Masters Project Final Report

Design of a Detector for Fast Treatment Plan Verification in Proton Radiotherapy

Department of Physics & Astromony

University College London

Daniel Walker

PHASM201 for MSci Physics

supervised by

Dr. Simon Jolly

Prof. Ruben Saakyan

27th September 2018 - 26th March 2018

UCL DEPARTMENT OF PHYSICS AND ASTRONOMY

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 Daniel Walker

Dated 26 / 03 / 2018

Title	Date Received	Examiner	Examiner's Signature	MARK

University College London, Gower Street, London WC1E 6BT
Tel: +44 (0)20 7679 2000
email@ucl.ac.uk
www.ucl.ac.uk

Abstract: The proton beam therapy group at UCL is developing a compact scintillatior calorimeter for use in treatment plan verification. A project was undertaken to first develop and extend tools used to process the output from the group's detector prototype as recorded by a Teledyne LeCroy HDO6104 Oscilloscope, and then to employ these tools in making measurements of beam energy in conjunction with positional measurements made using a tracker module developed by PRaVDA. The aim was to provide proof of principle for the use of the calorimeter alongside the tracker module in reconstructing dose depositions from therapeutic beams. The data processing tools were shown to recreate the group's previous results using other readout hardware, and results suggest that the use of the LeCroy oscilloscope for readout may improve the resolution of the detector. unfortunatley, proof of concept for the concurrent use of the tracker and calorimeter for measurements reconstructing dose distributions was not achieved due to a failure to identify paired triggering events across the two detectors. Furthermore, the PRaVDA tracker module appeared to have unexpected effects on the distribution of energies in the beam.

Contents

1 Introduction

1.1 Cancer and Cancer Treatment

Cancer is the name given to a class of diseases caused by unregulated and abnormal cell division, resulting in the formation of malignant tumours and the destruction of surrounding healthy tissues [1]. A variety of treatment modalities exist, each with strengths and weaknesses with regards to the nature of different cancers. Modes of treatment include:

- The direct surgical removal of tumours in excisional therapy, which is highly effective but only possible when tumours are localised and accessible to surgery.
- The introduction of cycotoxic agents in chemotherapy, which is effective for treating cancers that are distributed widely through the patient but can lead to side effects. [2]
- The targeted irradiation of tumours in radiotherapy, which allows for non-invasive treatment of cancers with minimal side-effects but requires that radiation doses can be sufficiently well targeted and controlled.

Often, multiple modes of treatment are used in conjunction.

1.2 Radiotherapy

The aim of radiotherapy is to initiate apoptosis and thereby cause cell death in cancerous cells, while minimising damage to healthy tissues. This is achieved by the disruption of the DNA of cancerous cells by ionising radiation. In the case of external beam radiotherapies, a class of radiotherapy including x-ray radiotherapy and proton beam therapy, the source of this ionising radiation is a beam of energetic particles directed towards a patient's tumour from apparatus outside of their body. External beam radiotherapy procedures are non-invasive, meaning that they can be employed

to treat tumours which are inoperable. The effects of treatment by external beam radiotherapy are also much more localised than in typical chemotherapy procedures, leading to reduced side effects for the patient. In order for the beam to irradiate the tumour, however, it must pass through healthy tissues which will consequently recieve some proportion of the dose delivered to the region targeted for treatment. This dose of ionising radiation to healthy tissues can lead to damage or even induce carcinogenesis. The central problem of radiotherapy then becomes that of how to sufficiently irradiate the tumour while sparing surrounding tissues, especially where those tissues are sensitive to radiation damage.

1.2.1 X-Ray Radiotherapy

X-ray radiotherapy seeks to deliver ionising radiation to tumours using a beam of x-ray photons. Photons have no electric charge and therefore do not directly ionise matter. Instead, the x-ray beam deposits energy by giving rise to free, energetic, charged particles within the patient. This occurs by the following three mechanisms:

1. The recoil of essentially free electrons in the patient's tissues from Compton scattering

2. The liberation of bound electrons via the photoelectric effect

3. Pair production in the electric field of an atomic nucleus

These particles go on to interact Coulombically with electrons and nuclei, resulting in a path of ionisation along the trajectory of the beam. [3]

The photon-matter interactions occur with a probability proportional to the number of photons present and dependent on the properties of the material through which the beam passes. This gives rise to an exponential attenuation of the beam with depth in the patient, given in Eq. 1.1.

$$I(x) = I_0 e^{-\mu(h\nu, Z)x} \tag{1.1}$$

where $I(x)$ is the beam intensity at a depth x, I_0 is some initial intensity, and $\mu(h\nu, Z)$ is the linear attenuation coefficient, which describes the interaction probability per unit length for a photon traversing a given medium. There is a corresponding exponential decay in the dose of ionising radiation delievered with depth in the patient, after a short build-up period over which the charged particles are initially released. This behaviour is illustrated by the dashed curve in Fig. 1.

A consequence of this exponential decay in dose is that tissues between the x-ray source and the tumour will always be more heavily irradiated than the target tumour for treatment delivered along a single beam axis, and the dose delivered to healthy tissues must increase in order to treat tumours at greater depths. There will also be a non-negligible dose deposition beyond the tumour, meaning that doses delivered along an axis connecting the tumour to critical or highly radiosensitive tissues, for instance in the brain or spine, carry a risk of side effects and long term health issues for the patient after treatment.

1.2.2 Proton Radiotherapy

The very different dose-deposition mechanisms between protons and photons make therapeutic proton beams a promising technology for the safe and effective delivery of radiotherapy, especially when tumours lie close to critical or radiosensitive tissues. Coulombic interactions between protons and electrons cause protons deposit energy in a medium according to the Bethe equation [4]:

$$-\left\langle \frac{dE}{dx} \right\rangle = kz^2 \frac{Z}{A} \frac{1}{\beta^2} \left[\frac{1}{2} \ln \left(\frac{2m_e c^2 \beta^2 \gamma^2 T_{max}}{I^2} \right) - \beta^2 - \frac{\delta(\beta\gamma)}{2} \right] \tag{1.2}$$

where Z and A are the atomic and mass numbers of the medium, z is the charge of the incident charged particle, c, β, γ, and m_e take their usual meanings, T_{max}

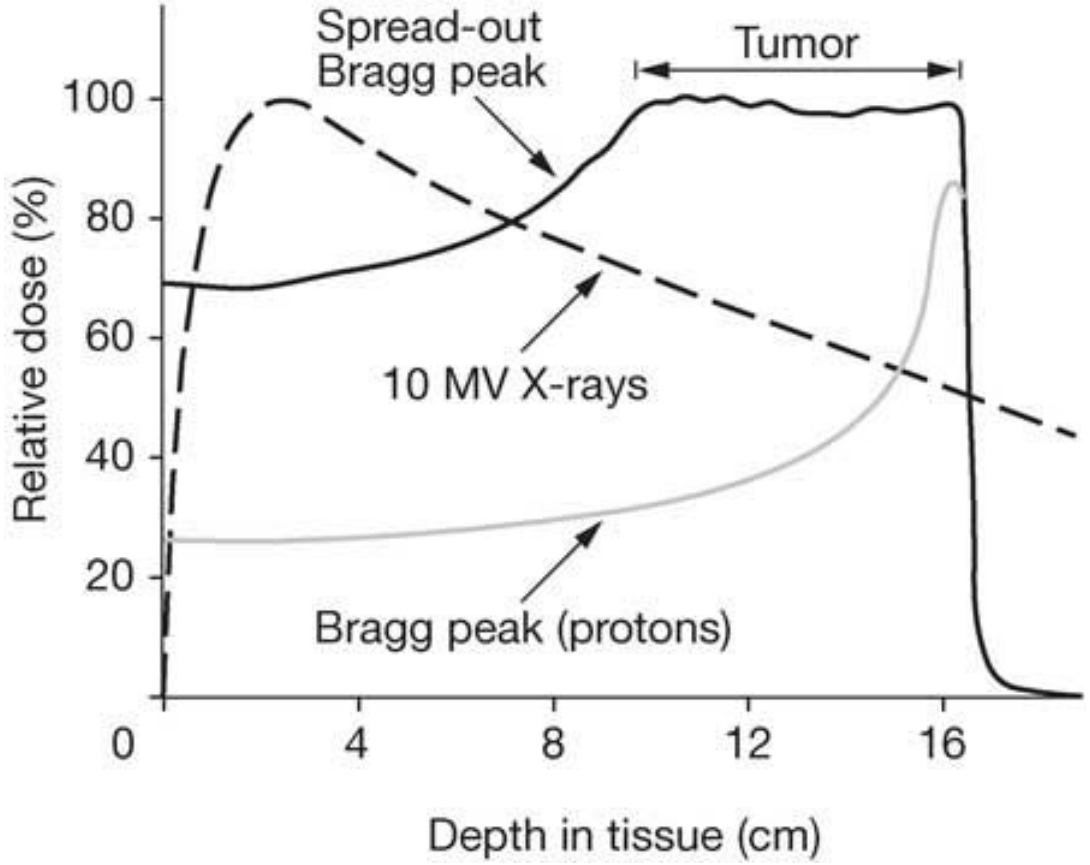

Figure 1: Examples of dose deposition profiles for x-rays (dashed curve), single protons (grey curve), and superposed proton Bragg peaks (solid black curve). [6]

is the maximum possible kinetic energy transfer from the particle to an electron in the medium in a collision, I is the mean ionisation potential, $k = 4\pi N_A r_e^2 m_e c^2$, and $\delta(\beta\gamma)$ is a density correction term. A key feature of the Bethe equation is the proportionality of the rate of energy loss $-\left\langle \frac{dE}{dx} \right\rangle$ to the inverse square of the particle velocity $\frac{1}{\beta^2}$. This proportionality means that at high energies protons lose energy to the medium they travel through at a low rate. As they slow, this rate sharply increases, until the proton comes to a sudden stop. This behaviour gives rise to the "Bragg curve" [5] dose deposition profile shown in grey in Fig. 1.

The proton dose deposition profile has a number of features that make it very attractive for radiotherapy, in particular the very low entry dose, the finite range, and the very high, very localised dose known as a "Bragg peak" at the end of the

proton's path. The low entry dose helps to spare healthy tissues between a proton beam source and a tumour, while the energy of the beam can be selected (given sufficient knowledge of the density and composition of the patient's tissues) to place the highly localised peak energy deposition inside the tumour. The finite range means that no dose will be given to critical or radiosensitive tissues provided they lie beyond the Bragg peak. [7] Many Bragg peaks can be superposed to create a region of near-uniform dose, a "spread-out Bragg peak" [8], to cover the entirety of a patient's tumour while maintaining the advantages of the Bragg curve.

It should be noted that while the primary mode of interaction for theraputic protons is the electronic interaction described by the Bethe equation and Bragg curve, there are other interactions of interest in the study of proton radiotherapy. In particular:

- Coulombic interactions between theraputic protons and atomic nuclei, which cause the protons to scatter, resulting in a broading of the beam described by the theory of multiple Coulomb scattering. [9]

- High energy protons can undergo inelastic collisions with atomic nuclei, resulting in high energy ions, secondary protons and neutrons as products. Charged products typically only travel a short distance, creating a small but non-negligible spread in the deposition from the beam, but neutrons are very highly penetrating and may present a safety hazard for patients and medical staff. [10]

These interactions are important to consider in a clinical setting, but are but are largely outside of the focus of this project.

1.3 Difficulties Faced by Proton Radiotherapy

The key assumption on which the success of proton beam therapy relies is that the peak dose deposition from a proton beam can be reliably made to coincide with a patient's tumour. If the energy of the beam is incorrectly calibrated for the treatment

of the patient, some region of healthy tissue will be irradiated with the maximum dose of the beam, while some region of the target tumour may go untreated. This not only fails to adequately treat a patient's cancer, but may additionally do considerable harm through the irradiation of the healthy tissue. This is especially true in cases where proton beam therapy has been selected specifically for use near structures too vulnerable for other modes of caner treatment or in patients such as children who are particularly susceptible to long-term harm from treatment side-effects. [11] Unfortunately, the technical complexity of treatment hardware for proton beam therapy and the methodologies by which treatment plans are created[1] challenge this assumption. This necessitates that treatment plans are verified as part of quality assurance procedures before they can be delivered to a patient.

1.4 Treatment Plan Verification

A patient's treatment plan is the sequence of beams which has been determined optimal for the treatment of their tumour. Treatment plan verification is the process by which a patient's treatment plan is first delivered to detector hardware. In the particular case of proton beam therapy, the goal of treatment plan verification is to make certain that the Bragg peak for each beam will be delivered at the intended depth and energy by the execution of the treatment plan. This is often accomplished by delivering the treatment plan to a water tank dosimeter, water phantom, or a detector consisting of parallel ionisation chambers separated by layers of a water-equivalent material. Typically, treatment plan verification procedures using these devices are slow, wasting time which could be used to treat patients. It is proposed that a sufficiently small, sufficiently fast scintillation-based calorimeter could be mounted to the nozzle of a medical proton beam, and make verification measurements of the

[1] Proton beam therapy treatment plans are currently based on the use of x-ray tomography to map the density of patient's tissues. The very different interaction mechanisms for photons and protons with matter give rise to uncertainties in the region of 1 - 3 mm in the location of the Bragg peak when x-ray derived density maps are used to design treatment plans using protons. [12]

beam output without the slow setup proceedures associated with current techniques for treatment plan verification. Coupled with a tracker, such a detector could yield rich information about the distribution of the dose from the treatment beam.

1.5 Goals for the Project

The UCL Proton Beam Therapy Group is developing a calorimeter of this type by leveraging in-house expertise from the design of the detector for the SuperNEMO experiment. In this project, the aims were to:

- Develop tools for data processing and analysis of the output of the prototype calorimeter recorded by a Teledyne LeCroy HDO6104 Oscilloscope.
- Provide proof of principle for the use of the UCL single-module calorimeter prototype in conjunction with a tracker module developed by PRaVDA in fast treatment plan verification.

1.5.1 The UCL Single-Module Calorimeter Prototype

The prototype calorimeter that is the focus of this project is based on R&D for calorimeter modules for SuperNEMO. The SuperNEMO experiment is searching for evidence of neutrinoless double-β decays, and the detector required exacting standards of its calorimetry in terms of both energy and time resolution in order to identify low-energy ($\mathcal{O}(1$ MeV)) β emission and account for a γ-ray background, but cost efficiency and hardware longevity were also considered. [13]. These are all highly desirable properties for the intended medical-use calorimeter.

