Science and Technology Facilities Council

Technical Report
STFC-TR-2026-017

Event identification for digitised traces from muon decay detectors

B Shustin, J Fowkes, R Stewart, D Kirk, D Pooley, A Lim, W Mattar, R Waite, PJ Baker, N Gould

August 2026

Enquiries concerning this report should be addressed to:

RAL Library
STFC Rutherford Appleton Laboratory
Harwell Oxford
Didcot
OX11 0QX

Tel: +44(0)1235 445577
email: library@stfc.ac.uk

Science and Technology Facilities Council reports are available online at:
https://epubs.stfc.ac.uk

Accessibility: a Microsoft Word version of this document (for use with assistive technology) may be available on request.

DOI: 10.5286/stfctr.2026017

ISSN 2753-5797

Event Identification for Digitised Traces from Muon Decay Detectors

Boris Shustin[1], Jaroslav Fowkes[1], Rhea Stewart[2], Daniel Kirk[2], Daniel Pooley[2], Anthony Lim[2], Wael Mattar[3], Richard Waite[2], Peter J. Baker[2], and Nick Gould[1]

[1]STFC Rutherford Appleton Laboratory, Scientific Computing Department, Continuous Optimization Group
[2]STFC Rutherford Appleton Laboratory, ISIS Neutron and Muon Source
[3]Tel Aviv University, Israel

August 6, 2026

Abstract

In this work, we address the problem of identifying muon decay events in detector output time-series data. Several event detection approaches are reviewed, including direct thresholding, filtering, and hybrid methods. Building on these, we propose and evaluate algorithms based on fixed thresholds, derivative-based discrimination, and multiscale preprocessing using a pyramid approximation.

Performance is assessed on both simulated and real datasets, focusing on detection accuracy, temporal resolution, amplitude sensitivity for the simulated data (where ground truth is known), and dead time analysis for the real datasets. The results demonstrate that derivative-based and multiscale methods improve robustness and event separation compared to standard thresholding, particularly in noisy conditions. Comparison of the fitted muon lifetimes indicates that derivative-based methods could enable approximately two- to four-fold higher usable count rates, potentially reducing data acquisition times by a similar factor for count-limited μSR experiments.

Contents

1 Introduction

Muon spin rotation/relaxation/resonance (μSR) is a technique used to probe the magnetic properties of materials by implanting spin-polarised positive or negative muons into the sample of interest [1]. The muon spins experience the magnetic fields where they stop within the sample and, when they decay, the emitted positron or electron provides information about the muon spin direction at the moment of decay. By collecting this information for millions of muon decays the behaviour of the ensemble of muon spins implanted in the sample is recorded. This method probes a wide variety of solid, liquid, and gaseous materials, providing information on diverse topics such as the flux line lattice of superconductors, the motion of ions within battery materials, and the reaction kinetics of the light hydrogen analogue muonium.

μSR experiments can be performed at two types of muon source: continuous sources where one muon at a time is implanted into the sample, and pulsed sources where time-separated bunches of $> 10^4$ muons arrive in the sample over a period far shorter than the muon lifetime of 2.2 μs [1]. For pulsed muon sources such as ISIS, the large number of muon decays occurring in each pulse requires instruments with many detectors, providing an engineering challenge in construction that is mitigated by ensuring each detector can record several muon decays after each muon pulse. The traditional design of muon instrument detectors has been a plastic scintillator coupled to a photomultiplier tube, with the signal processed by either a fixed threshold or constant fraction discriminator before being recorded in histograms of counts vs time after muon implantation. Currently, the counting rate of pulsed muon instruments is limited by their number of detectors (~ 100) and the recovery time of the discrimination electronics (~ 10 ns).

The Super-MuSR instrument [2] under construction at ISIS will have 960 detectors, upgrading the existing MuSR instrument with only 64, and this significant increase has motivated a novel detector system. Plastic scintillators are coupled to silicon photomultipliers with their signals read out by high-speed digitisers that need signal processing algorithms to find the events due to positron or electron detections [3]. Following the successful development of this digitising system for Super-MuSR, it is expected to be applied to other muon instruments with fewer detectors, which have higher count rates per detector and will therefore benefit more from better signal processing.

The aim of this project was to develop a computationally effective method for finding the time and amplitude of peaks in the digitised traces coming from muon instrument detectors. The key performance indicators were the time resolution with which the peaks are determined, the pulse-pair resolution that allows closely spaced peaks to be separately detected, and the resolution in the peak amplitude that allows noise peaks to be separated from signal peaks. In order to facilitate the aforementioned goals, we investigated a range of signal-processing methods for event detection. These include several classes of approaches, direct methods, data filtering methods, and combined filtering–detection methods. In particular, four computationally efficient algorithms were selected for detailed study: fixed threshold discrimination, as well as more advanced techniques based on first- and second-derivative thresholding, and multiscale preprocessing using a pyramid approximation.

The structure of this report is as follows: Section 2 introduces the problem of

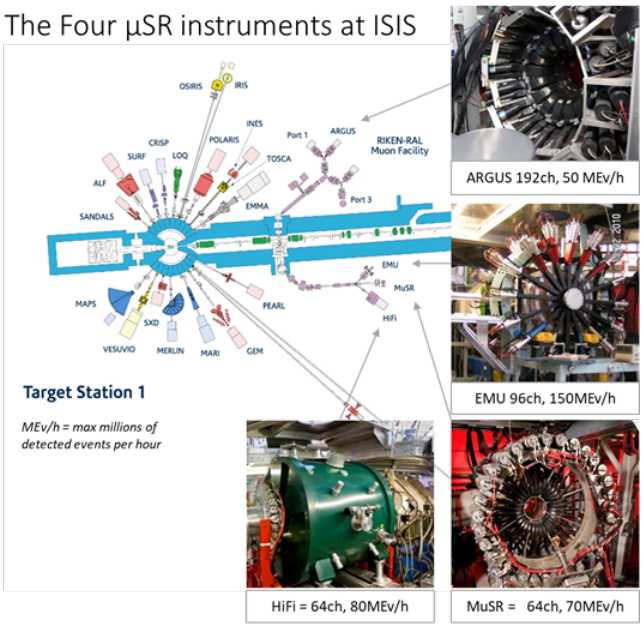


Figure 1: The ISIS μSR instruments.

muon event detection and its experimental context; Section 3 provides an overview of existing event detection algorithms from the literature, and a new proposed multiscale approach; Section 4 presents the four selected methods in depth and their implementations; Sections 5 and 6 evaluate their performance on simulated and real datasets, respectively; and in Section 7 we conclude the report with the key findings and outline directions for future work.

2 Muon Event Detection

Pulsed muon spectroscopy techniques require detector systems to count decay positrons at high instantaneous counting rate. At ISIS, each muon pulse can implant $O(10^4)$ muons into a sample in ~ 70 ns, after which they decay with a lifetime of 2.2 μs. To reduce the effects of pile-up (failing to detect events that are temporally too close) the detector arrays are pixelated into independent detectors with independent electronic readout. Pixelation alone is insufficient, so the beam flux is reduced via an adjustable aperture (slits). The ISIS instruments, with their numbers of channels and event rates are shown above in Fig. 1.

The detector systems use scintillators (a material that emits light), a light sensor, and readout electronics to count incident positrons. This is illustrated in Fig. 2.

The count-rate bottleneck in this system is partially attributed to the signal processing, which is an analogue implementation of a 'leading edge discriminator', the CAEN V895. Leading edge discrimination ('fixed-threshold discrimination') is not well suited to high rate applications, as the inherent dead time is at least as long as the time taken for the signal to fall below the discrimination level and the system to re-arm. This project seeks to improve count rate capability by developing bespoke digital signal processing algorithms, a new opportunity brought about by the recent development of Nuclear Instruments DAQ-121. The data pipeline concept and DAQ121 are described in [3].

