Data analysis and detector development for LEGEND

By Tarkan Yzeiri

Supervision from Prof. Ruben Saakyan

and

Dr. Matteo Agostini

MSci Physics

Department of Physics and Astronomy

University College London

UCL DEPARTMENT OF PHYSICS & ASTROMOMY

Submission of coursework for Physics and Astronomy course **PHAS0097/PHAS0096/PHAS0048** 2021/22

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

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 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 

PrintName Tarkan Yzeiri

Dated 25/03/22

Title	Date Received	Examiner	Examiner's Signature	Mark

ABSTRACT

LEGEND (Large Enriched Germanium Experiment for Neutrinoless double-beta Decay) is an experiment that utilises high purity ^{76}Ge detectors bathed in liquid argon to probe for neutrinoless double-beta decay. The experiment is split into two stages; LEGEND-200 and LEGEND-1000; that aim to probe a neutrinoless double-beta decay half-life sensitivity of order $T^{0\nu}_{1/2} \sim 10^{27}$ yr, and $T^{0\nu}_{1/2} \sim 10^{28}$ yr respectively. The following report will summarise multiple pulse discrimination analysis techniques performed on ^{228}Th calibration data for detectors to be used in LEGEND-200.

CONTENTS

REPORT

1. INTRODUCTION

Neutrinos were discovered in 1956 [1] and are one of the fundamental particles of the standard model, which were later discovered to possess and oscillate between three flavours; electron, muon [2], and tau [3]. The phenomenon in which neutrinos can change flavour is known as neutrino oscillation theory [4]. Neutrino oscillation theory has been pivotal in advancing our understanding of neutrinos; proving that neutrinos have mass; however it only allows us to find the difference of the mass squared for each neutrino eigenstate. In addition to this, neutrino oscillations also imply that the flavour lepton number is not conserved, as a neutrino can oscillate between these three flavours while propagating through space. By probing neutrinos physics and therefore the discovery of neutrino oscillations, there has been new evidence for physics beyond the standard model. However, to further probe into neutrino physics, we must first understand whether neutrinos are Dirac [5][6] or Majorana [7] fermions. Dirac fermions are spin-$^1/_2$, non-zero mass particles that have an identical anti-particle, but with opposite charge. A Majorana fermion is a fermion that is its own antiparticle. All the fermions in the standard model, bar neutrinos, are of Dirac nature. The mass term for both Dirac and Majorana neutrinos differ and therefore discovering whether neutrinos are Dirac or Majorana fermions is pivotal in improving our understanding of the standard model. Furthermore, there are theories such as the seesaw mechanism [8] that require neutrinos to be Majorana particles. The seesaw mechanism is a model used to describe why neutrino masses are much smaller than the other fermions in the standard model. This is because the Yukawa coupling of fermions such as the top quark ($m_t = 172.76 \pm 0.30$ GeV [9]) to the Higgs field gives a value of approximately one, while if you assume that neutrino masses are also associated with the Higgs field, you obtain values of the Yukawa coupling to the order of $\lesssim 10^{-12}$ [10]. This low Yukawa coupling compared to the other fermions suggests that there is another mechanism that generates the neutrino masses different to how other fermion masses are

generated. This motivates the search for the Majorana nature of neutrinos, where one of the most feasible methods of determining this, is to probe neutrinoless double-beta decay ($0\nu\beta\beta$) [11].

1.1. Neutrinoless double-beta decay ($0\nu\beta\beta$)

Two-neutrino double beta decay [12] is a type of radioactive decay by which the charge of the parent nucleus decreases by two via an emission of a pair of electrons and anti-electron neutrinos, where the two original neutrons in the parent nucleus transmute into a pair of protons. This decay mode is only possible for nuclei with an even number of neutrons and protons where single-beta decay is heavily suppressed, and is shown by:

$$ {}^{A}_{Z}X \longrightarrow {}^{A}_{Z+2}Y + 2e^{-} + 2\bar{\nu}_{e} \tag{1} $$

Where X and Y are the initial and final nuclear species respectively. If neutrinos are Majorana particles, then it is possible for this double beta decay event to occur without emission of a pair of neutrinos. This is called neutrinoless double-beta decay ($0\nu\beta\beta$):

$$ {}^{A}_{Z}X \longrightarrow {}^{A}_{Z+2}Y + 2e^{-} \tag{2} $$

As can be seen above the neutrinoless double-beta decay event violates lepton number by two units, and is therefore forbidden by standard electroweak theory. As neutrinoless double-beta decay is a lepton violating decay, the discovery of this event could explain the matter-antimatter asymmetry in our universe via theories such as leptogenesis [13]. Neutrinoless double-beta decay is allowed as the anti-neutrino pairs can annihilate during creation, leaving just a pair of electrons to be emitted, as shown in Figure 1.1.

Where ν_M in Figure 1.1 refers to the Majorana neutrino. This decay mode is distinguishable from two-neutrino double beta decay due to the energy of the emitted electrons. In two-neutrino double beta decay the electrons have a reduced kinetic energy due to the shared total kinetic energy with the emitted neutrinos, however, in neutrinoless double-beta decay the electrons have the maximum available kinetic energy. This results in a mono-energetic peak at value $Q_{\beta\beta}$, as shown in Figure 1.2.

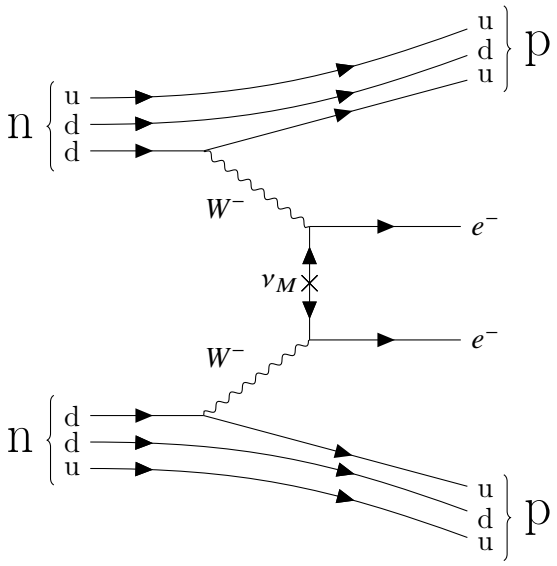


Figure 1.1: Feynman diagram of a neutrinoless double-beta ($0\nu\beta\beta$) decay. This event is only possible if neutrinos are Majorana fermions.

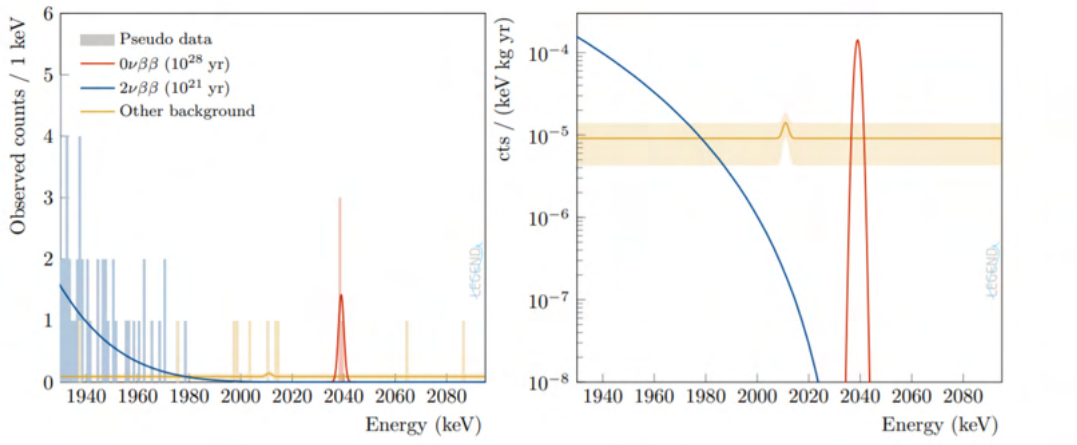


Figure 1.2: A Monte-Carlo pseudo-dataset of LEGEND-1000, generated for the full background model, 10 t yr of exposure, showcasing the electron spectra of both two-neutrino double beta decay ($2\nu\beta\beta$) and neutrinoless double beta decay ($0\nu\beta\beta$) with respective half life values of 10^{28} years and 10^{21} years. The left graph shows the number of observed counts and the right graph shows the observed counts normalised to exposure. The uncertainty on the overall background model is covered by the yellow band. From Ref. [14]

The half life of the neutrinoless double-beta decay process can be calculated by using [15]:

$$(T_{1/2}^{0\nu})^{-1} = G_{0\nu}|\mathcal{M}|^2 m_{\beta\beta}^2 \tag{3}$$

Where $G_{0\nu}$ is the phase space factor, $\mathcal{M}$ the nuclear matrix element, and $m_{\beta\beta}$ the effective Majorana neutrino mass in neutrinoless double-beta decay ($m_{\beta\beta}$). By measuring a value for the half life of neutrinoless double beta decay, you obtain a upper limit for the effective Majorana neutrino mass, which is defined by:

$$m_{\beta\beta} = \left| \sum_{i=1}^{3} U_{ei}^2 m_i \right| \tag{4}$$

Where U_{ei} are the neutrino mixing matrix elements and m_i the neutrino mass eigenvalues. If the effective Majorana neutrino mass is measured, $m_{\beta\beta}$, it would be possible to disclose the ordering of the neutrino mass eigenstates as shown in Figure 1.3.

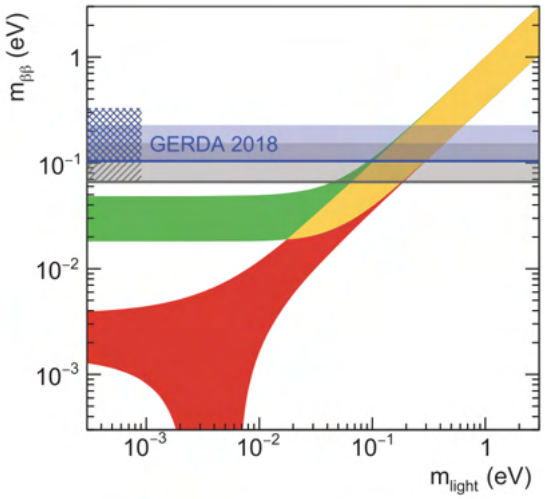


Figure 1.3: Parameter space for $m_{\beta\beta}$ as a function of the lightest neutrino mass m_l for both normal (red, $m_{\text{light}} = m_1$) and inverted (green, $m_{\text{light}} = m_3$) mass ordering, with the yellow region showing the overlap. The blue horizontal band shows the upper limits of $m_{\beta\beta}$ obtained by GERDA [16], with the grey band combining the upper limits from all leading experiments in the field. From Ref. [17]

1.2. Experimental $0\nu\beta\beta$ searches

For $0\nu\beta\beta$ experiments, one of the most important factors is the half life sensitivity. The following section will detail the calculation, and previous results of half-life sensitivity from other $0\nu\beta\beta$ experiments.

For a $0\nu\beta\beta$ experiment, the half-life is proportional to:

$$T_{1/2}^{0\nu} \propto \sqrt{\frac{M \cdot t}{b \cdot \Delta E}} \tag{5}$$

Where M is the active mass of the detector, t the run time of the experiment, b the background index, and ΔE the energy resolution of the detector. Therefore to increase the sensitivity of a $0\nu\beta\beta$ experiment, the active mass of the detector and the run time should be maximised, while minimising the background index, and utilising a detector with the lowest possible energy resolution. Where energy resolution of a $0\nu\beta\beta$ experiment is limited to the choice of material, but the active mass and the run time can be scaled, however this comes at a compromise of cost for the experiment. In addition to this, it is pivotal to decrease the background index of a $0\nu\beta\beta$ experiment from background events about the energy of $Q_{\beta\beta}$, whereby most of these contributions can come from ^{238}U and ^{232}Th decay in array components. Typical $0\nu\beta\beta$ experiments are setup such that the material of the detector used is also an isotope that can undergo neutrinoless double-beta decay, examples of these experiments are of the following:

Table 1.1: Results of other neutrinoless double-beta decay ($0\nu\beta\beta$) experiments on the lower limit of the neutrinoless double-beta decay half-life, with the corresponding upper limit of effective Majorana mass measurements (at 90% C.L.).

Experiment	$T_{1/2}^{0\nu}$ (yr)	$m_{\beta\beta}$ (eV)	Isotope	Detection Method
GERDA [16]	5.3×10^{25}	$0.15 - 0.33$	^{76}Ge	semiconductor detector
KamoLAND-Zen [18]	2.3×10^{26}	$0.036 - 0.156$	^{136}Xe	liquid scintillator
EXO-200 [19]	1.1×10^{25}	$0.2 - 0.4$	^{136}Xe	liquid TPC
MAJORANA [20]	2.7×10^{25}	$0.200 - 0.433$	^{76}Ge	semiconductor detector
NEMO-3 [21]	1.1×10^{24}	$0.33 - 0.62$	^{100}Mo	tracking

2. LARGE ENRICHED GERMANIUM EXPERIMENT FOR NEUTRINOLESS DOUBLE-BETA DECAY

2.1. Design of LEGEND

The Large Enriched Germanium Experiment for Neutrinoless double beta Decay (LEGEND) aims to probe neutrinoless double beta decay of ^{76}Ge using enriched high purity germanium (HPGe) detectors [22] bathed in liquid argon that is used as an active shield. LEGEND is split into two experiments; LEGEND-200 and LEGEND-1000, which will house 200 kg and 1000 kg of Germanium detectors respectively. This will initially start with LEGEND-200 that uses 70 kg of enriched detectors from GERDA and MAJORANA, with an additional 130 kg of the new ICPC detectors housed in the GERDA infrastructure at LNGS. LEGEND-200 will start data taking around early 2022 and will run for 5 years aiming to obtain a reduction in background by a factor of 2.5 with respect to GERDA, and to achieve a half-life sensitivity of $\sim 10^{27}$ yr ($m_{\beta\beta} \sim (30-70)$ meV) for neutrinoless double beta decay. LEGEND-1000 utilises five LEGEND-200 modules with further improvements and aims to achieve a half-life sensitivity of $\sim 10^{28}$ yr ($m_{\beta\beta} \sim (9-21)$ meV) and an additional 20-fold background reduction in 10 years of live time. Due to the $m_{\beta\beta}$ measuring sensitivity of LEGEND-1000, it would be possible to resolve the mass ordering for the neutrino mass eigenstates as shown in Figure 1.3.

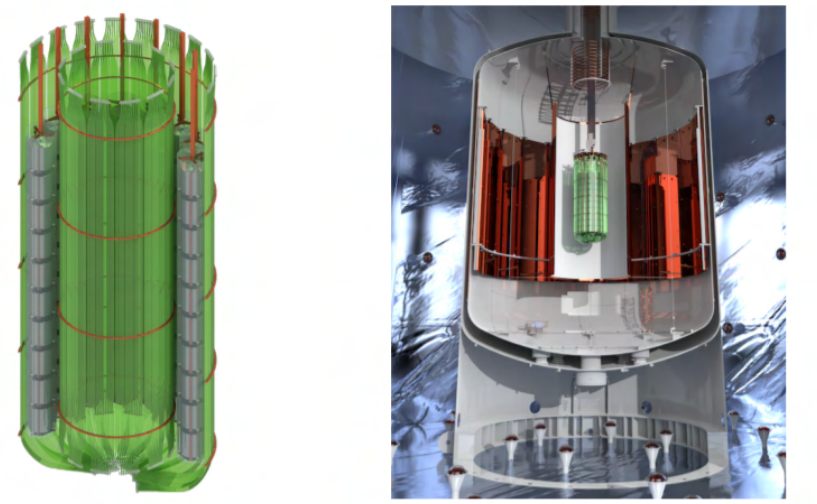


Figure 2.1: Left: LEGEND-200 germanium detectors mounted in strings and surrounded by optical fibers that are used to detect the LAr scintillation light. Right: Detector systems positioned in the center of a liquid argon (LAr) cryostat equipped with wavelength-shifting reflectors. The cryostat is placed in a water tank instrumented with photomultipliers and used as a Cherenkov muon detector. From Ref. [14]

2.2. Germanium detectors

Germanium detectors have been at the foundation of nuclear spectroscopy for more than half a century. Germanium (^{76}Ge) detectors are useful in detecting double-beta decay due to the nuclei having an even number of neutrons and protons, whereby single-beta decay is suppressed and double-beta decay can occur. Germanium detectors also exhibit a high energy resolution (0.1%), which is useful as examining energy peaks is the only viable method for the discovery of neutrinoless double-beta decay, and it results in an increased half-life sensitivity. In addition to this, germanium detectors also act as a time projection chamber, which utilises an electric field for particle trajectory reconstruction, which will aid in discriminating between neutrinoless double-beta decay signals and background signals later discussed in section 2.3. Moreover, the crystal growth process leads to extremely pure detectors, resulting in exceptionally low amounts of uranium and thorium contamination and detectors enriched with up to 90% germanium-76, which can then be converted into high purity germanium (HPGe) detectors. Furthermore, no known background sources originating from the germanium detectors produce an energy peak in the vicinity of $Q_{\beta\beta}$, adding no complications towards the event discrimination process. Germanium detectors have had a wide usage in previous neutrinoless double-beta decay experiments, most notably in both the GERDA [16] and MAJORANA [23] collaborations as shown in Figure 2.2.

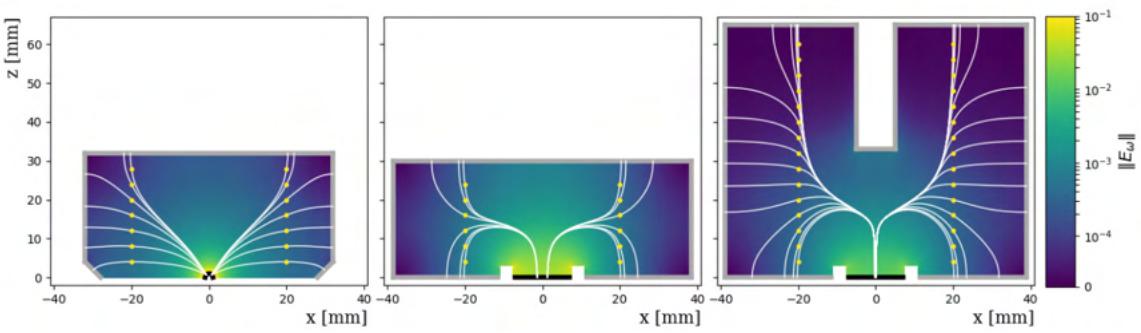


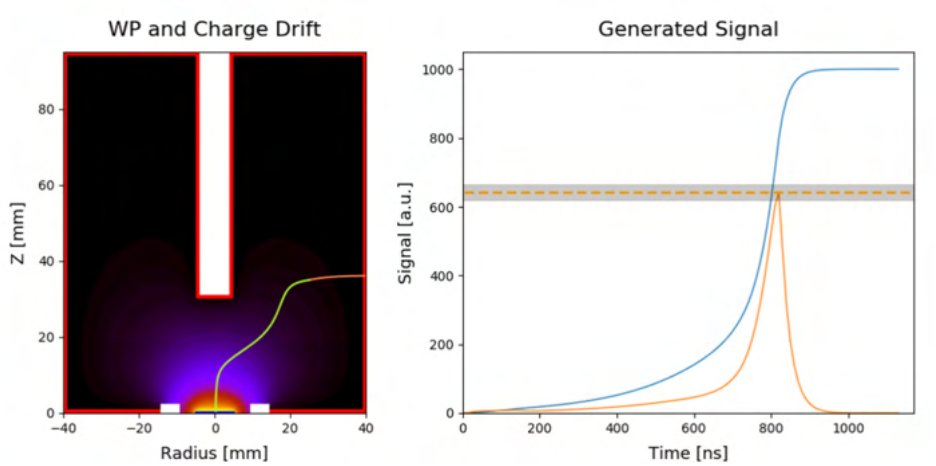
Figure 2.2: Detectors from the MAJORANA, GERDA and LEGEND collaborations (from left to right) respectively. The above plot shows the weighting field (E_ω) for the cross section of each of the respective detectors from each collaboration. The thick black and gray lines represent the p^+ and n^- electrodes respectively. The yellow points are example points of energy deposition, where the white lines show the trajectories of the holes and electrons to the p^+ and n^- electrodes respectively. From Ref. [24]

The detectors used in MAJORANA [23] and GERDA [16] (P-type point contact (PPC) [25] and broad energy germanium (BEGe) [26] detectors respectively) were limited to 1 kg

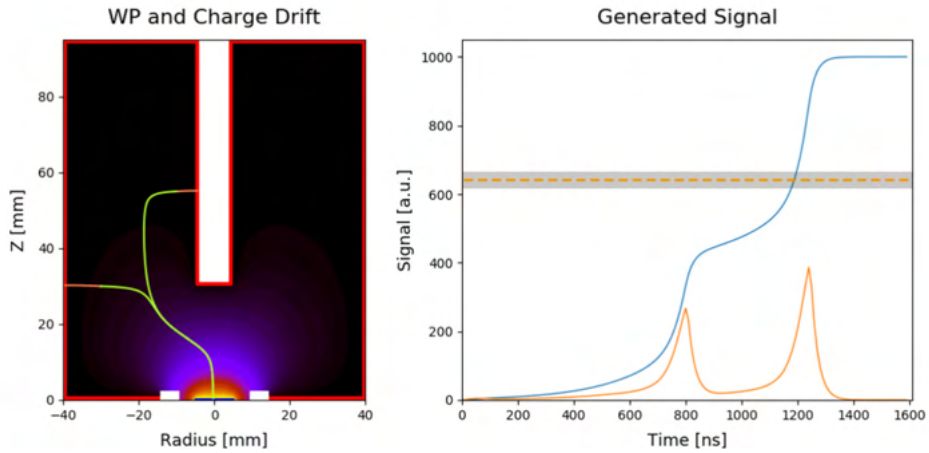
due to the electrode geometry. However, in an effort to increase the mass, research has led to new geometries such as the inverted coaxial point-contact (ICPC) detector [22] which is to be used in LEGEND. This increase in mass and size comes at a cost, which leads to an increase in time needed for the charge carriers to collect at each respective electrode, however by increasing the mass of each detector; therefore decreasing the overall number of detectors needed; the number of cables and detector support materials that provide background are all reduced. Moreover, there is an additional reduction in background from surface events due to the ICPC detectors having a smaller surface to volume ratio. The ^{76}Ge detectors used in GERDA and MAJORANA have achieved the lowest current background compared to previous $0\nu\beta\beta$ experiments, and has a quasi-background-free regime, whereby $2\nu\beta\beta$ is suppressed and a $0\nu\beta\beta$ decay signal will produce a single, sharp peak.

2.3. Detecting $0\nu\beta\beta$ decay

When a germanium-76 nuclei in the detector undergoes neutrinoless double-beta decay, it will emit a pair of electrons that will be absorbed, causing a point-like energy deposition that creates a large cluster of holes and electrons. As mentioned above, the detectors are ideally enriched with up to 90% germanium-76 such that the electrons are absorbed within the order of a few millimetres away from the initial neutrinoless double-beta decay event within the detector. The resultant holes and electrons will then drift along the electric field lines towards the respective p^+ and n^- electrodes along the surface of the germanium detector. The motion of these charge carriers will induce a current on the respective electrodes that can then be read as a signal using charge sensitive amplifiers [24]. Analysing the time structure of the read-out signal allows you to obtain information on the topology of the event, such as how many events occurred and where they happened along the detector. However, in order to access this information an accurate model of each individual detector would have to be made as the time structure of the read-out signal is dependant on the geometry of the detectors and the electrode scheme. It is important to note that surface events from external alpha and beta particles, and background gamma rays that penetrate the detector can also cause an energy deposition that will induce a signal. Therefore by utilising the topology of the signals, it is possible to differentiate between single-site event signals, such as a neutrinoless double-beta decay event, and multiple site event signals such as the Compton scattering of gamma rays in the germanium detector.



(a) A single-site event that acts as a proxy neutrinoless double-beta decay event. The single-site event contains a singular sharp current peak with a rapid accumulation of charge.



(b) A multi-site event that represents a background-like event such as Compton scattering of gamma rays. This produces a multiple peak signal that has lower values for the maximum current, A, compared to the single-site peak.

Figure 2.3: Showcasing Monte Carlo simulations of both single- and multi-site event types (from top to bottom respectively). The orange and blue lines in the graphs represent the current and the accumulation of charge respectively, and the dashed orange line is an example acceptance window. From Ref. [27].

This is achieved via a pulse shape analysis (PSA), where specific features of the read-out signals time structure allows for discrimination between neutrinoless double-beta decay signals and background-like signals. The type of pulse shape analysis used in LEGEND will be the A/E method [26], which was used in the GERDA collaboration. The A/E method uses the maximum value of the current signal (A) normalised by the deposited energy (E), where the deposited energy is proportional to the area under the signal. Single-site events will produce one peak with a maximum amplitude, A, as shown in Figure 2.3a. However, multi-site events will produce a signal that is a superposition of each of the smaller cluster of charge carriers; that arrive at different times; resulting in a multiple peak signal as shown in Figure 2.3b. This results in a lower A/E ratio for the multi-site events, which allows discrimination between single- and multi-site events.

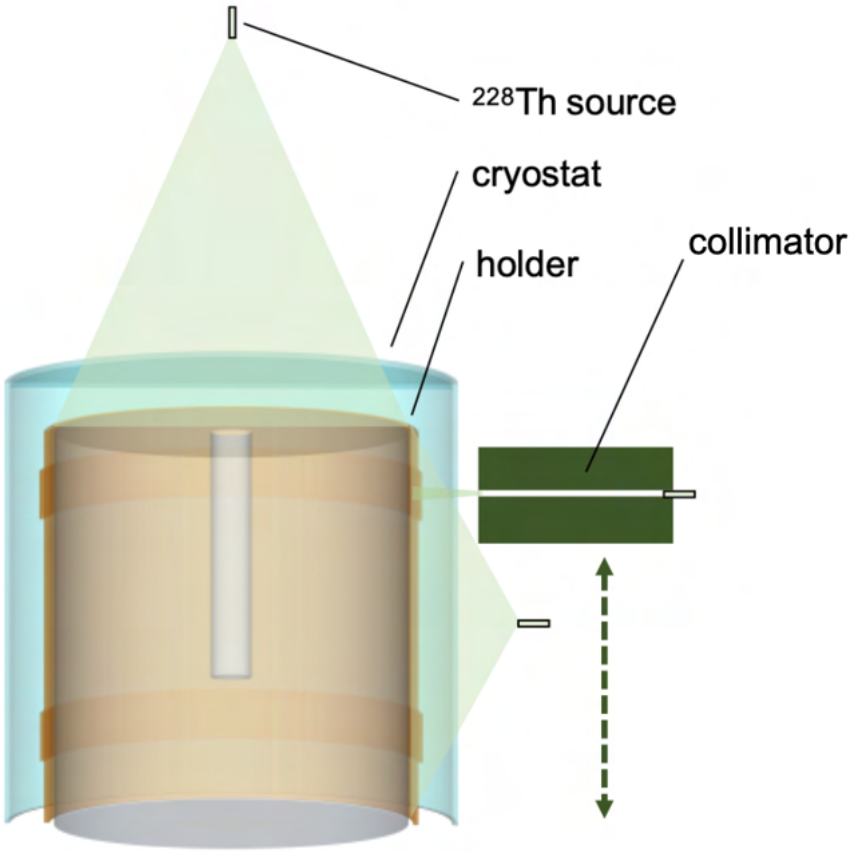


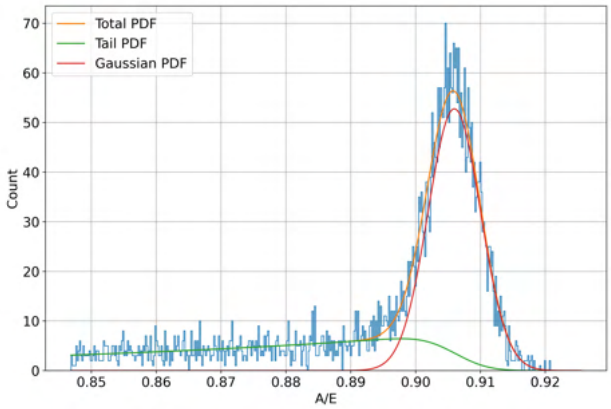
Figure 2.4: Configurations used for detector characterisation. Setup for Pulse shape discrimination studies with a flood top and side 13 kBq ^{228}Th source and with a collimated 250 kBq ^{228}Th source for lateral scans. The vacuum cryostat (cyan) and detector holder (orange), both made of aluminum, are added here for illustration. From [28].

In order to calibrate the event discrimination, a ^{228}Th source is utilised; used in both MAJORANA [29] and GERDA [30]; as it provides both single- and multi-site events within the detector. Where Figure 2.4 shows the three different types of scans: top, lateral, and collimator. The Thorium-228 atom decays via a series of α and β^- decays to Tl-208. The daughter nuclei in this decay chain emits a wide range of gamma and x-rays with energies ranging from 100 keV to 2.6 MeV that can be used for calibration. However, it is the 2615 keV gamma ray line from Tl-208 that becomes useful in replicating events around $Q_{\beta\beta} = 2039keV$ for $^{76}\mathbf{Ge}detectors$. The emitted gamma rays from Tl-208 can undergo pair production in the detector, resulting in an electron-positron pair. When the positron annihilates, it will emit two 511 keV photons. If both of these photons are absorbed, the energies sum up to that of the electron, resulting in a full energy peak (FEP) at 2.6 MeV. If one photon is absorbed and the other escapes, its results in a $(2.6 - 0.511)$ MeV peak, classified as a single escape peak (SEP). However, if none of the photons are absorbed then there is a resultant peak at 1.592 MeV which consists of an electron-positron single-site event that resembles the physics of neutrinoless double-beta decay; this is known as a double escape peak (DEP) [24].

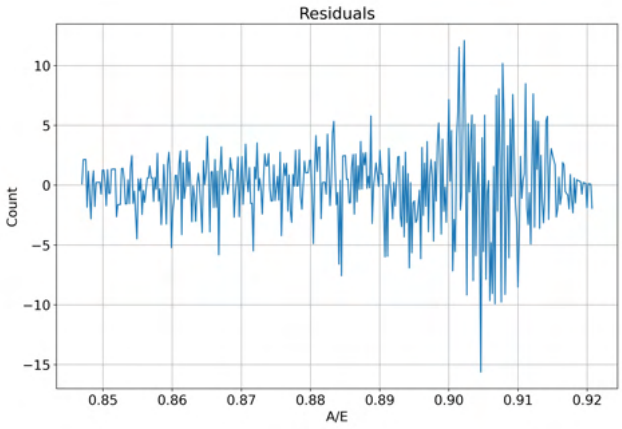
3. PULSE SHAPE ANALYSIS

3.1. A/E analysis

The following section will detail the A/E analysis performed on detector 'V04199A'. The code that contains the backbone of the A/E analysis was provided by George Marshall from the UCL LEGEND group, shown in Section B1. The below plot shows the overall distribution of the A/E values for the energy range of a energy band from (1085 – 1115) keV.



(a) Unbinned histogram of A/E fit for (1085 – 1115) keV energy band.



(b) Unbinned histogram of A/E residual fit for (1085 – 1115) keV energy band.

Figure 3.1: A/E fit for (1085 – 1115) keV energy band. Where the A/E distribution is modelled by a total PDF formed by a Gaussian PDF and a tail PDF representing single-site and multi-site events respectively.

Where in Figure 3.1 the total PDF is split into two separate distributions representing the respective single-site and multi-site events.

The A/E analysis will determine a minimum A/E cut to remove multi-site events represented by the tail PDF for the entire energy spectra shown in Figure 3.2 while also retaining a minimum of 90% of the DEP single-site events.

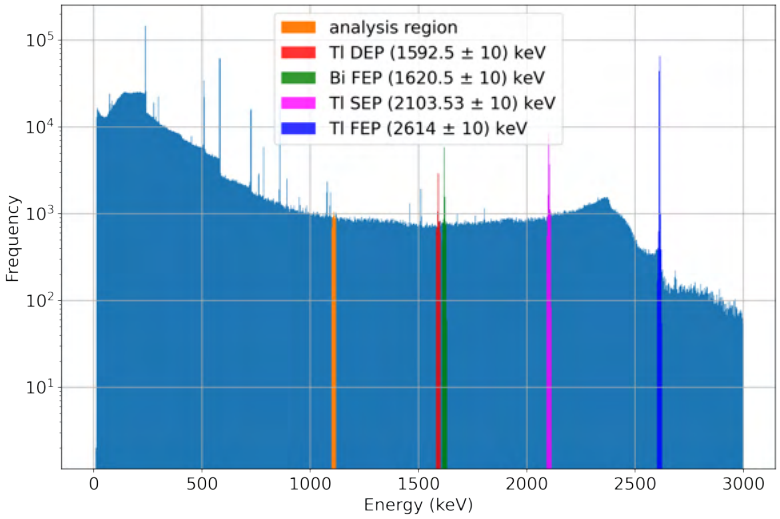


Figure 3.2: Shows energy spectra of ^{228}Th calibration data for detector 'V04199A' with the analysis energy band used to form Figures 3.1a and 3.1b. In addition showing regions of interest in order to determine the success for the A/E analysis.

Where Figure 3.2 also showcases the regions of interest which contain multiple DEP, SEP, and FEP sites. We would expect the A/E analysis to retain most DEP single-site events that resemble neutrinoless double-beta decay, while the other SEP and FEP regions to be eliminated as they contain a majority of multi-site events that resemble background.

However, there are added complications to this as the average A/E value is not constant for varying energy bands as shown in Figure 3.3.

Where it can be seen in Figure 3.3 that the median A/E decreases with an increase in energy. This relationship can then be plotted and fit to show the exact relation between the median A/E with the energy as shown in Figure 3.4, and this can then be used to build a classifier.

Where a linear model of the following:

$$y_{AoE} = aE + b \tag{6}$$

is used to model the A/E distribution medians against energy from Figure 3.4a. Where E is the energy deposition of the events, y_{AoE} the A/E distribution median values, and

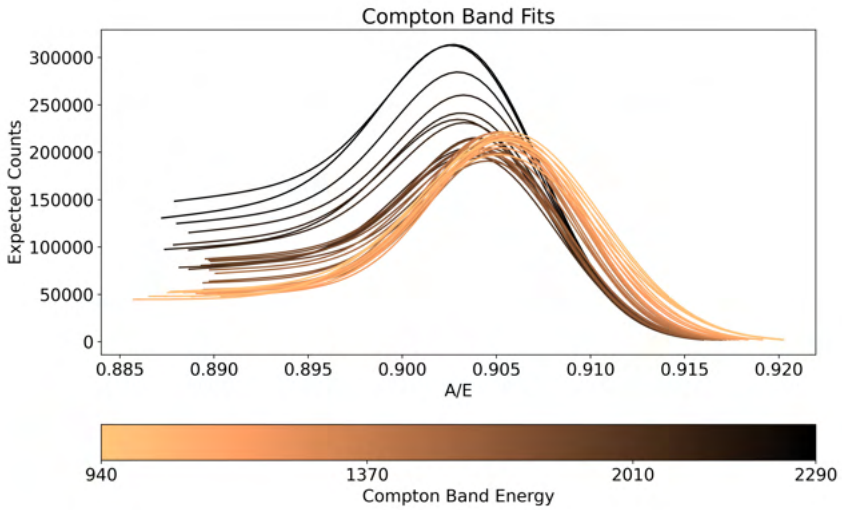


Figure 3.3: A/E distributions for range of 20 keV energy bands from 940 - 2290 keV.

a, b are constants which were calculated to be equal to -1.4×10^{-6} and 0.9 respectively for detector 'V04199A'. A root mean square fit is also applied to the A/E distribution standard deviation against energy from Figure 3.4b, with the following form:

$$\sigma_{AoE} = \sqrt{f + \frac{d}{E^c}} \tag{7}$$

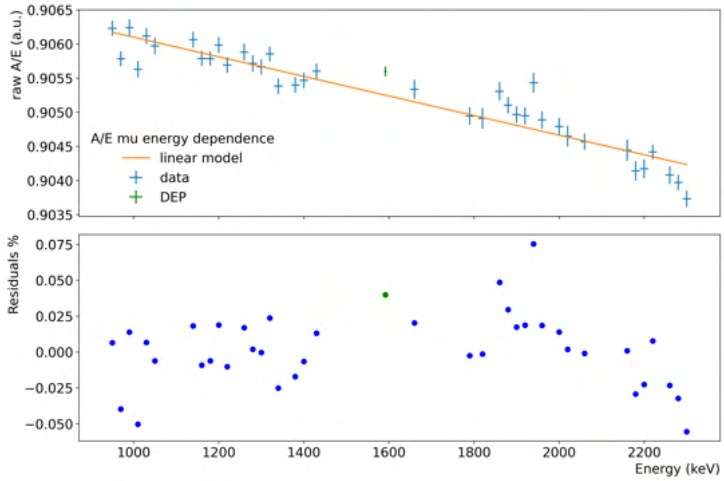
Where σ_{AoE} is the A/E distribution standard deviation values and c, d, and f constants to be determined, where values of $c = 7.4$, $d = 196.4$, and $f = 1.5 \times 10^{-5}$ were calculated. In GERDA a constant value of $c = 2$ was used, however this value does not suitably model all detectors for LEGEND. Both of the applied fits and parameters were calculated using pythons curve_fit program.

The above fits and parameters are then used to build a classifier of the form:

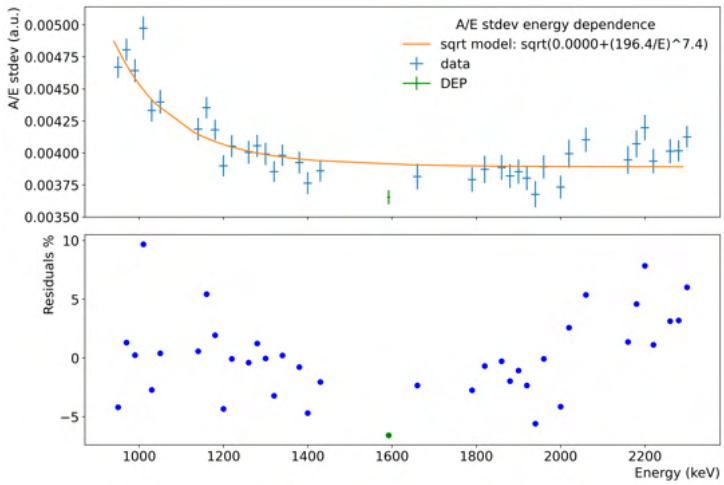
$$(A/E)' = \frac{\frac{y_{AoE}}{aE+b} - 1}{\sigma_{AoE}} \tag{8}$$

Where $(A/E)'$ are the classifier A/E values. The classifier centres the raw A/E values to the fitted median from Figure 3.4a, which is equal to a value of $(A/E)' = 0$. This is normalised by the standard deviation for a range of energies. This can be plotted as a 2D histogram as shown in Figure 3.5.

Most single-site events can be identified as the densely populated region about $(A/E)' = 0$ as this represents the median raw A/E values represented in the distribution from Figure 3.3. The multi-site regions would mostly be located at lower raw A/E



(a) A/E distribution medians against energy. A linear fit is applied to determine the A/E energy dependence with values of $a = -1.4 \times 10^{-6}$ and $b = 0.9$ from Equation 6.



(b) A/E distribution standard deviation against energy. A root mean square fit is applied to determine A/E standard deviation energy dependence with values of $c = 7.4$, $d = 196.4$, and $f = 1.5 \times 10^{-5}$ from Equation 7.

Figure 3.4: Raw A/E distribution mean and standard deviation against energy, with linear and root mean square fits respectively.