The UCL single-module calorimeter consists of a single 3 cm $\times$ 3 cm $\times$ 5 cm block of plastic scintillator, coupled head-on to a 2" Hamamatsu R13089-100 11 photomultiplier tube (PMT) by an optical gel. The PMT is powered by a Caen NDT1470 HV Supply, and the group has a variety of options with regards to readout electronics. This includes the aforementioned LeCroy oscilloscope, and also a digitiser and ADC manufactured by Caen.

1.5.2 The PRaVDA Tracker

The primary aim of the PRaVDA consortium is to develop tracking and calorimetry detectors for proton tomography. PRaVDA have demonstrated solid state tracker modules capable of simultaneously locating multiple therapeutic protons within a 2D plane at beam rates of 2.5 MHz to a high precision, and reconstructing the paths of protons between pairs of detector modules.[14, 15] This is achieved by the use of three layers of silicon strip detectors in each module, offset 60° from one another to provide an $x-u-v$ coordinate system in which to identify hits. While the intended use of the tracker modules is in proton tomography and the construction of better treatment plans for proton beam therapy, it is believed that they could be used in conjunction with a calorimeter such as the UCL prototype to reconstruct dose depositions in 3D.

2 Developing Analysis Tools

The first part of the project was to extend the work of a previous Master's student from the group, and develop tools to access, manipulate, and analyse data collected with a Teledyne LeCroy HDO6104 oscilloscope, which has been used to read out the electronics from the single-module calorimeter, and may be used for the readout of future detectors. The preferred mode of operation is for the scope to write a file in a binary format for every 1000 acquisitions in "sequence mode", in which data is recorded for a fixed time interval every time a trigger condition is met. This is to minimise the scope downtime incurred by writing the files, and has the added benefits of saving file storage space and being fast to manipulate. The previous student had used a Python script based on the Teledyne LeCroy specification template to first convert the binary files to ASCII format, which would then be read and processed by a macro written for the ROOT data analysis software. While effective, this approach was slow and failed to access and utilise some valuable data recorded by the scope, particularly timing information related to the acquisition triggers.

2.1 Goals for the Analysis Tools

The task was then to develop code which would enable access to all data contained in the binary files, and which could provide analysis of the contents quickly by avoiding the conversion to an intermediate ASCII format. Over the course of development, the following were decided to be desirable features for the code:

- Rapid, direct access to data contained in LeCroy binary .trc files, including timing information for acquisition triggers
- Easy interface to the ROOT data analysis package
- Fast construction of ADC spectra
- Identification and mitigation of pileup in recorded waveforms

2.2 The LeCroyData Class

Initially, code was written in Java to access information in LeCroy binary format .trc files and print it to ASCII format text files. The purpose of this exercise was to become familiar with the format of the files and identify the previously overlooked information stored in them. Java was chosen for speed and familiarity, but was inappropriate for integration with ROOT, which is primarily designed for use as a set of C++ libraries or as an interpreted C++ scripting tool.

The code responsible for reading the binary data was consequently re-written as a C++ class named LeCroyData, which implemented methods for access to the file contents. This was to enable the use of the class as an #include alongside ROOT libraries in compiled C++ code, to allow the ROOT interpreter to parse and use the class in interactive sessions, and most importantly to allow for manipulation of data from the binary files without the slow process of writing to and reading from ASCII format files.

A command-line application was written using this class to display the contents a binary file header to the screen, with the option of additionally displaying the recorded times of acquisition triggers. An example of the use of this application is shown in Fig. 2.

With access to the binary data and straightforward ROOT compatibility achieved, development moved towards expanding the code to handle the generation of ADC spectra.

2.2.1 Approaches to Calculating ADC Spectra

Given reasonable linearity of the light yield of the scintillator, and provided the PMT does not saturate, the integral of a signal pulse over its duration (a quantity known as ADC counts) is a measurement of the energy deposited in the detector.[16] The energies deposited in the detector are expected to follow a Landau distribution [17],

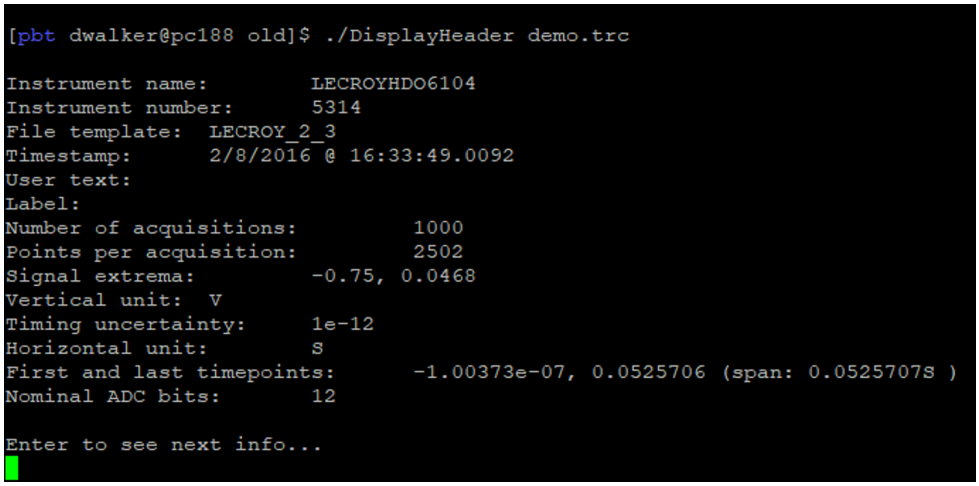

Figure 2: A command-line application for displaying information about a LeCroy binary file to screen.

with the distribution of ADC counts recorded by the detector taking the form of a Landau-Gaussian convolution, with the standard deviation of the Gaussian part corresponding to the finite resolution of the detector. The construction of ADC spectra from waveforms recorded from the calorimeter involves, for each waveform recorded:

1. Identifying the baseline of that waveform, which is the value recorded from the PMT in the absence of a proton signal. This is not typically zero on account of PMT dark current and noise.

2. Performing "baseline subtraction", in which the value of the waveform at each point is subtracted from the baseline. Examples of waveforms before and after baseline subtraction are shown in Fig. 3.

3. Defining a region of interest, to be considered the signal, within the waveform

4. Integrating over the signal to yield a measure of the energy deposited in the detector by the corresponding event.

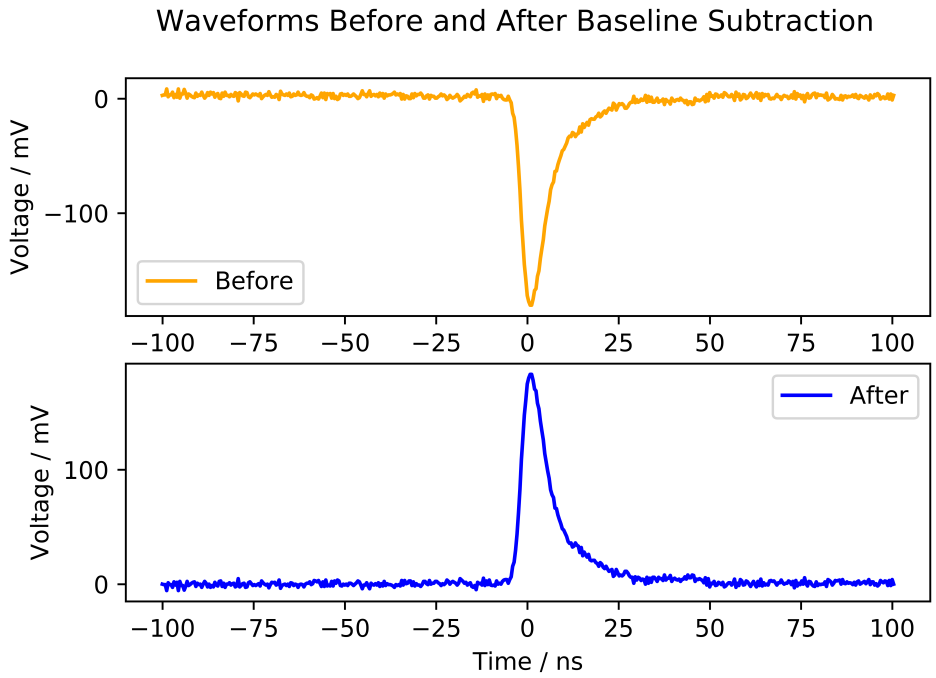

Figure 3: A waveform from data collected on the 21st of February, shown before and after baseline subtraction.

Fitting

The first method implemented for generating ADC spectra was based closely on a method previously used by the group for processing waveforms recorded from the detector by a Caen digitizer, and relied heavily on the technology available through ROOT to fit functions to data.

In the group's original method, the signal is initially approximated as a Gaussian function, and ROOT routines are called to fit a function of the form

$$f(t;\alpha,\beta,\gamma,\delta) = \alpha \exp\left[-\left(\frac{t-\beta}{\gamma}\right)^2\right] + \delta \tag{2.1}$$

to the raw signal waveform. The constant term δ is then extracted and taken to be the baseline of the waveform, and baseline subtraction is performed.

A second fit is then performed on the waveform after baseline subtraction. In

the group's original method, this is a piecewise function of the form

$$f(t;\vec{p}) = \begin{cases} a_1 \exp\left[-\left(\frac{t-b_1}{c_1}\right)^2 \right] & \text{in the rise time of the signal} \\ a_2 \exp\left[-\left(\frac{t-b_2}{c_2}\right)^2 \right] & \text{at the peak of the signal} \\ a_3 \exp\left[-b_3 t \right] & \text{for the decay of the signal} \end{cases} \tag{2.2}$$

where $\vec{p}$ is a list of 11 parameters containing all $a_i,\ b_i,\ c_i$, and also crossover conditions for each part of the piecewise function. The χ^2 statistic and number of degrees of freedom (NDF) for this fit are extracted and used to provide a measure of pileup in the waveform, since a waveform containing multiple signals is typically fit poorly by the piecewise function. Integration is then carried out on the waveforms, a threshold value of χ^2/NDF is chosen by inspection of the distribution of its value across all waveforms, and all waveforms with a χ^2/NDF less than this value have their integrals included in the spectrum.

The implementation of this approach written for use with the LeCroyData class was very similar, with the key differences being:

- The trialling of different functions for fitting the waveform after baseline subtraction
- The use of ROOT's TF1::Integral() method on the fitting function to calculate the integral over waveforms

Functions trialled for use in the fit included the Landau distribution with additional scaling factor, a piecewise function approximating the rise of a signal as a Gaussian and the decay of the signal as the sum of two exponential functions, and a function approximating the entire signal as the difference of two exponential functions. These are given as Eq. 2.3, Eq. 2.4, and Eq. 2.5 respectively. It should be noted that while Eq. 2.3 uses the exact form of the Landau distribution [17], the actual

implementation used ROOT's TMath::Landau() routine, which provides a high-order polynomial approximation.

$$f_1(t;\mu,\sigma,\alpha) = \frac{\alpha}{\pi}\int_0^\infty e^{-s}\cos\left(s\Big(\frac{t-\mu}{\sigma}\Big) + \frac{2s}{\pi}\log\Big(\frac{s}{\sigma}\Big)\right)ds \tag{2.3}$$

$$f_2(t;\mu,\sigma,A,B,C,\gamma_1,\gamma_2,t_0) = \begin{cases} A\exp\Big[-\big(\frac{t-\mu}{\sigma}\big)^2\Big] & \text{for } t \le t_0 \\ Be^{-\gamma_1 t} + Ce^{-\gamma_2 t} & \text{for } t > t_0 \end{cases} \tag{2.4}$$

$$f(t;J,\alpha,\beta,t_0) = \begin{cases} J(e^{-\alpha(t-t_0)} - e^{-\beta(t-t_0)}) & \text{for } t \ge t_0 \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

Applying this method with Eq 2.3 as the fitting function and a threshold χ^2/NDF of 13 yielded a spectrum qualitatively similar to the group's earlier results, but resulted in 189 of the 1000 waveforms in the processed binary file being disregarded. More importantly, the result of waveform integration was not strictly representative of the energy deposited in the detector as the integration was carried out on the fitting function and not the signal. Furthermore, the processing of a single binary file of 1000 waveforms took approximately 5-10 minutes, implying processing times of hours for a full data collection run of many tens of files. These critical issues were addressed by implementing a method for integration directly on the waveform, and by seeking more time-efficient approaches to identifying signals and waveform baselines.

Investigation of Methods for Numerical Integration

The first issue to resolve was the inappropriate approach to integrating over the signal. Familiarity with numerical methods such as the trapezium rule led to an investigation of other Newton-Cotes formulae: formulae for numerical integration which rely on the evaluation of the integrand at fixed intervals over the domain of

the integration. These formulae take the form

$$\int_a^b f(x)dx \approx \sum_{n=0}^{N} w_n f(a+nh) \tag{2.6}$$

where $h = \frac{b-a}{N}$ and w_n are weighting factors. They are an attractive option for integrating over signals recorded by an oscilloscope at a fixed sampling rate, where values of the integrand are only known at fixed intervals.

Three Newton-Cotes methods were trialled for use in the generation of ADC spectra from LeCroy oscilloscope output, and were compared to the method implemented by the previous student. The methods trialled were the five-point rule known as Boole's rule, the trapezium rule, and a "composite" rule formed by comparing terms in Simpson's three-point and 3/8 rules. [18] For the trials, each method was implemented in Python and used to integrate Eq. 2.5; a function chosen to emulate signals from the detector after baseline subtraction.

An example of this function fit to a detector signal is shown in Fig. 4, which demonstrates the appropriateness of the use of the function to test the performance of integration methods for actual detector signals. It is important that the chosen function should be possible to integrate analytically in order to properly evaluate the error introduced by the approximate integral (at least to within machine precision).

In each case, the limits of the integral and the values of the function parameters were kept fixed and the number of points at which the function was evaluated in the approximate integral was increased, and the error computed. The results are shown in Fig. 5.

