Applicability of pulse shape analysis methods in measurements of low energy gamma rays from human lungs

Pertti Hallikainen
University of Jyväskylä

25.6.2015

Abstract

Pulse shape analysis methods were applied to gamma spectroscopy measurements in an attempt to reduce the contribution from low-energy Compton background events, with focus on measuring inhaled plutonium from human lungs. No link between interaction depth and pulse rise time was found with the BEGe-detector used. The human torso phantom with plutonium-239 incorporated lungs used in the measurements had an undesired americium-241 contamination. Detector properties that would better enable pulse shape analysis to enhance detector performance are discussed.

Contents

1 Introduction

The health hazards of working with plutonium were understood fairly quickly after the element was first confirmed to exist in 1941. From past experiences with radium, it was recognized that the alpha active plutonium could be dangerous in the body even in small microgram amounts. In 1944, a tolerance level of five micrograms of plutonium in the body was established. In 1945 this limit was reduced to one microgram [1].

Because the alpha particles emitted by plutonium are absorbed in the body, plutonium workers' plutonium intake could only be measured from nose swipes and urine samples. The accompanying low energy gamma radiation is absorbed effectively in tissue and obscured by Compton scattered gamma rays in measured spectra, so that direct *in-vivo* measurements of plutonium were not possible until the mid-1960s [2].

The need to protect radiation workers and the general public from the harmful effects of ionizing radiation has motivated the development of whole-body radioactivity monitors in the past decades. The first ones being built in the 1950s, the International Atomic Energy Agency listed 181 monitors in operation in 1969 [3]. In addition to routine monitoring, the monitors have been used for medical diagnostics and metabolic studies of radioactive tracers. Today commercially available whole-body counters are widely used for nuclear safety and medical imaging.

In this master's thesis the capabilities of modern signal processing electronics and list mode data acquisition are investigated, with special interest in measurements of plutonium and americium from the human lungs. Pulse shape analysis methods are applied in an attempt to distinguish background events from true signal-like events.

2 Theoretical aspects

2.1 Plutonium

Plutonium is a metallic radioactive element that is produced from uranium in nuclear reactors and used in nuclear weapons, as a nuclear reactor fuel and in nuclear batteries, for example in satellites. Most plutonium isotopes decay through alpha radiation into uranium. ^{241}Pu and ^{239}Pu beta decay into americium. Although alpha particles have short range and cannot penetrate deep into material, inhaled plutonium can be dangerous because alpha particles are highly ionizing and because plutonium can stay in lungs and deposit in bone matter for many years.

Plutonium's alpha decay can be accompanied by low energy gamma rays when the daughter nucleus relaxes from it's excited state (Fig. 1). Some plutonium isotopes beta decay into americium and neptunium. A typical sample of plutonium contains a range of different isotopes and their respective decay products, which themselves are also radioactive. Table 1 lists the decay properties of a few plutonium isotopes.

Plutonium can be released into the environment from research facilities, nuclear weapons testing, disposal of nuclear waste or from nuclear weapons production facilities. Most of the plutonium found in nature originates from nuclear bomb tests conducted before the 1980's. Plutonium in the atmosphere is deposited on the ground through dry and wet deposition and can accumulate in food chains. Generally plutonium concentrations are very low, typical reported figures are of the order of a few Bq per kilogram of soil and a few hundred nBq per cubic meter of air.

When plutonium is inhaled, the emitted alpha particles damage lung cells, causing lung cancer and other diseases. Part of the plutonium can enter the bloodstream and travel to the kidneys, liver or spleen, or concentrate in bone material, causing further damage. If ingested through contaminated food, it passes through the body because the stomach cannot absorb it. The biological half-life of ^{239}Pu in the lungs is 500 days and on the bone surface up to 50 years [4].

Metabolic models for plutonium in man are derived from animal studies and a few studies made on man. The absorption of plutonium and other actinides can be influenced by chemical composition, particle size, form and mass of the ingested material, drugs and diet, among other variables. Measured absorption fractions have a wide spread of values, varying up to three orders of magnitude even in the same species and the same chemical form of the intake [5].

Epidemiological methods have been used to estimate the health effects of exposure to plutonium. Studies of plutonium workers in Los Alamos have been unable to link exposure to higher cancer risk or mortality rates [6]. Studies of workers at the Mayak nuclear plant in the former Soviet Union have shown increased lung cancer risks for higher plutonium doses [7]. Epidemiological studies' statistical effectiveness is limited by the fact that there are few people who have been exposed to plutonium.

Table 1: Decay properties of a few plutonium isotopes, extracted from the Nudat database [8].

Isotope	**Decay mode**	**Q-value**	**Halflife**	**Daughter nucleus**
^{237}Pu	EC: 100 %	$Q_{EC} = 220$ keV	45.7 d	^{237}Np
^{238}Pu	α: 100 %	$Q_\alpha = 5600$ keV	87.7 y	^{234}U
^{239}Pu	α: 100 %	$Q_\alpha = 5200$ keV	24100 y	^{235}U
^{240}Pu	α: 100 %	$Q_\alpha = 5300$ keV	6560 y	^{236}U
^{241}Pu	β: 100 %	$Q_\beta = 21$ keV	14.3 y	^{241}Am

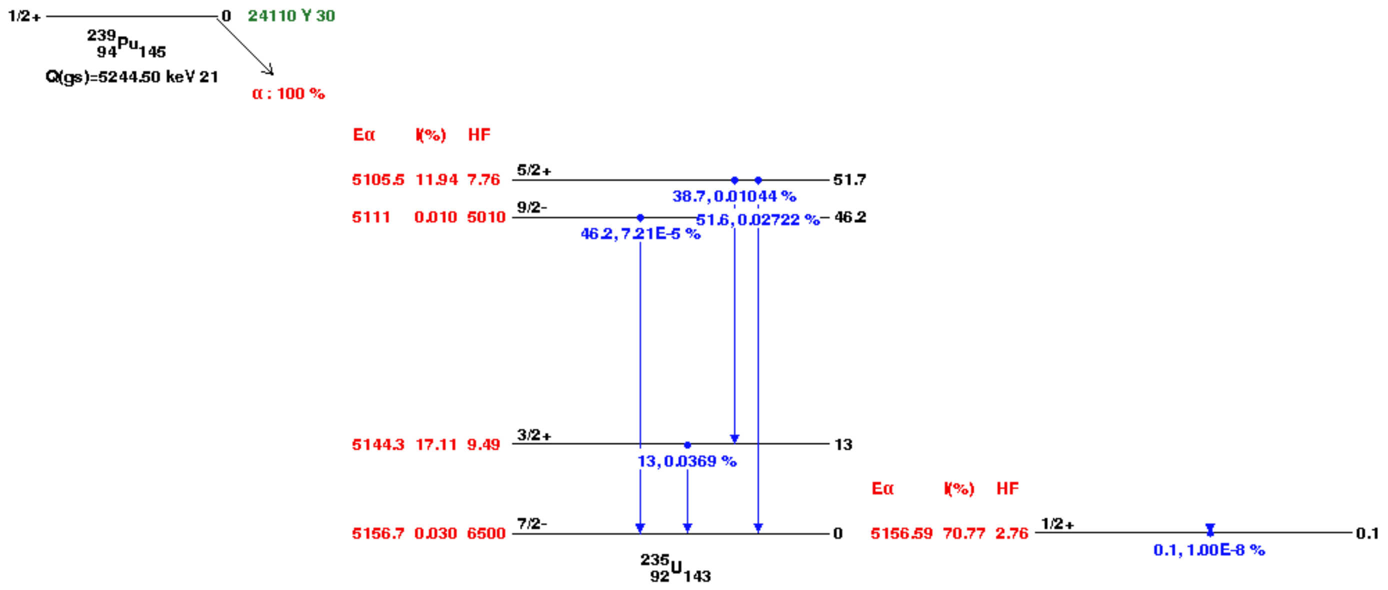


Figure 1: Decay level scheme for ^{239}Pu decaying to the groundstate of ^{235}U, extracted from the Nudat database [8]. The 51.6 keV gamma ray is the gamma ray of interest because it has higher intensity than the 38.7 keV gamma ray and is in the measurable region, unlike the 13 keV gamma ray.

2.2 Interactions of radiation with matter

The three most important interactions of gamma rays in radiation detectors are the photoelectric effect, Compton scattering and pair production. The energy of the incident photon is either partially or completely absorbed in these interactions, and the probability of a certain interaction occuring within a certain material depends on the energy of the photon and on the properties of the matter in question.

The mass attenuation coefficient μ/σ of an element (Fig. 2) is a measure of how strongly the radiation is absorbed in that element. It is defined with

$$\frac{I(x)}{I_0} = e^{-\mu x} = e^{-\frac{\mu}{\sigma}\cdot\sigma x}, \tag{1}$$

where I_0 is the original intensity of the radiation, $I(x)$ is the intensity after the radiation has traveled distance x in the material, μ is the linear attenuation coefficient and σ the density of the material.

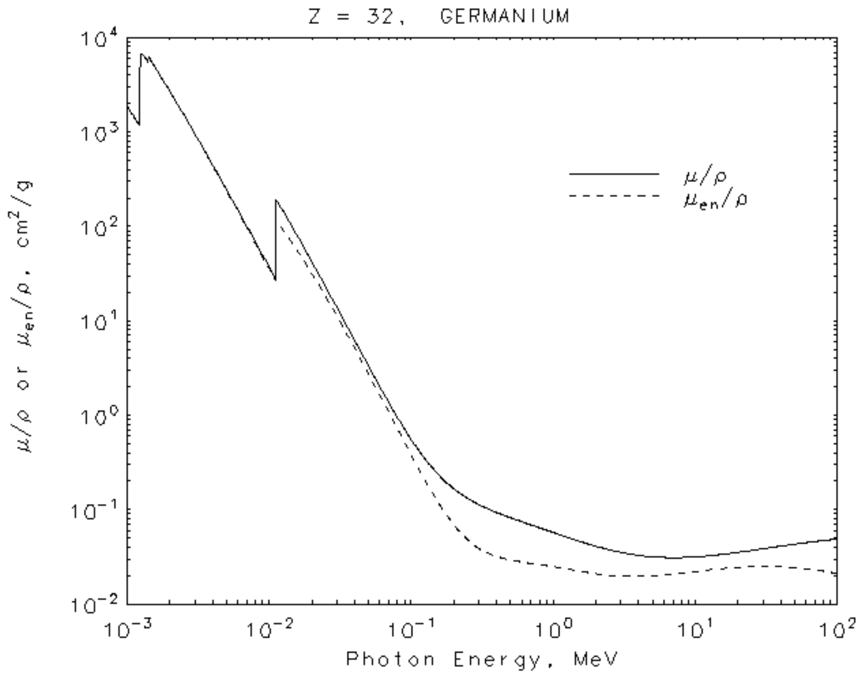


Figure 2: Mass attenuation coefficients of germanium as function of photon energy [9].

In the photoelectric effect the photon is completely absorbed by an atom, resulting in the atom ejecting a photoelectron. This creates a vacancy in the atom, which is quickly filled by another electron from the other electron shells, generating x-ray radiation characteristic to the shell structure of the atom. This mode of interaction is dominant for photons of relatively low energies.

In Compton scattering, the incident photon scatters from an electron, transferring part of the initial photon energy to the recoil electron in the process. The scattered photon has lower energy, but can interact again. The scattering probability increases linearly with the atomic number of the interacting matter, as the number of available electrons increases.

Pair creation may occur when the energy of the gamma ray is above the energy threshold required to create an electron-positron pair, namely 1.022 MeV. All the excess energy of the photon goes to the kinetic energy of these particles and the photon is destroyed in the process. The positron eventually annihilates with an electron, creating two 511 keV annihilation photons. This interaction mode is most probable for photons of high energies.

The attenuation coefficient varies with the energy of the incident radiation as the different interaction modes vary in strength with the energy (Fig. 2). The probability for photelectric absorbtion and Compton scattering decreases with increasing photon energy. There are discontinuous jumps at electron shell energies when electrons from higher shells become available for the photoelectric effect. Above the 1.022 MeV threshold when pair production becomes available, the attenuation coefficient starts increasing.

In gamma-ray spectroscopy, information about the original radiation is obscured by scattered photons, secondary x-ray radiation and annihilation photons. The favorable interaction mode is where the whole energy of the incident photon is deposited in the detector material [10].

2.3 Semiconductor detectors

The electrical conductivity of semiconductor materials falls between that of insulators and conductors. At room temperature, a small number of electrons are thermally excited from the valence band to the conduction band. The energy gap between the bands is of the order of 5 eV or greater for insulators and 1 eV for semiconductors.

The electrical conduction in seminconductors can be finely adjusted by adding dopant materials, either electron donors or acceptors, that respectively produce an excess of either electrons or holes in the material. These are called n-type and p-type semiconductors, corresponding to either negatively or positively charged primary charge carriers.

When p- and n-type semiconductors are brought into contact, the primary charge carriers from each material diffuse across the junction. Holes and electrons join, forming a depletion region void of charge carriers at the junction. The fixed donor sites left behind create an electric field across the depletion region, so that the diffusion of charge carriers eventually stops.

Semiconductor detectors consist of p- and n-type electrodes and a depletion region between them. Absorption of gamma rays in the active region generates charge carriers, both holes and electrons, which are then collected with an electric field. The charge carriers induce a charge of opposite polarity at the electrodes, which is in turn converted into a voltage pulse with a charge sensitive preamplifier [11].

Elements with four valence electrons can accept and give electrons as easily, making them a preferred choice for semiconductor applications. Carbon in it's diamond form has very strong covalent bonds with an energy gap of 5.5 eV at room temperature, making it behave more like an insulator. The remaining non-metallic elements in the carbon group, silicon and germanium, have energy gaps of 1.11 eV and 0.66 eV at room temperature respectively, and are the most used elements in semiconductor detectors. [12]

Germanium crystals can be manufactured with active regions centimeters wide, while silicon crystals can not be made thicker than a few millimeters. The wider active region is needed to detect higher energy gamma-rays. Silicon can be used to detect x-rays, low energy gamma-rays and short-ranged charged particle radiations. Germanium also has higher detection efficiency due to it's higher atomic number. Silicon's higher transparency to high energy gamma-rays can be an advantage over germanium in some applications [10].

2.4 Detector response to gamma radiation

While the generation of charge carriers at one interaction point can be considered instantaneous, the time it takes to collect the charge carriers depends on detector geometry and resistivity, location of the interaction, gamma-ray energy and the strength of the electric field inside the detector. When the specifics of the detector are known, the interaction location can in principle be deduced from the output signal [13, 14, 15].

The analysis of the detector response to deduce the interaction location is complicated by events where the gamma ray energy isn't absorbed in one interaction, but charge carriers are generated at multiple interaction points. These multi-site events are a result of Compton scattering and pair production. Often the majority of events in full energy peaks are multi-site events, where the full energy is captured after multiple scatterings [16].

Multi-site events are also caused by coincidence summing, where gamma rays from the same gamma ray cascade enter the detector at the same time and are detected as one, and in pile-up, where the measured activities are so high that the average time between two consecutive pulses is shorter than the length of an individual pulse, such that the electronics of the measurement system cannot distinguish between the two. Uncorrected pile-up results in reduced counts in the full energy peak and summation peaks higher up in the energy spectrum.

2.5 Pulse processing electronics

A traditional analog spectrum data acquisition system consists of a preamplifier, a shaping amplifier and a multi-channel analyzer. Different pulse shaping methods are used for different applications (Fig. 3), such as fast timing or coincidence counting. Common pulse shaping tasks include pile-up rejection, baseline corrections and discriminating signal from background noise.

Alternatively, a digital signal processor (DSP) can be used. The continuous preamplifier pulse is digitized with a flash ADC and pulse shaping tasks in the discrete time domain are implemented in the digital signal processor. Digital signal processing allows pulse shaping parameters to be optimized in software and pulse shapes to be recorded for off-line analysis. This is known as list mode data acquisition [17].

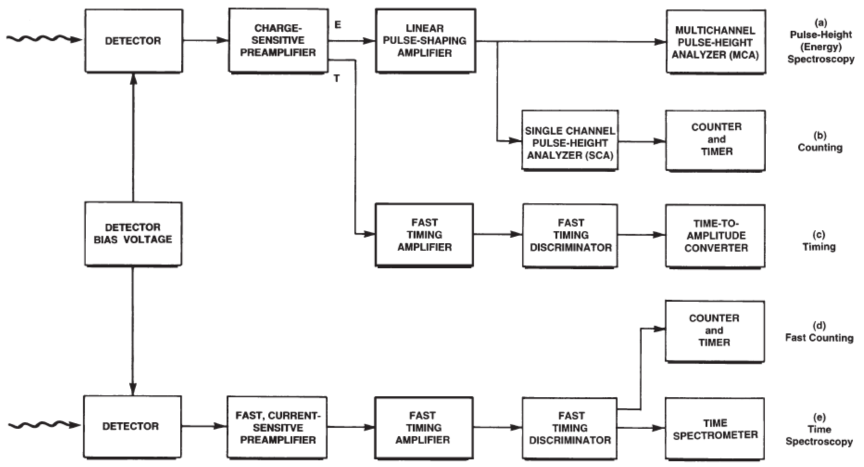

DETECTOR
CHARGE-SENSITIVE PREAMPLIFIER
E
T
LINEAR PULSE-SHAPING AMPLIFIER
MULTICHANNEL PULSE-HEIGHT ANALYZER (MCA)
(a) Pulse-Height (Energy) Spectroscopy
SINGLE CHANNEL PULSE-HEIGHT ANALYZER (SCA)
COUNTER and TIMER
(b) Counting
DETECTOR BIAS VOLTAGE
FAST TIMING AMPLIFIER
FAST TIMING DISCRIMINATOR
TIME-TO-AMPLITUDE CONVERTER
(c) Timing
COUNTER and TIMER
(d) Fast Counting
DETECTOR
FAST, CURRENT-SENSITVE PREAMPLIFIER
FAST TIMING AMPLIFIER
FAST TIMING DISCRIMINATOR
TIME SPECTROMETER
(e) Time Spectroscopy

Figure 3: Pulse processing modules needed for different spectroscopy applications [17].

2.6 Whole-body counting

Whole-body counting is used to measure the amount of gamma ray emitting radionuclides in the human body, to estimate the radiation dose they give. Typical applications include monitoring of persons occupationally exposed to radioactive substances, routine monitoring of members of the public and screening for internal contamination in a radiation emergency. Whole-body counting can also give information about the route of intake and biokinetic behavior of the radionuclides in the body.