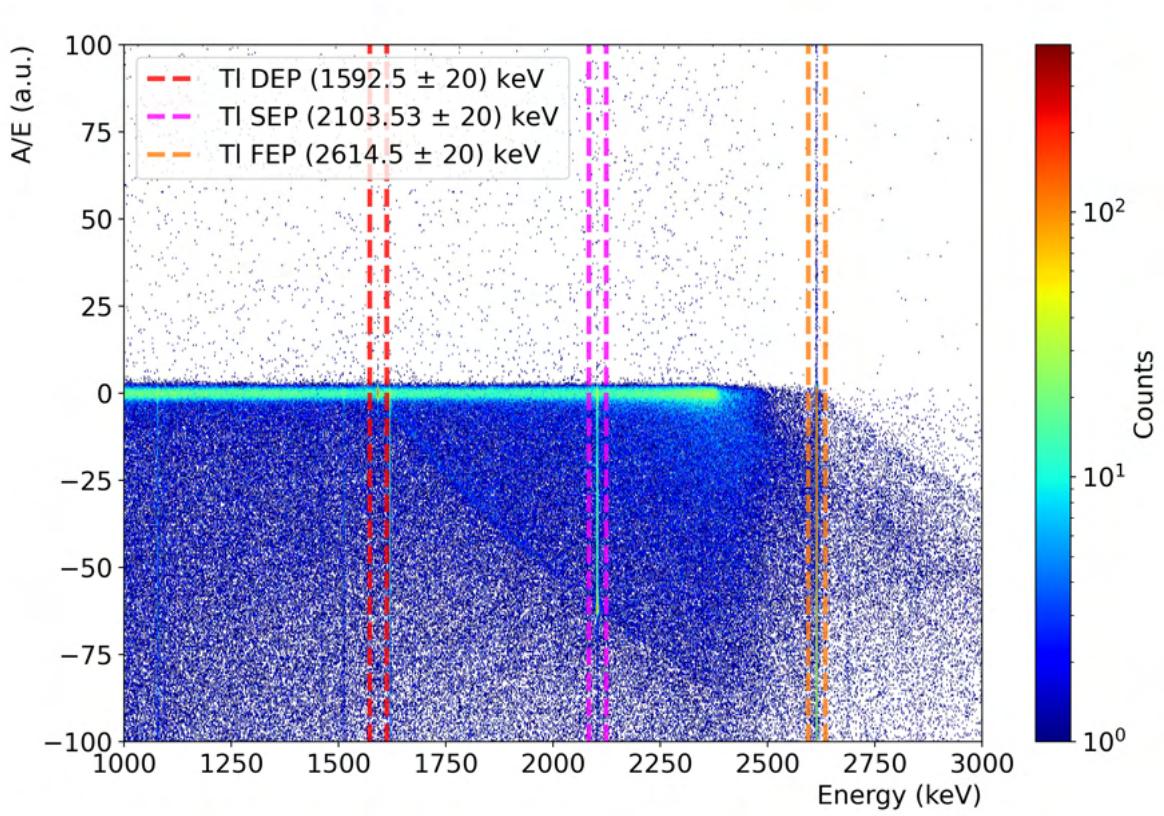


Figure 3.5: 2D Histogram of classifier A/E-Energy plot. The red, magenta, and orange lines identify the Tl-208 DEP, SEP, and FEP regions respectively. The A/E on the y-axis here refers to the classifier $(A/E)'$. Referenced in Equation 3.5.

values, which are represented by $(A/E)' < 0$. The above plot shows that the Tl DEP is mostly dominated by single-site events as there is an extremely dense region centered about $(A/E)' = 0$. The region above $(A/E)' = 0$ mostly consists of events near the p^+ contact in the detectors, where additional effects from the electrons flowing away from the contact increase the maximum current of the induced input signal on the charge sensitive amplifiers opposite the p^+ electrode, resulting in a higher A/E value. This effect is dominant in events that occur close to the p^+ electrode.

After the classifier has been built, the A/E cut can be applied. The A/E cut value is defined to be the smallest A/E value for which at least 90% of the DEP events about (1592 ± 10) keV survive. There is also and additional high sided cut defined to be one sigma, this will cut events with the highest A/E values which will be dominated by p^+ contact events. The aforementioned lower A/E cut is obtained by sweeping through A/E values within the range $-5 < (A/E)' < -0.5$ in slices of 0.1 as shown in Figure 3.6.

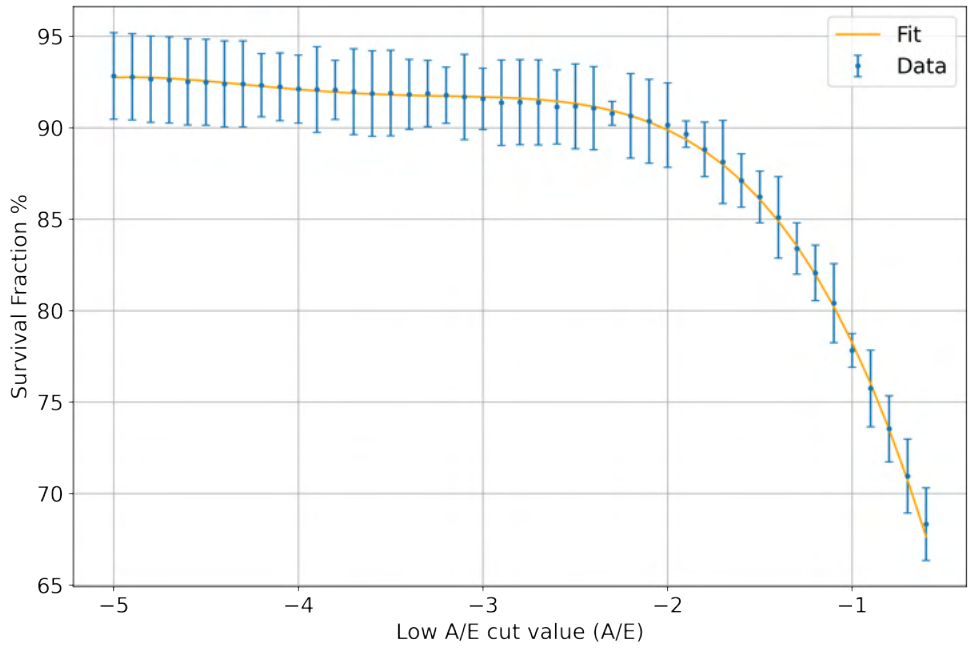


Figure 3.6: Survival fraction of DEP events about (1592 ± 10) keV against classifier A/E value. The orange line is a fourth order polynomial fit to the survival fraction data. The A/E on the y-axis here refers to the classifier $(A/E)'$. Referenced in Equation 3.5.

A low A/E cut value can then be calculated using the fourth order polynomial fit, where a value of $(A/E)' = -2.03$ is defined. The low A/E cut can then be applied to the entire energy spectra and the survival fractions for points of interest can be calculated.

Where the survival fractions with energy bands of 20 keV width about some regions of interest are:

- Tl DEP (1592.5 ± 10 keV): (90.9 ± 0.9)%
- Bi FEP (1620.5 ± 10 keV): (8.0 ± 0.2)%
- Tl SEP (2103.53 ± 10 keV): (5.6 ± 0.1)%
- Tl FEP (2614 ± 10 keV): (6.5 ± 0.1)%

The survival fractions for the regions of interest above achieved from detector 'V04199A' are then deemed to be acceptable as a majority of FEP and SEP events from both Tl and Bi are cut due to the dominant multi-site nature, while the DEP retains a majority of events due to the dominant single-site nature.

After the cuts have been performed, the classifier A/E-energy plot (Figure 3.5) can be re-plotted:

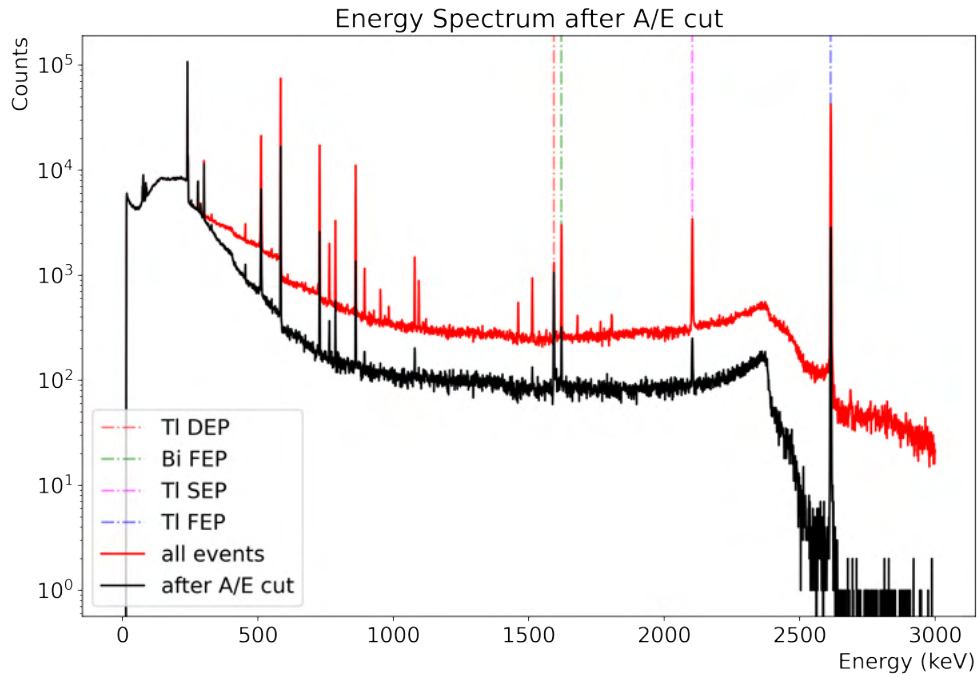


Figure 3.7: Number of events throughout the energy spectrum pre- and post-A/E cut, highlighting the FEP, SEP, and DEP regions of interest.

Where as well as an elimination of a majority of events from peaks with dominant multi-site nature, there is also a strong removal of background events across the Compton Continuum that result in a stronger and sharper double escape peak surrounded by

lower background, allowing an increased discovery potential for single-site events such as neutrinoless double-beta decay.

3.2. Batch-5 detectors

The bulk of this report will now focus on the application of the A/E algorithm to other detectors to be used in LEGEND-200, and to optimise and improve the A/E algorithm by testing new ideas. The three detectors that are to be focused on are the 'V05612A', 'V05612B', and 'V05267A' batch-5 detectors. The number of events used from the ^{228}Th calibration data for detectors V05612A, V05612B, V05267A, and V04199A are $3,169,528$, $2,783,396$, $3,010,272$, and $3,778,516$ respectively. Where the lateral scan files used can be located on the code in Appendix B.

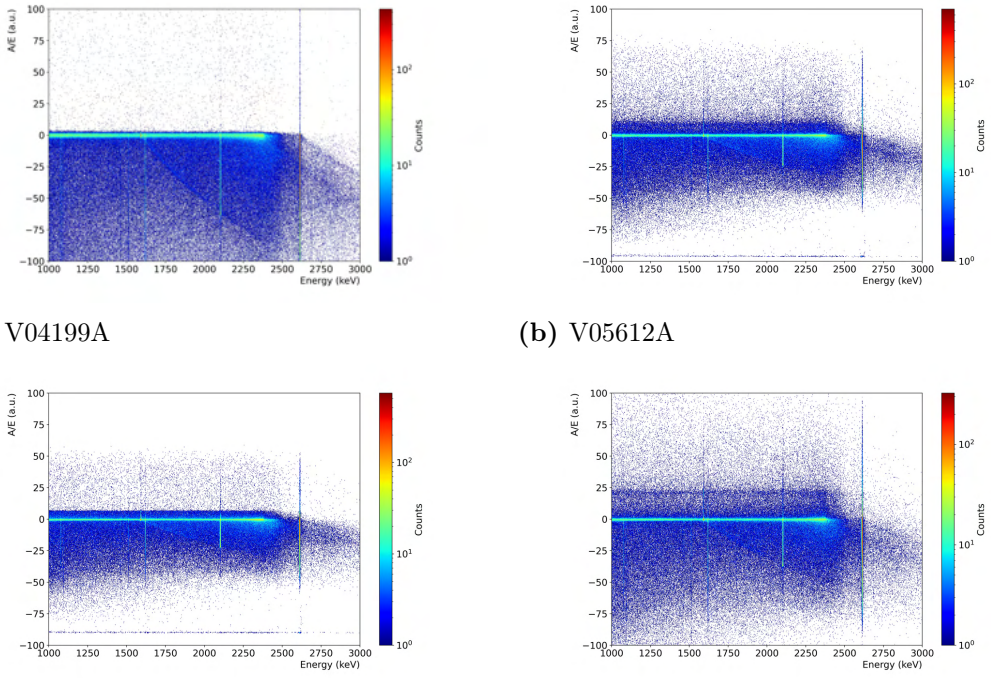
Modelling the median A/E-energy dependence for both the raw A/E and standard deviation allows you to obtain the following parameters for the detectors:

Table 3.1: Table of parameters used in modelling the median A/E-energy dependence for both the raw A/E and standard deviation of the A/E distributions. Corresponding plots can be found in Figures A1.1 to A1.3, A2.1 to A2.3, and A3.1 to A3.3 for detectors V05612A, V05612B, and V05267A respectively.

	Detectors			
Parameters	V05612A	V05612B	V05267A	V04199A
a	-4.0×10^{-6}	-3.3×10^{-6}	-1.9×10^{-6}	-1.4×10^{-6}
b	0.7	0.9	0.7	0.9
c	3.0	74.5	3.3	7.4
d	10.0	828.8	0.6	196.4
f	10.7×10^{-5}	12.3×10^{-5}	4.4×10^{-5}	1.5×10^{-5}

The raw A/E parameters for the detectors are performing normally with similar values that model the data adequately, where the highest percentage residuals for V05612A, V05612B, and V05267A are in the order of 0.159%, 0.193%, and 0.177% respectively. These are more than double the percentage residual values calculated for detector V04199A, where the highest is 0.075%, this will lead to a poorer representation of the median A/E in the classifier plot. The median A/E-energy dependence also differs between the detectors, with detector V05612A having more than double the linear dependence in energy to V04199A.

However, there are big inconsistencies and problems in the modelling of the standard deviation for the A/E distributions, which can be seen in Figures A1.3b, A2.3b, and A3.3b. The square root model poorly models the standard deviation from the A/E distributions. This is especially evident when comparing the model plots to that of



(a) V04199A **(b)** V05612A

(c) V05612B **(d)** V05267A

Figure 3.8: 2D Histogram of classifier A/E-Energy plot for each of the analysed detectors.

detector V04199A. In addition to this the uncertainties are much bigger with residual percentages of up to 36%, 29%, and 46% for detectors V05612A, V05612B, and V05267A respectively. The percentage residuals for detector V05612B; lowest average percentage residual batch-5 detector; are on average 3.58 times larger than the percentage residuals obtained for detector V04199A, meaning the root square mean fit is not properly and effectively representing the data.

After determining the modelling parameters for the detectors, the classifiers can be built and plotted.

Where it can be seen that the low $(A/E)'$ distribution is more condensed for the batch-5 detectors. This will lead to a lower percentage of multi-site events that dominate the negative $(A/E)'$ region being removed. In addition it can be seen that the newly analysed detectors contain a more dense region of A/E values for $(A/E)' > 0$, which will lead to more p^+ contact events, where the current is increased due to charge carrier effects.

Once the classifier plots have been completed, the survival fractions for the regions of interest can be calculated for the detectors.

While detector V05612A retains the most DEP events, the survival fractions for the

Table 3.2: Survival fractions of regions of interest for detectors V05612A, V05612B, and V05267A from the A/E analysis, compared to detector V04199A which was initially analysed at the start of the A/E section.

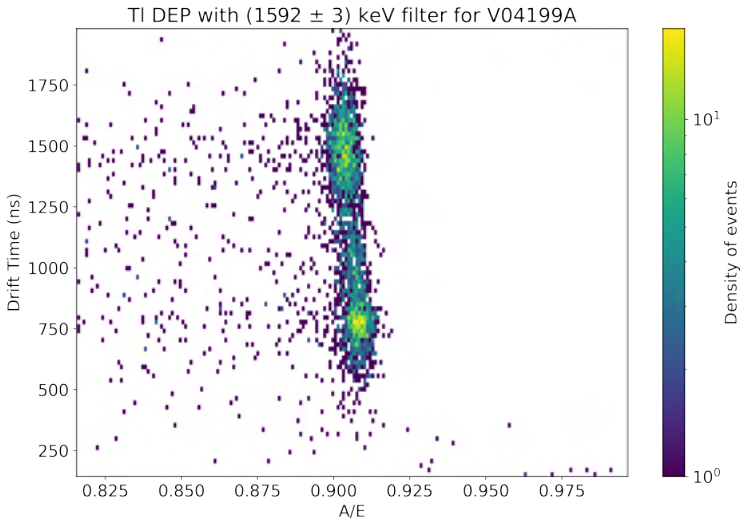
	Detectors			
Regions of interest	V05612A	V05612B	V05267A	V04199A
Tl DEP (1592.5 ± 10 keV)	(93.6 ± 3.9)%	(84.2 ± 5.9)%	(89.9 ± 3.7)%	(90.9 ± 0.9)%
Bi FEP (1620.5 ± 10 keV)	(17.3 ± 0.8)%	(11.6 ± 1.0)%	(15.2 ± 1.1)%	(8.0 ± 0.2)%
Tl SEP (2103.53 ± 10 keV)	(13.8 ± 0.6)%	(9.2 ± 1.0)%	(14.2 ± 0.6)%	(5.6 ± 0.1)%
Tl FEP (2614 ± 10 keV)	(17.4 ± 0.2)%	(12.9 ± 0.2)%	(16.1 ± 0.2)%	(6.5 ± 0.1)%

SEP and FEP events on all the batch-5 detectors are on average 1.83 times the value of that for detector V04199A. This shows that the A/E analysis is not as effective due to the poor classifier modelling. This could be due to an excess of p^+ contact events where the current in the A/E is measured at a higher value due to charge carrier effects.

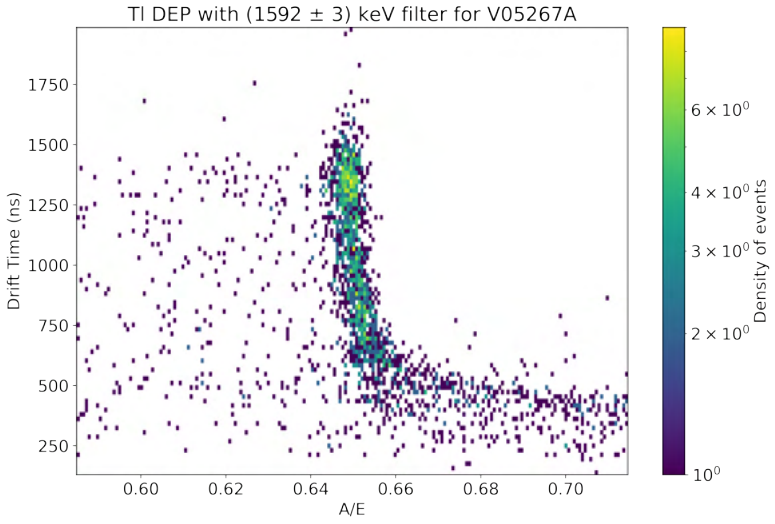
All the above data suggests that additional calibration would be needed, or that the A/E analysis needs to be improved or optimised to effectively eliminate background-like events for the detectors used in LEGEND.

4. DRIFT TIME ANALYSIS

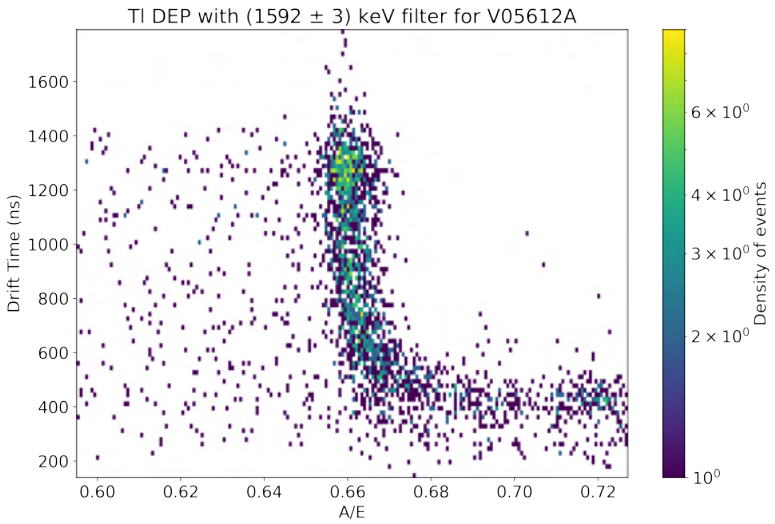
In January 2022 the acknowledgement of a A/E - drift time dependence was presented by George Marshall [31]. This combined with the above data in the A/E section that suggests an increase in the amount of p^+ contact events resulting in a higher A/E value, suggests that a drift time analysis should be performed on the detectors. The drift time analysis performed will be centred around the DEP (Figures A1.7, A2.7. and A3.7) as to make an effort to improve the low A/E cut that is determined by the survival fraction of the A/E events as seen in Figure 3.6.



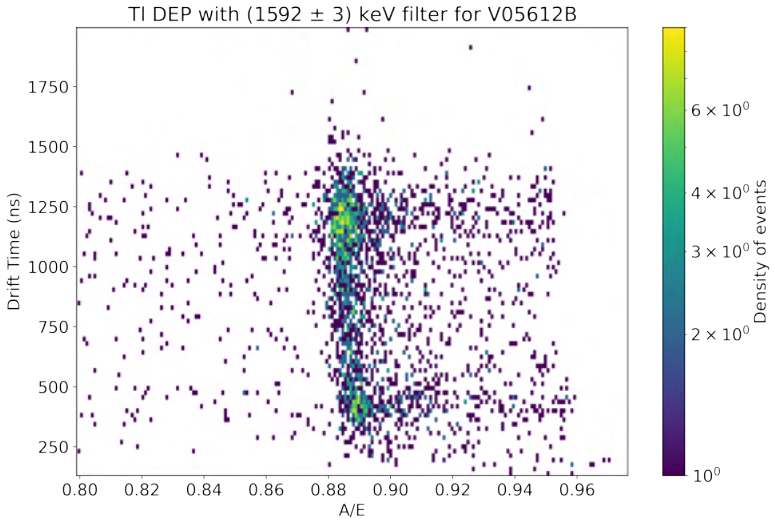
(a) V04199A



(b) V05267A



(c) V05612A



(d) V05612B

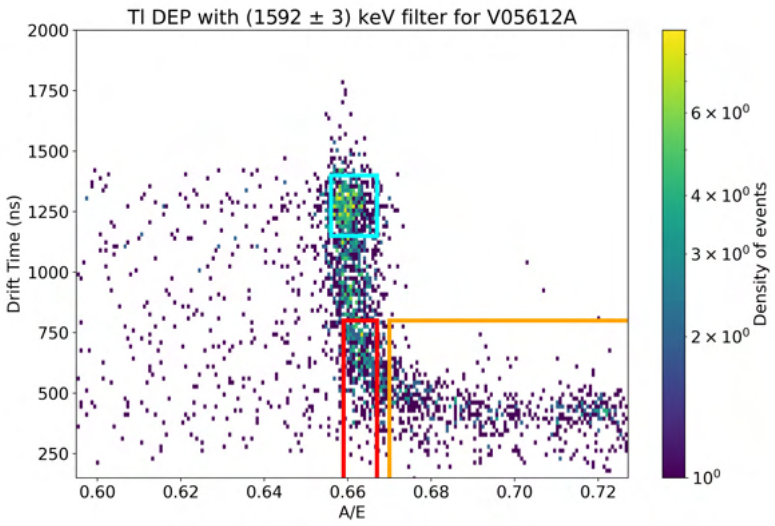
Figure 4.1: 2D Histogram of A/E against drift time with a colour bar representing the density of events per bin. The above plot is centered at the Tl DEP around (1592 ± 3) keV.

Where it can be seen in Figure 4.1a that there is a dense and narrowly spread A/E region for all drift times in the selected energy region around the Tl DEP of (1592 ± 3) keV, with a small number of events lagging in A/E value. The batch-5 detectors all appear to have low drift time tails following the dense region events, with detector V05612B (Figure 4.1d) containing two denser tails following both regions of high A/E density. This behaviour matches the classifier plots in Figure 3.8, whereby the batch-5 detectors contained a high number of inflated A/E events in comparison to detector V04199A.

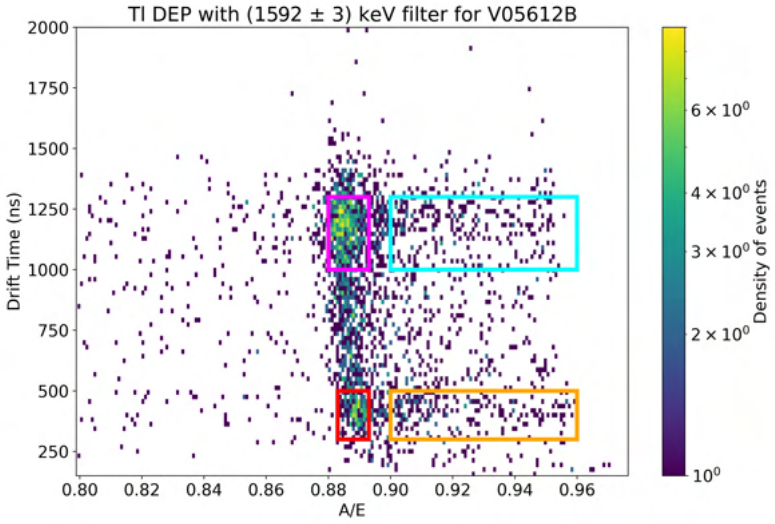
4.1. Waveform analysis

As these inflated A/E events are a product of the *p+* contact events due to charge carrier effects of the electrons drifting away increasing the measured current, one could analyse the waveform of the calibration events to see if there are any key differences in shape that could be corrected or allow us to distinguish between the inflated and non-inflated A/E events.

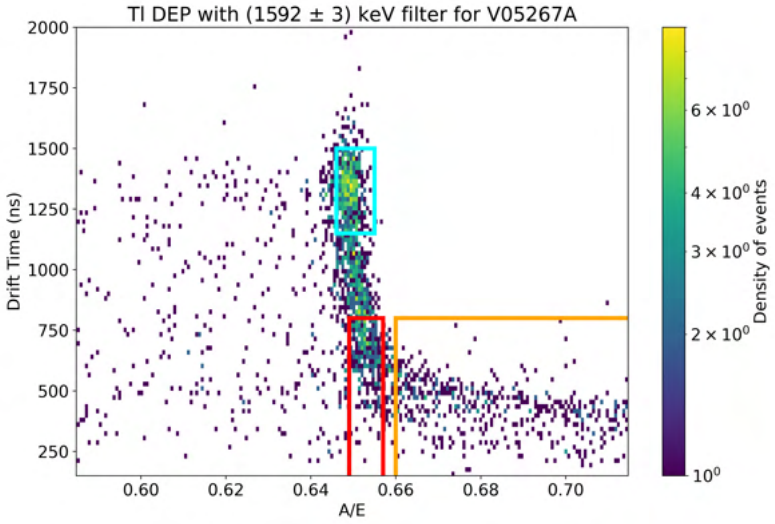
This will begin by analysing the corresponding regions shown in Figure 4.2 for the batch-5 detectors.



(a) Detector V05612A: red region: 100ns<dt<800ns and 0.659<A/E<0.667, orange region: 100ns<dt<800ns and 0.67<A/E<0.68, and cyan region: 1150ns<dt<1400ns and 0.656<A/E<0.667



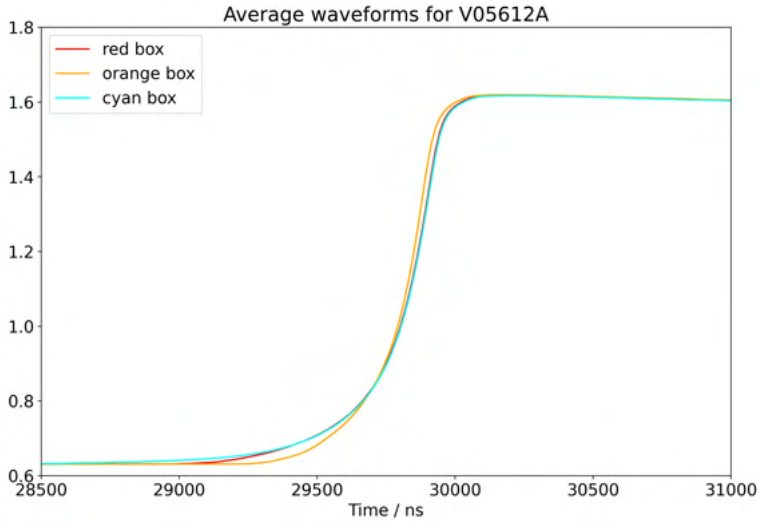
(b) Detector V05612B: red region: 300ns<dt<500ns and 0.883<A/E<0.893, orange region: 300ns<dt<500ns and 0.9<A/E<0.96, cyan region: 1000ns<dt<1300ns and 0.9<A/E<0.96, and magenta region: 1000ns<dt<1300ns and 0.88<A/E<0.893



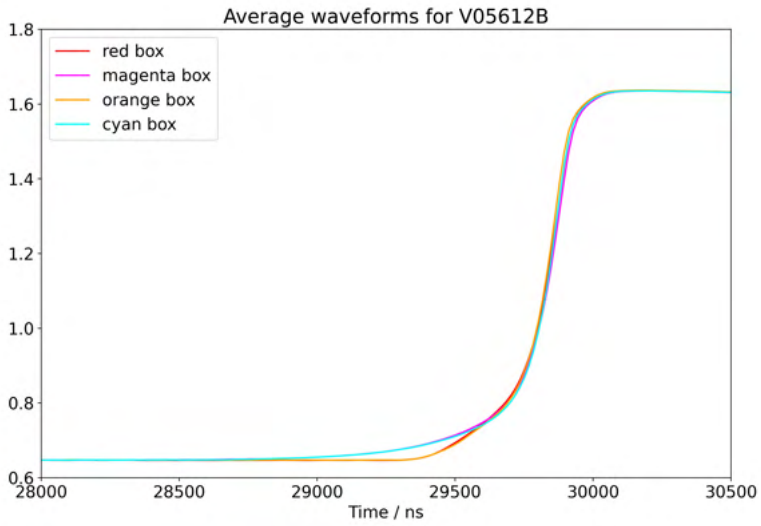
(c) Detector V05267A: red region: 100ns<dt<800ns and 0.649<A/E<0.657, orange region: 100ns<dt<800ns and 0.66<A/E<0.8, and cyan region: 1150ns<dt<1500ns and 0.646<A/E<0.655

Figure 4.2: 2D Histogram of A/E against drift time (dt) with a colour bar representing the density of events per bin. This plot includes analyses points for regions of high event density with a small spread of A/E values, and for tail regions with inflated A/E values.

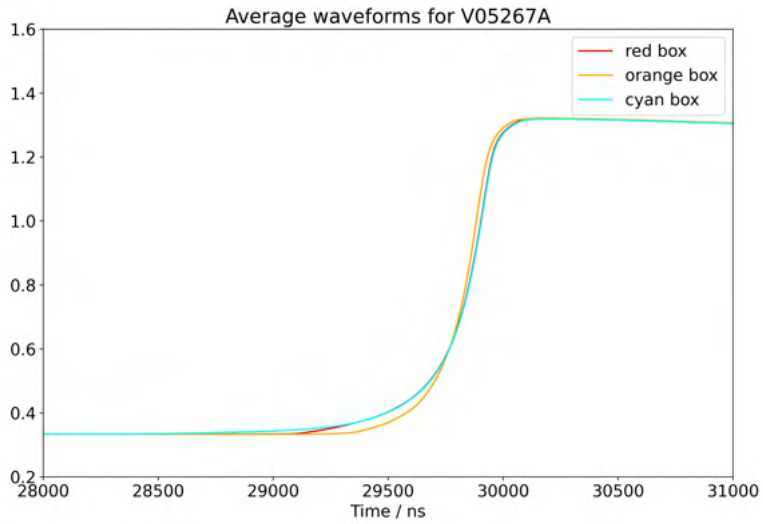
Where the selected analyses regions were constrained to only contain a majority of either inflated A/E events along the tail, or high density event regions with a small spread in A/E. Once the analyses regions have been selected, an average waveform can be plotted for each region. In addition to this, 10 randomly selected waveform events can be plotted onto each graph for one analysis box to inspect the variance of the waveform shape.



(a) V05612A



(b) V05612B



(c) V05267A

Figure 4.3: Average shape of collated waveform in each respective analyses region. The y-axis is a unitless number that corresponds to the shape of the waveform over time.

Figures A1.8 to A1.10, A2.8 to A2.11, and A3.8 to A3.10 for each respective batch-5 detector shows that the orange and red analysis box waveforms feature a kink about the waveform start time. Where the kinks in the orange regions that correspond to the inflated A/E tail, are highly exaggerated. This kink behaviour that shows in the plotted waveforms is a general behaviour, hence the orange region shown in Figure 4.3 having a distinct waveform shape in comparison to the other comparison regions. However the kink behaviour shown in detector V05612B is not as strong compared to the other two batch-5 detectors. This in addition with the fact that both the inflated A/E tail region and high density regions have waveforms that almost imitate each other for both high and low drift time could make it difficult in differentiating between the two. This behaviour is visible in Figure 4.3b, where the high and low drift time waveforms have a slightly different shape, which doesn't allow us to analyse and differentiate between the tail (cyan and orange for detector V05612B) and high density (red and magenta for detector V05612B) regions.

However, the reason for the visible kinks in detectors V05612A and V05267A could be due to a specific property of the batch-5 detectors. The batch-5 detectors have a larger p^+ electrode which could allow for unusual behaviour compared to the batch-4 detectors. In addition work done by Seemab Haider from the UCL LEGEND group could explain the kink feature observed:

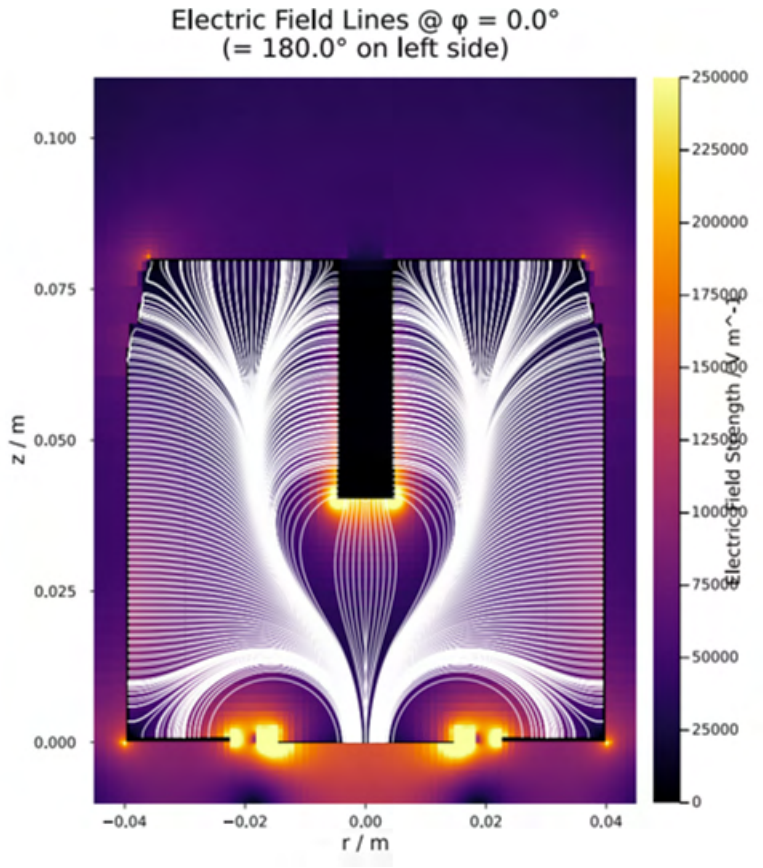


Figure 4.4: Electric field lines for detector V05267A throughout the detector. The colour bar indicates the strength of the electric field within the detector. The above figure was plotted by Seemab Haider from the UCL LEGEND group.

Where it can be seen that the motion of the holes near the borehole of the detector do not converge with the other charge carrier points about the centre of the p^+ electrode. This could lead to different waveform shape which could be what is being observed in Figures A1.8, A2.8, and A3.8. However, a simulation of these borehole events in the Germanium detectors used in LEGEND would be needed to confirm this. In addition to this the behaviour could be due to the p^+ contact events discussed before, where the electrons close to the electrode would induce an additional current, leading to the kink feature being observed.

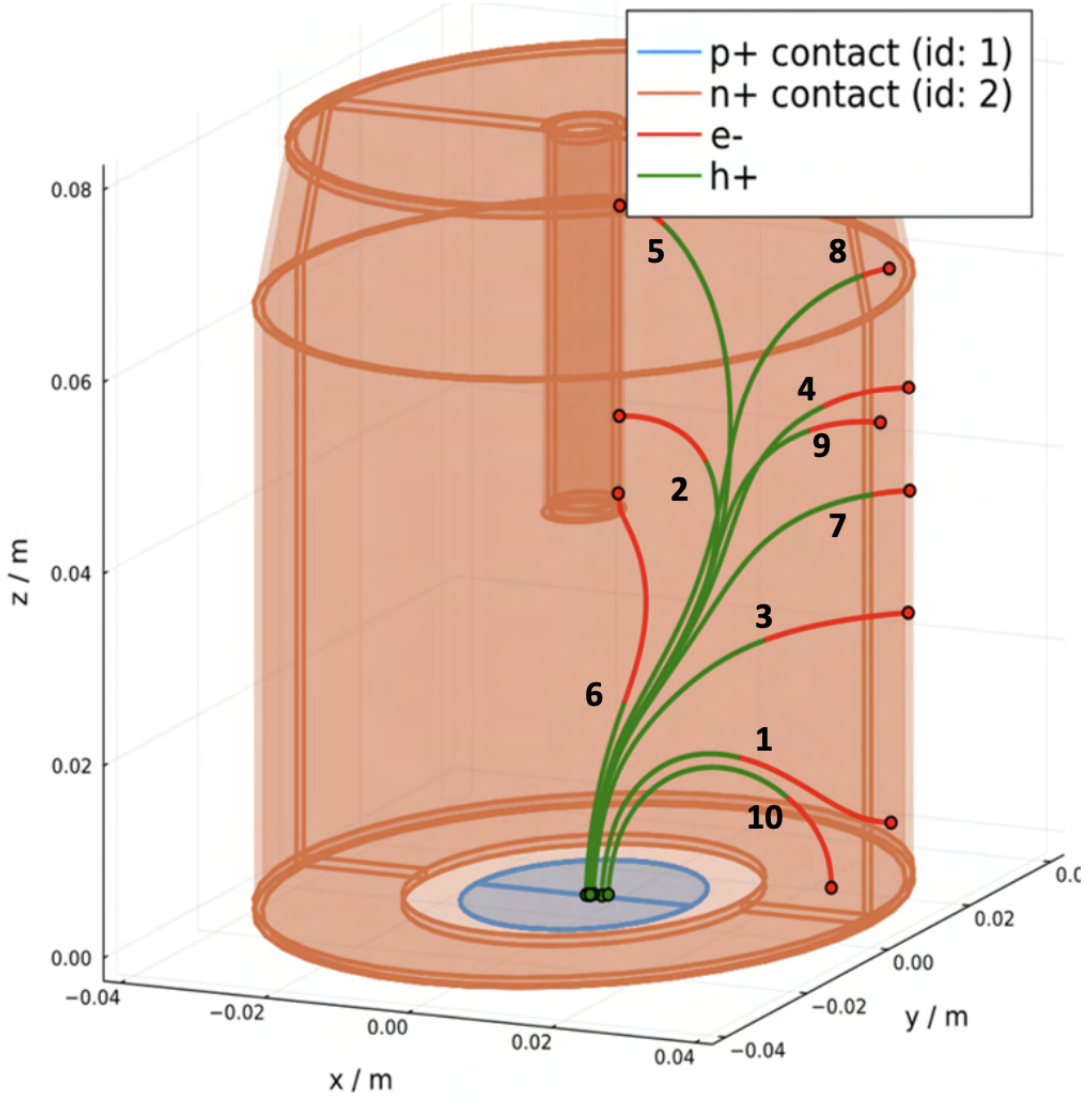
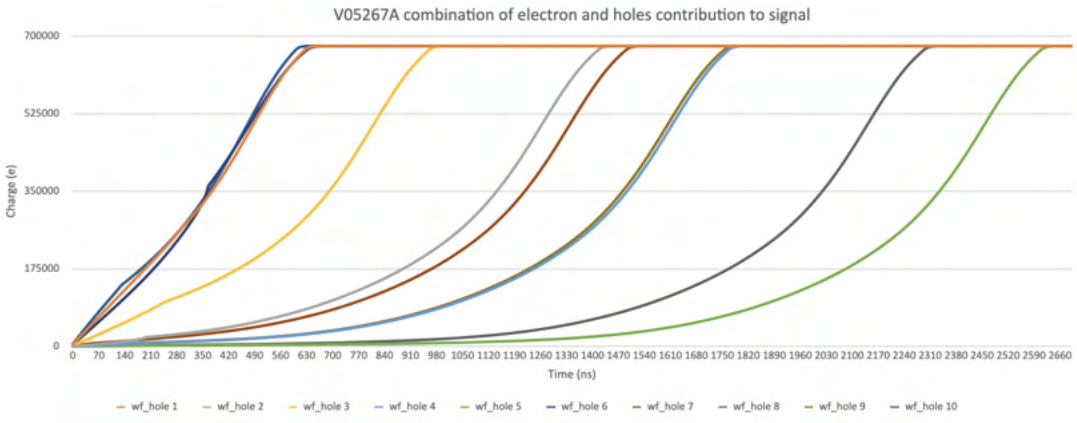
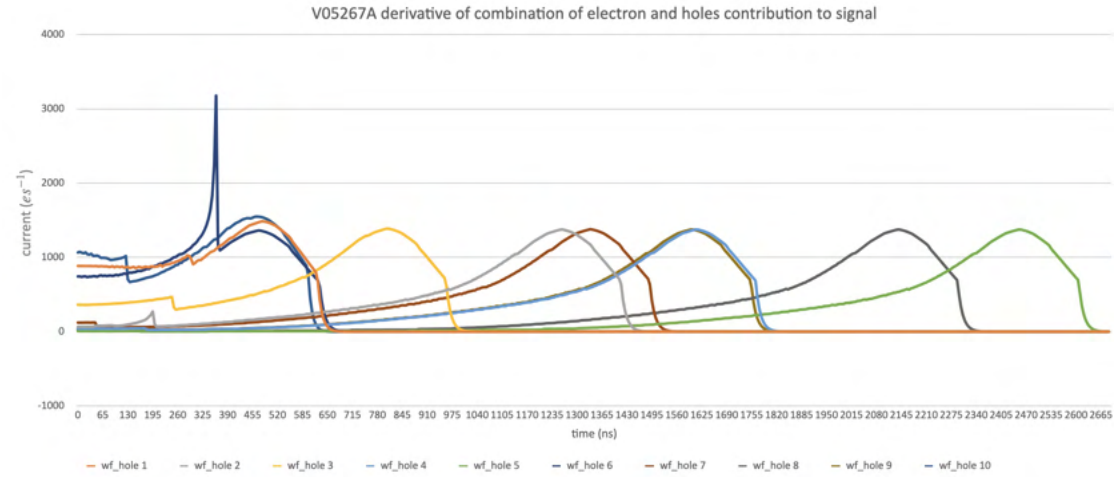


Figure 4.5: Simulated single-site events within detector V05267A with the motion of charge carriers (electrons and holes) to their respective electrodes (n^- and p^+). The above figure was plotted by Seemab Haider from the UCL LEGEND group.