While for very few sample points Boole's rule gives the smallest fractional error, the error incurred by the composite method rapidly falls below that of every other method presented as the number of points sampled increases. It is surprising that Boole's rule, a fifth-order method, is quickly outperformed by all other methods trialled and in fact performs worse at higher numbers of sampling points than when

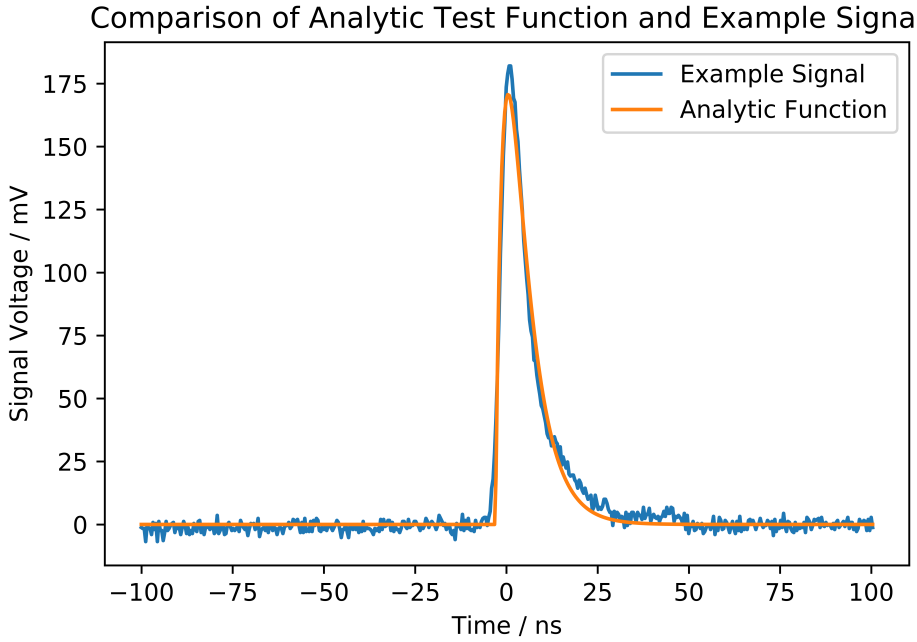

Figure 4: Eq. 2.5 fit to a detector signal using the curve_fit() method from Python's SciPy module. The function and signal share similar peaks and curvature.

given fewer sampling points. This is likely due to an error in the implementation, but the source of this error could not be identified.

The composite rule integration method was selected as the best of the trialled methods, and an implementation was written in C++ for use in all subsequent code used to generate ADC spectra.

Waveform Median Baseline, Schmitt Trigger Signal Identification

In order to speed up processing, approaches to calculating waveform baselines and identifying signals were sought that had no reliance on ROOT's function fitting routines.

In the first attempt at developing an approach under this restriction, the median voltage recorded across the waveform was used as an estimate for the waveform baseline. This was motivated by the assumption that the "typical" value of a waveform was a value from the baseline behaviour, given that the acquisition window is

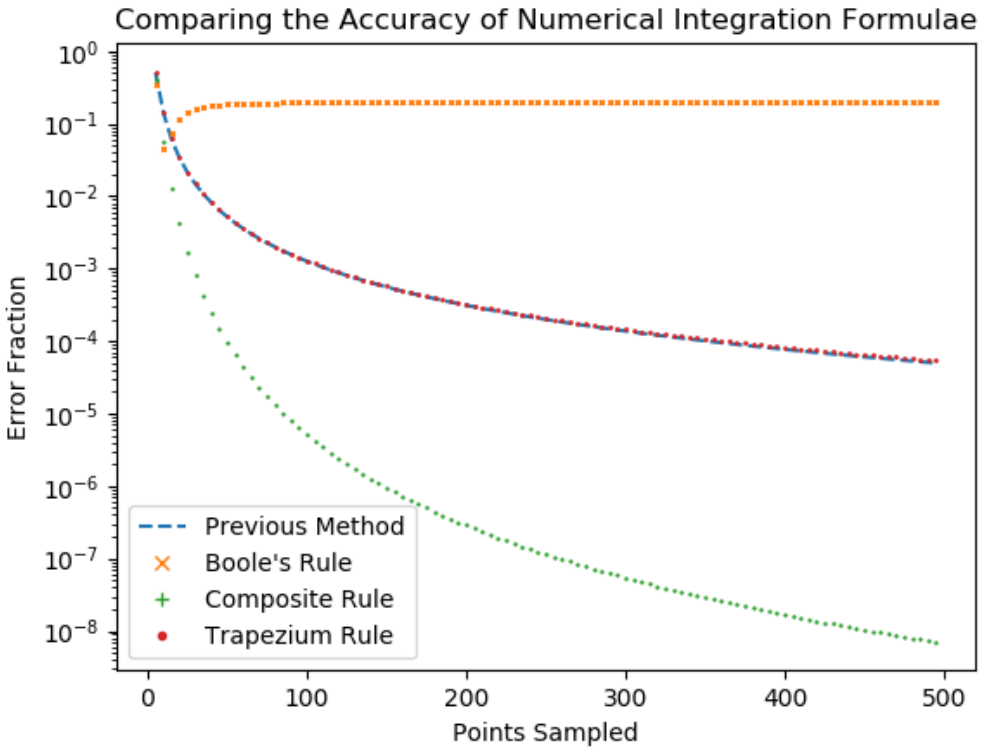

Figure 5: Comparing the accuracy of numerical integration formulae. The error fraction is computed as $\frac{|\int_a^b f(x)dx - F(x,a,b)|}{\int_a^b f(x)dx}$ where $F(x, a, b)$ is the result of the numerical integration method on $f(x)$ between a and b.

much wider than the width of a signal pulse while the mean average would be skewed by the presence of the signal, and the mode average rendered unreliable by noise.

To avoid discarding data where possible, a method was sought to select and integrate over signals within a waveform while ignoring pileup. One attempt looked to identify signals using the structure of a typical signal pulse. With function fitting techniques too computationally expensive, the logic of a Schmitt trigger [19] was implemented to select signals based on their rising and falling edges. By this method, the user was able to specify a rising edge threshold and a falling edge threshold. When the waveform voltage rises above the rising edge threshold, the waveform voltage and corresponding time point are recorded and until the waveform voltage falls below the falling edge threshold all voltage values are appended to an array of "signal" voltages.

Integration is carried out on this array of voltages, yielding a measure of the energy deposited by a particle detected at the recorded time point. A graphic representing the selection of a signal pulse by Schmitt trigger logic is given in Fig. 6.

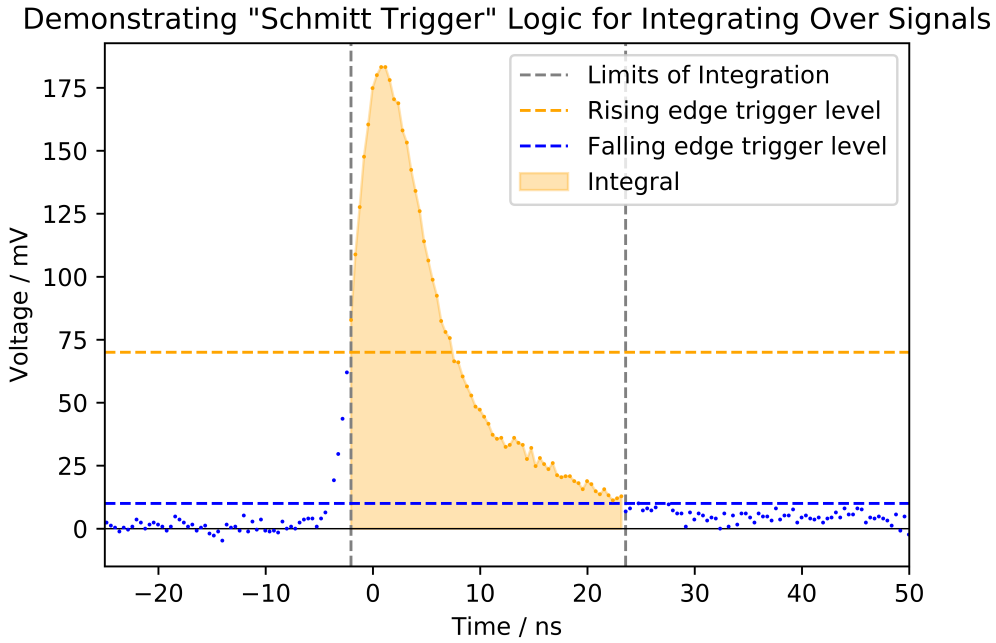

Figure 6: A demonstration of Schmitt trigger logic for the definition of a signal pulse within a waveform. Orange points show the data points selected as part of the signal, with the area underneath showing the corresponding ADC counts.

The median baseline and Schmitt trigger methods were applied to data taken by the group using the LeCroy oscilloscope on the 2nd of August 2016 at the Clatterbridge Cancer Centre, for which equivalent runs had been taken with the Caen ADC. A spectrum was generated from a single file binary file containing 1000 waveforms. There was a significant speedup using the new methods, with the time taken to produce a spectrum reduced to something in the region of half a second. The resultant spectrum was qualitatively similar to the expected Landau-Gaussian convolution, but a direct side-to-side comparison with the equivalent Caen ADC run demonstrates that this approach is not compatible with the group's previous work, as it underestimates the position of the peak in ADC counts. This is shown in Fig. 7.

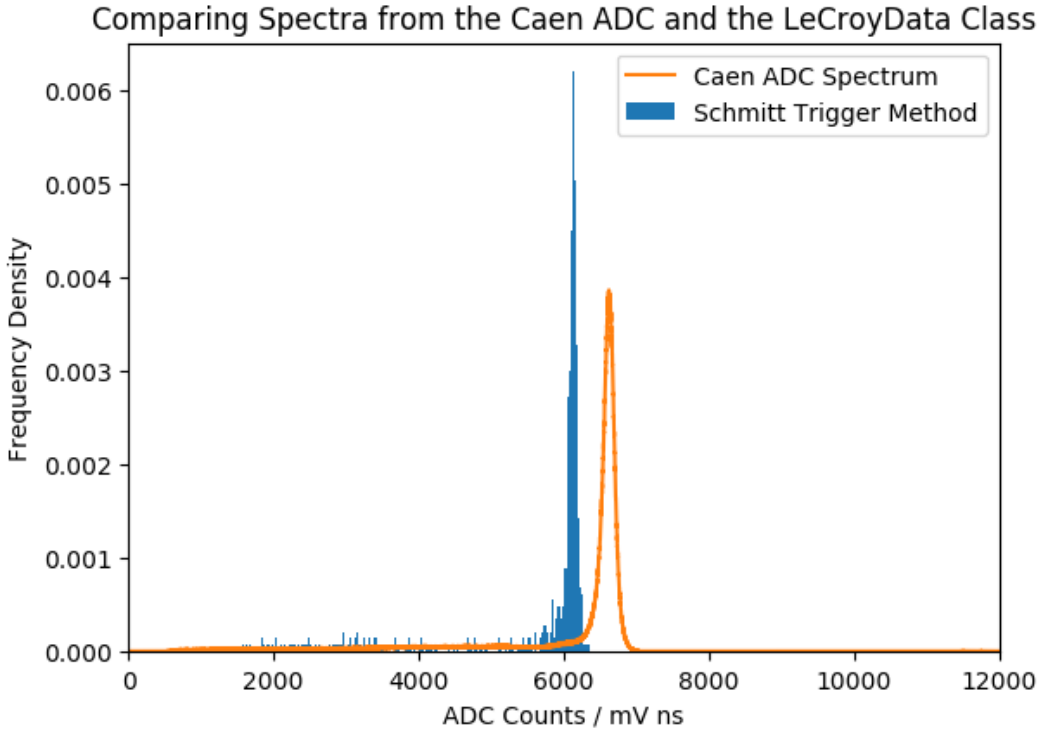

Figure 7: Results of the calculation of ADC spectra from data collected by the group in August 2016 by use of the Schmitt trigger logic for signal selection and the waveform median to estimate the baseline. The distribution takes the expected form, but the position of the peak in ADC counts in underestimated by comparison to the group's results at the time.

There are a number of weaknesses to the methods described here. Approximating the waveform baseline with the median value of the waveform relies on the low density of signal pulses in the waveform and will fail to yield a value representative of the baseline behaviour in cases of heavy pileup. Even in cases where the assumption of low density of signals in a waveform is valid, the median will only yield one value sampled from the baseline behaviour or an average of two such values, where the desired value for the baseline estimate would account for all values of the baseline behaviour of the waveform. Though an attractive statistical trick that can be evaluated "mindlessly" across a waveform, the median is an inappropriate estimator of the baseline. The Schmitt trigger logic for detecting signals in the waveform was

found to be disrupted by noise in the baseline, and for reliable use required that the rising and falling thresholds were set significanlty above the level of the noise. This left the early rises and long tails of signal pulses inaccessible, which was likely a key contributor to the underestimation of the peak ADC counts by comparison to the Caen spectrum. Furthermore, the Schmitt trigger logic is only able to distinguish between pulses which are sufficiently separated in time, as the long tail of one signal pulse may not fall below the falling threshold before the rising edge of the next pulse. This effect is illustrated in Fig. 8.

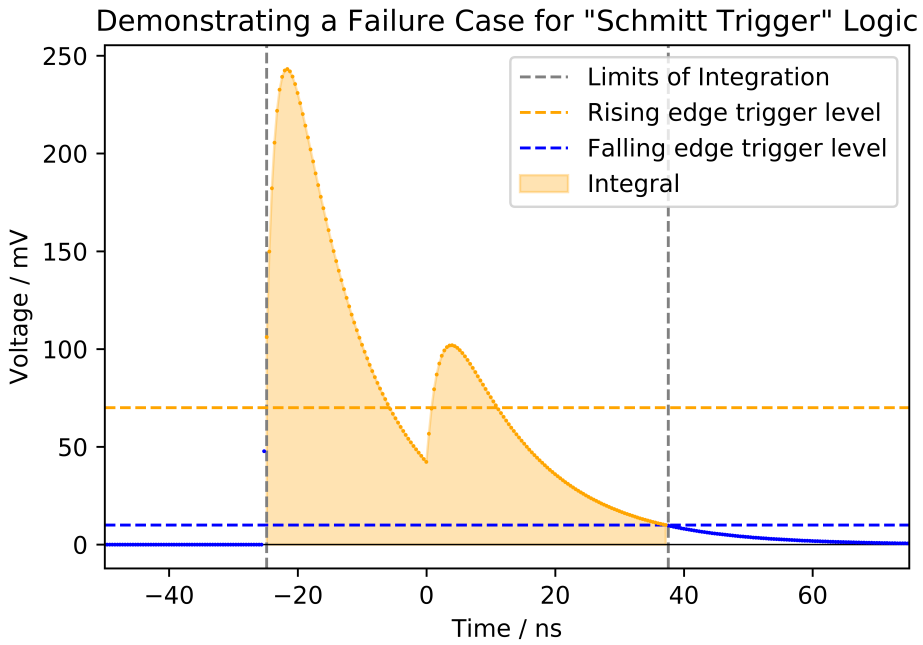

Figure 8: An example of a failure condition for the Schmitt trigger logic for signal identification. This example has been constructed from the sum of two signal-like functions as given in Eq. 2.5, but this behaviour was observed in actual test datasets.