Measuring living subjects brings with it a range of limiting factors and uncertainties. For one, the measurement cannot last too many hours, at least not in one sitting. Collecting enough statistics to detect low activities is not possible without very effective detectors and shielding from background radiation. Different subjects having different anatomy give rise to uncertainties in calibration. Calculations of effective doses require one to know the time and route of intake and the metabolism of the materials in the body, which also introduces uncertainties [18].

Both semiconductor detectors and scintillator detectors can be used in whole-body counters (Fig. 4). HPGe detectors are more expensive to use because of the cooling they require, but have a superior energy resolution. Scintillator detectors operate in room temperature and offer a better detection efficiency for applications where short measurement length is valued over the ability to identify specific radioisotopes. NaI(Tl), CsI(Tl) and $LaBr_3$(Ce) are typical scintillating materials used. The use of silicon detectors for detecting low energy gamma radiation in whole-body counting has also been investigated [19].

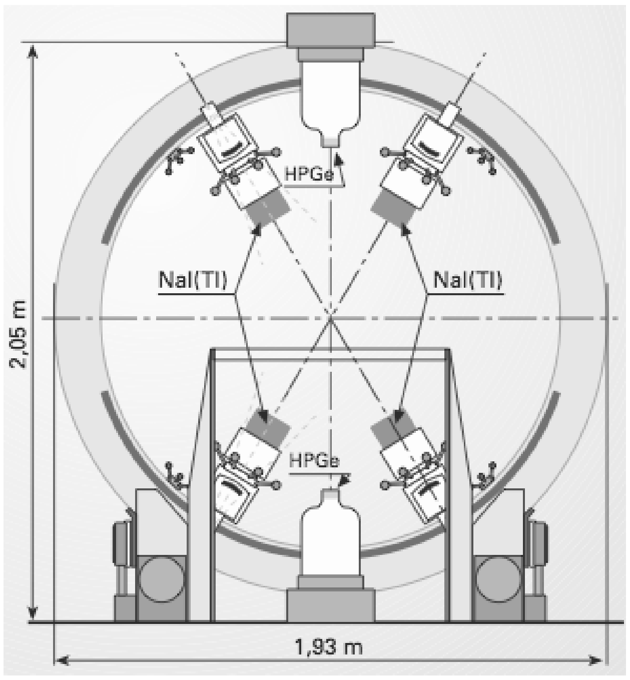


Figure 4: Example geometry of a whole-body measurement setup. NaI scintillation detectors and the topmost HPGe semiconductor detector are installed in a support ring around a motorized bed. An additional HPGe detector is installed under the bed. The bed moves slowly during measurement, so that the subject's complete profile is measured. The whole setup is installed in a low background chamber to reduce the amount of background radiation [20].

3 Statistical methods

3.1 Counting statistics

Because radioactive decay is a random process, any two identical measurements will most likely not give identical results. Analysis of counting statistics is needed to calculate results and estimate uncertainties from measured spectra.

The measurement of radioactivity can be thought of as a sequence of independent measurements, where each nucleus that has a chance to decay represents an independent trial [10]. The probability that a gamma ray is detected in a small time frame t is

$$p = \epsilon F(1 - e^{-\lambda t}), \tag{2}$$

where ϵ is the detector efficiency, F is gamma ray emission probability and $(1 - e^{-\lambda t})$ is the decay probability with time constant λ being a property of the radioisotope. With n trials, the probability distribution of x successful detections is the binomial distribution

$$P_n(x) = \frac{n!}{(n-x)!x!} p^x (1-p)^{n-x}. \tag{3}$$

When the detection probability p is small, the distribution can be approximated with a Poisson distribution, and when the mean number of counts $\bar{x}$ is large enough, the distribution can be further simplified to a Gaussian form

$$G(x) = \frac{\sqrt{2}}{\pi \bar{x}} e^{\frac{-(x-\bar{x})^2}{2\bar{x}}}. \tag{4}$$

The longer the measurement is, the closer the mean of the Gaussian distribution is to the theoretical true value. The standard deviation of a Gaussian distribution is the square root of its mean:

$$\sigma_G = \sqrt{\bar{x}}. \tag{5}$$

If the background count rate B changes between measurements, a best estimate of the rate is given by the arithmetic mean. With N measured background count rates B_i, the average background count rate is

$$\bar{B} = \frac{\sum\limits_{i=1}^{N} B_i}{N}. \tag{6}$$

The experimental standard deviation of the background count rate $\bar{B}$ is

$$\sigma_{exp.,\bar{B}} = \sqrt{\sum_{i=1}^{N} \frac{(B_i - \bar{B})^2}{N-1}}. \tag{7}$$

3.2 Energy calibration

The spectrum given by a multichannel analyzer or a digital signal processor is a spectrum of channels, each corresponding to a specific range of signal heights. The energy calibration that relates channels to energies is done by recognizing peaks with known gamma ray energies and then finding a linear or quadratic relation between channels and energies. Common peaks used for energy calibration are the ^{60}Co 1173 keV and 1320 keV peaks, the 511 keV annihilation peak and the 1460 keV peak of ^{40}K.

3.3 Efficiency calibration

The two ways to calculate the counts in a given peak are to either sum directly the number of counts in the peak area or to fit an analytical function, usually a Gaussian distribution, to the data and integrate the area under it. Sometimes an exponential trailing edge is added to the low energy side of the peak to account for incomplete charge collection. The sum limits are a fixed number of channels around the peak, the integration limits a number of Gaussian widths. Background counts under a peak of interest can be removed by estimating a continuous background from the areas around the peak.

An efficiency calibration is needed to convert the observed counts to a more meaningful measure of source activity. The efficiency depends on the intrinsic efficiency of the detector for detecting photons and the geometrical efficiency of the measurement setup. For point sources the geometrical efficiency is defined by the opening angle to the detector window, for more complex source geometries the efficiency can be calculated with Monte Carlo simulations.

For *in vivo* measurements, the efficiency calibration is done by using phantoms with incorporated radionuclides. When the detector response to specific levels of radiation in the phantom is known, the radioactive contents of a measurement subject can be calculated. The accuracy of the calibration depends on how well the geometry, absorption properties and distribution of radioactive elements match between the calibration phantom and the subject [10, 18].

3.4 Limits of detection

A common problem in measurements of low activities is how to decide if a signal is truly present in the spectrum or if a peak is caused by random fluctuations of the background. There exist a number of standards on how to present such decision limits [21, 22]. These concepts, most famously presented by Lloyd Currie [23], are shortly reviewed here for completeness of this section.

For a measurement of given length, the critical limit L_C is the minimum number of counts needed for the sample to be considered to have non-zero activity. For each limit L_C there is a corresponding probability α that the signal was actually caused by random chance, called false detection probability. False detections, or false positives, are called type I errors.

In the following it is assumed that the measured number of counts is a random variable given by a related Gaussian distribution describing each measurement setup. If the distribution of counts in blank background measurements is known to have mean $\bar{B}$ and standard deviation σ_B and a measurement of a sample has S counts with standard deviation σ_S, the net counts $N = S - \bar{B}$ has standard deviation

$$\sigma_N = \sqrt{{\sigma_S}^2 + {\sigma_B}^2}. \tag{8}$$

For a sample with no activity the standard deviation of the net counts becomes

$$\sigma_0 := \sigma_{N=0} = \sqrt{{\sigma_B}^2 + {\sigma_B}^2} = \sqrt{2}\sigma_B. \tag{9}$$

Net counts in a measurement of a blank sample has probability α to exceed the critical limit. The critical limit for net counts is given by

$$L_C = k_{1-\alpha}\sigma_0, \tag{10}$$

where $k_{1-\alpha}$ corresponds to the $(1-\alpha)$-quantile of the standard normal distribution (Fig. 5).

Type II errors, false negatives, happen when a true activity is falsely discarded as blank. The detection limit L_D is the smallest net signal that has probability β of giving a measurement result that is then discarded by the critical limit L_C. Typically α and β are chosen to be 0.05, but different values can be used. With this definition, the detection limit is

$$L_D = L_C + k_{1-\beta}\sigma_N, \tag{11}$$

where $k_{1-\beta}$ corresponds to the (1-β)-quantile of the standard normal distribution.

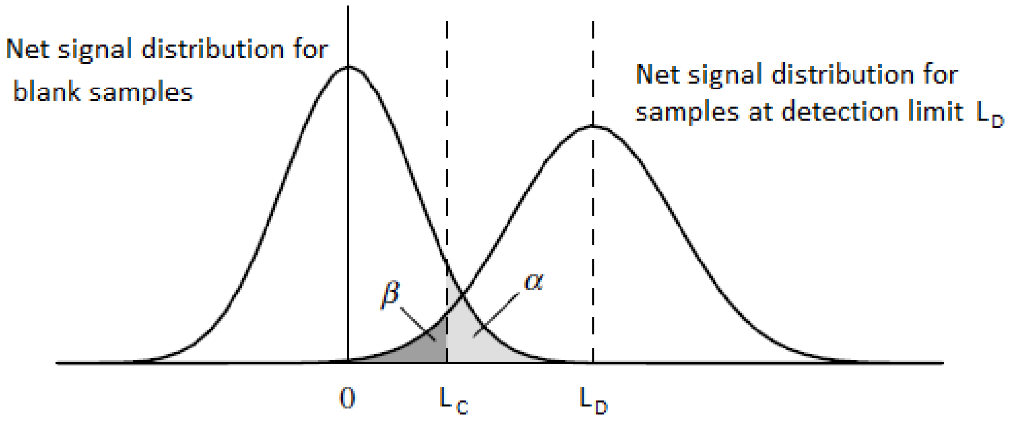

Net signal distribution for blank samples
Net signal distribution for samples at detection limit L_D
β
α
0
L_C
L_D

Figure 5: At the detection limit L_D the $(1-\beta)$ confidence limit equals the $(1-\alpha)$ detection threshold [24, 25].

When α and β are chosen to be 0.05, the mean blank count $\bar{B}$ is sufficiently high and the counting statistics allow the variances to be calculated with equation (5), the expression for the detection limit can be simplified using equations (8-11) to

$$L_D = k_{1-\beta}^2 + 2L_C = 2.71 + 4.65\sqrt{\bar{B}}. \tag{12}$$

The detection limit can be converted to the minimum detectable activity A_{MDA} with

$$A_{MDA} = \frac{L_D}{f\epsilon t}, \tag{13}$$

where f is radiation yield per disintegration, ϵ is the absolute counting efficiency and t is the counting time.

The minimum detectable activity is a measure of the performance of a measurement setup. Like all statistical evaluations, it works best when there are enough statistics. If the count rates are low, say only a few counts can be measured in a reasonable amount of time, the real chances for type I and II errors to happen will differ a lot from the parameters used in this model [10, 24, 25].

As an example use of the above formulae, let's consider a measurement of a sample that may contain ^{137}Cs. The blank background sample is measured for 3000 seconds, giving 5000 counts with an absolute counting efficiency $\epsilon = 15\%$. The radiation yield per disintegration for the 661.7 keV gamma ray of ^{137}Cs is $f = 0.851$ [8]. Using equations (12) and (13), the minimum detectable activity is

$$A_{MDA} = \frac{2.71 + 4.65\sqrt{\bar{B}}}{f\epsilon t} = \frac{2.71 + 4.65 * \sqrt{5000}}{0.851 * 0.15 * 3000s} \approx 0.866Bq. \qquad (14)$$

4 Pulse shape analysis

4.1 Pulse shape analysis

The goal in pulse shape analysis is to define parameters to describe the pulse shapes and to formulate a discrimination rule based on these parameters, where as many signal-like events are accepted and as many background-like events are rejected as possible. The improved peak-to-background ratio should then allow smaller activities of radioactive materials to be found.

The typical preamplifier signal of a germanium detector features a sharp rising edge followed by an exponential decay as the charge storing capacitor in the amplifier discharges (Fig. 6). The amplitude of the charge pulse is directly proportional to the absorbed gamma-ray energy and the time constant of the exponential decay is a property of the amplifier circuit. Information about the charge collection can be looked for in the rising edge of the pulse. Signal rise times for germanium detectors are typically below 300 ns [17, 26].

The time derivative of the charge pulse, the current pulse, gives information about the charge collection speed. Peaks in the current pulse originate from the highest densities of charge carriers arriving at the charge collecting electrode. If there are multiple peaks, each corresponds to one cluster of charge carriers created at one interaction location. A wider current peak corresponds to a more spatially spread out distribution of charge carriers [16].

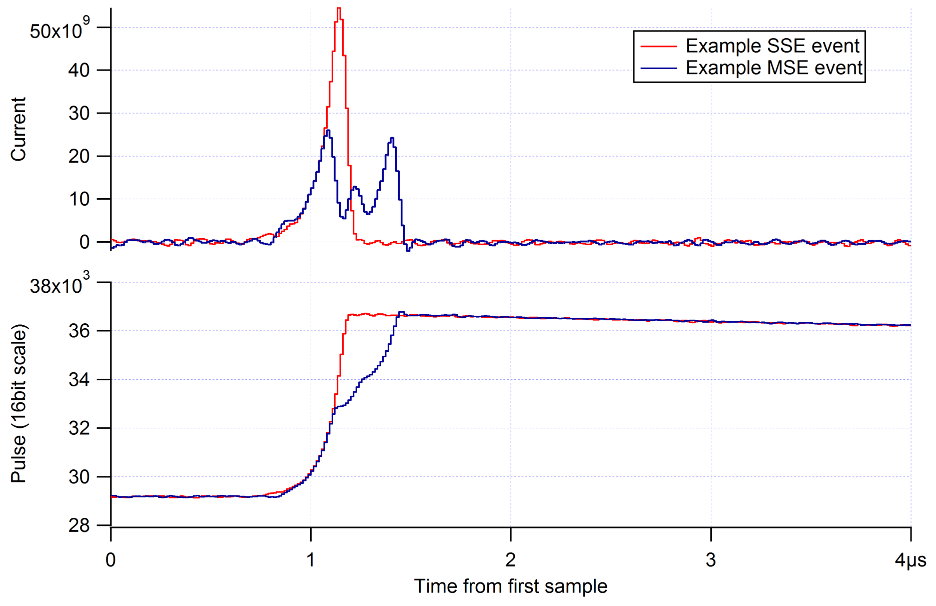


Figure 6: Example SSE and MSE pulses with equal energy. The MSE current pulse has much smaller amplitude compared to the SSE pulse.

While counting the number of peaks in the current pulse would be sufficient in identifying multi-site events from single-site, identifying small overlapping peaks from signal noise can be difficult. A more effective method is to consider the width and amplitude of the current pulse (Fig. 6). Because the area of the current pulse is proportional to energy E, current pulses with just one peak generally have higher amplitude A than pulses with multiple peaks. This makes the ratio A/E an interesting parameter for rejecting multi-site events. Similarly MSE pulses are generally wider than SSE pulses.

When measuring localized sources placed in front of the detector, the density of interactions in the detector crystal decreases with detector depth as the radiation is attenuated in material. Assuming the interactions of background events are evenly distributed in the detector volume, it is beneficial to limit data collection to the events happening closest to the source. Selecting a depth limit is a trade-off between detection efficiency and improved peak-to-background ratio. The ability to implement depth limits through pulse shape analysis would be cheaper than creating detector crystals with application specific optimal crystal depths.

The pulse rise time t_P when the signal has accumulated a percentage P of its amplitude can be used as an indicator of interaction depth [26]. Charge generated close to the collection electrode is collected faster than charge generated near the edges of the detector. Relating signal rise times to specific interaction depths requires detailed information about the detector.

Databases of pulse shapes for each interaction location can be built with either measurements or computer simulations. A measurement setup would consist of a tightly collimated incident beam and an additional collimated detector for detecting 90° scattered gamma rays [27]. Pulse shape simulations are done by defining the geometry of the detector, calculating the electric field and the trajectories of charge carriers inside the detector and using the Shockley-Ramo theorem to calculate the induced charge pulse.

Figure 7 shows T30 risetimes calculated from the simulated pulses of a BEGe detector [26]. The z-coordinate refers to the depth of the detector and the x-coordinate to the position along the disk shaped detector's diameter. The T30 risetimes vary between 0-120 ns. The speed of charge collecting depends on the mobility of charge carriers in the semiconducting material and the strength and geometry of the electric field. Charge generated near the edges of the detector form slower rising pulses. The region where x is between 26-57mm looks promising for linking interaction depth with pulse risetime.

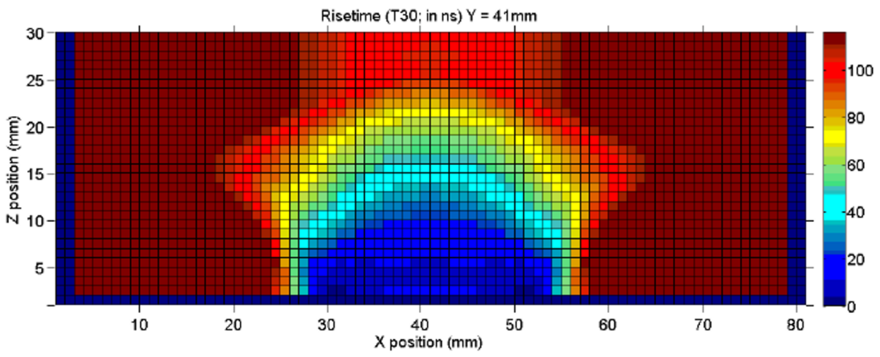


Figure 7: Signal rise times in nanoseconds for a BEGe detector with 3500 V operating voltage, calculated from simulated pulses in a 1 mm grid [26]. Assuming the disk shaped crystal is completely symmetrical and has no imperfections, the pictured middle slice gives complete information of the rise times in the whole crystal volume.

4.2 Rejection limit and rejection fraction

Rejecting background events from signal-like events requires a suitable parameter to describe the pulse shapes and a rejection limit to tell the different event types apart. Ideally there would be a binary parameter, 1 for true event and 0 for background, but since a full energy event and a background event can produce identical pulse shapes, only a fraction r_B of the background events can ever be rejected. At the same time a fraction r_S of true signal-like events are rejected. The background rejection fraction r_B can be calculated from the background count rates before, $A_{B,i}$, and after, $A_{B,f}$, the rejection rule has been applied:

$$r_B = \frac{A_{B,i} - A_{B,f}}{A_{B,i}}, \tag{15}$$

Similarly, the signal rejection fraction r_S is

$$r_S = \frac{A_{S,i} - A_{S,f}}{A_{S,i}}, \tag{16}$$

for count rates $A_{S,i}$ and $A_{S,f}$ of the peak before and after discrimination. For the ideal discrimination method $r_B = 1$ and $r_S = 0$, so that all background events are rejected. For a more realistic but still effective method $r_B >> r_S$ should hold.