(a) Charge against time for simulated single-site events in V05267A. These charge waveforms are the same type of waveforms averaged in Figure 4.3c.



(b) Current against time for simulated single-site events in V05267A.

Figure 4.6: Simulated single-site events in detector V05267A with charge and current against time plots, whereby the waveforms produced are as a result of both hole and electron contributions to the signal. The label wf_hole corresponds to the events labelled on Figure 4.5. The above figures were plotted by Seemab Haider from the UCL LEGEND group.

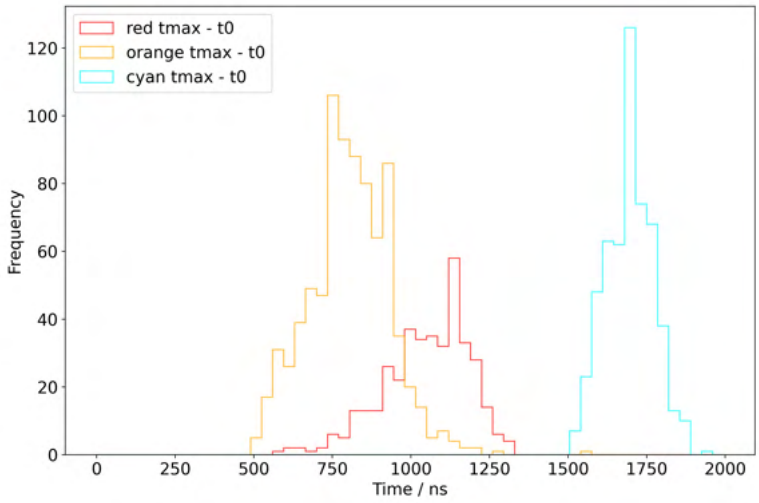
Additional simulations were provided by Seemab as shown in Figures 4.5 and 4.6 showing the motion of charge carriers, and the charge and current readings for the single-site events respectively. Figure 4.6 shows inflated current values for events with low drift time values. The events that correspond to these are waveforms 1, 6, and 10. Waveform 6 is close to the p^+ electrode, therefore resulting in a p^+ contact event where there is an increase current value due to charge carrier effects.

However, waveforms 1 and 10; which are also close to the detector; converge towards the p^+ electrode at a different point with respect to the other events, which converge towards the middle. These are the borehole events described above which are visible in the electric field lines in Figure 4.4. The corresponding waveforms in Figure 4.6a for events 1 and 10 also feature the kink behaviour observed in Figure A3.8, which supports

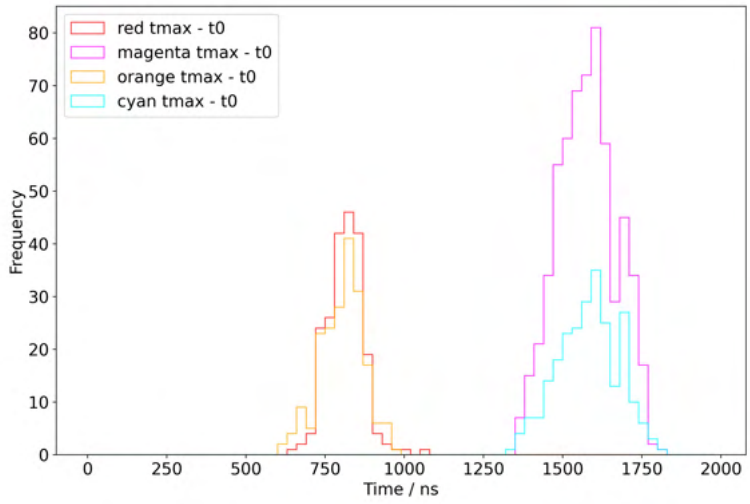
the idea that the borehole events lead to an increased A/E value due to the kink in charge deposition that leads to an increase in current, visible in Figure 4.6b. Where waveform 10 has a clear discontinuity in the current signal due to the kink behaviour of the waveform in Figure 4.6a, and waveforms 1 and 6 have an increase in the maximum current reading A of the signal compared to the other waveforms with the same energy deposition. This increase in A/E value for low drift time waveforms that correspond to borehole and p^+ contact events then leads to the inflated A/E tail that is seen in the ^{228}Th calibration data for the above batch-5 detectors analysed (shown in Figures 4.1c, 4.1d, and 4.1b).

4.2. Time-point analysis

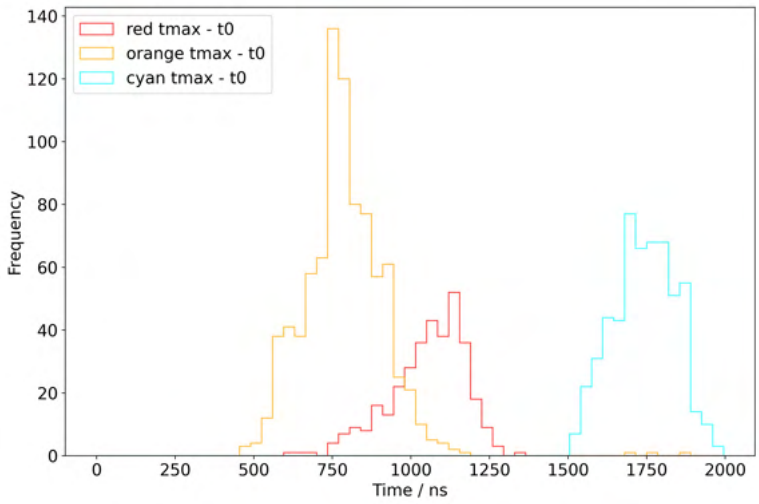
As we now can observe that the inflated A/E tail regions in the orange box have a shorter and sharper waveform shape, we can compare and analyse the waveforms using the time-point parameters for the waveforms. Where a time-point (tN) parameter is the total time from the start of a waveform to reach N percentage of the total energy. So 't80' would indicate the time to reach 80% of a waveforms energy from the start time.



(a) V05612A, bin width: 35 ns



(b) V05612B, bin width: 30 ns



(c) V05267A, bin width: 35 ns

Figure 4.7: Distribution of waveforms for each analysis region for the batch-5 detectors. The 'tmax' parameter is equivalent to the time-point parameter t100 as mentioned above, so tmax - t0 indicates the entire waveform from the estimated start to finish time.

Where it can be seen that for Figure 4.7b, the waveform distributions for each of the corresponding high density and inflated A/E tail regions almost completely overlap for both high and low drift times. This means that the following analysis methods of investigating the time-point distributions will not be effective on detector V05612B. Therefore for the remaining analysis, the focus will be on the 'single-tailed' batch-5 detectors (V05612A and V05267A).

In order to analyse how similar the waveform shapes are at different time-points, the overlap percentages between the high density time-point distributions (cyan and red) and the inflated A/E tail regions (orange) will be calculated by using:

$$\text{overlap} = \frac{O(\text{orange}, \text{red}) + O(\text{orange}, \text{cyan})}{A} \tag{9}$$

Where $O(\chi, \eta)$ is the overlap between histogram distributions χ and η, and A the total area of all the time-point distributions plotted. The tail region waveforms that differ the most at selected time-points, would therefore have a lower overlap percentage.

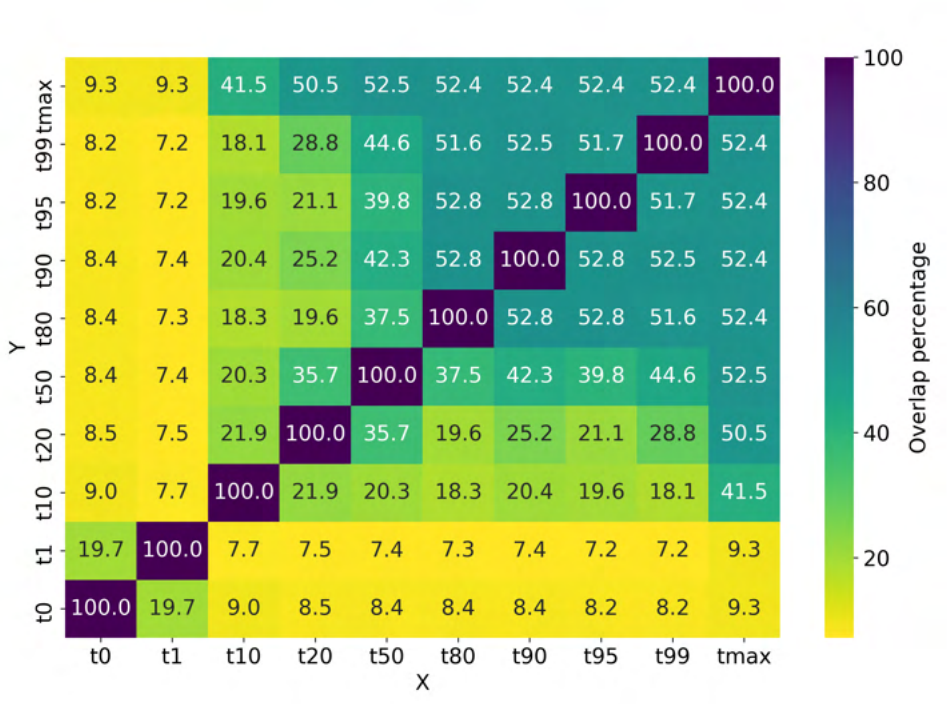


Figure 4.8: Parameter space of time-point distributions with a figure of merit that indicates the overlap percentage between the time-point distributions $|Y - X|$ for detector V05612A.

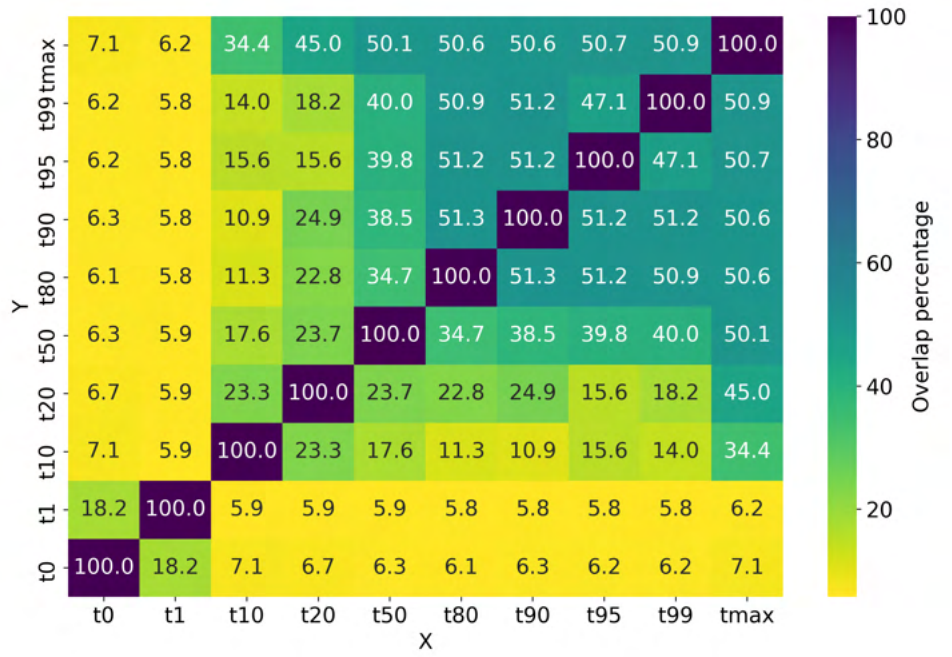


Figure 4.9: Parameter space of time-point distributions with a figure of merit that indicates the overlap percentage between the time-point distributions $|Y - X|$ for detector V05267A.

Where the best performing time-point distributions are |t99 – t1| and |t95 – t1|, and |t99 – t1| to |t80 – t1| for detectors V05612A and V05267A respectively. The time-point differences about the top of the detector have a majority overlap, which suggests most of the waveforms have the same shape towards the end, however the time-point differences from |t10 – t1| and beyond have an overlap that is an order smaller than that from the end of the waveforms, suggesting that the differences arise from the kinks at the start of the waveform as talked about above.

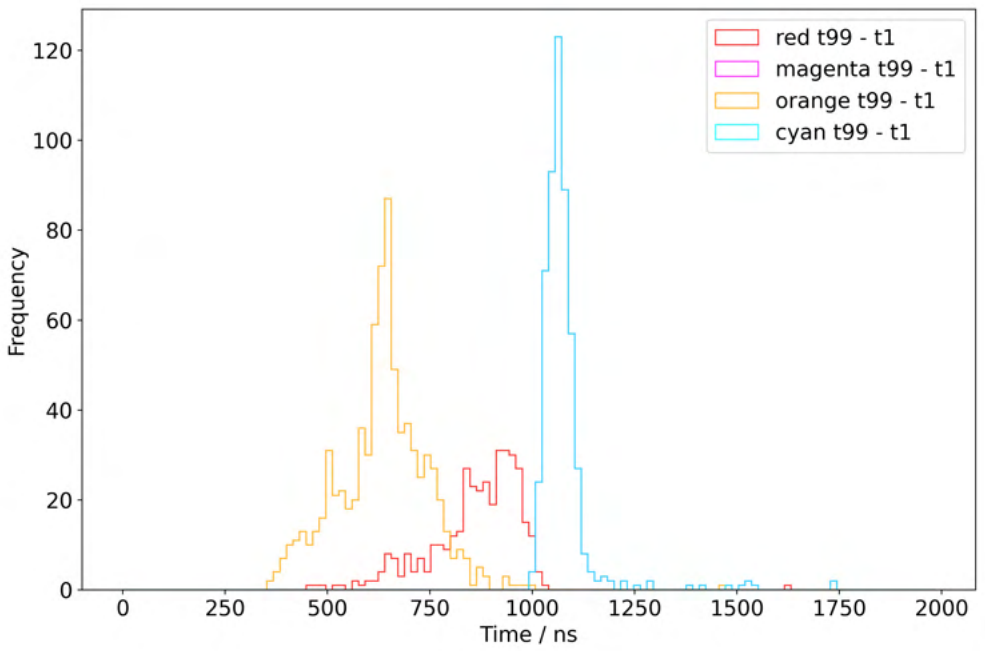


Figure 4.10: Distribution of waveforms in each analysis region for detector V05612A with a bin width of 16 ns. The |t99 – t1| time-point range is used as determined by the parameter space plot from Figure 4.8.

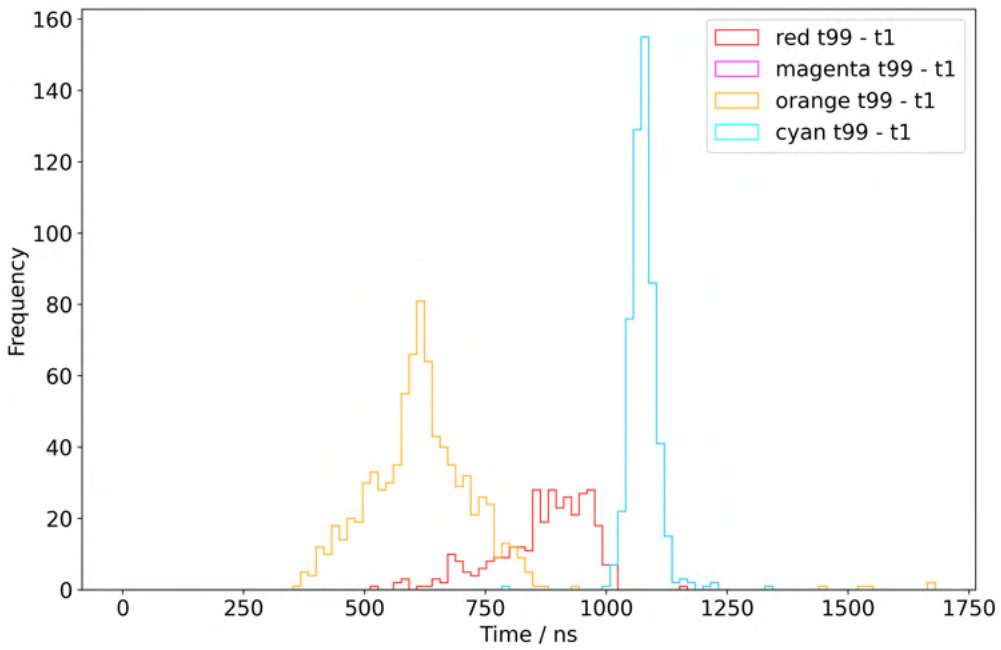


Figure 4.11: Distribution of waveforms in each analysis region for detector V05267A with a bin width of 16 ns. The $|t99 - t1|$ time-point range is used as determined by the parameter space plot from Figure 4.9.

Where for the above plots the smallest possible bin width (16 ns) was chosen to keep the histogram distributions continuous and have the most accurate value of overlap for the distribution of data. The 'single-tail' detectors analysed above show a characteristic distribution for the $|t99 - t1|$, however more detectors of this fashion would have to be analysed to confirm this. Once the overlap has been minimised by inspecting the parameter space plots, a 2D histogram of the selected time-points against the A/E values can be plotted. This should highlight differences in the orange analysis region that aren't shown in Figures 4.1c and 4.1b. It is expected in the new time-point - A/E plot that the tail regions will have a more visible negative gradient or decrease in drift time due to the selected time-points from the parameter space plot minimising the difference in the time distributions, whereby the tail regions have a visibly lower median time (from Figures 4.10 and 4.11). This would allow a re-calibration of the A/E value for the inflated A/E tail events, reducing the A/E value and resulting in the drift time - A/E tail being removed. The resultant drift time - A/E plot would have similar shape to Figure 4.1a.

In addition to the time-point overlaps chosen via Figures 4.8 and 4.9, the smaller time-point difference $|t50 - t1|$ (distributions shown in Figures A1.11 and A3.11) will be analysed for detectors V05612A and V05267A as it should highlight the difference due to the kinks at the start of the waveforms.

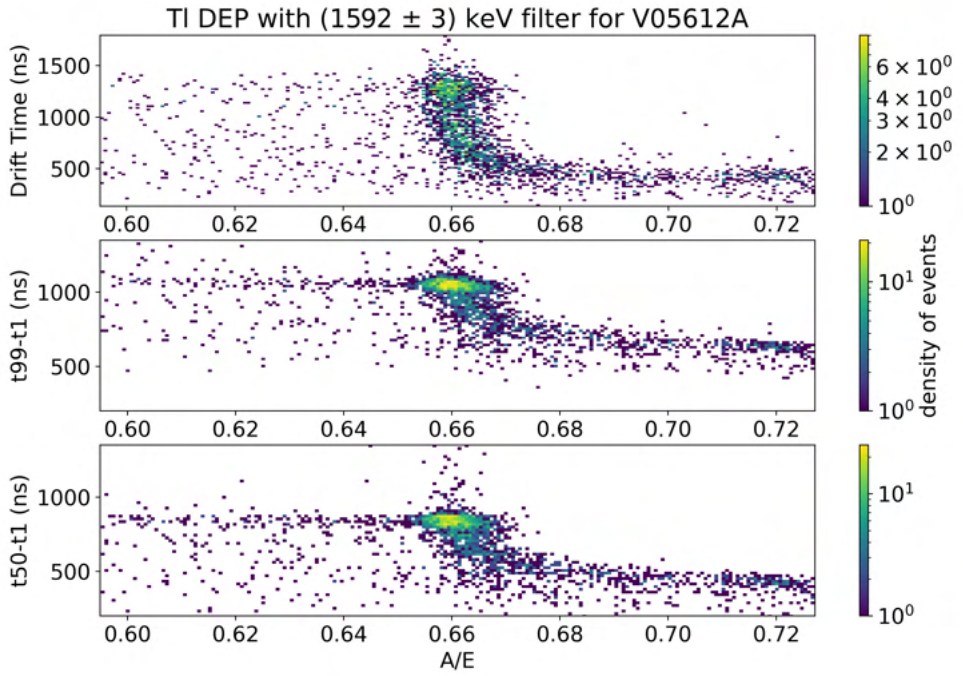


Figure 4.12: 2D Histogram of A/E against time-point difference for detector V05612A with a colour bar representing the density of events per bin. The above plot is centered at the Tl DEP around (1592 ± 3) keV.

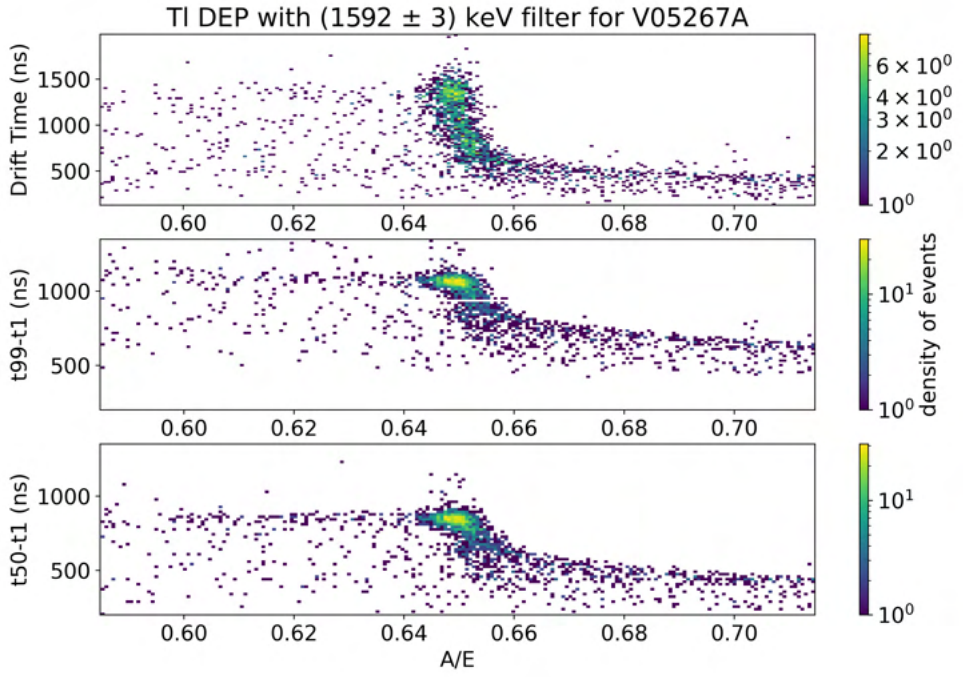
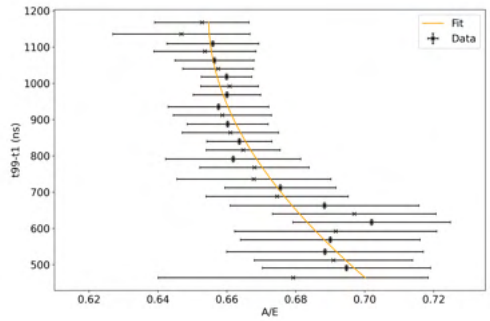
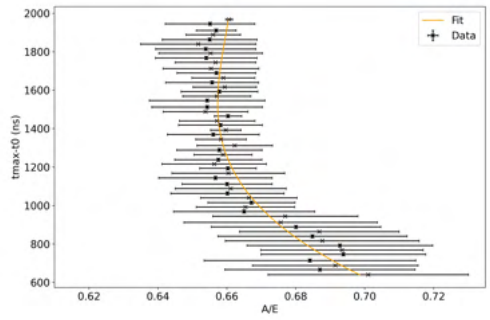


Figure 4.13: 2D Histogram of A/E against time-point difference for detector V05267A with a colour bar representing the density of events per bin. The above plot is centered at the Tl DEP around (1592 ± 3) keV.

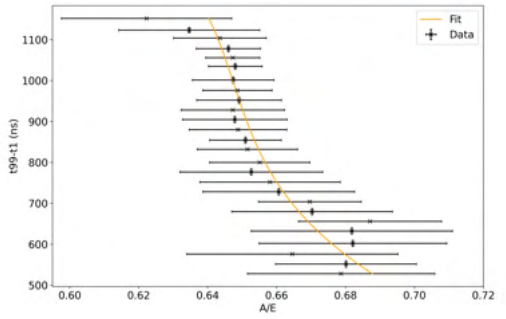


(a) |t99 – t1|

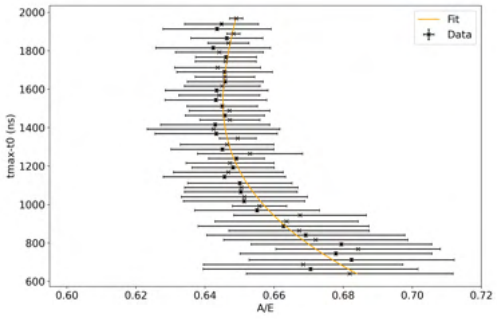


(b) |tmax – t0|

Figure 4.14: Distribution of time bands against A/E. Time bands of 25 ns were selected and the median A/E in the time band was calculated. Errorbars indicate one standard deviation of the A/E values for each respective time band. Data is fit to a 3rd order polynomial fit for detector V05612A.



(a) |t99 – t1|



(b) |tmax – t0|

Figure 4.15: Distribution of time bands against A/E. Time bands of 25 ns were selected and the median A/E in the time band was calculated. Errorbars indicate one standard deviation of the A/E values for each respective time band. Data is fit to a 3rd order polynomial fit for detector V05267A.

Where it can be seen in Figures 4.12 and 4.13 that the time-point difference plots exhibit a tail with a steeper gradient than that of the normal drift time plots for low A/E values. Although the inflated A/E tail has a steeper gradient via a polynomial fit as can be seen in Figures 4.14a and 4.15a, there is still a massively spread distribution of A/E values. This A/E spread doesn't allow for calibration as it would also shift A/E values centered about the high density region to a lower A/E value. This analysis therefore tells us that a simple shift calibration cannot be applied to the A/E via drift time analysis due to the high standard deviation of the A/E values for low drift times.

Alternatives to this for LEGEND would be to program a neural network that re-calibrates the inflated A/E tail values to account for the mentioned p^+ contact and borehole events. Another alternative would be to use a hard-cut A/E value. This will remove a majority of the inflated A/E tail events while trying to retain the high density events. However removing additional data is not an optimal outcome as neutrinoless double-beta decay is an already rare event with a half-life $\sim 10^{28}$yr, therefore retaining the most amount of data is paramount.

5. CONCLUSION

The A/E analysis for the batch-4 detector V04199A retains $(90.9 \pm 0.9)\%$ of the DEP events containing a majority single-site nature that imitates neutrinoless double-beta decay, while retaining $(8.0 \pm 0.2)\%$, $(5.6 \pm 0.1)\%$, and $(6.5 \pm 0.1)\%$ of the Bi FEP, Tl SEP, and Tl FEP events respectively which contain a majority multi-site nature that represents background events such as Compton scattering of gamma rays within the Germanium detector. The A/E analysis doesn't perform as well for the batch-5 detectors V05612A, V05612B, and V05267A with the post A/E analysis containing 1.83 times the amount of Bi FEP, Tl SEP, and Tl FEP events compared to detector V04199A. A resultant drift time analysis shows characteristic inflated A/E tail regions in the detector that are believed to be a result of p^+ contact events and borehole events found in simulation and modelling work by Seemab Haider from the UCL LEGEND group. Where both the simulations by Seemab Haider, and the analysis of ^{228}Th calibration data on the batch-5 detectors support this by observing a kink characteristic in the start of the waveforms which results in an increase in current, caused by charge carrier effects from p^+ contact events, and the motion of charge carriers from borehole events. However, more simulation work will be needed to determine the contribution of these borehole events. Time-point distributions of the waveforms were then analysed to find at what times the average waveforms from the low A/E spread regions differ to that from the inflated A/E tail regions, in order to re-calibrate the inflated A/E values along the tail. Waveforms from detector V05612B, containing two inflated A/E regions at both high and low drift times, were non-differentiable due to the similarities in the waveforms from both the low A/E spread and inflated A/E regions. Therefore, the analysis continued with detectors V05612A and V05687A that exhibited a single inflated A/E region for low drift times. However, it was found from the time-point analysis that the 'single-tail' detectors (V05612A and V052687A) possess a high A/E standard deviation that doesn't allow for a simple re-calibration of shifting the A/E values.

This suggests that alternative methods will be needed to deal with the degenerate A/E values. One method would be to program a neural network that re-calibrates the inflated A/E tail values to account for the mentioned p^+ contact and borehole events. An additional alternative is to apply a hard-cut A/E value, however this is not optimal as this removes additional data in an experiment that seeks to observe an already rare neutrinoless double-beta decay with a half-life of $T_{1/2}^{0\nu} \sim 10^{28}$yr.

REFERENCES

1. Cowan, C. L., Reines, F., Harrison, F. B., Kruse, H. W. & McGuire, A. D. Detection of the Free Neutrino: a Confirmation. *Science* **124,** 103–104 (1956).
2. Danby, G. *et al.* Observation of High-Energy Neutrino Reactions and the Existence of Two Kinds of Neutrinos. *Phys. Rev. Lett.* **9,** 36–44 (1962).
3. Kodama, K. *et al.* Observation of tau neutrino interactions. *Physics Letters B* **504,** 218–224. ISSN: 0370-2693 (2001).
4. Maki, Z., Nakagawa, M. & Sakata, S. Remarks on the Unified Model of Elementary Particles. *Progress of Theoretical Physics* **28,** 870–880 (1962).
5. Dirac, P. A. M. & Fowler, R. H. The quantum theory of the electron. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* **117,** 610–624 (1928).
6. Dirac, P. A. M. & Fowler, R. H. The quantum theory of the Electron. Part II. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* **118,** 351–361 (1928).
7. Majorana, E. Theory of the symmetry of electrons and positrons. **14,** 171–184 (1937).
8. Mohapatra, R. Seesaw mechanism and its implications. *25* (2005).
9. Zyla, P. *et al.* Review of Particle Physics. *PTEP* **2020,** 083C01 (2020).
10. Thomson, M. *Modern Particle Physics* (Cambridge University Press, 2013).
11. Kayser, B. Are neutrinos their own antiparticles? **173,** 012013 (2009).
12. Haxton, W. & Stephenson, G. Double beta decay. *Progress in Particle and Nuclear Physics* **12,** 409–479. ISSN: 0146-6410 (1984).
13. Davidson, S., Nardi, E. & Nir, Y. Leptogenesis. *Physics Reports* **466,** 105–177. ISSN: 0370-1573 (2008).
14. LEGEND *et al. LEGEND-1000 Preconceptual Design Report* 2021. arXiv: 2107.11462 [physics.ins-det].

15. Dell'Oro, S., Marcocci, S., Viel, M. & Vissani, F. Neutrinoless Double Beta Decay: 2015 Review. *Advances in High Energy Physics* **2016** (2016).

16. GERDA. Background-free search for neutrinoless double-β decay of ^{76}Ge with GERDA. *Nature* **544,** 47–52 (2017).

17. Agostini, M. *et al.* Probing Majorana neutrinos with double-β decay. *Science* **365,** 1445–1448. ISSN: 1095-9203 (Sept. 2019).

18. Zen Collaboration. *First Search for the Majorana Nature of Neutrinos in the Inverted Mass Ordering Region with KamLAND-Zen* 2022.

19. Search for Majorana neutrinos with the first two years of EXO-200 data. *Nature* **510,** 229–234 (June 2014).

20. Alvis, S. I. *et al.* Search for neutrinoless double-β decay in ^{76}Ge with 26 kg yr of exposure from the Majorana Demonstrator. *Phys. Rev. C* **100,** 025501 (2 Aug. 2019).

21. Arnold, R. *et al.* Result of the search for neutrinoless double-b decay in ^{100}Mo with the NEMO-3 experiment (2015).

22. Cooper, R., Radford, D., Hausladen, P. & Lagergren, K. A novel HPGe detector for gamma-ray tracking and imaging. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **665,** 25–32. ISSN: 0168-9002 (2011).

23. Aalseth, C. E. *et al.* Search for Neutrinoless Double-β Decay in ^{76}Ge with the Majorana Demonstrator. *Phys. Rev. Lett.* **120,** 132502 (2018).

24. Comellato, T., Agostini, M. & Schönert, S. Charge-carrier collective motion in germanium detectors for $\beta\beta$-decay searches. *The European Physical Journal C* **81.** ISSN: 1434-6052 (2021).

25. Mertens, S. *et al.* MAJORANA Collaboration's Experience with Germanium Detectors. **606,** 012005 (2015).

26. Budjáš, D., Heider, M. B., Chkvorets, O., Khanbekov, N. & Schönert, S. Pulse shape discrimination studies with a Broad-Energy Germanium detector for signal identification and background suppression in the GERDA double beta decay experiment. **4,** P10007–P10007 (2009).

27. Schönert, S. "LEGEND-1000". *North America - Europe Workshop on Future of Double Beta Decay,* "E. Fermi" auditorium (Gran Sasso National Laboratory). `https://agenda.infn.it/event/27143/timetable/#20210929.detailed` (30th September 2021).

28. Agostini, M. *et al.* Characterization of inverted coaxial {76}Ge detectors in GERDA for future double-\beta decay experiments. *The European Physical Journal C* **81** (June 2021).

29. Abgrall, N. *et al.* The Majorana Demonstrator calibration system. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **872,** 16–22. ISSN: 0168-9002 (2017).

30. Baudis, L. *et al.* Production and characterization of ^{228}Th calibration sources with low neutron emission for GERDA. *Journal of Instrumentation* **10,** P12005–P12005. ISSN: 1748-0221 (2015).

31. Marshall, G. & Agostini, M. "A/E update and Issues". *LEGEND Analysis and Simulations* (Dec. 2022).

A. FURTHER PLOTS

A1. V05612A

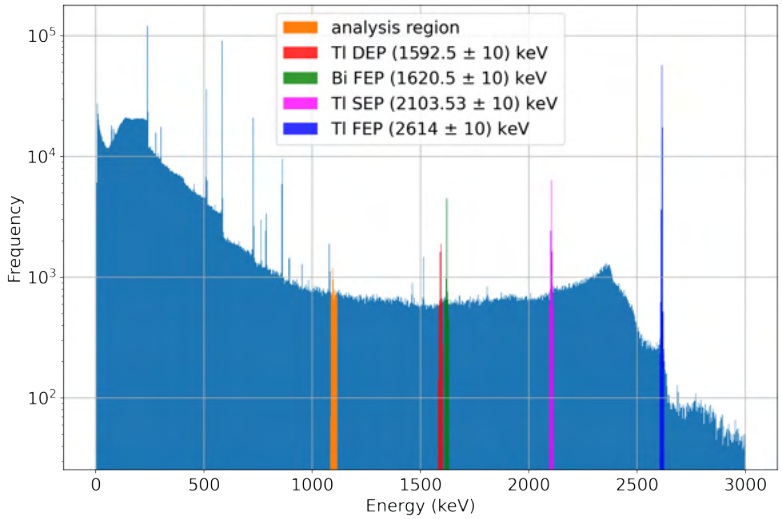


Figure A1.1: Shows energy spectra of Th-228 calibration data for detector V05612A with the analysis energy band used to form Figures 3.1a and 3.1b. In addition showing regions of interest in order to determine the success for the A/E analysis.

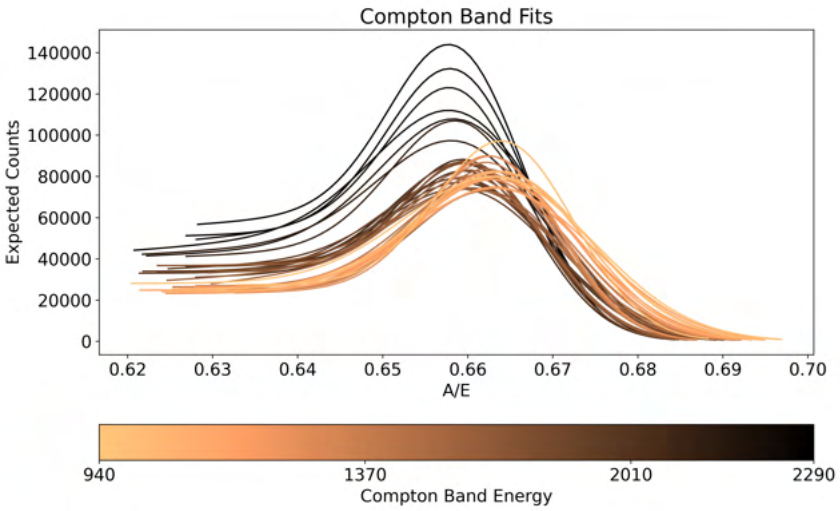
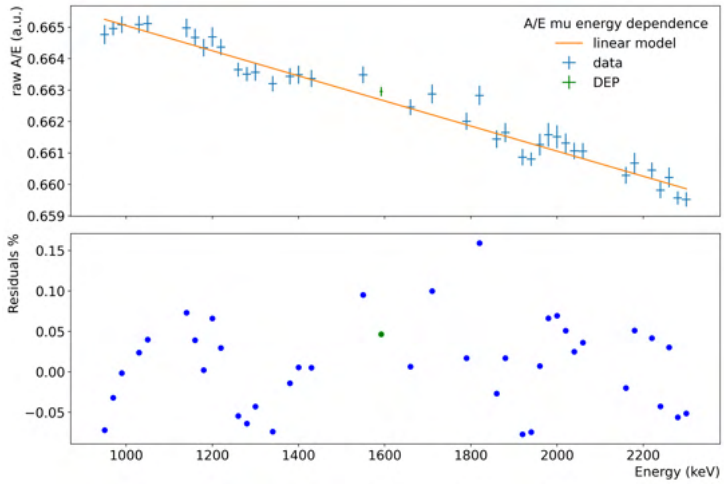
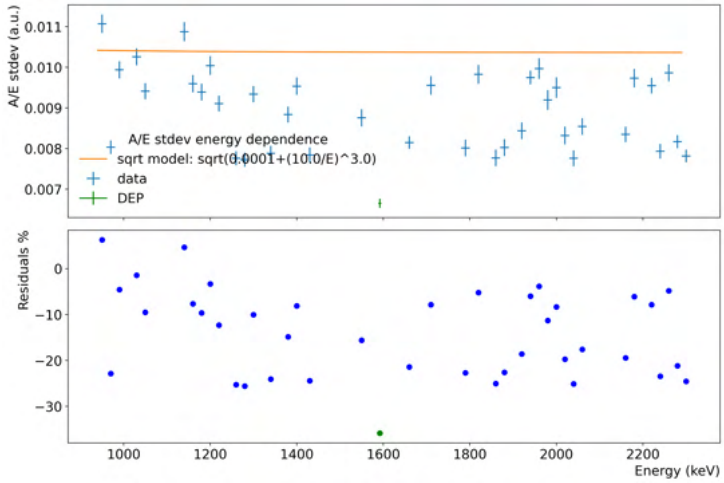


Figure A1.2: A/E distributions for range of 20 keV energy bands from 940 - 2290 keV for detector V05612A.



