
2.5	This graphic displays a SOBP produced via the usage of absorbers with $n = 10$ leading to the required energy stacking[32]. . . . .	17
3.1	This graphic displays one of the detector modules for SuperNEMO. From left to right the module consists of a calorimeter wall containing photo-multiplier tubes, a tracker, the source foil, and then another tracker and calorimeter wall. (Taken from [36]) . . . . .	19
4.1	This figure depicts the Landau and Vavilov distributions for the non-relativistic case. The y axis depicts the probability of the detected number of photoelectrons corresponding to the energy I, with the $x$ axis depicting the number of photoelectrons detected[40] . . . . .	26

4.2	This figure depicts one of the ADC spectra produced from the experimental runs. The data taken is shown in blue, the Gaussian fit can be seen faintly around the peak in black, and the convolution fit is shown in red. .	28
4.3	This figure depicts a plot using the improved ROOT macro file with an equivalent range of $\pm 4\sigma$ but with four times as many calculated points in the convolution integral. . . . .	29
4.4	This figure depicts a plot using the original ROOT macro file with a range of $\pm 4\sigma$ . . . . .	29
5.1	This figure depicts an example of a waveform pulse that contains multiple peaks. As can be seen the peak that would be investigated would be the second, larger peak. Due to the initial smaller peak overlapping with the main peak it would be impossible to differentiate the two. . . . .	31
5.2	This figure depicts an example of an ADC spectrum that has been reproduced from scope waveform data. The data in question had 54 rejected pulses due to multiple peaks out of a total 9216 pulses. . . . .	34
6.1	This figure depicts the ADC spectrum produced from the original fit file for run 019. The wheel position for the acquisition of this data was 165. Note that this position is directly before the Bragg peak position of 166. A clear discrepancy can be seen throughout most of the curve, as demonstrated by the extremely large $\chi^2$ value. . . . .	36
6.2	This figure depicts the ADC spectrum produced from the improved fit file for run 019. The wheel position for the acquisition of this data was 165. Note again that this position is directly before the Bragg peak position of 166. Improvements have been made to the discrepancies, however they have not been completely removed. . . . .	36

# Contents

<b>Acknowledgements</b>	<b>3</b>
<b>Abstract</b>	<b>4</b>
<b>List of Figures</b>	<b>5</b>
<b>1 Introduction</b>	<b>9</b>
1.1 Cancer Treatment Modalities . . . . .	9
1.1.1 Surgery and Chemotherapy . . . . .	9
1.1.2 Radiotherapy . . . . .	10
1.2 Proton Beam Therapy . . . . .	12
<b>2 Proton Beam Therapy</b>	<b>13</b>
2.1 The Bethe Formula . . . . .	13
2.2 Interactions in Proton Therapy . . . . .	14
2.3 The Bragg Peak . . . . .	15
2.4 Imaging and Treatment . . . . .	17
<b>3 Methodology</b>	<b>19</b>
3.1 The Detector . . . . .	19
3.2 The Project . . . . .	20
3.2.1 Preliminaries . . . . .	20
3.2.2 ADC Data . . . . .	21
3.2.3 Waveform Scope Data . . . . .	22
3.2.3.1 Binary Conversion . . . . .	22
<b>4 ADC Spectra Analysis</b>	<b>24</b>
4.1 The Landau Distribution . . . . .	24
4.2 The Vavilov Distribution . . . . .	25
4.3 Fitting . . . . .	26
4.4 The Convolution Integral . . . . .	27
4.5 Improving the fit . . . . .	28
4.6 ADC Spectrum Results . . . . .	29
<b>5 Waveform Pulse Analysis</b>	<b>31</b>
5.1 Multi-peak Signals . . . . .	31
5.2 Waveform Pulse Fitting . . . . .	32
5.3 Reproduction of an ADC Spectrum . . . . .	33
5.4 Waveform Analysis Results . . . . .	33
<b>6 Conclusions</b>	<b>35</b>
6.1 ADC Analysis Summary and Further Work . . . . .	35

6.2	Waveform Analysis Summary and Further Work . . . . .	36
	<b>Bibliography</b>	<b>38</b>
	<b>A Results</b>	<b>42</b>
A.1	ADC Spectrum Results . . . . .	42
	<b>B Written Code</b>	<b>51</b>
B.1	ADC Spectrum Fit Macro . . . . .	51
B.2	Binary Data Conversion Jar . . . . .	61
B.3	Waveform Analysis Macro . . . . .	77



# Introduction

Within the UK alone in 2014 there were over 300,000 new cases of cancer. There were also over 100,000 deaths due to cancer within the same year. The survival rate for cancer (where survival is defined as living for 10 or more years) in 2010 – 11 was 50%. Meanwhile only 42% of these cases were preventable[1]. These figures alone demonstrate the necessity for improvement in cancer treatment methods. Cancers also have a high occurrence rate in the elderly [1], particularly hard to treat cancers such as pancreatic cancer[2]. Incidences of hard to treat cancers in both elderly and young patients have required the development of improved treatment modalities to help combat the increasing number of cases. These treatment modalities are required to be less invasive (to reduce the probability of side effects), more greatly localised (with regards to radiotherapy in particular to reduce the probability of a secondary cancer developing due to treatment), whilst also leading to better quality of life for the patient post-treatment.

## 1.1 Cancer Treatment Modalities

Conventional cancer therapy consists of three main modalities; radiotherapy, chemotherapy, or surgery. In general, with malignant cancers, the greatest success with treatment is achieved if multiple modes are used for therapy at once [3].

### 1.1.1 Surgery and Chemotherapy

Surgery is the excision of a tumour surgically. This is most commonly used if the cancer is localised and it is likely that all of the cancer can be removed. One of the major advantages of surgery compared to the other two modes is that there is very little chance of any long term side effects. However, some research has shown that there may be adverse effects on patients as surgery may cause perioperative tumour growth [4].

The next treatment is chemotherapy. This involves treatment of cancers via the use of drugs that have multiple effects on the biology of tumour cells that can interrupt their metabolism or their cell cycle. Chemotherapy has an advantage that it can target tumour cells specifically based on the cell type. However a current issue with chemotherapy is that some tumours are seemingly becoming resistant to the drugs used [5]. A major disadvantage of chemotherapy is that it involves usage of drugs that are metabolised (some of which may be neurotoxic[6]), therefore it is not restricted to treating a single area, but rather affects the whole body, thereby normally causing side effects in treated patients.

### 1.1.2 Radiotherapy

The final treatment mode is radiotherapy. Radiotherapy most commonly uses X-Rays, although other particles can be used, and has a significant advantage over chemotherapy in that it targets a smaller fraction of the patients body due to it being a localised form of treatment. However, radiotherapy is not cell specific in the same way as chemotherapy. Therefore, this can lead to healthy cells being damaged throughout the treatment process if they are in line with the beam path, because of the way that photons interact with matter (the dose deposition profile of the beam). The irradiation of healthy cells can then lead to complications or side effects later on in the patients life. The reason for this is that, when conducting radiotherapy using X-Rays, there are a variety of different volumes that need to be targeted to ensure that the tumour is treated correctly, as shown below in Fig. 1.1.

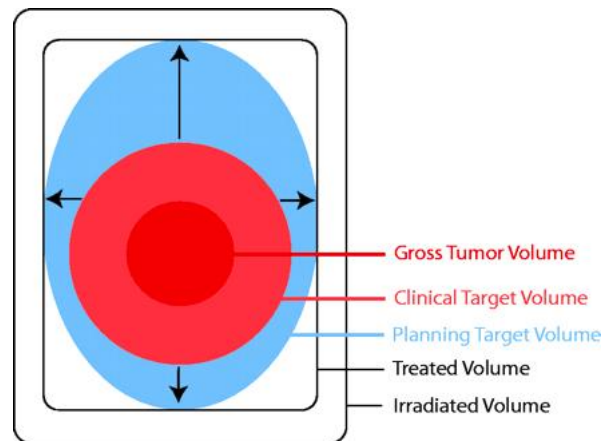


FIGURE 1.1: This graphic displays the various volumes that are of importance in conventional radiotherapy using X-Rays. As can be seen the irradiated volume is actually much larger than the Gross Tumour Volume which leads to many healthy cells being irradiated. (Taken from [7])

However because of the extra volume targeted, healthy cells are also killed. Conventional radiotherapy has been developed to produce modern methods such as Intensity Modulated Radiotherapy (IMRT)[8], Volumetric Modulated Arc Therapy (VMAT)[9] or Image Guided Radiotherapy (IGRT)[10], which reduce the amount of radiation delivered to healthy tissues.

IMRT involves the adjustment of the strength of the radiation beam as necessary via the usage of treatment planning software to increase the dosage delivered to the tumour, whilst reducing the dosage delivered to healthy cells surrounding the tumour[8]. VMAT involves the usage of a linear accelerator attached to a rotating gantry that delivers a three dimensional dose profile to the tumour, thereby ensuring there is a more precise

dosage delivery[9]. Finally, IGRT involves the usage of imaging throughout treatment to aid radiotherapists in locating the tumour, and also adjusting the treatment as necessary to account for tumour movement, such as for tumours in lung cancer[10][11]. Modern methods also involve the usage of fractionation, the splitting of treatment into multiple sessions, therefore reducing the overall dosage delivered in one session, which is beneficial to patients as it takes advantage of the fact that healthier cells will recover faster than cancer cells, whilst also sparing cells that are not as radiosensitive as tumour cells[12]. However with certain cancers, such as medulloblastoma[13] or prostate cancer[14], this can lead to serious complications as these areas of the body are near vital organs with tissues that are highly radiosensitive and thus may not recover from any damage inflicted by the therapy.

Photons also demonstrate an exponential decrease in energy as the depth of penetration increases (thereby leading to a lower dose as shown in Fig. 1.2) which means that the majority of the ionising radiation is deposited superficially and therefore higher doses would need to be given to treat tumours that are located deep within the body. Photons also do not stop passing through the body once the dose is deposited in the tumour, therefore leading to an exit dose [15] which can lead to a greater proportion of healthy tissue being damaged, however modern methods have begun to improve upon this[8][9] which is now leading to patients having a better quality of life post-therapy due to reduced toxicity[16].

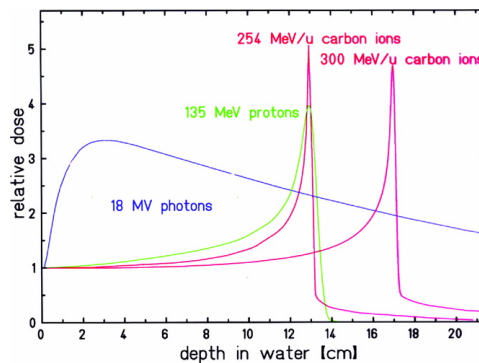


FIGURE 1.2: This figure depicts dose depositions of different particles as a function of the WEPL(Water equivalent path length)[17]

An issue that is currently present in all forms of conventional radiotherapy is that, in general, enough imaging does not take place. An initial CT scan will be conducted on the patient to image the tumour, with less precise scans carried out at intervals throughout the treatment to monitor the progress. This means that throughout treatment, any movement by the patient or movement of the tumour will affect the treatment area and can therefore lead to the probability of side effects being increased, especially with

cancers such as prostate cancer or lung cancer. However, this has been improved upon using methods such as IGRT[10], and can also be further improved using Proton beam therapy.

## 1.2 Proton Beam Therapy

Proton beam therapy is another form of radiotherapy that is currently used for hard to treat cancers (e.g. ocular melanoma)[18]. The procedure involves using an accelerator which provides a large number of protons which possess a very specific energy. The energy of the protons produced via this method may also be roughly mono-energetic which is beneficial in reducing the dosage to other tissue outside the site of the tumour[19]. This procedure has been present for some time[20] and has advanced significantly to become a prescribed treatment method. Protons are considered a form of hadron therapy, which involves the usage of protons, neutrons, or other heavy ions[21]. The usage of heavy ions provides significant advantages over the use of photons and light ions (e.g. electrons) due to the narrower scattering of the particle[20] as, unlike photons which interact via the Compton effect[21], heavier ions do not undergo such large amounts of scattering, therefore losing less energy as they slow whilst travelling through matter[21].

# Proton Beam Therapy

## 2.1 The Bethe Formula

The main advantage with using protons is the depth of deposition of the energy is intrinsically linked to the energy of the protons as they enter the body. This relationship is described by the Bethe (Eq. 2.1) formula, where  $E$  is energy,  $x$  is distance into the target,  $v$  is the particle velocity,  $z$  is the particles charge in multiples of electron charge,  $n$  is electron number density,  $I$  is mean excitation potential,  $\beta$  is  $\frac{v}{c}$ ,  $c$  is the speed of light,  $\epsilon_0$  is vacuum permittivity,  $e$  is electron charge, and finally  $m_e$  is the electron rest mass. This demonstrates that, to first order, the penetration depth is directly proportional to velocity and therefore energy.

$$-\left\langle \frac{dE}{dx} \right\rangle = \frac{4\pi}{m_e c^2} \frac{n z^2}{\beta^2} \left( \frac{e^2}{4\pi\epsilon_0} \right)^2 \left[ \ln \left( \frac{2m_e c^2 \beta^2}{I(1 - \beta^2)} - \beta^2 \right) \right] \quad (2.1)$$

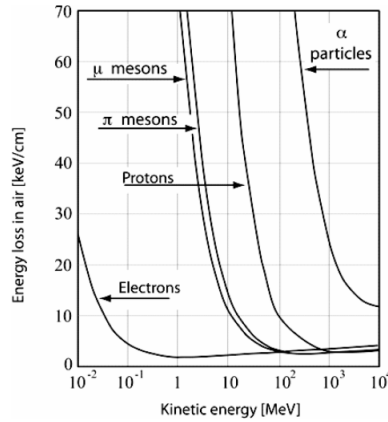


FIGURE 2.1: This figure depicts the energy loss of multiple charged particles in air as calculated using Eq. 2.1. [22]

Proton beam therapy improves upon one of the significant issues still present within conventional radiotherapy where a proportion of healthy tissue is irradiated in addition to the tumour. This benefit is not limited to protons, but applies to all charged particles, and can be seen in 2.1. However, as seen in Fig. 1.2 and Fig. 2.2 protons have a sharp Bragg peak (discussed in greater detail in Sec. 2.3). The heavier the ion the sharper the peak; however, as can be seen in Fig. 1.2, heavier ions have a longer trailing tail therefore leading to a deposition of energy in healthy cells beyond the tumour. This however is not described by the Bethe formula but is due to the fact that heavier ions also undergo nuclear interactions[23].

## 2.2 Interactions in Proton Therapy

Various interactions take place when using heavy ions for radiotherapy that are significantly different to those that take place when using photons. The most significant interaction modes for photons are dependant on the energy of the incident photon and the atomic number of the incident material[25]. For low energies, the photoelectric effect is dominant, the Compton effect dominates for intermediate energies, and pair production dominates for high energies, as shown in Fig. 2.3[25]. Protons (and other heavy ions) also undergo ionisation interactions where energy is imparted from the projectile ion to the orbital electrons of the atoms in the target material via Coulomb interactions. These interactions are correctly defined by the Bethe formula and the formula correctly accounts for the increase in the Linear Energy Transfer (LET) until the energy of the incoming ion decreases below a threshold, after which a significant decrease in the LET is seen[23][26]. This significant decrease in the LET is known as the Bragg peak and occurs approximately before the ions come to rest[24].

Heavy ions also undergo nuclear interactions that are ultimately described by quantum chromodynamics (QCD) as it is a many body problem involving the composite particles of the nucleons themselves (quarks)[23]. However due to the many body nature of this problem, computational modelling is difficult to conduct, therefore multiple semi-empirical models have been produced[27]. In nuclear fragmentation, the incoming particle can cause two types of fragmentation, depending on if the incoming ion contains only one particle or if it is a nucleus.

The two types are "projectile" and "target" fragments[23]. Projectile fragments are simply particles that are formed from the incident nucleus after a collision has taken place. These fragments commonly retain a significant fraction of the velocity of the incident ion. However, there may be some slight deviations in the velocity, with the change

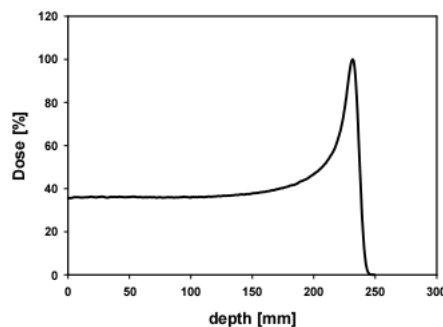


FIGURE 2.2: This graphic displays a typical dose deposition pulse for a proton beam. The Bragg Peak can clearly be seen at the end of the signal. (Taken from [24])

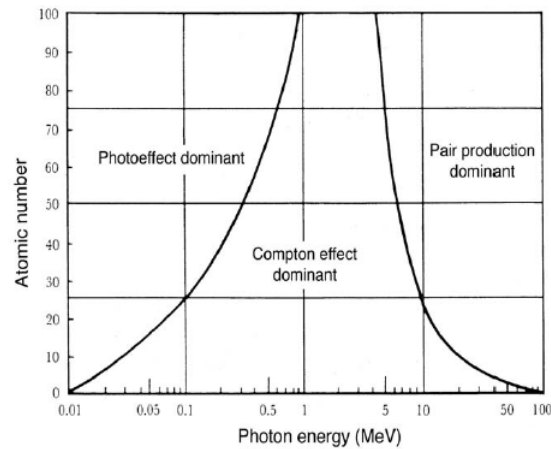


FIGURE 2.3: This figure depicts the relative dominance of the photon interaction modes that take place with matter[25].

in the Cartesian coordinate being modelled by a normal distribution[23][27]. Target fragments are formed from the expulsion of nucleons from atoms within the target material. These fragments, which may be charged or neutral (which may penetrate long distances), will emerge from the struck nucleus potentially in an excited state. They will then decay into the ground state, causing further emission of particles. These fragments are emitted isotropically in the rest frame and can affect the dose, especially if neutrons are emitted[23]. If neutrons are produced then this further affects the dosimetry as they may cause the production of further high LET secondary particles[23].

The target fragments have a significant effect on dosimetry as they lead to the changing of the particle fluence of the initial incoming beam from a beam of pure primary ions into a beam of mixed heavy and light ions that may also contain neutrons. This can then lead to a broader energy distribution which is an explanation for the broad straggling tail that is seen in energy distribution plots as shown in Fig. 2.2[28]. There are significant differences between using heavy ions such as Carbon atoms, and light ions such as protons. Foremost amongst these is that there is an increased LET for heavier ions (as shown in Fig. 2.4) which leads to an increase in relative biological effectiveness. Also the heavier the ion, the higher the probability of inelastic nuclear interactions taking place, leading to a greater chance of fragmentation, again shown in Fig. 2.4[28].