Pulse shape discrimination methods can only work if the background events are sufficiently different from the signal events. One such case is when signal events are dominantly SSE and background events dominantly MSE. The discrimination rule can either be decided from a calibration measurement and then applied to an actual measurement of the sample, or formed from the actual measurement itself. A calibration measurement is useful when the same rejection rule is to be used for multiple measurements, or to monitor changes in the parameter values over time.

Figure 8 illustrates the different situations where pulse shape discrimination is possible, of limited use and near impossible. The pulse shape parameter values of background events form a continuum in the energy region of interest. The pulse shape parameter values of events in peak A are easily separated from the background. Rejection limits can be placed above and under the parameter values of peak B to remove some of the background events. The variance of parameter values for peak C is larger than the variance for background events. After removing the events in the background continuum, only few signal events are left.

The same idea holds when using multiple pulse shape parameters: the discrimination is more effective when the value ranges for the background events and the events of interest do not overlap too much.

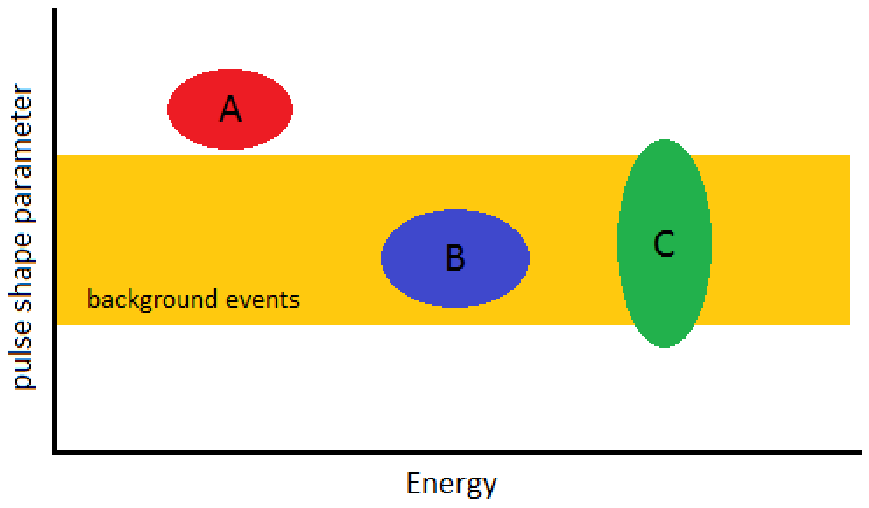


Figure 8: Sketch of the use of pulse shape parameter distributions in finding discrimination limits. The ovals A, B and C represent higher concentrations of events, corresponding to their respective peaks in the energy spectrum. Events in A can be completely separated from background, the background for B can be reduced and background rejection can be done at the cost of counting efficiency for events in C.

4.3 Formulae for pulse shape parameters

The software used in this work, Igor Pro, operates in arrays of data called waves. Each array consists of indexes i starting from 0, their corresponding x-values $x[i]$ and data points $y(x) = y[i]$. Indexes and x-values are linked through an x-scaling factor, so that the x-values are always equally spaced. Igor Pro has a number of inbuilt routines for working with waveform data; these were used wherever applicable [28]. The formulae for the pulse shape parameters are presented here in the format they were implemented in code (Appendix 1), and are different from their original definitions [26, 29] only by a few normalization constants.

The digital signal processor digitizes waveforms by taking samples of the analog signal's amplitude in fast time intervals. The resulting array is filled with integer measures of pulse height and each array index corresponds to a measure of time through the fixed sampling rate (Fig. 9). All pulse shape parameters are calculated from this charge pulse $Q(t)$ and it's derivative, the current pulse $j(t) = dQ/dt$.

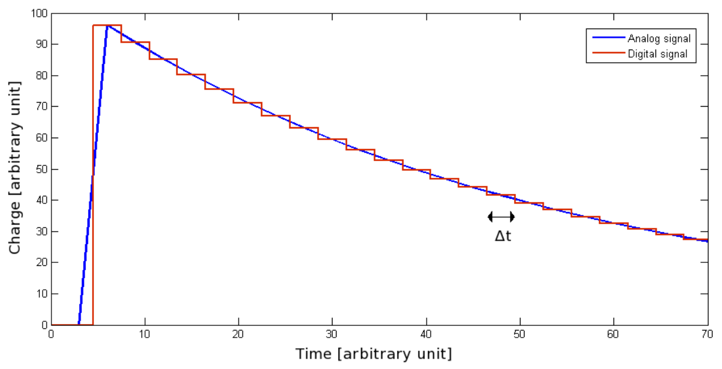


Figure 9: An analog-to-digital converter (ADC) measures the height of an analog pulse at a constant sampling frequency $f = 1/\Delta t$. At higher sampling frequencies the resulting digital signal comes closer and closer to the original analog pulse shape. The digital pulse can be stored to be analyzed after the measurement.

The amplitude of the current pulse A divided by energy E is an efficient parameter for separating MSE and SSE events [16]. The current pulse is numerically differentiated from the charge pulse $Q(t)$. The electronic noise of the detector and data acquisition is smoothed with a moving average algorithm with a 67 ns window. Figure 6 on page 17 shows example SSE and MSE pulses with equal energy.

Rise time t_P is defined as the time when the charge pulse $Q(t)$ has reached $P\%$ of the maximum. For digitized signals with a non-zero baseline, the rise time is defined from the relation

$$Q(t_P) = Q_{min} + \frac{P}{100}(Q_{max} - Q_{min}). \tag{17}$$

To find t_P, the rising edge of the charge pulse is searched from the maximum towards the left. To make the resulting rise time more independent of variations in triggering, the 30%/90% of full height rise times $T30$ and $T90$ are calculated with $T30 = t_{30} - t_{10}$ and $T90 = t_{90} - t_{10}$.

The width of the current pulse is another good parameter for finding multi-site events. Wider, more spread out pulses are usual for MSE events. Full width at 10% maximum is calculated from

$$W = t_{10,trailing\ edge} - t_{10,leading\ edge}, \tag{18}$$

where the t_{10} values are searched to the left and to the right of the current pulse maximum.

The asymmetry of the current pulse is a measure of the pulse "skewness". Asymmetry tells if the majority of charge carriers arrived "early" or "late" relative to the middle point of the pulse (Fig. 10). It is calculated from the front and back areas F and B of the current pulse with

$$S = \frac{F - B}{F + B}, \tag{19}$$

where

$$F = \sum_{i=N_0}^{N_{mid}-1} j[i]\Delta t, \tag{20}$$

$$B = \sum_{i=N_{mid}}^{N_f} j[i]\Delta t. \tag{21}$$

The time granularity of the indices i is given by Δt, the summation limits N_0 and N_f are the indices corresponding to the t_{10} values the width of the current pulse was calculated from, and N_{mid} is the middle point of the pulse, calculated with

$$N_{mid} = \frac{N_0 + N_f}{2}. \tag{22}$$

Asymmetry values vary between -1 and 1. Forward-leaning pulses have positive values and backward-leaning pulses have negative values.

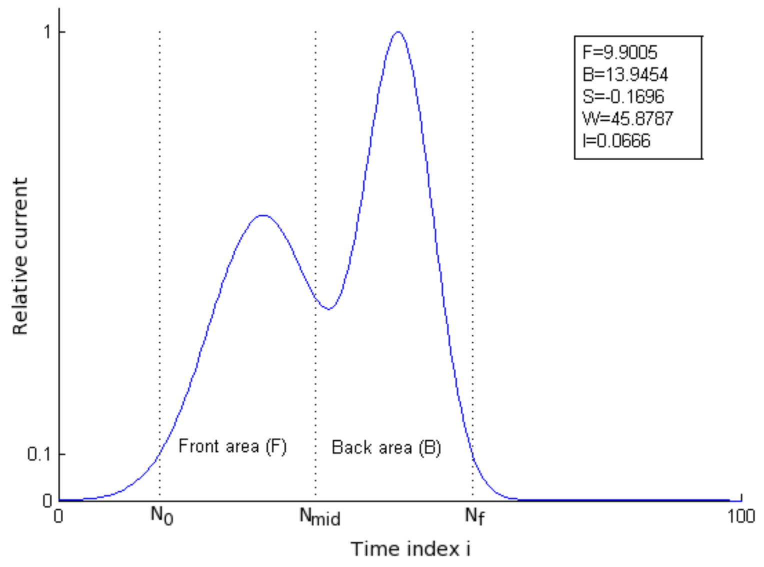


Figure 10: Current pulses are typically not symmetrical. The asymmetry parameter S tells if the majority of the charge carriers arrived ”fast” or ”slow”, relative to the rest of the pulse.

Asymmetry can be used to tell MSE and SSE events apart, since MSE pulses feature multiple peaks and SSE pulses just one peak; SSE pulses are more likely to be symmetrical. Asymmetry can also be useful when interactions in one part of the detector tend to produce forward leaning pulses and backward leaning pulses can be attributed to another region. Usually this distinction would be between regions close to the collecting electrode and regions further away from it. Both holes and electrons contribute to the net signal, but must travel paths of different length at their respective drift velocities, explaining the different shapes of SSE events [29].

One problem with using the t_{10} values for the summation limits is that they are blind to secondary peaks that occur beyond those points (Fig. 11). Alternatively the front and back areas can be calculated relative to the location of the current pulse maximum, and N_0 and N_f values chosen so the front and back areas have a fixed width. Fixed limits may leave out part of the pulse too, or extend far beyond it, but they may work better if there are a lot of pulses with secondary peaks that can not otherwise be accounted for.

Pulse moment I is another measure of how the current is spread in the current pulse. While asymmetry distinguishes forward and backward leaning pulses from symmetric ones, pulse moment tells if most of the current is close to the centre of the pulse or further away from it. Pulse moment I is calculated with

$$I = \sum_{i=N_0}^{N_f} \frac{j[i](N_{mid} - i)^2}{(F+B)W^2}, \tag{23}$$

where F, B and W, from equations (20), (21) and (18), are used to normalize the values. If the current pulses have a lot of noise extending to negative values, it may be helpful to replace $j[i]$ with $max(0, j[i])$ to limit the negative effect of noise in these equations.

Figure 12 illustrates how the pulse moment I can distinguish pulses of equal asymmetry S. MSE pulses are generally more spread out than SSE pulses, so their pulse moments are also typically higher.

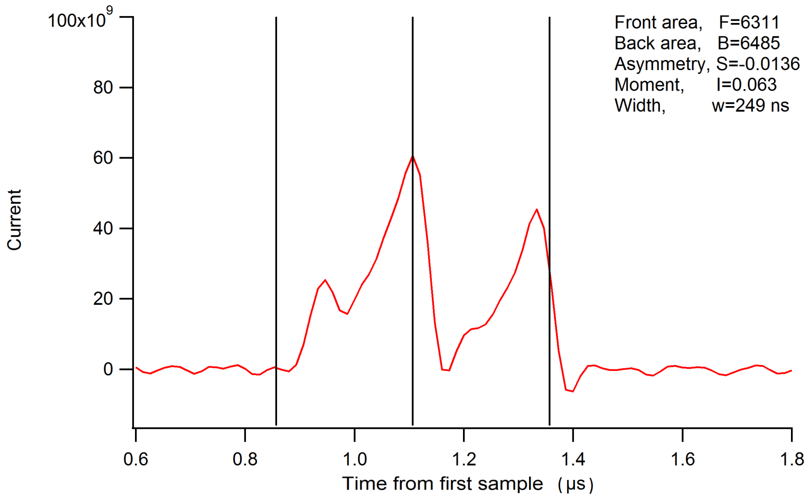


Figure 11: Calculating asymmetry with fixed limits has the chance to find secondary peaks that would occur beyond t_{10} values. Here the values are calculated with respect to the current maximum, with 250 ns windows in both directions.

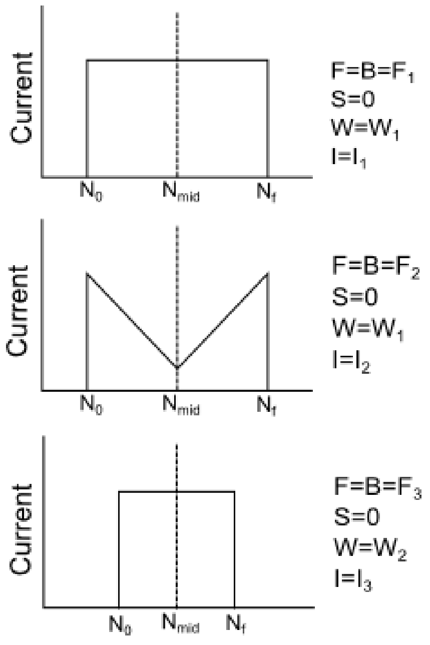


Figure 12: The pulse moment I describes how tightly packed the charge carriers are to the middle of the pulse. Pulses with the same asymmetry S can be distinguished by differening pulse moments.

5 Experimental setup

5.1 Whole body phantom

The anthropomorphic radiodosimetric whole body phantom, Radek model ARDF-09T, consists of head, neck, torso and knee joint phantoms, each with a number of detachable parts. The phantom is modeled after an 18-20 year old male and is made of materials with radiation absorption properties comparable to that of human tissue and bones.

The chest phantom used in this work has replacement lungs, heart, liver and kidneys with incorporated radionuclides ^{241}Am and ^{239}Pu, but of the radioactive organs, only the replacement lungs were used in this work. The lung activities were 43 kBq of ^{241}Am and 191.3 kBq of ^{239}Pu, respectively for the two sets of radioactive lungs. The radioactive materials are evenly distributed within the phantom material. Any trace quantities of other radioactive isotopes within the phantom were not documented by the manufacturer.

The phantom was placed 5 cm away from the detector window, so that the detector was centered on the middle of the chest (Fig. 13). To simulate natural radiation in humans, ^{40}K rods were placed under the phantom, such that the 1460 keV peak count rate was matched with a reference human measurement. These rods increase the Compton background in the low-energy region, and an attempt is made to reject the Compton-scattered events through pulse shape analysis.

5.2 Low background measurement room

The steel measurement room has walls of 150 mm steel, 3 mm lead and 4 mm copper to shield against background radiation. The room accommodates enough space for bed-geometry whole-body counting, but the new setup had not yet been built. The air-conditioning system and concrete for surrounding walls have been chosen so the background radiation level would be as low as possible.

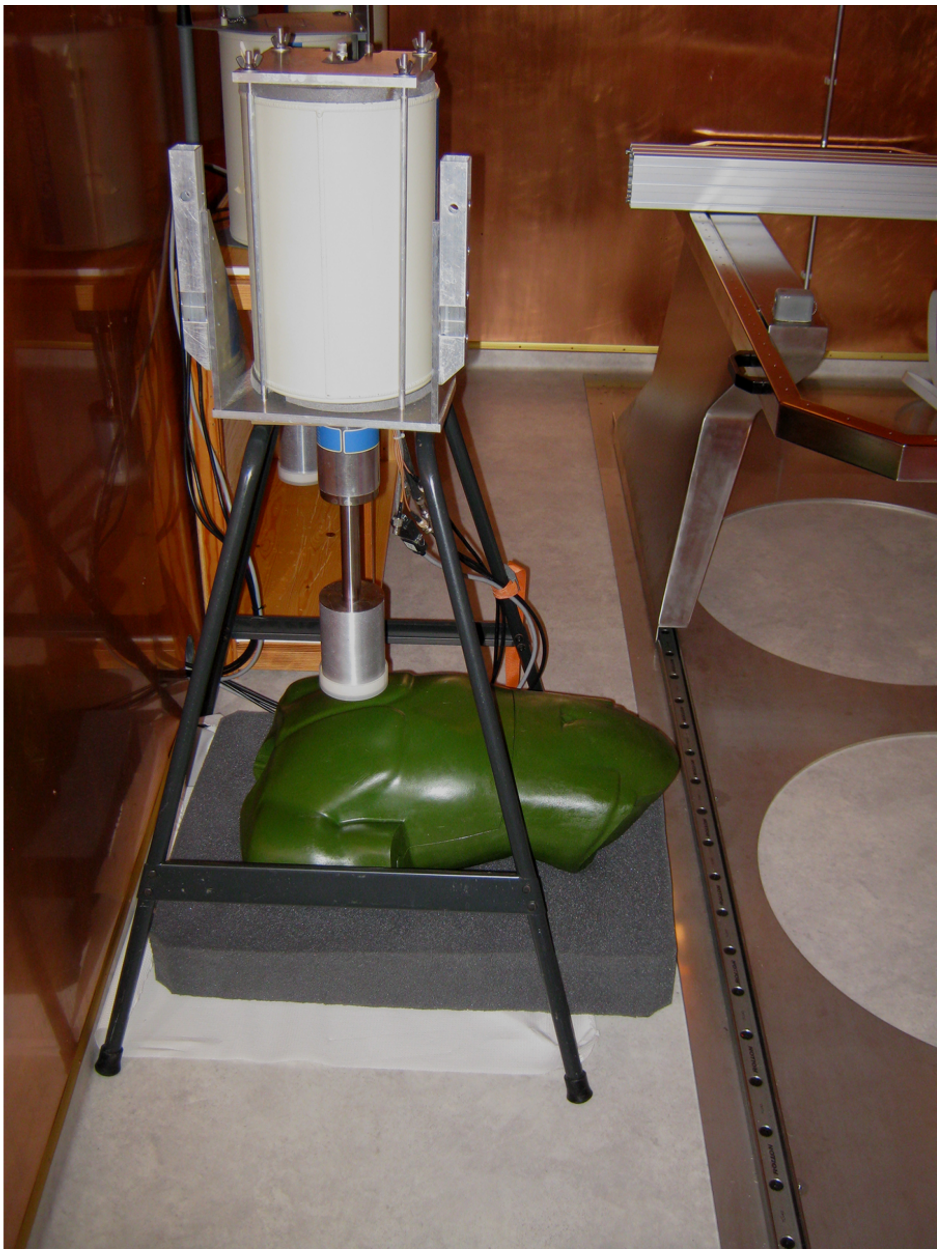


Figure 13: Photo of the measurements with the phantom in the low background chamber.

5.3 Detector specifications

The Broad Energy Germanium (BEGe) detector used was Canberra model BE3820. BEGe detectors cover the energy range from 3 keV to 3 MeV and have high resolution even at low energies (Table 2). The disk shaped crystal has 70 mm diameter, 20 mm thickness and an active area of 38 cm^2. The 0.5 mm thick carbon epoxy entrance window is 5 mm away from the crystal. All measurements were done at the recommended bias voltage of 4000 V.