(a) A/E distribution medians against energy. A linear fit is applied to determine the A/E energy dependence with values of $a = -4.0 \times 10^{-6}$ and $b = 0.7$ from Equation 6.



(b) A/E distribution standard deviation against energy. A root mean square fit is applied to determine A/E standard deviation energy dependence with values of $c = 3.0$, $d = 10.0$, and $f = 10.7 \times 10^{-5}$ from Equation 7.

Figure A1.3: Raw A/E distribution mean and standard deviation against energy, with with linear and root mean square fits respectively for detector V05612A.

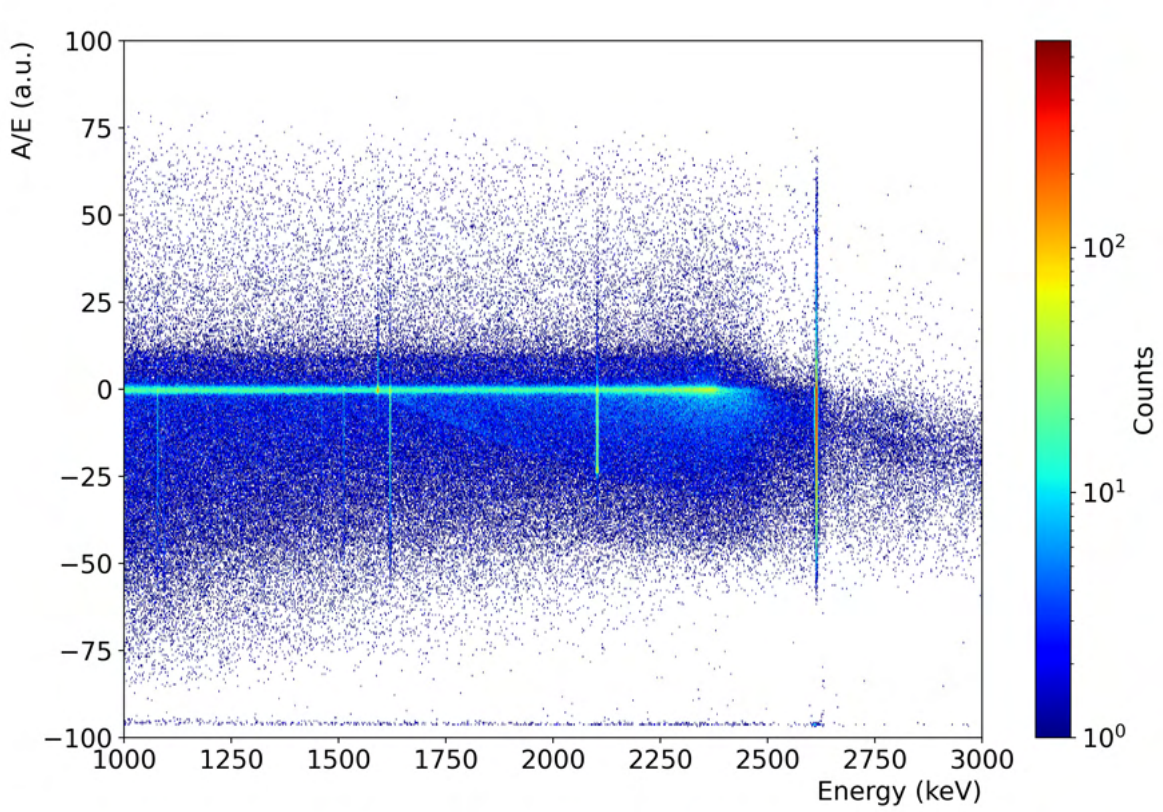


Figure A1.4: 2D Histogram of classifier A/E-Energy plot for detector V05612A.

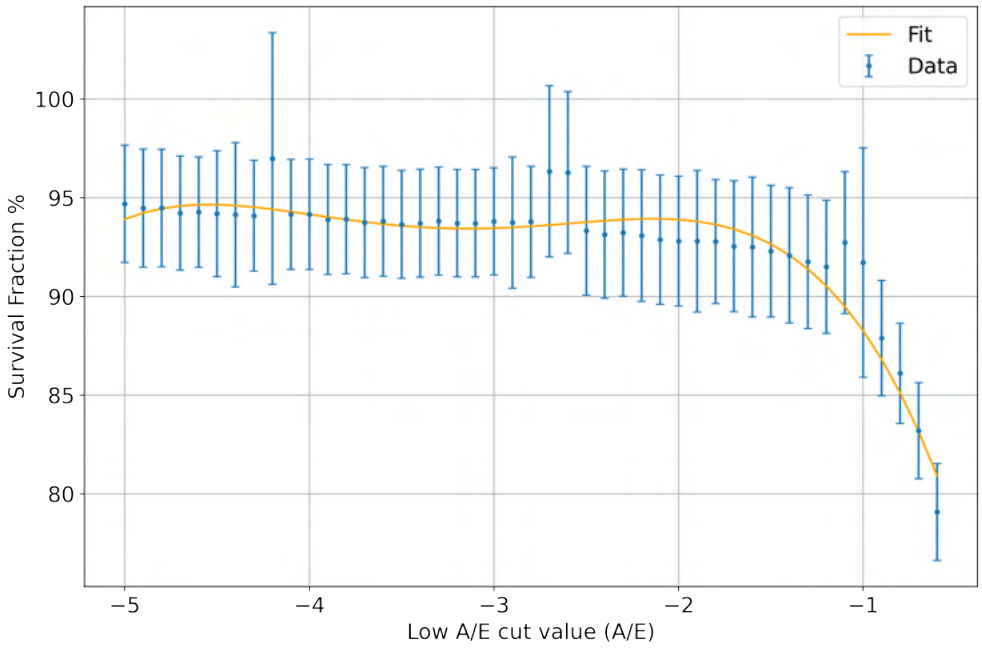


Figure A1.5: Survival fraction of DEP events about (1592 ± 10) keV against classifier A/E value. The orange line is a fourth order polynomial fit to the survival fraction data for detector V05612A.

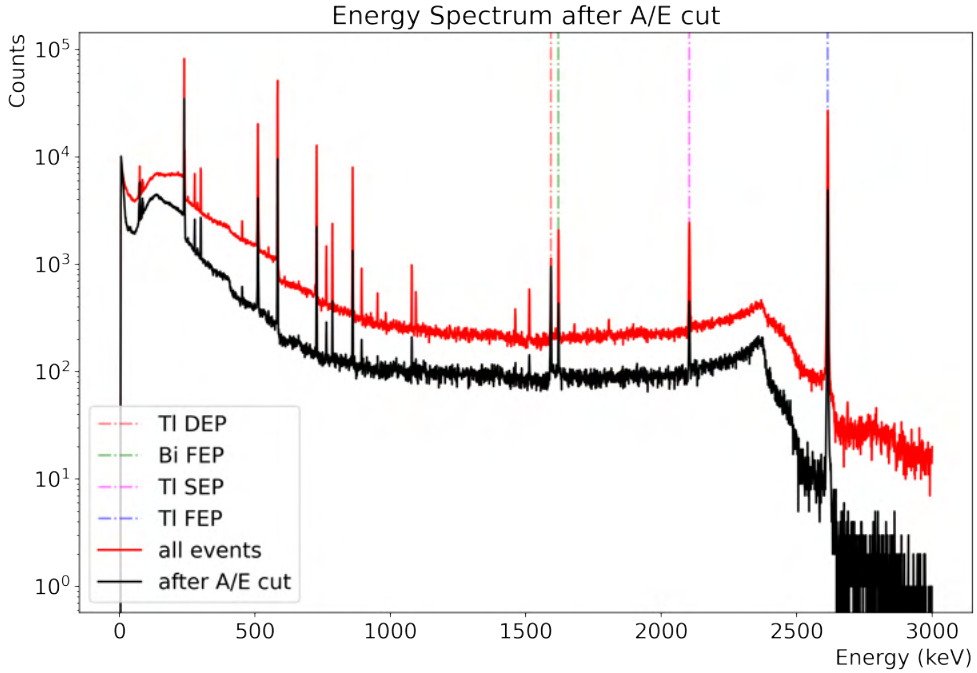


Figure A1.6: Number of events throughout the energy spectrum pre- and post-A/E cut, highlighting the FEP, SEP, and DEP regions of interest for detector V05612A.

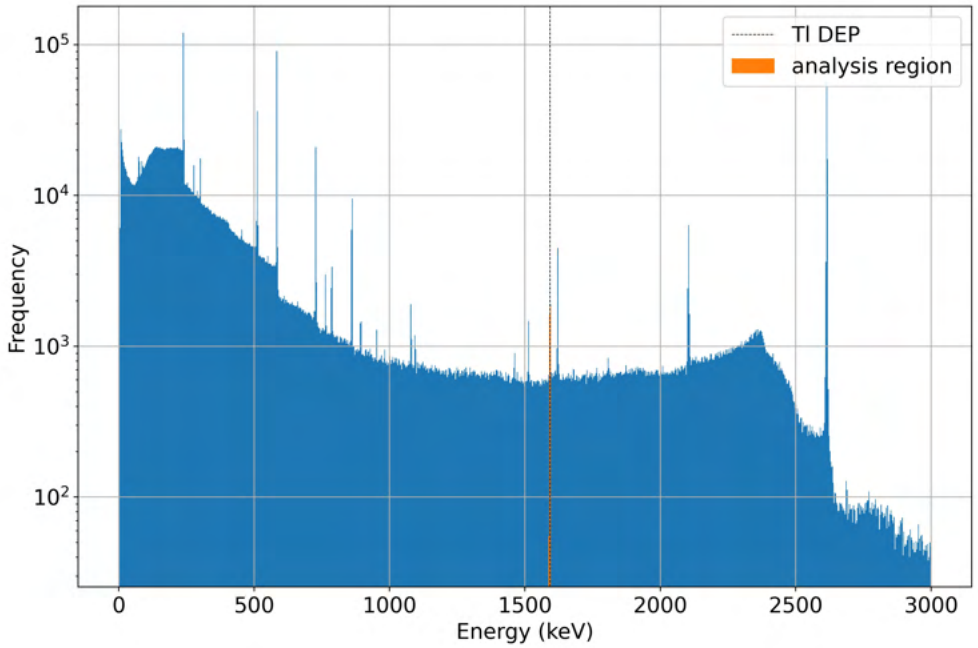


Figure A1.7: Energy spectra of Th-228 calibration dtaa for detector V05612A with analysis region centered about the Tl DEP (1592 ± 3) keV.

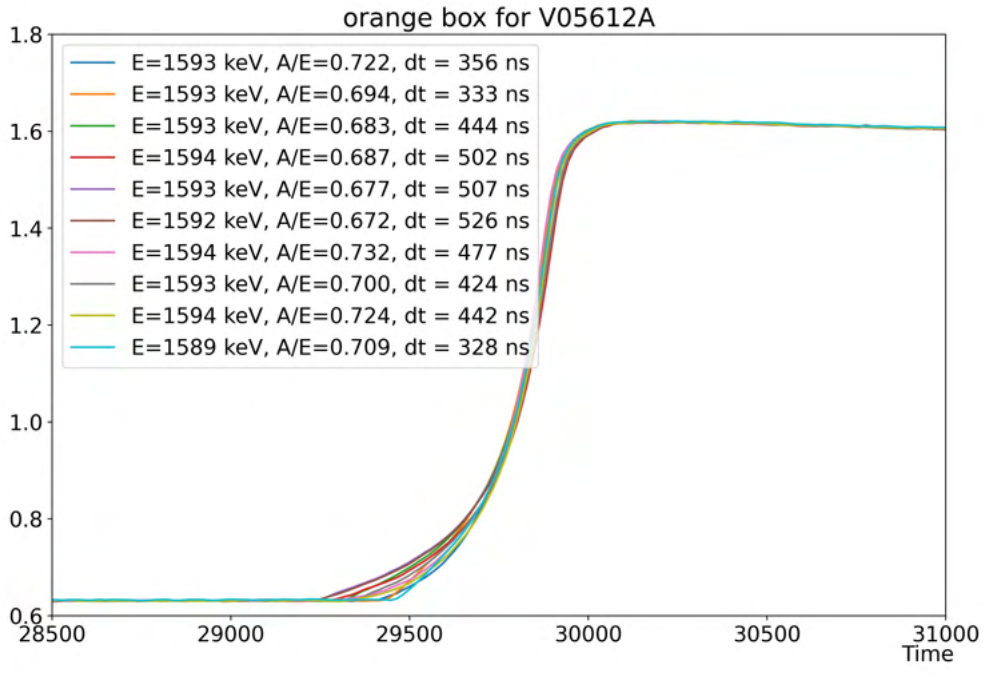


Figure A1.8: Selection of waveforms contained within the orange analysis region for detector V05612A.

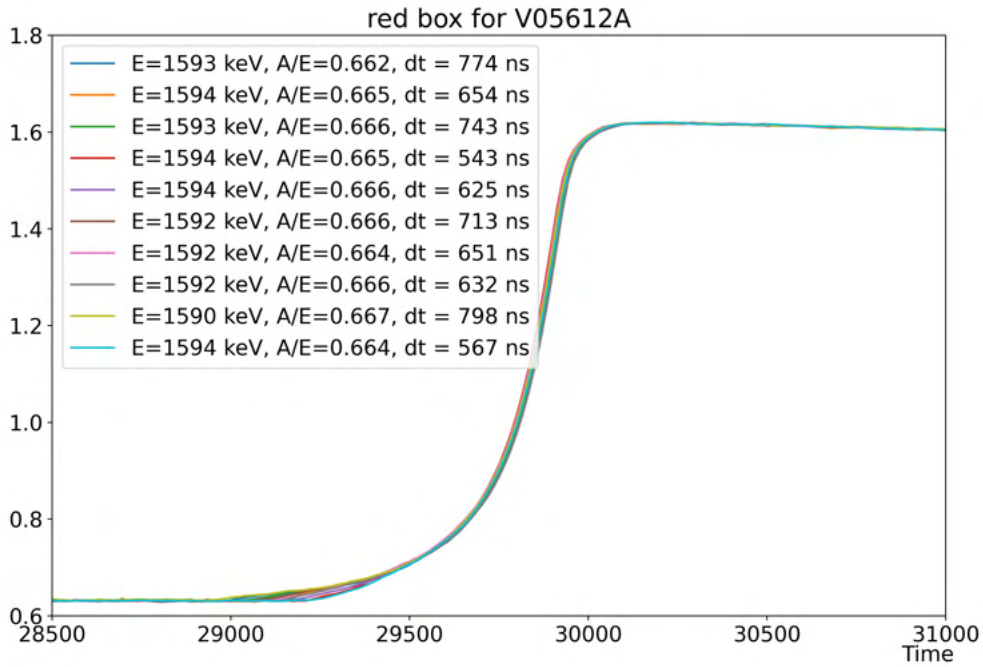


Figure A1.9: Selection of waveforms contained within the red analysis region for detector V05612A.

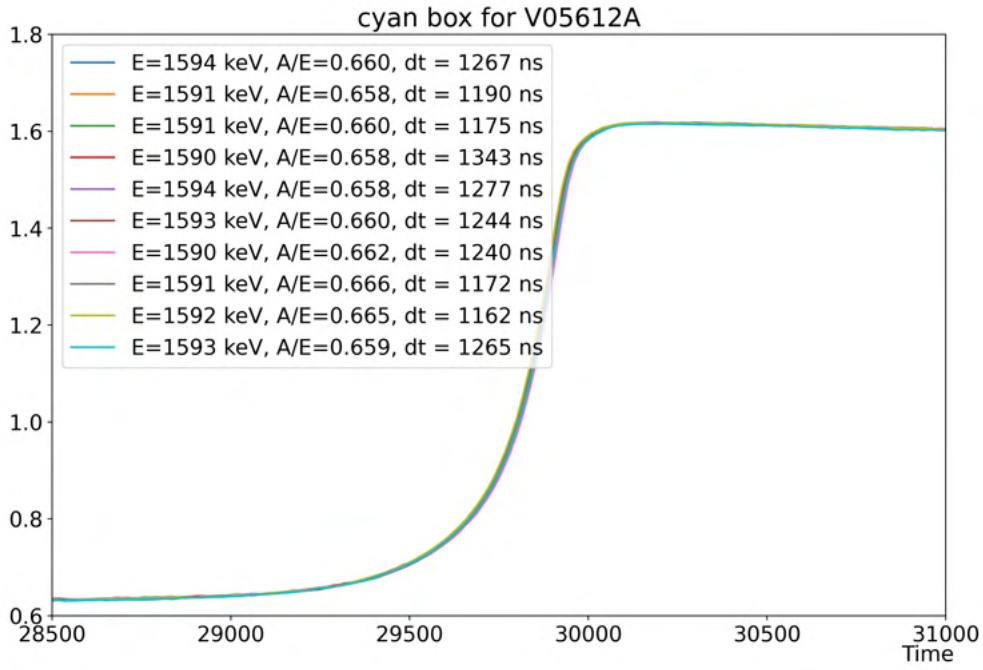


Figure A1.10: Selection of waveforms contained within the cyan analysis region for detector V05612A.

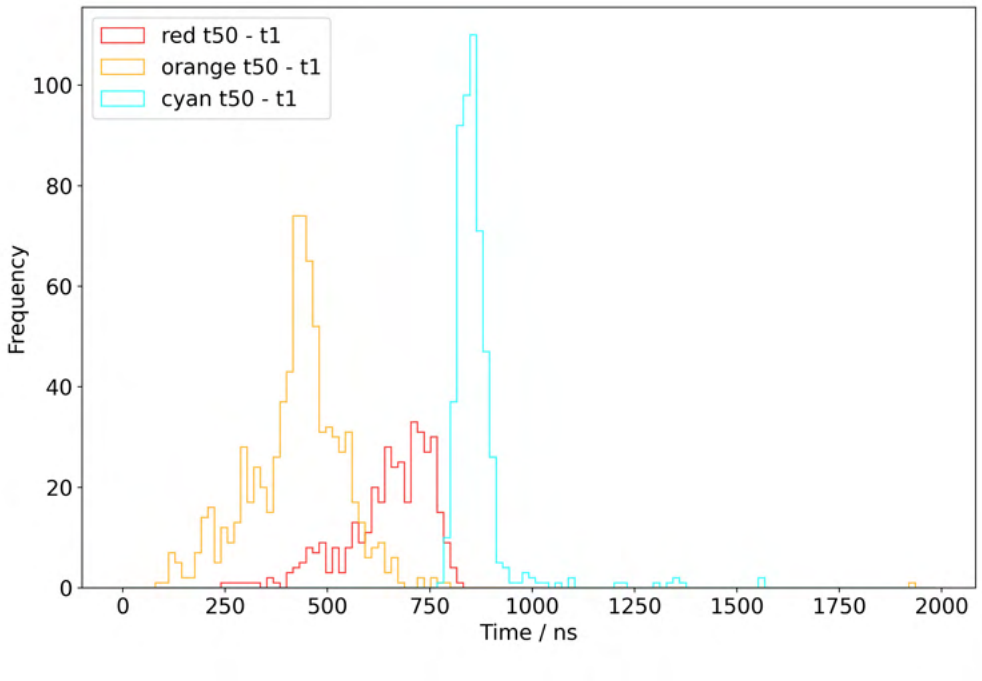


Figure A1.11: Distribution of waveforms in each analysis region for detector V05612A with a bin width of 16 ns. Where the $|t50 - t1|$ time-point range is used.

A2. V05612B

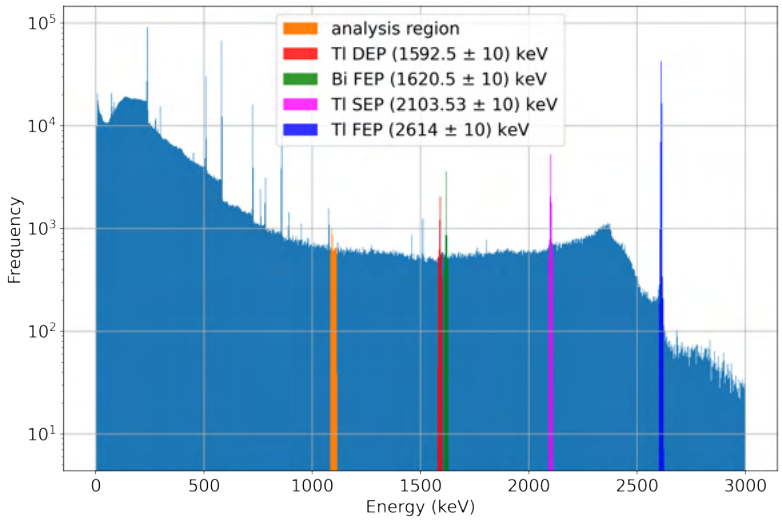


Figure A2.1: Shows energy spectra of Th-228 calibration data for detector V05612B with the analysis energy band used to form Figures 3.1a and 3.1b. In addition showing regions of interest in order to determine the success for the A/E analysis.

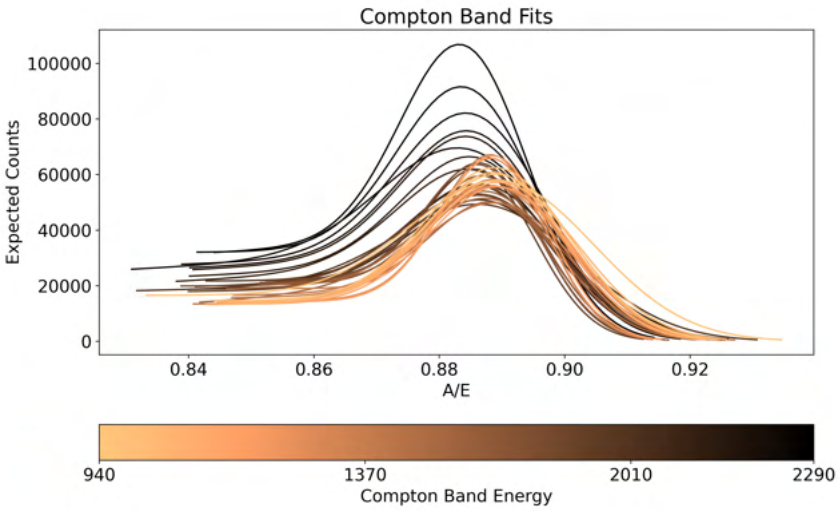
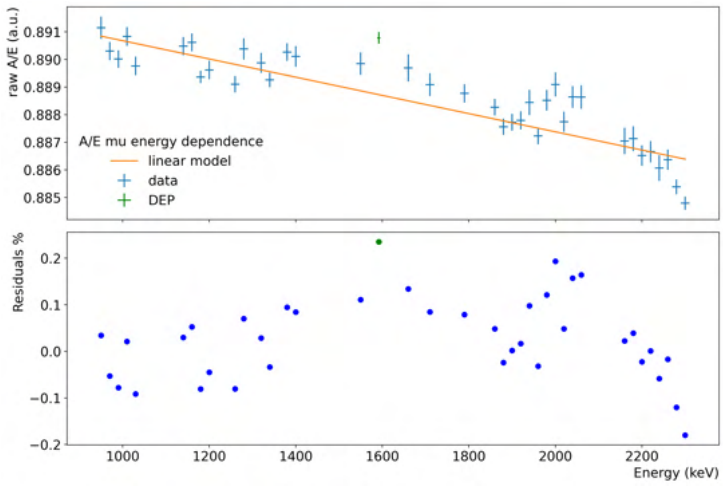
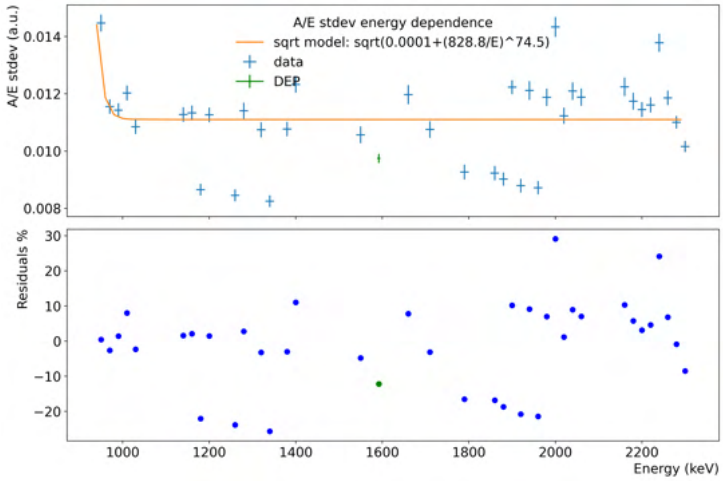


Figure A2.2: A/E distributions for range of 20 keV energy bands from 940 - 2290 keV for detector V05612B.



(a) A/E distribution medians against energy. A linear fit is applied to determine the A/E energy dependence with values of $a = -3.3 \times 10^{-6}$ and $b = 0.9$ from Equation 6.



(b) A/E distribution standard deviation against energy. A root mean square fit is applied to determine A/E standard deviation energy dependence with values of $c = 74.5$, $d = 828.8$, and $f = 12.3 \times 10^{-5}$ from Equation 7.

Figure A2.3: Raw A/E distribution mean and standard deviation against energy, with with linear and root mean square fits respectively for detector V05612B.

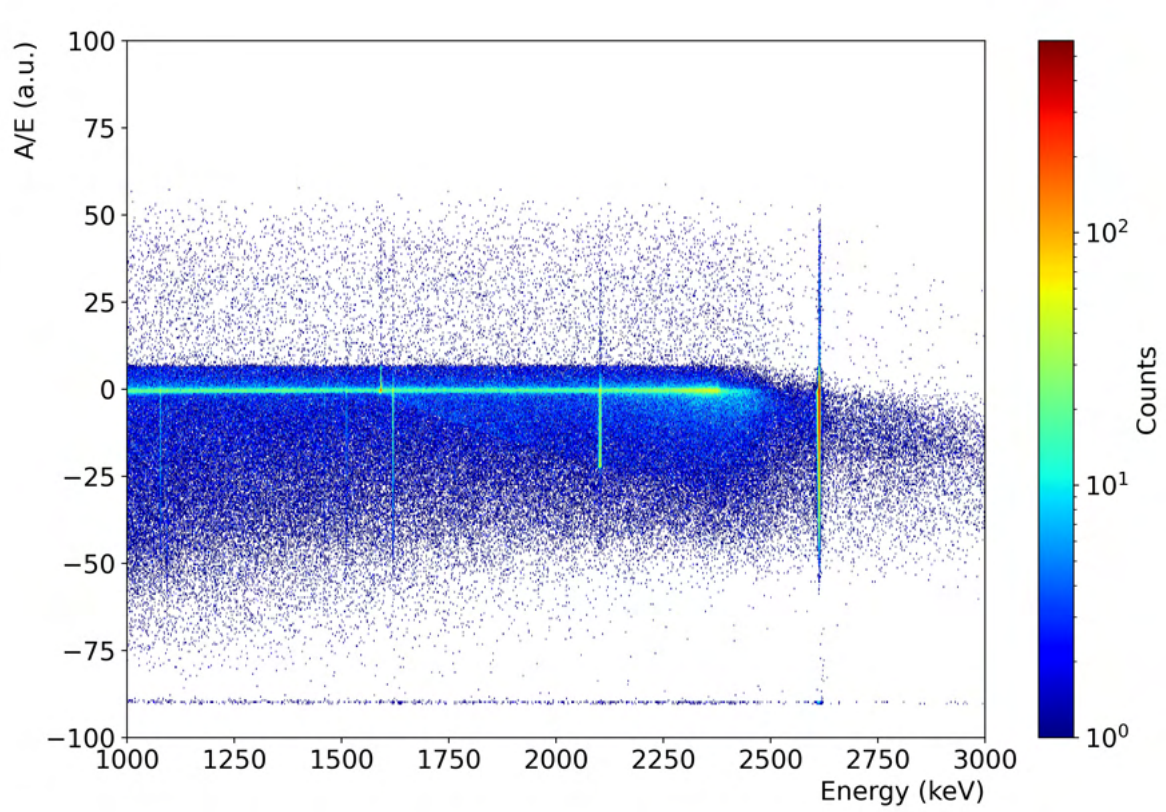


Figure A2.4: 2D Histogram of classifier A/E-Energy plot for detector V05612B.

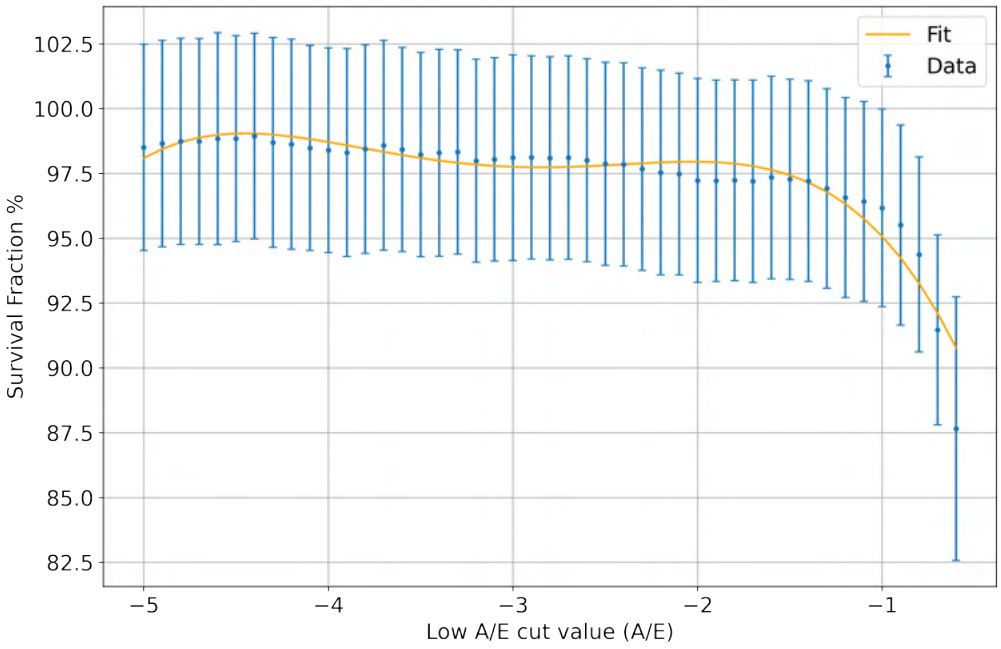


Figure A2.5: Survival fraction of DEP events about (1592 ± 10) keV against classifier A/E value. The orange line is a fourth order polynomial fit to the survival fraction data for detector V05612B.

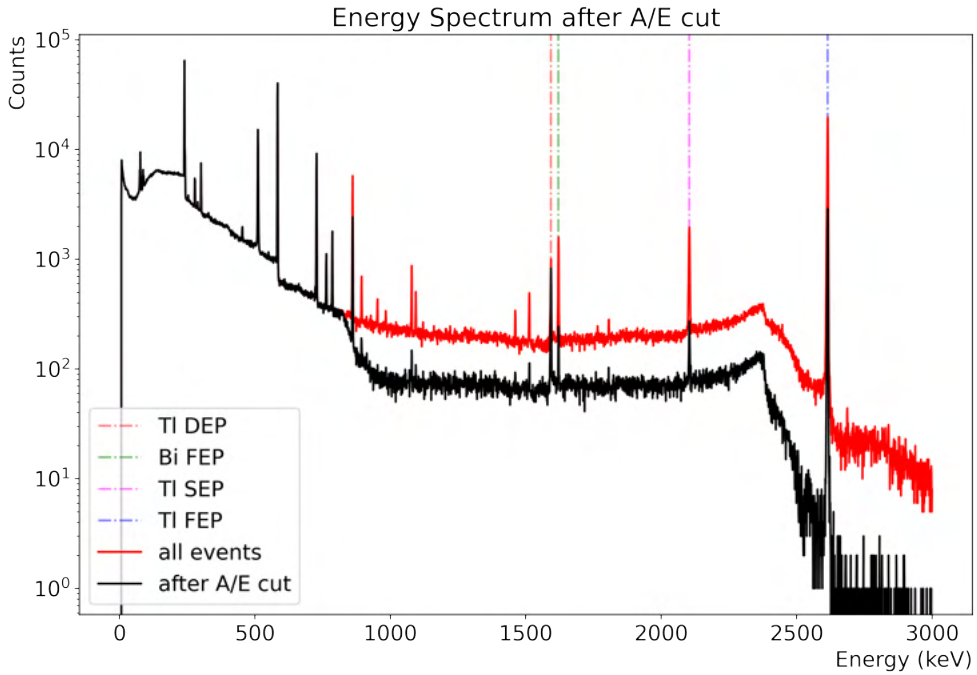


Figure A2.6: Number of events throughout the energy spectrum pre- and post-A/E cut, highlighting the FEP, SEP, and DEP regions of interest for detector V05612B.

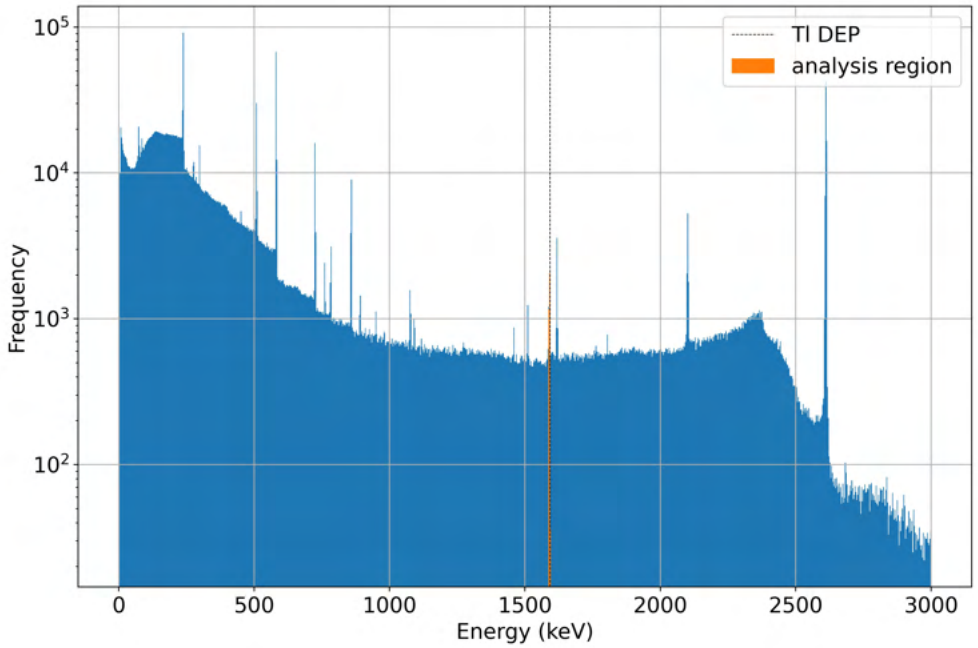


Figure A2.7: Energy spectra of Th-228 calibration dtaa for detector V05612A with analysis region centered about the Tl DEP (1592 ± 3) keV.

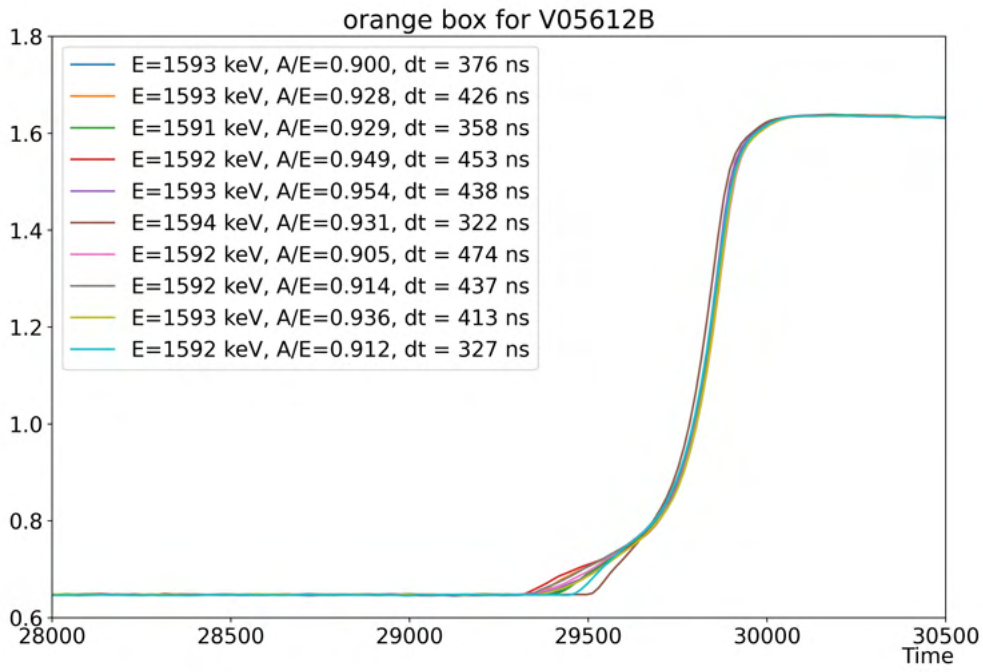


Figure A2.8: Selection of waveforms contained within the orange analysis region for detector V05612B.

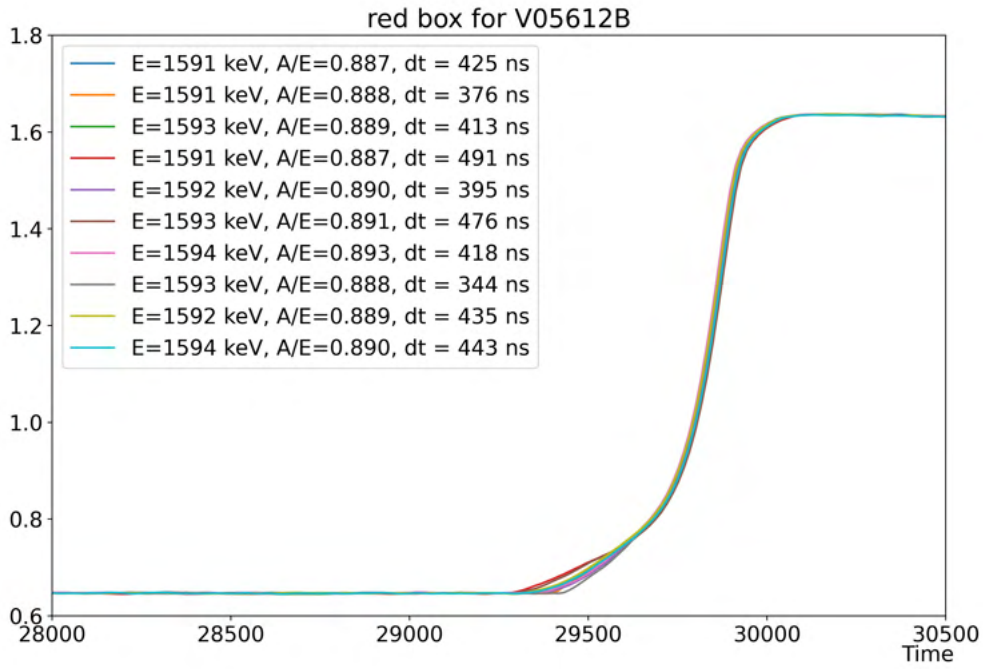


Figure A2.9: Selection of waveforms contained within the red analysis region for detector V05612B.