## 2.3 The Bragg Peak

The final aspect of proton beam therapy that must be described further is the Bragg peak. A monoenergetic beam of protons will produce a Bragg peak at a specified depth, as has been shown previously. However, a monoenergetic beam is not sufficient for

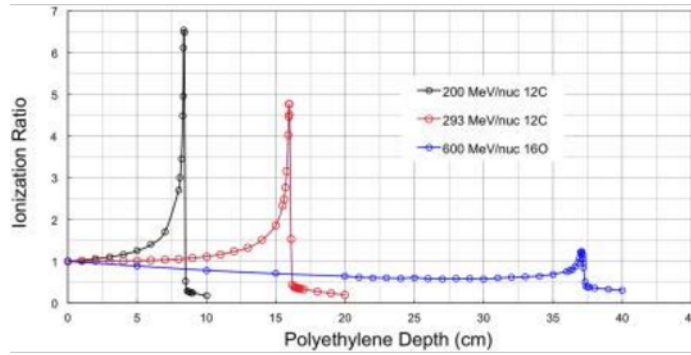


FIGURE 2.4: This graphic displays a plot of the Bragg peak of three beams of varying energies travelling through polyethelene. As can be seen, an increase in energy leads to a greater penetration depth. However, the greater energy also leads to a larger amount of fragmentation taking place when compared with ionisation energy loss[23].

cancer treatment due to the narrow longitudinal Bragg peak[29]. A solution to this is the application of a spread out Bragg peak (SOBP) which allows uniform dosage to be applied within the target volume by providing a weighted energy distribution as the incident beam[29]. The weighting of the individual proton beams is determined using the power law relationship that relates the range  $R$  of the protons to their initial energy  $E(z = 0) = E_0$  as shown in Eq. 2.2.

$$R = \alpha E_0^p \quad (2.2)$$

where  $\alpha$  is roughly proportional to  $\sqrt{A_{eff}}$  with  $A_{eff}$  being the effective atomic mass of the absorbing medium. This is known as the Bragg-Kleeman rule[30] and it should be noted that  $\alpha$  is also inversely proportional to the medium's mass density[31]. The exponent  $p_0$  defines the shape of the curve of Eq. 2.2 and also determines the energy range for which Eq. 2.2 is valid. For example, a value of  $p = 1.5$  (known as Geiger's rule) is valid for protons with energies  $\leq 10 MeV$ , whilst for  $10 < E_0 < 250 MeV$  a value of  $p \approx 1.8$  is more appropriate[31]. Research carried out by [31] determined that, for the most relevant energies ( $0 \leq E_0 \leq 200 MeV$ ) the best fit values of the constants are, for a water phantom;  $p \approx 1.77$  and  $\alpha \approx 0.0022$ . In order to produce a SOBP, the range  $\chi$  of the SOBP must be determined (which will be equivalent to the tumour width, length, and depth for a 3d volume). Once determined,  $\chi$  will be divided into  $n$  equal intervals where  $n$  is the number of beams to be used. The SOBP will then extend from  $(1 - \chi)R_0$  to  $R_0$ , where  $R_0$  will be the depth of the highest energy proton beam (and the edge of the SOBP)[29]. The ranges of each beam are then given by Eq. 2.3, with their respective energies given by Eq. 2.4 (which is Eq. 2.2 rearranged for  $E_k$  with the constants possessing the same definition as in Eq. 2.2), and the normalised weights of the beams given by Eq. 2.5 where  $p$  is used instead of  $p_0$  to allow the variable to vary[29]. The weights calculated using this method define the fractional weighting of



a proton beam possessing an initial energy as defined by the parameter  $p_0$ . It is of significance to note that the value of  $p_0$  also affects the value of  $\chi$ [29].

$$r_k = \left[ 1 - \left( 1 - \frac{k}{n} \right) \chi \right] R_0 \quad (2.3)$$

$$E_k = \left( \frac{r_k}{\alpha} \right)^{\frac{1}{p_0}} \quad (2.4)$$

$$w_k = \begin{cases} 1 - \left( 1 - \frac{1}{2n} \right)^{1 - \frac{1}{p}}, & \text{for } k = 0 \\ \left[ 1 - \frac{1}{n} \left( k - \frac{1}{2} \right) \right]^{1 - \frac{1}{p}} - \left[ 1 - \frac{1}{n} \left( k + \frac{1}{2} \right) \right]^{1 - \frac{1}{p}}, & \text{for } k = 1, \dots, n - 1 \\ \left( \frac{1}{2n} \right)^{1 - \frac{1}{p}}, & \text{for } k = n. \end{cases} \quad (2.5)$$

Fig. 2.5 depicts an example of a SOBP produced using absorbers of varying widths to transform protons from a monoenergetic beam into protons with varying energies weighted by Eq. 2.5 that will then sum to produce the depicted SOBP[32]. It can clearly be seen that a beam with this dose distribution would be preferable over the standard Bragg peak distribution as uniform dose can be applied over a predetermined range.

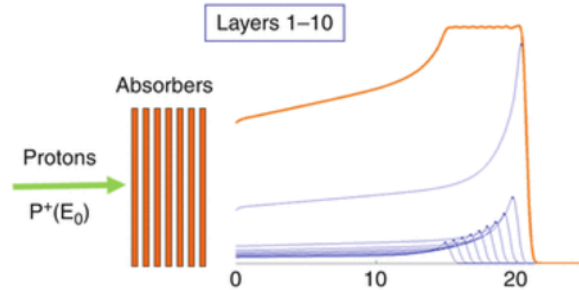


FIGURE 2.5: This graphic displays a SOBP produced via the usage of absorbers with  $n = 10$  leading to the required energy stacking[32].

## 2.4 Imaging and Treatment

If used correctly, the Bragg peak can allow for better treatment of tumours due to the fact that the LET significantly decreases after the Bragg peak, allowing for a greater amount of the ions energy to be deposited within the tumour volume, with a minimal amount of dose leakage due to secondary particles produced by fragmentation. However, to carry out this treatment, precise imaging must take place to correctly determine the depth of the tumour within the patient as the energy of the proton is directly related to the penetration depth as described by Eq. 2.1. If a detector was developed that could

process high rates of protons efficiently and with a very precise energy resolution then this detector could also be used as a scanner by using a beam of higher energy protons to image the tumour as a form of proton computed tomography (pCT) with the patient located in situ[33][34]. A prototype proton computed radiography (pCR) device has been tested and can be developed further to produce a pCT device[35]. The method of producing a CT scan using protons would be a similar process to that used for x-ray CT scanning, albeit slightly more difficult due to the fact that protons will undergo multiple coulomb scattering because of their charge[35]. These areas are key advantages of using protons rather than photons for radiotherapy as, although imaging takes place in IGRT and IMRT throughout treatment using CT scanners mounted to the gantry[9], this does not remove the issue of exit dose leakage due to the photons.

A current issue with proton beam therapy centres is that a large proportion of time is spent setting up the equipment to ensure that the correct depth is reached by the protons for an assortment of energies. These are the quality assurance (QA) procedures that provide assurance that the beam is functioning correctly and are essential for determining that the energy is being deposited at the site of the tumour rather than anywhere else. The aim of this project is to reduce the amount of time spent carrying out the QA procedures by leading to the development of an advanced detector that will allow varying energy measurements to be carried out at a greater rate.

# Methodology

The group is directly involved with the development of the proton beam therapy centre at UCLH. The beam design that has been proposed is a narrow "pencil" beam that will allow for a better conformal dose to be delivered to the tumour volume. The dosage will be delivered throughout the whole tumour volume by scanning the pencil beam across the target tumour. The group is currently focussed on the development of a new detector (adapted from the SuperNEMO detector), and also the improvement of current procedures to allow for faster and more accurate QA measurements that are carried out multiple times throughout a treatment day.

## 3.1 The Detector

The detector that is being developed has been adapted from the SuperNEMO experiment, which in turn was developed from the NEMO3 experiment. These detectors used a tracker-calorimeter technique to identify particles and determine their energy [36]. A schematic of one of the detector modules is shown in Fig. 3.1. The tracker modules are used primarily for particle identification. They consist of numerous drift cells that operate in Geiger mode [37].

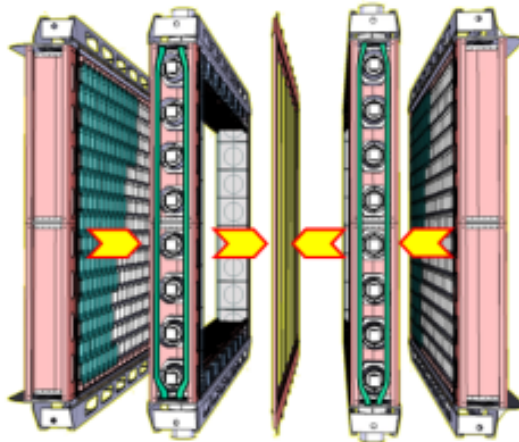


FIGURE 3.1: This graphic displays one of the detector modules for SuperNEMO. From left to right the module consists of a calorimeter wall containing photomultiplier tubes, a tracker, the source foil, and then another tracker and calorimeter wall. (Taken from [36])

The next module of the detector that is of importance with regards to Proton Beam Therapy is the calorimeter. The original NEMO3 calorimeter units consisted of a scintillator which was coupled to low radioactivity photomultiplier tubes and also included

lightguides [38]. The scintillator material was polystyrene which was chosen because it would reduce back-scattering of low energy electrons due to polystyrene containing a low  $Z$  value when compared to other scintillators [39]. The SuperNEMO calorimeter design is slightly different. Rather than using polystyrene as the scintillator material it instead uses polyvinyl-toluene and does not make use of lightguides[37]. The main function of the scintillator is to determine the energy of incoming particles via measurement of light that is emitted due to the collision of the incoming charged particles causing photon emission from the scintillator material molecules.

The calorimeter is the part of the detector that is the focus of this project. The energy resolution of the calorimeter is the parameter that is of importance as, a small energy resolution means that the energy of the protons is known with great precision and therefore implies better QA.

## 3.2 The Project

The overarching task that was conducted throughout this project was the data analysis of the current data taken by the Proton Beam Therapy group throughout August 2016 at the Clatterbridge therapy centre. This involved the development of scripts that can automatically analyse pulses to fit and produce required data (such as the energy resolution of the beam), reject "bad" pulses from the analysis, and scripts to convert data files into readable formats.

### 3.2.1 Preliminaries

The initial tasks of the project involved the installation of Scientific Linux, in order to allow for the usage of ROOT. ROOT is a program that contains a large number of libraries that are useful for the statistical analysis of large data sets in the High Energy Physics field. Upon installation of the operating system, it then had to be set up with the required packages that were necessary to install ROOT. This required multiple days to complete before the ROOT installation could be carried out. Once ROOT was successfully installed, various tutorials were used to gain the required familiarity with it to begin the analysis of the data. A secondary task that was also conducted was the determination of if ROOT was the most appropriate package to be used for the data analysis. A competitor was the software package PyROOT. It was determined that ROOT would be used as, in essence, PyROOT is simply a Python wrapper for the ROOT libraries. This means that developing in PyROOT rather than ROOT would simply be a matter of preference of using Python over C++ however, as there were no issues with C++ (and previous analysis had been done in ROOT) it was unnecessary to change software

packages.

Once the software environment had been established, the next step was to understand the raw data files that were being analysed. Data from the beamline was taken via two distinct methods. The first method was via the usage of a Caen Analog-Digital converter (ADC), and the second method involved the usage of a LeCroy oscilloscope. Each of these will be spoken about in greater detail below.

### 3.2.2 ADC Data

The Caen Analog-Digital converter reads the response of the detector and automatically combines the individual pulses recorded within a range specified by the user by carrying out the charge integration to produce an ADC spectrum from the data which can then be fitted to in order to determine the energy resolution from the full width half maximum (FWHM) of the curve. The charge integration was carried out over a specified "long gate" which may span up to  $150ns$ .

The gate was defined as starting immediately before the trigger time, the time at which the proton is being detected by the detector, and ended shortly after the voltage returns to the baseline value. The integration was carried out automatically by the converter over the specified gate. This method had a significant drawback as, the converter itself is not capable of determining if there is a pile up of signals within the detector. A pile up is when multiple particles are detected at once. This can lead to multiple peaks being recorded and integrated in the same long gate, which can lead to an imprecise value of the energy resolution being calculated.

The ADC data was taken with a high voltage of  $-900V$ , this remained constant for all data sets taken. A  $1.98mm$  collimator was also used. The variables that were changed throughout the data collection process were: the rate, some of the electronic setup data, the absorber size, and the arc current. The rate corresponds to the number of protons making up the beam. This was varied to determine the response of the detector to both high and low rates as, one of the aims is to produce a detector that is capable of processing high proton rates. The absorber size is also a relevant variable that was varied. This corresponds to the modulator wheel position that is placed in front of the beam to attenuate the energy of the beam. Each segment of the wheel is  $4mm$  in thickness, therefore a wheel position of 10 would correspond to  $40mm$  thickness. In some cases a block of PMMA was used instead of the wheel.

### 3.2.3 Waveform Scope Data

A LeCroy HDO6000 oscilloscope was used for the collection of the waveform data. The waveform pulse data was also taken with similar parameters to the ADC spectrum data. The waveform data is a record of the charge variation in the detector with respect to time. This data is recorded by the scope, however, because of the high rate of protons that are detected, the scope does not have the capability of recording the waveform data in ASCII format. Because of this, the data taken for all recorded waveforms was consolidated into a single binary file to allow for efficient data recording. However, this presented significant issues that had to be dealt with before the analysis of the data could take place.

Foremost amongst these issues was the fact that, because of the consolidation, the recorded timebase was a continuous spectrum. There was no distinction within the binary data that would allow for easy determination of when one waveform ended and another began. This presented an issue because, in order to carry out the analysis it would be necessary to read the waveform pulses into ROOT individually to be analysed. A secondary issue was that the oscilloscope itself also cannot distinguish between waveforms containing multiple peaks and those with only a single clear peak. This means that multiple waveform files exist within the binary file that also exhibit the piling up of the pulses that was discussed in the previous section.

#### 3.2.3.1 Binary Conversion

In order to analyse the waveform data, a program had to be written with multiple functionalities. The first required function was the efficient conversion of the binary data into an ASCII format that could be input directly into ROOT without any need for further conversion. Using a template provided by LeCroy, a Java program was written that would allow the data to be converted. This program took advantage of various addresses that were defined in the header of the binary file that detailed the location of any relevant information that was required for the conversion e.g. the horizontal interval of the timebase etc. Despite the template provided by LeCroy, there are a number of variables with defined addresses that are unknown quantities.

The second required functionality was the splitting of the single binary file into individual ASCII files for every waveform file contained. The difficulty with this, as mentioned, was the continuous timebase spectrum. In order to split the files, the number of lines stored in the binary file was determined from the header, and the approximate length

(in lines) of each pulse was determined from the average time at which the trigger would end and return to the baseline. This information was then used to split the binary file into individual data files with the timebase reset for each file and spanning  $0 - 400ns$  with intervals of  $0.4ns$ . An additional function that was included was the option to specify the number of files to be converted from the binary file. This was included to allow for initial plotting of the pulses to be undertaken to determine the quality of the files included in the data. The current version of the program is depicted in [Appendix B.2](#). Upon completion of the program, the binary files were converted with each file containing 127000 individual pulses and taking  $1hr$  to completely convert.

# ADC Spectra Analysis

The ADC data is stored in simple text files that contains the number of events for each ADC count record as recorded by the digitiser. This data is imported by ROOT to produce a simple histogram depicting the data. The code used to import the data is based on a previously used ROOT macro that has been adapted as shown in Appendix B.1. Initially, the code was studied to better understand how it functions and to also precisely determine the underlying issues that were affecting the results. It was found that the issue with the fitting was that the fit was not able to span the whole range of the data as it could not precisely model the straggling edge of the spectrum. This issue with the fit not being applicable to the whole spectrum was noticed as the calculated  $\chi^2$  value would continuously worsen as the range of the fit was increased. Because of this issue, the range of the fit in the original macro was restricted to  $\pm 2\sigma$

## 4.1 The Landau Distribution

Before discussing the fitting, it is important to mention the probability distributions that are used. With regards to the passage of ionising particles through matter, there are three main distributions that are relevant as, the process of energy loss via ionisation of particles is a statistical one, and therefore can only be described by probability distributions. The interaction processes that have been previously described can significantly affect the resolution of the calorimeter as these interactions may take place in the active layers with these interactions causing significant fluctuations in the energy deposited within these layers.

The probability function that describes the probability ( $f(E, x, \Delta)$ ) of a particle of energy  $E$  passing through a medium with thickness  $x$  to lose energy that is between  $\Delta$  and  $\Delta + d\Delta$  must satisfy the kinetic equation 4.1 where;  $w(\epsilon)$  is the probability per unit length for a particle with energy  $E$  to lose an amount of energy  $\epsilon$  during a collision[40].

$$\frac{\partial f(E, x, \Delta)}{\partial x} = \int_0^b w(E + \epsilon, \epsilon) f(E + \epsilon, x, \Delta - \epsilon) d\epsilon - f(E, x, \Delta) \int_0^{\epsilon_{max}} w(E, \epsilon) d\epsilon \quad (4.1)$$

The Laplace transform of the energy loss distribution satisfying Eq. 4.1 can be described by Eq. 4.2 where  $F$  is the probability density function, and  $t$  is the time at which the collision takes place[40].

$$\phi(t) = \ln F(x, t) = -x \int_0^{\epsilon_{max}} w(\epsilon) (1 - \exp^{-t\epsilon}) d\epsilon \quad (4.2)$$



Eq. 4.2 was solved by Landau by using the collisional probability Eq. 4.3[41].

$$xw(\epsilon) = \frac{\xi}{\epsilon^2} \quad (4.3)$$

where

$$\xi = \frac{2\pi Nq^2q_e^4\rho x \Sigma Z}{m_e\beta^2 \Sigma A} \quad (4.4)$$

In Eq. 4.4;  $N$  is Avogadro's number,  $m_e$  and  $q_e$  the mass and charge of the electron respectively,  $\beta$  the velocity of the incident particle and  $q$  its charge,  $\rho$  the density of the medium, and finally  $Z$  and  $A$  the atomic mass and number of the component elements of the target material. An assumption that affects this solution is that the incident particles are highly relativistic. This is described by allowing the integral of Eq. 4.2 to have an upper limit of infinity[41]. However, due to this assumption, the Landau distribution does not have a finite mean or standard deviation[40]. The assumption leads to the inclusion of a non-physical contribution where the incident particle can have an infinite energy transfer to the electrons of the medium. Because of this the Landau distribution must have a specified cut off when being calculated to prevent unphysical results from being obtained[40].

## 4.2 The Vavilov Distribution

It is also of note that another probability distribution exists that improves upon the work of Landau by removing the assumption that the incident particle is highly relativistic[42]. This distribution, known as the Vavilov distribution, is suitable for non-relativistic particles and uses the collision probability Eq. 4.5 rather than Eq. 4.3[40].

$$xw(\epsilon) = \frac{\xi}{\epsilon^2} \left(1 - \beta^2 \frac{\epsilon}{\epsilon_{max}}\right) \quad (4.5)$$

where  $\epsilon_{max}$  is the maximum energy that can be transferred within a collision. This distribution can be considered to be more precise due to the limitation of the upper bound of the integral in Eq. 4.2. Both distributions are complex, however the Vavilov distribution requires an integral to be calculated in the complex plane which leads to it being more computationally intensive to calculate when compared to the Landau distribution. It should also be noted that for large values of  $\epsilon_{max}$  the Vavilov distribution tends towards the Landau distribution[42].

The introduction of the significance parameter  $\kappa = \frac{\xi}{\epsilon_{max}}$  allows an inequality to be introduced to approximate when either distribution should be used. If  $x$  is large then  $\kappa$  will also tend to increase, and vice versa if  $x$  is small. The respective inequalities are; if

$\kappa < 0.01$  then the Landau distribution is applicable, and if  $0.01 < \kappa < 10$  the Vavilov distribution is applicable. A Gaussian distribution may also be used to approximate the energy loss distribution if  $\kappa > 10$  (for example when the mean energy loss within the scintillator material is greater than the maximum energy transfer in a single collision). It is of note that the value of  $\kappa$  is directly dependant on the thickness of the absorber in question. A Gaussian distribution is most appropriate for thick absorbers whilst the Landau and Vavilov distributions are more applicable for thin absorbers, as is the case for the detector used by the group where the main absorber is a thin film of Mylar[43]. An example depicting both the Landau and Vavilov distributions can be seen in Fig. 4.1. As can be seen, the Landau distribution has a long, straggling tail whereas the Vavilov distribution has a more pronounced cutoff point. The  $\kappa$  value was calculated for the Mylar of the detector, using the AutoZeff software to determine the relative molecular mass of the substance. The subsequent calculation led to the conclusion that, for the thickness of Mylar used in the detector, the Landau distribution is currently the most appropriate.

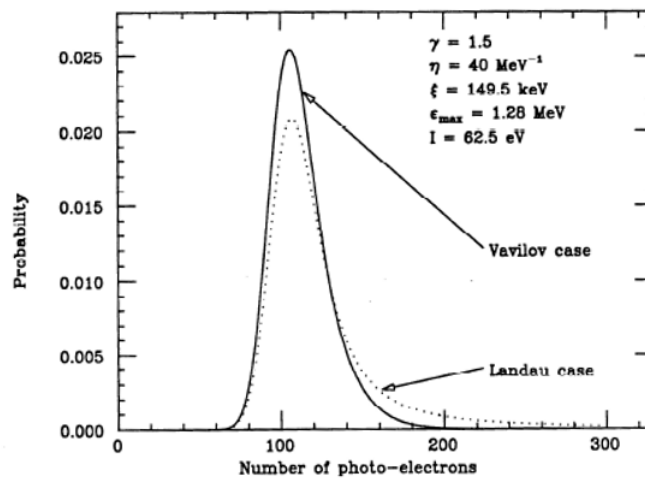


FIGURE 4.1: This figure depicts the Landau and Vavilov distributions for the non-relativistic case. The y axis depicts the probability of the detected number of photoelectrons corresponding to the energy  $I$ , with the  $x$  axis depicting the number of photoelectrons detected[40]

### 4.3 Fitting

The fitting of the data involves the integration of the Landau distribution. This distribution is used in conjunction with a Gaussian distribution to compute a Landau-Gaussian convolution integral for each  $x$  coordinate in the spectrum. Initially, a Gaussian fit is applied over a small range of the spectrum with the normal being set to the height of

the peak of the number of events, the  $\mu$  being set to the ADC counts number of the peak value, and the  $\sigma$  being set to  $\sqrt{\mu}$ . These initial values are used to determine the  $\sigma$  value of the distribution as ROOT varies the parameters of the initial fit in order to reduce the  $\chi^2$  value of the initial fit. However, it is of note that the initial is only carried out over  $\pm 2\sigma$  with  $\sigma$  being the value previously specified. Therefore, the initial fit is only accurate over the width of the peak and does not apply to the data values beyond the peak.