Table 2: Resolution of the BEGe detector used.

Energy	FWHM, manufacturer	FWHM, measured
59.5 keV	0.660 keV	0.53 keV
1332.5 keV	2.100 keV	1.37 keV

5.4 Electronics and data acquisition

The energies and pulse shapes are collected with a XIA Pixie-4 Digital Gamma Finder board, which was mounted in a National Instruments PXI-1033 chassis (Fig. 17). Pixie-4 has four input channels which can be operated separately or in coincidence mode. Pulse heights are calculated to 16-bit precision and can be binned up to 32000 channels. Events are timestamped with 37.5 MHz clock frequency. Pile-up inspection, pulse shaping and triggering parameters can be adjusted through software [30].

To measure the energy of a gamma ray from the digitized preamplifier pulse, Pixie-4 uses a fast trigger filter to detect the arrival of the gamma ray and a slow energy filter to determine an amplitude that is directly proportional to the gamma-ray energy (Fig. 15). The equation used for these trapezoidal filters is

$$V_{filter,k} = \frac{1}{L}(-\sum_{i=k-2L-G+1}^{k-L-G} V_i + \sum_{i=k-L+1}^{k} V_i), \tag{24}$$

where V_i are the digitized voltage points of the preamplifier pulse, L is the trapezoid's rise time and G is the length of the trapezoid's flat top (Fig. 14).

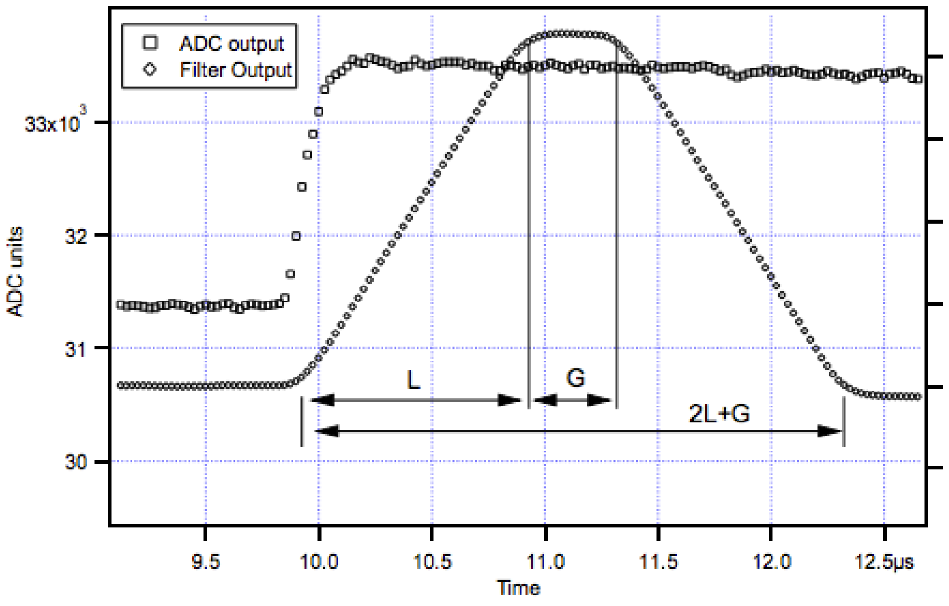


Figure 14: Trapezoidal filtering in the Pixie-4. The amplitude of the ADC pulse and the amplitude of the filter pulse are both proportional to the energy of the detected gamma ray. [30]

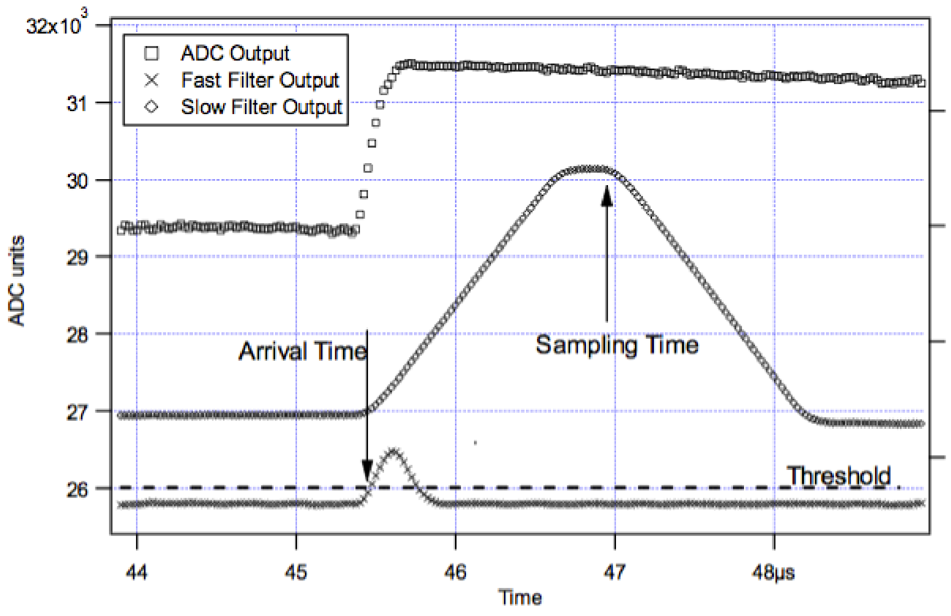


Figure 15: The fast trigger filter is used to detect when a gamma ray comes in and the slow energy filter is used to measure the energy. The trigger threshold needs to be high enough to not trigger from signal noise, but low enough to catch low energy events. [30]

The energy filter gives a correct measure of the gamma-ray energy only if the pulse is sufficiently separated from the neigbouring pulses (Fig. 16). With long filtering times or high pulse frequencies, unhandled pulse pile-up can become a serious problem. Pixie-4 automatically rejects stacked pulses by requiring trigger pulses to be atleast a duration *PeakSep* apart from each other. This duration is calculated from the energy filter parameters L_{slow} and G_{slow} with

$$PeakSep = L_{slow} + G_{slow} + c, \tag{25}$$

where constant c depends on how many ADC samples are averaged before entering the filtering logic [31].

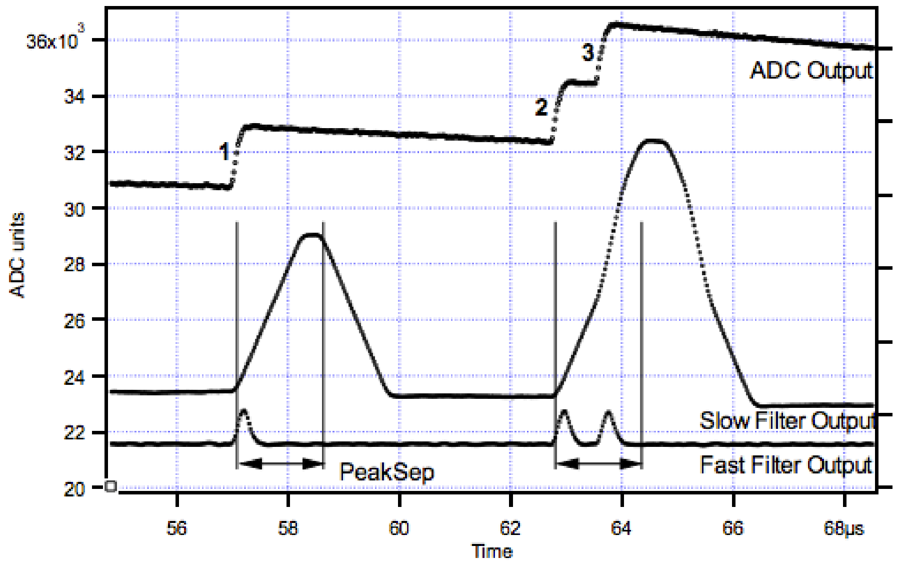


Figure 16: Pixie-4 detects pile-up by inspecting the fast filter for a duration *PeakSep* after the arrival of a pulse. If another pulse arrives in this time, both pulses are rejected. In this figure, the first pulse passes this test, and pulses two and three fail it. [30]

The Pixie Viewer user interface is based on Wavemetrics' Igor Pro [28]. In this work, Igor Pro was also used as a platform for pulse shape analysis. To streamline the analysis process, a custom graphical user interface was programmed in Igor Pro. The interface allows the user to define rejection limits individually for each pulse shape parameter, create energy spectra and histograms of parameter values, calculate rejection fractions and view the rejected and accepted pulse shapes (Fig. 18). The Igor Pro code file for the parameter calculations and the custom interface is appended (Appendix 1).

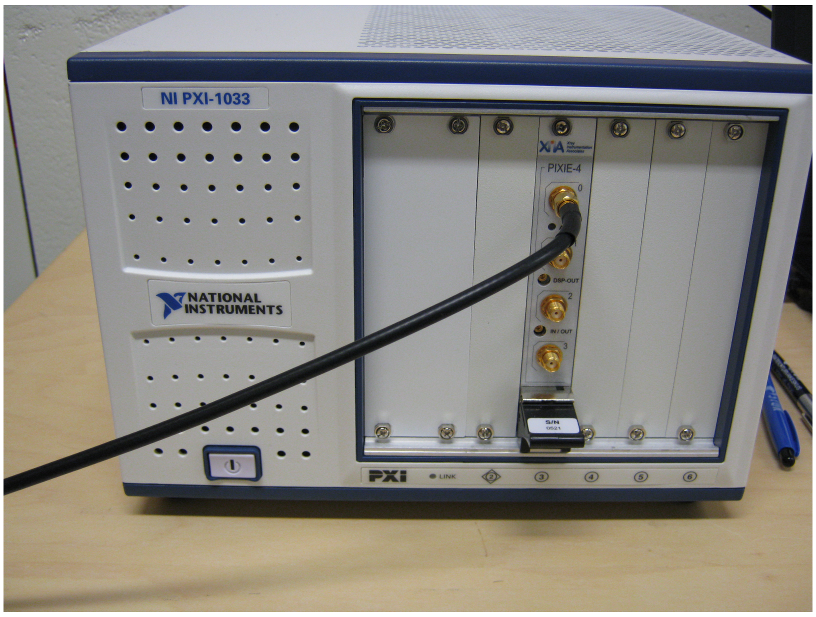
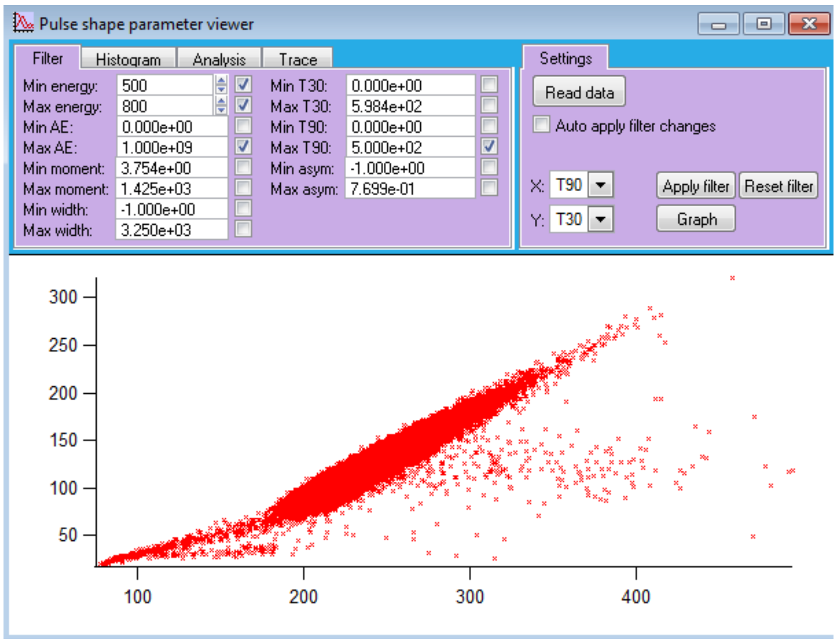


Figure 17: Pixie-4 board mounted in the PXI-1033 chassis.



Min energy:	500	✓	Min T30:	0.000e+00	☐
Max energy:	800	✓	Max T30:	5.984e+02	☐
Min AE:	0.000e+00	☐	Min T90:	0.000e+00	☐
Max AE:	1.000e+09	✓	Max T90:	5.000e+02	✓
Min moment:	3.754e+00	☐	Min asym:	-1.000e+00	☐
Max moment:	1.425e+03	☐	Max asym:	7.699e-01	☐
Min width:	-1.000e+00	☐			
Max width:	3.250e+03	☐			

☐ Auto apply filter changes

X: T90

Y: T30

Figure 18: Screen capture of the interface programmed for the pulse shape analysis. The scatter plot here is the (T90, T30) plot of the 59.5 keV ^{241}Am peak from a calibration measurement. The shape is fairly typical; the majority of events fall in a linear region, while a few events straggle outside it because of their higher T90 values.

6 Results

6.1 Calibration and background measurements

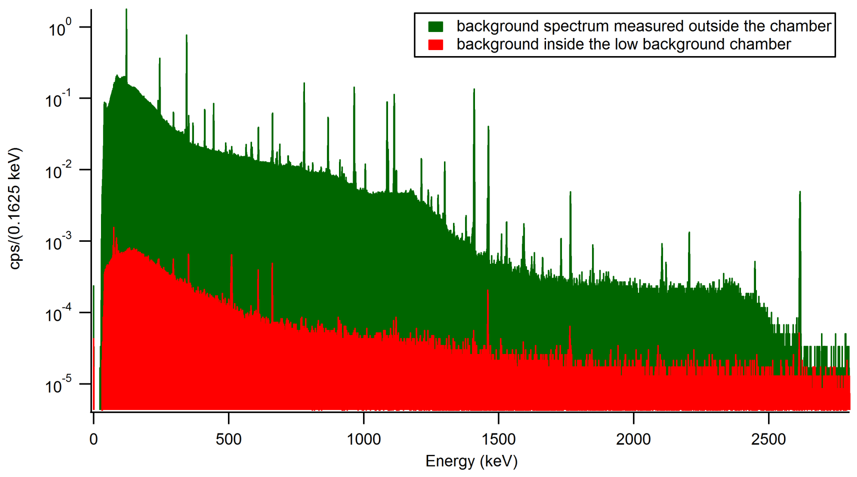


Figure 19: Bacgkround spectra measured inside the low background chamber and in the room next to it.

The energy calibration was done with ^{241}Am and ^{60}Co point sources (Fig. 20). The pulse shaping parameter tau, corresponding to the time constant of the preamplifier, was optimized as instructed in the Pixie-4 user's manual to $\tau = 56.47$. Traces were recorded with length 4.0 μs and delay 1.0 μs. Other trigger filter and energy filter parameters were left to their default values (Table 3) [30]. The background was measured both inside and outside the low background chamber to get an idea how effective the shielding was (Figure 19).

Table 3: Pixie-4 energy and trigger filter parameters (Eq. 24).

	Energy filter	**Trigger filter**
Rise time (μs)	5.973	0.080
Flat top (μs)	1.173	0.080
Threshold	-	25
Tau (μs)	56.47	-

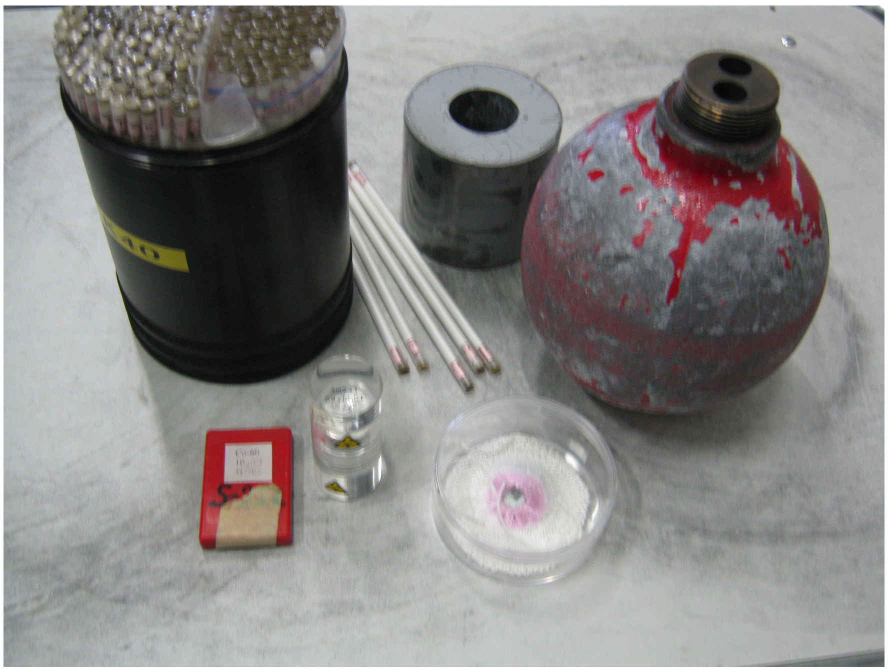


Figure 20: Equipment used in the measurements: ^{40}K rods, collimators, 370 kBq ^{60}Co source, 2582.6 Bq ^{241}Am source and gas mantle containing thorium and other radioactive materials from it's decay chain.

6.2 Measurements with the phantom

Measurements of the phantom revealed a ^{241}Am contamination in the supposedly non-radioactive reference phantom (Fig. 21). According to the manufacturer, the background phantom should not have contained radioactive materials. Table 4 shows the measured count rates in the ^{241}Am 59.5 keV peak for various background phantom pieces. The detector-source distance in these measurements was about 20 cm, varying with the size of the phantom parts. The background was removed by taking the area left of the peak to be indicative of the background counts.

The low energy radiation from americium adds to the background in measurements done with the phantom. The contamination is problematic for pulse shape analysis because the pulse shapes introduced by it are very similar in shape with the signals of the plutonium peaks of interest. More importantly, the assumption that interaction locations of background events would be evenly distributed in the detector volume does not hold, since the gamma rays producing the low energy background come from the same direction and undergo their interactions in the same part of the detector as the radiation of interest.

It is likely that there were multiple isotopes of plutonium in the phantom from the manufacturing process, as separating these isotopes from each other is difficult. The quantities of these trace isotopes were not documented by the manufacturer. While some of the ^{241}Am may come about as a decay product of ^{241}Pu, it is unclear how the radioactive materials ended up in the background phantom.

Rods of ^{40}K placed under the phantom increased the background count rate only little compared to the background caused by the contamination. The appropriate number of rods was determined by matching the 1460 keV peak count rate to a reference human measurement.