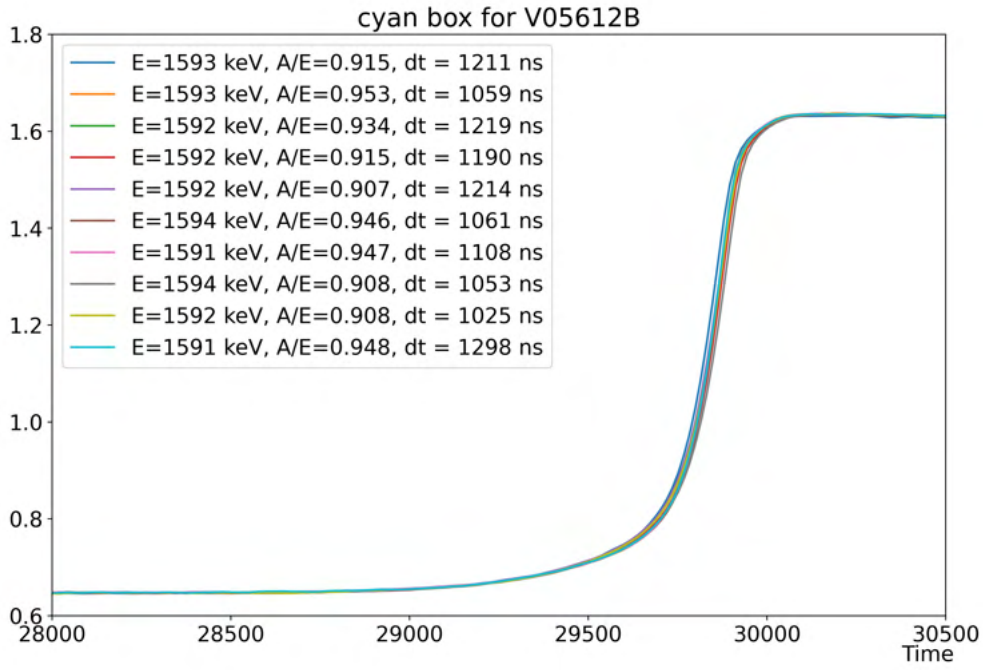


Figure A2.10: Selection of waveforms contained within the cyan analysis region for detector V05612B.

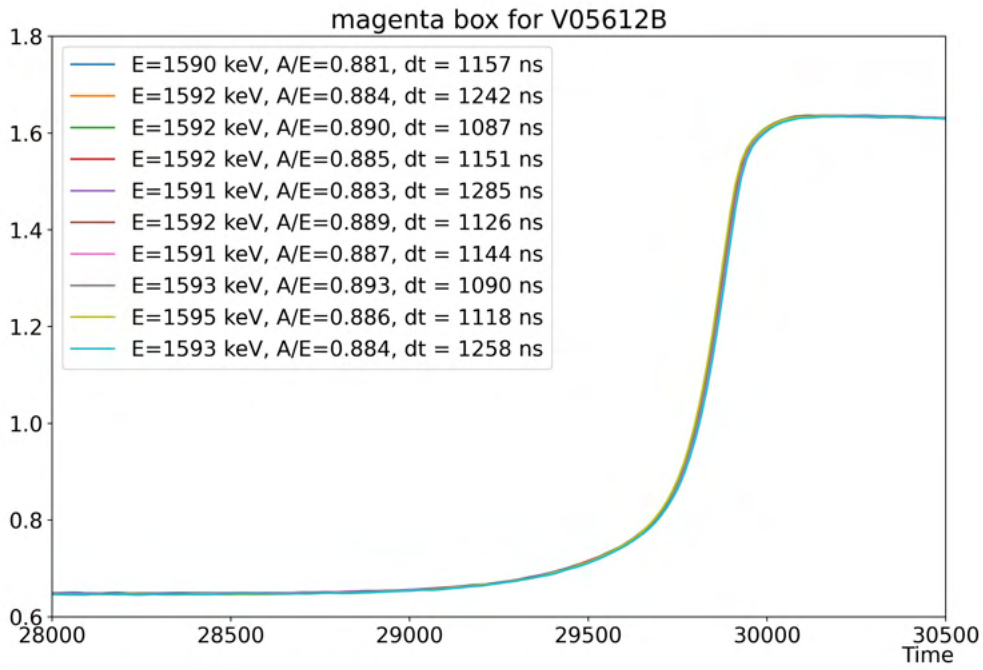


Figure A2.11: Selection of waveforms contained within the magenta analysis region for detector V05612B.

A3. V05267A

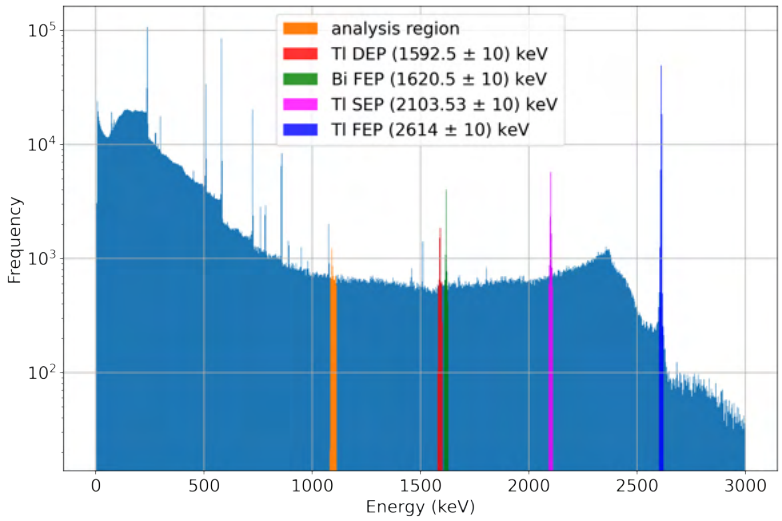


Figure A3.1: Shows energy spectra of Th-228 calibration data for detector V05267A with the analysis energy band used to form Figures 3.1a and 3.1b. In addition showing regions of interest in order to determine the success for the A/E analysis.

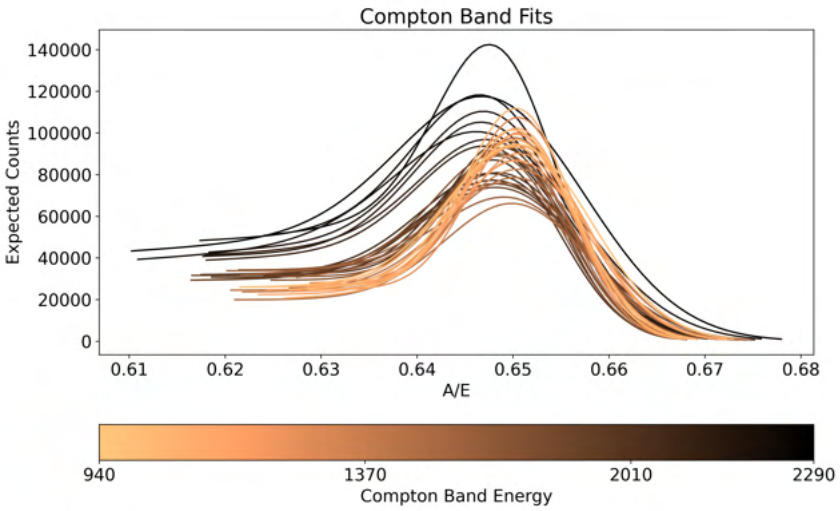
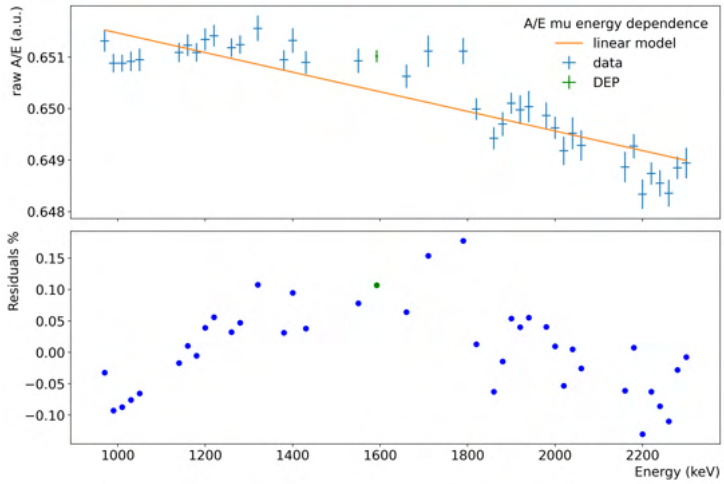
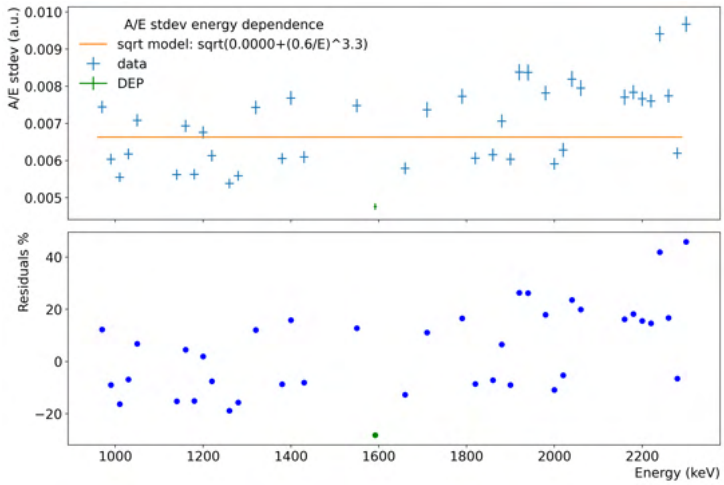


Figure A3.2: A/E distributions for range of 20 keV energy bands from 940 - 2290 keV for detector V05267A.



(a) A/E distribution medians against energy. A linear fit is applied to determine the A/E energy dependence with values of $a = -1.9 \times 10^{-6}$ and $b = 0.7$ from Equation 6.



(b) A/E distribution standard deviation against energy. A root mean square fit is applied to determine A/E standard deviation energy dependence with values of $c = 3.3$, $d = 0.6$, and $f = 4.4 \times 10^{-5}$ from Equation 7.

Figure A3.3: Raw A/E distribution mean and standard deviation against energy, with with linear and root mean square fits respectively for detector V05267A.

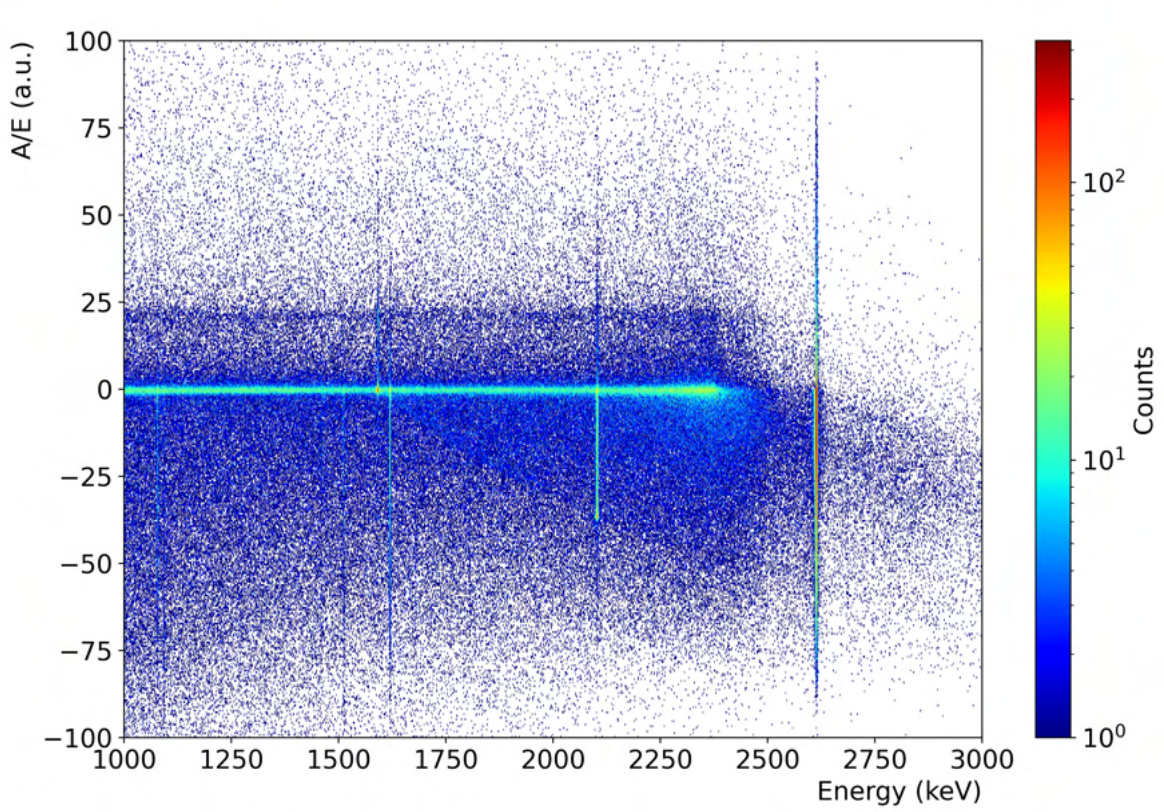


Figure A3.4: 2D Histogram of classifier A/E-Energy plot for detector V05267A.

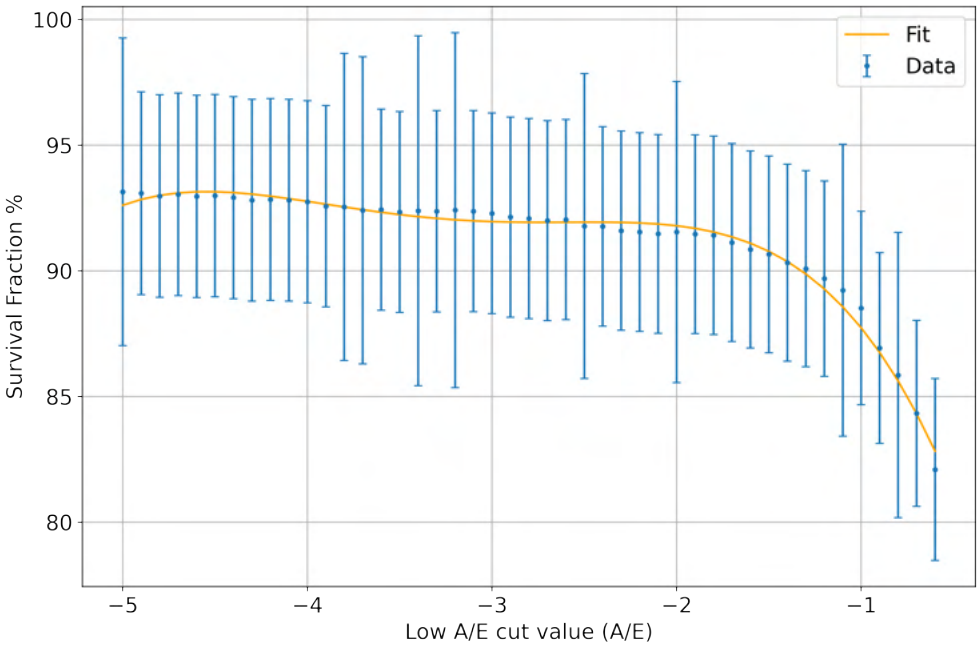


Figure A3.5: Survival fraction of DEP events about (1592 ± 10) keV against classifier A/E value. The orange line is a fourth order polynomial fit to the survival fraction data for detector V05267A.

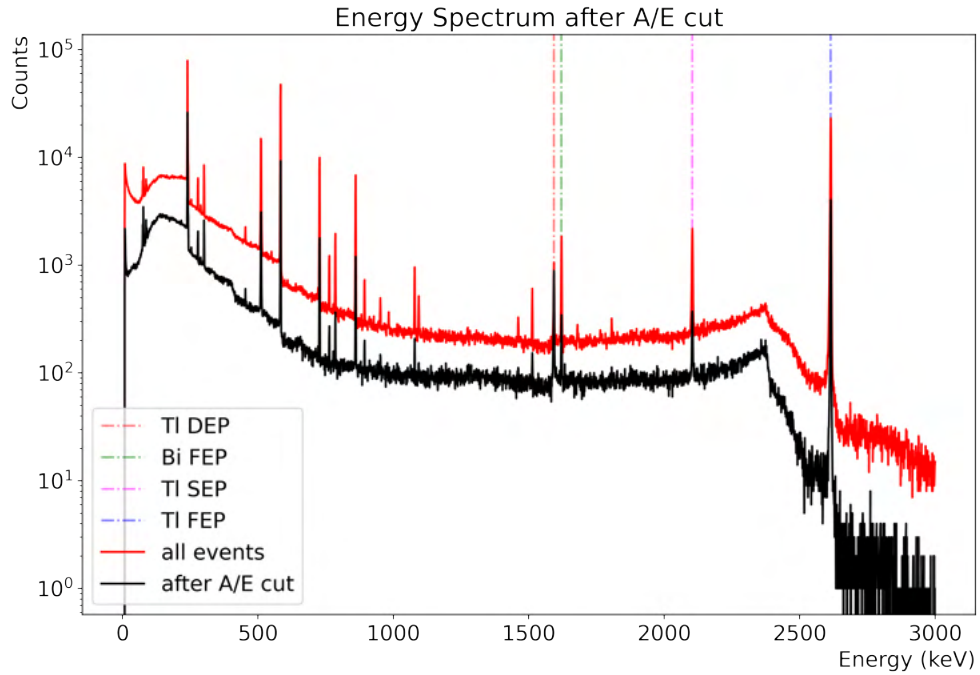


Figure A3.6: Number of events throughout the energy spectrum pre- and post-A/E cut, highlighting the FEP, SEP, and DEP regions of interest for detector V05267A.

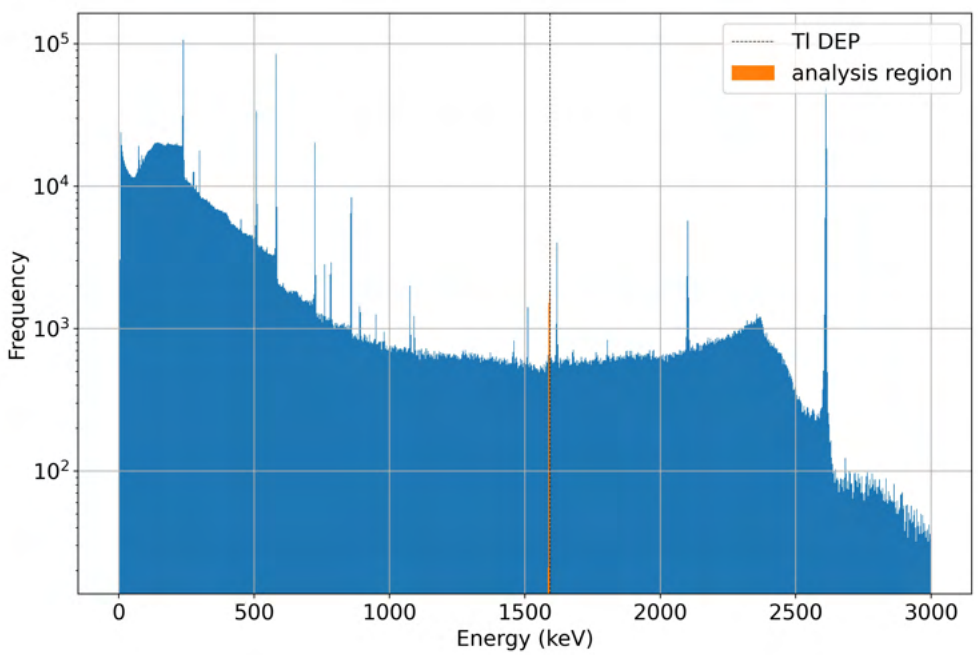


Figure A3.7: Energy spectra of Th-228 calibration dtaa for detector V05267A with analysis region centered about the Tl DEP (1592 ± 3) keV.

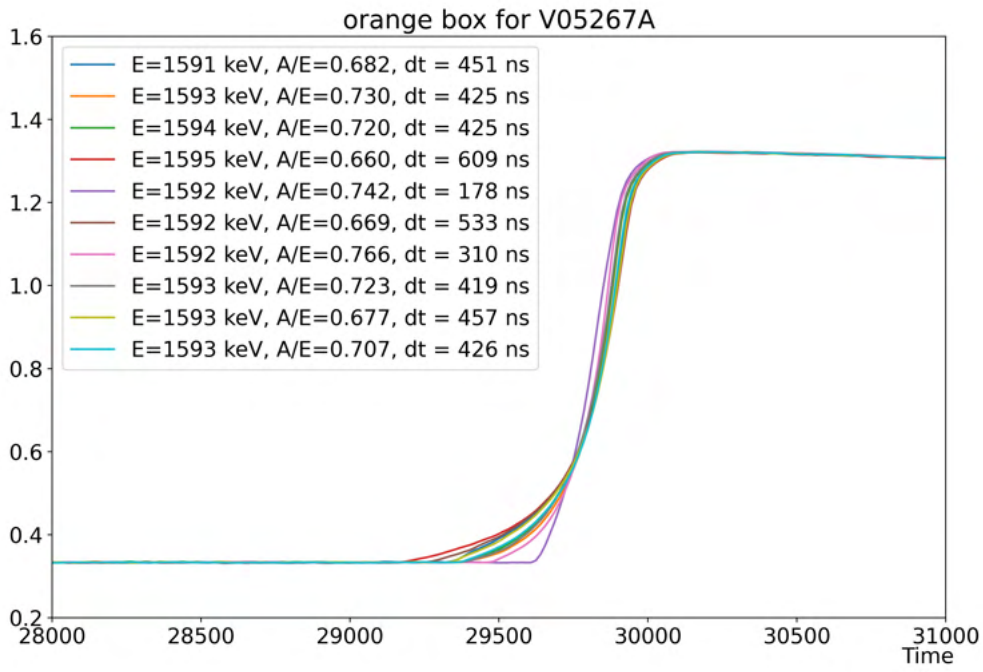


Figure A3.8: Selection of waveforms contained within the orange analysis region for detector V05267A.

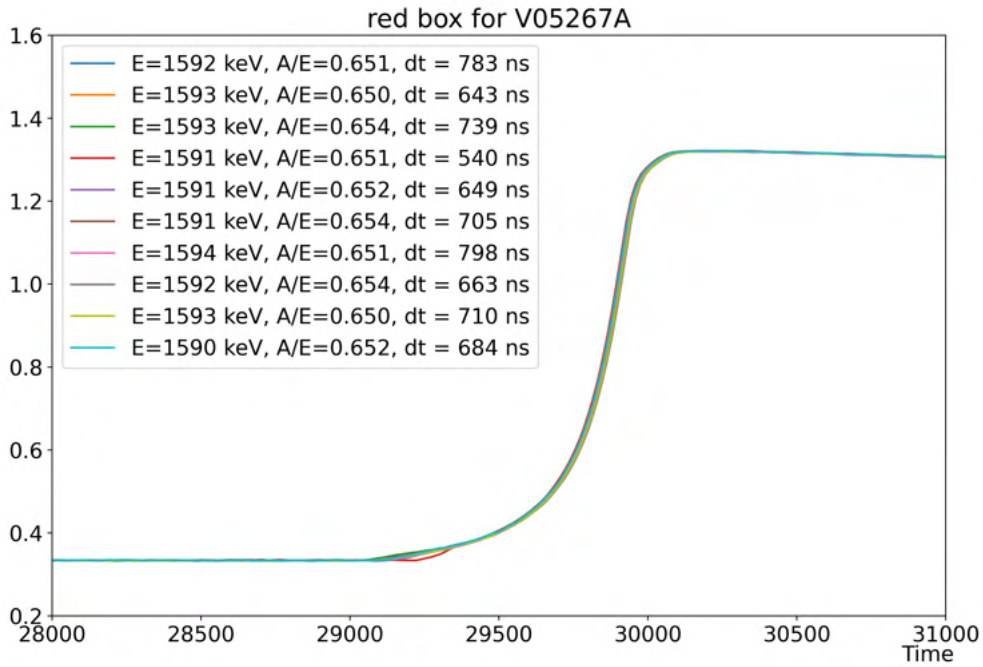


Figure A3.9: Selection of waveforms contained within the red analysis region for detector V05267A.

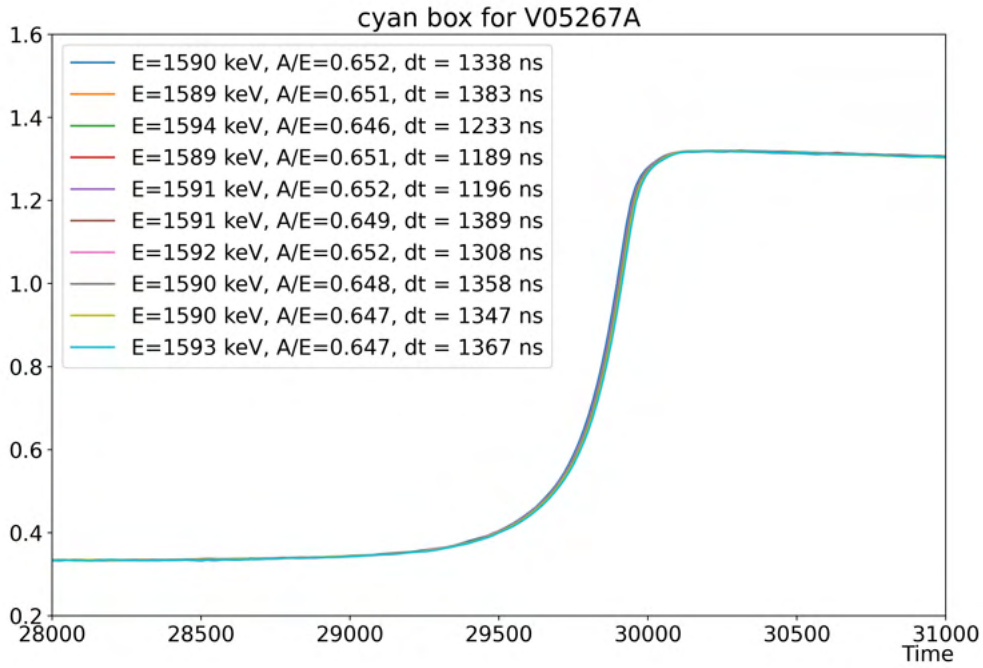


Figure A3.10: Selection of waveforms contained within the cyan analysis region for detector V05267A.

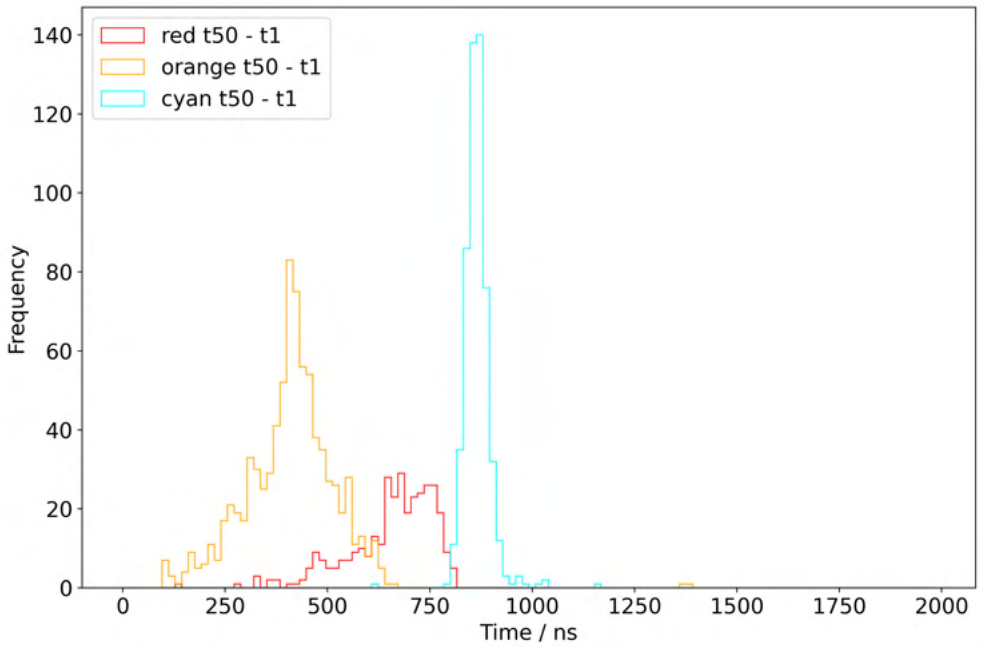


Figure A3.11: Distribution of waveforms in each analysis region for detector V05267A with a bin width of 16 ns. Where the $|t50 - t1|$ time-point range is used.

B. CODE

B1. A/E CODE

In [1]:
```
import pygama.lh5 as lh5
import matplotlib.pyplot as plt
import numpy as np
import glob
from matplotlib.colors import LogNorm
import pathlib
from iminuit import cost,Minuit
from scipy.special import erf, erfc
from scipy.stats import norm, poisson
from scipy.integrate import simps
```

In [2]:
```
import os,json
import math
from scipy.optimize import curve_fit
import pygama.analysis.histograms as pgh
import pygama.analysis.calibration as cal
import pygama.analysis.peak_fitting as pgp
import pygama.analysis.peak_fitting as pgf
import scipy.optimize as opt
import matplotlib.colors as mcolors
import matplotlib.cm as cmx
from scipy.integrate import quad
from matplotlib.lines import Line2D
from matplotlib.backends.backend_pdf import PdfPages
import numba as nb
from math import erfc
from iminuit import Minuit, cost
from iminuit.util import propagate
import sys

# Set defaults for figures
plt.rcParams['figure.figsize'] = (12, 8)
plt.rcParams['font.size'] = 16
```

In [3]:
```
kwd = {"parallel": False, "fastmath": True}
limit = np.log(sys.float_info.max)/10
```

A/E PDFs fit definition

In [4]:
```
def PDF_AoE(x, lambda_s, lambda_b, gaussian_mean, gaussian_sigma,
                        htail,htau):

    pdf = (lambda_b * gauss_tail_norm(x,gaussian_mean, gaussian_sigma,
                                      htail,htau)+\
             lambda_s *  norm_pdf(x, gaussian_mean, gaussian_sigma))
    #(1./(lambda_s+lambda_b))*
    return lambda_s+lambda_b, pdf
```

In [5]:

```
@nb.njit(**kwd)
def norm_pdf(x, mu, sigma):
    if sigma ==0: invs=np.nan
    else: invs = 1.0 / sigma
    z = (x - mu) * invs
    invnorm = 1 / np.sqrt(2 * np.pi) * invs
    return np.exp(-0.5 * z ** 2) * invnorm
```

In [6]:

```
@nb.njit(**kwd)
def nb_erf(x):
    y = np.empty_like(x)
    for i in nb.prange(len(x)):
        y[i] = erfc(x[i])
    return y

#@nb.njit(**kwd)
def gauss_tail(x, mu, sigma, tail, tau):
    """
    A gaussian tail function template
    Can be used as a component of other fit functions
    """
    x = np.asarray(x)
    tmp = ((x-mu)/tau) + ((sigma**2)/(2*tau**2))
    tail_f = np.where(tmp < limit,
                      gauss_tail_exact(x, mu, sigma, tail, tau),
                      gauss_tail_approx(x, mu, sigma, tail, tau))
    return tail_f

#@nb.njit(**kwd)
def gauss_tail_exact(x, mu, sigma, tail, tau):
    tmp = ((x-mu)/tau) + ((sigma**2)/(2*tau**2))
    abstau = np.absolute(tau)
    tmp = np.where(tmp < limit, tmp, limit)
    z = (x-mu)/sigma
    tail_f = (tail/(2*abstau)) * np.exp(tmp) * nb_erf( (tau*z + sigma)
                                                      /(np.sqrt(2)*abstau))
    return tail_f

@nb.njit(**kwd)
def gauss_tail_approx(x, mu, sigma, tail, tau):
    den = 1/(sigma + tau*(x-mu)/sigma)
    tail_f = tail * sigma * norm_pdf(x, mu, sigma) * den
                * (1.-tau*tau*den*den)
    return tail_f

def gauss_tail_norm(x,mu, sigma, htail, tau):

    xs = np.linspace(np.nanmin(x), np.nanmax(x), 10001)#
    try:
        norm_pdf_values = gauss_tail(xs, mu, sigma, htail, tau)
    except ZeroDivisionError:
        return 0
    normalisation = simps(norm_pdf_values, xs, axis = 0)
    exp_part = gauss_tail(x, mu, sigma, htail, tau)
    return exp_part / normalisation
```

In [7]:
```
@nb.njit(**kwd)
def fexpo(x,p1,p2):
    return np.exp(p1+x*p2)

def norm_fexpo(x,z,p1,p2):
    xs = np.linspace(np.nanmin(x), np.nanmax(x), 10001)
    norm_pdf_values = fexpo(xs, p1,p2)
    normalisation = simps(norm_pdf_values, xs, axis = 0)

    return z*fexpo(x,p1,p2)/normalisation
```

Energy PDFs

In [8]:
```
@nb.njit(**kwd)
def step_pdf(x,  mu, sigma, hstep):
    invs = (np.sqrt(2)*sigma)
    z = (x-mu)/invs
    step_f = 1 + hstep * nb_erf(z)
    return step_f
```

In [9]:
```
def peak_pdf(x, mu, sigma, htail, tau):
    peak = norm_pdf(x,mu,sigma)
    tail = gauss_tail(x, mu, sigma, htail, tau)
    return peak+tail
```

In [10]:
```
def norm_energy_pdf(x, pdf_func, *params):
    xs = np.arange(np.nanmin(x), np.nanmax(x), 0.01)
    norm_pdf_values = pdf_func(xs, *params)
    normalisation = simps(norm_pdf_values, xs, axis = 0)

    return pdf_func(x, *params)/normalisation
```

In [11]:
```
def radford_pdf(data, lambda_s, lambda_b, mu, sigma, hstep, htail, tau):
    pdf = (lambda_b * norm_energy_pdf(data, step_pdf, mu, sigma, hstep)
           +\lambda_s *  norm_energy_pdf(data, peak_pdf, mu, sigma,
                                        htail, tau))

    return lambda_s+lambda_b, pdf
```

A/E fit

In [12]:
```
def AoEcorrection(e,aoe,eres, display=1, plot_all=False):
```

```
comptBands_width = 20;
comptBands = np.array([940,960,980,1000,1020,1040,1130,1150,
                       1170,1190,1210,1250,1270,1290,
                      1310,1330,1370,1390,1420,1540,1650,1700,
                       1780,1810,1850,1870,1890,1910,1930,1950,
                      1970,1990,2010,2030,2050,2150,2170,2190,
                       2210,2230,2250,2270,2290])
comptBands = comptBands[::-1]
peaks = np.array([1080,1094,1459,1512, 1552, 1592,1620, 1650,
                  1670,1830,2105])
compt_aoe = np.zeros(len(comptBands))
aoe_sigmas = np.zeros(len(comptBands))
compt_aoe_err = np.zeros(len(comptBands))
aoe_sigmas_err = np.zeros(len(comptBands))
ratio = np.zeros(len(comptBands))
ratio_err = np.zeros(len(comptBands))

copper = cm = plt.get_cmap('copper')
cNorm  = mcolors.Normalize(vmin=0, vmax=len(comptBands))
scalarMap = cmx.ScalarMappable(norm=cNorm, cmap=copper)

plt.figure()
for i, band in enumerate(comptBands):
    aoe_tmp = aoe[(e>band) & (e<band+comptBands_width)
                  & (aoe>0)][:10000]

    hist, bins,var = pgh.get_hist(aoe_tmp,bins=500)
    bin_center = (bins[:-1] + bins[1:]) / 2
    pars,errs = unbinned_aoe_fit(aoe_tmp, display=display)
    compt_aoe[i] = pars[2]
    aoe_sigmas[i] = pars[3]
    compt_aoe_err[i] = errs[2]
    aoe_sigmas_err[i] = errs[3]
    ratio[i] = pars[0]/pars[1]
    ratio_err[i] = ratio[i]*np.sqrt((errs[0]/pars[0])**2
                                    + (errs[1]/pars[1])**2)
    xs = np.arange(pars[2]-4*pars[3], pars[2]+3*pars[3],
                   pars[3]/10)
    if np.isnan(errs[2])|np.isnan(errs[3])|(errs[2]==0)
                                          |(errs[3]==0): pass
    else:
        colorVal = scalarMap.to_rgba(i)
        plt.plot(xs,PDF_AoE(xs, *pars)[1], color = colorVal)
def poll(x,a,b):
    return a * x + b
plt.xlabel('A/E')
plt.ylabel("Expected Counts")
plt.title("Compton Band Fits")
cbar = plt.colorbar(cmx.ScalarMappable(norm=cNorm,
                    cmap=plt.get_cmap('copper_r')),
                    orientation='horizontal',
                    label='Compton Band Energy',
                    ticks=[0,16,32,len(comptBands)])
cbar.ax.set_xticklabels([comptBands[::-1][0],comptBands[::-1][16],
                         comptBands[::-1][32],comptBands[::-1][-1]])
plt.savefig(f'AoE figures/AoE_E_band_{det}.jpg', dpi = 400)
plt.show()
```

```
ids = np.isnan(compt_aoe_err)|np.isnan(aoe_sigmas_err)
                        |(aoe_sigmas_err==0)|(compt_aoe_err==0)
#print(compt_aoe_err, aoe_sigmas_err)

plt.figure()
plt.errorbar(comptBands[~ids], ratio[~ids], xerr=10,
             yerr = ratio_err[~ids], linestyle=' ')
plt.xlabel("Energy (keV)")
plt.ylabel("N_sig/N_bkg")
plt.show()


pars, cov = opt.curve_fit(pol1,comptBands[~ids],
                          compt_aoe[~ids],sigma = compt_aoe_err[~ids]
                          , absolute_sigma=True)
errs = np.sqrt(np.diag(cov))

sig_pars, sig_cov = opt.curve_fit(pol1,comptBands[~ids],
                                  aoe_sigmas[~ids],sigma =
                                  aoe_sigmas_err[~ids],
                                  absolute_sigma=True)
sig_errs = np.sqrt(np.diag(sig_cov))

def sigma_fit(x, a,b,c):
    return np.sqrt(a+(b/x)**c)

p0 = [0.001,10, 3]
c = cost.LeastSquares(comptBands[~ids],aoe_sigmas[~ids],
                      aoe_sigmas_err[~ids], sigma_fit)

#print(p0)
c.loss = "soft_l1"
m = Minuit(c, *p0)
m.migrad()
m.hesse()
sig_pars2 = m.values
sig_errs2 = m.errors

model = pol1(comptBands,*pars)
sig_model = pol1(comptBands,*sig_pars)
sig_model2 = sigma_fit(comptBands,*sig_pars2)

sigma = np.sqrt(eres[0]+1592*eres[1])/2.355
n_sigma = 4
peak = 1592
emin           = peak - n_sigma*sigma
emax           = peak + n_sigma*sigma
dep_pars, dep_err = unbinned_aoe_fit(aoe[(e>emin)
                                         & (e<emax) & (aoe>0)])

fig, (ax1, ax2) = plt.subplots(2, 1, constrained_layout=True,
                               sharex=True)
ax1.errorbar(comptBands[~ids]+10,compt_aoe[~ids],
               yerr=compt_aoe_err[~ids],
```

```
                xerr=10,
                label='data', linestyle= ' ')
ax1.plot(comptBands[~ids]+10,model[~ids],label='linear model')
plt.grid()
ax1.errorbar(1592, dep_pars[2], xerr = n_sigma*sigma,
                yerr = dep_err[2],
                label='DEP', color='green', linestyle= ' ')

ax1.legend(title='A/E mu energy dependence', frameon=False)

ax1.set_ylabel("raw A/E (a.u.)", ha='right', y=1)
#ax1.ylim([np.amax(model), np.amin(model)])
ax2.scatter(comptBands[~ids]+10,100*(compt_aoe[~ids]-model[~ids])
           /model[~ids], lw=1, c='b')
print(max(abs(100*(compt_aoe[~ids]-model[~ids])
              /model[~ids])), 100*(dep_pars[2]-pol1(1592,*pars))
              /pol1(1592,*pars))
print(np.mean(abs(100*(compt_aoe[~ids]-model[~ids])
                  /model[~ids]))))
ax2.scatter(1592,100*(dep_pars[2]-pol1(1592,*pars))
           /pol1(1592,*pars), lw=1, c='g')
ax2.set_ylabel("Residuals %", ha='right', y=1)
ax2.set_xlabel("Energy (keV)", ha='right', x=1)
plt.grid()
plt.savefig(f'AoE figures/aoe_energy_dependence_{det}.jpg',
           bbox_inches='tight', transparent=True, dpi = 400)
plt.show()

fig, (ax1, ax2) = plt.subplots(2, 1, constrained_layout=True,
                               sharex=True)
plt.grid()
ax1.errorbar(comptBands[~ids]+10, aoe_sigmas[~ids],
             yerr= aoe_sigmas_err[~ids],
            xerr = 10, label='data', linestyle= ' ')
ax1.plot(comptBands[~ids],sig_model2[~ids],
         label=f'sqrt model: sqrt({sig_pars2[0]:1.4f}
         +({sig_pars2[1]:1.1f}/E)^{sig_pars2[2]:1.1f})')
ax1.errorbar(1592, dep_pars[3], xerr = n_sigma
             *sigma,yerr = dep_err[3], label='DEP', color='green')
ax1.set_ylabel("A/E stdev (a.u.)", ha='right', y=1)
plt.grid()
ax1.legend(title='A/E stdev energy dependence', frameon=False)
ax2.scatter(comptBands[~ids]+10,
            100*(aoe_sigmas[~ids]-sig_model2[~ids])
            /sig_model2[~ids], lw=1, c='b')
ax2.scatter(1592,100*(dep_pars[3]-sigma_fit(1592,*sig_pars2))
            /sigma_fit(1592,*sig_pars2), lw=1, c='g')
print(max(abs(100*(aoe_sigmas[~ids]-sig_model2[~ids])
              /sig_model2[~ids])), 100
              *(dep_pars[3]-sigma_fit(1592,*sig_pars2))
              /sigma_fit(1592,*sig_pars2))
print(np.mean(abs(100*(aoe_sigmas[~ids]-sig_model2[~ids])
                        /sig_model2[~ids]))))
ax2.set_ylabel("Residuals %", ha='right', y=1)
ax2.set_xlabel("Energy (keV)", ha='right', x=1)
plt.grid()
plt.savefig(f'AoE figures/aoe_energy_deviation_{det}.jpg',
```

```
                bbox_inches='tight', transparent=True, dpi = 400)
    plt.show()


    return pars, sig_pars2, comptBands[~ids], aoe_sigmas[~ids],
                                            aoe_sigmas_err[~ids]
```