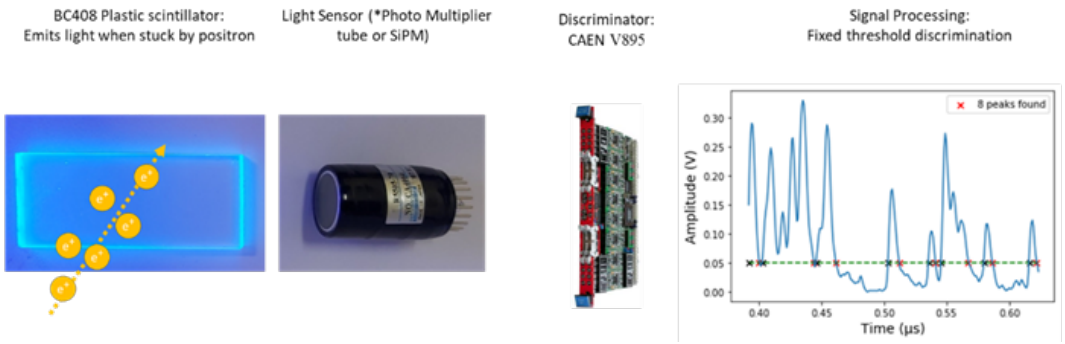


Figure 2: Key components of a muon spectrometer detector system.

DAQ121 is a specialised 'digitisation' technology, developed for muon instruments that will stream raw traces for further processing using a Kafka streaming data pipeline, where the algorithm developed can be embedded within the event-formation-unit. Data from this system connected to the Super-MuSR prototype detector are discussed in section 6.5.

The other data analysed in this project were collected with a 6000 series Picoscope digitiser (sampling at 1.25 GS/s), saving detector signals to disk. The data is compiled into an archive of characteristic signals from the ISIS detector suite. Data were also taken under various experimental conditions, which can be analysed to better understand the performance of the signal processing being developed. One example is the Radio-Frequency (RF) data set collected on the EMU instrument, which gives a characteristic frequency in the time domain histogram.

Further information on the instruments and upgrades are well described in [2].

3 Overview of Event Detection Algorithms

Event (peak) detection in particle detector time-series, such as the traces coming from μSR instruments, is a well-studied problem in signal processing. The challenge typically involves detecting transient pulses under noise, overlapping events, and baseline fluctuations. Existing approaches can broadly be categorised into direct methods, data filtering methods, and combined filtering–detection methods. The methods that were reviewed and tested in this work are listed in Table 1. In this section, we provide a short literature review for these methods.

Direct	Filter	Filter and Find
Fixed Thresholding	Deconvolution	Smoothed 2nd Derivative
1st Derivative Thresholding	Wiener Deconvolution	Wavelet-based
Peak Finding	Savitzky-Golay Filter	
Function Fitting	Pyramid Approximation	

Table 1: Classification of event detection algorithms into three classes.

3.1 Direct Approaches

Direct methods operate on the raw (or minimally processed) signal and identify peaks based on simple criteria such as amplitude or local structure. In this work, we focused on fixed thresholding, peak finding, and function fitting. In the following, we provide a short review of the aforementioned methods.

- **Fixed threshold discrimination - the baseline method** - Fixed threshold discrimination is one of the earliest developed and most widely used techniques in nuclear and particle instrumentation [4]. A signal is classified as an event when its amplitude exceeds a predefined amplitude threshold. This approach can be extended to derivative-based thresholding, e.g., first derivative thresholding enhances sensitivity to rapid signal changes (edges of pulses), second derivative thresholding highlights curvature, which helps in identifying local maxima, allowing for identifying peaks more robustly. These types of methods are generally computationally simple and fast; however, they are prone to noise and baseline drift.

- **Peak finding - local maxima detection** - Methods based on identifying local maxima use finite differences to approximate the first derivative from the time-series, see for example *scipy.signal.find_peaks* from SciPy [5], and *indexes* from Peakutils [6], which were utilised in this work. This allows us to find local maxima at points i, such that

$$y_{i-1} < y_i \ \wedge \ y_i > y_{i+1}$$

  and also account for the more general cases, such as plateaus. These methods are widely used due to their balance between simplicity and flexibility.

- **Function Fitting** - Fitting signals to known pulse shapes [7] is common practice where detector response functions are well characterised. It converts the problem of detection into that of parameter estimation. As part of this project, various pulse shapes were tested on a number of muon detector traces. It was found that the asymmetric peaks representing the fast rise and slow decay times found in muon traces are best modeled using Back-To-Back Exponential functions, see *BackToBackExponential* in MANTID [8]. The function fitting method allows greater accuracy and robustness, but it is computationally expensive due to the optimisation involved when finding the best fit to the data. This method was the first to be considered in this work; however, it was decided to proceed with other methods due to the high computational cost and long processing time, which fall well short of the throughput required for real-time processing. Specifically, in Table 2, function fitting runtime was 4980 seconds in our preliminary test. It was estimated that only methods that reached a runtime of less than 100 seconds could potentially satisfy the time requirement for production hardware.

3.2 Data Filtering Approaches

Filtering methods aim to enhance signal-to-noise ratio (SNR) before or during detection. This approach allows us to then apply peak-finding methods after the filtering,

resulting in improved detection as peak-finding methods are sensitive to noise. In this work, the methods we focused on are deconvolution, Wiener deconvolution, the Savitzky-Golay filter, and a multiscale subdivision method for denoising (the Pyramid Approximation method [9]). In the following, we provide a short review of the aforementioned methods.

- **Deconvolution and Wiener Deconvolution** - If the detector response (pulse shape) is known, as in this case with the Back-To-Back Exponential, deconvolution can help recover underlying events. More specifically, deconvolution replaces the pulse shape with a delta function at the moment of detection, allowing the peaks to be easily detected. However, deconvolution is highly sensitive to noise and model mismatch. To account for the high noise sensitivity, Wiener Deconvolution is a regularised deconvolution (frequency-domain) that utilises Wiener filtering to minimise mean-square error under noise assumptions [10]. In this work, we tested the implementations of these methods in SciPy [5] in *scipy.signal.deconvolve* and *scipy.signal.wiener* respectively.

- **Savitzky-Golay Filter** - The Savitzky-Golay filter performs a local polynomial regression within a sliding window to smooth the data [11]. Unlike simple moving average filters, it preserves important signal features such as peak height, width, and relative position, making it particularly suitable for peak detection tasks. In addition to smoothing, the method can be used to compute stable numerical derivatives by differentiating the fitted polynomial, which enables more robust identification of local maxima and minima. In this work, it is used as a preprocessing step to improve the SNR, and the implementation that we consider is *scipy.signal.savgol_filter* from SciPy [5].

- **Pyramid Approximation for denoising** - A pyramid transform is a multiscale tool that decomposes a signal or image into a hierarchical structure consisting of a coarse, low-resolution approximation and multiple layers of detail coefficients representing information at various scales [9, 12, 13]. In the context of signal denoising, the transform is particularly effective because it can distinguish between actual signal features and random noise by analysing the decay of the detail coefficients; the process works by concentrating the essential signal information into a few large coefficients while spreading noise across many small ones, allowing for the stable reconstruction of a "cleaner" version of the original signal (see for example [9]). To the best of our knowledge, this method was not previously utilised as a preprocessing tool for peak detection.

3.3 Filter-and-Detect Approaches

These methods combine signal transformation with explicit detection. In this work, we focused on a method for smoothing the second derivative (Gaussian–Laplace filtering) and then thresholding, and a wavelet-based method. In the following, we provide a short review of the aforementioned methods.

- **Smoothed 2nd Derivative** - The Gaussian-Laplace filter, more commonly known as the Laplacian of Gaussian (LoG) filter, is a second-order differential operator used primarily for edge detection and blob detection in image

processing. By combining a Gaussian smoothing function with the Laplacian operator, the filter reduces sensitivity to high-frequency noise while highlighting regions of rapid intensity change [14]. This allows the use of thresholding of the smoothed 2nd derivative for the detection of local maxima (peaks). This method was developed by Richard Waite, and is inspired by MANTID's *FindPeaks*[1] [8], where a different (averaging) smoothing [15] for the second derivative is performed before peak finding.

- **Wavelet-Based Detection** - Various multiscale methods based on wavelets for peak detection appear in the literature, see for example [16–18]. In this work, we rely on the implementation in *scipy.signal.find_peaks_cwt* from SciPy [5]. This function implements peak detection using the Continuous Wavelet Transform (CWT), where the signal is convolved with wavelets at different scales. Peaks are identified as ridges, i.e. sequences of local maxima that persist across multiple scales. This multi-scale approach provides robustness to noise and allows reliable detection of peaks with varying widths. Consequently, wavelet-based methods are especially effective for analysing non-stationary signals such as detector traces.