Waveform Median Baseline, Rising Edge and Fixed Window Signal Identification

Developed concurrently with the Schmitt trigger method was a method for selecting signals within a waveform more closely based on the charge integration procedure carried out by the Caen ADC. Coming from the same stage in development, it sim-

ilarly was implemented alongside the flawed baseline subtraction routine based on the median value of the waveform. In this method, the rising edge trigger from the Schmitt method was used to determine a starting point for a window of integration which spanned a fixed interval in time. Similarly to the Schmitt trigger method, the user was free to specify the rising edge trigger level and the width of the integration window. The results of applying this method to a full run of 51000 waveforms is shown in Fig. 9 alongside the equivalent Caen ADC run. The processing time for this method, per 1000 waveform binary file, was not perceptably different from that for the Schmitt trigger method within an interactive ROOT session.

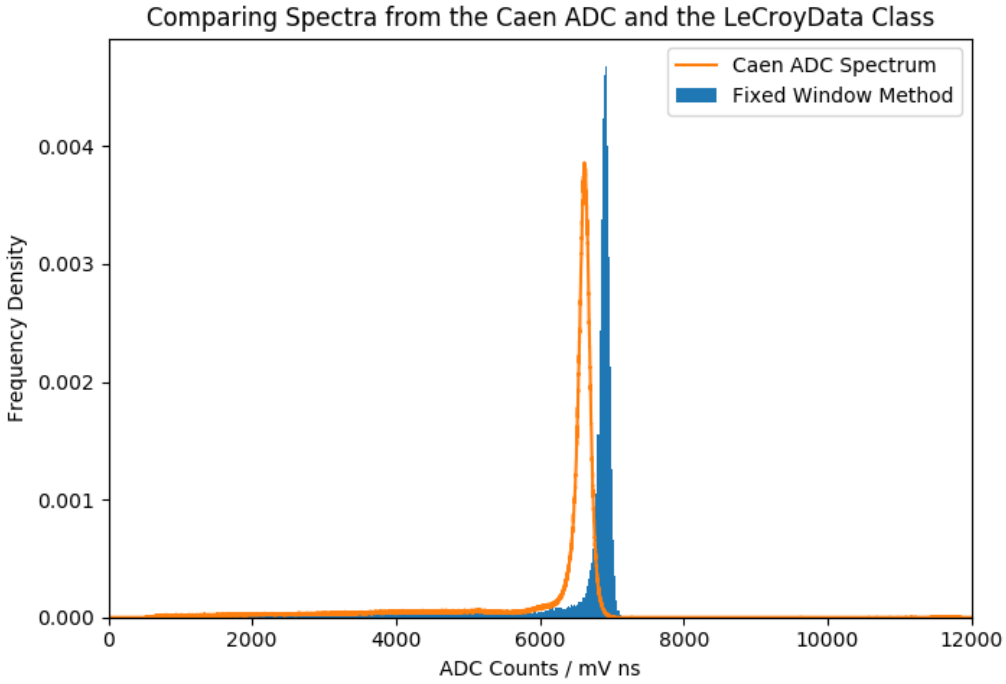

Figure 9: The spectrum generated by estimation of the baseline as the median of the waveform, with the signal defined as a fixed time interval after a user-set rising edge trigger.

It was again observed that the spectrum took qualitatively the same form as the Caen spectrum, with a promisingly narrow peak. However, this method was shown to overestimate the peak ADC counts by comparison to the Caen result. This may

have been in part due to the inappropriate baseline subtraction strategy, but could also have arisen from differences in how the fixed integration window was defined for the Caen and LeCroyData procedures. Specifically, the Caen integration window has an offset parameter that can be used to determine a point before the trigger at which the integration should start. This feature was absent in this implementation of the LeCroyData method.

Pre-Trigger Mean Baseline, Full Waveform Integration

To better estimate the value of the baseline, a method was devised to identify a region of the waveform as baseline and evaluate the mean value over just this region. This differs from both the fitting and median approaches implemented above in that these aim to estimate the baseline by an operation on the entire waveform. Initially, the region considered as baseline was taken to be all time points $t < 0$ with the zero point in time for a given waveform corresponding to the time of the trigger for that waveform's acquisition by the LeCroy oscilloscope. The mean and standard deviation of this region were calculated, with the mean taken as the value of the baseline for the waveform and the standard deviation used to test for the presence of unexpected structure such as signal pulses in the pre-trigger region. Here, the standard deviation was compared to a threshold level which could be set by the user and if the threshold was exceeded the waveform was discarded. This method was initially tested without a corresponding method for selecting signals within a waveform, with integration taking place across the entire waveform. This yielded a spectrum with a much broader peak than the Caen ADC, with a much higher peak value of ADC counts. More importantly, however, it also yielded a spectrum containing negative values of ADC counts, which was recognised as being distinctly unphysical and necessarily a consequence of the method for baseline estimation and subtraction. Closer inspection revealed that a part of the leading edge of the signal pulses occurred prior to the time point $t = 0$, and this very small structure was

heavily skewing the value of the mean whilst passing under default values for the standard deviation threshold. The baseline region was refined to be the first 90% of points in the region prior to time $t = 0$ and the default value of the standard deviation reduced, resolving the issue.

Pre-Trigger Mean Baseline, Caen-Like Window

With a more valid approach to baseline estimation established, another attempt was made to recreate the group's previous results with the Caen ADC. Here, the aim was to as accurately as possible mimic the procedure by which the Caen ADC performs integration. As described above, the Caen ADC integrates a signal from an offset before a signal trigger for a fixed period of time. Here, the trigger used was the trigger time defined by the LeCroy oscilloscope when the waveform was recorded and the offset and integral width were parameters to be set by the user. The width used for the Caen ADC integration was 150 ns and this was made the default value of the width parameter for the LeCroyData integration. The offset parameter was varied to see how the location of the window of integration affected the resulting spectrum. The closest recreation of the Caen ADC spectrum was achieved with the window of integration set to be symmetric about the trigger time. The results of processing the full run of 51000 waveforms and a comparison to the target Caen ADC spectrum are shown in Fig. 10 and with a logarithmic frequency density axis for improved detail in Fig. 11.

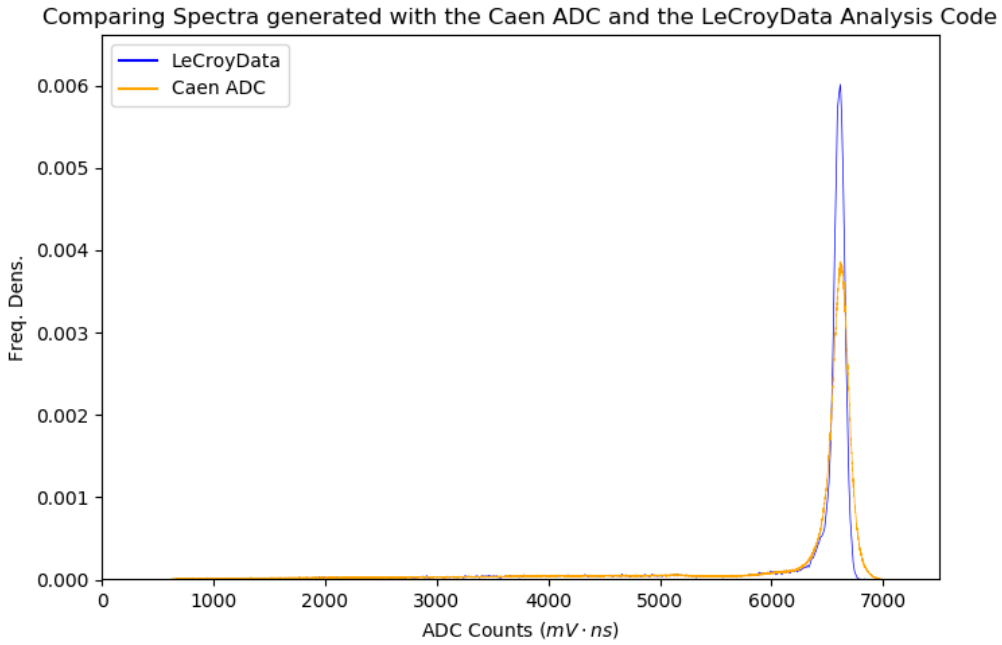

Figure 10: Demonstration of the LeCroyData analysis code recreating the group's previous results using the Caen ADC.

For both the LeCroyData and Caen ADC spectra, a Gaussian function was fit to the peak, again using the SciPy curve_fit() method, and its mean extracted as a measure of the peak position. By this measure, the LeCroyData spectrum peak lies within 0.003% of the value of the peak of the Caen ADC spectrum. In Fig 11, it can be seen that the Caen spectrum is closely tracked by the LeCroyData spectrum across almost its entire domain, with the exceptions being at the peak where the LeCroyData spectrum's peak is narrower, and at ADC counts fewer than approximately 2000, where the Caen spectrum continues to ADC counts fewer than 1000 while none are recorded in the LeCroyData spectrum. The narrower peak implies an improved resolution in the measurement of detector ADC counts and consequently proton energy, while the shorter tail implies that the LeCroy oscilloscope's trigger level was set too high to register these lower-energy protons.

The code for the LeCroyData class, including minor later modifications required by the project and described in section 3, are included in appendix 4.1.

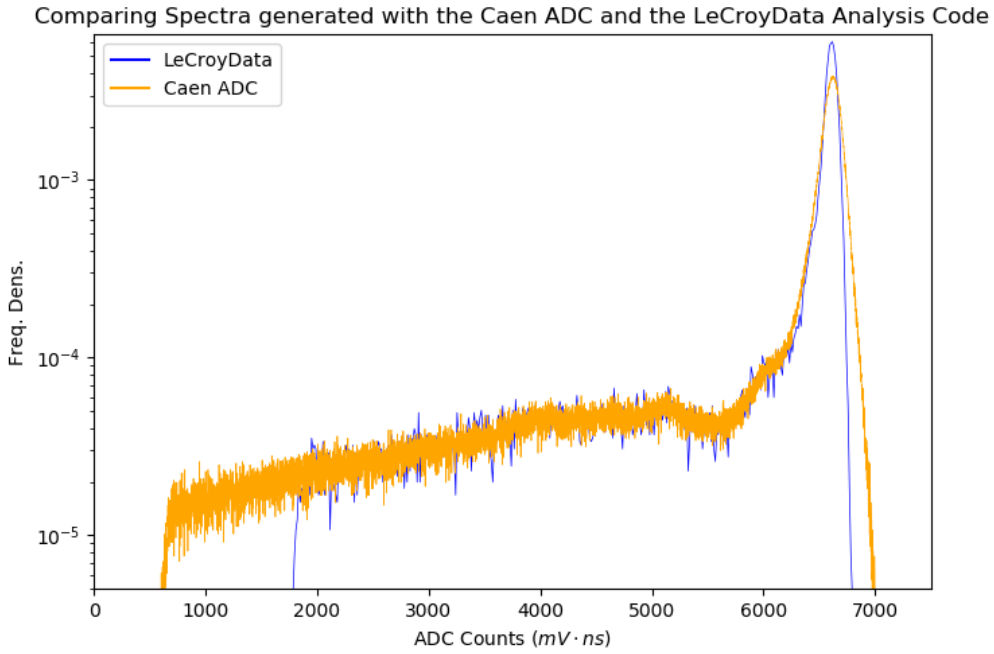

Figure 11: Fig 10 redrawn with frequency density on a lograithmic scale. Features such as the different minimum detected ADC counts and unusual kink in both spectra in the region from approx. 5000 to approx. 6000 ADC counts are more clearly visible.

3 Data Collection with a PRaVDA Tracker Module

On the 21st of February, the UCL Single-Module Calorimeter and LeCroy HDO6104 were taken to the University of Birmingham for a test beam, in which calorimeter data was collected concurrently with tracking data from one PRaVDA tracker module.

3.1 Goals for the Experiment

The goal of the experiment was to demonstrate proof of principle for the use of the two detectors as a system for fast treatment plan verification. This was to be achieved by using the detectors to jointly assign a measurement of position (from the PRaVDA tracker) and a measurement of energy (from the UCL calorimeter) to protons passing first through the tracker and then into the calorimeter. The beam was first to be directed straight into the detector array, then via a target to degrade

the beam energy, then finally partially degraded by a target covering half the lateral profile of the beam. It was believed that timing information extracted from each detector could be used to correlate a positional measurement with one of energy and generate a profile of energy deposition in 2D. Proof of principle would be achieved by reconstructing the shape of the beam-degrading target in the energy deposition profile.

3.2 Experimental Setup and Procedure

On the day of the experiment, the cyclotron at the University of Birmingham was experiencing difficulties with the RF circuit used to drive the electrodes and consequently delivered a 28 ± 0.15 MeV proton beam as opposed to the initial target of 36 MeV. The cyclotron produced a beam of 50mm diameter, which was then passed through collimators of either 2mm or 5mm diameter, having the effect of reducing the rate of particles entering the detectors from the rate at which they were emitted by the accelerator in addition to collimating the beam.