For the final Landau-Gaussian convolution fit, there are 6 parameters; the width of the left hand side of the distribution (which is set to 1 by default with no variation), the most probable location of the parameter describing the Landau density (which is set to the  $\mu$  value calculated from the parameter variation of the initial Gaussian fit), the left hand normal of the distribution (which is set to the value of the distribution peak as calculated from the initial fit), the  $\sigma$  of the full distribution (which is set to the value calculated from the initial fit), the width of the right hand side of the distribution (which is set to 1 initially), and finally the right hand normal of the distribution (which is initially sent to 100). Parameters 3 and 4 also have specified limits of 0 – 1000 and 0 – 100000000 which defines the limits of variation that ROOT will conduct for the parameters.

## 4.4 The Convolution Integral

The Landau-Gaussian fit makes use of a convolution integral. This mathematical technique is defined as shown in Eq. 4.6 where;  $f$  and  $g$  are functions of the variable  $x$  (where, for the purposes of this fitting  $x$  is the ADC counts number), and  $\tilde{x}$  is a dummy integration variable that is used to define an arbitrary translation of the function  $g(x)$ .

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\tilde{x})g(x - \tilde{x})d\tilde{x} \quad (4.6)$$

The concept of the convolution integral is to produce a third function,  $h(x)$ , that may be viewed as being the area overlap between the function,  $f(x)$ , and the mirror of the function,  $g(x)$ , where, with regards to our fit,  $f(x)$  is the Gaussian distribution of the ADC counts, and  $g(x)$  is the Landau distribution of ADC counts.

The reason that this fit is used is based on the characteristics of the curves that have been produced from the experimental data as shown in Fig. 4.2. From Fig. 4.2, it can clearly be seen that the area around the peak of the curve approximates a Gaussian distribution. This is also even more apparent as the initial fit of the Gaussian around

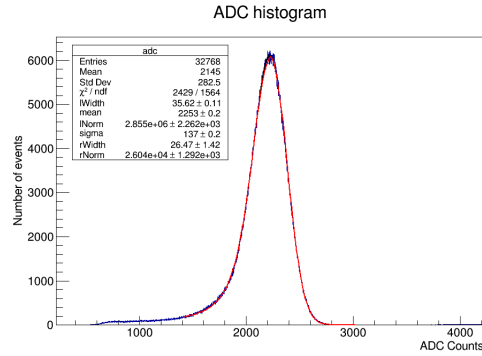


FIGURE 4.2: This figure depicts one of the ADC spectra produced from the experimental runs. The data taken is shown in blue, the Gaussian fit can be seen faintly around the peak in black, and the convolution fit is shown in red.

the peak very closely follows the data. The long tail that begins the curve is also reminiscent of the long straggling tail that is present in the Landau distribution, as seen in Fig. 4.1, hence why the Landau distribution is used as the mirror function in the convolution integral.

## 4.5 Improving the fit

As previously mentioned, the original fit being used was not able to span a large range of the fit. In order to rectify this issue, the original fit was thoroughly studied to determine if any mistakes had been made mathematically that may have affected the calculation of the convolution. The initial fit was also studied to determine if the initial parameters used could be determined more precisely which would allow the final fit to converge faster. The initial fit was adjusted, with a ROOT function being used to determine the absolute location of the peak of the data. Various mathematical changes were also made to the method used to calculate the convolution integral e.g. the addition of an extra Gaussian term in order to account for an issue where the peak of the fitting curve did not align with the peak of the fitted data, however this led to little improvements.

A final improvement attempt was to increase the precision of the convolution integral by rewriting the macro such that for each individual ADC counts point, between the upper and lower bounds of the integration, the numerical integration was divided into a larger number of slices, therefore leading to a more precise result being produced for the value of  $h(x)$ .

## 4.6 ADC Spectrum Results

An example of a fit using the improved macro is shown in Fig. 4.3 with an original fit depicted in Fig. 4.4 for reference. The three main characteristics of the ADC spectra are; the peak, the straggling curve at the beginning of the curve, and the shorter decay curve at the end. Fig. 4.3 clearly depicts an improvement in the fit at all three points of the curve. There is no deformity in the fitted peak when compared to that shown in Fig. 4.4. The fitted curve also more closely matches the experimental data towards either edge of the curve. The improvement in the fitting is also clearly indicated by the fact that the  $\chi^2$  value has decreased by a factor of almost 2.4 (from 3.7 to 1.6). It is of note that, although the fit has significantly improved, the greater amount of calculations required is computationally intensive, leading to each fit taking almost twice as much time as was previously required. The fit was applied to all data taken from the August 2016 Clatterbridge run, with a clear improvement in the goodness of fit demonstrated in all of the fitted data when compared to the original fit over the same range. A fit could also be produced for a larger range of the data, indicating an improved fit.

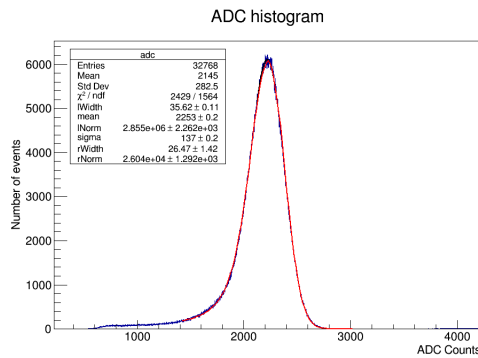


FIGURE 4.3: This figure depicts a plot using the improved ROOT macro file with an equivalent range of  $\pm 4\sigma$  but with four times as many calculated points in the convolution integral.

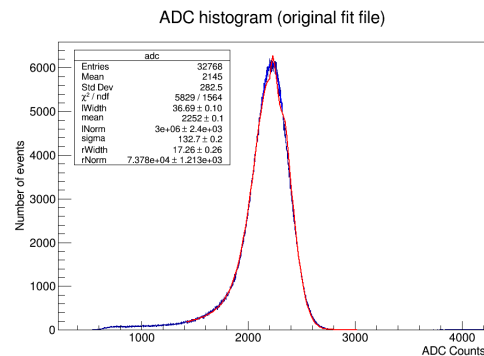


FIGURE 4.4: This figure depicts a plot using the original ROOT macro file with a range of  $\pm 4\sigma$ .

$$Res = 2.35 \cdot 100 \cdot \frac{\sigma}{\mu} \quad (4.7)$$

$$\Delta Res = \sqrt{\frac{\Delta \mu^2}{\mu^2} \cdot \frac{\Delta \sigma^2}{\sigma^2}} \quad (4.8)$$

The energy resolution is calculated from the fit using Eq. 4.7 and has an associated error as shown in Eq. 4.8. The changes that were made to the fit led to the energy

resolution of the fitted files increasing but with a decrease in the associated error of the energy resolution. This indicates that the previously calculated energy resolutions were not as precise as possible due to limitations of the fit.

# Waveform Pulse Analysis

Upon conversion of the binary files, work could begin on the analysis of the waveform data. The waveform data was stored in individual files with the only two columns retained from the binary file being the timebase value and the voltage value. The units for each of these readings was  $ns$  and  $mV$  respectively however, the data was stored as integers rather than in standard form to allow for efficient storage. The number of decimal points was also conjugated to prevent the storage of additional numbers that were due to the conversion from a binary value to a floating point number. A ROOT macro was then written that was capable of reading each waveform in individually and producing a plot to depict the data. The macro also determines the peak voltage value and then loops over all the voltage values, adding the magnitude of the peak value to each one to make the signal a positive one for convenience. The key work that needed to be done on the waveform data was the automation of a technique that could be used to detect "bad" waveforms.

## 5.1 Multi-peak Signals

Multi-peak signals occur when there are multiple particles triggering the scintillating material, leading to the detection of multiple peak signals by the detector. These multi-peak signals can be seen in data taken from a wide range of rates. The reason they are considered to be bad is because they cannot be easily fit to and also are not easily integrated to allow for the reproduction of an ADC spectrum as it is not clear when one of the pulses ended and the second or third one began. An example bad peak is depicted in 5.1. As can be seen there is a very distinct overlap between the two signals, which further demonstrates the difficulty of either fitting or integrating this data. In order

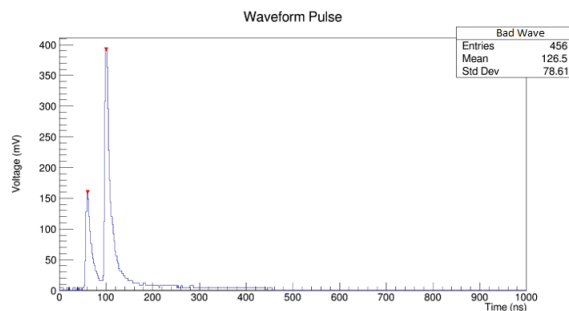


FIGURE 5.1: This figure depicts an example of a waveform pulse that contains multiple peaks. As can be seen the peak that would be investigated would be the second, larger peak. Due to the initial smaller peak overlapping with the main peak it would be impossible to differentiate the two.

to develop a script that would automatically detect and remove these peaks, research

was conducted into the built in statistical libraries that ROOT possesses in order to determine if any of the methods were relevant in this area. The required method would need to have a variable argument to account for the possibility that the peaks may not be separated by a standard amount. The research conducted yielded the TSpectrum class in ROOT. This class has a search algorithm that makes use of the Markov chain algorithm taken from [44] that were developed to find  $m$  peaks from an  $m$ -dimension spectrum. The algorithm searches for peaks based on two user specified variables; the  $\sigma$  of the peaks to be searched, and a threshold value  $T$ . The  $\sigma$  value details the distance from the main peak that the algorithm will search up to for any secondary peaks whilst the  $T$  value details how large a peak should be, as a fraction of the largest peak, before it is discarded. These values may vary based on the waveform data that is being processed, however, thus far it seems that a value of  $\sigma = 1$  and  $T = 0.5$  is sufficient to determine any secondary peaks. Values of  $T < 0.5$  led to points in the curve being detected as peaks regardless of their position whilst values of  $\sigma > 1$  led to no secondary peaks being detected.

The search for multi-peak signals is carried out as the data is read into the TTree structure. If the function returns  $N_{peaks} > 1$  then the histogram is added to a TTree containing only the bad signals that will be rejected from any further processing. The histograms are still drawn for reference to ensure that the removed signals are multi-peak signals. Fig. 5.1 is an example of a detection of a multi-peak signal using the method described above.

## 5.2 Waveform Pulse Fitting

After the processing of the waveforms, work began on the fitting of the waveform data. Although this work is yet to be completed, a brief outline of the methods being applied is outlined below. The form of the curve observed on the oscilloscope can be described approximately by Eq. 5.1 where;  $G$  is the gain of the PMT,  $N$  is the number of photoelectrons emitted by the cathode,  $e$  is the charge of the electron,  $\tau_s$  is the decay constant of the scintillator,  $R$  is the resistance of the PMT,  $\tau = \frac{R}{C}$  where  $C$  is the capacitance, and finally  $t$  is the time variable. Eq. 5.1 describes how the voltage signal produced from a photomultiplier tube (PMT) varies with time[43].

$$V(t) = \begin{cases} -\frac{GNeR}{\tau - \tau_s} \left[ \exp\left(-\frac{t}{\tau_s}\right) - \exp\left(-\frac{t}{\tau}\right) \right] & \text{for } \tau \neq \tau_s \\ \left(\frac{GNeR}{\tau_s^2}\right)t \exp\left(-\frac{t}{\tau_s}\right) & \text{for } \tau = \tau_s \end{cases} \quad (5.1)$$



Eq. 5.1 is a double exponential decay function with dependencies that are intrinsically related to the circuit of the PMT[43]. An attempt was made to fit the data with a Gaussian and a single exponential function, as shown in Appendix B.3. The Gaussian part of the function seemed to be an appropriate fit for the initial segment of the curve, before the peak of the data was reached. After the peak, the exponential part would then be used to fit the rest of the curve. However, an appropriate functional form for the exponential could not be determined meaning that the fitting of the waveform data remained incomplete. Further work will need to be conducted in this area to determine the correct functional form of the single or double exponential decay as described in Eq. 5.1.

### 5.3 Reproduction of an ADC Spectrum

The final part of the waveform analysis that was also attempted, although as of yet is incomplete, is the reproduction of an ADC spectrum via the usage of the waveform scope data. The aim of this is to produce an ADC spectrum without the "bad" pulses that may be included when the Caen ADC device is used to produce a spectrum and then compare the two to see how much of an effect the pile up of pulses may cause, if any. The reproduction of an ADC spectrum takes place by following the previous procedure of the removal of any pulses that include multiple peaks. After this, an internal ROOT function is used to calculate the integral of each pulse included in the data set containing single peak waveforms individually. The integration is numerical, and therefore does not require a fitted function. The integration took place between a user specified long gate of  $80 - 250ns$ . It should be noted that this long gate may not be applicable to all of the waveforms as the trigger time is not consistently the same.

The integration takes place over the specified window of charge to produce an ADC number (a value that would have units of  $mVs$ ) that is treated as being without units. This value is directly related to the energy of the proton that caused the pulse. The integral is calculated for all pulses and stored in a vector structure in ROOT. This vector is then sorted from smallest to largest and is looped over to determine how many times each integral appears. This value is the number of ADC events value and is stored in a separate vector.

### 5.4 Waveform Analysis Results

An example ADC spectrum produced using this method is displayed in Fig. 5.2. As can

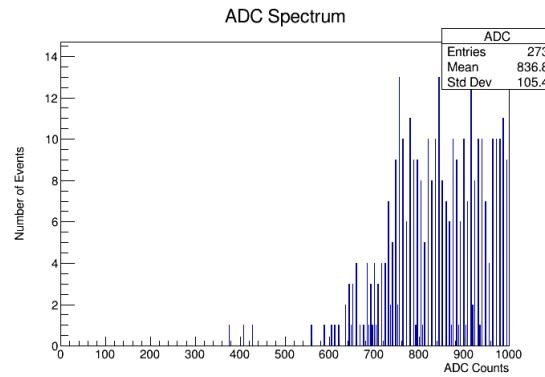


FIGURE 5.2: This figure depicts an example of an ADC spectrum that has been reproduced from scope waveform data. The data in question had 54 rejected pulses due to multiple peaks out of a total 9216 pulses.

be seen in Fig. 5.2, there are multiple issues. Foremost amongst these is the lack of a continuous spectrum in the ADC data. The data displayed only has 273 entries out of a total number of pulses of 9162 (excluding pulses containing multiple peaks). This may be a clear indicator that the integration that was carried out may not be of a sufficient precision to determine if the ADC number of two waveforms is distinct. The spectrum is also clearly skewed towards higher energy protons which is another indicator of the integration value most likely being incorrect. The expected distribution would be the Landau-Gaussian convolution due to the fact that the protons will have suffered energy loss whilst travelling through the medium before the detector, therefore leading to an attenuated signal. A potential explanation for the low entry number may be that, in comparison with other files, the scope data said had 10 times less waveforms. However, this does not explain the significant skew that can be seen, nor does it explain why only 3% of the pulses were recognised as being distinct.

# Conclusions

## 6.1 ADC Analysis Summary and Further Work

The first part of the project involved the improvement of the analysis of the ADC spectra. The result of this analysis has yielded an improved fit in all data sets that have been analysed thus far as demonstrated by the  $\chi^2$  value showing a significant decrease and tending towards 1. In turn, this shows that a fit following the form of a Landau-Gaussian convolution is appropriate. However, further work can be done with regards to the fitting to allow for even greater improvements. Firstly, as has been previously suggested, the significance parameter  $\kappa$  can be calculated for all absorbing materials that occupy the medium directly before the detector. This will then allow for the correct assertion of whether or not the Landau distribution is the most appropriate energy loss distribution for the data or if the Vavilov distribution may be more suitable. The ROOT macro could also be adapted to allow for automatic determination of  $\kappa$  which would then be used to select the relevant distribution.

Furthermore, it has been demonstrated that the number of steps used in the convolution integral is of high importance with regards to precise determination of the energy resolution. Improvements to this area of the code displayed in Appendix B.1 allowed the significant decrease in the  $\chi^2$  value. However, a limitation on the increase in the number of steps is the amount of time taken to process the spectrum data. Further work can be conducted into the convolution integral, determining if the order of the integral is correct, and also if the offset that has been defined is of the correct order. Once this has been complete, various numerical integration methods can be applied such as Simpson's rule, which allows improvements in the precision of an integral to be made by increasing the accuracy of the functions that approximate the integrand rather than the decreasing of the step size[45]. The Gaussian quadrature rule[45] may also be used as libraries already exist in languages such as Python's SciPy which may have equivalents within ROOT. The development of a more efficient integration algorithm will lead to even greater speed within the QA procedures.

Despite the noticeable improvements in the results, there are still some discrepancies. As shown for the data in Fig. 6.1, the fitted curve displays severe differences when compared with the data, leading to an enormous  $\chi^2$  value. The improved fitting shown in Fig. 6.2 demonstrates a significant improvement, albeit with large variations between the fit and the data still present. These variations seem to be present most abundantly in the data runs with the higher wheel settings ( $> 150$ ) however, the distribution remains

a Landau-Gaussian and should still be roughly approximated by the fit. The reason for this is currently unknown and requires further investigation. A potential explanation could be due to the constraints that are placed on some of the parameters that are used in the fit. The width of the curve on the left hand side and the normalisation constants for both the left and right hand side of the curve have constraints placed on them within the code. The fitting should be conducted again without the constraints or with the limits increased to determine if these restrictions are leading to the observed fluctuations on the curve. It should also be noted that the range of the fitting ( $\pm 4\sigma$ ) extends beyond the curve which may also explain the poor fitting as points may be included in the parameter variation that ROOT automatically carries out but these points may not appear in the curve. The macro used to fit the data could be adjusted to automatically determine the most appropriate range to fit over, ensuring that the fit does not extend beyond the length of the curve.

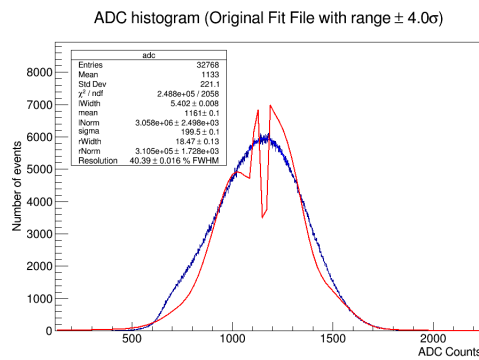


FIGURE 6.1: This figure depicts the ADC spectrum produced from the original fit file for run 019. The wheel position for the acquisition of this data was 165. Note that this position is directly before the Bragg peak position of 166. A clear discrepancy can be seen throughout most of the curve, as demonstrated by the extremely large  $\chi^2$  value.

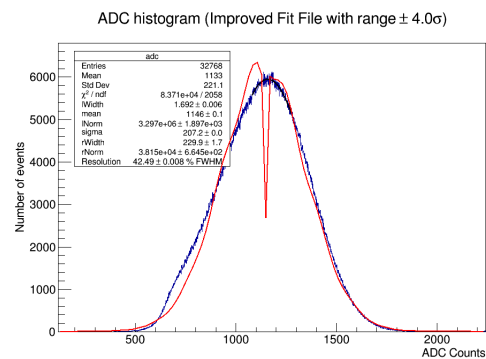


FIGURE 6.2: This figure depicts the ADC spectrum produced from the improved fit file for run 019. The wheel position for the acquisition of this data was 165. Note again that this position is directly before the Bragg peak position of 166. Improvements have been made to the discrepancies, however they have not been completely removed.

## 6.2 Waveform Analysis Summary and Further Work

The second part of the project was the processing and analysis of waveform files. The first result of this section yielded a Java program (shown in Appendix B.2) that is capable of swiftly converting the LeCroy scope binary files and also includes other functionality. The program is capable of converting a binary file containing 127,601 files in  $1hr$  which is an improvement over previous conversion attempts. The total file size has also been

reduced due to the removal of unnecessary information throughout processing. Further improvements can be made to the Java file to improve the efficiency of conversion. For example the program could be rewritten as a multithreaded jar which would reduce the amount of time taken to convert each file. Improvements are also going to be made as further information has been received detailing how to retrieve individual waveform segments from a continuous spectrum. This will allow for the definition of each individual waveform file to be more precise, rather than the current method being used where a new file is produced for every 1000 lines of text in the binary file.