Table 4: Results from measurements of individual reference phantom parts, compared to a ^{241}Am point source. Calculated point source activities do not take into account the geometries or absorption properties of different phantom parts, but give an estimate of the order of magnitude of the activity relative to the reference point source.

	^{241}Am 59.5 keV net count rate (cps)	**point source activity (Bq)**
^{241}Am point source	2.84	2582.6
left lung	0.020	17.9
right lung	0.008	6.9
back of torso	0.016	14.1
chest	0.082	74.6
liver	0.014	12.5
stomach	0.0007	0.64
pancreas	0.175	159
heart	0.004	3.8

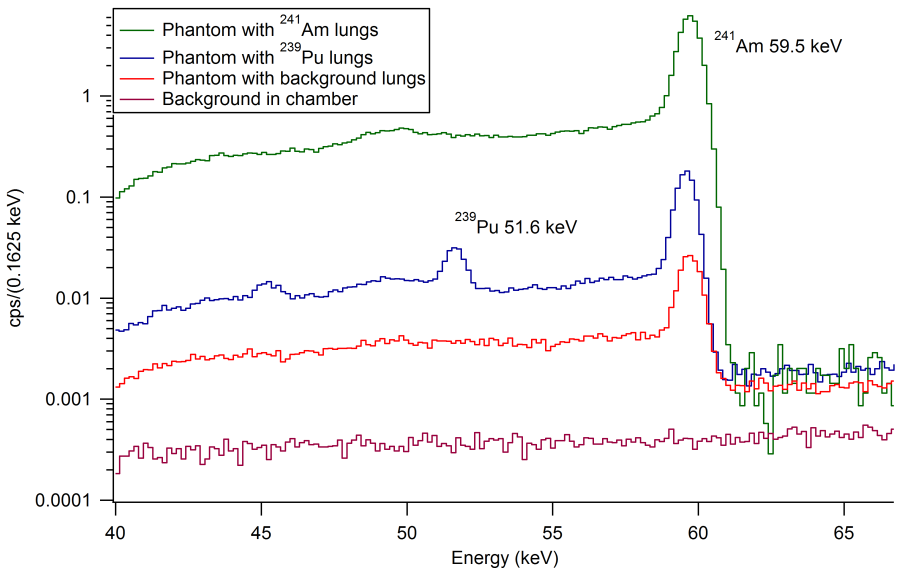


Figure 21: Spectra of the chamber background, background phantom and phantom with ^{239}Pu and ^{241}Am incorporated lungs. Most of the background events in the low energy region are caused by the americium in the phantom pieces. Phantom was measured with ^{40}K rods placed under it, increasing the Compton background slightly.

6.3 Pulse shape analysis in the low energy region

The pulse shape parameters explored were the ratio of current pulse amplitude and energy A/E, charge pulse rise times $T30$ and $T90$, current pulse width W, asymmetry S and normalized moment I. No single parameter or parameter combination was found that would separate background pulse from signal pulse in the low energy region. Plots of pulse shape parameters against energy (Fig. 22) show that the ranges of parameter values for the peaks of interest and their backgrounds are near identical. Few events can be discarded to achieve $r_B > r_S$ for the rejection fractions, but the achieved improvement in peak-to-background ratio is not statistically significant.

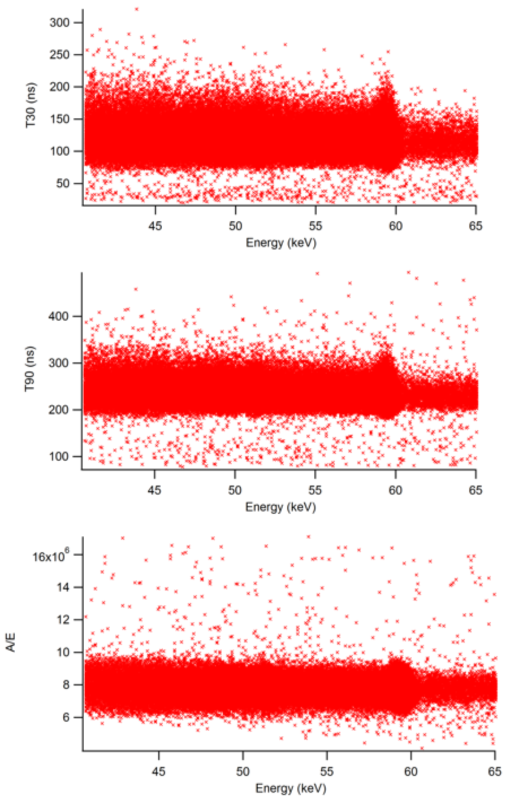


Figure 22: $T30$, $T90$ and A/E distributions in the low energy region for the measurement of the phantom with lungs containing ^{239}Pu. Comparing these with the sketch in Figure 8, it is apparent that effectively no discrimination between signal and background events can be done. Similar distributions were found for the other pulse shape parameters.

6.4 A/E for separating MSE and SSE

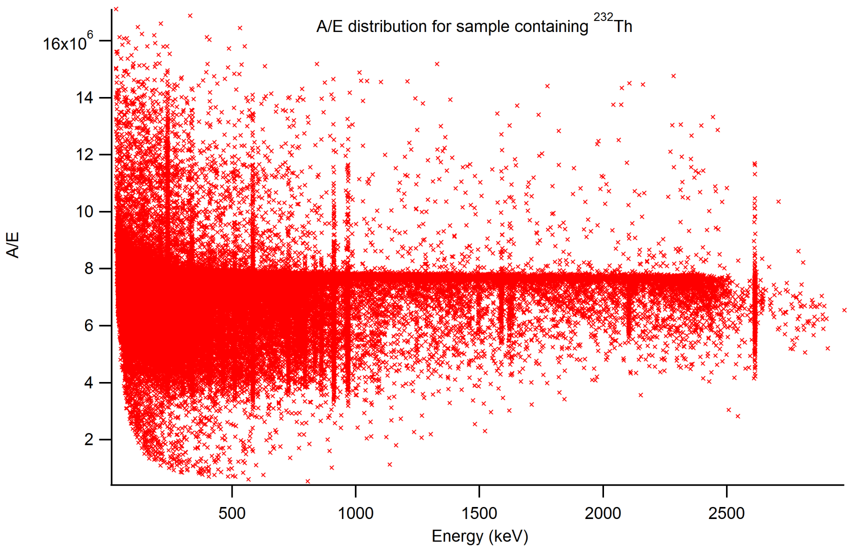


Figure 23: A/E distribution for a sample containing ^{232}Th. The SSE band at $A/E = 8 \cdot 106$, with a slight energy dependence, separates SSE and MSE events.

The distribution of A/E values against energy (Fig. 23) consists of a band of SSE events, with events under the band being more likely to be MSE events. The A/E values are more spread out in the low energy region, mostly because noise in the current pulse is more pronounced at low energies.

The performance of the A/E parameter in separating MSE and SSE events was tested by measuring a source containing ^{232}Th, as its granddaughter nucleus ^{228}Ac has a gamma branch at 1588.2 keV, right next to the double-escape peak of ^{208}Tl at 1592.5 keV. Double escape peaks are dominantly SSE, since a scattered gamma ray is less likely to still be able to produce a electron-positron pair. The 1588.2 keV gamma rays give dominantly MSE pulse shapes, since they are typically fully absorbed only after multiple scatterings [16].

The A/E distribution for the two peaks of interest (Fig. 24) show the expected result, where events in the 1588.2 keV peak have lower A/E values than events in the 1592.5 keV peak. The rejection limit was set at $A/E = 7.5245 \cdot 10^6$, with events below the limit being rejected. With this limit, the rejection fraction for the dominantly MSE peak was 74.8 % and for the dominantly SSE peak 10.4 % (Fig. 25).

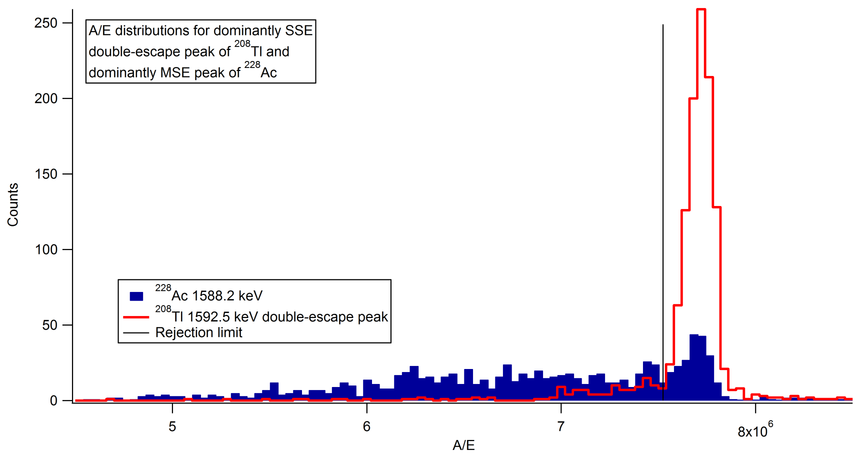


Figure 24: A/E distributions for the double-escape peak of ^{208}Tl at 1592.5 keV and for the 1588.2 keV peak of ^{228}Ac. The rejection limit is at $A/E = 7.5245 \cdot 10^6$.

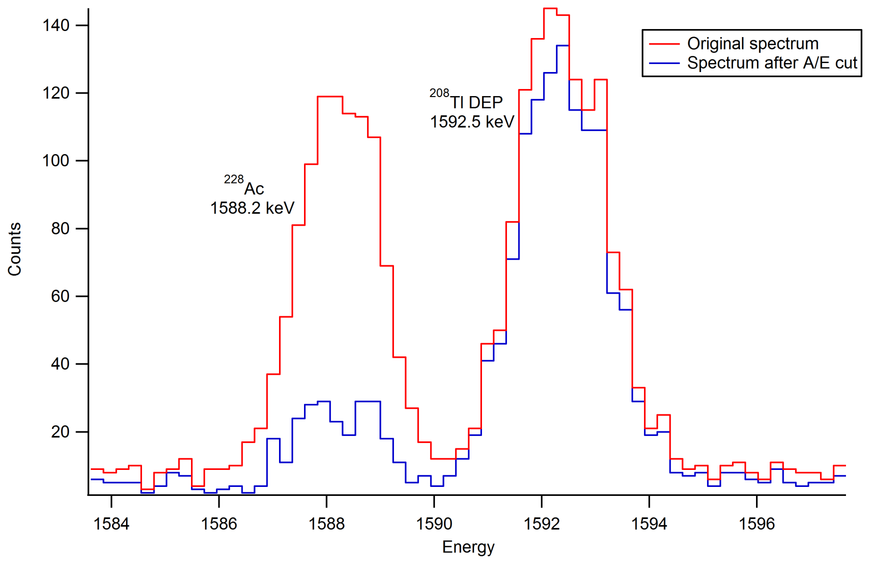


Figure 25: The two peaks of interest before and after the $A/E = 7.5245 \cdot 10^6$ rejection limit was applied.

6.5 Pulse width, asymmetry and moment

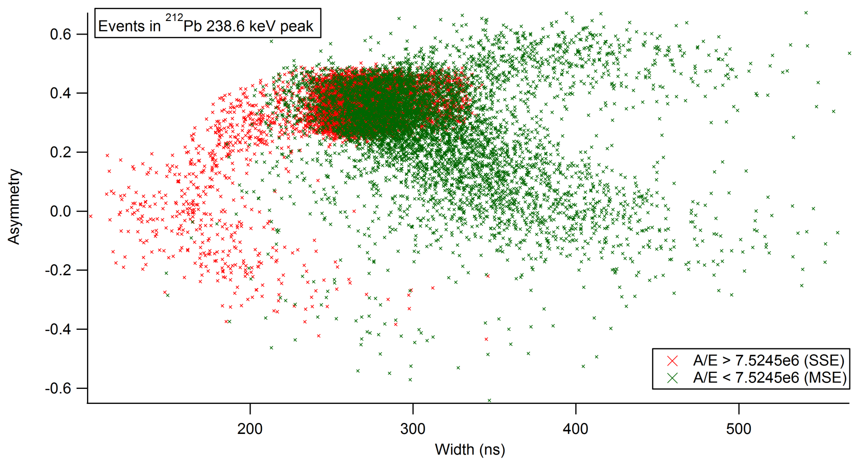


Figure 26: Asymmetry-width distribution of events in ^{212}Pb 238.6 keV peak.

The other pulse shape parameters couldn't separate MSE and SSE events as efficiently as the A/E parameter. Some general observations can be made from their distributions, for example that wider pulses and pulses with higher moment are more likely to be MSE, as expected. Pulses with high width and moment are degenerate with respect to the asymmetry parameter (Fig. 26), corresponding to secondary pulses occuring either before or after the maximum of the current pulse.

6.6 Pulse rise times for depth of interaction separation

Separating events based on depth of interaction was tested with collimated point sources. The charge carriers are collected slower when they are generated further away from the collecting electrode. Low energy gamma rays from ^{241}Am and scattered gamma rays from ^{60}Co were used, with the expectation that high energy gamma rays from ^{60}Co would get closer to the electrode and thus their scattering events would have faster rise times. Collimating to the centre and to the side of the detector was tested to see if there was any difference in pulse rise times.

Collimators with widths 3.5 cm and 1.2 cm were used. Adding blocks of lead to collimate through a 2 mm slit was also tested, but no differences in pulse rise times between low energy gamma rays and scattered high energy gamma rays were found (Fig. 27). The measured values are in line with pulse shape simulations done for this detector model (Fig. 28).

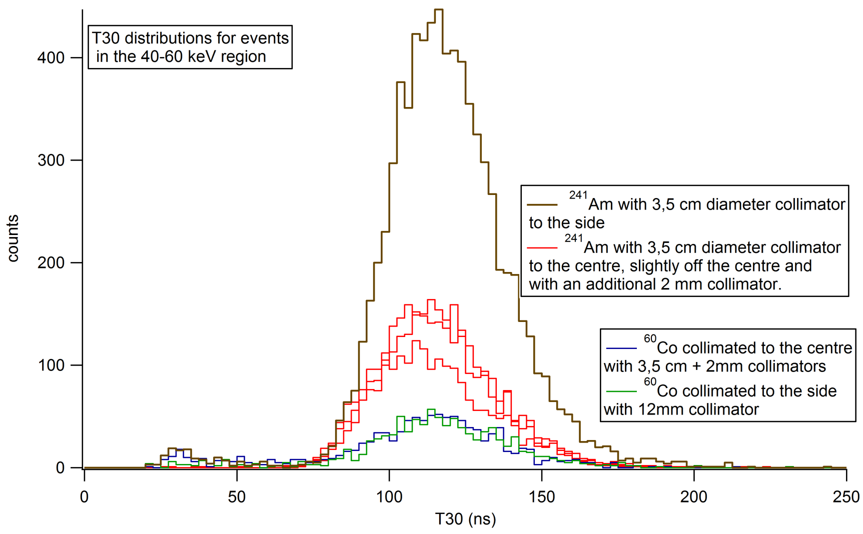


Figure 27: Results from collimator tests with cobolt and americium sources. Rise time $T30$ can not distinguish cobolt's gamma rays' scattering events from americium's full energy events. Few gamma rays get close to the electrode, where $T30 < 50ns$.

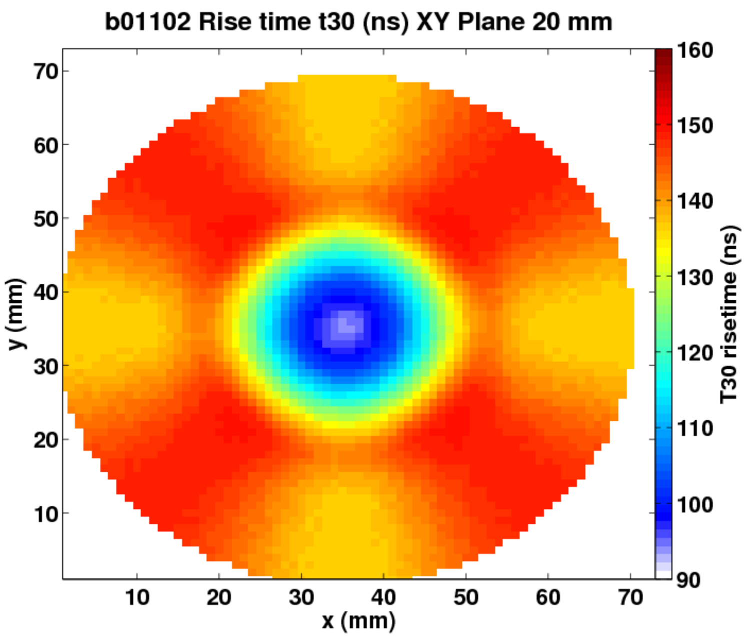


Figure 28: Simulated $T30$ values for the detector model used in the measurements. The simulation method is described in [26].

6.7 Anomalous pulses

Sometimes the digital signal analysis board lets through anomalous pulse shapes, where the energy determination is most likely incorrect (Fig. 29). These pulse shapes are fairly easy to find, as the pulse shape parameters calculated from them are often orders of magnitude different from the usual values. While these anomalies are fairly rare and thus don't affect the measurement statistics much, tracking their numbers may be useful in making sure the measurement system stays stable.

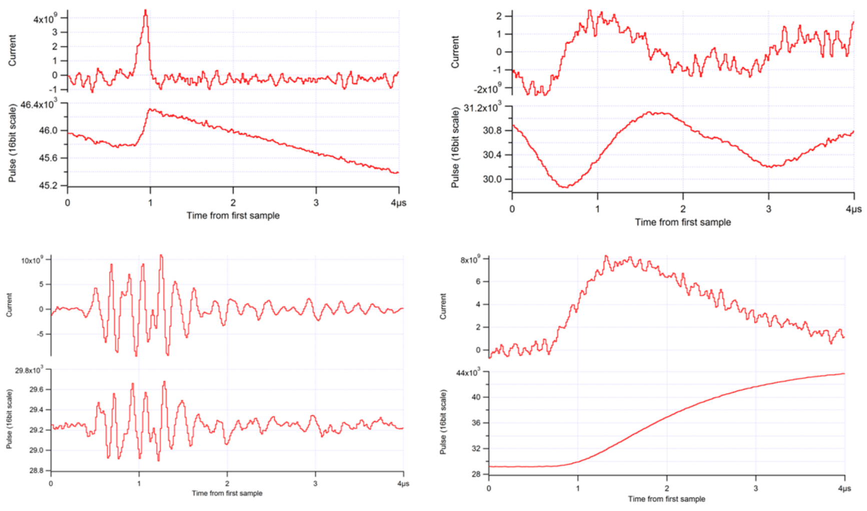


Figure 29: Examples of anomalous pulse shapes captured by the digital signal processor.