The PRaVDA tracker was mounted to a bench near to the beam nozzle, with the UCL calorimeter placed on a stand behind it, as shown in figures 12 and 13. The detectors were aligned by the application of a small piece of radiographic film to the entry window of the UCL calorimeter, running the beam, and then adjusting the position of the calorimeter and repeating until the resulting dark spot on the film was approximately centred. The calorimeter PMT was powered by a Caen NDT1470 high voltage supply over an SHV cable to the detector enclosure and PMT signals were read out by the LeCroy HDO6104 via an SMA to BNC cable from the enclosure. The HV supply was connected to a control laptop for monitoring and operation. The HV supply, the oscilloscope, and the control laptop remained in the experiment room with the detector. The oscilloscope and control laptop were both connected to a network hub in the experiment room via ethernet cables, which in turn was connected to a second network hub in the control room. Two laptops were

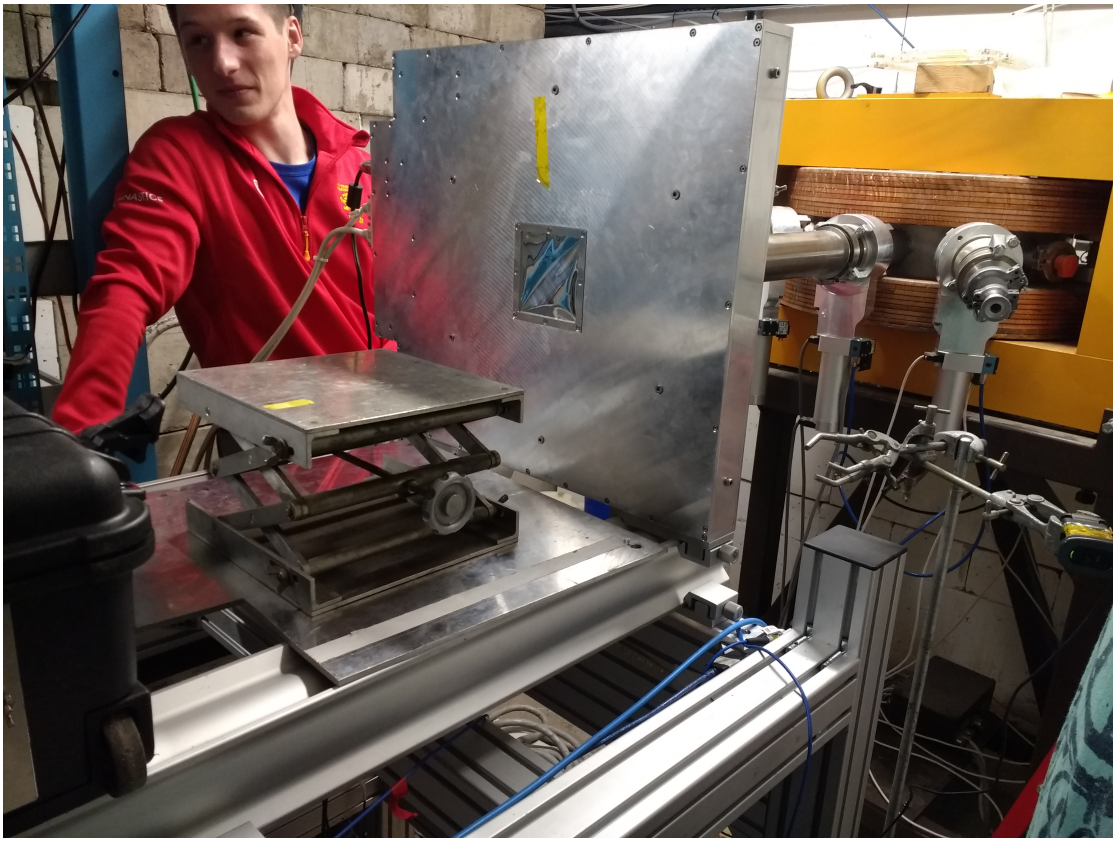

Figure 12: The PRaVDA tracker module mounted on the bench in front of the beam nozzle. The stand on which the UCL calorimeter prototype was placed can be seen in front of the tracker.

connected to this control room hub. Each one ran a remote desktop, one to control the oscilloscope and one to control the HV supply. A schematic of the control setup is shown in Fig. 14. Inside the detector enclosure, the PMT was mounted in a holder on an optical breadboard and coupled head-on to the scintillator block with optical gel.

The lights in the experimental room were switched off, and the HV supply set to -900 V. The PMT dark current was recorded as 150 muA. The oscilloscope was set to trigger on a negative edge. Data was collected for a range of conditions, summarised in Table 1. Where the beam energy was degraded, this was achieved by affixing 1 mm polyethylene (PE) sheets to the collimator, before any component of the detector array. Examples are shown in Fig. 15 of sheets used to degrade the entire beam and

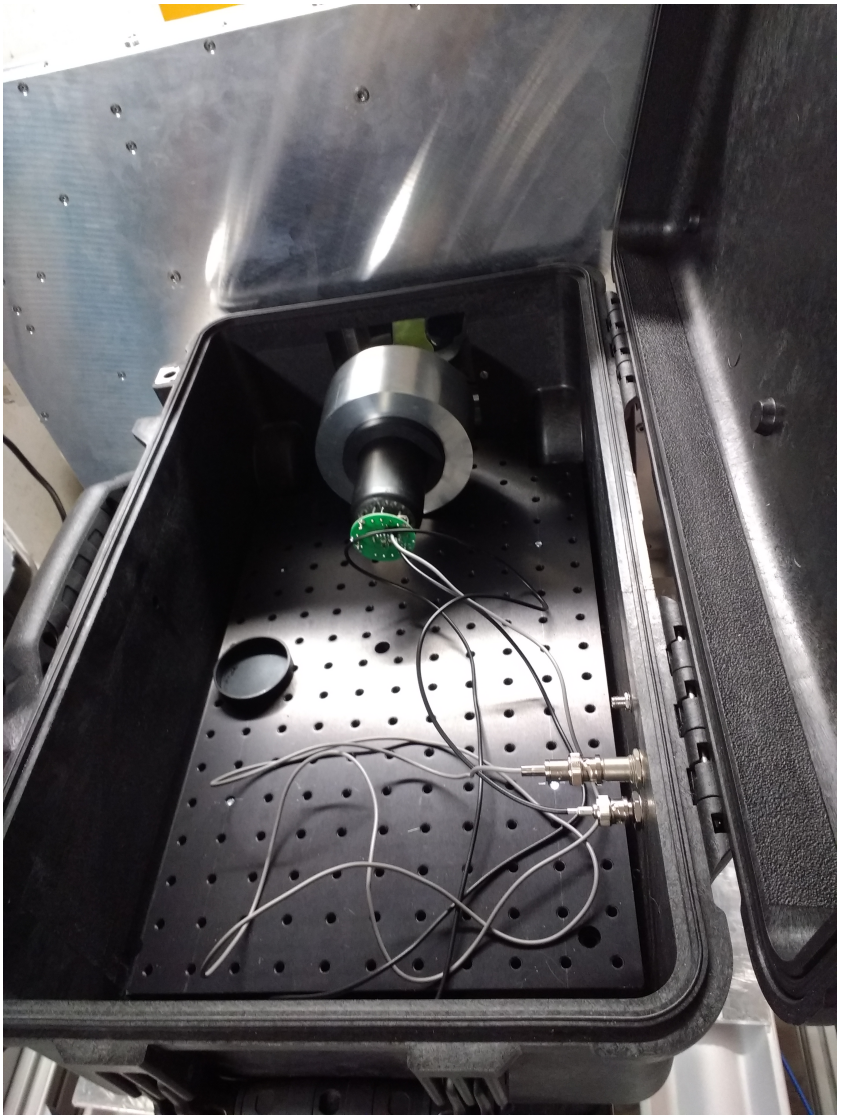

Figure 13: The interior of the UCL calorimeter prototype enclosure. The PMT and its holder are visible, as is the PRaVDA tracker module to which the calorimeter is aligned.

part of the beam profile respectively.

The first four entries in Table 1 correspond to investigating appropriate trigger levels and beam rates to use in order to minimise pileup. The PRaVDA tracker was used to establish the rate of protons entering the detector, and the response of the calorimeter was monitored to identify pileup. It was found that appropriate

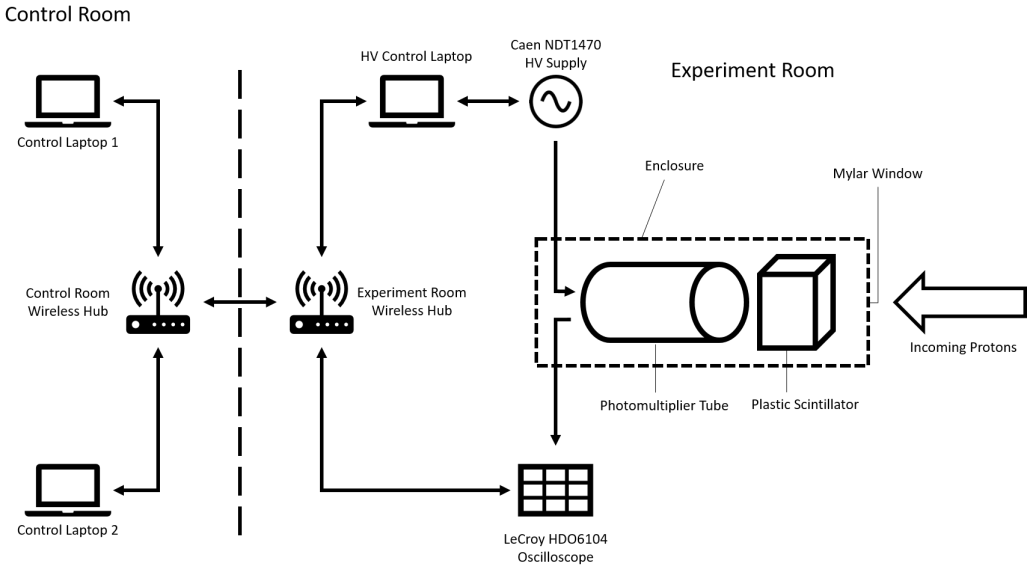

Figure 14: Schematic representing the control setup for the calorimeter.

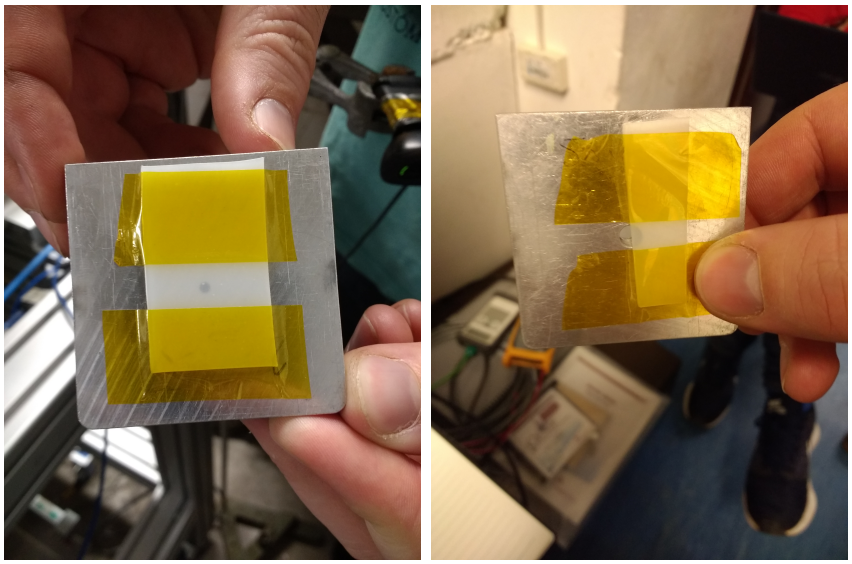

Figure 15: Left: The 2 mm collimator, with 1 mm polyethylene cover. Right: The 5 mm collimator with 1 mm polyethylene half-cover.

detection rates were on the order of 10 kHz. The trigger level was set to 140 mV in order to reduce background effects. The remaining entries in Table 1 correspond to tests of the detectors in the presence and absence of a target to degrade the beam energy.

Between the two test types, a total of 11 runs were performed with both detectors,

Scope Trigger Level	Beam Current / Rate	Collimator	Degrader
-100 mV	≈10 kHz	2 mm	None
-140 mV	150 pA / ≈10 kHz	2 mm	None
-70 mV	260 pA	2 mm	None
-40 mV	260 pA	2 mm	None
-140 mV	1 nA	2 mm	None
-140 mV	220 pA	2 mm	1 mm PE
-140 mV	160 pA	2 mm	1 mm PE
-140 mV	10 pA / ≈30 kHz	5 mm	None
-140 mV	10 pA	5 mm	1 mm PE
-140 mV	10 pA	5 mm	1 mm PE half-cover

Table 1: Summary of test conditions used at the beam test of the combined calorimeter and tracker detector on the 21st of February.

including repeat runs. In each run, the LeCroy oscilloscope first began recording the PMT output, then the PRaVDA tracker began recording for a period of 60 s. During this period, the beam was started. Once the PRaVDA recording period had elapsed, first the beam, and then recording of the calorimeter output, were stopped. The recording startup sequence was intended to provide a reference point for the start of the beam in the data collected by each detector.

3.3 Analysis and Results

ASCII format files containing nanosecond-precision timestamps and Cartesian positional coordinates for tracker hits in each of the eleven runs were provided by Marianna Chiesa, a Master's student working with the PRaVDA tracker modules. Modifications and exstensions were made to the LeCroyData code to make representation of timestamps consistent across datasets taken with both detectors and to allow easy manipulation of the tracker data alongside LeCroyData objects. With these adjustments made, the expanded code was deployed in the analysis of the data collected on the 21st of February.

Initially, calorimeter data from five runs were loaded: for the 2 mm collimator

with and without the 1 mm PE sheet, the 5 mm collimator with and without the 1 mm PE sheet, and the 5 mm collimator half-covered by the 1 mm PE sheet. In all cases, only the data recorded with the 140 mV trigger level was considered. The ADC spectra generated for each of the 2 mm collimator runs are shown in Fig. 16 and the spectra for the 5 mm collimator runs are shown at the top of Fig. 17. In each case, the spectra were seen to be much less well defined than was expected from results shown in section 2.2. Peaks of the distributions were much broader than those seen previously. Long tails towards low ADC counts are only observed in the absence of the PE collimator cover, and the heights of such tails are only marginally less than the heights of the peaks in these distributions. Upon seeing this result, it was initially assumed that the width or offset of the integration window needed to be adjusted for the new datasets, but varying these parameters yielded virtually identical results when the bounds of the integration enclosed the peak of the pulse. This wide spread is inconsistent with the stated energy spread in the cyclotron output, and it is suspected that it is caused by the presence of the PRaVDA tracker module upstream of the calorimeter.

Importantly for the goal of reconstructing the shape of the PE half-cover in the beam profile, it was seen that the spectrum corresponding to that test condition featured a higher and a lower peak in the ADC counts. Comparison of the sum of the spectra with and without the complete cover of PE to the spectrum for the PE half-cover in the lower part of Fig. 17 indicates that the half-cover spectrum is composed of samples from each of the two distributions as expected. It is interesting to note that the low ADC count peak is smaller and the high ADC count peak larger in the half-cover spectrum than in the sum of the covered and uncovered spectra. This reflects the fact that the "half-cover" in fact covers less than half of the collimator aperture, which can be seen by closer inspection of Fig. 15.