The second aim of the waveform analysis that was also completed was the development of a program that automatically identifies any multi-peak pulses. Preliminary results on the analysed file shown in Fig. 5.2 demonstrate a low ratio of bad to good pulses with only 54 pulses being rejected. However, further work will need to be conducted on the section of the program (shown in Appendix B.3) that reproduces an ADC spectrum. The preliminary result only contains 273 entries from a total number of 9216 pulses. This is an indication that there may be an issue with the integration method used not being precise enough i.e. the rounding of the integrated value may cut-off decimal places that are required to distinguish pulses. The code also needs to be rewritten to subtract the baseline of the pulse to prevent the tail of the curve from affecting the integration. The determination of the baseline value could potentially be implemented via the usage of another method in the TSpectrum class that can be used to determine background noise. Once this is determined, the pulse will be subtracted from the background and then the integration can be carried out. One final improvement that can be made is the reproduction of ADC spectra using the various long gates that are used by the Caen converter (50, 100, and 150ns). The subtraction of the baseline may also correct the skew that can be seen in Fig. 5.2 as the skew may be due to the integration including values from the straggling tail. Once this has been done, the ADC fitting code (Appendix B.1) can be applied to the reproduced fits to determine the energy resolution for comparison to the equivalent ADC spectrum.

Overall the project has aided in the improvement of the QA methods that will be used in future PBT treatments via the development of multiple programs that can be used to further data analysis. Although further work needs to be conducted to ensure that the programs have all required functions, they have significantly advanced the processing of both types of data that are currently being taken and are particularly useful due to either precision or efficiency improvements.

---

# Bibliography

- [1] Cancer Statistics for the UK, 2017. <http://www.cancerresearchuk.org/health-professional/cancer-statistics>.
- [2] Paul E Oberstein and Kenneth P Olive. Pancreatic cancer: why is it so hard to treat? *Therapeutic advances in gastroenterology*, 6(4):321–37, jul 2013.
- [3] D R Beil and L M Wein. Analysis and comparison of multimodal cancer treatments. *IMA journal of mathematics applied in medicine and biology*, 18(4):343–76, dec 2001.
- [4] JC Coffey, JH Wang, MJF Smith, et al. Excisional surgery for cancer cure: therapy at a cost, 2003.
- [5] Herbert M Pinedo and Giuseppe Giaccone. Chemotherapy. *The Lancet*, 349:S7–S9, 1997.
- [6] Lisa M DeAngelis and Jerome B Posner. Side Effects of Chemotherapy. In *Neurologic Complications of Cancer*, chapter 12, pages 447–510. Oxford University Press, oct 2011.
- [7] R. Gahbauer, T. Landberg, J. Chavaudra, et al. 1 INTRODUCTION. *Journal of the ICRU*, 4(1):21–24, jun 2004.
- [8] Vedang Murthy and Alan Horwich. Intensity Modulated Radiation Therapy. *European Journal of Cancer*, 40(16):2349–2351, nov 2004.
- [9] Karl Otto. Volumetric modulated arc therapy: IMRT in a single gantry arc. *Medical Physics*, 35(1):310, 2008.
- [10] David A. Jaffray. Image-guided radiotherapy: from current concept to future perspectives. *Nature Reviews Clinical Oncology*, 9(12):688–699, nov 2012.
- [11] T Gupta and C Anand Narayan. Image-guided radiation therapy: Physician’s perspectives. *Journal of medical physics*, 37(4):174–82, oct 2012.
- [12] H. Rodney Withers. Biologic basis for altered fractionation schemes. *Cancer*, 55(S9):2086–2095, may 1985.
- [13] Joo-Young Na, Min-Cheol Lee, Shin Jung, et al. A mass lesion in the skull after radiotherapy for medulloblastoma. *Neuropathology*, 32(5):583–585, oct 2012.

- [14] Hoppe Bradford, Henderson Randal, Mendenhall William M., et al. Proton Therapy for Prostate Cancer. *Oncology*, 25(7):644–650,652, 2011.
- [15] W P Levin, H Kooy, J S Loeffler, et al. Proton beam therapy. *British Journal of Cancer*, 93(8):849–854, oct 2005.
- [16] Abraham Al-Mamgani, Wilma D. Heemsbergen, Stephanie T.H. Peeters, et al. Role of Intensity-Modulated Radiotherapy in Reducing Toxicity in Dose Escalation for Localized Prostate Cancer. *International Journal of Radiation Oncology\*Biophysics*, 73(3):685–691, 2009.
- [17] Uli Weber and Gerhard Kraft. Comparison of Carbon Ions Versus Protons. *The Cancer Journal*, 15(4):325–332, jul 2009.
- [18] Bertil Damato, Andrzej Kacperek, Mona Chopra, et al. Proton beam radiotherapy of choroidal melanoma: The Liverpool-Clatterbridge experience. *International Journal of Radiation Oncology\*Biophysics*, 62(5):1405–1411, 2005.
- [19] Edward C. Creutz and Robert R. Wilson. MonoEnergetic Protons from a Cyclotron. *Review of Scientific Instruments*, 17(10):385–388, oct 1946.
- [20] J. M. Leggate. Proton-beam Therapy. *BMJ*, 1(5118):363–364, feb 1959.
- [21] Yi Rong and James Welsh. Basics of Particle Therapy II Biologic and Dosimetric Aspects of Clinical Hadron Therapy. *American Journal of Clinical Oncology*, 33(6):646–649, dec 2010.
- [22] Vsevolod V. Balashov. *Interaction of particles and radiation with matter*. Springer, 2012.
- [23] Cary Zeitlin and Chiara La Tessa. The Role of Nuclear Fragmentation in Particle Therapy and Space Radiation Protection. *Frontiers in oncology*, 6:65, 2016.
- [24] Harald Paganetti and Thomas Bortfeld. Proton Beam Radiotherapy -The State of the Art 1.
- [25] Ervin D. Podgorsak and International Atomic Energy Agency. *Radiation oncology physics : a handbook for teachers and students*. International Atomic Energy Agency, 2005.
- [26] Harold Elford. Johns and John Robert. Cunningham. *The physics of radiology*. Charles C. Thomas, 1983.
- [27] J.W. Wilson, J.L. Shinn, L.W. Townsend, et al. NUCFRG2: A semiempirical nuclear fragmentation model. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 94(1):95–102, 1994.

- [28] Armin Lühr, David C Hansen, Ricky Teiwes, et al. The impact of modeling nuclear fragmentation on delivered dose and radiobiology in ion therapy. *Physics in Medicine and Biology*, 57(16):5169–5185, aug 2012.
- [29] David Jette and Weimin Chen. Creating a spread-out Bragg peak in proton beams. *Physics in Medicine and Biology*, 56(11):N131–N138, jun 2011.
- [30] Robley Dungalison Evans. *The atomic nucleus*. R.E. Krieger, 1982.
- [31] Thomas Bortfeld. An analytical approximation of the Bragg curve for therapeutic proton beams. *Medical Physics*, 24(12):2024–2033, dec 1997.
- [32] Lindsay G. Jensen, Loren K. Mell, Christin A. Knowlton, et al. Spread-Out Bragg Peak (SOBP). In *Encyclopedia of Radiation Oncology*, pages 809–810. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [33] M. Prall, M. Durante, T. Berger, et al. High-energy proton imaging for biomedical applications. *Scientific Reports*, 6(November 2015):27651, jun 2016.
- [34] V. A. Bashkirov, R. W. Schulte, R. F. Hurley, et al. Novel scintillation detector design and performance for proton radiography and computed tomography. *Medical Physics*, 43(2):664–74, 2016.
- [35] V. Sipala, M. Bruzzi, M. Bucciolini, et al. A proton imaging device: Design and status of realization. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 612(3):566–570, 2010.
- [36] Anastasia Freshville. Calorimeter R&D for the SuperNEMO double beta decay experiment. *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 623(1):255–257, 2010.
- [37] A. Basharina-Freshville. *Search for the neutrinoless double beta decay of  $^{100}\text{Mo}$  with the NEMO3 detector and calorimeter research and development for the SuperNEMO experiment*. PhD thesis, 2012.
- [38] Ruben Saakyan, the NEMO3 Collaborations, SuperNEMO, et al. Topological detection of double beta decay with NEMO3 and SuperNEMO. *Journal of Physics: Conference Series*, 179(1):012006, jul 2009.
- [39] Matthew B Kauer. *Search for the double beta decay of  $\text{Zr-96}$  with NEMO-3 and calorimeter development for the SuperNEMO experiment*. PhD thesis, 2010.



- 
- [40] D. Besset. PROPERTIES OF THE PRIMARY IONIZATION OR PHOTOELECTRON DISTRIBUTION FOR AN ENERGY LOSS DETECTOR. *Nucl.Instrum.Methods*, 1985.
- [41] L. Landau. On the energy loss of fast particles by ionization. *J.Phys.(USSR)*, 8:201–205, 1944.
- [42] P.V. Vavilov. Ionization losses of high-energy heavy particles. *Zh.Eksp.Teor.Fiz.*, 32:749–751, 1957.
- [43] William R. Leo. *Techniques for nuclear and particle physics experiments : a how-to approach*. Springer, 1994.
- [44] Miroslav Morháč, Ján Kliman, Vladislav Matoušek, et al. Identification of peaks in multidimensional coincidence  $\gamma$ -ray spectra. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 443(1):108–125, mar 2000.
- [45] K. F. (Kenneth Franklin) Riley, M. P. (Michael Paul) Hobson, and S. J. (Stephen John) Bence. *Mathematical methods for physics and engineering*. Cambridge University Press, 2006.

# Results

## A.1 ADC Spectrum Results

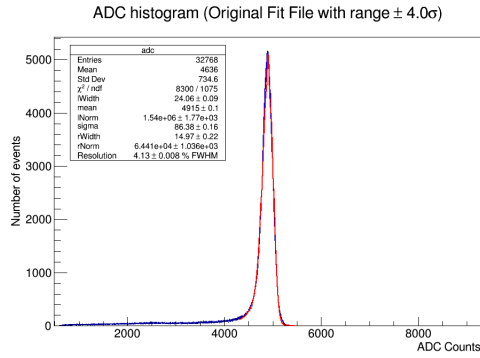


FIGURE A.1: This figure depicts the ADC spectrum produced from the original fit file for run 039 with a rate of  $7kH_z$ . This set of data had a  $13.4mm$  PMMA block in position upstream of the beam. The beam was shifted during this set of data.

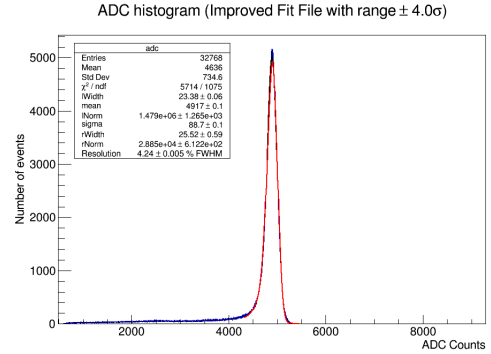


FIGURE A.2: This figure depicts the ADC spectrum produced from the improved fit file for run 039. The energy resolution has slightly increased but with a decreased error. There is a discrepancy between the peak of the fit and the data.

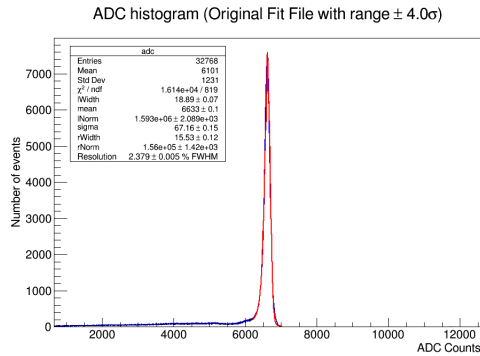


FIGURE A.3: This figure depicts the ADC spectrum produced from the original fit file for run 029. The rate of the beam for this run was  $20kH_z$ . The wheel was set to 0 and there was no PMMA present.

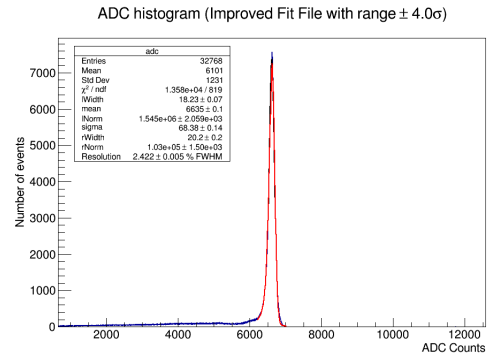


FIGURE A.4: This figure depicts the ADC spectrum produced from the improved fit file for run 029. The peak of this fit seems to match more closely than the original fit. There was no improvement in the resolution error.

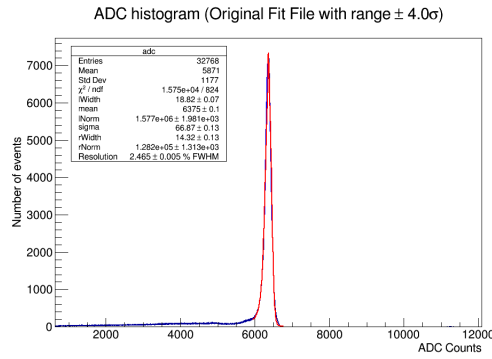


FIGURE A.5: This figure depicts the ADC spectrum produced from the original fit file for run 001. The rate of the beam for this run was  $25\text{kHz}$ . The wheel was set to 0 and there was no PMMA present.

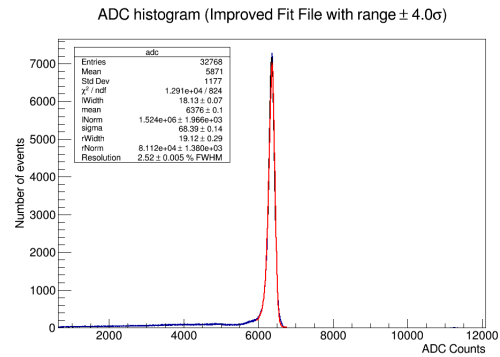


FIGURE A.6: This figure depicts the ADC spectrum produced from the improved fit file for run 001.

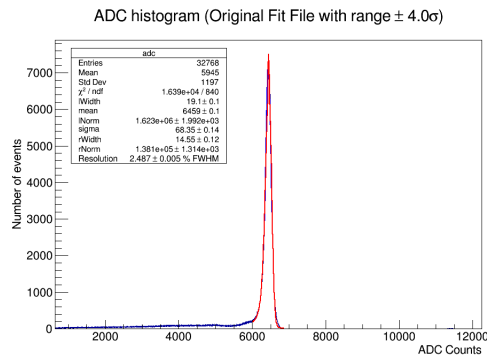


FIGURE A.7: This figure depicts the ADC spectrum produced from the original fit file for run 027. The rate of the beam for this run was  $25\text{kHz}$ . The wheel was set to 0 and there was no PMMA present.

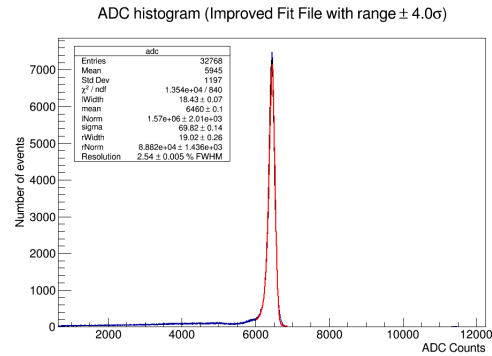


FIGURE A.8: This figure depicts the ADC spectrum produced from the improved fit file for run 027.

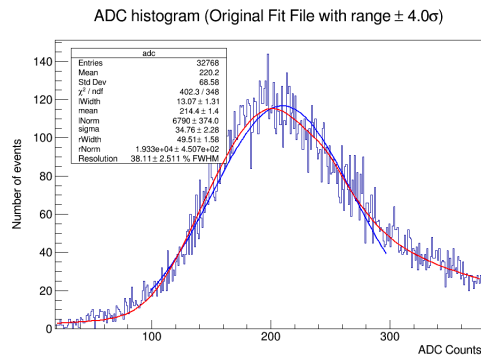


FIGURE A.9: This figure depicts the ADC spectrum produced from the original fit file for run 028. This run was to determine the background radiation with the beam off and the wheel position at 0.

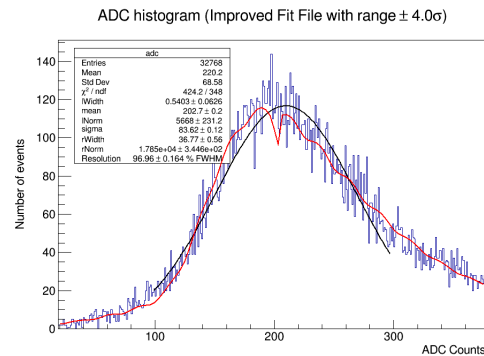


FIGURE A.10: This figure depicts the ADC spectrum produced from the improved fit file for run 028. Any differences seen here are not of particular relevance as the particles may not be protons.

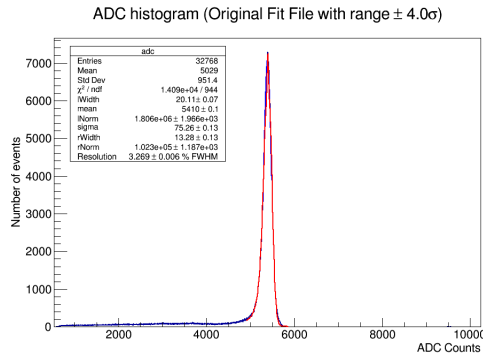


FIGURE A.11: This figure depicts the ADC spectrum produced from the original fit file for run 032. The rate was  $25\text{kHz}$ . A  $7.24\text{mm}$  PMMA block was used here corresponding to a wheel position of 40.

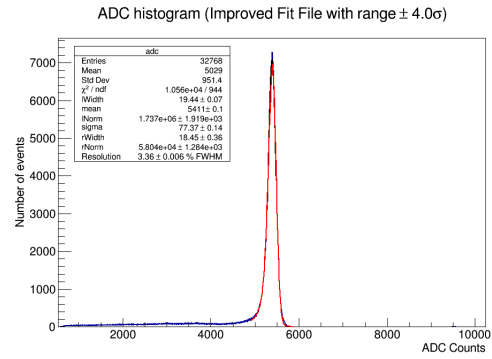


FIGURE A.12: This figure depicts the ADC spectrum produced from the improved fit file for run 032. The discrepancy between the peak and the fit seems to have improved.

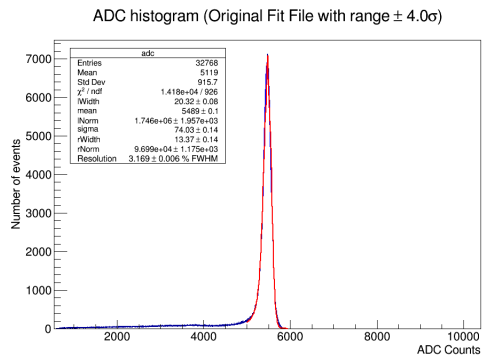


FIGURE A.13: This figure depicts the ADC spectrum produced from the original fit file for run 035. The rate was  $12\text{kHz}$ . A  $7.24\text{mm}$  PMMA block was used here upstream of the beam corresponding to a wheel position of 40.

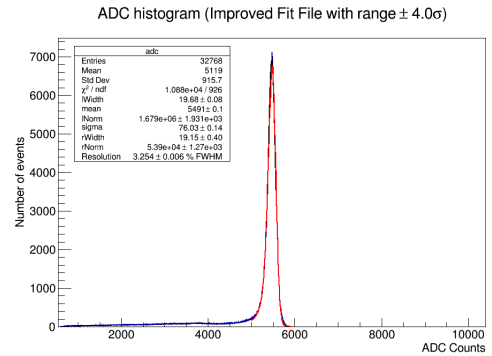


FIGURE A.14: This figure depicts the ADC spectrum produced from the improved fit file for run 035.

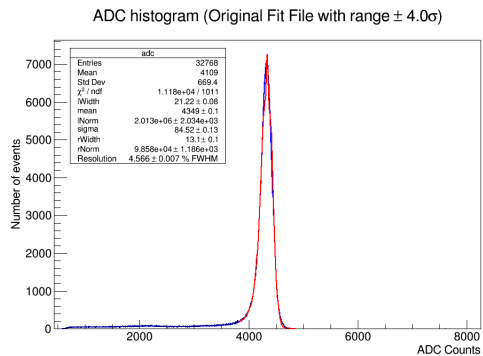


FIGURE A.15: This figure depicts the ADC spectrum produced from the original fit file for run 037. The rate was  $25\text{kHz}$ . A  $13.4\text{mm}$  PMMA block was used here. Beam issues led to the actual rate being  $20\text{kHz}$ .