4 Proposed Event Detection Algorithms

In this section, we focus on four methods that were selected from the methods listed in Table 1 for implementation (in Python and Rust) and analysis as part of this work (see Section 5). The implementations of the algorithms in Python can be found in the GitHub repository [19]. First, we would like to address the initial selection criteria for the methods. There were two goals in selecting the methods: the primary goal was that each selected method should satisfy the **time requirement** of this work (see the next paragraph); the secondary goal was that we aimed to have at least one method from each of the categories described in Section 3.

The **time requirement** is driven by the need to avoid any backlog during live data acquisition. The processing methods must therefore sustain a throughput of approximately 10^6 events per second on suitable hardware. The incoming data are organised as "traces", each consisting of 32×10^3 12-bit (2-byte) integer samples. The Super-MuSR system comprises 960 independent channels, each producing one $\sim$ 30 μs trace per ISIS frame, with frames arriving every 20 ms. Overall, this corresponds to roughly 40 ISIS frames per second, from which we expect to identify on the order of 10^6 valid events per second, while rejecting a comparable number as noise.

To determine which methods could potentially satisfy this requirement, the methods from Table 1 were tested on a short trace dataset[2] from EMU that contains 4 channels, 401 frames, and $30,000$ time samples per frame. Thus, on production hardware, it should be analysed in approximately 0.04 seconds. However, as an initial means of benchmarking their performance, the peak finding methods were applied to the test data set and run on a standard laptop. The subset of methods that took less than 100 seconds to run in this test were selected for more detailed investigation. A summary of the results can be seen in Table 2, where the runtime on

[1] https://docs.mantidproject.org/nightly/algorithms/FindPeaks-v1.html
[2] EMU_A14_B13_C61_D62_June2021_Q_RF_LF118_Slit12_short

the Short EMU Trace dataset is presented, the methods that satisfy the sub-100 second requirement are Fixed Thresholding, 1st Derivative Thresholding, Peak Finding (SciPy), Smoothed 2nd Derivative, Savitzky-Golay Filter (SciPy), and Pyramid Approximation Preprocessing. Since both Peak Finding (SciPy) and Savitzky-Golay Filter (SciPy) are implemented in SciPy, we choose to focus on implementing and analysing the performance of the other 4 methods.

Event Finding Algorithm	Runtime (sec)
Fixed Thresholding	**5.11**
1st Derivative Thresholding	**5.56**
Peak Finding (peakutils)	141
Peak Finding (SciPy)	1.41
Smoothed 2nd Derivative	**3.13**
Deconvolution (SciPy)	5480
Wiener Deconvolution (SciPy)	6276
Wavelet-based (SciPy)	746
Savitzky-Golay Filter (SciPy)	1.70
Pyramid Approximation Preprocessing	**33.0**
Function Fitting	4980

Table 2: Runtime (in seconds) for the event finding algorithms on the Short EMU Trace dataset. The methods selected for further investigation are in bold.

In the following, we provide additional details for each of the algorithms from Table 2 that were chosen for further study.

4.1 Thresholding Methods

In this subsection, we provide implementation details for a generic thresholding method, and in the subsequent subsections (4.1.1, 4.1.2, and 4.1.3) we detail the specifics for each method.

A generic thresholding method identifies contiguous regions in which a selected feature (signal amplitude, first derivative, or smoothed second derivative) exceeds a predefined threshold for at least a specified number of consecutive samples (the *trigger duration*). This suppresses noise fluctuations that may cause the selected feature to briefly cross the threshold. The algorithm processes each frame sequentially, sample by sample. When the selected feature crosses the threshold, a candidate region is initiated, and the corresponding sample index (or time) is recorded as the region starting point. Once the selected feature drops below the threshold (or any other chosen value), the duration of the region is evaluated. If it exceeds the trigger duration, the region is accepted and its start time is stored. Otherwise, it is discarded as noise.

The choice of thresholding feature affects the characteristics of the detected peaks, with each feature presenting different trade-offs in terms of detection accuracy, robustness, and computational efficiency. The relative performance of each of the methods based on a different thresholding feature is evaluated in the following sections using both simulated and experimental detector traces.

4.1.1 Fixed Threshold Discrimination - The Baseline Method

In this subsection, we provide implementation details for the Fixed Threshold Discrimination algorithm. A pseudocode is provided in Alg. 1.

Fixed Threshold Discrimination applies the generic thresholding procedure described in Subsection 4.1 using the signal amplitude as the thresholding feature. Thus, candidate regions are identified where the signal amplitude exceeds a predefined threshold for at least the trigger duration. For each accepted region, the maximum signal amplitude is recorded as the event amplitude (pulse height).

To avoid multiple detections of a single physical pulse due to small fluctuations around the threshold, a *cool-off* condition is applied. A new event can only be triggered if a sufficient number of samples have passed since the signal was last below the threshold. Finally, if the trace ends while still above the threshold, the same validation is applied to determine whether the final segment constitutes a valid event.

Note that, arbitrarily, the event time is defined as the first threshold crossing. Thus, for every detected event, the recorded event time cannot exceed the true event time and, except for the unlikely case where the threshold coincides exactly with the peak sample, will underestimate it.

Amplitude thresholding is simple, computationally efficient, and requires minimal signal processing, making it well suited to high-throughput applications. However, because detection relies solely on the signal amplitude, closely spaced events are often merged, reducing its ability to resolve overlapping pulses. Furthermore, the detected event amplitude systematically underestimates the true event time, as described previously. Consequently, while amplitude thresholding provides excellent computational performance, its detection accuracy deteriorates at high event rates where pulse pile-up becomes significant.

Algorithm 1 Fixed Threshold Discrimination for Event Detection

Require: Time samples t, trace amplitudes A, threshold θ, trigger duration d, cool-off c
Ensure: Event times T_e and event amplitudes A_e

Initialize $T_e \leftarrow [\,]$, $A_e \leftarrow [\,]$
$inEvent \leftarrow$ False
$maxAmp \leftarrow 0$
$startIdx \leftarrow$ None
$lastBelow \leftarrow$ None
for each sample index i in trace **do**
    if $A[i] > \theta$ **then**
        if not $inEvent$ **and** ($lastBelow$ is None **or** $i - lastBelow \geq c$) **then**
            $inEvent \leftarrow$ True
            $startIdx \leftarrow i$ ▷ Event start (threshold crossing)
            $maxAmp \leftarrow A[i]$
        else
            $maxAmp \leftarrow \max(maxAmp, A[i])$
        end if
    else
        if $inEvent$ **then**
            if $i - startIdx \geq d$ **then**
                Append $t[startIdx]$ to T_e
                Append $maxAmp$ to A_e
            end if
            $inEvent \leftarrow$ False
            $maxAmp \leftarrow 0$
            $startIdx \leftarrow$ None
            $lastBelow \leftarrow i$
        end if
    end if
end for
if $inEvent$ **and** $startIdx \neq$ None **and** $N - startIdx \geq d$ **then**
    Append $t[startIdx]$ to T_e
    Append $maxAmp$ to A_e
end if
return T_e, A_e

4.1.2 1st Derivative Fixed Threshold Discrimination

In this subsection, we provide implementation details for the 1st Derivative Fixed Threshold Discrimination algorithm, which is similar in spirit to Fixed Threshold Discrimination from the previous subsection. A pseudocode is provided in Alg. 2.

1st Derivative Fixed Threshold Discrimination applies the generic thresholding procedure described in Subsection 4.1 using the first derivative of the signal as the thresholding feature. The derivative is computed using finite differences on consecutive samples, enhancing sensitivity to the rapidly rising edges typical of detector pulses. This approach improves robustness to slow baseline variations and low-frequency noise. The event amplitude is taken as the maximum value of the

original signal within each accepted region, while the event time is defined by the first derivative threshold crossing. The event is considered to end when the derivative becomes non-positive, corresponding to the transition from the rising edge to the peak or falling edge of the pulse.

As in the standard fixed-threshold method, a *cool-off* condition is applied. A new event can only be triggered if a sufficient number of samples have passed since the derivative last returned to zero or below, preventing multiple detections of the same physical event. The event time is defined as the sample at which the derivative first crosses the threshold (after satisfying the trigger duration), which as before, may introduce a systematic offset relative to the true event maximum, and the event amplitude is the maximum signal value observed during the event. However, note that the derivative emphasises the leading edge of detector pulses, improving timing sensitivity compared to amplitude thresholding.

1st Derivative thresholding detects events based on the rate of change of the signal rather than its amplitude. This enables earlier detection of rising pulse edges and improves the separation of closely spaced events, reducing the effects of pulse pile-up. The method is, however, more sensitive to high-frequency noise because differentiation amplifies noise fluctuations, often requiring additional smoothing or careful selection of the trigger duration. It is also computationally more demanding than amplitude thresholding, although it remains suitable for real-time processing.