Attempts were then made to reconstruct the shape of the half-cover in a 2D

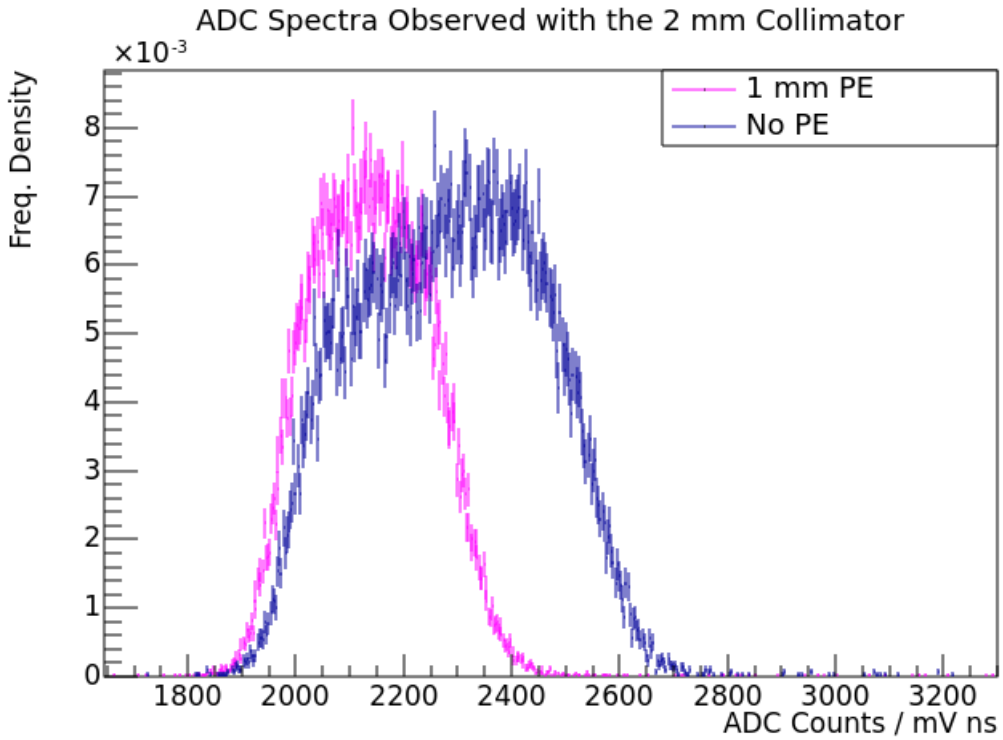

Figure 16: The ADC spectra calculated from runs using a 2 mm collimator with and without the 1 mm PE cover.

distribution of ADC counts using the data from each detector combined. The tracker data for the run was loaded and a histogram of the x and y coordinates of the hits was constructed. This plot is shown in Fig. 18.

Timestamps were then compared for the calorimeter and tracker hits. The first recorded hit for the calorimeter was found to be removed by approximately 3 minutes and 39 seconds from the first hit recorded by the tracker, with the duration of each run being on the order of a minute. This was attributed to a discrepancy between the clocks on the two detectors, as the runs were certainly taken concurrently even if the first hits in each detector were not simultaneous. Unfortunately, this discrepancy meant that correlation of a tracker hit with a calorimeter hit through a shared "global" timestamp was impossible.

Instead, an offset was applied to each timestamp in the tracker dataset such that one hit selected from the calorimeter dataset coincided perfectly with one hit

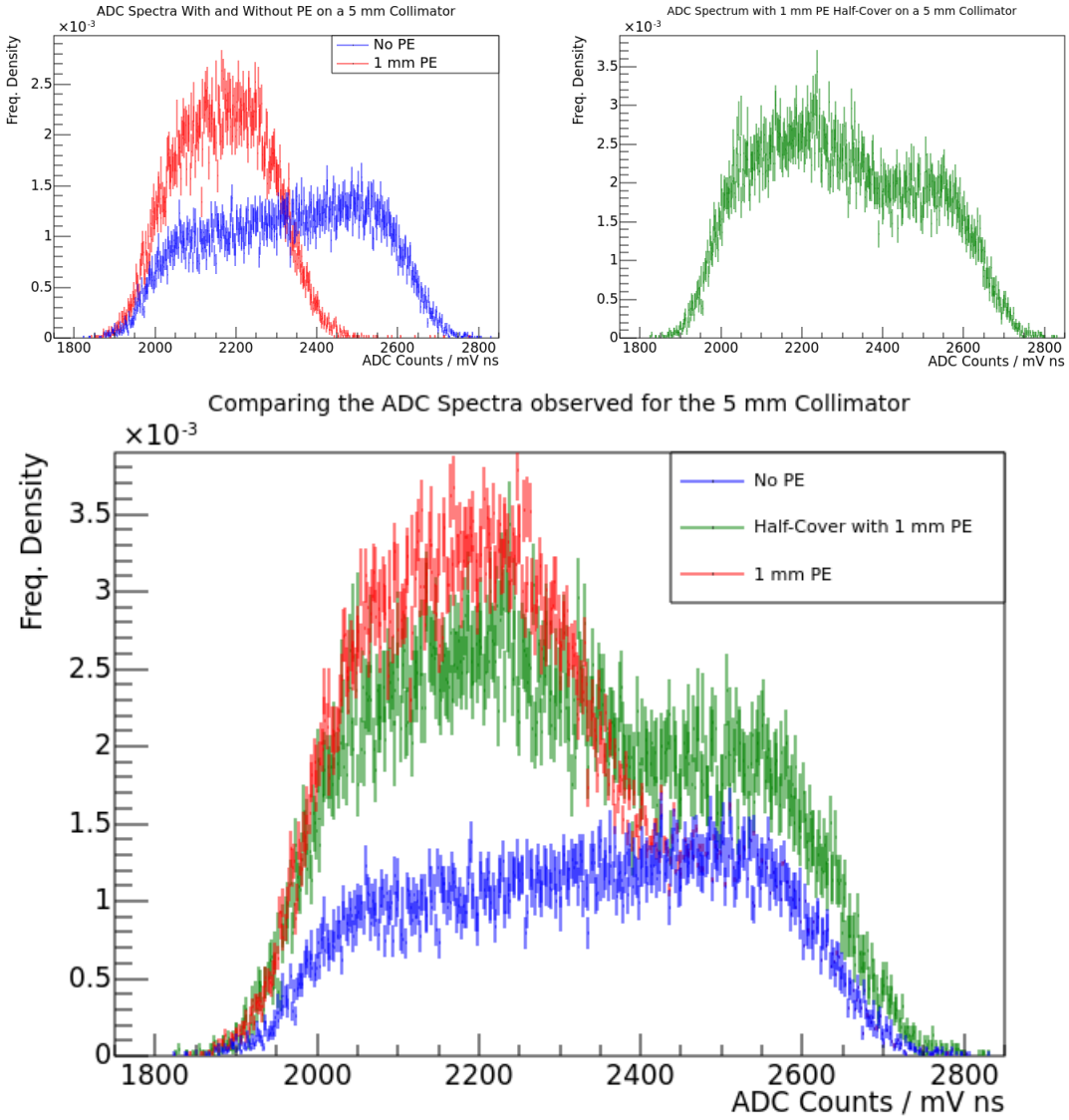

Figure 17: Top left: The ADC spectra calculated from runs using a 5 mm collimator with and without the 1 mm PE cover. Top right: The ADC spectrum calculated from the run using a 5 mm collimator half-covered by 1 mm PE. Bottom: The spectrum on the top right demonstrating a similar structure to the sum of the two shown in the top left.

selected from the tracker dataset to force a pairing. Each hit in the calorimeter dataset was then paired with the hit in the tracker dataset which had the closest possible timestamp. Pairs with an absolute time difference greater than a threshold value were rejected, with the remaining pairs considered "matched". This process was repeated for many choices of forced pairs in order to seek the choice of offset resulting

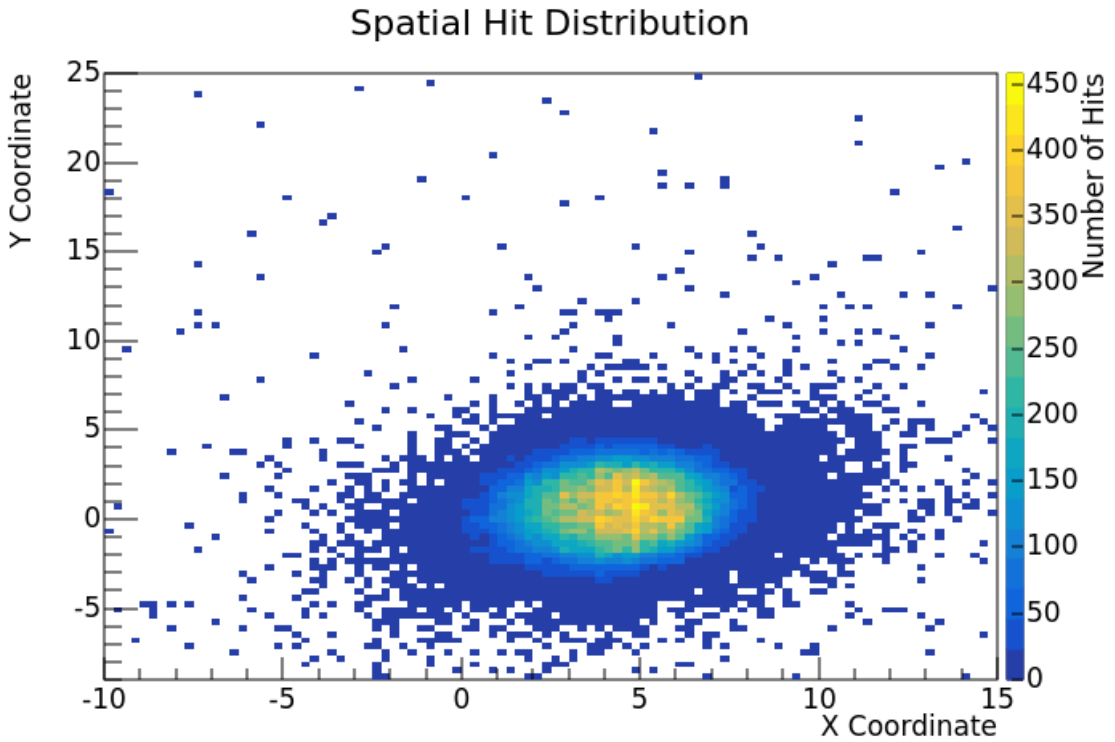

Figure 18: The spatial distribution of hits recorded by the PRaVDA tracker module in the case of the 5 mm collimator half-covered by 1 mm PE.

in the most matching pairs, which was assumed to correspond to a "correct" offset. Initially, the procedure was to be carried out for all possible pairs in the two datasets for completeness, but this was found to be unreasonably computationally intensive. Instead, the first 1000 hits in the calorimeter dataset and the first 2000 points in the tracker dataset were considered, the ratio of these numbers being approximately the ratio of the number of hits recorded by each detector. The threshold for allowed pairings was first set to 10^{-7} seconds. This was motivated by a calculation of the time of flight for 28 MeV protons over a distance on the order of 1 cm, found to be on the order 10^{-10} to 10^{-9} seconds, making this choice of threshold quite generous. Under these conditions, the only matched pairs found were the forced matches assumed at the start of the search. The threshold was relaxed to 10^{-5} seconds, resulting in a maximum of 999 matches for the same search region. The positional coordinates of these matched hits were binned in a 2D histogram as in Fig. 18, weighted by the

corresponding ADC counts. Division by the the unweighted spatial distribution of tracker hits yielded the average ADC counts for a proton passing through each region of the tracker under the assumption that the matched hits corresponded to the same proton. The shape of the PE half-cover was not evident in the distribution, and the validity of the matching process with this relaxed threshold is dubious. It was concluded that correlation of hits across the two detectors had failed.

It is suspected that this failure is due to the different systems responsible for triggering each detector. With each detector triggered automatically and independently on incoming signals, there is no guarantee of correspondence between a hit recorded by one detector and any hit recorded by the other. With no correlation between hit position and hit time for particles in the beam, a "best fitting times" approach is insufficient to pair hits across the two detectors.

4 Summary and Conclusions

The first part of the project can be considered a success. Access to data stored in LeCroy binary files, and calculation of the ADC spectra from this data, was made to be fast and straightforward, and the process was made to integrate directly with the group's chosen data analysis package. Spectra generated were shown to be consistent with the group's previous work, and a narrowing of the spectrum peak may imply that the LeCroy oscilloscope could provide readout of the prototype single-module calorimeter with an improved resolution by comparison to the Caen ADC. It should be noted, however, that the final implementation of the LeCroyData class does not particularly satisfy the goal of identifying and mitigating the effects of pileup in the calorimeter output.

The objective of the second part of the project was failed. Proof of principle could not be established for the use of the calorimeter prototype with PRaVDA tracker modules for the recreation of dose depositions from proton beams. This is

primarily due to the insufficiency of the timestamps recorded by the two detectors for the pairing of events between them. Furthermore, the spreading of the beam energy spectrum apparently caused by the tracker module may present difficulties for the use of the two detectors as a system for mapping dose depositions in a clinical setting, where ideally measurements are directly representative of the energies delievered by the beam.

4.1 Directions for Further Work

- The PRaVDA tracker recently recieved an upgrade allowing the output of a signal when the tracker is triggered. Such a signal could be used to trigger the LeCroy oscilloscope, and thereby overcome the difficulties encountered here with syncronising events between the two detectors. This would require only minimal modifications to the code written in this project for the analysis of the oscilloscope output, and may provide the results which could not be achieved here.

- In order to better identify the source of the spectrum smearing observed in measurements taken with the two detectors, a physics simulation package such as GEANT4 should be used to model the PRaVDA tracker and its interactions with the incoming proton beam.

- There are a number of modifications that could be made to the LeCroyData class and supporting code in order to improve its efficiency and user friendliness. Efficiency gains could be made by parallelisation of the calculation of ADC counts across many waveforms, and the structure of the class could be made more modular in order to allow the use of customised proceedures without requiring modifications to the base class responsible for loading binary data. LeCroyData objects also currently have a relatively large footprint in memory. Unwanted data fields could be identified and removed from the class

to reduce the size of the resulting objects. Steps could also be taken to better quantify the resolution achieved by the LeCroy oscilloscope through the use of the LeCroyData code.