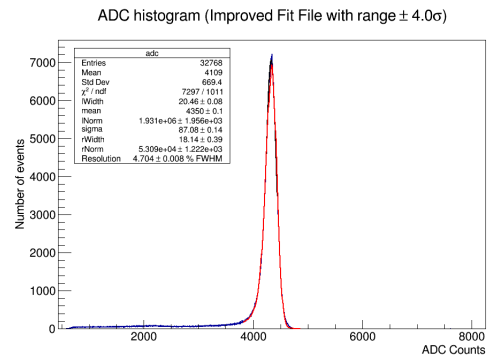


FIGURE A.16: This figure depicts the ADC spectrum produced from the improved fit file for run 037. The rate was  $25\text{kHz}$ . Beam issues led to the actual rate being  $20\text{kHz}$ .

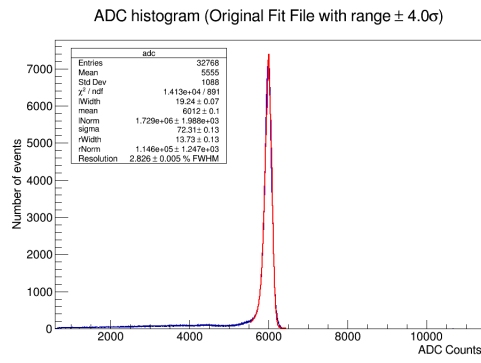


FIGURE A.17: This figure depicts the ADC spectrum produced from the original fit file for run 005. The rate was  $25\text{kHz}$ . A wheel position of 20 was used.

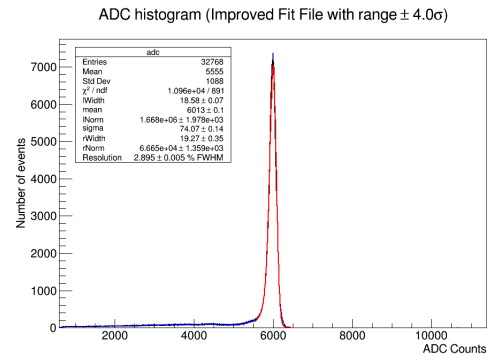


FIGURE A.18: This figure depicts the ADC spectrum produced from the improved fit file for run 005.

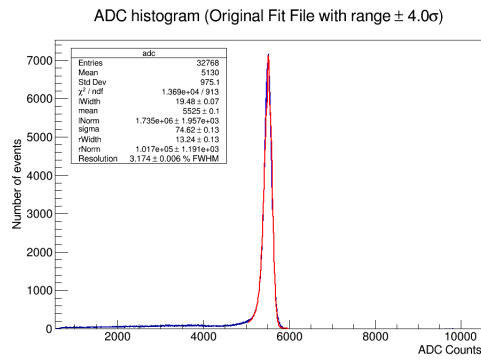


FIGURE A.19: This figure depicts the ADC spectrum produced from the original fit file for run 006. The rate was  $25\text{kHz}$ . A wheel position of 40 was used for this run.

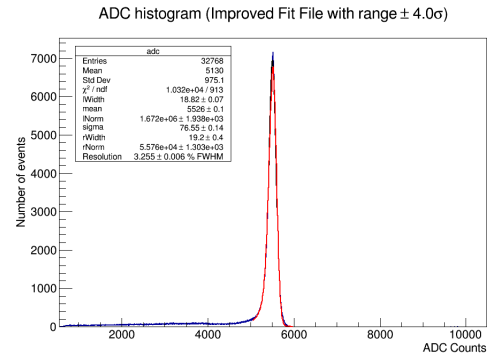


FIGURE A.20: This figure depicts the ADC spectrum produced from the improved fit file for run 006.

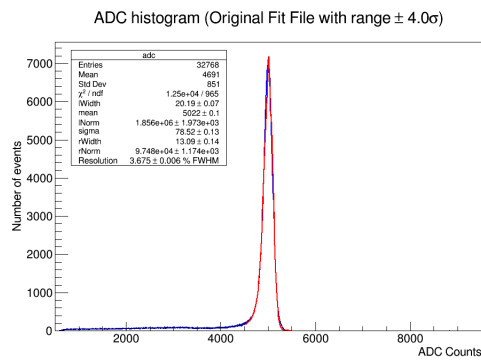


FIGURE A.21: This figure depicts the ADC spectrum produced from the original fit file for run 007. The rate was  $25\text{kHz}$ . A wheel position of 60 was used for this run.

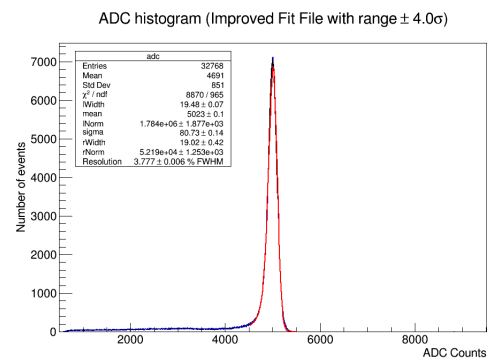


FIGURE A.22: This figure depicts the ADC spectrum produced from the improved fit file for run 007.

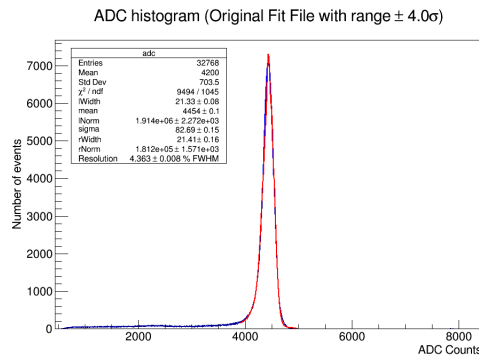


FIGURE A.23: This figure depicts the ADC spectrum produced from the original fit file for run 008. The rate was  $25\text{kHz}$ . A wheel position of 80 was used for this run.

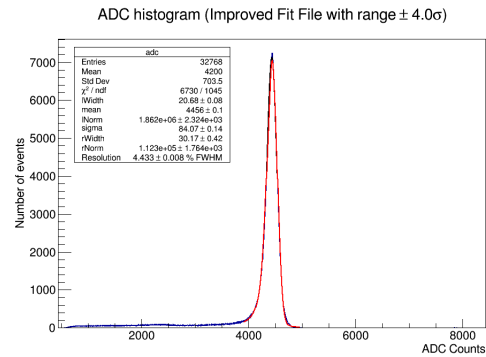


FIGURE A.24: This figure depicts the ADC spectrum produced from the improved fit file for run 008.

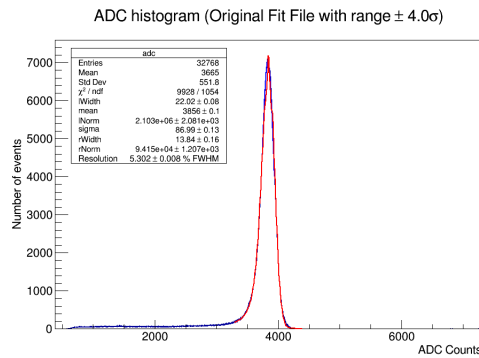


FIGURE A.25: This figure depicts the ADC spectrum produced from the original fit file for run 009. The rate was  $25\text{kHz}$ . A wheel position of 100 was used for this run.

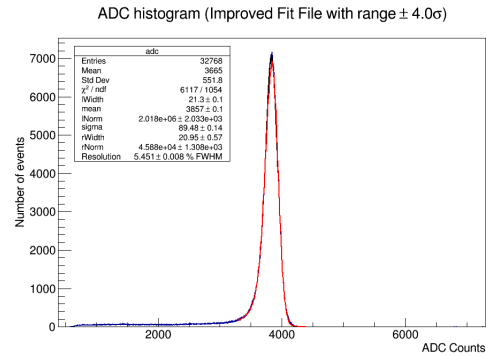


FIGURE A.26: This figure depicts the ADC spectrum produced from the improved fit file for run 009.

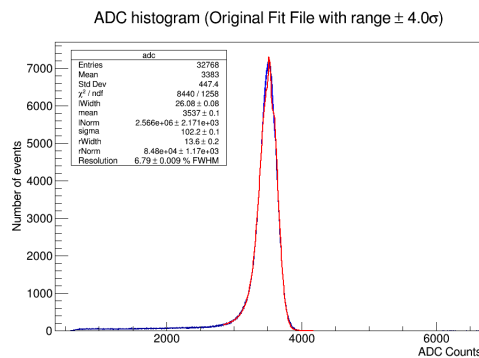


FIGURE A.27: This figure depicts the ADC spectrum produced from the original fit file for run 014. The rate was  $25\text{kHz}$ . A wheel position of 120 was used for this run.

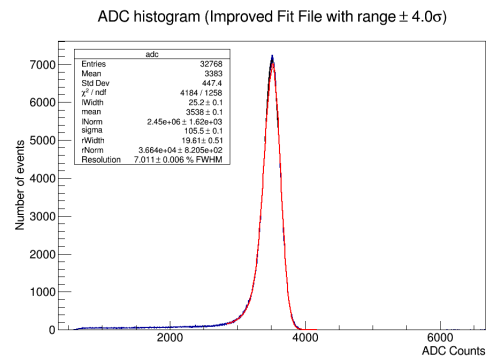


FIGURE A.28: This figure depicts the ADC spectrum produced from the improved fit file for run 014.

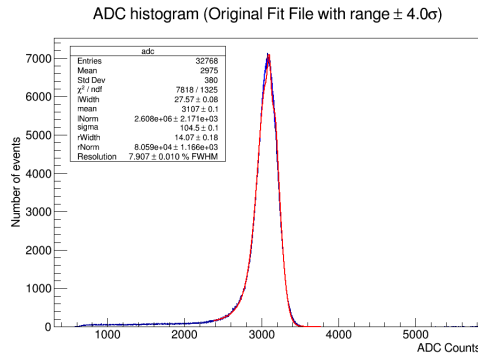


FIGURE A.29: This figure depicts the ADC spectrum produced from the original fit file for run 017. The rate was  $25\text{kHz}$ . A wheel position of 130 was used for this run.

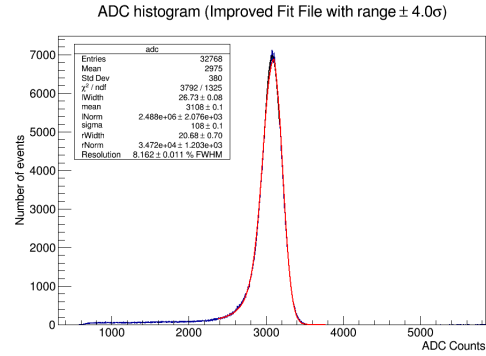


FIGURE A.30: This figure depicts the ADC spectrum produced from the improved fit file for run 017.

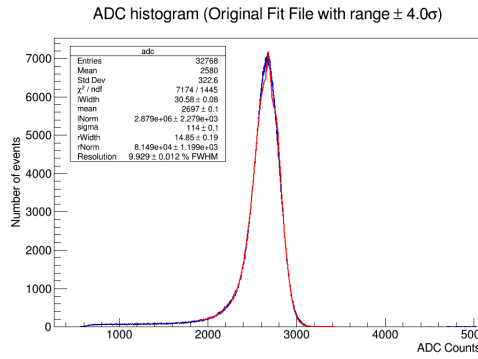


FIGURE A.31: This figure depicts the ADC spectrum produced from the original fit file for run 015. The rate was  $25\text{kHz}$ . A wheel position of 140 was used for this run.

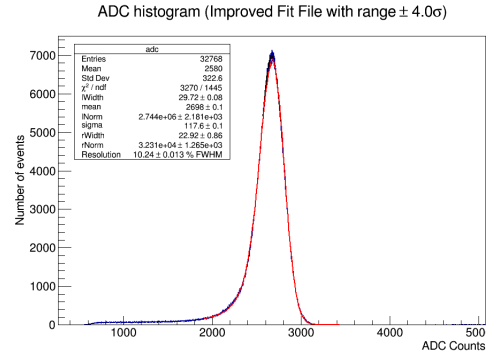


FIGURE A.32: This figure depicts the ADC spectrum produced from the improved fit file for run 015.

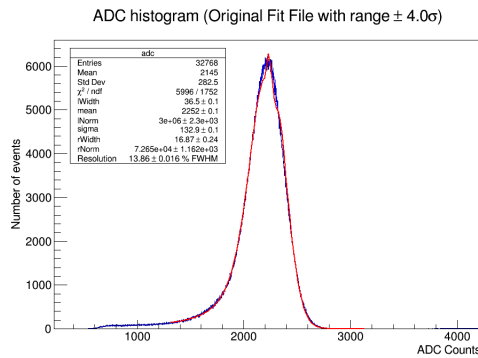


FIGURE A.33: This figure depicts the ADC spectrum produced from the original fit file for run 018. The rate was  $25\text{kHz}$ . A wheel position of 150 was used for this run.

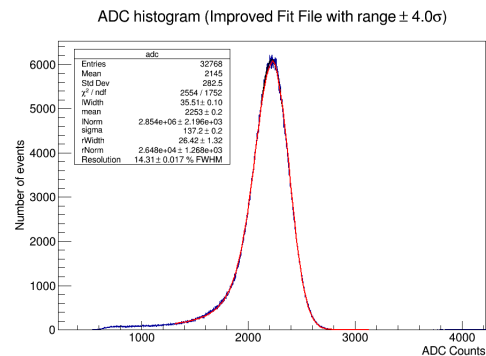


FIGURE A.34: This figure depicts the ADC spectrum produced from the improved fit file for run 018.

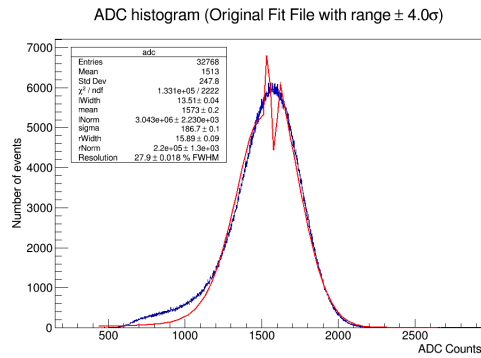


FIGURE A.35: This figure depicts the ADC spectrum produced from the original fit file for run 016. The rate was  $25\text{kHz}$ . A wheel position of 160 was used for this run.

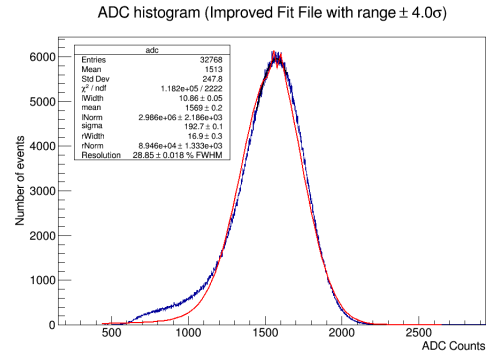


FIGURE A.36: This figure depicts the ADC spectrum produced from the improved fit file for run 016.

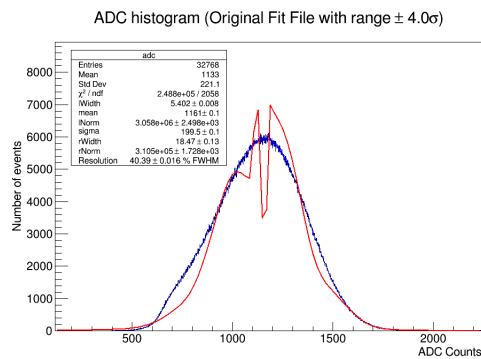


FIGURE A.37: This figure depicts the ADC spectrum produced from the original fit file for run 019. The wheel position for the acquisition of this data was 165. Note that this position is directly before the Bragg peak position of 166. A clear discrepancy can be seen throughout most of the curve, as demonstrated by the extremely large  $\chi^2$  value.

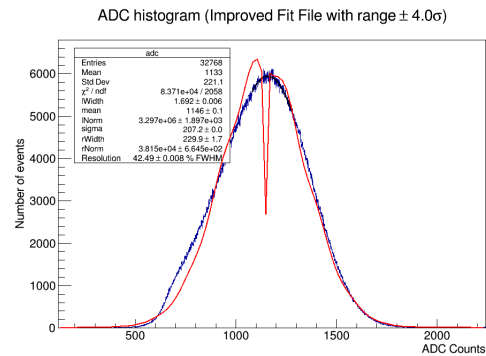


FIGURE A.38: This figure depicts the ADC spectrum produced from the improved fit file for run 019. Improvements have been made to the discrepancies, however they have not been completely removed.



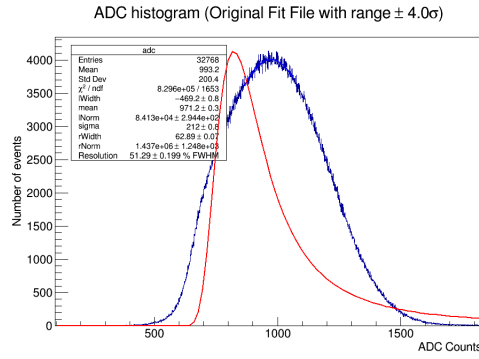


FIGURE A.39: This figure depicts the ADC spectrum produced from the original fit file for run 026. The rate was  $25\text{kHz}$ . A wheel position of 167 was used for this run. This position was directly before the Bragg peak.

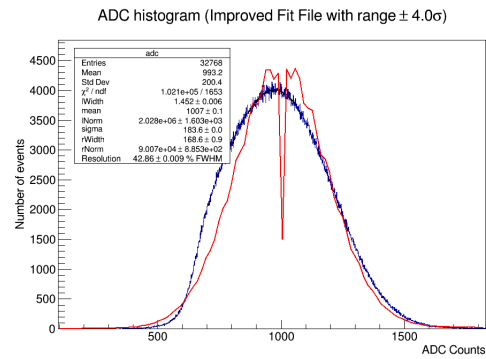


FIGURE A.40: This figure depicts the ADC spectrum produced from the improved fit file for run 026.

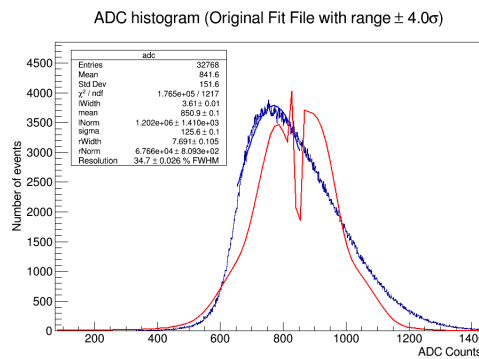


FIGURE A.41: This figure depicts the ADC spectrum produced from the original fit file for run 020. The rate was  $25\text{kHz}$ . A wheel position of 170 was used for this run. The beam dropped during this run.

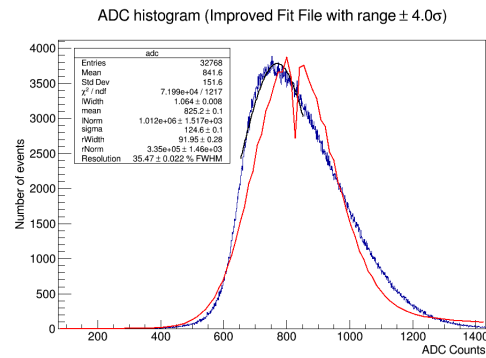


FIGURE A.42: This figure depicts the ADC spectrum produced from the improved fit file for run 020.

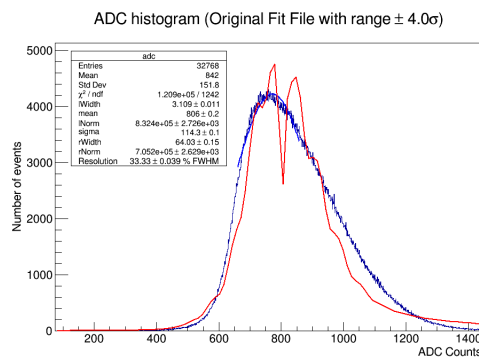


FIGURE A.43: This figure depicts the ADC spectrum produced from the original fit file for run 021. The rate was  $25\text{kHz}$ . A wheel position of 170 was used for this run. This measurement was taken as a repeat of run 020 due to the beam dropping.

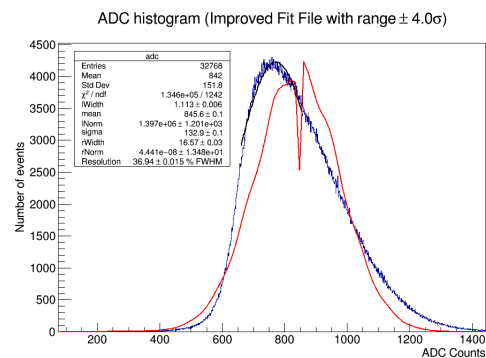


FIGURE A.44: This figure depicts the ADC spectrum produced from the improved fit file for run 021.