7 Discussion

The attempt to reduce Compton background at low gamma ray energies with various pulse shape parameters was ineffective. Parameter distributions for measurements of ^{241}Am and ^{60}Co point sources collimated to the centre of the BEGe detector were effectively identical. Different depth of interaction distributions for high and low energy gamma radiation were expected to produce differing parameter distributions.

Probable causes for the failure of this method are imperfect collimating and pronounced signal noise in low energy pulses. Most likely the variance of pulse shapes coming from a given interaction location is comparable to the variance accross the detector volume, making the determination of interaction location very difficult.

Pulse shape analysis in this fashion is an interesting proposition for many gamma spectroscopy applications. At this point the limiting factor is not necessarily the required computing power, but stability and predictability of the detector response and electronics and the ability of the detector to produce different pulse shapes for different event types, preferably without the need to use collimators to make these differences visible.

The antropomorphic phantom that was to be used in the application of this pulse shape analysis method was found to contain a contamination of radioactive ^{241}Am. The low energy background caused by this contamination would have made the background filtering task very difficult, as the pulse shapes of these background events were effecively identical to the pulses caused by the plutonium radiation. Although any real plutonium inhalation case may also be accompanied by such low energy gamma ray emitting "contaminants", this Compton-reducing method was to be tested against a background of scattered high energy radiation, not against a background of near-identical radiation.

Digital pulse shape analysis may prove to be useful in improving detection limits at low energies, but it will require detailed information about the detector response and more advanced algorithms to find rejection rules for different event types. Meanwhile, the equipment may be used to monitor stability of a measurement setup or to demonstrate the multitude of pulse shapes given by a detector.

It is probably fair to say that a specifically designed Low Energy Germanium Detector (LEGe) is always going to be better suited for low energy measurements than the more general purpose BEGe detector, even if digital pulse shape analysis could be used. Count rate and peak-to-background ratio can be maximized with the correct choice of entrance windows, detector depth and size of contacts among other available manufacturing choices. Another option to get the Compton-reducing effect is to use a Compton shield, a surrounding array of detectors run in anti-coincidence mode with the primary detector. The allure of digital pulse shape analysis is that perhaps the same effect could be achieved without needing to purchase new detectors for a very specific gamma spectroscopy application, if the old detector can be digitally enhanced to do the same work.

Acknowledgements

I would like to thank Research Professor, Dr Paul Greenlees for supervising this work, Kari Peräjärvi and Maarit Muikku from the Finnish Radiation and Nuclear Safety Authority STUK for the topic and their help during the measurements and Dr Laura Harkness-Brennan from the University of Liverpool for providing us with the simulations for our detector.

References

[1] William Moss and Roger Eckhardt. “The Human Plutonium Injection Experiments”. In: *Los Alamos Science* 26 (1995), pp. 177–233.

[2] George A. Taylor. *The Evolution of Internal Dosimetry Bioassay Methods at the Savannah River Site.* Tech. rep. WSRC-MS-2000-00290. Westinghouse Savannah River Company, 2000.

[3] International Atomic Energy Agency. *Directory of whole-body radioactivity monitors.* Vienna, 1970.

[4] U.S. Department of Health and Human Services, Public Health Service, Agency for Toxic Substances and Disease Registry. *Toxicological profile for plutonium.* 2010.

[5] D. M. Taylor. “The metabolism of plutonium and related elements: ICRP Publication 48”. In: *Radiation Protection Dosimetry* 26.1 (1989), pp. 137–140.

[6] George L. Voelz and Ileana G. Buican. “Plutonium and Health: How great is the risk?” In: *Los Alamos Science* 26 (2000), pp. 74–89.

[7] Z. B. Tokarskaya et al. “Interaction of radiation and smoking in lung cancer induction among workers at the Mayak nuclear enterprise”. In: *Health Phys.* 83.6 (2002), pp. 833–846.

[8] National Nuclear Data Center. *Information extracted from the NuDat 2 database.* URL: `http://www.nndc.bnl.gov/nudat2/`.

[9] J. H. Hubbell & S. M. Seltzer. *Tables of X-Ray Mass Attenuation Coefficients and Mass Energy-Absorption Coefficients from 1 keV to 20 MeV for Elements Z = 1 to 92 and 48 Additional Substances of Dosimetric Interest.* The National Institute of Standards and Technology, U.S. Department of Commerce. 1996. URL: `http://www.nist.gov/pml/data/xraycoef/`.

[10] Glenn F. Knoll. *Radiation Detection and Measurement.* John Wiley & Sons, 2000.

[11] Canberra Industries. *Germanium Detectors, User's Manual.* 2003.

[12] Ioffe Physico-Technical Institute. *Physical properties of semiconductors.* URL: `http://www.ioffe.ru/SVA/`.

[13] *Review of the Physics of Semiconductor Detectors.* ORTEC. URL: `http://www.ortec-online.com/download/Review%20of%20the%20Physics%20of%20Semiconductor%20Detectors.pdf`.

[14] P. Désesquelles et al. "Direct determination of the hit locations from experimental HPGe pulses". In: *Nuclear Instruments and Methods in Physics Research A* 729 (2013), pp. 198–206.

[15] A. Olariu et al. "Pulse Shape Analysis for the location of the gamma interaction in AGATA". In: *IEEE Transaction on Nuclear Science* 53.3 (2006).

[16] Raquel Gonzáles de Orduña et al. *Pulse shape analysis for background reduction in BEge detectors.* Tech. rep. EUR 24521 EN - 2010. European Commission JRC, 2010.

[17] *Introduction to Amplifiers.* ORTEC. URL: `http://www.ortec-online.com/download/Amplifier-Introduction.pdf`.

[18] International Atomic Energy Agency. *Assessment of Occupational Exposure Due to Intakes of Radionuclides.* IAEA Safety standards series No. RS-G-1.2. Vienna, 1999.

[19] Jean Louis Genicot et al. "Direct Determination of Radionuclides in the Body, Optimisation of Measurements Parameters and Results Analysis". In: *World Journal of Nuclear Science and Technology* 1 (2011), pp. 87–110.

[20] Tarja K. Ikäheimonen, ed. *Säteily ja sen havaitseminen.* Säteilyturvakeskus, 2002.

[21] *ISO 11929:2010, Determination of the characteristic limits (decision threshold, detection limit and limits of the confidence interval) for measurements of ionizing radiation - Fundamentals and application.* 2010.

[22] *ANSI N13.30 Performance Criteria for Radiobioassay.* 1995.

[23] Lloyd A. Currie. "Limits for Qualitative Detection and Quantitative Determination: Application to Radiochemistry". In: *Anal. Chem.* 40 (1968), pp. 586–593.

[24] *Multi-Agency Radiological Laboratory Analytical Protocols Manual (MARLAP).* Chapter 20. 2004.

[25] J. M. Kirkpatrick, R. Venkataraman, and B. M. Young. "Minimum detectable activity, systematic uncertainties, and the ISO 11929 standard". In: *J Radioanal Nucl Chem* 296 (2013), pp. 10005–1010.

[26] D. Barrientos et al. "Characterisation of a Broad Energy Germanium (BEGe) detector. Simulation and experimental results." In: *Nuclear Science Symposium Conference Record (NSS/MIC).* IEEE. 2010, pp. 662–666.

[27] H. Y. Cho et al. "Pulse Shape Analysis of Induced Charges in a Segmented Germanium Detector by Using the Weighting Potential Method". In: *Journal of the Korean Physical Society* 45.6 (Dec. 2004), pp. 1485–1489.

[28] *IGOR Pro.* Wavemetrics. URL: `http://www.wavemetrics.com/products/igorpro/igorpro.htm`.

[29] R. Suazrez et al. "Real-time digital signal processor implementation of self-calibrating pulse-shape discriminator for high purity germanium". In: *Nuclear Instruments and Methods in Physics Research A* 586 (2008), pp. 276–285.

[30] *User's Manual, Digital Gamma Finder (DGF) Pixie-4.* Version 2.54. XIA LLC. 2013. URL: `http://www.xia.com/Manuals/Pixie4_UserManual.pdf`.

[31] *Programmer's Manual, Digital Gamma Finder (DGF) Pixie-4, Pixie-500.* Version 2.63. XIA LLC. 2014. URL: `http://www.xia.com/Manuals/Pixie4_ProgrammerManual.pdf`.