References

[1] Elizabeth Martin. *Concise colour medical dictionary*. Oxford University Press, 2015.

[2] A. Coates et al. "On the receiving end–patient perception of the side-effects of cancer chemotherapy." In: *Eur. J. Cancer* 19.2 (Feb. 1983), pp. 203–208.

[3] Hervé Fanet. *Photon-based medical imagery*. John Wiley & Sons, 2013.

[4] C. Patrigani et al. "Review of Particle Physics". In: *Chinese Physics C* 40.10 (2016).

[5] William Henry Bragg and R Kleeman. "LXXIV. On the ionization curves of radium". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 8.48 (1904), pp. 726–738.

[6] Torunn I. Yock and Nancy J. Tarbell. "Technology Insight: proton beam radiotherapy for treatment in pediatric brain tumors". In: *Nature Clinical Practice Oncology* 1 (Dec. 2004). Review Article. URL: http://dx.doi.org/10.1038/ncponc0090.

[7] Robert R Wilson. "Radiological use of fast protons". In: *Radiology* 47.5 (1946), pp. 487–491.

[8] David Jette and Weimin Chen. "Creating a spread-out Bragg peak in proton beams". In: *Physics in Medicine & Biology* 56.11 (2011), N131.

[9] Jerry B Marion and Barbara A Zimmerman. "Multiple scattering of charged particles". In: *Nuclear Instruments and Methods* 51.1 (1967), pp. 93–101.

[10] Wayne D Newhauser and Rui Zhang. "The physics of proton therapy". In: *Physics in Medicine & Biology* 60.8 (2015), R155.

[11] G. T. Armstrong et. al. "Aging and Risk of Severe, Disabling, Life-Threatening, and Fatal Events in the Childhood Cancer Survivor Study". In: *Journal of Clinical Oncology* 32.12 (Apr. 2014).

[12] B Schaffner and E Pedroni. "The precision of proton range calculations in proton radiotherapy treatment planning: experimental verification of the relation between CT-HU and proton stopping power". In: *Physics in Medicine & Biology* 43.6 (1998), p. 1579.

[13] Anastasia Freshville, SuperNEMO Collaboration et al. "Calorimeter R&D for the SuperNEMO Double Beta Decay Experiment". In: *Journal of Physics: Conference Series*. Vol. 293. 1. IOP Publishing. 2011, p. 012037.

[14] Robert P Johnson. "Review of medical radiography and tomography with proton beams". In: *Reports on Progress in Physics* 81.1 (2017).

[15] JT Taylor et al. "A new silicon tracker for proton imaging and dosimetry". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 831 (2016), pp. 362–366.

[16] J.B. Birks. *The Theory and Practice of Scintillation Counting*. Oxford: Pergamon, 1964.

[17] D.H. Wilkinson. "Ionization energy loss by charged particles Part I. The Landau distribution". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 383.2 (1996), pp. 513–515. ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(96)00774-7. URL: http://www.sciencedirect.com/science/article/pii/S0168900296007747.

[18] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. U.S. Department of Commerce,

NIST. URL: https://app.knovel.com/hotlink/khtml/id:kt00A81VR1/handbook-mathematical/integration.

[19] John Daintith and Edmund Wright. *A Dictionary of Computing (6 ed.)* URL: http://www.oxfordreference.com/view/10.1093/acref/9780199234004.001.0001/acref-9780199234004-e-4622 (visited on 20/03/2018).