Algorithm 2 1st Derivative Fixed Threshold Discrimination for Event Detection

Require: Time samples t, trace amplitudes A, derivative threshold θ, trigger duration d, cool-off c
Ensure: Event times T_e and event amplitudes A_e

Compute discrete derivative: $\Delta A[i] = A[i] - A[i-1]$
Initialize $T_e \leftarrow [\,]$, $A_e \leftarrow [\,]$
$timeCrossed \leftarrow$ None
$tempTime \leftarrow$ None
$lastReturn \leftarrow$ None
$maxAmp \leftarrow 0$
for each sample index $i = 1$ to $N - 1$ **do**
    if $timeCrossed$ is None **then**
        if $\Delta A[i] > \theta$ **then**
            if $lastReturn$ is None **or** $i - lastReturn \geq c$ **then**
                $timeCrossed \leftarrow i$
                $maxAmp \leftarrow A[i]$
            end if
        end if
    else
        $maxAmp \leftarrow \max(maxAmp, A[i])$
        if $i - timeCrossed = d$ **then**
            $tempTime \leftarrow timeCrossed$ ▷ Confirm event start
        end if
        if $\Delta A[i] \leq 0$ **then**
            $lastReturn \leftarrow i$
            if $tempTime \neq$ None **then**
                Append $t[tempTime]$ to T_e
                Append $maxAmp$ to A_e
            end if
            Reset $timeCrossed$, $tempTime$, $maxAmp$
        end if
    end if
end for
if $timeCrossed \neq$ None **and** $tempTime \neq$ None **and** $N - timeCrossed \geq d$ **then**
    Append $t[tempTime]$ to T_e
    Append $maxAmp$ to A_e
end if
return T_e, A_e

4.1.3 Smoothed 2nd Derivative Thresholding Discrimination

In this subsection, we provide implementation details for the Smoothed 2nd Derivative Thresholding Discrimination algorithm. A pseudocode is provided in Alg. 3.

Smoothed 2nd Derivative Thresholding Discrimination applies the generic thresholding procedure described in Subsection 4.1 using the smoothed second derivative of the signal as the thresholding feature. The second derivative is computed using a Laplacian of Gaussian (LoG) filter, which enhances peak-like structures while

suppressing high-frequency noise. The Gaussian kernel width (σ) controls the characteristic scale of the features that are enhanced, effectively acting as a tunable filter for pulse width. The threshold is defined as a multiple of the estimated noise standard deviation, which is computed from the late-time region of each frame where no physical pulses are expected.

Candidate event regions are then identified as continuous segments where the second derivative falls below a threshold defined as a multiple of the noise standard deviation. These regions correspond to concave structures in the signal. For each candidate region, the algorithm determines the event locations by identifying local minima of the second derivative (equivalently, the peak of the original signal). Optionally, noisy peaks are filtered as regions smaller than a minimum size to reduce spurious detections caused by noise fluctuations.

If no local minimum is found within a candidate region (e.g. due to residual noise), a fallback strategy is used by selecting the beginning of the region, ensuring that potential events are not missed. The event time is defined as the position of the selected minimum, and the event amplitude is taken directly from the original signal at that position.

Note that a few improvements can be made in this implementation. The first is in the case that no local minimum is found in a region, to define the peak location in the region, the maximal amplitude location is chosen. The second is to add for the user an option to choose how to define peaks in each candidate region. Currently, if no minimal region size is given, the peak is taken as the global minimum in the region to prevent additional noise peak detection. If a minimal region is given by the user, then, in each candidate region, the peaks are defined as all the local minima in the region, to allow for the case of pile-up. However, it can be left up to the user to select between these two options of global or local maxima detection in each region.

Smoothed second-derivative thresholding exploits changes in the signal curvature, allowing closely overlapping events to be separated more effectively than either amplitude or first-derivative thresholding. The smoothing suppresses much of the noise amplification introduced by differentiation, improving robustness. However, the additional filtering increases computational cost and introduces parameters, such as the smoothing scale, that must be selected appropriately. The method therefore offers improved discrimination of overlapping events at the expense of greater algorithmic complexity.

Algorithm 3 Smoothed 2nd Derivative Thresholding Discrimination for Event Detection

Require: Time samples t, amplitudes A, noise percentile p, kernel width σ, noise factor k, minimum size m
Ensure: Event times T_e and event amplitudes A_e

Compute smoothed second derivative:
    $D_2 \leftarrow \text{LoG}(A, \sigma)$ ▷ Compute LoG of A with kernel width σ
Estimate noise level:
    Select indices where $t > \text{percentile}(t, p)$
    $\sigma_{noise} \leftarrow \text{std}(D_2 \text{ in noise region})$
Identify candidate regions:
    Label contiguous regions where $D_2 < -k \cdot \sigma_{noise}$
Initialize empty list of peak indices I_{peaks}
if m is None **then**
    For each labeled region, find index of maximum of $-D_2$
    Add these indices to I_{peaks}
else
    for each labeled region **do**
        Determine region start i_{start} and size
        if region size $> m$ **then**
            Find local minima of D_2 within region
            if no minima found **then**
                Use region start index as fallback
            end if
            Add selected indices to I_{peaks}
        end if
    end for
end if
$T_e \leftarrow t[I_{peaks}]$
$A_e \leftarrow A[I_{peaks}]$
return T_e, A_e

4.2 Pyramid Approximation Preprocessing

In this subsection, we provide implementation details for the Pyramid Approximation Preprocessing algorithm. A pseudocode is provided in Alg. 4. Additional technical details of the Pyramid Approximation algorithm are given in Appendix A.

Pyramid Approximation Preprocessing applies a multiscale (pyramidal) decomposition to detector traces in order to enhance SNR prior to event detection. The approach is based on subdivision schemes, which decompose the signal into a hierarchy of coarse approximations and detail coefficients across multiple scales [9]. Given a detector frame, the signal is decomposed into a multiscale pyramid using a subdivision mask. Each level of the pyramid captures features at a different characteristic scale, with finer scales representing high-frequency components (e.g. noise and sharp pulse features), and coarser scales representing slower variations (e.g. baseline). Several optional processing steps can be applied to the multiscale representation:

- **Denoising:** Small detail coefficients (below a fixed threshold) are suppressed at higher scales (the specific scales choice is a hyper-parameter) that capture rapid, high-frequency variations (often where white noise, baseline ripples, or very fast pulse edges reside). This removes high-frequency noise while preserving significant pulse structures. **This approach was chosen for preprocessing in this work**.
- **Enhancement:** Detail coefficients above a threshold are amplified at selected scales, increasing the prominence of pulse features relative to background.
- **Scale Selection / Multiplication:** Specific scales can be attenuated or removed entirely, allowing selective filtering of the signal.

After preprocessing, the modified signal is reconstructed using the inverse subdivision scheme, yielding an enhanced/filtered version of the original trace.

Event detection can then be performed on the reconstructed signal using any peak-finding method. For the purposes of this work, to allow performance comparisons, we implemented the other three methods listed in this section. However, note that any other peak finding method can be used for the event detection step. The detected event times are taken from the processed trace, while the corresponding amplitudes are extracted from the original trace to preserve the physical pulse height.

The multiscale framework utilised within the Pyramid Approximation provides a flexible and adaptive approach to signal conditioning, allowing improved detection performance in the presence of noise, baseline drift, and overlapping pulses.

Why Subdivision Schemes Instead of Wavelets? While wavelet transforms are a standard tool for multiscale signal analysis, subdivision schemes provide a simpler and more flexible alternative for real-time detector signal processing.

Subdivision-based pyramids are constructed using local, shift-invariant filtering operations with compact support. This makes them computationally efficient and well-suited for streaming applications, where low latency and predictable performance are essential.

In contrast to wavelet transforms, subdivision schemes do not require strict orthogonality or perfect reconstruction constraints, allowing greater flexibility in the choice of filters. This enables the design of masks that are better matched to the characteristic shapes of detector pulses.

Furthermore, subdivision schemes naturally separate the signal into coarse approximations and detail components in a way that is straightforward to manipulate. This facilitates intuitive operations such as threshold-based denoising and scale-dependent enhancement.

Overall, subdivision methods offer a practical balance between multiscale representation, computational efficiency, and adaptability to detector-specific signal characteristics.