Energy fit

In [13]:

```
def unbinned_aoe_fit(aoe, display=0, verbose=False):
    aoe_len = len(aoe)
    hist, bins,var = pgh.get_hist(aoe,bins=500)
    bin_centers = (bins[:-1]+bins[1:])/2

    pars, cov = pgf.gauss_mode_max(hist, bins, var)
    mu = bin_centers[np.argmax(hist)]
    amp = np.amax(hist)

    sigma=pgh.get_fwhm(hist, bins)[0]/2.355
    ls_guess = 2*np.sum(hist[(bin_centers>mu)
                             &(bin_centers<(mu+2.5*sigma))])
    #sigma=0.0035
    def gauss(x,z,mu,sigma):
        return z * norm.pdf(x,mu,sigma)
    c1_min= mu-2*sigma #0.495
    c1_max= mu+5*sigma #0.52
    c1 = cost.UnbinnedNLL(aoe[(aoe<c1_max)&(aoe>c1_min)], gauss)
    m1 = Minuit(c1, ls_guess, mu,sigma)
    m1.limits = [(0, len(aoe[(aoe<c1_max)&(aoe>c1_min)])),
                 (mu*0.8, mu*1.2),(0.8*sigma,sigma*1.2)]
    m1.migrad()
    ls_guess =m1.values[0]
    mu = m1.values[1]
    sigma = m1.values[2]

    fmin=  mu-15*sigma #0.45
    fmax = mu+5*sigma #0.52
    c2_max = mu-4*sigma
    #print(fmin,fmax)
    c2 = cost.UnbinnedNLL(aoe[(aoe<c2_max)&(aoe>fmin)], norm_fexpo)
    m2 = Minuit(c2, len(aoe[(aoe<c2_max)&(aoe>fmin)]), -20,10)
    m2.limits=[(0,len(aoe[(aoe<c2_max)&(aoe>fmin)])),(-100,0),(0,20)]
    m2.migrad()
    f = m2.values[2]
    pars = [mu,sigma,amp,0,f,10]

    bg_guess = len(aoe[(aoe<fmax)&(aoe>fmin)])-ls_guess
    x0 = [ls_guess,bg_guess,pars[0],pars[1],pars[4],pars[5]]
    if verbose:print(x0)

    c = cost.ExtendedUnbinnedNLL(aoe[(aoe<fmax)&(aoe>fmin)], PDF_AoE)

    m = Minuit(c, *x0)
    m.migrad()
```

```
    m.hesse()
    if verbose:print(m.values)
    if display>1:
        plt.figure()
        xs = np.linspace(fmin,fmax,1000)
        counts, bins, bars = plt.hist(aoe[(aoe<fmax)&(aoe>fmin)],
                                      bins=400, histtype='step')
        dx = np.diff(bins)
        plt.plot(xs, PDF_AoE(xs,*m.values)[1]* dx[0], label="Total PDF")
        #plt.yscale('log')
        plt.plot(xs , (m.values[1])*gauss_tail_norm(xs,*m.values[2:])
                 * dx[0], label="Tail PDF")
        plt.plot(xs , (m.values[0])*norm_pdf(xs,  *m.values[2:4])
                 * dx[0], label="Gaussian PDF")
        plt.legend(loc='upper left')
        plt.title("Fit Components")
        plt.grid()
        plt.show()

        plt.figure()
        bin_centers= (bins[1:]+bins[:-1])/2
        plt.plot(bin_centers, (PDF_AoE(bin_centers,*m.values)[1]
                               * dx[0]) - counts)
        plt.title("Residuals")
        plt.grid()
        plt.show()
        return m.values, m.errors
    else: return m.values, m.errors
```

In [14]:
```
def unbinned_energy_fit(energy, peak, fwhm):
    energy_len = len(energy)
    hist, bins,var = pgh.get_hist(energy,dx=0.1,
                                  range= (np.amin(energy),
                                          np.amax(energy)))
    sigma = fwhm/2.355
    bg0 = np.sum(hist[-5:])/5
    step = np.sum(hist[:5])/5 - bg0
    htail = 1./5
    tau = 6.*sigma

    # now compute amp and return
    height = np.amax(hist)
    height -= (bg0 + step/2)
    amp = height / (htail*0.87/35 + (1-htail)/(sigma*np.sqrt(2*np.pi)))
    hstep = step/(2*amp)
    guess = [peak, sigma, hstep, htail, tau, bg0, amp]

    x0 = [guess[-1]*guess[1]*10,energy_len-(guess[-1]
                                            *guess[1]*10),*guess]
    x0= x0[:-2]
    bounds = [(0,energy_len),(0,energy_len), (x0[2]-2
                                              *x0[3], x0[2]+2*x0[3]),
              (0, x0[3]*2), (0, 1), (0, 1),
              (x0[3], 100*x0[3])]
    c = cost.ExtendedUnbinnedNLL(energy, radford_pdf)
    m = Minuit(c, *x0)
    m.limits = bounds
    m.migrad()
    m.hesse()

    return m.values, m.errors
```

Load in Data

In [15]:
```
energy_params = ['cuspEmax', 'cuspEmax_ctc', 'A_max','dt_eff']
```

In [16]:
```
det      = 'V05612B'

datatype = 'th_HS2_lat_psa'
run = '001'
energy_param = energy_params[1]
```

In [17]:
```
datapath   = f'/unix/legend/shared/test_data/{det}/tier2/{datatype}'
files      = os.listdir(datapath)
files.sort()
for i,file in enumerate(files):
    files[i] = os.path.join(datapath,file)
files = files[0:10]
```

In [18]:
```
print(files)
```

```
['/unix/legend/shared/test_data/V05612B/tier2/th_HS2_lat_psa/char_data-V056
12B-th_HS2_lat_psa-run0001-200911T113049_tier2.lh5', '/unix/legend/shared/t
est_data/V05612B/tier2/th_HS2_lat_psa/char_data-V05612B-th_HS2_lat_psa-run0
001-200911T114054_tier2.lh5', '/unix/legend/shared/test_data/V05612B/tier2/
th_HS2_lat_psa/char_data-V05612B-th_HS2_lat_psa-run0001-200911T115056_tier2
.lh5', '/unix/legend/shared/test_data/V05612B/tier2/th_HS2_lat_psa/char_dat
a-V05612B-th_HS2_lat_psa-run0001-200911T120057_tier2.lh5', '/unix/legend/sh
ared/test_data/V05612B/tier2/th_HS2_lat_psa/char_data-V05612B-th_HS2_lat_ps
a-run0001-200911T121102_tier2.lh5', '/unix/legend/shared/test_data/V05612B/
tier2/th_HS2_lat_psa/char_data-V05612B-th_HS2_lat_psa-run0001-200911T122102
_tier2.lh5', '/unix/legend/shared/test_data/V05612B/tier2/th_HS2_lat_psa/ch
ar_data-V05612B-th_HS2_lat_psa-run0001-200911T123104_tier2.lh5', '/unix/leg
end/shared/test_data/V05612B/tier2/th_HS2_lat_psa/char_data-V05612B-th_HS2_
lat_psa-run0001-200911T124109_tier2.lh5', '/unix/legend/shared/test_data/V0
5612B/tier2/th_HS2_lat_psa/char_data-V05612B-th_HS2_lat_psa-run0001-200911T
125113_tier2.lh5', '/unix/legend/shared/test_data/V05612B/tier2/th_HS2_lat_
psa/char_data-V05612B-th_HS2_lat_psa-run0001-200911T130113_tier2.lh5']
```

In [19]:
```
calpath = f'/unix/legend/shared/aoe/{det}.json'
with open(calpath, 'r') as o:
    cal_dict = json.load(o)
cal_pars = cal_dict[energy_param]['Calibration_pars']
eres_pars = [cal_dict[energy_param]['m0'], cal_dict[energy_param]['m1']]
print(cal_pars, eres_pars)
```

```
[0.07057924123918036, 1.8047795200005456] [0.36666104815033235, 0.002618975
5727986145]
```

In [20]:
```
print(len(files),'files found in',datapath)
uncal_pass = lh5.load_nda(files,energy_params,'raw',verbose=False)
print("done")
```

```
10 files found in /unix/legend/shared/test_data/V05612B/tier2/th_HS2_lat_ps
a
done
```

In [21]:
```
ecal_pass = pgp.poly(uncal_pass[energy_param], cal_pars)
```

In [22]:
```
curr = uncal_pass['A_max']
aoe = np.divide(curr,pgp.poly(uncal_pass['cuspEmax'], cal_pars))
print(len(aoe))
```

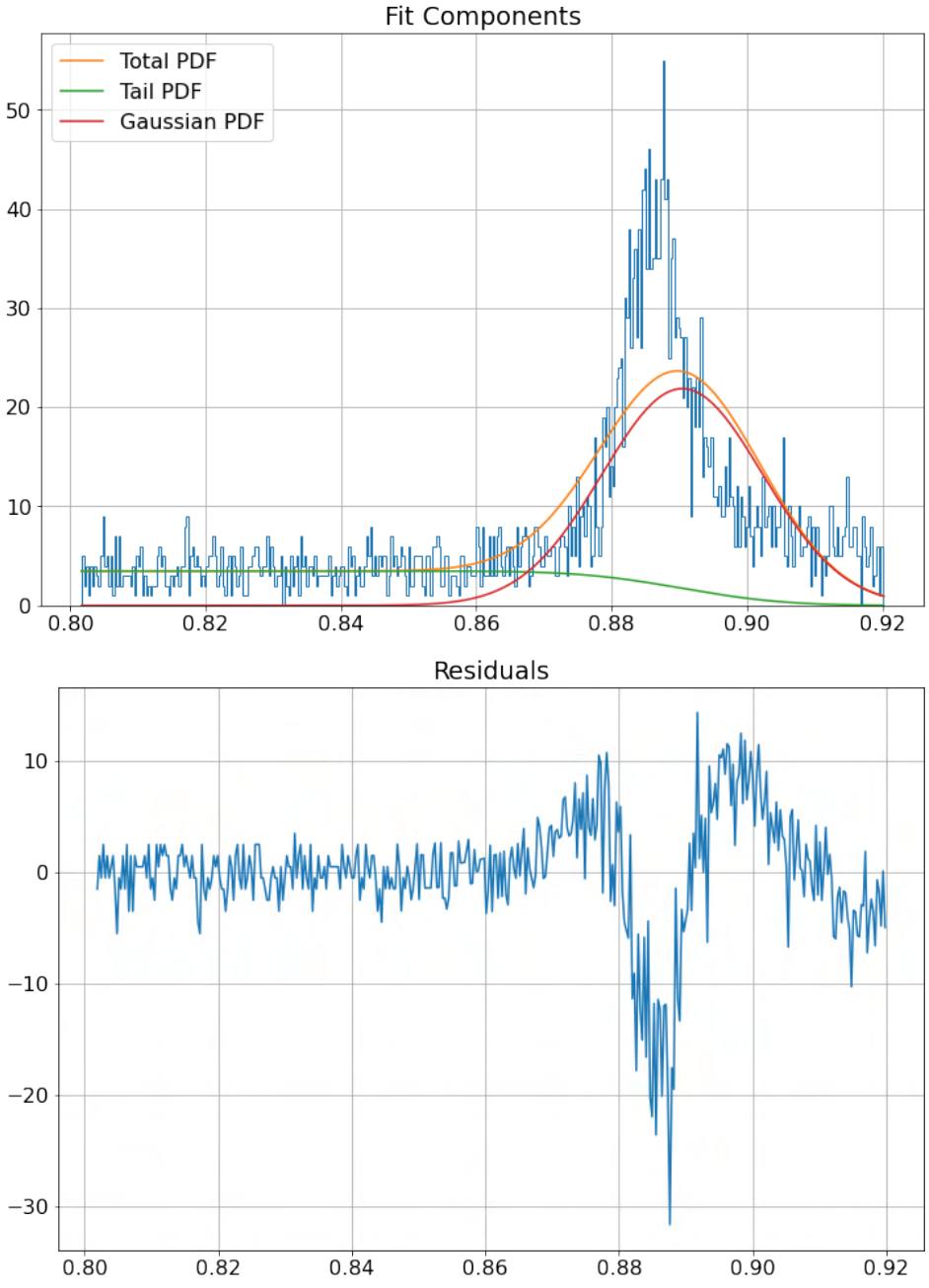
```
2783396
```

In [23]:
```
df = lh5.load_dfs(files, ['cuspEmax_ctc' , 'A_max', 'dt_eff',
                          'tp_0_est', 'tp_01', 'tp_10', 'tp_20', 'tp_50',
                          'tp_90', 'tp_95', 'tp_99', 'tp_max'], '/raw');
trapE = df['cuspEmax_ctc'];
adc = cal_pars[0];
cal_E = trapE*adc

E_DEP = 1592
pm = 3
DEP_range_cal = [(E_DEP-pm), (E_DEP+pm)]
energy_selection_cal = (ecal_pass>DEP_range_cal[0]) & (ecal_pass<DEP_range_
```

```
loading data for /unix/legend/shared/test_data/V05612B/tier2/th_HS2_lat_psa
/char_data-V05612B-th_HS2_lat_psa-run0001-200911T113049_tier2.lh5 /unix/leg
end/shared/test_data/V05612B/tier2/th_HS2_lat_psa/char_data-V05612B-th_HS2_
lat_psa-run0001-200911T114054_tier2.lh5 /unix/legend/shared/test_data/V0561
2B/tier2/th_HS2_lat_psa/char_data-V05612B-th_HS2_lat_psa-run0001-200911T115
056_tier2.lh5 /unix/legend/shared/test_data/V05612B/tier2/th_HS2_lat_psa/ch
ar_data-V05612B-th_HS2_lat_psa-run0001-200911T120057_tier2.lh5 /unix/legend
/shared/test_data/V05612B/tier2/th_HS2_lat_psa/char_data-V05612B-th_HS2_lat
_psa-run0001-200911T121102_tier2.lh5 /unix/legend/shared/test_data/V05612B/
tier2/th_HS2_lat_psa/char_data-V05612B-th_HS2_lat_psa-run0001-200911T122102
_tier2.lh5 /unix/legend/shared/test_data/V05612B/tier2/th_HS2_lat_psa/char_
data-V05612B-th_HS2_lat_psa-run0001-200911T123104_tier2.lh5 /unix/legend/sh
ared/test_data/V05612B/tier2/th_HS2_lat_psa/char_data-V05612B-th_HS2_lat_ps
a-run0001-200911T124109_tier2.lh5 /unix/legend/shared/test_data/V05612B/tie
r2/th_HS2_lat_psa/char_data-V05612B-th_HS2_lat_psa-run0001-200911T125113_ti
er2.lh5 /unix/legend/shared/test_data/V05612B/tier2/th_HS2_lat_psa/char_dat
a-V05612B-th_HS2_lat_psa-run0001-200911T130113_tier2.lh5
```

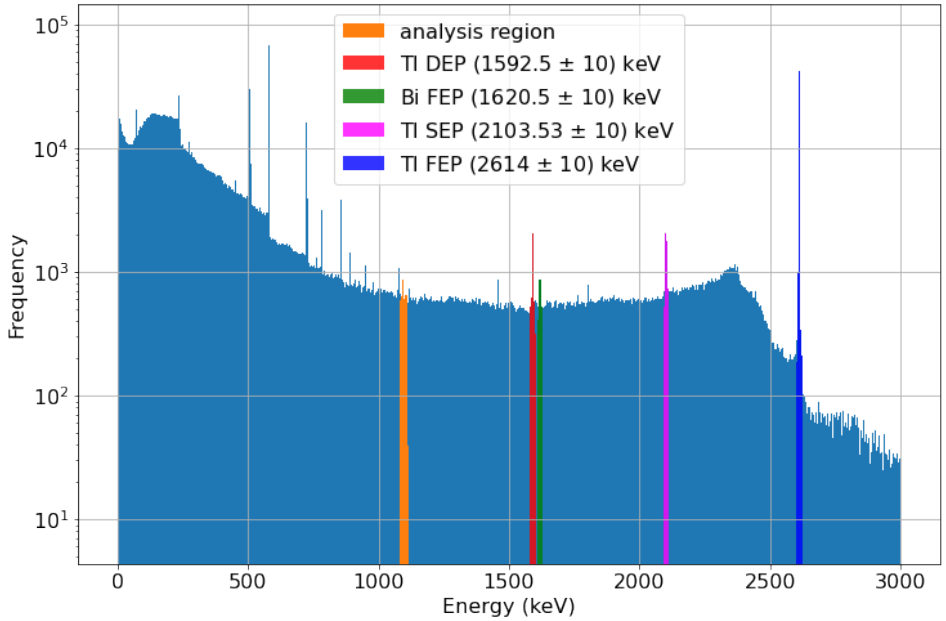
Fit example

In [24]:
```
comptb = 1100
pm1 = 15
results,errs = unbinned_aoe_fit(aoe[(ecal_pass>comptb-pm1)
                                    &(ecal_pass<comptb+pm1)],
                                display=2,verbose=False)
```


Fit Components
Total PDF
Tail PDF
Gaussian PDF
0
10
20
30
40
50
0.80
0.82
0.84
0.86
0.88
0.90
0.92
Residuals
10
0
−10
−20
−30
0.80
0.82
0.84
0.86
0.88
0.90
0.92

In [25]:

```
peaks_id = ['Tl DEP', 'Bi FEP', 'Tl SEP', 'Tl FEP']
peaks = [1592.5,1620.5,2103.53,2614]
pm2 = 10
colours = ['red', 'green', 'magenta', 'blue']


cal_E.hist(bins=1000, range=(0, 3000))
cal_E[(ecal_pass>comptb-pm1)&(ecal_pass<comptb+pm1)].hist(bins=1000,
                                range=(0, 3000), label='analysis region')
[cal_E[(ecal_pass>peaks[i]-pm2)&(ecal_pass<peaks[i]+pm2)].hist(
                                bins=1000, range=(0, 3000),
        label=f'{peaks_id[i]} ({peaks[i]} $\pm$ {pm2}) keV',
                    alpha = 0.8, color = colours[i])
 for i in range(0,len(peaks))]


plt.yscale('log')
plt.legend(loc='best')
plt.xlabel('Energy (keV)')
plt.ylabel('Frequency')
plt.savefig(f'AoE figures/E_Spectrum_{det}', dpi = 400);
```

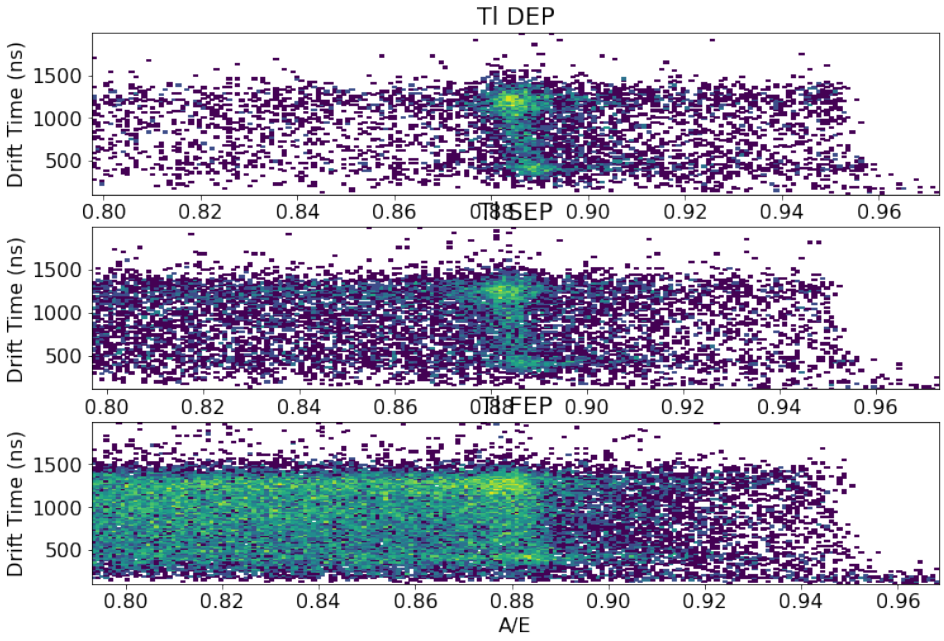


Drift time plots

In [26]:
```
def plot_dt_dep(aoe, energy,dt, erange,title):

    hist, bins,var = pgh.get_hist(aoe[(energy>erange[0])
                                      &(energy<erange[1])],bins=500)
    bin_cs = (bins[1:]+bins[:-1])/2
    mu = bin_cs[np.argmax(hist)]
    aoe_range = [mu*0.9, mu*1.1]

    idxs = (energy>erange[0]) &(energy<erange[1])
                  &(aoe>aoe_range[0])
                  &(aoe<aoe_range[1])&(dt<2000)
    plt.hist2d(aoe[idxs], dt[idxs], bins=[200,100], norm=LogNorm())
    plt.ylabel('Drift Time (ns)')
    plt.xlabel('A/E')
    plt.title(title)
```

In [27]:
```
dt = uncal_pass["dt_eff"]
```

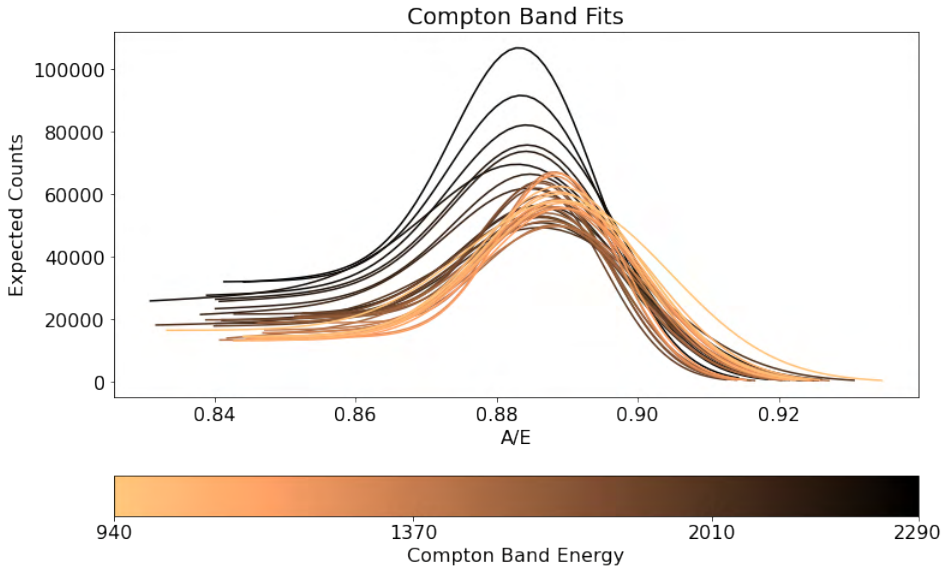
In [28]:
```
plt.figure()
plt.subplot(3,1,1)
plot_dt_dep(aoe, ecal_pass, dt, [1572,1612], f'Tl DEP')
plt.subplot(3,1,2)
plot_dt_dep(aoe, ecal_pass, dt, [2070,2130], f'Tl SEP')
plt.subplot(3,1,3)
plot_dt_dep(aoe, ecal_pass, dt, [2550,2670], f'Tl FEP')
plt.show()
```

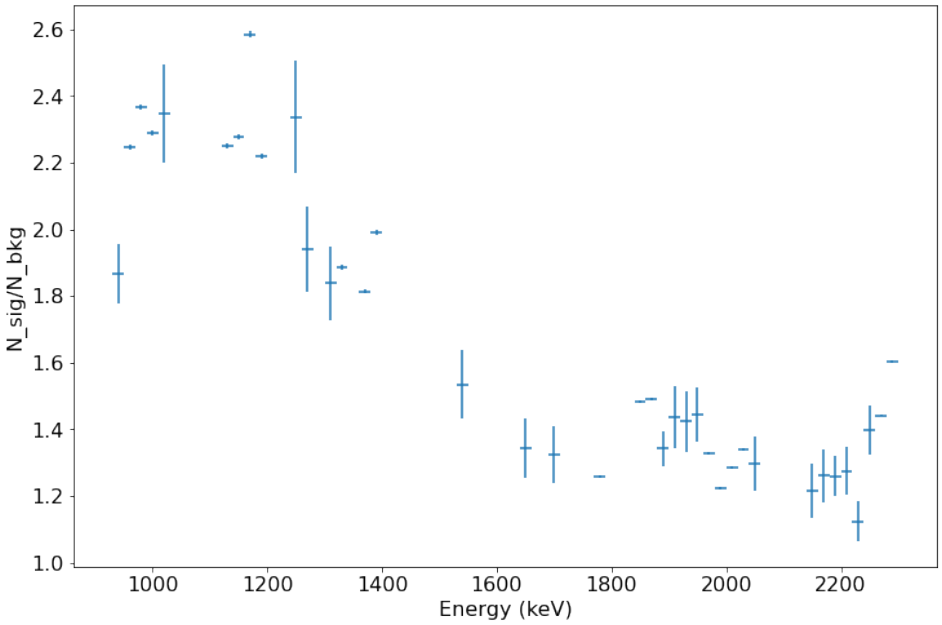


A/E Correction/ Calibration

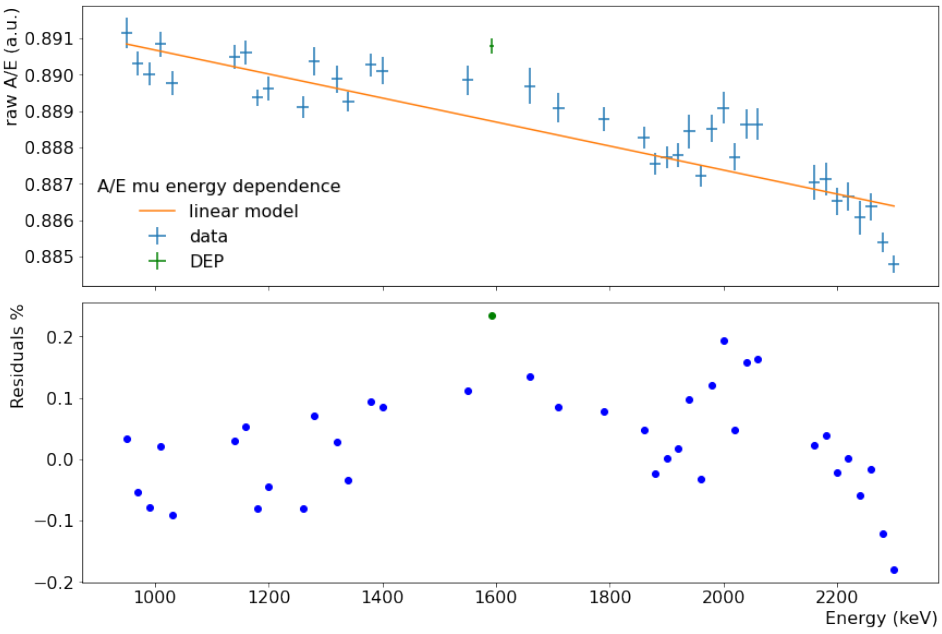
In [29]:
```
mu_pars,sigma_pars, es,sigs,serrs =AoEcorrection(ecal_pass,
                                        aoe,eres_pars, display=1)
```

```
<ipython-input-6-4e722906b952>:45: RuntimeWarning: invalid value encountere
d in true_divide
  return exp_part / normalisation
```

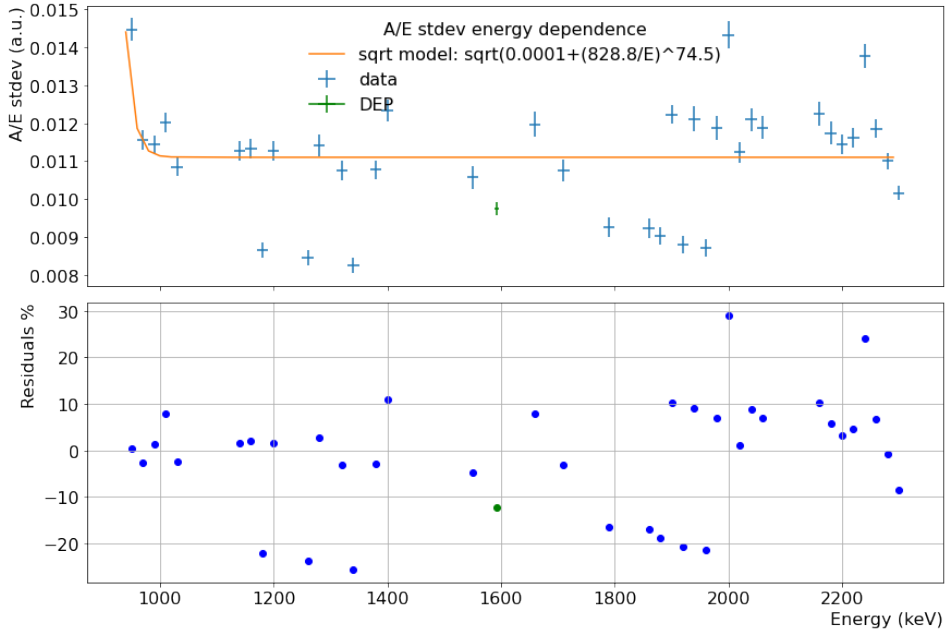





```
<ipython-input-12-0cb592760c55>:72: RuntimeWarning: invalid value encounter
ed in sqrt
  return np.sqrt(a+(b/x)**c)
0.19321134938344098 0.23492380925167047
0.06974409540191004
```




```
29.110354948453754 -12.20311735463444
9.428197802654934
```



From this we can build the classifier

In [30]:

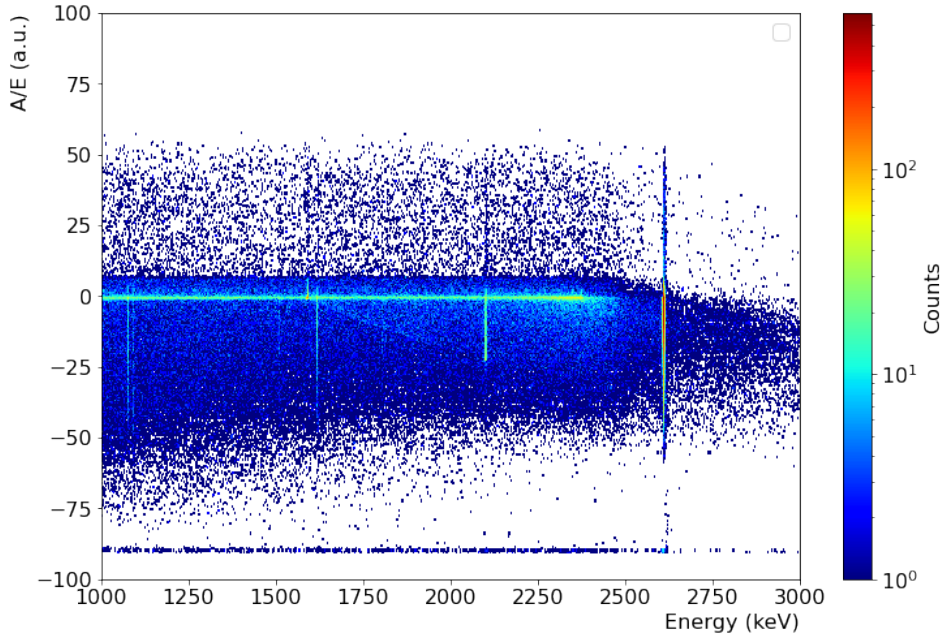
```
classifier = aoe/(mu_pars[0]*ecal_pass + mu_pars[1])
print(mu_pars)
classifier = (classifier - 1)/ np.sqrt(sigma_pars[0] +
                                       (sigma_pars[1]/ecal_pass)**sigma_pars[2])
print(sigma_pars)
```

```
[-3.29829059e-06  8.93942038e-01]
<ValueView a=0.00012322175403684866 b=828.8438680876363 c=74.54821439586617
>
<ipython-input-30-77e849a6617b>:3: RuntimeWarning: overflow encountered in
power
  classifier = (classifier - 1)/ np.sqrt(sigma_pars[0] + (sigma_pars[1]/eca
l_pass)**sigma_pars[2])
<ipython-input-30-77e849a6617b>:3: RuntimeWarning: invalid value encountere
d in power
  classifier = (classifier - 1)/ np.sqrt(sigma_pars[0] + (sigma_pars[1]/eca
l_pass)**sigma_pars[2])
```

In [31]:

```
plt.figure()
plt.hist2d(ecal_pass, classifier, bins=[2000,400],
           range=[[1000, 3000], [-100,100]], norm=LogNorm(), cmap='jet')
cbar = plt.colorbar()
plt.xlabel("Energy (keV)", ha='right', x=1)
plt.ylabel("A/E (a.u.)", ha='right', y=1)
cbar.ax.set_ylabel('Counts')

alpha = 0.8
energy = [1592.5, 2103.53, 2614.5]
lcolour = ['red', 'magenta', 'tab:orange']
peaks = ['DEP','SEP','FEP']
lpm = 10
llabel = [(f'Tl {peaks[i]} ({energy[i]} $\pm$ {lpm}) keV')
          for i in range(0,3)]
plt.legend(loc='best')
plt.savefig(f'AoE figures/Classifier_{det}.jpg', dpi = 400)
plt.show()
```

```
No handles with labels found to put in legend.
```



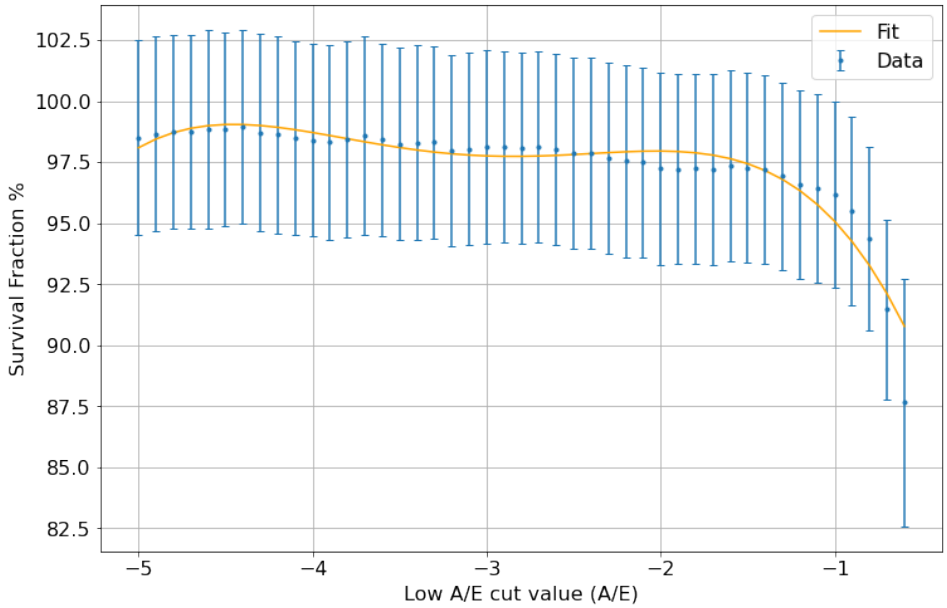
Cut Determination

In [32]:

```
def get_aoe_cut_fit(energy,aoe,peak,eres,dep_acc, display=1):
    fwhm = np.sqrt(eres[0]+peak*eres[1])
    min_range = peak-10*fwhm
    max_range = peak+10*fwhm
    #print(min_range, max_range)

    peak_energy = energy[(energy>min_range)&(energy<max_range)][:10000]
    peak_aoe = aoe[(energy>min_range)&(energy<max_range)][:10000]
    cut_vals = np.arange(-5,-0.5, 0.1)
    pars, errors = unbinned_energy_fit(peak_energy, peak, fwhm)
    pc_n = pars[0]
    pc_err = errors[0]
    sfs = []
    sf_errs=[]
    for cut_val in cut_vals:
        idxs = peak_aoe>cut_val
        cut_pars,ct_errs = unbinned_energy_fit(peak_energy[idxs],
                                               peak, fwhm)
        ct_n = cut_pars[0]
        ct_err = ct_errs[0]
        sf = (ct_n/pc_n)*100
        sfs.append(sf)
        err = sf*np.sqrt((pc_err/pc_n)**2 + (ct_err/ct_n)**2)
        sf_errs.append(err)
    ids =  (sf_errs<(1.5*sf_errs[0]))&(~np.isnan(sf_errs))
    fit = np.polynomial.polynomial.polyfit( cut_vals[ids],
                      np.array(sfs)[ids],w=1
                          /np.array(sf_errs)[ids], deg=4)

    if display>0:
        plt.figure()
        plt.errorbar(cut_vals, sfs, yerr=sf_errs, label='Data',
                     fmt = '.', capsize = 3)
        plt.plot(cut_vals,
                 np.polynomial.polynomial.polyval(cut_vals, fit) ,
                 label='Fit', color = 'orange')
        plt.xlabel('Low A/E cut value (A/E)')
        plt.ylabel('Survival Fraction %')
        plt.legend(loc='best')
        plt.grid()
        plt.savefig(f'AoE figures/Survival_Fraction_{det}', dpi = 400)
        plt.show()
    xs = np.arange(-5,-0.5,0.01)
    p = np.polynomial.polynomial.polyval(xs, fit)
    cut_val = xs[np.argmin(np.abs(p-(100*dep_acc)))]

    return cut_val
```

In [33]:

```
cut = get_aoe_cut_fit(ecal_pass,classifier,1592,eres_pars, 0.9, display=1)
```



Sweep through classifier values to find the 90% survival fraction

Calculate Survival Fractions

In [34]:

```
def get_compton_sf(energy,aoe, aoe_cut_val, peak,eres,display=1):
    fwhm = np.sqrt(eres[0]+peak*eres[1])

    emin           = peak - 2*fwhm
    emax           = peak + 2*fwhm
    sfs = []
    final_cut_vals=[]
    aoe = aoe[(energy>emin) & (energy<emax)]
    cut_vals = np.arange(-5,0,0.2)
    for cut_val in cut_vals:
        sf = 100*len(aoe[(aoe>cut_val)])/len(aoe)
        sfs.append(sf)
        final_cut_vals.append(cut_val)
        if sf<0.5:
            break
    sf = 100*len(aoe[(aoe>cut)])/len(aoe)
    return sf, final_cut_vals,sfs
```