Appendix

The final implementation of the LeCroyData class used in this project, including modifications required in section 3. Additional helper scripts and classes, such as those handling the event timestamps and the pairing of hits between the calorimeter and tracker are excluded for brevity.

```
//LECROYHIT CLASS//

class lecroyHit{
  // To be generated by the LeCroyData class as a way to handle a
    single
  // ADC counts record along with its associated TimeStamp
  // This makes it easier to handle runs spanning multiple binary files
  private:
    TimeStamp* t;
    double adc_counts;

  public:
    lecroyHit(double e, TimeStamp* t);
    TimeStamp* getTime();
    double getCounts();
};


lecroyHit::lecroyHit(double e, TimeStamp* t){ adc_counts = e; this->t =
    t; }
TimeStamp* lecroyHit::getTime(){return t;}
double lecroyHit::getCounts(){return adc_counts;}
```

```
25 //LECROYDATA CLASS//
26
27 class LeCroyData {
28
29   private: //Declaring fields to store data
30
31     friend class TimeStamp;
32
33     std::string fileName, templateName, instrumentName, traceLabel,
     timestamp,
34     userText, vertUnit, horizUnit;
35
36     char* fileBuffer; //An array of all the bytes in the binary .trc
     file
37
38     int wavedescIndex, fileSize, pointsPerAcq, pointsPerWindow;
39
40     short commOrder, commType, nominalBits;
41
42     long waveDescriptorSize, userTextSize, waveArraySize,
     trigtimeArraySize, subArrayCount,
43     instrumentNumber, firstValidPoint, lastValidPoint, sparsingFactor,
     segmentIndex;
44
45     float verticalGain, verticalOffset, horizInterval, horizUncertainty
     , acquisitionDuration, maxValue, minValue;
46
47     double *trigtimeArray, *horizOffsetArray, *timeArray, *waveArray,
     maxBaselineSigma = 15., blStart, blEnd, offset = -60., window= 150.,
      dx;
48
49     std::vector<double> spectrumTime;
50     std::vector<double> spectrumADCCounts;
```

```
51      std::vector<double*> goodAcqs;
52      std::vector<int> goodAcqNums;
53      int spectrumSize;
54
55      bool isTrace(std::string fileName); //Returns true if the last 4
        characters in fileName are consistent with the file extension for
56
57      short getShort(int byteLocation); //Creates a short from the
        fileBuffer, starting at byteLocation
58      long getLong(int byteLocation); //Creates a long from the
        fileBuffer, starting at byteLocation
59      float getFloat(int byteLocation); //Creates a float from the
        fileBuffer, starting at byteLocation
60      double getDouble(int byteLocation); //Creates a double from the
        fileBuffer, starting at byteLocation
61      int getInt(int byteLocation); //Creates an int from the fileBuffer,
         starting at byteLocation
62      signed char getByte(int byteLocation); //Returns the byte at
        byteLocation in the fileBuffer as a signed char
63      std::string get16CharString(int byteLocation); //Creates a string
        of 16 characters from the fileBuffer, starting at byteLocation
64      std::string getText(int byteLocation, long byteLength); //Creates a
         string of byteLength characters from the fileBuffer, starting at
        byteLocation
65      std::string makeTimestamp(int byteLocation); //Creates a string
        describing when the file was created using information from
        byteLocation in the fileBuffer
66
67      void buildSpectrum();
68      double compositeIntegrate(double* yPoints, long nPoints, double
        xStep);
69      void baselineSubtraction();
```

```
70      double meanEstimateBaseline(double* y, double* timeArray, double
        sigLimit);
71
72  public:
73      LeCroyData(std::string fileName); //Construct a LeCroyData object
        using the fileName specifying the file to load
74      ~LeCroyData(void);
75
76      double* getWaveArray(void); //Getter for vector of oscilloscope
        voltage values
77      double* getAcqWave(int segment); //Getter for vector of voltage
        values for a single acquisition indexed by the int segment
78
79      double* getTimeArray(void); //Getter for vector of oscilloscope
        time values
80      double* getAcqTime(int segment); //Getter for vector of time values
         for a single acquisition indexed by the int segment
81
82      double* getOffsetArray(void); //Get a vector of the horizontal
        offsets associated with each trigger
83      double* getTriggerArray(void); //Get a vector of trigger times (in
        seconds, relative to first trigger)
84
85      std::string getInstrumentName(void); //Returns the name of the
        oscilloscope, if it exists
86      std::string getTemplateName(void); //Returns the name of the data
        template, if it exists
87      std::string getTraceLabel(void); //Returns the label for the trace
        file, if it exists
88      std::string getTimestamp(void); //Returns a string describing when
        the file was created
89      std::string getHeader(void); //Returns a string describing the file
```

```
90      std::string getFileName(void); //Returns the name of the file
        represented by the object
91
92      float getTimeUncertainty(void); //Returns the uncertainty in time
        measurements (in seconds)
93      float getMaxSignal(void); //Returns the highest voltage recorded in
         the file
94      float getMinSignal(void); //Returns the lowest voltage recorded in
        the file
95
96      long getAcqCount(void); //Returns the number of acquisitions
        recorded in the file
97      long getInstrumentNumber(void); //Returns a number identifying the
        oscilloscope
98
99      int getPointsPerAcq(void); //Returns the number of voltage readings
         in each acquisition
100
101     double* getSpectrum(); //Returns an array containing the value of
        ADC counts for each pulse in the file
102     double* getSpectrumTime(); //Returns an array containing the time
        at which each pulse in the spectrum was seen (starting from the
        first trigger in the file)
103
104     std::vector<TimeStamp*> getSpectrumTimestamps();
105
106     int getSpectrumSize(); //Returns the number of points in the
        spectrum
107
108     void setMaxBaselineSigma(double level); //Set the maximum allowed
        standard deviation in the baseline before an acquisition is rejected
109     void setIntegrationWindow(double offset, double width = 150.);
110
```

```
111     std::vector<lecroyHit*> yieldHits();
112
113 };
114
115
116 LeCroyData::LeCroyData(std::string fileName){
117
118   if(!isTrace(fileName)){throw "Given filename is not a LeCroy binary
        trace file!";}
119   this->fileName = fileName;
120
121   //open the file at the last byte to determine the file size in bytes
122   std::ifstream target(fileName, std::ios::in|std::ios::binary|std::ios
        ::ate);
123   fileSize = target.tellg();
124
125   //Identify the byte corresponding to the start of "WAVEDESC"
126   char memblock[50];
127   target.seekg(0);
128   target.read(memblock, 50);
129   std::string blockString(memblock);
130   wavedescIndex = blockString.find("WAVEDESC");
131
132   //read the file into memory
133   target.seekg(0);
134   fileBuffer = new char[fileSize];
135   target.read(fileBuffer, fileSize);
136   target.close();
137
138   //read in each data field from its known byte location
139   templateName = get16CharString(wavedescIndex + 16);
140   commType = getShort(wavedescIndex + 32);
141   commOrder = getShort(wavedescIndex + 34);
```

```
142  waveDescriptorSize = getLong(wavedescIndex + 36);
143  userTextSize = getLong(wavedescIndex + 40);
144  trigtimeArraySize = getLong(wavedescIndex + 48);
145  waveArraySize = getLong(wavedescIndex + 60);
146  instrumentName = get16CharString(wavedescIndex + 76);
147  instrumentNumber = getLong(wavedescIndex + 92);
148  traceLabel = get16CharString(wavedescIndex + 96);
149  firstValidPoint = getLong(wavedescIndex + 124);
150  lastValidPoint = getLong(wavedescIndex + 128);
151  sparsingFactor = getLong(wavedescIndex + 136);
152  segmentIndex = getLong(wavedescIndex + 140);
153  verticalGain = getFloat(wavedescIndex + 156);
154  verticalOffset = getFloat(wavedescIndex + 160);
155  maxValue = getFloat(wavedescIndex + 164);
156  minValue = getFloat(wavedescIndex + 168);
157  nominalBits = getShort(wavedescIndex + 172);
158  horizInterval = getFloat(wavedescIndex + 176);
159  vertUnit = getText(wavedescIndex + 196, 48);
160  horizUnit = getText(wavedescIndex + 244, 48);
161  horizUncertainty = getFloat(wavedescIndex + 292);
162  timestamp = makeTimestamp(wavedescIndex + 296);
163
164  subArrayCount = trigtimeArraySize/(2*sizeof(double));
165
166  userText = getText(wavedescIndex +(int)waveDescriptorSize,
       userTextSize);
167
168  //Loads the TRIGTIME_ARRAY block into a pair of double[]s
169  trigtimeArray = new double[subArrayCount];
170  horizOffsetArray = new double[subArrayCount];
171  for(int i = 0; i<subArrayCount; i++){
172    trigtimeArray[i] = 1e9*(getDouble(wavedescIndex + (int)
       waveDescriptorSize + (int)userTextSize + 2*i*sizeof(double)));
```

```
173     horizOffsetArray[i] = 1e9*(getDouble(wavedescIndex + (int)
    waveDescriptorSize + (int)userTextSize + (2*i+1)*sizeof(double)));
174   }
175
176   if(commType == 0){ //If the WAVE_ARRAY is expressed in bytes...
177     waveArray = new double[waveArraySize];
178     for(int i = 0; i<waveArraySize; i++){ //Load the waveArray using
    bytes
179       waveArray[i] = 1e3*((double)verticalGain*getByte(wavedescIndex +
    waveDescriptorSize + userTextSize + trigtimeArraySize + i) - (double
    )verticalOffset);
180       }
181     }
182   else{
183     waveArray = new double[waveArraySize/2];
184     for(int i = 0; i<waveArraySize/2; i++){ //If the WAVE_ARRAY is
    expressed in words, load the waveArray using words
185       waveArray[i] = 1e3*((double)verticalGain*getShort(wavedescIndex +
     waveDescriptorSize + userTextSize + trigtimeArraySize + 2*i) - (
    double)verticalOffset);
186       }
187     }
188
189   //Calculate the number of points in each triggered acquisition
190   pointsPerAcq = waveArraySize/(subArrayCount*(commType+1));
191
192   //Calculate the time in seconds of each data point in the waveArray
     and append it to the timeArray
193   timeArray = new double[subArrayCount*pointsPerAcq];
194   for(int i = 0; i<subArrayCount; i++){
195     for(int j = 0; j<pointsPerAcq; j++){
196       timeArray[j + i*pointsPerAcq] = (horizOffsetArray[i] +
    trigtimeArray[i] + 1e9*j*horizInterval);
```

```
197       }
198
199     }
200
201     dx = timeArray[1] - timeArray[0]; //Integration timestep
202     pointsPerWindow = (int)(window/dx);
203
204     //Clean up
205     delete fileBuffer;
206
207     baselineSubtraction();
208     buildSpectrum();
209 }
210
211
212 LeCroyData::~LeCroyData(){
213     //Clean up arrays when we delete the object:
214     delete[] waveArray; delete[] timeArray; delete[] horizOffsetArray;
        delete[] trigtimeArray;
215     for(double* a : goodAcqs){ delete[] a; }
216     goodAcqs.clear();
217 }
218
219
220 double* LeCroyData::getWaveArray(void){return waveArray;}
221
222
223 double* LeCroyData::getTimeArray(void){return timeArray;}
224
225
226 double* LeCroyData::getOffsetArray(void){return horizOffsetArray;}
227
228
```

```
229 long LeCroyData::getAcqCount(void){return subArrayCount;}
230
231
232 int LeCroyData::getPointsPerAcq(void){return pointsPerAcq;}
233
234
235 std::string LeCroyData::getTimestamp(void){return timestamp;}
236
237
238 std::string LeCroyData::getTraceLabel(void){return traceLabel;}
239
240
241 std::string LeCroyData::getInstrumentName(void){return instrumentName;}
242
243
244 long LeCroyData::getInstrumentNumber(void){return instrumentNumber;}
245
246
247 double* LeCroyData::getTriggerArray(void){return trigtimeArray;}
248
249
250 float LeCroyData::getTimeUncertainty(void){return horizUncertainty;}
251
252
253 float LeCroyData::getMaxSignal(void){return maxValue;}
254
255
256 float LeCroyData::getMinSignal(void){return minValue;}
257
258
259 std::string LeCroyData::getFileName(){return fileName;}
260
261
```

```
262 double* LeCroyData::getAcqWave(int segment){
263   //Getter method for an individual waveform acquisition, indexed by
      the int segment
264   double *waveform = new double[pointsPerAcq];
265   for(int i = 0; i<pointsPerAcq; i++){ //FOr each point in the
      acquisition...
266     waveform[i] = waveArray[segment*pointsPerAcq + i]; //Fetch the
      point from the specified segment
267   }
268   return waveform;
269 }
270
271
272 double* LeCroyData::getAcqTime(int segment){
273   //Getter method for the timing data for the individual waveform
      acquisition indexed by the int segment
274   double *segmentTimes = new double[pointsPerAcq];
275   for(int i = 0; i<pointsPerAcq; i++){ //For each point in the
      aquisition...
276     segmentTimes[i] = timeArray[segment*pointsPerAcq + i]-trigtimeArray
      [segment]; //Fetch the point from the specified segment
277   }
278   return segmentTimes;
279 }
280
281
282 std::string LeCroyData::getText(int byteLocation, long byteLength){
283   //Converts byteLength bytes from the fileBuffer, starting from
      byteLocation, into a string
284   std::ostringstream text;
285   for(int n = n; n<byteLength; n++){ //For each character in the
      specified region of the fileBuffer...
```

```
286     text<<fileBuffer[byteLocation+n]; //Append the character to the
        string
287   }
288   return text.str();
289 }
290
291
292 std::string LeCroyData::get16CharString(int byteLocation){
293   //Converts bytes from the fileBuffer, starting from byteLocation,
        into a 16-character string
294   return getText(byteLocation, 16);
295 }
296
297
298 std::string LeCroyData::makeTimestamp(int byteLocation){
299   //Constructs a string describing the date and time at which the file
        was created
300   //as specified in the LeCroy X-Stream manual
301   std::ostringstream stamp;
302
303   //Get timing info
304   double sec = getDouble(byteLocation);
305   signed char min = getByte(byteLocation + 8);
306   signed char hrs = getByte(byteLocation + 9);
307   signed char day = getByte(byteLocation + 10);
308   signed char mon = getByte(byteLocation + 11);
309   short yrs = getShort(byteLocation + 12);
310
311   int intsec = (int)sec;
312
313   double millis = (sec-intsec)*1e3;
314   int intms = (int)millis;
315
```

```
316   double micros = (millis - intms)*1e3;
317   int intmus = (int)micros;
318
319   double nanos = (micros - intmus)*1e3;
320   int intns = (int)nanos;
321
322   //Construct the string
323   stamp<<yrs<<"/"<<(int)mon<<"/"<<(int)day<<"/"<<(int)hrs<<"/"<<(int)
       min<<"/"<<intsec<<"/"<<intms<<"/"<<intmus<<"/"<<intns;
324
325   return stamp.str();
326 }
327
328
329 std::string LeCroyData::getHeader(void){
330   //Returns a string describing the file
331
332   std::ostringstream header;
333
334   header<<"Instrument name: \t"<<instrumentName<<std::endl;
335   header<<"Instrument number: \t"<<instrumentNumber<<std::endl;
336   header<<"File template: \t"<<templateName<<std::endl;
337   header<<"Timestamp: \t"<<timestamp<<std::endl;
338   header<<"User text: \t"<<userText<<std::endl;
339   header<<"Label: \t"<<traceLabel<<std::endl;
340   header<<"Number of acquisitions: \t"<<subArrayCount<<std::endl;
341   header<<"Points per acquisition: \t"<<pointsPerAcq<<std::endl;
342   header<<"Signal extrema: \t"<<(verticalGain*minValue-verticalOffset)
       <<", "<<(verticalGain*maxValue-verticalOffset)<<std::endl;
343   header<<"Vertical unit: \t"<<vertUnit<<std::endl;
344   header<<"Timing uncertainty: \t"<<horizUncertainty<<std::endl;
345   header<<"Horizontal unit: \t"<<horizUnit<<std::endl;
346   double a, b;
```

```
347   a = timeArray[0];
348   b = timeArray[pointsPerAcq*subArrayCount-1];
349   header<<"First and last timepoints: \t"<<a<<", "<<b<<" (span: "<<(b-a
        )<<horizUnit<<" )"<<std::endl;
350   header<<"Nominal ADC bits: \t"<<nominalBits<<std::endl;
351   return header.str();
352
353 }
354
355
356 bool LeCroyData::isTrace(std::string fileName){
357   //Identifies whether a file name represents a LeCroy trace file based
        on the file extension
358   std::string lastFour = fileName.substr(fileName.length() - 4);
359   if(lastFour == ".trc"){return true;}
360   return false;
361 }
362
363
364 long LeCroyData::getLong(int byteLocation){
365   //Get the vlaue of a long from byteLocation in the fileBuffer
366   long l;
367   char bytes[sizeof l];
368   for(int n = 0; n<(sizeof l); n++){
369     bytes[n] = fileBuffer[n+byteLocation]; //Select bytes to form the
        long
370   }
371   std::memcpy(&l, &bytes, sizeof l); //Copy the bit pattern into the
        long
372   return l;
373 }
374
375
```

```
376 int LeCroyData::getInt(int byteLocation){
377   //Get the value of an int from byteLocation in the fileBuffer
378   int i;
379   char bytes[sizeof i];
380   for(int n = 0; n<(sizeof i); n++){
381     bytes[n] = fileBuffer[n+byteLocation]; //Select bytes to form the
      int
382   }
383   std::memcpy(&i, &bytes, sizeof i); //Copy the bit pattern into the
      int
384   return i;
385 }
386
387
388 short LeCroyData::getShort(int byteLocation){
389   //Get the value of a float from byteLocation in the fileBuffer
390   short s;
391   char bytes[sizeof s];
392   for(int n = 0; n<(sizeof s); n++){
393     bytes[n] = fileBuffer[n+byteLocation]; //Select bytes to form the
      short
394   }
395   std::memcpy(&s, &bytes, sizeof s); //Copy the bit pattern into the
      short
396   return s;
397 }
398
399
400 float LeCroyData::getFloat(int byteLocation){
401   //Get the value of a float from byteLocation in the fileBuffer
402   float f;
403   char bytes[sizeof f];
404   for(int n = 0; n<(sizeof f); n++){
```

```
405     bytes[n] = fileBuffer[n+byteLocation]; //Select bytes to form the
        float
406   }
407   std::memcpy(&f, &bytes, sizeof f); //Copy the bit pattern into the
        float
408   return f;
409 }
410
411
412 double LeCroyData::getDouble(int byteLocation){
413   //Get the value of a double from byteLocation in the fileBuffer
414   double d;
415   char bytes[sizeof d];
416   for(int n = 0; n<(sizeof d); n++){
417     bytes[n] = fileBuffer[n+byteLocation]; //Select bytes to form the
        double
418   }
419   std::memcpy(&d, &bytes, sizeof d); //Copy the bit pattern into the
        double
420   return d;
421 }
422
423
424 signed char LeCroyData::getByte(int byteLocation){
425   //Get the value of a signed char from byteLocation in the fileBuffer
426   signed char b = fileBuffer[byteLocation];
427   return b;
428 }
429
430
431 double LeCroyData::meanEstimateBaseline(double* y, double* timeArray,
      double sigLimit){
```

```
432   //Uses the voltage recorded up to the trigger event to estimate the
       baseline
433   //Labels the value for discarding if the standard deviation of the
       voltage in the pre-trigger region is higher than sigLimit
434
435   int startPoints = 0;
436   while(timeArray[startPoints+1] < 0){startPoints++;} //Identify the
       number of points before the trigger
437
438   startPoints = (startPoints*9)/10; //To avoid the triggering signal
       skewing the value of the baseline
439
440   double startRegion[startPoints];
441   for(int i = 0; i<startPoints; i++){startRegion[i] = y[i];} //Collect
       the voltage values before the trigger in an array...
442   double baseLine = TMath::Mean<double>(startPoints, startRegion); //
       And evaluate the mean
443
444   if(TMath::StdDev<double>(startPoints, startRegion) > sigLimit){return
        -9999.;} //If standard deviation too high, return something
       obviously wrong
445   else{ return baseLine; }  //Otherwise, the mean is the baseline
446 }
447
448
449 void LeCroyData::baselineSubtraction(){
450
451   //Reset the record of "good" acquisitions:
452   for(double* a : goodAcqs){delete[] a;}
453   goodAcqs.clear();
454   goodAcqNums.clear();
455
456   for(int i = 0; i<subArrayCount; i++){ //For each acquisition
```

```
457     double* data = getAcqWave(i);
458     double baseline = meanEstimateBaseline(data, getAcqTime(i),
    maxBaselineSigma); //Calculate the baseline for the acquisiton
459     if(baseline != -9999.){ //Baselines with standard deviation too
    high are tagged by this value
460       //Make a record of the "good" acquisitions:
461       int windowStart = (int)((offset - horizOffsetArray[i])/dx);
462       //std::cout.flush()<<"Integration start index in acquisition "<<i
    <<": "<<windowStart<<std::endl;
463       //std::cout.flush()<<"Bounds within acquisition: "<<((windowStart
     + pointsPerWindow <= pointsPerAcq)&&(windowStart >= 0))<<std::endl;
464       goodAcqs.push_back(new double[pointsPerWindow]);
465       goodAcqNums.push_back(i);
466       for(int j = 0; j<pointsPerWindow; j++){
467         goodAcqs[goodAcqs.size()-1][j] = baseline - data[j+windowStart
    ]; //Store a copy of the acquisition after baseline subtraction
468       }
469     }
470   }
471 }
472
473
474 double LeCroyData::compositeIntegrate(double* yPoints, long nPoints,
    double xStep){
475
476   //An n-point composite Newton-Cotes formula, as found on Wolfram
    Mathworld
477   //Pretty good for nPoints > 8
478   double sum = 0;
479
480   for(long i = 0; i<nPoints; i++){
481     if(i == 0 || i == nPoints -1){sum += 17*yPoints[i]/48.;}
482     else if(i == 1 || i == nPoints -2){sum += 59*yPoints[i]/48.;}
```

```
483     else if(i == 2 || i == nPoints -3){sum += 43*yPoints[i]/48.;}
484     else if(i == 3 || i == nPoints -4){sum += 49*yPoints[i]/48.;}
485     else{sum+=yPoints[i];}
486   }
487
488   return sum*xStep;
489 }
490
491
492 void LeCroyData::buildSpectrum(){
493
494   spectrumADCCounts.clear(); //Clearing any old spectrum data to build
      the new in its place
495   spectrumTime.clear();
496
497   spectrumSize = goodAcqs.size(); //Number of points in the spectrum
498
499   for(int i = 0; i<spectrumSize; i++){ //Calculate a point in the
      spectrum from every "good" acquistion
500     spectrumADCCounts.push_back(compositeIntegrate(goodAcqs[i],
      pointsPerWindow, dx));
501     spectrumTime.push_back(trigtimeArray[goodAcqNums[i]]);
502   }
503 }
504
505
506 void LeCroyData::setMaxBaselineSigma(double level){
507   if(maxBaselineSigma != level){ //If a change is requested
508     maxBaselineSigma = level; //Update the threshold standard deviation
      for baseline calculation
509     baselineSubtraction(); //And recalculate the spectrum
510     buildSpectrum();
511   }
```

```
512 }
513
514
515 void LeCroyData::setIntegrationWindow(double offset, double length){
516
517   this->offset = offset; //Set the location and length of the
      integration window
518   window = length;
519
520   pointsPerWindow = (int)(window/dx); //Calculate the number of data
      points in the window
521
522   baselineSubtraction(); //Recalculate the spectrum
523   buildSpectrum();
524 }
525
526
527 double* LeCroyData::getSpectrum(){ return spectrumADCCounts.data(); }
528
529
530 double* LeCroyData::getSpectrumTime(){ return spectrumTime.data(); }
531
532
533 std::vector<TimeStamp*> LeCroyData::getSpectrumTimestamps(){
534   //Calculate nanosecond-precision TimeStamps for each point in the
      calculated spectrum
535
536   std::vector<TimeStamp*> stamps;
537   TimeStamp* ts;
538
539   for(double t : spectrumTime){
540     ts = new TimeStamp(timestamp);
541     ts->addNanos(t);
```

```
542     stamps.push_back(ts);
543   }
544
545   return stamps;
546 }
547
548
549 std::vector<lecroyHit*> LeCroyData::yieldHits(){
550   //Returns a vector<> of objects more easily handled than LeCroyData
        objects for the processing
551   //of data across many files.
552   std::vector<TimeStamp*> t = getSpectrumTimestamps();
553   std::vector<lecroyHit*> ret;
554
555   for(int i = 0; i < spectrumADCCounts.size(); i++){
556     ret.push_back(new lecroyHit(spectrumADCCounts[i], t[i]));
557   }
558
559   return ret;
560 }
561
562 int LeCroyData::getSpectrumSize(){ return spectrumSize; }
```

Acknowledgments

Thank you to my supervisors, in particular Simon, for providing me with two terms of engaging and meaningful work and valued insight. Thank you also to the UCL PBT group for your advice, your patience, and the welcoming atmosphere. Thank you to Tony Price for the experience I was able to have at the University of Birmingham and for the exciting collaboration the PRaVDA team brought to my project. Finally, thank you to Jordan Silverman, Andy Morris, and Leya George for their unfaltering

moral support.