Algorithm 4 Pyramid Approximation Preprocessing for Event Detection

Require: Time samples t, amplitudes A, subdivision mask α, support S, number of layers L, enable denoise, denoise scales S_d, denoise thresholds $\{\tau_s\}$, enable enhance, enhance scales S_e, enhance thresholds $\{\tau_s\}$, enhancement factor f, enable scale multiplication, multiplication scales S_m, multiplication factors $\{w_s\}$

Ensure: Event times T_e and event amplitudes A_e

Compute reverse mask γ from (α, S)
Construct multiscale pyramid:
    $P \leftarrow \text{Pyramid}(A, \alpha, \gamma, L)$
if denoising enabled **then**
    for $s \in S_d$ **do**
        Set $P[s][i] \leftarrow 0$ if $|P[s][i]| < \tau_s$
    end for
end if
if enhancement enabled **then**
    for $s \in S_e$ **do**
        Amplify $P[s][i] \leftarrow P[s][i] \cdot f$ if $P[s][i] > \tau_s$
    end for
end if
if scale multiplication enabled **then**
    for $s \in S_m$ **do**
        $P[s] \leftarrow P[s] \cdot w_s$
    end for
end if
Reconstruct enhanced trace:
    $A_{\text{enh}} \leftarrow \text{InversePyramid}(P, \alpha)$
if fixed-threshold method selected **then**
    Detect indices I using fixed-threshold on A_{enh} (Alg. 1)
else if derivative-based method selected **then**
    Detect indices I using derivative threshold on A_{enh} (Alg. 2)
else if Smoothed 2nd Derivative method selected **then**
    Detect indices I using Smoothed 2nd Derivative method on A_{enh} (Alg. 3)
end if
$T_e \leftarrow t[I]$
$A_e \leftarrow A[I]$ ▷ Amplitudes from original trace
return T_e, A_e

4.3 Parameter Tuning

In this subsection, we provide additional details on tuning the parameters of the methods presented in Alg. 1, Alg. 2, Alg. 3, and Alg. 4, for the experiments presented in sections 5 and 6. We remark that the study of systematic methods for parameter tuning is out of the scope of this work, and it is left for future work.

In sections 5 and 6, the parameters of each of the methods were tuned manually, by testing the performance on a small subset of the data, i.e., 5 randomly selected frames from the dataset. The parameters were then selected for each method as those that allowed identification of all the peaks in the sampled frames. Since for the real datasets, there was no ground truth available, only visual confirmation

could be made. We note that for each type of simulation, and each type of detector dataset, parameters had to be tuned again. Thus, for each type of experiment, we provide the selected parameters explicitly in the appropriate section.

Finally, we remark that for the Pyramid Approximation Preprocessing method, some parameters were chosen in advance as defaults for all the experiments, to allow for easier parameter tuning. Specifically, we took a subdivision mask $\alpha = [1/8, 1/2, 6/8, 1/2, 1/8]$, support $S = [-2, -1, 0, 1, 2]$, number of layers $L = 4$, to get a pyramid approximation that corresponds to the non-interpolatory cubic B-spline. The main reason for this choice is as follows. Since the data is noisy, a non-interpolatory scheme will reduce the bias caused by noise. In addition, we enabled only denoising and took the Fixed Threshold Discrimination method as the method of peak finding after the preprocessing step.

5 Numerical Performance on Simulated Data

In this section, we report the results and analysis of each of the algorithms described in Section 4. Specifically, we report a number of key metrics of algorithm performance. The key metrics that we seek to measure:

- Good reliability in identifying significant events.
- The ability to automatically reject very low-energy events (i.e. noise).
- A high time resolution matching the sampling rate of new and existing electronics.
- An improved pulse pair resolution compared to the ~ 10 ns currently achieved with analogue electronics.
- The ability to extract the peak amplitudes and event time with good accuracy.

The three **key performance metrics** that we report are defined as follows:

- **Pulse pair resolution** is defined as the distance (in time) between two neighboring peaks, such that a peak detection method can identify them as two independent peaks.
- **Temporal resolution** is defined as the error in the detected peaks' location (in time) with respect to the known ground truth.
- **Amplitude resolution** is defined as the error in the detected peaks' amplitudes with respect to the known ground truth.

The essential and desirable values for the **key performance metrics** used to evaluate the performance of the selected algorithms are given in Table 3 below.

As the key performance metrics require knowledge of the ground truth, we need to generate simulated data. The simplest way to do this is to simulate trace data based on existing datasets. Here, we based our simulations on the short trace EMU dataset[3]. This allows us to compare the detected peaks of each method with the ground truth of the simulations. In order to evaluate all the metrics, two types of simulations were done: one to evaluate the pulse pair resolution (pulse pair simulations, Subsection 5.1) and one to evaluate the temporal and amplitude resolution

[3]EMU_A14_B13_C61_D62_June2021_Q_RF_LF118_Slit12_short

Metric	Essential	Desirable
Pulse pair resolution	10 ns	3 ns
Temporal resolution	1 ns	0.25 ns
Amplitude resolution	5% of threshold value	1% of threshold value

Table 3: Essential and desirable values for the key metrics used to evaluate the performance of the algorithms from Section 4.

(full trace simulations, Subsection 5.2). Full details are provided in the following subsections. Note that the algorithms' parameters were tuned for each type of simulation.

Computing the Amplitude Threshold. To compute the amplitude resolution, the amplitude threshold value between the noise and the signal is required. In order to find it, a double-Gaussian model is fitted to the amplitude histogram to characterise the underlying noise and signal distributions, see Fig. 3 for an illustration. The resulting fit, together with its individual Gaussian components, is used to guide threshold selection. A dataset-dependent search interval is defined, within which the fitted curve is evaluated to identify a local minimum corresponding to the separation between noise and signal populations. This minimum is determined by locating extrema in the negated fitted function and is subsequently adopted as an adaptive amplitude threshold for event filtering.

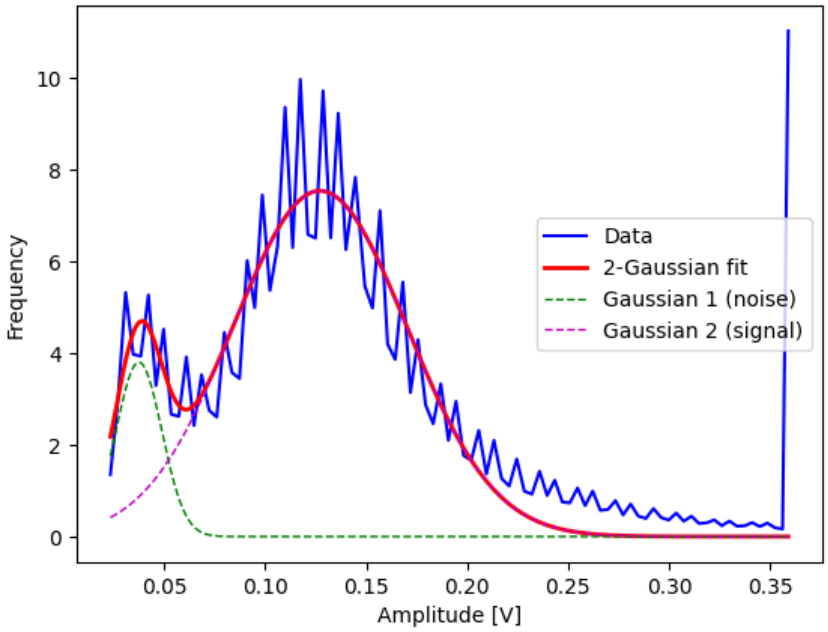


Figure 3: An illustration of computing the amplitude threshold.