In [35]:
```
def get_sf(energy,aoe,aoe_cut_val, peak,fit_width, eres,display=1):
    fwhm = np.sqrt(eres[0]+peak*eres[1])
    min_range = peak-fit_width[0]
    max_range = peak+fit_width[1]
    peak_energy = energy[(energy>min_range)&(energy<max_range)][:50000]
    peak_aoe = aoe[(energy>min_range)&(energy<max_range)][:50000]
    pars, errors = unbinned_energy_fit(peak_energy, peak, fwhm)
    pc_n = pars[0]
    pc_err = errors[0]
    sfs = []
    sf_errs=[]
    final_cut_vals=[]

    cut_vals = np.arange(-5,0,0.2)
    #cut_vals = np.append(cut_vals, aoe_cut_val)

    for cut_val in cut_vals:
        idxs = peak_aoe>cut_val
        cut_pars,ct_errs = unbinned_energy_fit(peak_energy[idxs],  peak, fw
        ct_n = cut_pars[0]
        ct_err = ct_errs[0]
        sf = (ct_n/pc_n)*100
        sfs.append(sf)
        err = sf*np.sqrt((pc_err/pc_n)**2 + (ct_err/ct_n)**2)
        sf_errs.append(err)
        final_cut_vals.append(cut_val)
        if sf<0.5:
            break

    idxs = peak_aoe>aoe_cut_val
    cut_pars,ct_errs = unbinned_energy_fit(peak_energy[idxs],  peak, fwhm)
    ct_n = cut_pars[0]
    ct_err = ct_errs[0]
    sf = (ct_n/pc_n)*100
    sf_err = sf*np.sqrt((pc_err/pc_n)**2 + (ct_err/ct_n)**2)
    return sf, sf_err, final_cut_vals, sfs,sf_errs
```

Here are the peak definitions with their energy

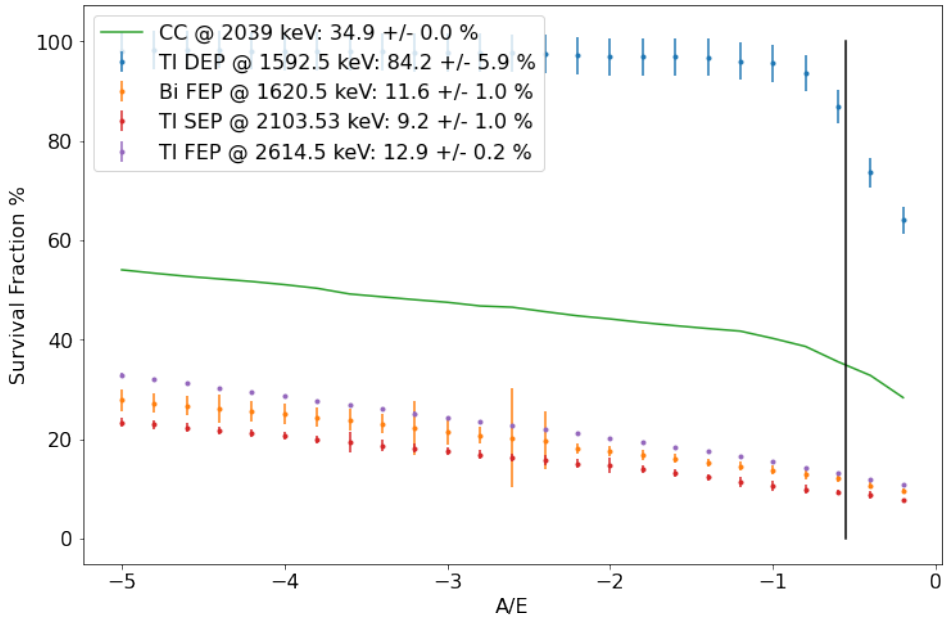
In [36]:
```
def get_peak_label(peak):
    if peak == 2039:
        return 'CC @'
    elif peak == 1592.5:
        return 'Tl DEP @'
    elif peak == 1620.5:
        return 'Bi FEP @'
    elif peak == 2103.53:
        return 'Tl SEP @'
    elif peak == 2614.5:
        return 'Tl FEP @'
```

Determining cuts:

In [37]:

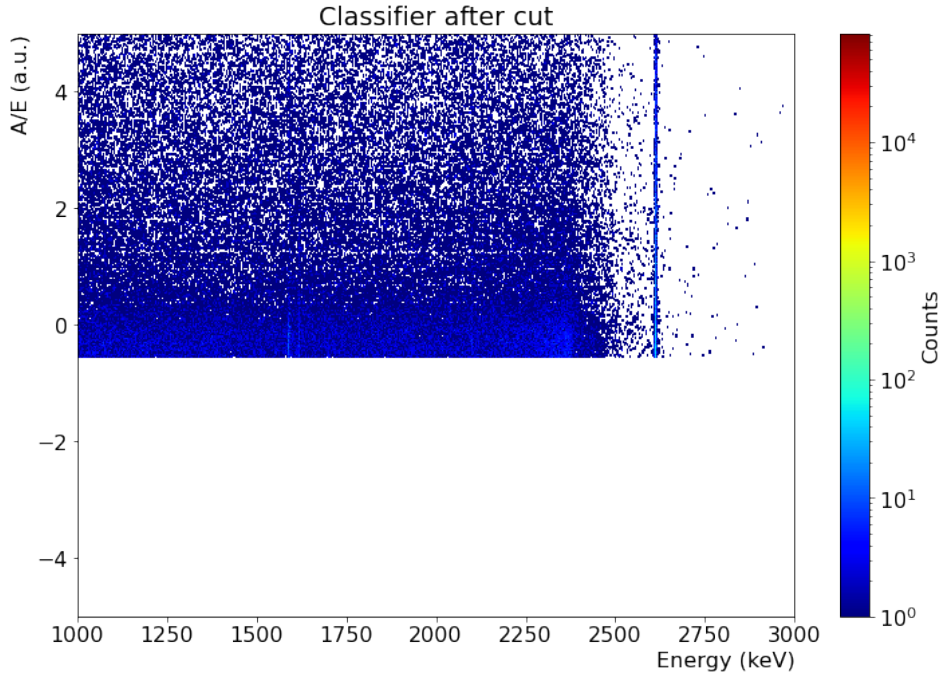
```
print("  Compute survival fractions: ")
peaks_of_interest = [1592.5, 1620.5, 2039, 2103.53, 2614.50]
sf = np.zeros(len(peaks_of_interest))
sferr = np.zeros(len(peaks_of_interest))

fit_widths = [(40,25),(25,40), (0,0),(25,40), (50,50)]
plt.figure()


for i,peak in enumerate(peaks_of_interest):
    if peak == 2039:
        sf[i], cut_vals,sfs = get_compton_sf(ecal_pass,classifier,cut, peal
        sferr[i] =0
        plt.plot(cut_vals, sfs, label = f'{get_peak_label(peak)} {peak} keV
    else:
        sf[i], sferr[i],cut_vals, sfs,sf_errs = get_sf(ecal_pass,classifier
        plt.errorbar(cut_vals, sfs, yerr=sf_errs, fmt = '.',
                     label = f'{get_peak_label(peak)} {peak} keV: {sf[i]:2.1

    print(f'{peak}keV: {sf[i]:2.1f} +/- {sferr[i]:2.1f} %')

plt.plot([cut,cut],[0,100], color = 'k')
plt.xlabel('A/E')
plt.ylabel('Survival Fraction %')
plt.legend(loc='upper left')
plt.show()
```

```
  Compute survival fractions:
1592.5keV: 84.2 +/- 5.9 %
1620.5keV: 11.6 +/- 1.0 %
2039keV: 34.9 +/- 0.0 %
2103.53keV: 9.2 +/- 1.0 %
2614.5keV: 12.9 +/- 0.2 %
```



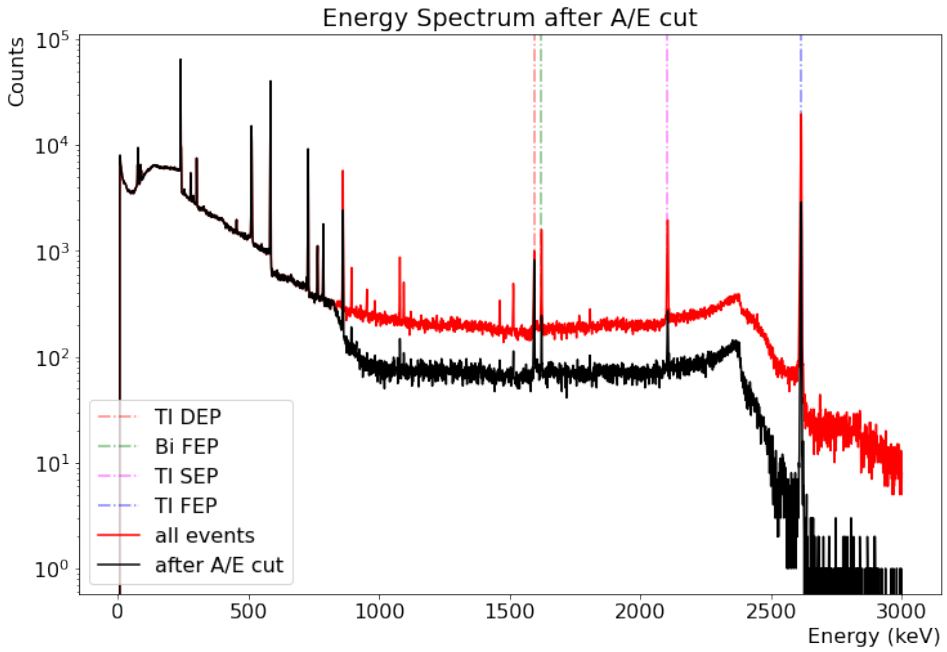
In [38]:

```
plt.figure()
plt.hist2d(ecal_pass[(classifier>=cut)], classifier[(classifier>=cut)], bi
cbar = plt.colorbar()
plt.xlim([1000,3000])
plt.xlabel("Energy (keV)", ha='right', x=1)
plt.ylabel("A/E (a.u.)", ha='right', y=1)
cbar.ax.set_ylabel('Counts')
plt.title("Classifier after cut")
plt.show()
```



In [39]:

```
peaks_id = ['Tl DEP', 'Bi FEP', 'Tl SEP', 'Tl FEP']
peaks = [1592.5,1620.5,2103.53,2614]
pm2 = 10
colours = ['red', 'green', 'magenta', 'blue']

# Set defaults for figures
plt.rcParams['figure.figsize'] = (12, 8)
plt.rcParams['font.size'] = 16

plt.figure()
cal_cut = ecal_pass[(classifier>=cut)]
hist, bins = np.histogram(ecal_pass, bins=3000, range=[0,3000])
hist1, bins1 = np.histogram(cal_cut, bins=3000, range=[0,3000])
plt.clf()
mins = 0.55
maxs= 1
[plt.axvline(peaks[i], ls='-.', color = colours[i], alpha = 0.5, label=peal
            ) for i in range(0,len(peaks))]
plt.plot(bins[1:], hist, color='red', linewidth=1.5, label='all events')
plt.plot(bins1[1:], hist1, '-r', color = 'black', linewidth=1.5, label='af
plt.ylabel('Counts', ha='right', y=1)
plt.xlabel('Energy (keV)', ha='right', x=1)
plt.yscale('log')
plt.legend(loc='lower left')
plt.title("Energy Spectrum after A/E cut")
plt.savefig(f'AoE figures/E_spectrum_cut_{det}', dpi = 400)
plt.show()
```


Energy Spectrum after A/E cut
Counts
10^5
10^4
10^3
10^2
10^1
10^0
TI DEP
Bi FEP
TI SEP
TI FEP
all events
after A/E cut
0
500
1000
1500
2000
2500
3000
Energy (keV)

In [ ]:

B2. DRIFT TIME ANALYSIS CODE

In [1]:
```
%matplotlib inline

import pygama.lh5 as lh5
import matplotlib.pyplot as plt
import numpy as np
import glob
from matplotlib.colors import LogNorm
import pathlib
from iminuit import cost,Minuit
from scipy.special import erf, erfc
from scipy.stats import norm, poisson
from scipy.integrate import simps
```

In [2]:
```
import os,json
import math
from scipy.optimize import curve_fit
import pygama.analysis.histograms as pgh
import pygama.analysis.calibration as cal
import pygama.analysis.peak_fitting as pgp
import pygama.analysis.peak_fitting as pgf
import scipy.optimize as opt
import matplotlib.colors as mcolors
import matplotlib.cm as cmx
from scipy.integrate import quad
from matplotlib.lines import Line2D
from matplotlib.backends.backend_pdf import PdfPages
import numba as nb
from math import erfc
from iminuit import Minuit, cost
from iminuit.util import propagate
import sys

%matplotlib inline
import pygama.io.lh5 as lh5
from pygama.dsp.WaveformBrowser import WaveformBrowser
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import os, json

# Set defaults for figures
plt.rcParams['figure.figsize'] = (12, 8)
plt.rcParams['font.size'] = 16
```

```
Warning: pygama.io.lh5 is deprecated and will be removed in a future releas
e. Instead import pygama.lh5.
```

In [3]:
```
kwd = {"parallel": False, "fastmath": True}
limit = np.log(sys.float_info.max)/10
```

A/E PDFs fit definition

In [4]:
```
def PDF_AoE(x, lambda_s, lambda_b, gaussian_mean, gaussian_sigma,
            htail,htau):

    pdf = (lambda_b * gauss_tail_norm(x,gaussian_mean, gaussian_sigma,
            htail,htau)+\lambda_s *
               norm_pdf(x, gaussian_mean, gaussian_sigma))
    #(1./(lambda_s+lambda_b))*
    return lambda_s+lambda_b, pdf
```

In [5]:
```
@nb.njit(**kwd)
def norm_pdf(x, mu, sigma):
    if sigma ==0: invs=np.nan
    else: invs = 1.0 / sigma
    z = (x - mu) * invs
    invnorm = 1 / np.sqrt(2 * np.pi) * invs
    return np.exp(-0.5 * z ** 2) * invnorm
```

In [6]:

```
@nb.njit(**kwd)
def nb_erf(x):
    y = np.empty_like(x)
    for i in nb.prange(len(x)):
        y[i] = erfc(x[i])
    return y

#@nb.njit(**kwd)
def gauss_tail(x, mu, sigma, tail, tau):
    """
    A gaussian tail function template
    Can be used as a component of other fit functions
    """
    x = np.asarray(x)
    tmp = ((x-mu)/tau) + ((sigma**2)/(2*tau**2))
    tail_f = np.where(tmp < limit,
                      gauss_tail_exact(x, mu, sigma, tail, tau),
                      gauss_tail_approx(x, mu, sigma, tail, tau))
    return tail_f

#@nb.njit(**kwd)
def gauss_tail_exact(x, mu, sigma, tail, tau):
    tmp = ((x-mu)/tau) + ((sigma**2)/(2*tau**2))
    abstau = np.absolute(tau)
    tmp = np.where(tmp < limit, tmp, limit)
    z = (x-mu)/sigma
    tail_f = (tail/(2*abstau)) * np.exp(tmp) *
            nb_erf( (tau*z + sigma)/(np.sqrt(2)*abstau))
    return tail_f

@nb.njit(**kwd)
def gauss_tail_approx(x, mu, sigma, tail, tau):
    den = 1/(sigma + tau*(x-mu)/sigma)
    tail_f = tail * sigma * norm_pdf(x, mu, sigma) *
            den * (1.-tau*tau*den*den)
    return tail_f

def gauss_tail_norm(x,mu, sigma, htail, tau):

    xs = np.linspace(np.nanmin(x), np.nanmax(x), 10001)#
    try:
        norm_pdf_values = gauss_tail(xs, mu, sigma, htail, tau)
    except ZeroDivisionError:
        return 0
    normalisation = simps(norm_pdf_values, xs, axis = 0)
    exp_part = gauss_tail(x, mu, sigma, htail, tau)
    return exp_part / normalisation
```

In [7]:
```
@nb.njit(**kwd)
def fexpo(x,p1,p2):
    return np.exp(p1+x*p2)

def norm_fexpo(x,z,p1,p2):
    xs = np.linspace(np.nanmin(x), np.nanmax(x), 10001)
    norm_pdf_values = fexpo(xs, p1,p2)
    normalisation = simps(norm_pdf_values, xs, axis = 0)

    return z*fexpo(x,p1,p2)/normalisation
```

Energy PDFs

In [8]:
```
@nb.njit(**kwd)
def step_pdf(x,  mu, sigma, hstep):
    invs = (np.sqrt(2)*sigma)
    z = (x-mu)/invs
    step_f = 1 + hstep * nb_erf(z)
    return step_f
```

In [9]:
```
def peak_pdf(x, mu, sigma, htail, tau):
    peak = norm_pdf(x,mu,sigma)
    tail = gauss_tail(x, mu, sigma, htail, tau)
    return peak+tail
```

In [10]:
```
def norm_energy_pdf(x, pdf_func, *params):
    xs = np.arange(np.nanmin(x), np.nanmax(x), 0.01)
    norm_pdf_values = pdf_func(xs, *params)
    normalisation = simps(norm_pdf_values, xs, axis = 0)

    return pdf_func(x, *params)/normalisation
```

In [11]:
```
def radford_pdf(data, lambda_s, lambda_b, mu, sigma, hstep, htail, tau):
    pdf = (lambda_b * norm_energy_pdf(data, step_pdf, mu, sigma, hstep)
               +\lambda_s *  norm_energy_pdf(data, peak_pdf, mu, sigma,
                    htail, tau))

    return lambda_s+lambda_b, pdf
```

A/E fit

In [12]:
```
def unbinned_aoe_fit(aoe, display=0, verbose=False):
    aoe_len = len(aoe)
    hist, bins,var = pgh.get_hist(aoe,bins=500)
    bin_centers = (bins[:-1]+bins[1:])/2
```

```
pars, cov = pgf.gauss_mode_max(hist, bins, var)
mu = bin_centers[np.argmax(hist)]
amp = np.amax(hist)

sigma=pgh.get_fwhm(hist, bins)[0]/2.355

ls_guess = 2*np.sum(hist[(bin_centers>mu)&
                         (bin_centers<(mu+2.5*sigma))])
def gauss(x,z,mu,sigma):
    return z * norm.pdf(x,mu,sigma)
c1_min= mu-2*sigma #0.495
c1_max= mu+5*sigma #0.52
c1 = cost.UnbinnedNLL(aoe[(aoe<c1_max)&(aoe>c1_min)], gauss)
m1 = Minuit(c1, ls_guess, mu,sigma)
m1.limits = [(0, len(aoe[(aoe<c1_max)&(aoe>c1_min)])),
             (mu*0.8, mu*1.2),(0.8*sigma,sigma*1.2)]
m1.migrad()
ls_guess =m1.values[0]
mu = m1.values[1]
sigma = m1.values[2]

fmin=  mu-15*sigma #0.45
fmax = mu+5*sigma #0.52
c2_max = mu-4*sigma

c2 = cost.UnbinnedNLL(aoe[(aoe<c2_max)&(aoe>fmin)], norm_fexpo)
m2 = Minuit(c2, len(aoe[(aoe<c2_max)&(aoe>fmin)]), -20,10)
m2.limits=[(0,len(aoe[(aoe<c2_max)&(aoe>fmin)])),(-100,0),(0,20)]
m2.migrad()
f = m2.values[2]

pars = [mu,sigma,amp,0,f,10]

bg_guess = len(aoe[(aoe<fmax)&(aoe>fmin)])-ls_guess
x0 = [ls_guess,bg_guess,pars[0],pars[1],pars[4],pars[5]]
if verbose:print(x0)

c = cost.ExtendedUnbinnedNLL(aoe[(aoe<fmax)&(aoe>fmin)], PDF_AoE)

m = Minuit(c, *x0)

m.migrad()
m.hesse()
if verbose:print(m.values)
if display>1:
    plt.figure()
    xs = np.linspace(fmin,fmax,1000)
    counts, bins, bars = plt.hist(aoe[(aoe<fmax)&(aoe>fmin)],
                                  bins=400, histtype='step')
    dx = np.diff(bins)
    plt.plot(xs, PDF_AoE(xs,*m.values)[1]* dx[0], label="Total PDF")
    #plt.yscale('log')
    plt.plot(xs , (m.values[1])*gauss_tail_norm(xs,*m.values[2:])
             * dx[0], label="Tail PDF")
    plt.plot(xs , (m.values[0])*norm_pdf(xs,
            *m.values[2:4])*dx[0], label="Gaussian PDF")
```

```
        plt.legend(loc='upper left')
        plt.title("Fit Components")
        plt.show()

        plt.figure()
        bin_centers= (bins[1:]+bins[:-1])/2
        plt.plot(bin_centers, (PDF_AoE(bin_centers,*m.values)[1]
                               * dx[0]) - counts)
        plt.title("Residuals")
        plt.show()
        return m.values, m.errors
    else: return m.values, m.errors
```

In [13]:

```
def AoEcorrection(e,aoe,eres, display=1, plot_all=False):



    comptBands_width = 20;
    comptBands = np.array([940,960,980,1000,1020,1040,1130,1150,
                           1170,1190,1210,1250,1270,1290,
                          1310,1330,1370,1390,1420,1540,1650,1700,
                           1780,1810,1850,1870,1890,1910,1930,1950,
                          1970,1990,2010,2030,2050,2150,2170,2190
                           ,2210,2230,2250,2270,2290])
    comptBands = comptBands[::-1]
    peaks = np.array([1080,1094,1459,1512, 1552, 1592,1620,
                      1650, 1670,1830,2105])
    compt_aoe = np.zeros(len(comptBands))
    aoe_sigmas = np.zeros(len(comptBands))
    compt_aoe_err = np.zeros(len(comptBands))
    aoe_sigmas_err = np.zeros(len(comptBands))
    ratio = np.zeros(len(comptBands))
    ratio_err = np.zeros(len(comptBands))

    copper = cm = plt.get_cmap('copper')
    cNorm  = mcolors.Normalize(vmin=0, vmax=len(comptBands))
    scalarMap = cmx.ScalarMappable(norm=cNorm, cmap=copper)

    plt.figure()
    for i, band in enumerate(comptBands):
        aoe_tmp = aoe[(e>band) & (e<band+comptBands_width)
                      & (aoe>0)][:10000]

        hist, bins,var = pgh.get_hist(aoe_tmp,bins=500)
        bin_center = (bins[:-1] + bins[1:]) / 2
        pars,errs = unbinned_aoe_fit(aoe_tmp, display=display)
        compt_aoe[i] = pars[2]
        aoe_sigmas[i] = pars[3]
        compt_aoe_err[i] = errs[2]
        aoe_sigmas_err[i] = errs[3]
        ratio[i] = pars[0]/pars[1]
        ratio_err[i] = ratio[i]*np.sqrt((errs[0]/pars[0])**2
                                        + (errs[1]/pars[1])**2)
        xs = np.arange(pars[2]-4*pars[3], pars[2]+3*pars[3],
                       pars[3]/10)
```

```
    if np.isnan(errs[2])|np.isnan(errs[3])|(errs[2]==0)
                                          |(errs[3]==0): pass
    else:
        colorVal = scalarMap.to_rgba(i)
        plt.plot(xs,PDF_AoE(xs, *pars)[1], color = colorVal)
def pol1(x,a,b):
    return a * x + b
plt.xlabel('A/E')
plt.ylabel("Expected Counts")
plt.title("Compton Band Fits")
cbar = plt.colorbar(cmx.ScalarMappable(norm=cNorm,
               cmap=plt.get_cmap('copper_r')),
                   orientation='horizontal',
                       label='Compton Band Energy',
                           ticks=[0,16,32,len(comptBands)])
cbar.ax.set_xticklabels([comptBands[::-1][0],
                         comptBands[::-1][16],comptBands[::-1][32],
                         comptBands[::-1][-1]])
plt.show()



ids = np.isnan(compt_aoe_err)|np.isnan(aoe_sigmas_err)
                             |(aoe_sigmas_err==0)|(compt_aoe_err==0)

plt.figure()
plt.errorbar(comptBands[~ids], ratio[~ids],
             xerr=10,yerr = ratio_err[~ids], linestyle=' ')
plt.xlabel("Energy (keV)")
plt.ylabel("N_sig/N_bkg")
plt.show()


pars, cov = opt.curve_fit(pol1,comptBands[~ids],
                          compt_aoe[~ids],sigma
                          = compt_aoe_err[~ids],
                          absolute_sigma=True)
errs = np.sqrt(np.diag(cov))

sig_pars, sig_cov = opt.curve_fit(pol1,comptBands[~ids],
                                  aoe_sigmas[~ids],
                                  sigma = aoe_sigmas_err[~ids],
                                  absolute_sigma=True)
sig_errs = np.sqrt(np.diag(sig_cov))

def sigma_fit(x, a,b,c):
    return np.sqrt(a+(b/x)**c)

p0 = [0.001,10, 3]
c = cost.LeastSquares(comptBands[~ids],aoe_sigmas[~ids],
                      aoe_sigmas_err[~ids], sigma_fit)

#print(p0)
c.loss = "soft_l1"
m = Minuit(c, *p0)
m.migrad()
m.hesse()
```

```
sig_pars2 = m.values
sig_errs2 = m.errors

model = pol1(comptBands,*pars)
sig_model = pol1(comptBands,*sig_pars)
sig_model2 = sigma_fit(comptBands,*sig_pars2)

sigma = np.sqrt(eres[0]+1592*eres[1])/2.355
n_sigma = 4
peak = 1592
emin           = peak - n_sigma*sigma
emax           = peak + n_sigma*sigma
dep_pars, dep_err = unbinned_aoe_fit(aoe[(e>emin)
                                         & (e<emax) & (aoe>0)])

fig, (ax1, ax2) = plt.subplots(2, 1,
                               constrained_layout=True, sharex=True)
ax1.errorbar(comptBands[~ids]+10,compt_aoe[~ids],
             yerr=compt_aoe_err[~ids],
              xerr=10,
              label='data', linestyle= ' ')
ax1.plot(comptBands[~ids]+10,model[~ids],label='linear model')
ax1.errorbar(1592, dep_pars[2], xerr = n_sigma*sigma,
                                yerr = dep_err[2], label='DEP',
                                 color='green', linestyle= ' ')

ax1.legend(title='A/E mu energy dependence', frameon=False)

ax1.set_ylabel("raw A/E (a.u.)", ha='right', y=1)
#ax1.ylim([np.amax(model), np.amin(model)])
ax2.scatter(comptBands[~ids]+10,100*(compt_aoe[~ids]-model[~ids])
            /model[~ids], lw=1, c='b')
ax2.scatter(1592,100*(dep_pars[2]-pol1(1592,*pars))/pol1(1592,*pars)
                    , lw=1, c='g')
ax2.set_ylabel("Residuals %", ha='right', y=1)
ax2.set_xlabel("Energy (keV)", ha='right', x=1)
plt.show()

fig, (ax1, ax2) = plt.subplots(2, 1, constrained_layout=True,
                               sharex=True)
ax1.errorbar(comptBands[~ids]+10, aoe_sigmas[~ids],
             yerr= aoe_sigmas_err[~ids],
            xerr = 10, label='data', linestyle= ' ')
#ax1.plot(comptBands[~ids],sig_model[~ids],label='linear model')
ax1.plot(comptBands[~ids],sig_model2[~ids],
         label=f'sqrt model: sqrt({sig_pars2[0]:1.4f}
         +({sig_pars2[1]:1.1f}/E)^{sig_pars2[2]:1.1f})')
ax1.errorbar(1592, dep_pars[3], xerr = n_sigma*sigma,
             yerr = dep_err[3], label='DEP', color='green')
ax1.set_ylabel("A/E stdev (a.u.)", ha='right', y=1)
ax1.legend(title='A/E stdev energy dependence', frameon=False)
ax2.scatter(comptBands[~ids]+10,100
            *(aoe_sigmas[~ids]-sig_model2[~ids])/sig_model2[~ids],
            lw=1, c='b')
ax2.scatter(1592,100*(dep_pars[3]-sigma_fit(1592,*sig_pars2))
            /sigma_fit(1592,*sig_pars2), lw=1, c='g')
```

```
    ax2.set_ylabel("Residuals %", ha='right', y=1)
    ax2.set_xlabel("Energy (keV)", ha='right', x=1)
    plt.show()


    return pars, sig_pars2, comptBands[~ids], aoe_sigmas[~ids],
            aoe_sigmas_err[~ids]
```

Energy fit

In [14]:

```
def unbinned_energy_fit(energy, peak, fwhm):
    energy_len = len(energy)
    hist, bins,var = pgh.get_hist(energy,dx=0.1,
                                  range= (np.amin(energy),
                                          np.amax(energy)))
    sigma = fwhm/2.355
    bg0 = np.sum(hist[-5:])/5
    step = np.sum(hist[:5])/5 - bg0
    htail = 1./5
    tau = 6.*sigma

    # now compute amp and return
    height = np.amax(hist)
    height -= (bg0 + step/2)
    amp = height / (htail*0.87/35 + (1-htail)/(sigma*np.sqrt(2*np.pi)))
    hstep = step/(2*amp)
    guess = [peak, sigma, hstep, htail, tau, bg0, amp]

    x0 = [guess[-1]*guess[1]*10,energy_len
          -(guess[-1]*guess[1]*10),*guess]
    x0= x0[:-2]
    bounds = [(0,energy_len),(0,energy_len), (x0[2]-2*x0[3],
                                x0[2]+2*x0[3]), (0, x0[3]*2), (0, 1),
                                  (0, 1), (x0[3], 100*x0[3])]
    c = cost.ExtendedUnbinnedNLL(energy, radford_pdf)
    m = Minuit(c, *x0)
    m.limits = bounds
    m.migrad()
    m.hesse()

    return m.values, m.errors
```

In [15]:
```
def overlap(data1,data2):
    '''
    Calculates overlap between two histogram datasets
    '''

    diff = abs(data1-data2)
    overlap = 0

    for i in range(0,len(diff)):
        if (diff[i]!= max(data1[i],data2[i])):
            overlap = overlap + min(data1[i],data2[i])

    return overlap

def histogram_bins(tp_max,tp_min,width):
    '''
    Calculates histogram bins
    '''

    a = 0

    for i in range(0,4):
        atest = max(tp_max[i]-tp_min[i])

        if atest>a+2500:
            pass

        elif atest>a:
            a=atest

    if a == 0:
        a=3000

    return np.arange(0,a+width,width)
```

Load in Data

In [16]:
```
energy_params = ['cuspEmax', 'cuspEmax_ctc', 'A_max','dt_eff']
```

In [17]:
```
det      = ['V05612A','V05612B','V05267A','V04199A']
D=2
det = det[D]
datatype = 'th_HS2_lat_psa'
run = '001'
energy_param = energy_params[1]
```

In [18]:
```
datapath   = f'/unix/legend/shared/test_data/{det}/tier2/{datatype}'
files      = os.listdir(datapath)
files.sort()
for i,file in enumerate(files):
    files[i] = os.path.join(datapath,file)
files = files[0:10]

datapath1   = f'/unix/legend/shared/test_data/{det}/tier1/{datatype}'
raw_files      = os.listdir(datapath1)
raw_files.sort()
for i,raw_file in enumerate(raw_files):
    raw_files[i] = os.path.join(datapath1,raw_file)
```

In [19]:
```
calpath = f'/unix/legend/shared/aoe/{det}.json'

with open(calpath, 'r') as o:
    cal_dict = json.load(o)
cal_pars = cal_dict[energy_param]['Calibration_pars']
eres_pars = [cal_dict[energy_param]['m0'], cal_dict[energy_param]['m1']]
adc = cal_pars[0]
```

In [20]:
```
print(len(files),'files found in',datapath)
uncal_pass = lh5.load_nda(files,energy_params,'raw',verbose=False)
print("done")
```

```
10 files found in /unix/legend/shared/test_data/V05267A/tier2/th_HS2_lat_ps
a
done
```

In [21]:
```
ecal_pass = pgp.poly(uncal_pass[energy_param], cal_pars)
```

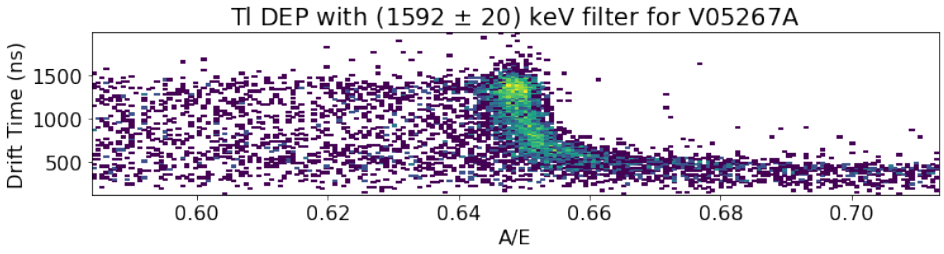
In [22]:
```
curr = uncal_pass['A_max']
aoe = np.divide(curr,pgp.poly(uncal_pass['cuspEmax'], cal_pars))
```

Drift time plots

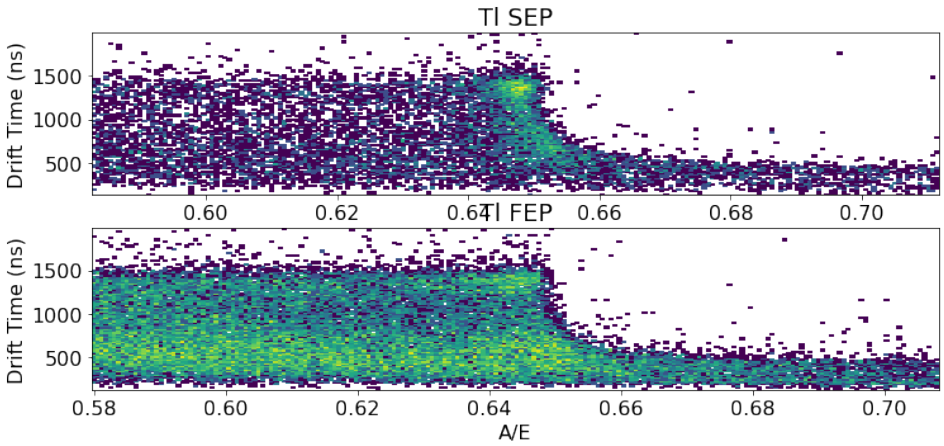
Have a play with these in different energy regions, also have a look at the a/e spectrum in different regions

In [23]:
```
def plot_dt_dep(aoe, energy,dt, erange,title):

    hist, bins,var = pgh.get_hist(aoe[(energy>erange[0]) &
                                      (energy<erange[1])],bins=500)
    bin_cs = (bins[1:]+bins[:-1])/2
    mu = bin_cs[np.argmax(hist)]
    aoe_range = [mu*0.9, mu*1.1]

    idxs = (energy>erange[0]) &(energy<erange[1])&(aoe>aoe_range[0])
                              &(aoe<aoe_range[1])&(dt<2000)
    plt.hist2d(aoe[idxs], dt[idxs], bins=[200,100], norm=LogNorm())
    plt.ylabel('Drift Time (ns)')
    plt.xlabel('A/E')
    plt.title(title)

def plot_dt_dep_aoe(aoe, energy,dt, erange,title):

    hist, bins,var = pgh.get_hist(aoe[(energy>erange[0])
                                      &(energy<erange[1])],bins=500)
    bin_cs = (bins[1:]+bins[:-1])/2
    mu = bin_cs[np.argmax(hist)]
    aoe_range = [mu*0.9, mu*1.1]

    idxs = (energy>erange[0]) &(energy<erange[1])&(aoe>aoe_range[0])
                              &(aoe<aoe_range[1])&(dt<2000)
    plt.hist2d(aoe[idxs], dt[idxs], bins=[200,100], norm=LogNorm())
    plt.ylabel('Drift Time (ns)')
    plt.xlabel('A/E')
    plt.title(title)
    return aoe_range
```

In [24]:
```
dt = uncal_pass["dt_eff"]
plt.figure()
plt.subplot(3,1,1)
plot_dt_dep(aoe, ecal_pass, dt, [1572,1612],
            f'Tl DEP with (1592 $\pm$ 20) keV filter for {det}')
```



In [25]:
```
plt.figure
plt.subplot(3,1,2)
plot_dt_dep(aoe, ecal_pass, dt, [2070,2130], f'Tl SEP')
plt.subplot(3,1,3)
plot_dt_dep(aoe, ecal_pass, dt, [2550,2670], f'Tl FEP')
plt.show()
```



Waveform Selector:

In [26]:
```
# First, load a dataframe from a DSP file that we can use to make our sele
df = lh5.load_dfs(files, ['cuspEmax_ctc' , 'A_max', 'dt_eff',
                          'tp_0_est', 'tp_01', 'tp_10', 'tp_20',
                          'tp_50', 'tp_80','tp_90', 'tp_95', 'tp_99',
                          'tp_max'], '/raw')
trapE = df['cuspEmax_ctc']
tp_0 = df['tp_0_est']
tp_01 = df['tp_01']
tp_10 = df['tp_10']
tp_20 = df['tp_20']
tp_50 = df['tp_50']
tp_80 = df['tp_80']
tp_90 = df['tp_90']
tp_95 = df['tp_95']
tp_99 = df['tp_99']
tp_max = df['tp_max']
dt_eff = df['dt_eff']


tps_ = [tp_0,tp_01,tp_10,tp_20,tp_50,tp_80,tp_90,tp_95,tp_99,tp_max]

cal_E = trapE*adc
```

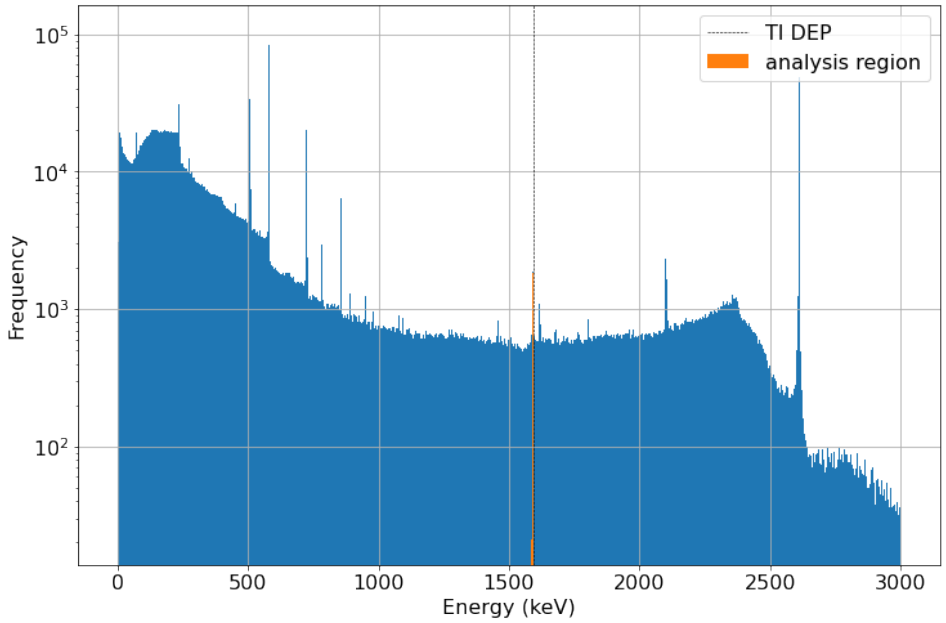
```
loading data for /unix/legend/shared/test_data/V05267A/tier2/th_HS2_lat_psa
/char_data-V05267A-th_HS2_lat_psa-run0001-201006T161531_tier2.lh5 /unix/leg
end/shared/test_data/V05267A/tier2/th_HS2_lat_psa/char_data-V05267A-th_HS2_
lat_psa-run0001-201006T162532_tier2.lh5 /unix/legend/shared/test_data/V0526
7A/tier2/th_HS2_lat_psa/char_data-V05267A-th_HS2_lat_psa-run0001-201006T163
534_tier2.lh5 /unix/legend/shared/test_data/V05267A/tier2/th_HS2_lat_psa/ch
ar_data-V05267A-th_HS2_lat_psa-run0001-201006T164535_tier2.lh5 /unix/legend
/shared/test_data/V05267A/tier2/th_HS2_lat_psa/char_data-V05267A-th_HS2_lat
_psa-run0001-201006T165539_tier2.lh5 /unix/legend/shared/test_data/V05267A/
tier2/th_HS2_lat_psa/char_data-V05267A-th_HS2_lat_psa-run0001-201006T170542
_tier2.lh5 /unix/legend/shared/test_data/V05267A/tier2/th_HS2_lat_psa/char_
data-V05267A-th_HS2_lat_psa-run0001-201006T171544_tier2.lh5 /unix/legend/sh
ared/test_data/V05267A/tier2/th_HS2_lat_psa/char_data-V05267A-th_HS2_lat_ps
a-run0001-201006T172547_tier2.lh5 /unix/legend/shared/test_data/V05267A/tie
r2/th_HS2_lat_psa/char_data-V05267A-th_HS2_lat_psa-run0001-201006T173551_ti
er2.lh5 /unix/legend/shared/test_data/V05267A/tier2/th_HS2_lat_psa/char_dat
a-V05267A-th_HS2_lat_psa-run0001-201006T174553_tier2.lh5
```