8 Appendix: Igor Pro code file

```
#pragma rtGlobals=3   // Use modern global access method and strict
    wave access.


//#######################
// Pulse Shape Analysis
//#######################

// Written by Pertti Hallikainen, 2014
//
// Contents
// - Examples of Igor Pro syntax
// - Calculating pulse shape parameters for all traces in a .bin
    file, filtering events based on rejection limits
// - Plotting scatter plots and histograms
// - Pulse shape parameter calculations
// - User interface code

// Usage:
// 0. Run user_globals() to initialize global waves.
// 1. Open a .bin binary file in the List Mode Trace window.
// 2. Macros>Parameter viewer, or run parameter_viewer()
// 3. Press Read Data (this takes a while)
// 4. Press Reset filter
// 5. Mess around with filter values, X and Y waves
// 6. Reset filter - button resets all the filter values that have
    a blank checkbox
// 7. Histogram tab to histogram parameter values
// 8. Analysis tab to compare filtered energy spectrum to original
    spectrum
//      analysis doesn't work if you don't have energy selected as
    the histogram input
// 9. Trace tab to view traces that passed/didn't pass the filter
//   constant fraction lines are at 10%, 30% and 90% of the charge
    pulse
//




//##################################
```

```
//Reference for displaying waves
//displayVs(temp_AE, temp_Energies)
//displayHistogram(root:user:energies,x0=0, x1=18000, Nbins=2000,
    bottomLabel="Channel")
//displayHistogram(root:user:AE, x0=0, x1=1e-14, Nbins=600,
    bottomLabel="A/E"))
//###################################

//##########################
// Examples of Igor pro syntax
//##########################

// Example of custom menu items
Menu "Macros"
   "Display current and charge pulses", Trace_display()
   "Parameter viewer", parameter_viewer()
End

// Example of optional parameters
// Test:
// parameter_test(1,2, c=3, d=4)
// parameter test(1,2)
Function parameter_test(a,b, [c,d])
   variable a,b,c,d
   if(ParamIsDefault(c) && ParamIsDefault(d))
      Print "missing optional parameters c and d."
   endif
   Print a,b,c,d
End

//#########################################################
// - Reading data from binary file
// - Selecting a collection of events based on pulse shape
    parameters
//#########################################################

// Reads data from the currently open .bin file. Use List mode
    trace window to change the file.
// Calculates pulse shape parameters for all the events. This is
    the most computationally intensive part of the analysis.
// Data is stored in global waves at root:user:
Function read_data()
   variable i, NumEvents
   Nvar ChosenModule = root:pixie4:ChosenModule
   wave listmodewave = root:pixie4:listmodewave

   NumEvents = listmodewave[ChosenModule]

// Create waves for parameter values
   Make/O/N=(NumEvents) root:user:AE
   Make/O/N=(NumEvents) root:user:Energies
   Make/O/N=(NumEvents) root:user:Broad
   make/O/N=(NumEvents) root:user:asymmetry
   make/O/N=(NumEvents) root:user:width
   make/O/N=(NumEvents) root:user:risetime30
   make/O/N=(NumEvents) root:user:risetime90
   make/O/N=(NumEvents) root:user:moment

   Wave AE    = root:user:AE
   Wave Energies = root:user:Energies
   Wave Broad    = root:user:Broad
   Wave asymmetry = root:user:asymmetry
   wave width    = root:user:width
   wave risetime30  = root:user:risetime30
   wave risetime90  = root:user:risetime90
   wave moment   = root:user:moment

// channel 0
   Wave trace = root:pixie4:trace0

   variable time_before = datetime
   for(i=0; i<NumEvents; i+=1) // loop through all events
      changeEvent(i)
      Energies[i]     = get_energy()
      AE[i]           = get_AE()
      Broad[i]      = get_broad()
      asymmetry[i]  = get_asymmetry()
      width[i]      = get_width()
      moment[i]        = get_moment()
      normalize(trace)
      wave normalizedWave
      risetime30[i]  = get_dt(normalizedWave, 0.1, 0.3, 1)
      risetime90[i]  = get_dt(normalizedWave, 0.1, 0.9, 1)
      if(!mod(i,25000))
         Print num2str(i)+"/"+num2str(NumEvents)+" events
            calculated."
      endif
   endfor

   print "Process took "+num2str(datetime-time_before)+" seconds,
       "+num2str(i)+"/"+num2str(NumEvents)+" total events"
End

// Clears filters that were not checked
Function reset_filter([checkLocks])
   variable checkLocks
   Wave AE    = root:user:AE
   Wave Energies = root:user:Energies
   wave risetime30  = root:user:risetime30
   wave risetime90  = root:user:risetime90
   wave moment   = root:user:moment
   wave width    = root:user:width
   wave asymmetry  = root:user:asymmetry
   variable lock_minEnergy, lock_maxEnergy, lock_minAE, lock_maxAE,
       lock_minT30, lock_maxT30, lock_minT90, lock_maxT90 =0
   variable lock_minMoment, lock_maxMoment, lock_minWidth,
       lock_maxWidth, lock_minAsymmetry, lock_maxAsymmetry = 0
   if(checkLocks)
      ControlInfo lockMinEnergy
      lock_minEnergy = V_value
      ControlInfo lockMaxEnergy
      lock_maxEnergy = V_value
      ControlInfo lockMinAE
      lock_minAE = V_value
      ControlInfo lockMaxAE
      lock_maxAE = V_value
      ControlInfo lockMinMoment
      lock_minMoment = V_value
      ControlInfo lockMaxMoment
      lock_maxMoment = V_value
      ControlInfo lockMinWidth
      lock_minWidth = V_value
      ControlInfo lockMaxWidth
      lock_maxWidth = V_value
      ControlInfo lockMinT30
      lock_minT30 = V_value
      ControlInfo lockMaxT30
      lock_maxT30 = V_value
      ControlInfo lockMinT90
      lock_minT90 = V_value
      ControlInfo lockMaxT90
      lock_maxT90 = V_value
      ControlInfo lockMinAsymmetry
      lock_minAsymmetry = V_value
      ControlInfo lockMaxAsymmetry
      lock_maxAsymmetry = V_value
   endif
   if(!lock_minEnergy)
      NVAR Emin = root:user:Emin
      Emin = 0
   endif
   if(!lock_maxEnergy)
      NVAR Emax = root:user:Emax
      Emax= Wavemax(energies)
   endif
   if(!lock_minAE)
      NVAR AEmin = root:user:AEmin
      AEmin=0
   endif
   if(!lock_maxAE)
      NVAR AEmax = root:user:AEmax
      AEmax=Wavemax(AE)
   endif
   if(!lock_minMoment)
      NVAR momentMin = root:user:momentMin
      momentMin = wavemin(moment)
   endif
   if(!lock_maxMoment)
      NVAR momentMax = root:user:momentMax
      momentMax = wavemax(moment)
   endif
   if(!lock_minWidth)
      NVAR widthMin = root:user:widthMin
      widthMin = wavemin(width)
   endif
   if(!lock_maxWidth)
      NVAR widthMax = root:user:widthMax
      widthMax = wavemax(width)
   endif
   if(!lock_minT30)
      NVAR T30min = root:user:T30min
      T30min=max(Wavemin(risetime30),0)
```

```
  endif
  if(!lock_maxT30)
     NVAR T30max = root:user:T30max
     T30max=Wavemax(risetime30)
  endif
  if(!lock_minT90)
     NVAR T90min = root:user:T90min
     T90min=max(Wavemin(risetime90),0)
  endif
  if(!lock_maxT90)
     NVAR T90max = root:user:T90max
     T90max=Wavemax(risetime90)
  endif
  if(!lock_minAsymmetry)
     NVAR Smin = root:user:asymmetryMin
     Smin=max(Wavemin(asymmetry),-1)
  endif
  if(!lock_maxAsymmetry)
     NVAR Smax = root:user:asymmetryMax
     Smax=Wavemax(asymmetry)
  endif
End

// Normalizes the wave to have values in [0,1].
Function normalize(w)
  Wave w
  Duplicate/O w, normalizedWave
  smooth/S=2 5,normalizedWave //smooth/B=1 5, normalizedWave
  variable wMax = waveMax(normalizedWave)
  variable wMin = waveMin(normalizedWave)
  normalizedWave = (normalizedWave-wMin)/(wMax-wMin)
End

// This function filters the events stored by read_data() based on
   rules defined in passed_filter()
// The events that pass the filter are stored in temp_parameterName
Function filter_events()
  variable i, NumEvents
  Nvar ChosenModule = root:pixie4:ChosenModule
  wave listmodewave = root:pixie4:listmodewave

// References to global waves
  Wave Energies = root:user:Energies
```

```
   Wave Broad = root:user:Broad
   Wave AE = root:user:AE
   Wave asymmetry = root:user:asymmetry
   wave width = root:user:width
   wave risetime30 = root:user:risetime30
   wave risetime90 = root:user:risetime90
   wave moment = root:user:moment

   NumEvents = numpnts(Energies)

// Create waves for those events that pass the filter
   Make/O/N=(NumEvents) temp_AE
   Make/O/N=(NumEvents) temp_Energies
   Make/O/N=(NumEvents) temp_Broad
   make/O/N=(NumEvents) temp_asymmetry
   make/O/N=(NumEvents) temp_width
   make/O/N=(NumEvents) temp_risetime30
   make/O/N=(NumEvents) temp_risetime90
   make/O/N=(NumEvents) temp_moment

   make/O/N=(NumEvents) temp_passed=-1
   make/O/N=(NumEvents) temp_failed=-1

   variable j=0, f=0
   for(i=0; i<NumEvents; i+=1) // loop through all events
      variable E = Energies[i]
      variable B = Broad[i]
      variable A = AE[i]
      variable S = asymmetry[i]
      variable w = width[i]
      variable t30 = risetime30[i]
      variable t90 = risetime90[i]
      variable m = moment[i]
      if(passed_filter(E,A,B,S,w, t30, t90, m, i))
         temp_Energies[j] = E
         temp_AE[j] = A
         temp_Broad[j]=B
         temp_asymmetry[j]=S
         temp_width[j]=w
         temp_risetime30[j]=t30
         temp_risetime90[j]=t90
         temp_passed[j]=i
         temp_moment[j]=m
```

```
            j+=1
         else
            temp_failed[f]=i
            f+=1
         endif
      endfor
      if(j>0)
         redimension/N=(j) temp_Energies, temp_AE, temp_Broad,
             temp_asymmetry, temp_width, temp_risetime30,
             temp_risetime90, temp_passed, temp_moment
      endif
      if(f>0)
         redimension/N=(f) temp_failed
      endif

   End

   // Filter rules
   // E=energy in 16bit units, A=current maximum/energy, B=Broad,
       S=asymmetry, W=width, i=event number
   // use the plot function displayVs(root:user:parameterName,
       root:user:energies) to get an idea of the range of values
   // use the plot function displayHistogram(root:user:parameterName,
       x0, x1, Nbins) for a more detailed picture.
   Function passed_filter(E,A,B,S,W, t30, t90, moment i)
      Variable E, A, B, S, W, t30, t90, moment, i
   // Get the filter limits
      NVAR Emin = root:user:Emin
      NVAR Emax = root:user:Emax
      NVAR AEmin = root:user:AEmin
      NVAR AEmax = root:user:AEmax
      NVAR T30min = root:user:T30min
      NVAR T30max = root:user:T30max
      NVAR T90min = root:user:T90min
      NVAR T90max = root:user:T90max
      NVAR momentMin = root:user:momentMin
      NVAR momentMax = root:user:momentMax
      NVAR widthMin = root:user:widthMin
      NVAR widthMax = root:user:widthMax
      NVAR asymmetryMin = root:user:asymmetryMin
      NVAR asymmetryMax = root:user:asymmetryMax
         if(AEmin<=A && A<=AEmax)
            if(asymmetryMin<=S && S<=asymmetryMax)
            if(widthMin<=w && w<=widthMax)
               if(momentMin<=moment && moment<=momentMax)
                  if(Emin<=E && E<=Emax)
                     if(T30min<=t30 && t30<=T30max && T90min<=t90 &&
                        t90<=T90max)
                        return 1
                     endif
                  endif
               endif
            endif
         endif
      endif
   return 0
End

//############################
// Plot functions
//############################

// Scatter plot
Function displayVs(yWave, xWave)
   Wave yWave, xWave
   Display yWave vs xWave
   ModifyGraph mode=3, marker=1, gmSize=1 // scatter display mode
End

// Sorts the values in inputWave into a histogram
// Running this function again overwrites the previous histogram
// Duplicate histogramWave to store the results
// inputWave = wave of pulse shape parameter values
    (root:user:energies, temp_asymmetry, etc)
// Optional parameters:
// x0 and x1 = histogram limits, Nbins = number of bins,
    bottomLabel = name of the parameter
//For example displayHistogram(root:user:energies, x0=0, x1=10000,
    Nbins=10000, bottomlabel="Channel")
Function displayHistogram(inputWave,[x0, x1, Nbins, bottomLabel])
   Wave inputWave
   Variable x0, x1, Nbins
   String bottomLabel

   if(ParamIsDefault(x0) || ParamIsDefault(x1))
      variable xmin=waveMin(inputWave)
      variable xmax=waveMax(inputWave)
      x0=xmin-0.1*abs(xmin)
      x1=xmax+0.1*abs(xmax)
   endif
   if(ParamIsDefault(Nbins))
      Nbins = 100
   endif
   if(ParamIsDefault(bottomLabel))
      bottomLabel = ""
   endif

   Make/O/N=1 histogramWave
   variable dx=(x1-x0)/Nbins
   if(dx>0)
      histogram/B={x0,dx,Nbins} inputWave, histogramWave
      histogramWave=histogramWave/dx
   endif

   //display histogramWave
   //ModifyGraph mode=5, hbFill=2 //histogram display mode
   //Label left "Counts"
   //Label bottom bottomLabel
End

//################################
//Pulse shape parameter calculations
//################################

// Current maximum/ energy
Function get_AE()
   Wave current
   variable currentMaximum = wavemax(current)
   variable E = get_energy()
   if(E>0)
      return currentMaximum/E
   else
      return -1
   endif
End
```

```
 //Asymmetry relative to current pulse maximum
 //Front and back have width 0.25e-6 seconds
Function get_asymmetry()
   Wave current

   variable x_at_max = get_x_at_max(current)

   variable F= get_F(x_at_max)
   variable B= get_B(x_at_max)

   if(F>0 && B>0) // current should be positive, areas should be
       positive
      return (F-B)/(F+B) // result is between -1 and +1
   else
      return -10
   endif
End

Function get_F(x_at_max)
   variable x_at_max
   wave current
   variable x0 = get_x0(x_at_max)
   variable n = numpnts(current)
   variable dt = pnt2x(current, n)/n
   return sum(current, x0, x_at_max)*dt
End

Function get_B(x_at_max)
   variable x_at_max
   wave current
   variable x1 = get_x1(x_at_max)
   variable n = numpnts(current)
   variable dt = pnt2x(current, n)/n
   return sum(current, x_at_max, x1)*dt
End

Function get_x0(x_at_max)
   variable x_at_max
   variable x0 = x_at_max - 0.25e-6
   return max(0, x0)
End
```

```
Function get_x1(x_at_max)
   variable x_at_max
   wave current
   variable x1 = x_at_max + 0.25e-6
   variable x2 = pnt2x(current, numpnts(current)-1)
   return min(x1, x2)
End

Function get_moment()
   wave current
   variable x_at_max = get_x_at_max(current)
   variable index_at_max = get_index_at_max(current)
   variable F = get_F(x_at_max)
   variable B = get_B(x_at_max)
   variable x0 = get_x0(x_at_max)
   variable x1 = get_x1(x_at_max)
   variable p0 = x2pnt(current, x0)
   variable p1 = x2pnt(current, x1)
   variable moment
   variable i
   variable dm
   for(i=p0;i<p1; i+=1)
      dm = current[i]
      if(dm>=0)
         dm = dm*(i-index_at_max)^2
         moment+=dm
      endif
   endfor
   variable dem = (F+B) *(x1-x0)^2
   if(dem>0)
      return moment/dem
   else
      return -1
   endif
End

// Width of the current pulse from 10% to 10% of maximum in seconds
Function get_width()
   wave current
   return get_dt(current, 0.1, 0.1, 0)
End
```

```
// Width from p0 of maximum to p1 of maximum
// e1 defines the edge at p1
// e1 = 1 for rising, 0 for declining
Function get_dt(w, p0, p1, e1)
   wave w
   variable p0, p1, e1
   if(p0>p1 || (e1!=0 && e1!=1))
      print "[ERROR]: Illegal arguments."
      return -1
   endif
   variable max_loc = get_index_at_max(w)
   variable x0 = get_tx(w, p0, 1, max_loc, max_loc, x1=0)
   variable x1 = x0
   if(e1==0)
      x1 = get_tx(w, p1, 0, max_loc, max_loc)
   else
      x1 = get_tx(w, p1, 1, max_loc, x2pnt(w,x0))
   endif
   if((numtype(x0)!=0) || (numtype(x1)!=0))
      return -1
   endif
   return (x1-x0)*1e9
End

// Returns time t where pulse has x% of it's maximum
// tx = the fraction, edge = 1 for rising, 0 for declining,
    max_loc = index of maximum, x0, x1 = search limits
// if x1 is not specified, search is started from x0 and ends at
    the end of the wave
// if x1 is smaller than x0, search is done from right to left
Function get_tx(w, tx, edge, max_loc, x0, [x1])
   wave w
   variable tx, edge  // tx between 0 and 1, edge=1 for rising, 0
       for declining
   variable max_loc, x0, x1  // search limits
   variable wMax = w[max_loc]
   if(ParamIsDefault(x1))
      FindLevel/EDGE=(edge)/Q/R=[(x0)] w, tx*wMax
   else
      FindLevel/EDGE=(edge)/Q/R=[(x0),(x1)] w, tx*wMax
   endif
   return V_LevelX
End
```

```
// Differentiates the charge pulse (trace)
Function get_current_pulse()
   Wave trace = root:pixie4:trace0 //trace1, trace2, trace3 for
       other channels
   Duplicate/O trace, current
   Differentiate current
   Smooth/S=2 5,current
End

// Returns the index of the location where the wave w has it's
    maximum.
Function get_index_at_max(w, [accuracy])
   Wave w
   variable accuracy
   //limit the search +-0.5 microseconds from edges:
   variable maxValue = wavemax(w, 0.5e-6, pnt2x(w,
       numpnts(w))-0.5e-6)
   FindValue/V=(maxValue)/T=(accuracy) w //accuracy 1e-7 by default
   return V_value
End

// Returns the x value of the location where the wave w has it's
    maximum
Function get_x_at_max(w)
   Wave w
   variable indexAtMax=get_index_at_max(w)
   return pnt2x(w, indexAtMax)
End

// Returns the energy of the chosen event
Function get_energy()
   Wave ListModeChannelEnergy = root:pixie4:ListModeChannelEnergy
   return ListModeChannelEnergy[0]
End


//####################################
// Utility functions
//####################################

Function getTrace(eventNumber, [channelNumber])
   Variable eventNumber, channelNumber
   if(ParamIsDefault(channelNumber))
      channelNumber=0
   endif
   changeEvent(eventNumber)
   String wav = "root:pixie4:trace"+num2str(channelNumber)
   Make/O/N=300 outputwave
   Duplicate/O $wav, outputwave
   return outputwave
End

// Test commands
// print root:pixie4:ChosenEvent
// changeEvent(10)
// print root:pixie4:ChosenEvent
Function changeEvent(eventNumber)
   Variable eventNumber
   NVAR ChosenEvent=root:pixie4:ChosenEvent //reference to global
       variable
   ChosenEvent = eventNumber
   Pixie_IO_ReadEvent()
End

Function displayTrace(eventNumber, [channel])
   Variable eventNumber, channel
   if(ParamIsDefault(channel))
      channel=0
   endif
   Make/O/N=300 tempWave
   tempWave = getTrace(eventNumber, channelNumber=channel)
   Display tempWave
End

Function energySpectrum()
   wave energies = root:user:energies
   displayHistogram(energies,x0=0, x1=18000, Nbins=18000,
       bottomLabel="Channel")
End
```

```
//#############################
// Here begins the GUI code
//#############################

   DoWindow/F ListModeTracesDisplay2
   if (V_Flag!=1)

      PauseUpdate; Silent 1    // building window...
      Display/K=1 /W=(250,175,700,500)
          root:pixie4:trace0,root:pixie4:trace1,root:pixie4:trace2,root:pixie4:trace3
          as "List Mode Traces"
      DoWindow/C ListModeTracesDisplay2

      AppendToGraph/L=L1 current

      ModifyGraph freePos(L1)=0
      ModifyGraph axisEnab(left)={0,0.47}
      ModifyGraph axisEnab(L1)={0.53,1}

      ModifyGraph cbRGB=(10000,44032,58880)
      ModifyGraph mode=6
      ModifyGraph grid=1
      ModifyGraph mirror=0
      ModifyGraph
          rgb(trace1)=(0,65280,0),rgb(trace2)=(0,15872,65280),rgb(trace3)=(0,26112,0)
      SetAxis/A/N=1 left
      Label bottom "Time from first sample"
      Label left "Pulse (16bit scale)"
      Label L1 "Current"
      ModifyGraph lblPos(L1)=64
      ModifyGraph lblPos(left)=64
      ControlBar 120 // Control bar size

      SetVariable TraceDataFile, value=root:pixie4:DataFile,
          pos={300,10},size={200,18},title="Data File"
      SetVariable TraceDataFile,
          fsize=10,proc=Pixie_Ctrl_CommonSetVariable//,bodywidth=100

      Button FindTraceDataFile,
          pos={520,8},size={40,20},proc=Pixie_Ctrl_CommonButton,title="Find",fsize=11
      SetVariable
          CallReadEvents,pos={300,30},size={200,18},proc=change_event,title="Event
          Number      "
      SetVariable CallReadEvents,format="%d",fsize=10//,bodywidth=70
      SetVariable CallReadEvents,limits={0,Inf,1},value=
          root:pixie4:ChosenEvent

      ValDisplay Hitpattern, pos = {300,55}, title = "Hit Pattern
          0x", format ="%4.4X", value =
          root:Pixie4:EventHitpattern, size={100,20},fsize=10
      ValDisplay TimeStampHI, pos = {420,55}, title = "Event Time",
          value = root:Pixie4:EventTimeHI, size={100,20},fsize=10
      ValDisplay TimeStampLO, pos = {530,55}, value =
          root:Pixie4:EventTimeLO, size={40,20},fsize=10


      // Initialize Channel Energy List Data
      Pixie_MakeList_Traces(0)


      Button EventFilterDisplay,
          pos={310,80},size={70,25},proc=Pixie_Plot_FilterDisplay,title="Digital
          Filter",fsize=11
      Button HelpList_Mode_Traces,
          pos={405,80},size={70,25},proc=Pixie_CallHelp,title="Help",fsize=11
      Button