False positive and false negative peaks. In this section, the simulated datasets allow us to utilise ground truth data on all the peak features (location, amplitude, etc.). This, in turn, allows us to define and report the rates of false positive and false negative peaks. Even though a complete analysis of false positive and false

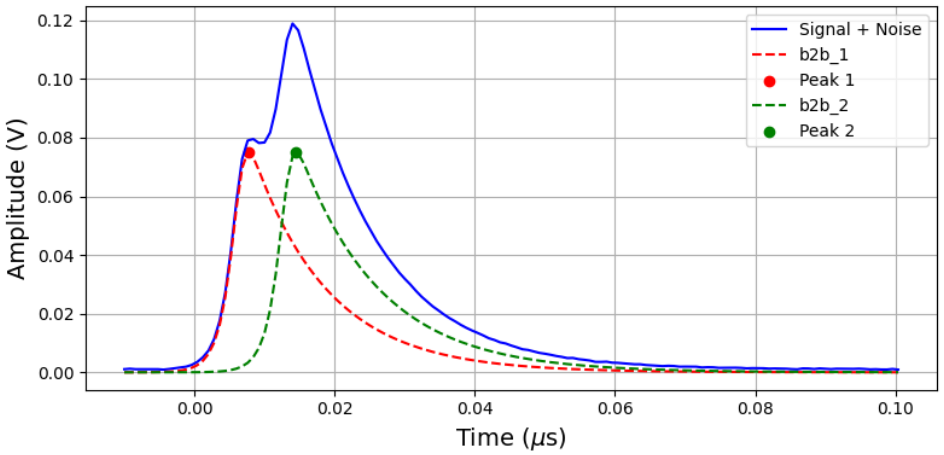
negative peak rates for each of the methods is out of the scope of this work, we exploit having ground truth data to report these rates for our simulated data.

For clarity, we explicitly define each of the terms. In the pulse pair simulations, we only count the number of identified peaks; thus, if a method finds more than two peaks in a specific simulation, it must be that it found a false positive peak. For the full trace simulations, false positive peaks are all the peaks that a method classified as peaks; however, they are not peaks according to the ground truth data. Similarly, false negative peaks, are all the ground truth peaks that were not identified or that the closest identified peak is too far (more than 50 ns away).

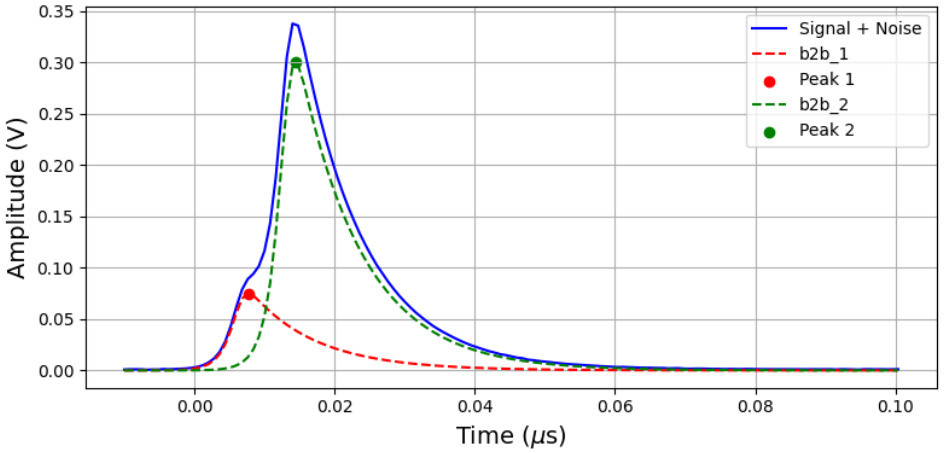
5.1 Pulse Pair Resolution Performance

We simulated various pairs of peaks, with 30 distances varying between the desired and three times the essential resolutions. We simulated three different cases: equal amplitude peaks, a low peak followed by a high peak, and a high peak followed by a low peak, where in both cases the low peak was ~ 4 times lower than the high peak. In total, $9,000$ simulations of a pair of peaks were performed.

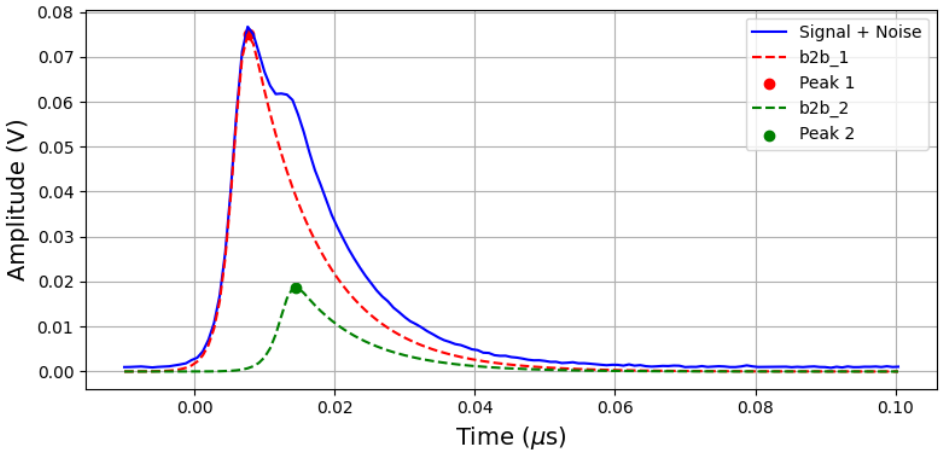
We used a Back-to-Back Exponential function to model the Muon trace peak shape. For each simulation, the parameters of each Back-to-Back Exponential component were randomly perturbed to mimic realistic experimental variations of 10%, from a baseline of a fitted Back-to-Back Exponential peak from the Short EMU Trace dataset. To simulate the experimental backgrounds, a linear baseline was added, followed by an additive Gaussian noise with fixed standard deviation. In Fig. 4, an example of each of the pulse pair simulation types is illustrated.



(a) Equal peaks



(b) Low-high peaks



(c) High-low peaks

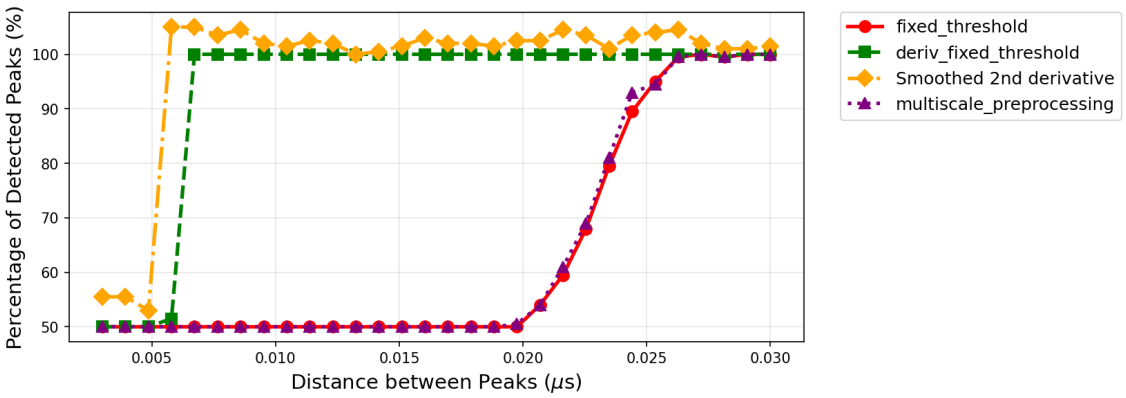
Figure 4: An example of simulated pulse-pair data with a distance of 6.1 ns.

Parameter tuning. The parameters of the methods were tuned on a randomly chosen simulated example of high and then low peaks, with a distance of 10 ns. The parameters can be found in Table 4.

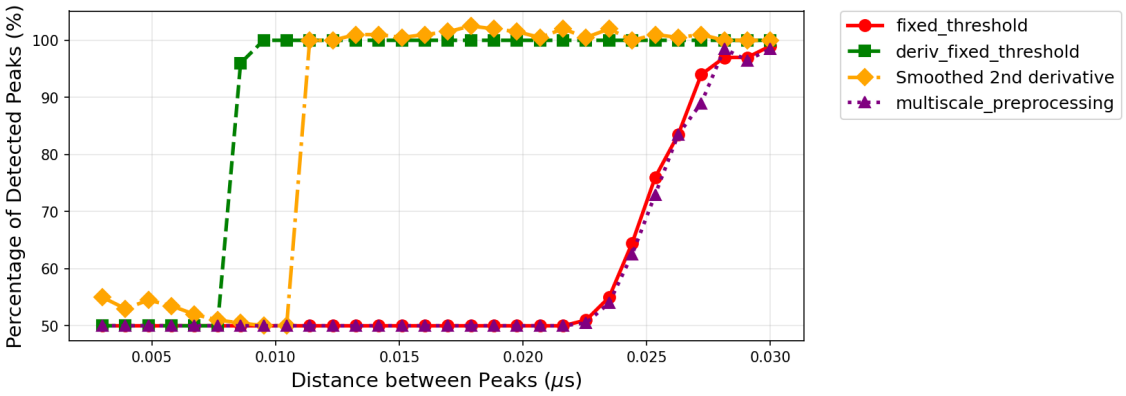
Method	parameters
Fixed Thresholding	$\theta = 0.02$ V, $d = 4$, $c = 0$
1st Derivative Thresholding	$\theta = 0.003 \frac{\text{V}}{\mu\text{s}}$, $d = 4$, $c = 0$
Smoothed 2nd Derivative	$p = 99$, $\sigma = 3$, $k = 10$, $m = 5$
Pyramid Approximation Preprocessing	denoise layers $1 - 4$, threshold 0.002 V

Table 4: Methods' parameters for the pulse pair simulation analysis. V stands for Volt.