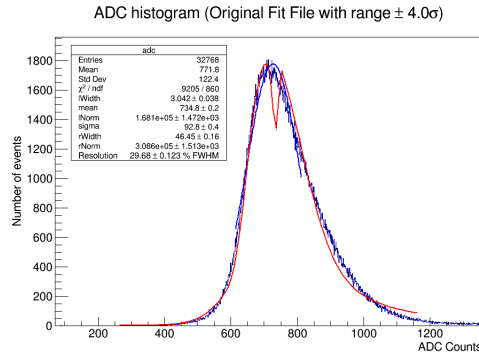


FIGURE A.45: This figure depicts the ADC spectrum produced from the original fit file for run 023. The rate was  $25\text{kHz}$ . A wheel position of 173 was used for this run. This measurement was taken as a repeat of run 022 due to the beam dropping.

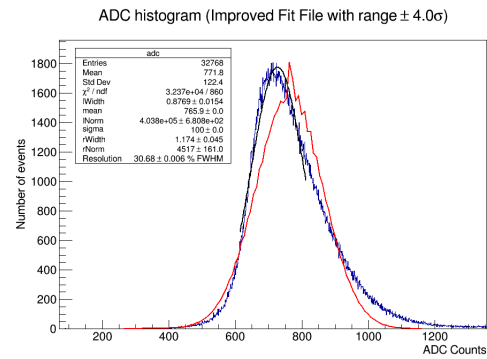


FIGURE A.46: This figure depicts the ADC spectrum produced from the improved fit file for run 023. The rate was  $25\text{kHz}$ .

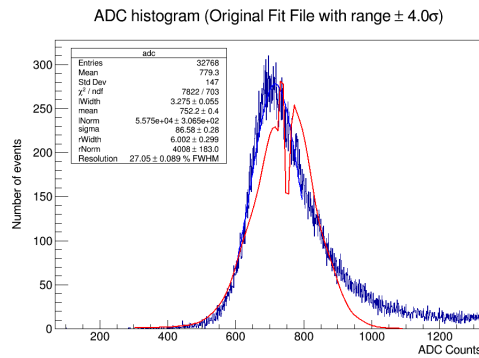


FIGURE A.47: This figure depicts the ADC spectrum produced from the original fit file for run 024. The rate was  $25\text{kHz}$ . A wheel position of 175 was used for this run.

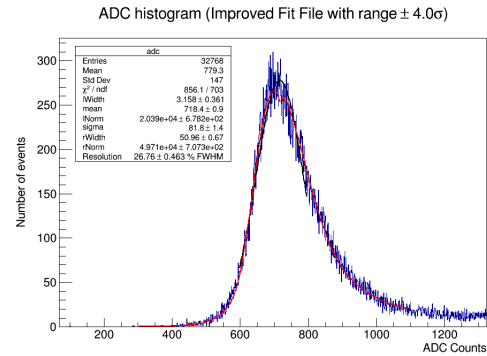


FIGURE A.48: This figure depicts the ADC spectrum produced from the improved fit file for run 024.

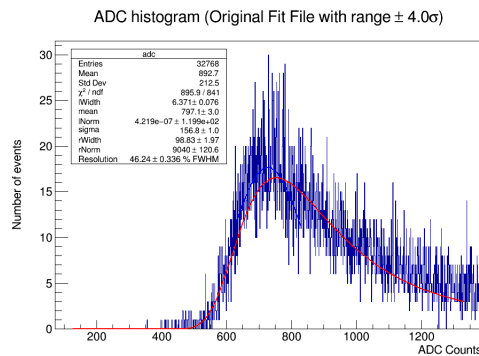


FIGURE A.49: This figure depicts the ADC spectrum produced from the original fit file for run 025. The rate was  $25\text{kHz}$ . A wheel position of 179 was used for this run. In this data set there is no peak. The signal is due to secondary particles.

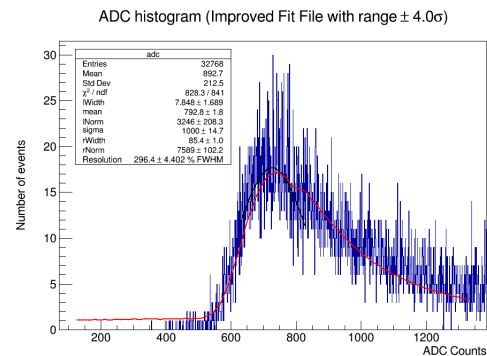


FIGURE A.50: This figure depicts the ADC spectrum produced from the improved fit file for run 025.

# Written Code

## B.1 ADC Spectrum Fit Macro

The code described below is the improved fitting function for the ADC spectrum that is run via ROOT. The main output of the program is a ROOT file containing the ADC histogram which in turn contains the data, the initial Gaussian fit, and the final convolution fit.

---

```
1 // Fit for 60MeV proton beam data
2 #include <iostream>
3 #include <string>
4 #include <fstream>
5 #include <sstream>
6 #include <iomanip>
7 #include "TCanvas.h"
8 #include "TF1.h"
9 #include "TFile.h"
10 #include "TGraph.h"
11 #include "TH1D.h"
12 #include "TLine.h"
13 #include "TLegend.h"
14 #include "TMath.h"
15 #include "TTree.h"
16 #include "TStyle.h"
17 #include "TPaveStats.h"
18 #include "TText.h"
19 #include "TLatex.h"
20 #include "TString.h"
21 #include "TImage.h"
22
23 using std::cout;
24 using std::endl;
25 using std::string;
26 using std::stringstream;
27 using std::ifstream;
28
```

```
29 static TCanvas* cResultsDisplay = 0;
30
31 //Variable initialisation
32 TH1D*      GetData(const string readInFile);
33 Double_t   GetPeak(TH1D* adc);
34 TF1*       FirstFit(TH1D* adc, Double_t peak, Double_t lowerRange,
    ↪ Double_t upperRange);
35 Double_t   GaussianFit(Double_t *x, Double_t *par);
36 void       FinalFit(TH1D* adc, TF1* firstFit, Double_t lowerRange,
    ↪ Double_t upperRange);
37 Double_t   LanGausFunc(Double_t *x, Double_t *par);
38 void       Print(TCanvas* canvas, const string readInFile);
39
40 //Main fit code
41 void fit(const string filename, Double_t lower, Double_t upper) {
42
43     gStyle->SetOptFit();
44     gStyle->SetPaintTextFormat("4.3f");
45     cResultsDisplay = new TCanvas("cResultsDisplay","Fit
    ↪ results",0,0,1000,700);
46
47     // Read in root file to fit
48     TH1D* adc = GetData(filename);
49
50     // Find peak to get initial fit parameters
51     Double_t peak = GetPeak(adc);
52
53     // Run first fit (Gaussian only) to extract fit parameters
54     Double_t range = 100.; // peak +/- range for initial fit, //100
55     Double_t lowerRange = peak - range;
56     Double_t upperRange = peak + range;
57
58     cout << endl;
59     cout << "!!! First fit with arbitrary parameters and range !!!"
    ↪ << endl;
60
61     TF1* firstFitResult = FirstFit(adc, peak, lowerRange,
    ↪ upperRange);
62     Double_t sigma = firstFitResult->GetParameter(2);
```

```
63     //FirstFit(adc, peak, lowerRange, upperRange);
64
65     // Define fit range for the second fit (+/- 2sigma)
66     Double_t finalLower = peak - (lower*sigma);
67     Double_t finalUpper = peak + (upper*sigma); //1.4
68
69     cout << endl;
70     cout << "!!! Final fit with defined range !!!" << endl;
71
72     FinalFit(adc, firstFitResult, finalLower, finalUpper);
73
74     Print(cResultsDisplay,filename);
75 }
76
77 // Function to read in data and create file
78
79 TH1D* GetData(const string readInFile) {
80
81     //Specify file to read in
82
83     string filename = readInFile;
84     string rootname;
85     rootname = (filename+".root");
86
87     TFile *file = new TFile(rootname.c_str(),"RECREATE");
88
89     cout << endl << "Taking data from and creating root file with
90     ↪ run: " << readInFile << endl;
91
92     //Int_t x = 20000; // number of bins and maximum bin (binnig
93     ↪ must be 1 to 1)
94     Int_t x = 40000; //for EJ Block
95     Int_t nBins = x;
96     Int_t maxBin = x;
97
98     TH1D *adc = new TH1D("adc","ADC histogram (Improved Fit File
99     ↪ with range #pm 4.0#sigma)",nBins,0,maxBin);
100
101     // Read in file
```

```
99
100     string inname;
101     inname = (filename+".dat");
102
103     ifstream resultsFile;
104     resultsFile.open(inname.c_str());
105
106     if (resultsFile.is_open()) {
107
108         string data;
109
110         while (getline(resultsFile, data)) {
111
112             stringstream dataReadin;
113             dataReadin << data;
114
115             Int_t bin = 0;
116             Int_t binValue = 0;
117
118             dataReadin >> bin >> binValue;
119             adc->SetBinContent(bin,binValue);
120         }
121         resultsFile.close();
122     }
123
124     adc->SetDirectory(0);
125     adc->GetXaxis()->SetTitle("ADC Counts");
126     adc->GetYaxis()->SetTitle("Number of events");
127     adc->GetYaxis()->SetTitleOffset(1.3);
128     adc->GetYaxis()->CenterTitle();
129     adc->GetXaxis()->SetNdivisions(506);
130
131     Double_t content = 0.;
132     Double_t upperRange = 0.;
133     Double_t maxContent = -1.;
134
135     for (Int_t bin = 0; bin <= adc->GetNbinsX(); bin++) {
136
137         content = adc->GetBinContent(bin);
```

```
138         if (content > maxContent) {
139             maxContent = content;
140             upperRange = bin;
141         }
142     }
143
144     cout << "Peak at " << upperRange << " ADC counts" << endl;
145
146     adc->Write();
147     adc->GetXaxis()->SetRangeUser(upperRange*0.1, upperRange*1.9);
148     adc->Draw();
149
150     // Manipulate stats box
151     cResultsDisplay->Update();
152     TPaveStats* st =
153         ↪ (TPaveStats*)(adc->GetListOfFunctions()->FindObject("stats"));
154     st->SetX1NDC(0.129518);
155     st->SetX2NDC(0.421687);
156     st->SetY1NDC(0.558383);
157     st->SetY2NDC(0.877246);
158
159     adc->GetListOfFunctions()->ls();
160     //adc->Write();
161     file->Close();
162
163     return adc;
164 }
165
166 Double_t GetPeak(TH1D* adc) {
167     //Find peak to get parameters for initial fit
168
169     Double_t peak = adc->GetMaximumBin();
170     cout << "Peak at " << peak << " ADC counts" << endl;
171
172     return peak;
173
174     /*
175     Double_t content = 0.;
```

```
176 Double_t peak = 0.;
177 Double_t maxContent = -1.;
178
179 for (Int_t bin = 0; bin <= adc->GetNbinsX(); bin++) {
180
181     content = adc->GetBinContent(bin);
182     if (content > maxContent) {
183         maxContent = content;
184         peak = bin;
185     }
186 }
187
188 cout << "Peak at " << peak << " ADC counts" << endl;
189 return peak;
190     */
191 }
192
193 TF1* FirstFit(TH1D* adc, Double_t peak, Double_t lowerRange, Double_t
    ↪ upperRange) {
194
195     TF1* initialFit = new TF1("initialFit",
196                               GaussianFit,
197                               lowerRange,
198                               upperRange,
199                               3);
200
201     Double_t height = adc->GetBinContent((Int_t)peak);
202
203     //Set initial parameters so root doesn't whine
204
205     initialFit->SetParameter(0, height);
206     initialFit->SetParameter(1, peak);
207     initialFit->SetParameter(2, sqrt(peak));
208     initialFit->SetParLimits(2, 0, 10000);
209
210     adc->Fit("initialFit","RN");
211     Double_t mean = initialFit->GetParameter(1);
212     Double_t sigma = initialFit->GetParameter(2);
213
```



```
214     initialFit->SetLineColor(kBlack);
215     initialFit->Draw("lsame");
216
217     cout << "Fit range: " << lowerRange << " - " << upperRange <<
    ↪ endl;
218     cout << "Fit parameters: mean of " << mean << " and sigma of "
    ↪ << sigma << endl;
219
220     return initialFit;
221
222 }
223
224 Double_t GaussianFit(Double_t *x, Double_t *par){
225
226     Double_t norm = par[0];
227     Double_t mean = par[1];
228     Double_t sigma = par[2];
229     Double_t xx = x[0];
230
231     Double_t arg = (xx - mean)/sigma;
232     Double_t equation = norm * TMath::Exp(-(arg*arg));
233
234     return equation;
235 }
236
237 void FinalFit(TH1D* adc, TF1* firstFit, Double_t lowerRange, Double_t
    ↪ upperRange) {
238
239     TF1* finalFit = new TF1("finalFit",
240                             LanGausFunc,
241                             lowerRange,
242                             upperRange,
243                             7);
244
245     finalFit->SetParNames("lWidth", "mean", "lNorm", "sigma", "rWidth",
    ↪ "rNorm", "Resolution");
246
247     finalFit->SetParameter(0, 1);
248     finalFit->SetParameter(1, firstFit->GetParameter(1));
```

```
249     finalFit->SetParameter(2, firstFit->GetParameter(0));
250     finalFit->SetParameter(3, firstFit->GetParameter(2));
251     finalFit->SetParLimits(2, 0, 100000000);
252     finalFit->SetParLimits(3, 0, 1000);
253     finalFit->SetParameter(4, 1);
254     finalFit->SetParameter(5, 100);
255     finalFit->SetParameter(6,0);
256     finalFit->SetParLimits(4, 0, 1000);
257     finalFit->SetParLimits(5, 0, 100000000);
258
259     adc->Fit("finalFit", "R");
260     finalFit->SetLineColor(kRed);
261     finalFit->Draw("lsame");
262     Double_t mean = finalFit->GetParameter(1);
263     Double_t errorMean = finalFit->GetParError(1);
264
265     Double_t sigma = finalFit->GetParameter(3);
266     Double_t errorSigma = finalFit->GetParError(3);
267     Double_t resolution = 2.35*100.*(sigma/mean);
268     Double_t resolutionError = (
269         ↪ sqrt((errorMean/mean)*(errorMean/mean) +
270             ↪ (errorSigma/sigma)*(errorSigma/sigma)));
269     cResultsDisplay->Update();
270     auto ps =
271         ↪ (TPaveStats*)adc->GetListOfFunctions()->FindObject("stats");
271     ps->SetName("mystats");
272     TList *listOfLines = ps->GetListOfLines();
273     TText *tconst = ps->GetLineWith("Resolution");
274     listOfLines->Remove(tconst);
275     TString d;
276     d = "Resolution = ";
277     d += Form("%.4g", resolution);
278     d += " #pm ";
279     d += Form("%.3f", (resolution*resolutionError));
280     d += " % FWHM";
281     TLatex *myt = new TLatex(0,0, d.Data());
282     myt ->SetTextFont(42);
283     myt ->SetTextSize(0.025);
284     myt ->SetTextColor(kBlack);
```

```

285     listOfLines->Add(myt);
286     adc->SetStats(0);
287     cResultsDisplay->Modified();
288
289     cout << "Fit range: " << lowerRange << " - " << upperRange <<
        ↪ endl;
290     cout << "Fit parameters: mean of " << mean << " and sigma of "
        ↪ << sigma << endl;
291
292     // Calculate energy resolution and error
293
294
295     cout << std::setiosflags(ios::fixed) << std::setprecision(3) <<
        ↪ "Resolution: " << resolution << " +/- " <<
        ↪ resolution*resolutionError << " % FWHM" << endl;
296
297 }
298 Double_t LanGausFunc(Double_t *x, Double_t *par) {
299
300     // Fit parameters:
301     Double_t lWidth = par[0]; // scale parameter of Landau density
        ↪ (to normalise Landau function)
302     Double_t lMP = par[1]; // most probable location of parameter
        ↪ of Landau density
303     Double_t norm = par[2]; // integral of total area
304     Double_t sigma = par[3]; // sigma of Gaussian function
305     Double_t rWidth = par[4]; // right handside Landau
306     Double_t norm2 = par[5];
307
308     // Numeric constants:
309     Double_t invsq2pi = 0.3989422804014; // (2 pi)^(-1/2)
310     Double_t mpshift = -0.22278298; // Landau maximum location
311
312     // Control constants:
313     Double_t np = 500.0; // number of convolution steps
314     //Double_t np = 200.0;
315     Double_t sc = 5.0; // convolution extends to +/-sc Gaussian
        ↪ sigmas
316

```

```

317     // Variables:
318     Double_t xx;
319     Double_t mpc;
320     Double_t fland;
321     Double_t sum = 0.0;
322     Double_t xlow, xupp;
323     Double_t step;
324     Double_t i;
325
326     // MP shift correction:
327
328     mpc = lMP - mpshift * lWidth;
329
330     // Range of convolution integral:
331     xlow = x[0] - sc * sigma;
332     xupp = x[0] + sc * sigma;
333     step = (xupp - xlow)/np; // bin width
334
335     // Convolution integral of Landau and Gaussian by sum:
336
337     for (i = 0; i < np/2; i++) {
338         xx = xlow + (i - .5) * step; // finding the value of the
339         ↪ middle of the bin
340         fland = TMath::Landau(-xx + 2*mpc, mpc, lWidth)/lWidth;
341         sum += (TMath::Gaus(x[0], xx, sigma)) * fland;
342
343         xx = xupp - (i - .5) * step;
344         fland = TMath::Landau(-xx + 2*mpc, mpc, lWidth)/lWidth;
345         sum += (TMath::Gaus(x[0], xx, sigma)) * fland;
346     }
347
348     Double_t result = (norm * step * sum * invsq2pi / sigma);
349     result = result + ((TMath::Landau(xx, mpc, rWidth)/rWidth)
350     ↪ *norm2);
351     return result;
352
353 void Print(TCanvas* canvas, const string readInFile){

```

```
354     TImage *img = TImage::Create();
355     img->FromPad(canvas);
356     img->WriteImage((readInFile+"_improved.png").c_str());
357 }
```

---

## B.2 Binary Data Conversion Jar

The code described below is used to convert and split a LeCroy binary file into multiple ASCII data files. The output files contain two columns; the timebase (in nanoseconds), and the voltage value (in millivolts).