          EventDisplayClose,pos={500,80},size={70,25},proc=Pixie_AnyGraphClose,title="C
          Panel"} ,fsize=11
   endif

EndMacro


Function change_event(ctrlName,varNum,varStr,varName):
    SetVariableControl
   String ctrlName, varStr,varName
   Variable varNum
   Pixie_IO_ReadEvent()
End
```

```
Window parameter_viewer() : Graph

   DoWindow/F risetimeWindow
   if (V_Flag!=1)

      PauseUpdate; Silent 1   // building window...
      Display/K=1 /W=(250,175,700,500) temp_risetime30 vs
         temp_risetime90 as "Pulse shape parameter viewer"
      ModifyGraph mode=3, marker=1, gmSize=1
      DoWindow/C risetimeWindow


      ModifyGraph cbRGB=(10000,44032,58880)
      ModifyGraph mirror=0
      ControlBar 157

      TabControl AnalysisTab pos={3,3}, size={365, 150}, fsize=12
      TabControl AnalysisTab tabLabel(0)="Filter"
      TabControl AnalysisTab value=0, labelBack=(51456,44032,58880)
      TabControl AnalysisTab, proc=change_tab,
         tabLabel(1)="Histogram", tabLabel(2)="Analysis",
         tabLabel(3)="Trace"

      TabControl AdvancedTab pos={371, 3}, size={226,150},
         fsize=12, value=0, labelBack=(51456,44032,58880)
      TabControl AdvancedTab tabLabel(0)="Settings", proc=change_tab


      //Filter tab
      SetVariable setMinEnergy,pos={10,25},size={150,18},title="Min
         energy: "
      SetVariable setMinEnergy,format="%d",fsize=10
      SetVariable setMinEnergy,limits={0,Inf,1},value=
         root:user:Emin, proc=changeFilterValues_control
      Checkbox lockMinEnergy, pos={163,26}, size={9,9}, title=""

      SetVariable setMaxEnergy,pos={10,40},size={150,18},title="Max
         energy: "
      SetVariable setMaxEnergy,format="%d",fsize=10
      SetVariable setMaxEnergy,limits={0,Inf,1},value=
         root:user:Emax, proc=changeFilterValues_control
      Checkbox lockMaxEnergy, pos={163,41}, size={9,9}, title=""
```

```
SetVariable setMinAE,pos={10,55},size={150,18},title="Min AE:
     "
SetVariable setMinAE,format="%.3e",fsize=10
SetVariable setMinAE,limits={0,Inf,0},value= root:user:AEmin,
    proc=changeFilterValues_control
Checkbox lockMinAE, pos={163,56}, size={9,9}, title=""

SetVariable setMaxAE,pos={10,70},size={150,18},title="Max AE:
    "
SetVariable setMaxAE,format="%.3e",fsize=10
SetVariable setMaxAE,limits={0,Inf,0},value= root:user:AEmax,
    proc=changeFilterValues_control
Checkbox lockMaxAE, pos={163,71}, size={9,9}, title=""

SetVariable setMinMoment,pos={10,85},size={150,18},title="Min
    moment: "
SetVariable setMinMoment,format="%.3e",fsize=10
SetVariable setMinMoment,limits={-Inf,Inf,0},value=
    root:user:momentMin, proc=changeFilterValues_control
Checkbox lockMinMoment, pos={163,86}, size={9,9}, title=""

SetVariable
    setMaxMoment,pos={10,100},size={150,18},title="Max
    moment:"
SetVariable setMaxMoment,format="%.3e",fsize=10
SetVariable setMaxMoment,limits={-Inf,Inf,0},value=
    root:user:momentMax, proc=changeFilterValues_control
Checkbox lockMaxMoment, pos={163,101}, size={9,9}, title=""

SetVariable setMinWidth,pos={10,115},size={150,18},title="Min
    width: "
SetVariable setMinWidth,format="%.3e",fsize=10
SetVariable setMinWidth,limits={0,Inf,0},value=
    root:user:widthMin, proc=changeFilterValues_control
Checkbox lockMinWidth, pos={163,116}, size={9,9}, title=""

SetVariable setMaxWidth,pos={10,130},size={150,18},title="Max
    width: "
SetVariable setMaxWidth,format="%.3e",fsize=10
SetVariable setMaxWidth,limits={0,Inf,0},value=
    root:user:widthMax, proc=changeFilterValues_control
Checkbox lockMaxWidth, pos={163,131}, size={9,9}, title=""
```

```
SetVariable setMinT30,pos={190,25},size={150,18},title="Min
   T30: "
SetVariable setMinT30,format="%.3e",fsize=10
SetVariable setMinT30,limits={0,Inf,0},value=
   root:user:T30min, proc=changeFilterValues_control
Checkbox lockMinT30, pos={342,26}, size={9,9}, title=""

SetVariable setMaxT30,pos={190,40},size={150,18},title="Max
   T30: "
SetVariable setMaxT30,format="%.3e",fsize=10
SetVariable setMaxT30,limits={0,Inf,0},value=
   root:user:T30max, proc=changeFilterValues_control
Checkbox lockMaxT30, pos={342,41}, size={9,9}, title=""

SetVariable setMinT90,pos={190,55},size={150,18},title="Min
   T90: "
SetVariable setMinT90,format="%.3e",fsize=10
SetVariable setMinT90,limits={0,Inf,0},value=
   root:user:T90min, proc=changeFilterValues_control
Checkbox lockMinT90, pos={342,56}, size={9,9}, title=""

SetVariable setMaxT90,pos={190,70},size={150,18},title="Max
   T90: "
SetVariable setMaxT90,format="%.3e",fsize=10
SetVariable setMaxT90,limits={0,Inf,0},value=
   root:user:T90max, proc=changeFilterValues_control
Checkbox lockMaxT90, pos={342,71}, size={9,9}, title=""

SetVariable
   setMinAsymmetry,pos={190,85},size={150,18},title="Min
   asym: "
SetVariable setMinAsymmetry,format="%.3e",fsize=10
SetVariable setMinAsymmetry,limits={-inf,Inf,0},value=
   root:user:Asymmetrymin, proc=changeFilterValues_control
Checkbox lockMinAsymmetry, pos={342,86}, size={9,9}, title=""

SetVariable
   setMaxAsymmetry,pos={190,100},size={150,18},title="Max
   asym:"
SetVariable setMaxAsymmetry,format="%.3e",fsize=10
SetVariable setMaxAsymmetry,limits={-inf,Inf,0},value=
   root:user:Asymmetrymax, proc=changeFilterValues_control
Checkbox lockMaxAsymmetry, pos={342,101}, size={9,9}, title=""
```

```
//Histogram tab
PopupMenu Select_histogramWave, pos ={10, 25}, title="Input:
   ", value="T30;T90;A/E;Energy"
PopupMenu Select_histogramWave, proc=select_Xwave, mode=4

SetVariable setLowLimit, pos={10, 50}, title="Low: ",
   format="%.3e", fsize=10,
   limits={0,inf,0},value=root:user:lowLimit, size={130,18}
SetVariable setHighLimit, pos={10, 65}, title="High: ",
   format="%.3e", fsize=10,
   limits={0,inf,0},value=root:user:highLimit, size={130,18}
SetVariable setNbins, pos={10, 80}, title="Nbins: ",
   format="%d", fsize=10,
   limits={10,inf,1},value=root:user:histo_bins,
   size={130,18}

Button HistogramButton, pos={100,125}, size={70,25},
   proc=histogram_button, title="Histogram", fsize=11

//Analysis tab
SetVariable setBgLow, pos={10, 25}, title="Bg Low: ",
   format="%d", fsize=10,
   limits={0,inf,1},value=root:user:bgLow, size={130,18}
SetVariable setBgHigh,pos={10, 40}, title="Bg High: ",
   format="%d", fsize=10,
   limits={0,inf,1},value=root:user:bgHigh, size={130,18}
SetVariable setSgLow, pos={10, 55}, title="Sg Low: ",
   format="%d", fsize=10,
   limits={0,inf,1},value=root:user:sgLow, size={130,18}
SetVariable setSgHigh,pos={10, 70}, title="Sg High: ",
   format="%d", fsize=10,
   limits={0,inf,1},value=root:user:sgHigh, size={130,18}

Button CalculateButton, pos={40,100}, size={70,25},
   proc=calculate_button, title="Calculate", fsize=11

//Trace tab
TitleBox eventNumberTitle, title="Event number",
   size={58,25}, pos={30,25}, frame=0, fsize=13
SetVariable setEventNumber, pos={30, 44}, title=" ",
   format="%d", fsize=10, limits={0,inf,1}
```

```
        SetVariable setEventNumber, value=root:pixie4:ChosenEvent,
           size={80,18}, proc=setEventNumber_control
        Button smallerEventNumberButton, pos={10,40}, size={20,22},
           title="<", fsize=15, proc=changeEventNumber_button
        Button largerEventNumberButton, pos={110,40}, size={20,22},
           title=">", fsize=15, proc=changeEventNumber_button
        Checkbox limitToPassedEventsCheckbox, pos={150,30},
           title="Limit to events that passed the filter", fsize=11,
           proc=limitEvent_checkbox
        Checkbox limitToFailedEventsCheckbox, pos={150, 45},
           title="Limit to events that didn't pass the filter",
           fsize=11, proc=limitEvent_checkbox
        Checkbox addConstantFractionLines, pos={150, 70}, title="Show
           constant fraction lines", fsize=11,
           proc=showConstantFraction_checkbox


        //Settings tab
        Button ReadDataButton, pos={380,25}, size={70,25},
           proc=readData_button, title="Read data", fsize=11
        Checkbox autoApplyFilterCheckbox, pos={380,55}, title="Auto
           apply filter changes", fsize=11
        PopupMenu Select_xWave, pos = {380,95}, title="X:",
           value="T30;T90;A/E;Energy;Moment;Width;Asymmetry",
           proc=select_Xwave, mode=2
        PopupMenu Select_yWave, pos = {380,120}, title="Y:",
           value="T30;T90;A/E;Moment;Width;Asymmetry",
           proc=select_Ywave


        Button ApplyFilterButton,
           pos={470,95},size={60,22},proc=apply_filter,title="Apply
           filter",fsize=11
        Button GraphButton,
           pos={470,119},size={60,22},proc=scatterPlot_button,title="Graph",fsize=11
        Button ResetFilterButton, pos={530,95}, size={60,22},
           proc=apply_filter, title="Reset filter", fsize=11

        change_tab("AnalysisTab", 0)
        change_tab("AdvancedTab", 0)
    endif

EndMacro
```

```
//Buttons and controls
Function limitEvent_checkbox(ctrlName, value)
   String ctrlName
   variable value
   if(value==1)
      SetVariable setEventNumber, disable=2
      if(!cmpstr(ctrlName, "limitToPassedEventsCheckbox"))
         Checkbox limitToFailedEventsCheckbox, disable=2
      else
         Checkbox limitToPassedEventsCheckbox, disable=2
      endif
   else
      SetVariable setEventNumber, disable=0
      Checkbox limitToFailedEventsCheckbox, disable=0
      Checkbox limitToPassedEventsCheckbox, disable=0
   endif
   NVAR p_index = root:user:passedList_Index
   NVAR f_index = root:user:failedList_index
   p_index = -1
   f_index = -1
End

Function showConstantFraction_checkbox(ctrlName, value)
   String ctrlName
   variable value
   wave t10_cfLine, t30_cfLine, t90_cfLine
   removeFromGraph/Z t10_cfLine, t30_cfLine, t90_cfLine
   if(value==1)
      wave trace = root:pixie4:trace0
      normalize(trace)
      wave normalizedWave
      variable max_loc = get_index_at_max(normalizedWave)
      variable t10 = get_tx(normalizedWave, 0.1, 1, max_loc,
          max_loc, x1=0)
      variable t30 = get_tx(normalizedWave, 0.3, 1, max_loc,
          max_loc, x1=0)
      variable t90 = get_tx(normalizedWave, 0.9, 1, max_loc,
          max_loc, x1=0)
      variable np = numpnts(trace)
      make/O/N=(np) t10_cfLine=trace(t10)
      make/O/N=(np) t30_cfLine=trace(t30)
      make/O/N=(np) t90_cfLine=trace(t90)
      setScale/P x, 0, pnt2x(trace, np-1)/np, t10_cfLine
      setScale/P x, 0, pnt2x(trace, np-1)/np, t30_cfLine, t90_cfLine
      appendToGraph t10_cfLine, t30_cfLine, t90_cfLine
      modifyGraph rgb(t10_cfLine)=(0,0,65535)
      modifyGraph rgb(t30_cfLine)=(0,65535,0)
      modifyGraph rgb(t90_cfLine)=(40000,40000,20000)
   endif
End

Function
    changeFilterValues_control(ctrlName,varNum,varStr,varName) :
    SetVariableControl
   String ctrlName, varStr, varName
   Variable varNum
   ControlInfo autoApplyfilterCheckbox
   if(V_Value)
      apply_filter("")
   endif
End

Function setEventNumber_control(ctrlName,varNum,varStr,varName) :
    SetVariableControl
   String ctrlName, varStr, varName
   Variable varNum
   Pixie_IO_ReadEvent()
   ControlInfo addConstantFractionLines
   if(V_Value)
      showConstantFraction_checkbox("", 1)
   endif
End

Function changeEventNumber_button(ctrlName): ButtonControl
   String ctrlName
   ControlInfo limitToPassedEventsCheckbox
   variable limitToPassed=V_Value
   ControlInfo limitToFailedEventsCheckbox
   variable limitToFailed=V_Value
   wave temp_passed, temp_failed
   variable newValue, newIndex
   NVAR p_index = root:user:passedList_Index
   NVAR f_index = root:user:failedList_index
   NVAR oldValue = root:pixie4:ChosenEvent
   wave energies = root:user:energies
```

```
if(!cmpstr(ctrlName, "smallerEventNumberButton"))
   if(limitToPassed)
      newIndex=p_index-1
      if(0<=newIndex && newIndex<numpnts(temp_passed))
         newValue=temp_passed[newIndex]
         p_index=newIndex
      else
         if(numpnts(temp_passed)>0)
            p_index=0
            newValue=temp_passed[p_index]
         endif
      endif
   elseif(limitToFailed)
      newIndex=f_index-1
      if(0<=newIndex && newIndex<numpnts(temp_failed))
         newValue=temp_failed[newIndex]
         f_index=newIndex
      else
         if(numpnts(temp_failed)>0)
            f_index=0
            newValue=temp_failed[f_index]
         endif
      endif
   else
      newValue = oldValue-1
   endif
elseif(!cmpstr(ctrlName, "largerEventNumberButton"))
   if(limitToPassed)
      newIndex=p_index+1
      if(0<=newIndex && newIndex<numpnts(temp_passed))
         newValue=temp_passed[newIndex]
         p_index=newIndex
      endif
   elseif(limitToFailed)
      newIndex=f_index+1
      if(0<=newIndex && newIndex<numpnts(temp_failed))
         newValue=temp_failed[newIndex]
         f_index=newIndex
      endif
   else
      newValue = oldValue+1
   endif
endif
```

```
  if(0<=newValue && newValue<numpnts(energies))
    oldValue = newValue
    setEventNumber_control("",0,"","")
  endif
End

Function select_Xwave(ctrlName, value, valueStr)
  String ctrlName, valueStr
  variable value
  get_xWave(value)

  if(!cmpstr(ctrlName,"Select_histogramWave"))
    wave temp_risetime30, xWave
    NVAR low = root:user:lowLimit
    NVAR high = root:user:highLimit
    variable xmin=waveMin(xWave)
    variable xmax=waveMax(xWave)
    low=xmin-0.1*abs(xmin)
    high=xmax+0.1*abs(xmax)
  endif
End

Function get_xWave(value)
  variable value
  wave temp_risetime30, temp_risetime90, temp_AE, temp_energies,
     temp_moment, temp_width, temp_asymmetry
  make/O/N=(numpnts(temp_risetime30)) xWave
  switch(value)
    case 1:
      xWave = temp_risetime30
      break
    case 2:
      xWave = temp_risetime90
      break
    case 3:
      xWave = temp_AE
      break
    case 4:
      xWave = temp_energies
      break
    case 5:
      xWave = temp_moment
      break
      case 6:
         xWave = temp_width
         break
      case 7:
         xWave = temp_asymmetry
         break
   endswitch

End

Function select_Ywave(name, value, valueStr)
   String name, valueStr
   variable value
   wave temp_risetime30, temp_risetime90, temp_AE, temp_moment,
       temp_width, temp_asymmetry
   make/O/N=(numpnts(temp_risetime30)) yWave
   switch(value)
      case 1:
         yWave = temp_risetime30
         break
      case 2:
         yWave = temp_risetime90
         break
      case 3:
         yWave = temp_AE
         break
      case 4:
         yWave = temp_moment
         break
      case 5:
         yWave = temp_width
         break
      case 6:
         yWave = temp_asymmetry
         break
   endswitch
End

Function apply_filter(ctrlName): ButtonControl
   String ctrlName
   if(!cmpstr(ctrlName, "ResetFilterButton"))
      reset_filter(checkLocks=1)
   endif
   filter_events()
   scatterPlot_button("applyFilter")
End

Function calculate_button(ctrlName): ButtonControl
   String ctrlName
   NVAR bgLow = root:user:bgLow
   NVAR bgHigh= root:user:bgHigh
   NVAR sgLow = root:user:sgLow
   NVAR sgHigh= root:user:sgHigh
   histogram_button("calculate")
   wave histogramWave, originalEnergies
   if(waveExists(histogramWave) && waveExists(originalEnergies))
      variable oBg = sum(originalEnergies, bgLow, bgHigh)
      variable oSg = sum(originalEnergies, sgLow, sgHigh)
      variable fBg = sum(histogramWave, bgLow, bgHigh)
      variable fSg = sum(histogramWave, sgLow, sgHigh)
      variable pBg = (oBg-fBg)*100/oBg
      variable pSg = (oSg-fSg)*100/oSg
      Print "Background counts reduced by "+num2str(pBg)+"%, signal
          counts reduced by "+num2str(pSg)+"%."
   endif
End

Function readData_Button(ctrlName): ButtonControl
   String ctrlName
   read_data()
End

Function setCollection_Button(ctrlName): ButtonControl
   String ctrlName
   Wave AE    = root:user:AE
   Wave Energies = root:user:Energies
   Wave Broad    = root:user:Broad
   Wave asymmetry = root:user:asymmetry
   wave width    = root:user:width
   wave risetime30   = root:user:risetime30
   wave risetime90   = root:user:risetime90
   wave moment   = root:user:moment
   wave temp_AE, temp_energies, temp_broad, temp_asymmetry,
       temp_width, temp_risetime30, temp_risetime90, temp_moment
   AE = temp_AE
   Energies = temp_energies
   Broad = temp_broad
   asymmetry = temp_asymmetry
   width = temp_width
   risetime30 = temp_risetime30
   risetime90 = temp_risetime90
   moment = temp_moment
End

Function scatterPlot_button(ctrlName): ButtonControl
   String ctrlName
   wave xWave, yWave
   ControlInfo select_xWave
   select_Xwave("select_xWave", V_Value, S_value)
   ControlInfo select_yWave
   select_Ywave("select_yWave", V_Value, S_value)
   if(waveExists(xWave) && waveExists(yWave))
      RemoveFromGraph/Z $"#0"
      RemoveFromGraph/Z $"#0"
      AppendToGraph yWave vs xWave
      ModifyGraph mode=3, marker=1, gmSize=1
   endif
End

Function histogram_button(ctrlName): ButtonControl
   String ctrlName
   wave xWave
   ControlInfo select_histogramWave
   get_Xwave(V_Value)
   if(waveExists(xWave))
      NVAR low=root:user:lowLimit
      NVAR high=root:user:highLimit
      NVAR histo_bins=root:user:histo_bins
      wave histogramWave
      wave oEnergies = root:user:energies
      displayHistogram(oEnergies, x0=low, x1=high, Nbins=histo_bins)
      duplicate/O histogramWave, originalEnergies
      displayHistogram(xWave, x0=low, x1=high, Nbins=histo_bins)

      RemoveFromGraph/Z $"#0"
      RemoveFromGraph/Z $"#0"
      AppendToGraph histogramWave
      ModifyGraph mode=5, hbFill=2
   endif
```

```
End

Function tracePlot()
   wave current
   RemoveFromGraph $"#0"
   AppendToGraph root:pixie4:trace0
   AppendToGraph/L=L1 current

   ModifyGraph freePos(L1)=0
   ModifyGraph axisEnab(left)={0,0.47}
   ModifyGraph axisEnab(L1)={0.53,1}
   ModifyGraph lblPos(L1)=64
End

Function change_tab(name, tab)
   String name
   Variable tab
   if(!cmpstr(name, "AnalysisTab"))
      //Filter tab
      SetVariable setMinEnergy, disable= (tab!=0)
      SetVariable setMaxEnergy, disable= (tab!=0)
      SetVariable setMinAE, disable= (tab!=0)
      SetVariable setMaxAE, disable= (tab!=0)
      SetVariable setMinT30, disable= (tab!=0)
      SetVariable setMaxT30, disable= (tab!=0)
      SetVariable setMinT90, disable= (tab!=0)
      SetVariable setMaxT90, disable= (tab!=0)
      SetVariable setMinMoment, disable= (tab!=0)
      SetVariable setMaxMoment, disable= (tab!=0)
      SetVariable setMinWidth, disable= (tab!=0)
      SetVariable setMaxWidth, disable= (tab!=0)
      SetVariable setMinAsymmetry, disable= (tab!=0)
      SetVariable setMaxAsymmetry, disable= (tab!=0)
      Checkbox lockMinEnergy, disable = (tab!=0)
      Checkbox lockMaxEnergy, disable = (tab!=0)
      Checkbox lockMinAE, disable = (tab!=0)
      Checkbox lockMaxAE, disable = (tab!=0)
      Checkbox lockMinT30, disable = (tab!=0)
      Checkbox lockMaxT30, disable = (tab!=0)
      Checkbox lockMinT90, disable = (tab!=0)
      Checkbox lockMaxT90, disable = (tab!=0)
      Checkbox lockMinMoment, disable= (tab!=0)
      Checkbox lockMaxMoment, disable= (tab!=0)
```

```
        Checkbox lockMinWidth, disable= (tab!=0)
        Checkbox lockMaxWidth, disable= (tab!=0)
        Checkbox lockMinAsymmetry, disable= (tab!=0)
        Checkbox lockMaxAsymmetry, disable= (tab!=0)
        //Histogram tab
        PopupMenu Select_histogramWave, disable = (tab!=1)
        PopupMenu Select_histogramWave, disable = (tab!=1)
        SetVariable setLowLimit, disable = (tab!=1)
        SetVariable setHighLimit, disable = (tab!=1)
        SetVariable setNbins, disable = (tab!=1)
        Button HistogramButton, disable = (tab!=1)
        //Analysis tab
        SetVariable setBgLow, disable = (tab!=2)
        SetVariable setBgHigh, disable = (tab!=2)
        SetVariable setSgLow, disable = (tab!=2)
        SetVariable setSgHigh, disable = (tab!=2)
        Button CalculateButton, disable = (tab!=2)
        //Trace tab
        TitleBox eventNumberTitle, disable = (tab!=3)
        SetVariable setEventNumber, disable = (tab!=3)
        Button smallerEventNumberButton, disable = (tab!=3)
        Button largerEventNumberButton, disable = (tab!=3)
        Checkbox limitToPassedEventsCheckbox, disable = (tab!=3),
            value=0
        Checkbox limitToFailedEventsCheckbox, disable = (tab!=3),
            value=0
        Checkbox addConstantFractionLines, disable = (tab!=3), value=0
        if(tab==3)
            tracePlot()
        else
            showConstantFraction_checkbox("", 0)
        endif
    else
        //Settings tab
        Button ReadDataButton, disable = (tab!=0)
        Checkbox autoApplyFilterCheckbox, disable = (tab!=0)
        PopupMenu Select_xWave, disable = (tab!=0)
        PopupMenu Select_yWave, disable = (tab!=0)
        Button ApplyFilterButton, disable= (tab!=0)
        Button GraphButton, disable = (tab!=0)
        Button ResetFilterButton, disable = (tab!=0)
    endif
End
```