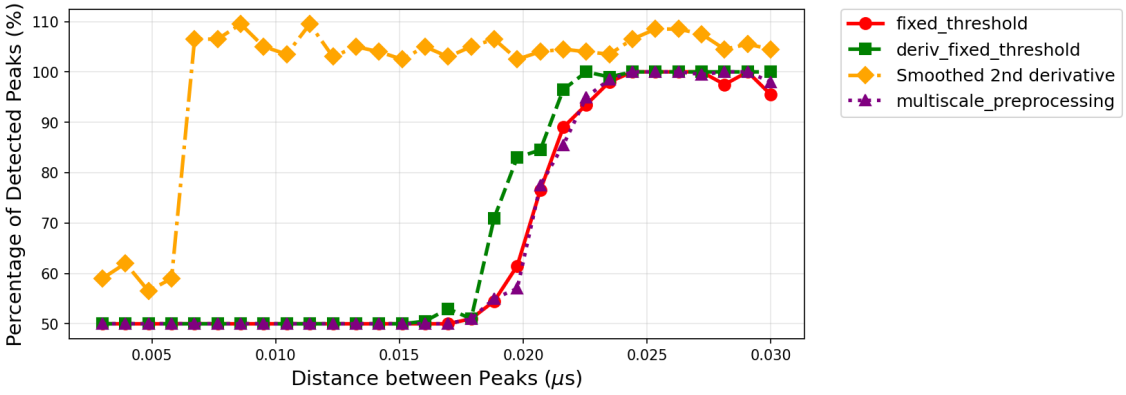
Results for each of the three simulation types are reported as the percentage of detected peaks as a function of the distance between peaks. The results are given in Fig. 5. From the results, it can be seen that both first and second-derivative thresholding algorithms perform very well; however, the second-derivative thresholding also finds more false positive peaks, as for some of the distances, there are more than 100% identified peaks.



(a) Pulse Pair Detection Comparison for equal amplitudes



(b) Pulse Pair Detection Comparison for low-high amplitudes



(c) Pulse Pair Detection Comparison for high-low amplitudes

Figure 5: Results for the pulse pair resolution of the selected algorithms, plotted as the percentage of detected peaks against the distance between peaks in μs.

5.2 Temporal and Amplitude Resolution Performance

To evaluate temporal and amplitude resolution, full trace datasets were simulated using an exponential decay rate of peak count with respect to muon lifetime, and amplitude distribution based on the Short EMU Trace dataset.

For each simulated frame, an initial population of (20) muons was assumed within a (20 μs) acquisition window. The decay time of each muon was independently sampled from an exponential distribution with mean lifetime ($\tau_\mu = 2.197$ μs), producing an exponentially decreasing temporal distribution of pulse centers (X_0). Events with decay times outside the acquisition window were discarded.

Pulse amplitudes were independently sampled from a Gaussian distribution with mean (0.075) and standard deviation (0.01), and truncated to the interval ([0.02, , 0.1]) to ensure physically meaningful positive intensities.

Each signal pulse was modelled using a Back-to-Back Exponential function. The baseline shape parameters, of a fitted Back-to-Back Exponential peak from the Short EMU Trace dataset, were perturbed by random variations to mimic realistic experimental fluctuations in detector response.

A linear background component was added to represent slow electronic drift, followed by additive Gaussian noise with fixed variance to reproduce the detector noise floor. Negative values after noise addition were clipped to zero. A total of 100 independent frames were simulated. An example frame is illustrated in Fig. 6.

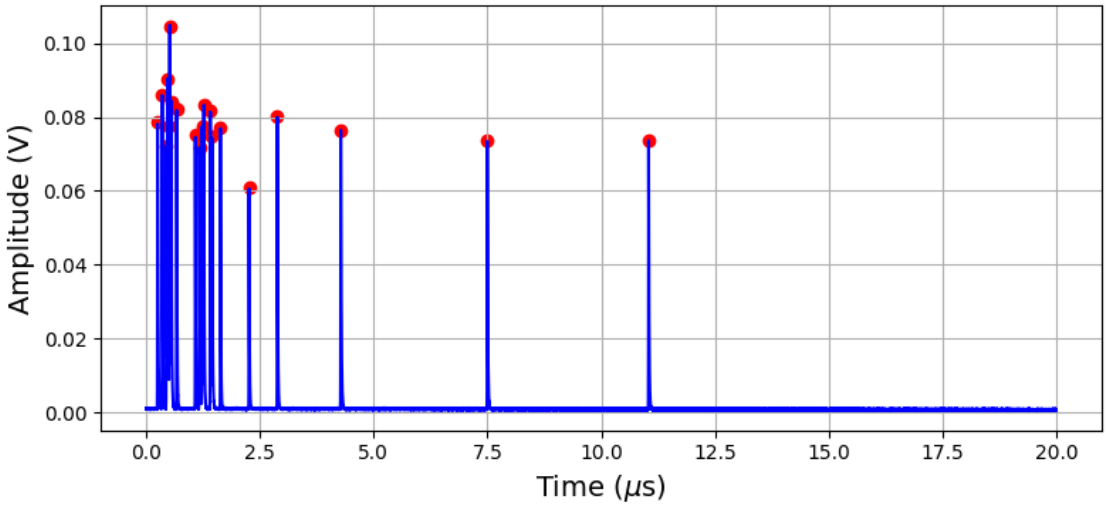


Figure 6: An example frame from a full trace simulation that is used for temporal and amplitude resolution performance evaluation of the selected algorithms.

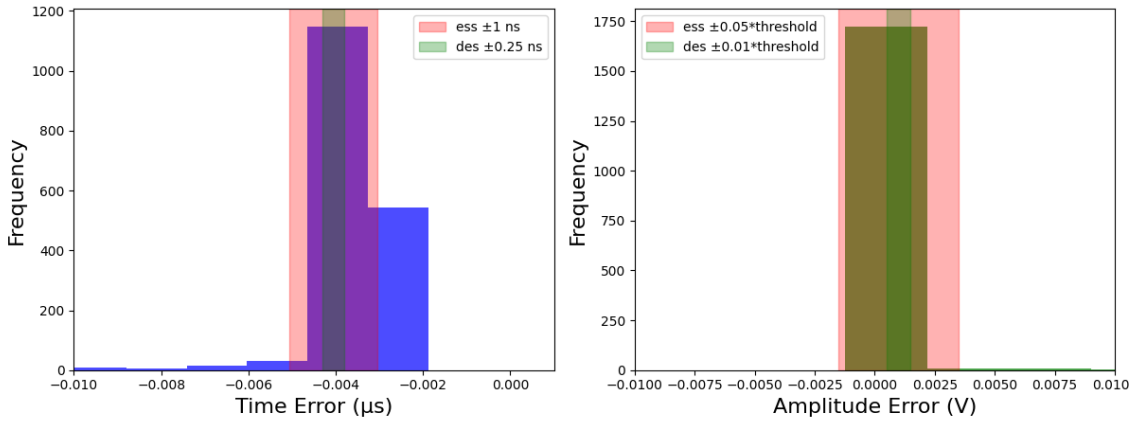
Parameter tuning. The parameters of the methods were tuned on a randomly chosen simulated frame, and their values remain the same as for the pulse pair simulation. The parameters can be found in Table 5.