---

```
1 package lecroy;
2
3 import java.io.*;
4 import java.nio.file.*;
5 import java.text.DecimalFormat;
6 import java.nio.ByteBuffer;
7 import java.nio.ByteOrder;
8 import java.util.*;
9
10 public class Reader {
11
12 //Instantiate these private variables once, they can be used throughout
13     ↪ all three methods.
14 private FileWriter fw;
15 private BufferedWriter bw;
16 private PrintWriter pw;
17
18 //This function is used to convert a specific number of files.
19 public void ReadConvertFiles(File[] listing, String output_directory, int
20     ↪ num_files) {
21 try{
22 long counter = 0;
23 //Loop over files whilst i < the number of user specified files to
24     ↪ convert.
25 for(int i = 0; i < num_files; i++) {
```

```
23 //Retrieves the required file from the listing.
24 File wavefile = listing[i];
25 BufferedReader reader = new BufferedReader(new
    ↪ FileReader(wavefile.toString()));
26 //Instantiate string to read data line by line.
27 String line = null;
28 InputStream bytread = null;
29 bytread = new FileInputStream(wavefile);
30 int WAVEDESC = 0;
31 //Read the header of the binary file.
32 while(WAVEDESC <= 0) {
33 line = reader.readLine();
34 CharSequence desc = "WAVEDESC";
35 WAVEDESC = line.indexOf((String)desc);
36 }
37 reader.close();
38 //Reads all the data from the binary file into memory.
39 byte[] bytedata = new byte[(int)wavefile.length()];
40 bytread.read(bytedata);
41 bytread.close();
42 ByteBuffer read = ByteBuffer.wrap(bytedata);
43 int aCOMM_ORDER = WAVEDESC+ 34; short COMM_ORDER
    ↪ = read.get(aCOMM_ORDER);
44 if(COMM_ORDER == 0) {
45 read = ByteBuffer.wrap(bytedata);
46 read.order(ByteOrder.BIG_ENDIAN);
47 }
48 else {
49 read = ByteBuffer.wrap(bytedata);
50 read.order(ByteOrder.LITTLE_ENDIAN);
51 }
52 //Instantiate required variables.
53 // String[] tmp = {"channel 1", "channel 2", "channel 3",
    ↪ "channel 4", "unknown"};
54 // String[] tmp2 = {"DC_50_Ohms", "ground ", "DC 1MOhm
    ↪ ","ground ", "AC 1MOhm "};
55 // String[] tmp3 = {"off", "on "};
```

```

56//      String[] tmp4 = {"single_sweep      ",      "interleaved
→      ", "histogram      ", "graph      ", "filter_coefficient",
→      "complex      ", "extrema      ", "sequence_obsolete
→      ", "centered_RIS      ", "peak_detect      "};
57//      String[] tmp5 = {"no_processing", "fir_filter      ", "interpolated
→      ", "sparsed      ", "autoscaled      ", "no_result      ", "rolling
→      ", "cumulative      "};
58//      int aTEMPLATE_NAME          = WAVEDESC+ 16;
59int aCOMM_TYPE          = WAVEDESC+ 32; short COMM_TYPE =
→ read.getShort(aCOMM_TYPE);
60int aWAVE_DESCRIPTOR    = WAVEDESC+ 36; long WAVE_DESCRIPTOR =
→ read.getLong(aWAVE_DESCRIPTOR);          // length of the descriptor
→ block short
61int aUSER_TEXT          = WAVEDESC+ 40; long USER_TEXT =
→ read.getLong(aUSER_TEXT); // length of the usertext block
62int aTRIGTIME_ARRAY    = WAVEDESC+ 48; long TRIGTIME_ARRAY =
→ read.getLong(aTRIGTIME_ARRAY);
63int aWAVE_ARRAY_1      = WAVEDESC+ 60; long WAVE_ARRAY_1 =
→ read.getLong(aWAVE_ARRAY_1); // length (in Byte) of the sample array
64//int aINSTRUMENT_NAME  = WAVEDESC+ 76; String INSTRUMENT_NAME =
→ ReadString(bytedata, aINSTRUMENT_NAME);
65//int aINSTRUMENT_NUMBER = WAVEDESC+ 92; long INSTRUMENT_NUMBER =
→ read.getLong(aINSTRUMENT_NUMBER);
66//int aTRACE_LABEL      = WAVEDESC+ 96;
67//      int aWAVE_ARRAY_COUNT      = WAVEDESC+ 116; long
→ WAVE_ARRAY_COUNT = read.getLong(aWAVE_ARRAY_COUNT);
68int aVERTICAL_GAIN      = WAVEDESC+ 156; float VERTICAL_GAIN =
→ read.getFloat(aVERTICAL_GAIN);
69int aVERTICAL_OFFSET    = WAVEDESC+ 160; float VERTICAL_OFFSET =
→ read.getFloat(aVERTICAL_OFFSET);
70//int aNOMINAL_BITS     = WAVEDESC+ 172; short NOMINAL_BITS =
→ read.getShort(aNOMINAL_BITS);
71int aHORIZ_INTERVAL     = WAVEDESC+ 176; float HORIZ_INTERVAL =
→ read.getFloat(aHORIZ_INTERVAL);
72int aHORIZ_OFFSET       = WAVEDESC+ 180; double HORIZ_OFFSET =
→ read.getDouble(aHORIZ_OFFSET);
73//int aVERTUNIT         = WAVEDESC+ 196;
74//int aHORUNIT          = WAVEDESC+ 244;

```

```

75 //int aTRIGGER_TIME          = WAVEDESC+ 296;String TRIGGER_TIME =
   → ReadTimestamp(read,aTRIGGER_TIME);
76 //int aRECORD_TYPE          = WAVEDESC+ 316;String RECORD_TYPE =
   → tmp4[read.getShort(aRECORD_TYPE)];
77 //int aPROCESSING_DONE      = WAVEDESC+ 318;String PROCESSING_DONE =
   → tmp5[read.getShort(aPROCESSING_DONE)];
78 //int aTIMEBASE             = WAVEDESC+ 324;String TIMEBASE =
   → Float_To_Eng((float)ReadTimebase(read,
   → aTIMEBASE)).toString()+"s/Div";
79 //int aVERT_COUPLING        = WAVEDESC+ 326;String VERT_COUPLING
   → = tmp[read.getShort(aVERT_COUPLING)];
80 //int aPROBE_ATT            = WAVEDESC+ 328;float PROBE_ATT
   → = read.getFloat(aPROBE_ATT);
81 //int aFIXED_VERT_GAIN      = WAVEDESC+ 332;float FIXED_VERT_GAIN =
   → Float.parseFloat(String.valueOf(ReadFixed_vert_gain(read,aFIXED_VERT_GAIN)));
82 //int aBANDWIDTH_LIMIT     = WAVEDESC+ 334;String BANDWIDTH_LIMIT =
   → tmp2[read.getShort(aBANDWIDTH_LIMIT)];
83 //int aVERTICAL_VERNIER     = WAVEDESC+ 336;
84 //int aACQ_VERT_OFFSET      = WAVEDESC+ 340;
85 //int aWAVE_SOURCE          = WAVEDESC+ 344;String WAVE_SOURCE =
   → tmp3[read.getShort(aWAVE_SOURCE)];
86 //String Gain_with_Probe =
   → Float_To_Eng(FIXED_VERT_GAIN*PROBE_ATT)+"V/Div";
87 //String SampleRate = Float_To_Eng(1/HORIZ_INTERVAL)+"S/Sec";
88
89 //Set the reader position to the start of the data.
90 read.position((int)(WAVEDESC+WAVE_DESCRIPTOR+USER_TEXT+TRIGTIME_ARRAY));
91 ArrayList<Float> y_values = new ArrayList<Float>();
92 //      ArrayList<Float> x_values = new ArrayList<Float>();
93
94 //Read data into arrays.
95 if(COMM_TYPE == 0) {
96 for(int j=0;j<WAVE_ARRAY_1;j++) {
97 float initial = 0;
98 if(j == 0) {
99 initial = (float)(j * HORIZ_INTERVAL - HORIZ_OFFSET);
100 y_values.add((VERTICAL_GAIN*(int)read.get() - VERTICAL_OFFSET)*1000);
101 //      x_values.add((float)(0.0));
102

```



```
103}
104else{
105y_values.add((VERTICAL_GAIN*(int)read.get() - VERTICAL_OFFSET)*1000);
106//x_values.add((float)(((j * HORIZ_INTERVAL) - HORIZ_OFFSET +
    ↪ initial)*Math.pow(10, 9)));
107}
108}
109}
110//Output data.
111for(long k = 0; k<(WAVE_ARRAY_1/1000);k++) {
112String file2 = output_directory+"WAVEFILE"+counter+".dat";
113File outputFile = new File(file2);
114fw = new FileWriter(outputFile);
115bw = new BufferedWriter(fw);
116pw = new PrintWriter(bw);
117counter++;
118for(int l = 0; l < 1000; l++){
119pw.println((((float)(l*HORIZ_INTERVAL)*Math.pow(10, 9))+
    ↪ "+-1*y_values.get(l)));
120}
121pw.close();
122}
123}
124}
125catch (Exception e) {
126e.printStackTrace();
127}
128
129}
130
131//Converts all files in the specified directory.
132public void ReadConvertAllFilesSplit(File[] listing, String
    ↪ output_directory) {
133try {
134long counter = 0;
135for(int i = 0; i< listing.length;i++) {
136File wavefile = listing[i];
137BufferedReader reader = new BufferedReader(new
    ↪ FileReader(wavefile.toString()));
```

```

138 String line = null;
139 InputStream byteread = null;
140 byteread = new FileInputStream(wavefile);
141 int WAVEDESC = 0;
142 while(WAVEDESC <= 0) {
143 line = reader.readLine();
144 CharSequence desc = "WAVEDESC";
145 WAVEDESC = line.indexOf((String)desc);
146 }
147 reader.close();
148 byte[] bytedata = new byte[(int)wavefile.length()];
149 byteread.read(bytedata);
150 byteread.close();
151 ByteBuffer read = ByteBuffer.wrap(bytedata);
152 int aCOMM_ORDER = WAVEDESC+ 34; short COMM_ORDER
    ↪ = read.get(aCOMM_ORDER);
153 if(COMM_ORDER == 0) {
154 read = ByteBuffer.wrap(bytedata);
155 read.order(ByteOrder.BIG_ENDIAN);
156 }
157 else {
158 read = ByteBuffer.wrap(bytedata);
159 read.order(ByteOrder.LITTLE_ENDIAN);
160 }
161
162 //      String[] tmp = {"channel 1", "channel 2", "channel 3",
    ↪ "channel 4", "unknown"};
163 //      String[] tmp2 = {"DC_50_Ohms", "ground ", "DC 1M0hm
    ↪ ", "ground ", "AC 1M0hm "};
164 //      String[] tmp3 = {"off", "on "};
165 //      String[] tmp4 = {"single_sweep ", "interleaved
    ↪ ", "histogram ", "graph ", "filter_coefficient",
    ↪ "complex ", "extrema ", "sequence_obsolete
    ↪ ", "centered_RIS ", "peak_detect "};
166 //      String[] tmp5 = {"no_processing", "fir_filter ", "interpolated
    ↪ ", "sparsed ", "autoscaled ", "no_result ", "rolling
    ↪ ", "cumulative "};
167 //      int aTEMPLATE_NAME = WAVEDESC+ 16;

```

```

168 int aCOMM_TYPE = WAVEDESC+ 32; short COMM_TYPE =
    → read.getShort(aCOMM_TYPE);
169 int aWAVE_DESCRIPTOR = WAVEDESC+ 36; long WAVE_DESCRIPTOR =
    → read.getLong(aWAVE_DESCRIPTOR); // length of the descriptor
    → block short
170 int aUSER_TEXT = WAVEDESC+ 40; long USER_TEXT =
    → read.getLong(aUSER_TEXT); // length of the usertext block
171 int aTRIGTIME_ARRAY = WAVEDESC+ 48; long TRIGTIME_ARRAY =
    → read.getLong(aTRIGTIME_ARRAY);
172 int aWAVE_ARRAY_1 = WAVEDESC+ 60; long WAVE_ARRAY_1 =
    → read.getLong(aWAVE_ARRAY_1); // length (in Byte) of the sample array
173 // int aINSTRUMENT_NAME = WAVEDESC+ 76; String
    → INSTRUMENT_NAME = ReadString(bytedata, aINSTRUMENT_NAME);
174 // int aINSTRUMENT_NUMBER = WAVEDESC+ 92; long INSTRUMENT_NUMBER
    → = read.getLong(aINSTRUMENT_NUMBER);
175 // int aTRACE_LABEL = WAVEDESC+ 96;
176 // int aWAVE_ARRAY_COUNT = WAVEDESC+ 116; long
    → WAVE_ARRAY_COUNT = read.getLong(aWAVE_ARRAY_COUNT);
177 int aVERTICAL_GAIN = WAVEDESC+ 156; float VERTICAL_GAIN =
    → read.getFloat(aVERTICAL_GAIN);
178 int aVERTICAL_OFFSET = WAVEDESC+ 160; float VERTICAL_OFFSET =
    → read.getFloat(aVERTICAL_OFFSET);
179 // int aNOMINAL_BITS = WAVEDESC+ 172; short
    → NOMINAL_BITS = read.getShort(aNOMINAL_BITS);
180 int aHORIZ_INTERVAL = WAVEDESC+ 176; float HORIZ_INTERVAL =
    → read.getFloat(aHORIZ_INTERVAL);
181 int aHORIZ_OFFSET = WAVEDESC+ 180; double HORIZ_OFFSET =
    → read.getDouble(aHORIZ_OFFSET);
182 // int aVERTUNIT = WAVEDESC+ 196;
183 // int aHORUNIT = WAVEDESC+ 244;
184 // int aTRIGGER_TIME = WAVEDESC+ 296; String
    → TRIGGER_TIME = ReadTimestamp(read, aTRIGGER_TIME);
185 // int aRECORD_TYPE = WAVEDESC+ 316; String
    → RECORD_TYPE = tmp4[read.getShort(aRECORD_TYPE)];
186 // int aPROCESSING_DONE = WAVEDESC+ 318; String
    → PROCESSING_DONE = tmp5[read.getShort(aPROCESSING_DONE)];
187 // int aTIMEBASE = WAVEDESC+ 324; String
    → TIMEBASE = Float_To_Eng((float)ReadTimebase(read,
    → aTIMEBASE)).toString()+"s/Div";

```

```

188 //      int aVERT_COUPLING          = WAVEDESC+ 326;String
    → VERT_COUPLING = tmp[read.getShort(aVERT_COUPLING)];
189 int aPROBE_ATT          = WAVEDESC+ 328;float PROBE_ATT =
    → read.getFloat(aPROBE_ATT);
190 int aFIXED_VERT_GAIN    = WAVEDESC+ 332;float FIXED_VERT_GAIN =
    → Float.parseFloat(String.valueOf(ReadFixed_vert_gain(read,aFIXED_VERT_GAIN)));
191 //      int aBANDWIDTH_LIMIT        = WAVEDESC+ 334;String
    → BANDWIDTH_LIMIT = tmp2[read.getShort(aBANDWIDTH_LIMIT)];
192 //      int aVERTICAL_VERNIER        = WAVEDESC+ 336;
193 //      int aACQ_VERT_OFFSET        = WAVEDESC+ 340;
194 //      int aWAVE_SOURCE              = WAVEDESC+ 344;String
    → WAVE_SOURCE = tmp3[read.getShort(aWAVE_SOURCE)];
195 //      String Gain_with_Probe =
    → Float_To_Eng(FIXED_VERT_GAIN*PROBE_ATT)+"V/Div";
196 //      String SampleRate = Float_To_Eng(1/HORIZ_INTERVAL)+"S/Sec";
197 read.position((int)(WAVEDESC+WAVE_DESCRIPTOR+USER_TEXT+TRIGTIME_ARRAY));
198 ArrayList<Float> y_values = new ArrayList<Float>();
199 //      ArrayList<Float> x_values = new ArrayList<Float>();
200 if(COMM_TYPE == 0) {
201 for(int j=0;j<WAVE_ARRAY_1;j++) {
202 float initial = 0;
203 if(j == 0) {
204 initial = (float)(j * HORIZ_INTERVAL - HORIZ_OFFSET);
205 y_values.add((VERTICAL_GAIN*(int)read.get() - VERTICAL_OFFSET)*1000);
206 //      x_values.add((float)(0.0));
207
208 }
209 else{
210 y_values.add((VERTICAL_GAIN*(int)read.get() - VERTICAL_OFFSET)*1000);
211 //x_values.add((float)(((j * HORIZ_INTERVAL) - HORIZ_OFFSET +
    → initial)*Math.pow(10, 9)));
212 }
213 }
214 }
215 else{
216 for(int j=0;j<WAVE_ARRAY_1;j++) {
217 float initial = 0;
218 if(j == 0) {
219 initial = (float)(j * HORIZ_INTERVAL - HORIZ_OFFSET);

```

```

220 y_values.add((VERTICAL_GAIN*(int)read.get() - VERTICAL_OFFSET)*1000);
221 //x_values.add((float)(0.0));
222
223 }
224 else{
225 y_values.add((VERTICAL_GAIN*(int)read.get() - VERTICAL_OFFSET)*1000);
226 //x_values.add((float)(((j * HORIZ_INTERVAL) - HORIZ_OFFSET +
    → initial)*Math.pow(10, 9)));
227 }
228 }
229 }
230 // int m = 0;
231 // String file2 =
    → output_directory+"WAVEFILE"+counter+".txt";
232 // File outputFile = new File(file2);
233 // fw = new FileWriter(outputFile);
234 // bw = new BufferedWriter(fw);
235 // pw = new PrintWriter(bw);
236 // while(m < y_values.size()) {
    → float y_val = y_values.get(m);
    → DecimalFormat decimalFormat =
    → new DecimalFormat("#.##");
239 // pw.println(Float.valueOf(decimalFormat.form
    → m++;
240 // }
241 // pw.close();
242 int m = 0;
243 while(m < y_values.size()){
244 for(long k = 0; k<(WAVE_ARRAY_1/1000);k++) {
245 String file2 = output_directory+"WAVEFILE"+counter+".dat";
246 File outputFile = new File(file2);
247 fw = new FileWriter(outputFile);
248 bw = new BufferedWriter(fw);
249 pw = new PrintWriter(bw);
250 counter++;
251 for(int l = 0; l < 1000; l++){
252 float y_val = y_values.get(m); m++;
253 float x_val = (float)((l*HORIZ_INTERVAL)*Math.pow(10, 9));
254 //Format the values to remove unnecessary information.

```

```
255 DecimalFormat decimalFormat = new DecimalFormat("#.##");
256 DecimalFormat decimalFormat2 = new DecimalFormat("#.####");
257 pw.println(Float.valueOf(decimalFormat.format(x_val))+
    ↪ "+-1.0*Float.valueOf(decimalFormat2.format(y_val)));
258 }
259 pw.close();
260 }
261 }
262
263 }
264 }
265 catch(Exception e) {
266 e.printStackTrace();
267 }
268
269 }
270
271 //This method prints all the data into one file. Provided for
    ↪ functionality but not necessary.
272 public void ReadConvertAllFiles(File[] listing, String
    ↪ output_directory, int num_files) {
273 ArrayList<Float> y_values = new ArrayList<Float>();
274 ArrayList<Float> x_values = new ArrayList<Float>();
275 String file2 = output_directory+"WAVEFILE_ALL.dat";
276 File outputFile = new File(file2);
277 try{
278 fw = new FileWriter(outputFile);
279 bw = new BufferedWriter(fw);
280 pw = new PrintWriter(bw);
281 for(int i = 0; i< num_files;i++) {
282 File wavefile = listing[i];
283 BufferedReader reader = new BufferedReader(new
    ↪ FileReader(wavefile.toString()));
284 String line = null;
285 InputStream bytread = null;
286 bytread = new FileInputStream(wavefile);
287 int WAVEDESC = 0;
288 while(WAVEDESC <= 0) {
289 line = reader.readLine();
```

```

290 CharSequence desc = "WAVEDESC";
291 WAVEDESC = line.indexOf((String)desc);
292 }
293 reader.close();
294 byte[] bytedata = new byte[(int)wavfile.length()];
295 bytread.read(bytedata);
296 bytread.close();
297 ByteBuffer read = ByteBuffer.wrap(bytedata);
298 int aCOMM_ORDER = WAVEDESC+ 34; short COMM_ORDER
    ↪ = read.get(aCOMM_ORDER);
299 if(COMM_ORDER == 0) {
300 read = ByteBuffer.wrap(bytedata);
301 read.order(ByteOrder.BIG_ENDIAN);
302 }
303 else {
304 read = ByteBuffer.wrap(bytedata);
305 read.order(ByteOrder.LITTLE_ENDIAN);
306 }
307
308 String[] tmp = {"channel 1","channel 2", "channel 3", "channel 4",
    ↪ "unknown"};
309 String[] tmp2 = {"DC_50_0hms", "ground ", "DC 1M0hm ", "ground
    ↪ ", "AC 1M0hm "};
310 String[] tmp3 = {"off", "on "};
311 String[] tmp4 = {"single_sweep ", "interleaved ",
    ↪ "histogram ", "graph ", "filter_coefficient",
    ↪ "complex ", "extrema ", "sequence_obsolete
    ↪ ", "centered_RIS ", "peak_detect "};
312 String[] tmp5 = {"no_processing", "fir_filter ", "interpolated
    ↪ ", "sparsed ", "autoscaled ", "no_result ", "rolling
    ↪ ", "cumulative "};
313 // int aTEMPLATE_NAME = WAVEDESC+ 16;
314 int aCOMM_TYPE = WAVEDESC+ 32; short COMM_TYPE =
    ↪ read.getShort(aCOMM_TYPE);
315 int aWAVE_DESCRIPTOR = WAVEDESC+ 36; long WAVE_DESCRIPTOR =
    ↪ read.getLong(aWAVE_DESCRIPTOR); // length of the descriptor
    ↪ block short
316 int aUSER_TEXT = WAVEDESC+ 40; long USER_TEXT =
    ↪ read.getLong(aUSER_TEXT); // length of the usertext block