In [27]:

```
# DEP range to be analysed
E_DEP = 1592
pm = 3
DEP_range = [(E_DEP-pm)/adc, (E_DEP+pm)/adc]
DEP_range_cal = [(E_DEP-pm), (E_DEP+pm)]
```

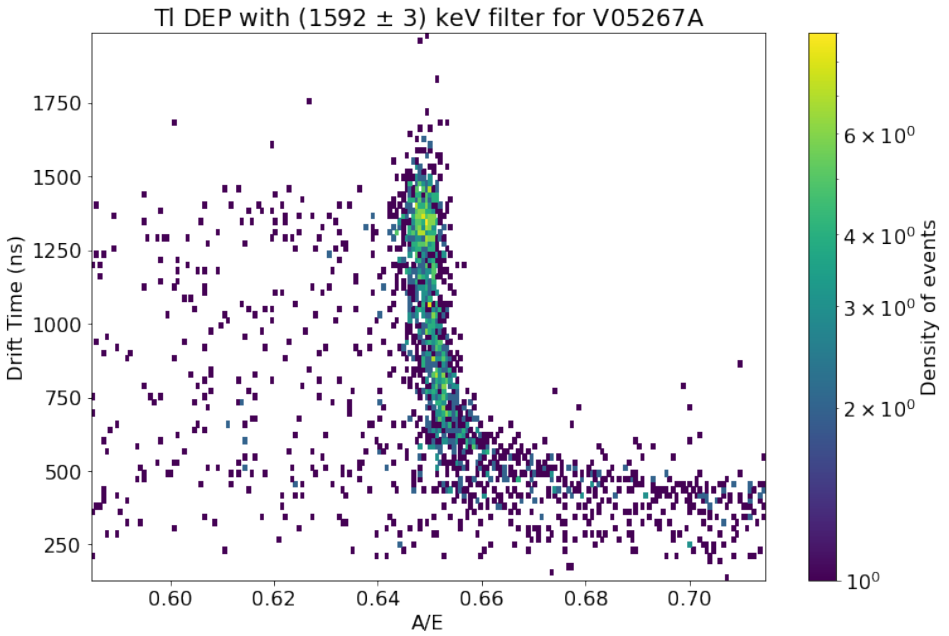
In [28]:

```
# Plots energy spectrum of detector with region of analysis about DEP
energy_selection_cal = (ecal_pass>DEP_range_cal[0])
            & (ecal_pass<DEP_range_cal[1])

cal_E.hist(bins=1000, range=(0, 3000))
cal_E[energy_selection_cal].hist(bins=1000, range=(0, 3000),
                                 label='analysis region')
plt.axvline(x=E_DEP, ls = '--', color = 'k', lw = '0.6',
                                     label = 'Tl DEP')
plt.yscale('log')
plt.legend(loc='best')
plt.xlabel('Energy (keV)')
plt.ylabel('Frequency')
#plt.title(f"Energy spectrum and analysis region for {det}")

plt.savefig(f'figures/E_Spectrum_{det}.jpg', dpi = 400)
```



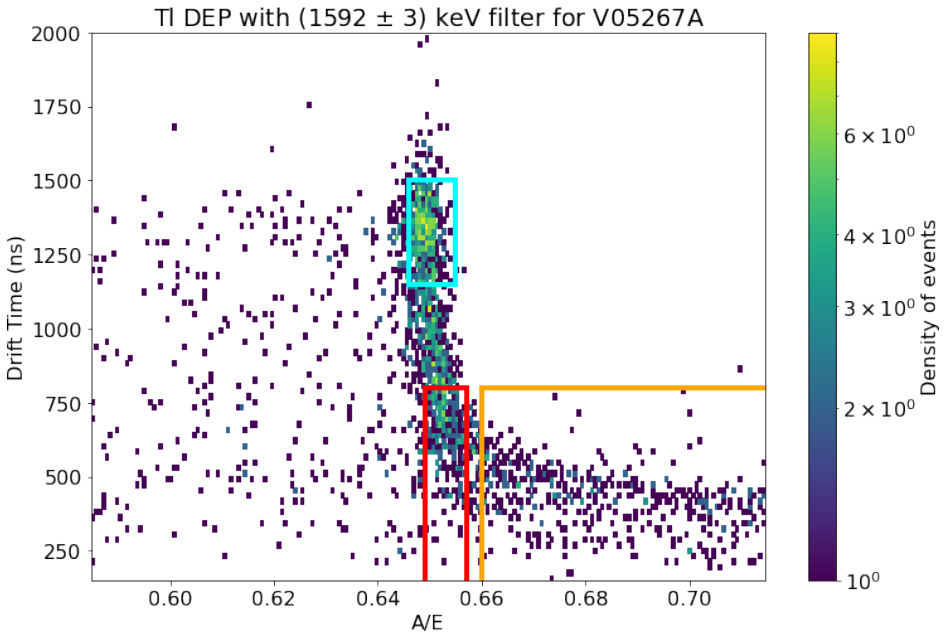
In [29]:

```
# Raw drift time plot for analysed detector
plt.figure()
aoe_range = plot_dt_dep_aoe(aoe, ecal_pass, dt, DEP_range_cal,
                            f'Tl DEP with ({E_DEP} $\pm$ {pm}) keV
                            'filter for {det}')
plt.xlim(min(aoe_range), max(aoe_range))
plt.colorbar(label = 'Density of events')
plt.savefig(f'dt_plot_{det}', dpi = 400);
```


Tl DEP with (1592 ± 3) keV filter for V05267A
Drift Time (ns)
1750
1500
1250
1000
750
500
250
0.60
0.62
0.64
0.66
0.68
0.70
A/E
6×10^0
4×10^0
3×10^0
2×10^0
10^0
Density of events

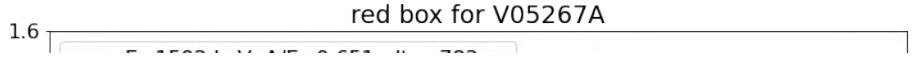
In [30]:

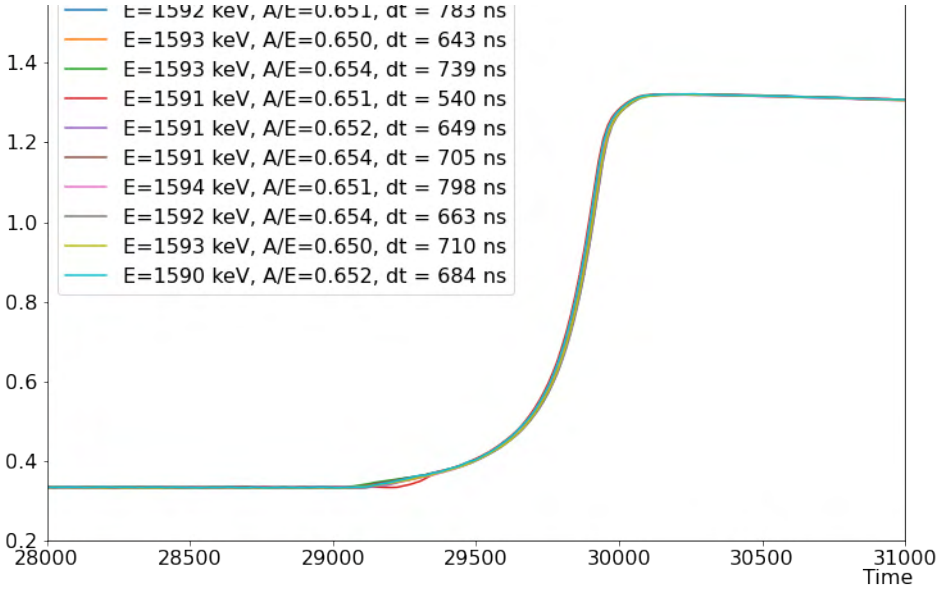
```
# Drift time plot with region of analyses
from matplotlib.ticker import ScalarFormatter, FormatStrFormatter
# [red,magenta,orange,cyan]
aoe_low = [[0.659,0.656,0.67,0.656],[0.883,0.88,0.9,0.9],
           [0.649,0.646,0.66,0.646]]
aoe_high = [[0.667,0.667,0.8,0.667],[0.893,0.893,0.96,0.96],
            [0.657,0.655,0.8,0.655]]
aoe_low = aoe_low[D]
aoe_high = aoe_high[D]

dtime_low = [[100,1150,100,1150],[300,1000,300,1000],
             [100,1150,100,1150]]
dtime_high = [[800,1400,800,1400],[500,1300,500,1300],
              [800,1500,800,1500]]
dtime_low = dtime_low[D]
dtime_high= dtime_high[D]

aoe_selection = []
dtime_selection = []
rect = []
colour = ['red','magenta','orange','cyan']

plt.figure()
aoe_range = plot_dt_dep_aoe(aoe, ecal_pass, dt, DEP_range_cal,
                            f'Tl DEP with ({E_DEP} $\pm$ {pm}) keV
                            'filter for {det}')
plt.xlim(min(aoe_range), max(aoe_range))
plt.colorbar(label = 'Density of events')
plt.ylim(150,2000)

for i in range(0,4):
    aoe_selection.append((aoe>aoe_low[i]) & (aoe<aoe_high[i]))
    dtime_selection.append((dt>dtime_low[i]) & (dt<dtime_high[i]))
    plt.gca().add_patch(plt.Rectangle((aoe_low[i],dtime_low[i]),
                            aoe_high[i]-aoe_low[i], dtime_high[i]
                                      -dtime_low[i], ec=colour[i],
                                          fc='none', lw=4))

plt.savefig(f'figures/dt_aoe_{det}.jpg', dpi = 400)
```

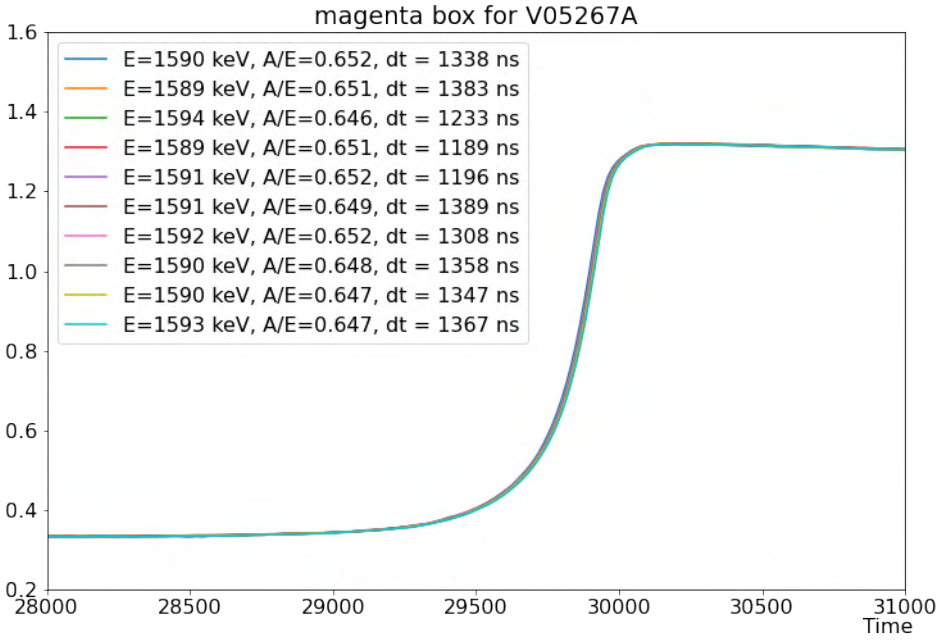


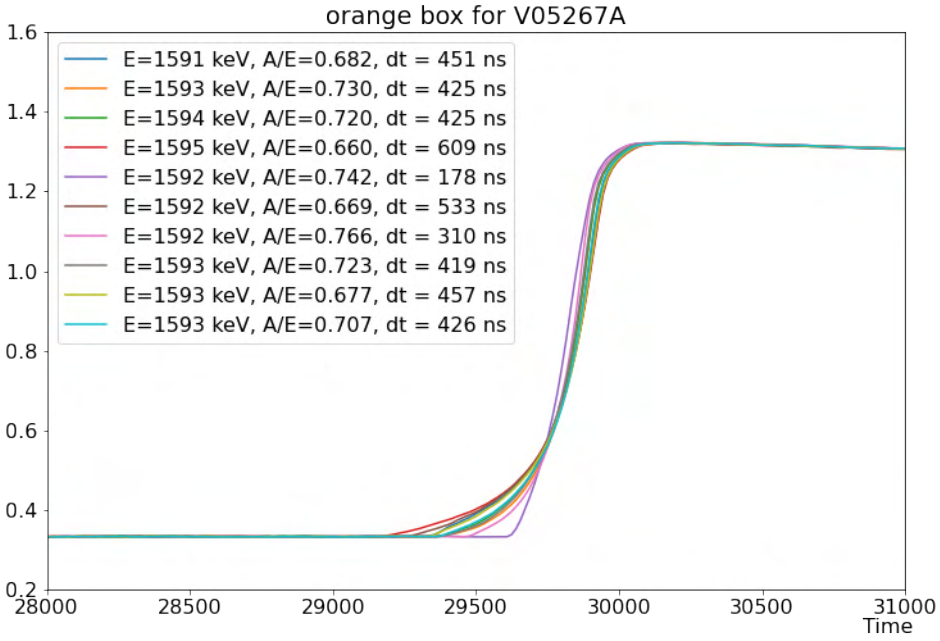
In [31]:

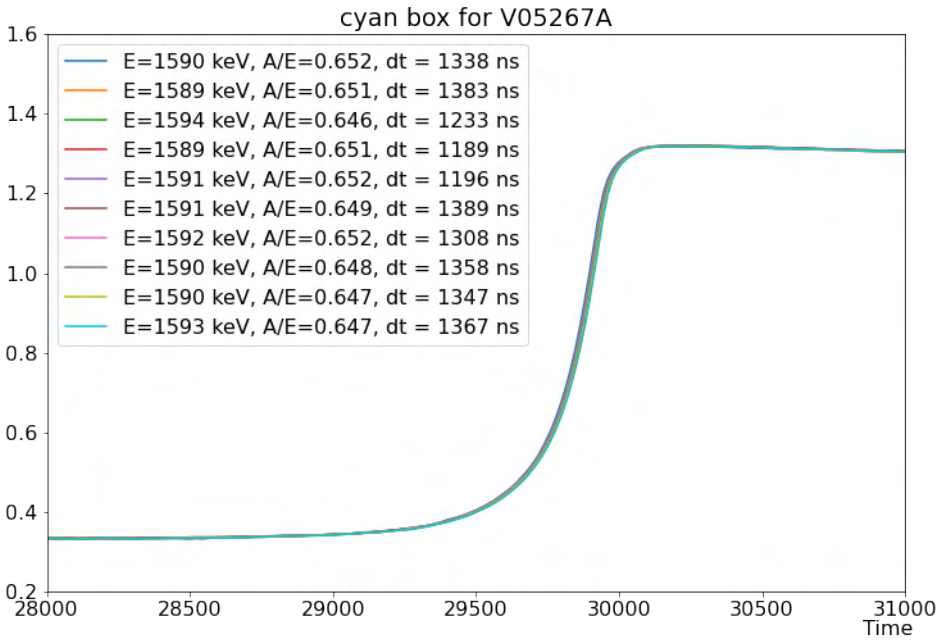
```
# Extracts waveforms from analysis regions using the LEGEND waveform brows
browser = []
axis = [[28500,31000,10000,34000],[28000,30500,14000,38000],
        [28000,31000,5000,30000]]
axis_norm = [[28500,31000,0.6,1.8],[28000,30500,0.6,1.8],
             [28000,31000,0.2,1.6]]

for i in range(0,4):
    browser.append(WaveformBrowser(raw_files, '/raw',
                                   verbosity   = 0,
                                   legend      = ("E={:.0f} keV,
                                                  'A/E={:.3f}, dt = {:.0f} ns'
                                                  , ecal_pass, aoe, dt),

                                   selection   = energy_selection_cal
                                           & aoe_selection[i]
                                            & dtime_selection[i],
                                   norm = trapE,
                                   n_drawn     = 10
                                  ))

    browser[i].draw_next()
    plt.title(f'{colour[i]} box for {det}')
    plt.xlabel('Time')
    plt.ylabel('')
    plt.axis(axis_norm[D])
    plt.savefig(f'figures/{colour[i]}_box_{det}.jpg', dpi = 400)
```




E=1592 keV, A/E=0.651, dt = 783 ns
E=1593 keV, A/E=0.650, dt = 643 ns
E=1593 keV, A/E=0.654, dt = 739 ns
E=1591 keV, A/E=0.651, dt = 540 ns
E=1591 keV, A/E=0.652, dt = 649 ns
E=1591 keV, A/E=0.654, dt = 705 ns
E=1594 keV, A/E=0.651, dt = 798 ns
E=1592 keV, A/E=0.654, dt = 663 ns
E=1593 keV, A/E=0.650, dt = 710 ns
E=1590 keV, A/E=0.652, dt = 684 ns
1.4
1.2
1.0
0.8
0.6
0.4
0.2
28000
28500
29000
29500
30000
30500
31000
Time

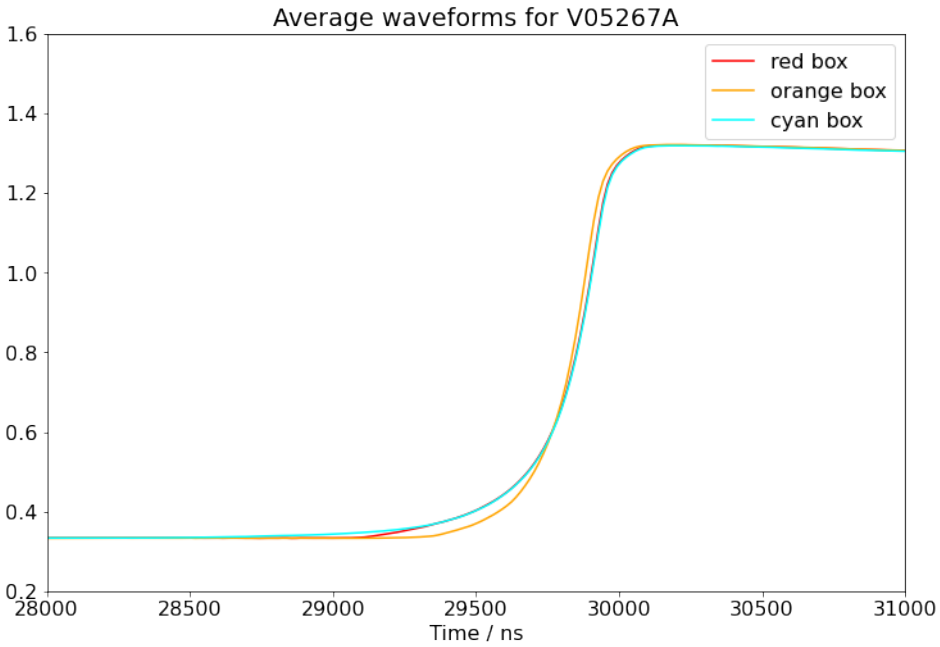

magenta box for V05267A
E=1590 keV, A/E=0.652, dt = 1338 ns
E=1589 keV, A/E=0.651, dt = 1383 ns
E=1594 keV, A/E=0.646, dt = 1233 ns
E=1589 keV, A/E=0.651, dt = 1189 ns
E=1591 keV, A/E=0.652, dt = 1196 ns
E=1591 keV, A/E=0.649, dt = 1389 ns
E=1592 keV, A/E=0.652, dt = 1308 ns
E=1590 keV, A/E=0.648, dt = 1358 ns
E=1590 keV, A/E=0.647, dt = 1347 ns
E=1593 keV, A/E=0.647, dt = 1367 ns
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
28000
28500
29000
29500
30000
30500
31000
Time


orange box for V05267A
E=1591 keV, A/E=0.682, dt = 451 ns
E=1593 keV, A/E=0.730, dt = 425 ns
E=1594 keV, A/E=0.720, dt = 425 ns
E=1595 keV, A/E=0.660, dt = 609 ns
E=1592 keV, A/E=0.742, dt = 178 ns
E=1592 keV, A/E=0.669, dt = 533 ns
E=1592 keV, A/E=0.766, dt = 310 ns
E=1593 keV, A/E=0.723, dt = 419 ns
E=1593 keV, A/E=0.677, dt = 457 ns
E=1593 keV, A/E=0.707, dt = 426 ns
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
28000
28500
29000
29500
30000
30500
31000
Time


cyan box for V05267A
E=1590 keV, A/E=0.652, dt = 1338 ns
E=1589 keV, A/E=0.651, dt = 1383 ns
E=1594 keV, A/E=0.646, dt = 1233 ns
E=1589 keV, A/E=0.651, dt = 1189 ns
E=1591 keV, A/E=0.652, dt = 1196 ns
E=1591 keV, A/E=0.649, dt = 1389 ns
E=1592 keV, A/E=0.652, dt = 1308 ns
E=1590 keV, A/E=0.648, dt = 1358 ns
E=1590 keV, A/E=0.647, dt = 1347 ns
E=1593 keV, A/E=0.647, dt = 1367 ns
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
28000
28500
29000
29500
30000
30500
31000
Time

In [32]:

```
# Plots average waveforms from each analysis region into one plot

if D == 1: # If detector = V05612B
    for j in range(0,4):
        y = []

        for i in range(0,10):
            y.append(browser[j].wf_data[0][i][1])

        y_avg = sum(y)/len(y)
        x = browser[j].wf_data[0][0][0]

        plt.plot(x,y_avg,color = colour[j], label = f'{colour[j]} box')
        plt.xlabel('Time / ns')
        plt.title(f'Average waveforms for {det}')
        plt.legend(loc='best')
        plt.axis(axis_norm[D])
        plt.savefig(f'figures/Avg_waveform_{det}.jpg', dpi = 400)

else:
    for j in [0,2,3]:
        y = []

        for i in range(0,10):
            y.append(browser[j].wf_data[0][i][1])

        y_avg = sum(y)/len(y)
        x = browser[j].wf_data[0][0][0]

        plt.plot(x,y_avg,color = colour[j], label = f'{colour[j]} box')
        plt.xlabel('Time / ns')
        plt.title(f'Average waveforms for {det}')
        plt.legend(loc='best')
        plt.axis(axis_norm[D])
        plt.savefig(f'figures/Avg_waveform_{det}.jpg', dpi = 400)
```



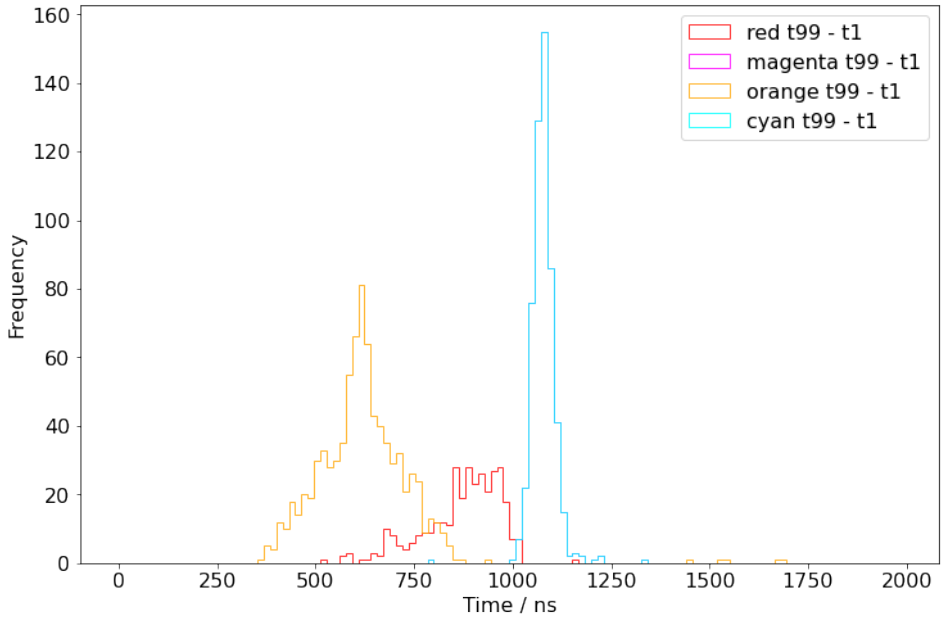
In [33]:

```
# Extracts timepoint data for analysis regions
t0 = []
t1 = []
t10 = []
t20 = []
t50 = []
t80 = []
t90 = []
t95 = []
t99 = []
tmax = []

for i in range(0,4):
    selection   = energy_selection_cal & aoe_selection[i]
                                       & dtime_selection[i]
    t0.append(tp_0[selection])
    t1.append(tp_01[selection])
    t10.append(tp_10[selection])
    t20.append(tp_20[selection])
    t50.append(tp_50[selection])
    t80.append(tp_80[selection])
    t90.append(tp_90[selection])
    t95.append(tp_95[selection])
    t99.append(tp_99[selection])
    tmax.append(tp_max[selection])
```

Overlap Method:

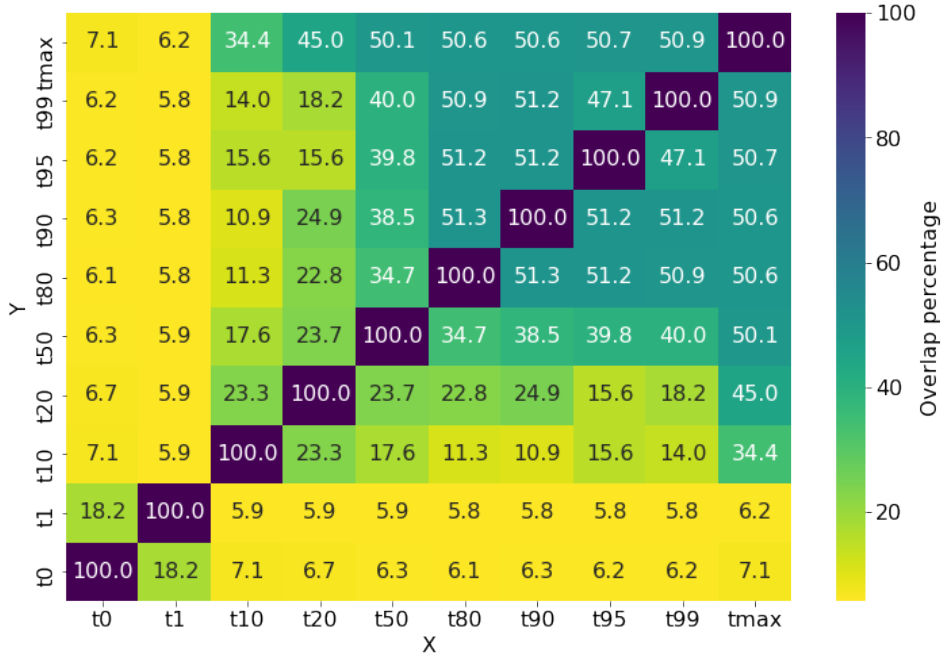
In [34]:
```
# Calculates and plots timepoint distributions with minimal overlap

tps = [t0,t1,t10,t20,t50,t80,t90,t95,t99,tmax]
tps_id = ['t0','t1','t10','t20','t50','t80','t90','t95','t99','tmax']

tps_reverse = reversed(tps)
overlap_value = 1000
width = 16
overlap_matrix = np.empty((len(tps),len(tps)))
overlap_matrixp = np.empty((len(tps),len(tps)))

for i in range(0,int(len(tps))):
    tps_reversed = next(tps_reverse)
    for j in range(0,len(tps) - (i+1)): # Iterates through each possible
                                        # timepoint difference of |Y-X|

        Itest = len(tps) - (i+1)
        Jtest = j

        p_test = []
        difference_test = []
        n_test = []
        bin0 = histogram_bins(tps_reversed,tps[j],width)

        for k in range(0,4):
            difference_test.append(abs(tps_reversed[k] - tps[j][k]))
            n_t, binstest = np.histogram(difference_test[k], bins = bin0)
            n_test.append(n_t)

        # Calculates overlap, as in Equation (9)
        overlap_test = overlap(n_test[0],n_test[2])
                +overlap(n_test[1],n_test[2])

        red = difference_test[0]
        cyan = difference_test[1]
        orange = difference_test[2]
        overlap_matrix[Itest][Jtest] = overlap_matrix[Jtest][Itest]
                            = overlap_test
        overlap_matrixp[Itest][Jtest] = overlap_matrixp[Jtest][Itest]
                            = 100*(overlap_test
                                   /(len(red)+len(cyan)+len(orange)))

        if (overlap_test < overlap_value): # If overlap decreases,
                                           # updates index I and J,
                                               # and overlap values
            overlap_value = overlap_test
            difference = difference_test
            I = len(tps) - (i+1)
            J = j

for i in range(0,len(tps)):
    overlap_matrix[i][i] = overlap_matrix.max()
    overlap_matrixp[i][i] = 100

bin0 = histogram_bins(tps[I],tps[J],width)

for X in np.arange(0,1000,5):
```

```
    test = difference[0]
    if (100*len(test[test>X])/len(test) < 90):
        break
    time_cut = X

n = []
cut_values = []
tphist_values = []

plt.figure()
for i in range(0,4):
    diff = difference[i]
    cut_values.append(100*len(diff[diff>time_cut])/len(diff))
    bin0 = np.arange(0,2000,16)
    ntest, binstest, patchestest = plt.hist(diff, bins = bin0,
                                            histtype='step', alpha=0.8,
                                            edgecolor = colour[i],
                                            label = f'{colour[i]}
                                            {tps_id[I]} - {tps_id[J]}',
                                            color = colour[i])
    tphist_values.append(ntest)
    plt.xlabel('Time / ns')
    plt.ylabel('Frequency')
    n.append(ntest)

plt.legend(loc='best');
plt.savefig(f'figures/tp_hist_{det}.jpg', dpi = 400)
```



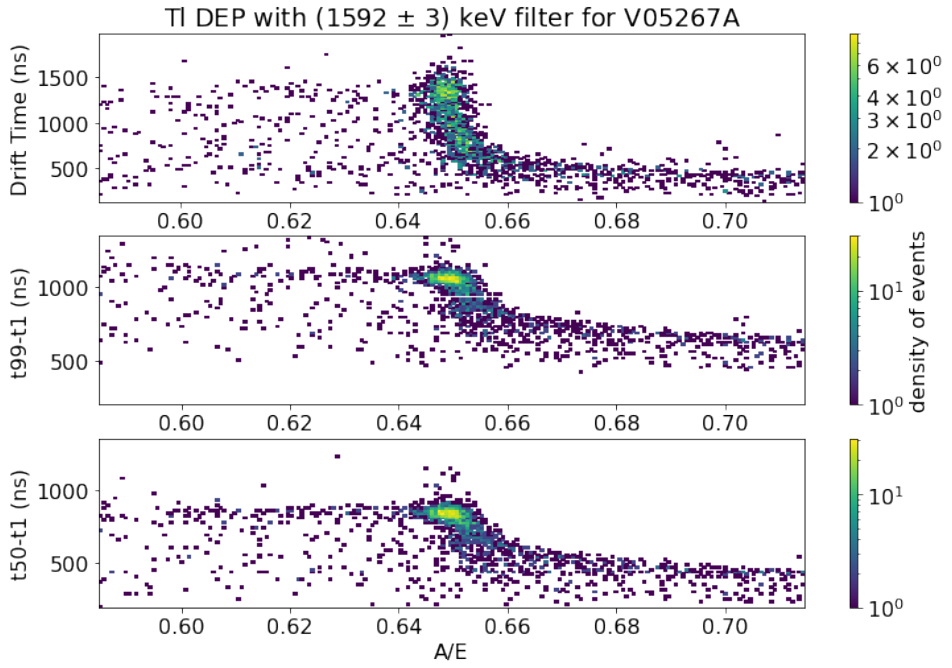
In [35]:

```
# Plots parameter space of timepoint distributions |Y-X|
# with an 'overlap percentage' figure of merit

import seaborn as sns
axesticks = [tps_id[i] for i in range(0,10)]
plt.figure()
sns.heatmap(overlap_matrixp[:len(tps),:len(tps)], annot = True,
            fmt = '.1f', cmap = 'viridis_r', cbar = True,
            xticklabels=axesticks, yticklabels=axesticks,
            cbar_kws={'label': 'Overlap percentage'})
plt.gca().invert_yaxis()
plt.xlabel('X')
plt.ylabel('Y')
#plt.title(f'percentage of overlap |Y - X| events for {det}');
plt.savefig(f'figures/Overlap_matrix_per{det}.jpg', dpi = 400)
```



In [36]:

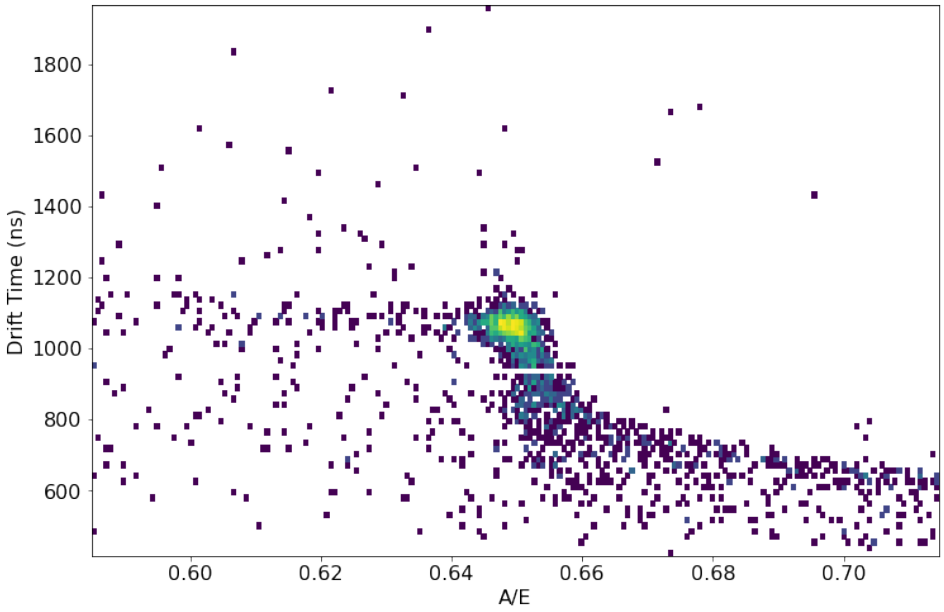
```
# Drift time plot, with optimal timepoint difference and t50-t1 plots

plt.figure()
plt.subplot(3,1,1)
aoe_range = plot_dt_dep_aoe(aoe, ecal_pass, dt_eff, DEP_range_cal,
                            f'Tl DEP with ({E_DEP} $\pm$ {pm}) keV
                            'filter for {det}')
plt.colorbar()

plt.subplot(3,1,2)
plot_dt_dep(aoe, ecal_pass, (tps_[I]-tps_[J]), DEP_range_cal, f'')
plt.ylabel(f'{tps_id[I]}-{tps_id[J]} (ns)')
plt.xlim(min(aoe_range),max(aoe_range))
plt.ylim(200,1350)
plt.colorbar(label = 'density of events');

plt.subplot(3,1,3)
delta = 4
plot_dt_dep(aoe, ecal_pass, (tps_[I-delta]-tps_[J]), DEP_range_cal, f'')
plt.ylabel(f'{tps_id[I-delta]}-{tps_id[J]} (ns)')
plt.xlim(min(aoe_range),max(aoe_range))
plt.ylim(200,1350)
plt.colorbar();

plt.savefig(f'figures/dt_aoe_cut_{det}.jpg', dpi = 400)
```

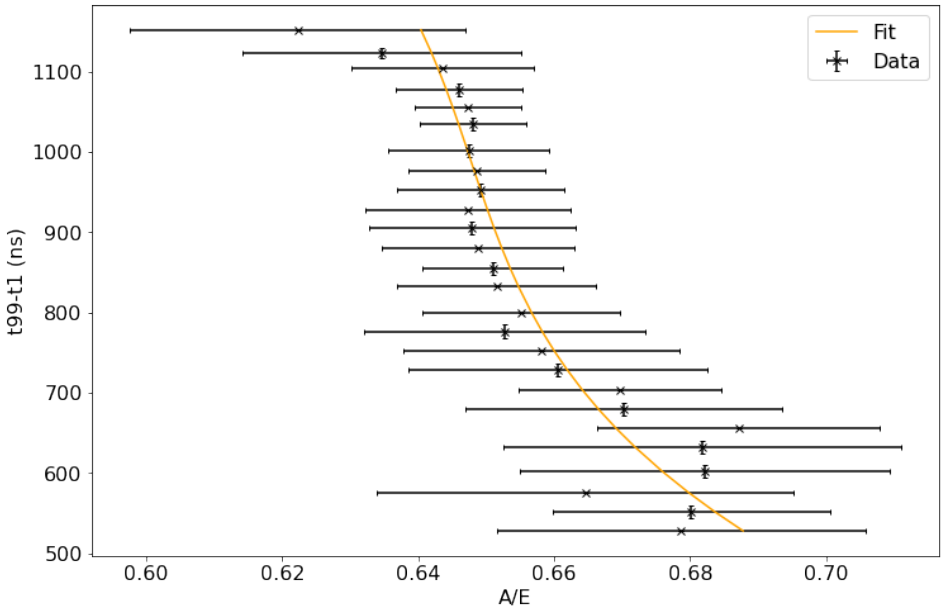


In [37]:

```
# Plots median A/E values for selected 25 ns drift time bands

aoe_range = plot_dt_dep_aoe(aoe, ecal_pass, (tps_[I]-tps_[J]),
                            DEP_range_cal, f'')
dts_range = (tps_[I]-tps_[J])[energy_selection_cal&(aoe>aoe_range[0])
                              &(aoe<aoe_range[1])&((tps_[I]-tps_[J])<2000)]
dts = (tps_[I]-tps_[J])[energy_selection_cal&(aoe>aoe_range[0])
                            &(aoe<aoe_range[1])&((tps_[I]-tps_[J])<1200)]
aoe1 = aoe[energy_selection_cal&(aoe>aoe_range[0])&(aoe<aoe_range[1])
                           &((tps_[I]-tps_[J])<1200)]
bands = np.arange(min(dts),max(dts),25)

aoe_means = []
dt_means = []
aoe_stds = []
dt_stds = []

for i in range(0,len(bands)-1):
    aoe_band = []
    aoe_band = aoe1[(dts>bands[i])&(dts<bands[i+1])]
    dt_band = dts[(dts>bands[i])&(dts<bands[i+1])]
    aoe_means.append(np.mean(aoe_band))
    dt_means.append(np.mean(dt_band))
    aoe_stds.append(np.std(aoe_band))
    dt_stds.append(np.std(dt_band))

plt.figure()
a = 4
plt.errorbar(aoe_means[a:], dt_means[a:], fmt = 'x', color = 'k',
             yerr = dt_stds[a:], xerr = aoe_stds[a:],
             label = 'Data', capsize = 2)

w = []
for i in range(a,len(aoe_stds)):
    w.append(1/aoe_stds[i])

b = np.polyfit(dt_means[a:], aoe_means[a:], w = w, deg = 3)
y = []
for dtm in dt_means[a:]:
    y.append(b[3] + b[2]*dtm + b[1]*dtm**2 + b[0]*dtm**3)

plt.plot(y,dt_means[a:], color = 'orange', label = 'Fit')
plt.xlabel('A/E')
plt.ylabel(f'{tps_id[I]}-{tps_id[J]} (ns)')
plt.legend(loc='best')
#plt.savefig(f'dt aoe fit {tps_id[I]}-{tps_id[J]} {det}.jpg', dpi = 400);
```

Out[37]: <matplotlib.legend.Legend at 0x7f5c343c7bb0>


Drift Time (ns)
1800
1600
1400
1200
1000
800
600
0.60
0.62
0.64
0.66
0.68
0.70
A/E


t99-t1 (ns)
1100
1000
900
800
700
600
500
0.60
0.62
0.64
0.66
0.68
0.70
A/E
Fit
Data

In [38]:

```
# Plots t50 - t1 for analysed detector

plt.figure()
for i in [0,2,3]:
    diff = t50[i]-t1[i]
    bin0 = np.arange(0,2000,16)
    ntest, binstest, patchestest = plt.hist(diff, bins = bin0,
                                            histtype='step', alpha=0.8,
                                            edgecolor = colour[i],
                                            label = f'{colour[i]}
                                            {tps_id[len(tps)-6]} -
                                            {tps_id[1]}', color =
                                            colour[i])
    tphist_values.append(ntest)
    plt.xlabel('Time / ns')
    plt.ylabel('Frequency')
    n.append(ntest)
plt.legend(loc='upper left')
plt.savefig(f'tp_hist_50-1_{det}.jpg', dpi = 400)
```