```

```

317 int aTRIGTIME_ARRAY      = WAVEDESC+ 48; long TRIGTIME_ARRAY =
    → read.getLong(aTRIGTIME_ARRAY);
318 int aWAVE_ARRAY_1        = WAVEDESC+ 60; long WAVE_ARRAY_1 =
    → read.getLong(aWAVE_ARRAY_1); // length (in Byte) of the sample array
319 //      int aINSTRUMENT_NAME      = WAVEDESC+ 76; String
    → INSTRUMENT_NAME = ReadString(bytedata, aINSTRUMENT_NAME);
320 //      int aINSTRUMENT_NUMBER    = WAVEDESC+ 92; long INSTRUMENT_NUMBER
    → = read.getLong(aINSTRUMENT_NUMBER);
321 //      int aTRACE_LABEL          = WAVEDESC+ 96;
322 //      int aWAVE_ARRAY_COUNT      = WAVEDESC+ 116; long
    → WAVE_ARRAY_COUNT = read.getLong(aWAVE_ARRAY_COUNT);
323 int aVERTICAL_GAIN       = WAVEDESC+ 156; float VERTICAL_GAIN =
    → read.getFloat(aVERTICAL_GAIN);
324 int aVERTICAL_OFFSET    = WAVEDESC+ 160; float VERTICAL_OFFSET =
    → read.getFloat(aVERTICAL_OFFSET);
325 //      int aNOMINAL_BITS         = WAVEDESC+ 172; short
    → NOMINAL_BITS = read.getShort(aNOMINAL_BITS);
326 int aHORIZ_INTERVAL     = WAVEDESC+ 176; float HORIZ_INTERVAL =
    → read.getFloat(aHORIZ_INTERVAL);
327 int aHORIZ_OFFSET       = WAVEDESC+ 180; double HORIZ_OFFSET =
    → read.getDouble(aHORIZ_OFFSET);
328 //      int aVERTUNIT              = WAVEDESC+ 196;
329 //      int aHORUNIT              = WAVEDESC+ 244;
330 //      int aTRIGGER_TIME         = WAVEDESC+ 296; String
    → TRIGGER_TIME = ReadTimestamp(read, aTRIGGER_TIME);
331 //      int aRECORD_TYPE          = WAVEDESC+ 316; String
    → RECORD_TYPE = tmp4[read.getShort(aRECORD_TYPE)];
332 //      int aPROCESSING_DONE      = WAVEDESC+ 318; String
    → PROCESSING_DONE = tmp5[read.getShort(aPROCESSING_DONE)];
333 //      int aTIMEBASE              = WAVEDESC+ 324; String
    → TIMEBASE = Float_To_Eng((float)ReadTimebase(read,
    → aTIMEBASE)).toString()+"s/Div";
334 //      int aVERT_COUPLING        = WAVEDESC+ 326; String
    → VERT_COUPLING = tmp[read.getShort(aVERT_COUPLING)];
335 int aPROBE_ATT           = WAVEDESC+ 328; float PROBE_ATT =
    → read.getFloat(aPROBE_ATT);
336 int aFIXED_VERT_GAIN     = WAVEDESC+ 332; float FIXED_VERT_GAIN =
    → Float.parseFloat(String.valueOf(ReadFixed_vert_gain(read, aFIXED_VERT_GAIN)));

```



```

337//      int aBANDWIDTH_LIMIT      = WAVEDESC+ 334;String
   → BANDWIDTH_LIMIT = tmp2[read.getShort(aBANDWIDTH_LIMIT)];
338//      int aVERTICAL_VERNIER      = WAVEDESC+ 336;
339//      int aACQ_VERT_OFFSET       = WAVEDESC+ 340;
340//      int aWAVE_SOURCE            = WAVEDESC+ 344;String
   → WAVE_SOURCE = tmp3[read.getShort(aWAVE_SOURCE)];
341//      String Gain_with_Probe =
   → Float_To_Eng(FIXED_VERT_GAIN*PROBE_ATT)+"V/Div";
342//      String SampleRate = Float_To_Eng(1/HORIZ_INTERVAL)+"S/Sec";
343read.position((int)(WAVEDESC+WAVE_DESCRIPTOR+USER_TEXT+TRIGTIME_ARRAY));
344if(COMM_TYPE == 0) {
345for(int j=0;j<WAVE_ARRAY_1;j++) {
346y_values.add((VERTICAL_GAIN*(int)read.get() - VERTICAL_OFFSET));
347x_values.add((float)(j * HORIZ_INTERVAL + HORIZ_OFFSET));
348}
349}
350else{
351for(int j=0;j<WAVE_ARRAY_1;j++) {
352y_values.add((VERTICAL_GAIN*(int)read.get() - VERTICAL_OFFSET));
353x_values.add((float)(j * HORIZ_INTERVAL + HORIZ_OFFSET));
354}
355}
356
357}
358for(int k = 0; k<y_values.size();k++) {
359pw.println((-1.0*y_values.get(k)+"      "+x_values.get(k)));
360}
361pw.close();
362
363}
364catch (Exception e) {
365
366}
367
368}
369
370//Method to produce a file listing of a directory.
371public File[] getFiles(Path filePath) {
372File[] listing = null;

```

```
373 try {
374 File dir = new File(filesPath.toString());
375 listing = dir.listFiles();
376 }
377 catch (Exception e) {e.printStackTrace();}
378 return listing;
379 }
380
381 //Converts binary data to a string.
382 public String ReadString(byte[] data, int addr) {
383 StringBuilder sb = new StringBuilder();
384 int i = addr; int j = i + 16;
385 while (i < j) {
386 byte character = data[i];
387 char c = (char) (character & 0xFF);
388 sb.append(c);
389 i++;
390 }
391 return sb.toString();
392 }
393
394 //Converts binary data to a timestamp.
395 public String ReadTimestamp(ByteBuffer b, int addr) {
396 String stamp; b.position(addr);
397 double seconds = b.getDouble(); byte minutes = b.get();
398 byte hours = b.get(); byte days = b.get();
399 byte months = b.get(); short year = b.getShort();
400 stamp = days + "." + months + "." + year + " ,
    ↪  " + hours + ":" + minutes + ":" + (int) seconds;
401 return stamp;
402 }
403
404 //Reads the timebase of the data.
405 public double ReadTimebase(ByteBuffer b, int addr) {
406 short e = b.getShort(addr);
407 int[] tmp = {1, 2, 5};
408 int mant = tmp[e % 3];
409 int ex = Math.floorDiv(e, 3) - 12 ;
410 double val = mant * Math.pow(10, ex);
```

```
411 return val;
412 }
413
414 //Reads the vertical gain of the data.
415 public double ReadFixed_vert_gain(ByteBuffer b, int addr) {
416 double vert_gain;  short e = b.getShort(addr);
417 int[] tmp = {1,2,5};  int mant = tmp[e%3];
418 int ex = Math.floorDiv(e, 3) -6;  vert_gain = mant* Math.pow(10, ex);
419 return vert_gain;
420 }
421
422 //Function to convert a float to its unit. Provided for functionality
→ but currently unused.
423 public String Float_To_Eng(float i) {
424 float ex = (float) Math.floor(Math.log10(i));
425 float exeng = ex - (Math.floorMod((int)ex, 3));
426 if(exeng<-18) { exeng = -18;}
427 if(exeng>18) {exeng = 18;}
428 float mant =(float) (i/(Math.pow(10, exeng)));
429 String[] prefix = {"a","f","p","n","u","m","","k","M","G","P","E"};
430 String result = mant+prefix[(int)(exeng+18)/3];
431 return result;
432 }
433
434 //Determines if a parsed string is an integer.
435 public boolean isStringInt(String s)
436 {
437 try
438 {
439 Integer.parseInt(s);
440 return true;
441 } catch (NumberFormatException ex)
442 {
443 return false;
444 }
445 }
446
447 //This main function should not be changed to retain command line
→ functionality.
```

```
448 public static void main(String[] args) {
449     Reader r = new Reader();
450     //Get current path incase it is not provided by user.
451     Path directoryPath; Path currentPath =
452         ↪ Paths.get(".").toAbsolutePath().normalize();
453     String outputDirectory;
454     String split;
455     int numFiles;
456     int numArgs = args.length;
457     //Default response of program if no arguments are provided.
458     if(numArgs == 0) {
459         directoryPath = currentPath;
460         outputDirectory = currentPath.toString();
461         numFiles = r.GetFiles(directoryPath).length;
462         r.ReadConvertAllFiles(r.GetFiles(directoryPath), outputDirectory, numFiles);
463     }
464     if(numArgs == 1) {
465         if(r.isStringInt(args[0].toString()) == true) {
466             numFiles = Integer.parseInt(args[0].toString());
467             directoryPath = currentPath;
468             outputDirectory = currentPath.toString();
469             r.ReadConvertFiles(r.GetFiles(directoryPath), outputDirectory, numFiles);
470         }
471         else{
472             directoryPath = Paths.get(args[0].toString());
473             outputDirectory = args[0].toString();
474             numFiles = r.GetFiles(directoryPath).length;
475             r.ReadConvertAllFiles(r.GetFiles(directoryPath), outputDirectory,
476                 ↪ numFiles);
477         }
478     }
479     if(numArgs == 2) {
480         if(r.isStringInt(args[1].toString()) == true) {
481             numFiles = Integer.parseInt(args[1].toString());
482             directoryPath = Paths.get(args[0].toString());
483             outputDirectory = args[0].toString();
484             r.ReadConvertFiles(r.GetFiles(directoryPath), outputDirectory, numFiles);
485         }
486         else{
```

```
485 directoryPath = Paths.get(args[0].toString());
486 outputDirectory = args[1].toString();
487 numFiles = r.GetFiles(directoryPath).length;
488 r.ReadConvertAllFiles(r.GetFiles(directoryPath), outputDirectory, numFiles);
489 }
490 }
491 if(numArgs == 3) {
492 directoryPath = Paths.get(args[0].toString());
493 outputDirectory = args[1].toString();
494 numFiles = Integer.parseInt(args[2].toString());
495 r.ReadConvertFiles(r.GetFiles(directoryPath), outputDirectory, numFiles);
496 }
497 if(numArgs == 4) {
498 directoryPath = Paths.get(args[0].toString());
499 outputDirectory = args[1].toString();
500 numFiles = Integer.parseInt(args[2].toString());
501 split = args[3].toString();
502 if(split.equalsIgnoreCase("split")) {
503 r.ReadConvertAllFilesSplit(r.GetFiles(directoryPath), outputDirectory);
504 }
505 else{
506 r.ReadConvertFiles(r.GetFiles(directoryPath), outputDirectory,
    ↪ numFiles);
507 }
508 }
509 }
510
511 }
```

---

### B.3 Waveform Analysis Macro

The code described below is the ROOT macro that was used to read in multiple waveform data files from a directory, analyse the files for multiple peaks, and also to integrate the pulses in order to produce an ADC spectrum (although this functionality is not complete). The main output of this file is a ROOT file that contains two TTree structures. One of the TTrees contains the "good" pulses and the ADC spectrum that is reproduced. The other TTree contains the "bad" pulses that are kept for reference.

---

```
1 #include <iostream>
2 #include <iomanip>
3 #include <fstream>
4 #include <string>
5 #include <cstring>
6 #include <vector>
7 #include <map>
8 #include <algorithm>
9 #include "TLine.h"
10 #include <sstream>
11 #include <cstdlib>
12 #include <math.h>
13 #include <stdio.h>
14 #include <stdlib.h>
15 #include <sys/types.h>
16 #include "TSystem.h"
17 #include "TSpectrum.h"
18 #include "TCanvas.h"
19 #include "TF1.h"
20 #include "TFile.h"
21 #include "TGraph.h"
22 #include "TH1D.h"
23 #include "TLegend.h"
24 #include "TMath.h"
25 #include "TTree.h"
26 #include "TStyle.h"
27 #include "TPaveStats.h"
28
29 using namespace std;
30
31 Double_t GetPeak(TH1D* wave);
32 void fitattempt(TH1D* wave, Double_t peak, Double_t lowerRange, Double_t
   ↪ upperRange);
33 Double_t gausPart(Double_t *x, Double_t *par);
34
35
36 void waveform(string s) {
37     ifstream resultsFile;
```

```
38     ofstream currentOutFile;
39
40     //Define the Tree structures that will contain the good and bad
41     ↪ waveforms
42     TTree* waveform_tree = new TTree("waveform_tree", "Tree
43     ↪ containing waveform data");
44     TTree* bad_waves = new TTree("bad_waves", "Tree containing bad
45     ↪ waveforms");
46     //Define the output file
47     TFile* output_file = new TFile("testtree.root", "RECREATE");
48
49     //Open the directory containing the individual waveform files
50     void *dirp = gSystem->OpenDirectory (s.c_str());
51     //Variable that will hold the file name
52     const char* afile;
53     //Skip the first two entries on Linux
54     gSystem->GetDirEntry(dirp);
55     gSystem->GetDirEntry(dirp);
56     //Keeps count of the number of bad waves
57     Int_t badwaves = 0;
58     //Vectors to store the integral number, adc number, and number
59     ↪ of counts for each waveform
60     vector<Int_t> integrals;
61     vector<Int_t> events;
62     vector<Int_t> counts;
63     //Loop over all files in specified directory
64     while ((afile = gSystem->GetDirEntry(dirp))) {
65         //Instantiate histograms for either a good or bad pulse
66         TH1D *wave = new TH1D("Good Wave","Waveform
67         ↪ Pulse",1000,0,1000);
68         TH1D *badwave = new TH1D("Bad Wave","Waveform
69         ↪ Pulse",1000,0,1000);
70         //Set waveform file name
71         string str(afile);
72         string file_name = s+(str);
73         //Open file containing data
74         resultsFile.open(file_name.c_str());
75         //This string will hold the data
76         string data;
```

```
71      //Keep track of number of ppoints
72      int pointsCounter = 0;
73      int counter = 0;
74      //Read in data
75      if(resultsFile.is_open()){
76          Int_t bin = 0;
77          Int_t maxValue = 0;
78          while(getline(resultsFile,data)){
79              stringstream dataReadin;
80              dataReadin << data;
81              ++pointsCounter;
82              Int_t time_val = 0;
83              Int_t volt_val = 0;
84              dataReadin >> time_val >> volt_val;
85              //Find maximum voltage value
86              if(volt_val > maxValue) {maxValue =
87                  ↪ volt_val;}
88          }
89          const unsigned int nLines = pointsCounter;
90          resultsFile.clear();
91          resultsFile.seekg(0, ios::beg);
92          Int_t timebase_value[pointsCounter],
93              ↪ voltage_value[pointsCounter];
94          //Fill the histogram with data
95          while (getline(resultsFile, data)) {
96              stringstream dataReadin;
97              dataReadin << data;
98              Int_t timeStamp = 0;
99              Int_t voltage = 0;
100
101
102              dataReadin >> timeStamp >> voltage;
103
104              // Fill arrays and tree
105              timebase_value[counter] = timeStamp;
106              voltage_value[counter] = voltage;
```



```
107         //This flips the sign of the voltage to
           ↪ make the pulse positive
108         wave->SetBinContent(timeStamp, (-1. *
           ↪ voltage));
109         counter++;
110     }
111
112     //TGraph *wave1 = new
           ↪ TGraph(counter, timebase_value,
           ↪ voltage_value);
113
114     //Instantiate searching method
115     TSpectrum *s = new TSpectrum();
116     //Find number of peaks in current waveform
117     Int_t nfound =
           ↪ s->Search(wave, 1.0, "nobackground", 0.40);
118     //Find largest peak
119     Double_t peak = GetPeak(wave);
120
121     //If number of peaks is greater than one it's a
           ↪ bad waveform
122     if(nfound > 1) {
123         //TCanvas* canvas = new
           ↪ TCanvas(afile, afile, 160, 160, 900, 675);
124         badwaves++;
125         badwave = wave;
126         badwave->SetDirectory(0);
127         badwave->GetXaxis()->SetTitle("Time
           ↪ (ns)");
128         badwave->GetYaxis()->SetTitle("Voltage
           ↪ (mV)");
129         badwave->GetYaxis()->SetTitleOffset(1.3);
130         badwave->GetYaxis()->CenterTitle();
131         //wave->GetXaxis()->SetNDivisions(506);
132         badwave->Draw();
133         //canvas->Print(("plots/"+str+".png").c_str());
134         badwave->Write();
135         resultsFile.close();
136         output_file->cd();
```

```

137         bad_waves->Write();
138     }
139     //Else good waveform
140     else{
141         wave->SetDirectory(0);
142         wave->GetXaxis()->SetTitle("Time (ns)");
143         wave->GetYaxis()->SetTitle("Voltage
144             ↪ (mV)");
145         wave->GetYaxis()->SetTitleOffset(1.3);
146         wave->GetYaxis()->CenterTitle();
147         //wave->GetXaxis()->SetNdivisions(506);
148         //wave->Write();
149         wave->Draw();
150         //fitattempt(wave, peak, 0, 455);
151         //Integrate the waveform, 80-250 ns is
152         ↪ the gate over which integration
153         ↪ takes place
154         Double_t integral =
155             ↪ wave->Integral(80.0,250.0,"width");
156         //Append to vector
157         integrals.push_back(integral);
158         resultsFile.close();
159         output_file->cd();
160         waveform_tree->Write();
161     }
162 }
163 }
164 //Print number of bad waveforms
165 cout << "Number of bad waveforms " << badwaves << endl;
166 //Sort the integrals in size order from smallest to largest
167 sort(integrals.begin(),integrals.end());
168 //for(std::size_t i = 0; i != integrals.size(); ++i) {
169 //cout << "integral value = " << integrals[i] << "\n";
170 //}
171 Int_t prev = 0;
172 Int_t current = 0;
173 Int_t count = 0; //keeps track of number of events

```

```

171     Int_t eventnumber = -1;           //keeps track of what event index
        ↪ we're looking at
172     for(std::size_t i = 0; i != integrals.size(); ++i) {
173         current = integrals[i]; //set the current value we are
        ↪ looking at
174         if(prev == current) { //check if current value equals
        ↪ previous one
175             //cout <<"prev val = "<<prev<<" cur val = "<<
        ↪ current<<"\n";
176             count = events[eventnumber]; //recall the count
        ↪ for the specified event
177             //cout << "count = " << count << "\n";
178             events[eventnumber] = count + 1; //increment by
        ↪ 1 as we have another count
179             prev = current;           //set the previous value
        ↪ as the current one
180         }
181         else {
182             Int_t initialval = 1;
183             events.push_back(initialval);
184             counts.push_back(current);
185             eventnumber = eventnumber + 1;
186             prev = current;
187         }
188     }
189     //Produce ADC spectrum
190     TH1D *adc = new TH1D("ADC", "ADC Spectrum", 1000, 0, 1000);
191     cout << "Number of counts = " << counts.size();
192     int numevents = 0;
193     for(std::size_t i = 0; i != events.size(); i++) {
194         adc->SetBinContent(counts[i], events[i]);
195         numevents = numevents+events[i];
196     }
197     cout << "Total number of events = " << numevents;
198     adc->SetDirectory(0);
199     adc->GetXaxis()->SetTitle("ADC Counts");
200     adc->GetYaxis()->SetTitle("Number of Events");
201     adc->GetYaxis()->SetTitleOffset(1.3);
202     adc->GetYaxis()->CenterTitle();

```

```
203     adc->Draw();
204     adc->Write();
205
206     output_file->Close();
207
208 }
209
210 Double_t GetPeak(TH1D* wave) {
211     //Find peak to get parameters for fit
212     Double_t peak = (Double_t)wave->GetMaximumBin();
213     // cout << "Peak at " << peak << " ns" << endl;
214     return peak;
215 }
216
217 void fitattempt(TH1D* wave, Double_t peak, Double_t lowerRange, Double_t
    ↪ upperRange) {
218     Double_t gaussmean = peak;
219     Double_t sigma = sqrt(peak);
220     Double_t Amp = (Double_t)wave->GetBinContent((Int_t)peak);
221     Double_t cutoff = sigma/2.0;
222     Double_t expfall = -0.2;
223
224
225     TF1* fitter = new TF1("fitter",
226                           gausPart,
227                           lowerRange,
228                           upperRange,
229                           5);
230     fitter->SetParameter(0,gaussmean);
231     fitter->SetParameter(1,sigma);
232     fitter->SetParameter(2,Amp);
233     fitter->SetParameter(3,cutoff);
234     fitter->SetParameter(4,expfall);
235     wave->Fit("fitter","R");
236     fitter->SetLineColor(kRed);
237     fitter->Draw("lsame");
238     wave->Write();
239 }
240
```

```
241 Double_t gausPart(Double_t *x, Double_t *par) {
242     Double_t cutoff = par[3] + par[1];
243     Double_t xdat;
244     Double_t ydat;
245     Double_t rescaledAmp = 0.0;
246     if(x[0] <= cutoff) {
247         xdat = x[0] ;
248         Double_t arg = (xdat - par[0])/par[1];
249         rescaledAmp = par[2]/TMath::Gaus(par[0],par[0],par[1]);
250         ydat = rescaledAmp * TMath::Gaus(xdat,par[0],par[1]);
251     }
252     else{
253         xdat = x[0] ;
254         Double_t gaussExpAmp =
255             ↪ rescaledAmp*TMath::Gaus(xdat,par[0],par[1]);
256         Double_t expFrac = 0.25;
257         Double_t expAmp = gaussExpAmp ;
258         Double_t C = expFrac*expAmp;
259         Double_t A = expAmp - C;
260         Double_t B = par[4];
261         //xdat = xdat- x[0];
262         ydat = A*TMath::Exp(B*xdat);
263     }
264     return ydat;
265 }
266
267 //Double_t expPart(Double_t *x, Double_t *par) {
268 //}
```

---