Method	parameters
Fixed Thresholding	$\theta = 0.02$ V, $d = 4$, $c = 0$
1st Derivative Thresholding	$\theta = 0.003$ $\frac{\text{V}}{\mu\text{s}}$, $d = 4$, $c = 0$
Smoothed 2nd Derivative	$p = 99$, $\sigma = 3$, $k = 10$, $m = 5$
Pyramid Approximation Preprocessing	denoise layers $1 - 4$, threshold 0.002 V

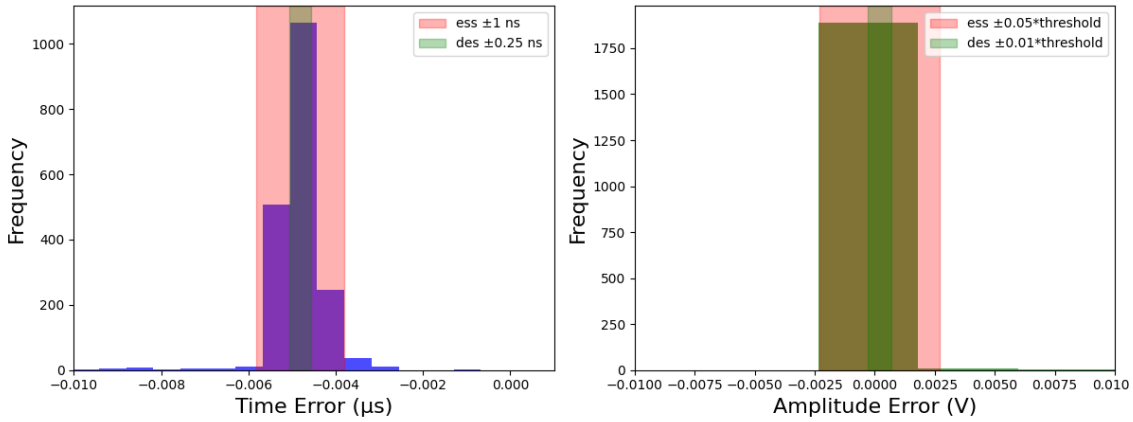
Table 5: Methods' parameters for the full trace simulation analysis.

Results are reported as histograms of the error for each of the algorithms, and the desired and essential resolutions are marked for reference, see Fig. 7 and Fig.

8. In Fig. 9 and Fig. 10, all the histograms are aligned for comparison of the temporal resolution and amplitude resolution, respectively. In addition, we report the percentages of false positive and false negative peaks for each method in Table 6. We restate that false positive peaks are defined as points that a method incorrectly identifies as peaks. False negative peaks are defined as peaks that a method did not identify. In addition, we report the percentage of errors included in the desired and essential resolutions in Table 6. From the results, it can be seen that the Smoothed 2nd Derivative Thresholding method performs better than the others.

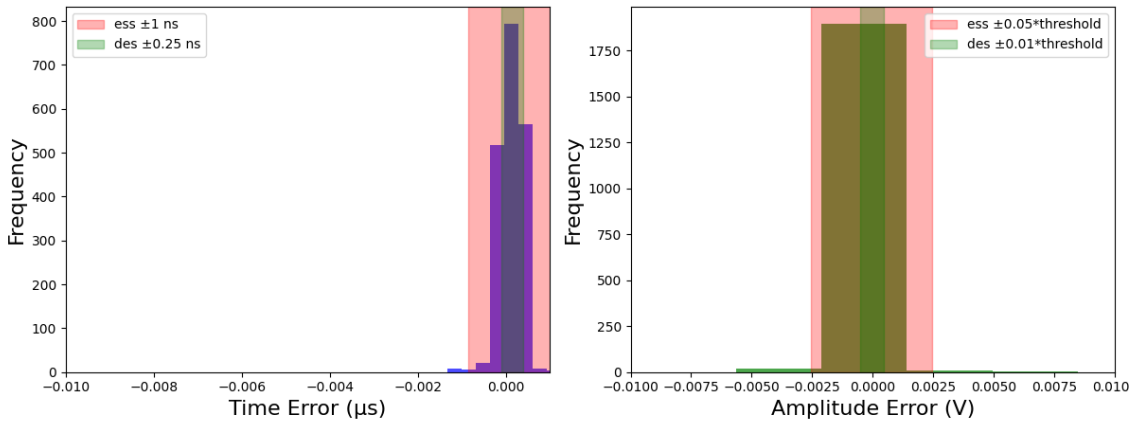


(a) Time errors (left) and amplitude errors (right) histograms for Fixed Threshold Discrimination.

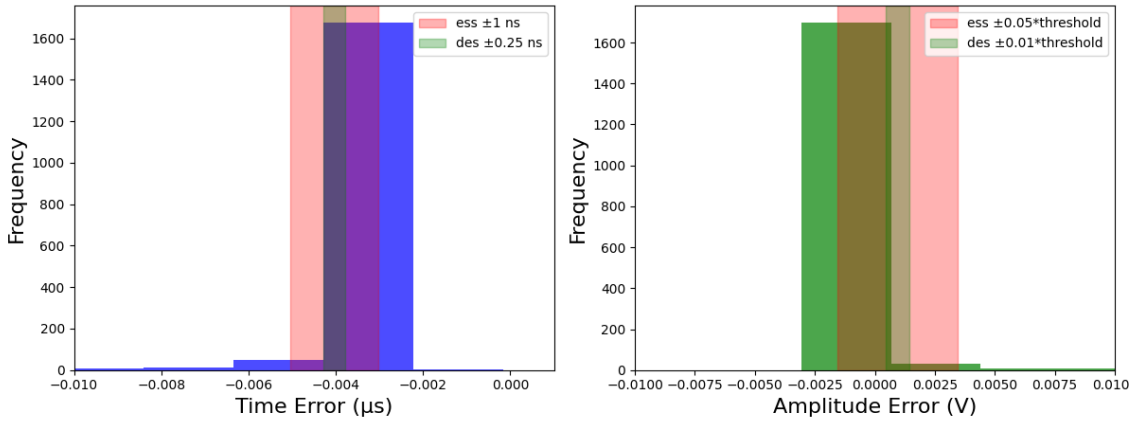


(b) Time errors (left) and amplitude errors (right) histograms for 1st Derivative Fixed Threshold Discrimination.

Figure 7: Results for the temporal resolution and amplitude resolution of the (a) fixed threshold and (b) 1st Derivative Fixed Threshold. Desired (des) and essential (ess) regions are highlighted in green and red correspondingly.



(a) Time errors (left) and amplitude errors (right) histograms for Smoothed 2nd Derivative Thresholding Discrimination.



(b) Time errors (left) and amplitude errors (right) histograms for Pyramid Approximation Preprocessing.

Figure 8: Results for the temporal resolution and amplitude resolution of the (a) Smoothed 2nd Derivative Thresholding and (b) Pyramid Approximation Preprocessing. Desired (des) and essential (ess) regions are highlighted in green and red correspondingly.

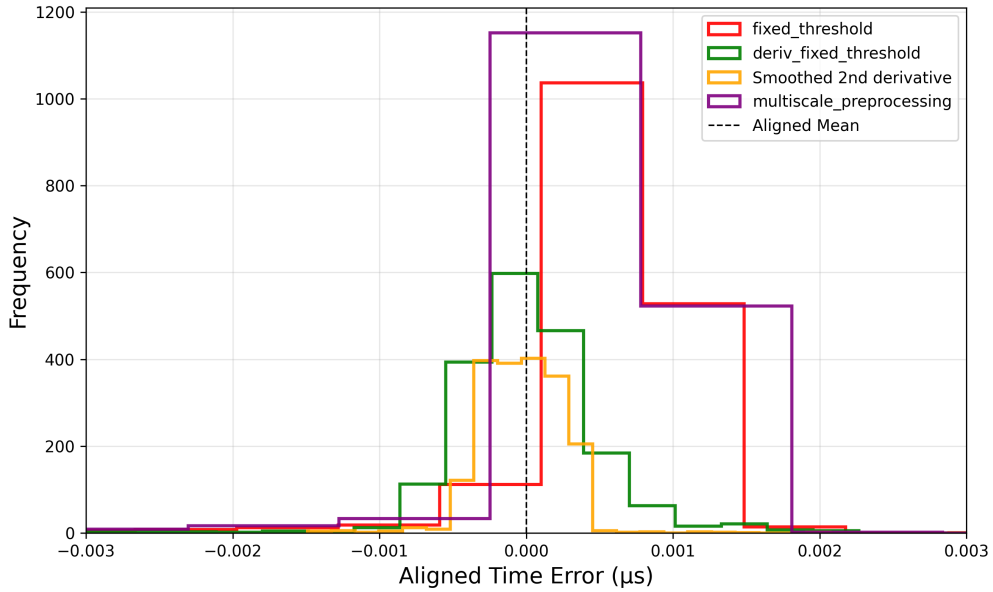


Figure 9: Aligned histograms of time errors for all the methods.

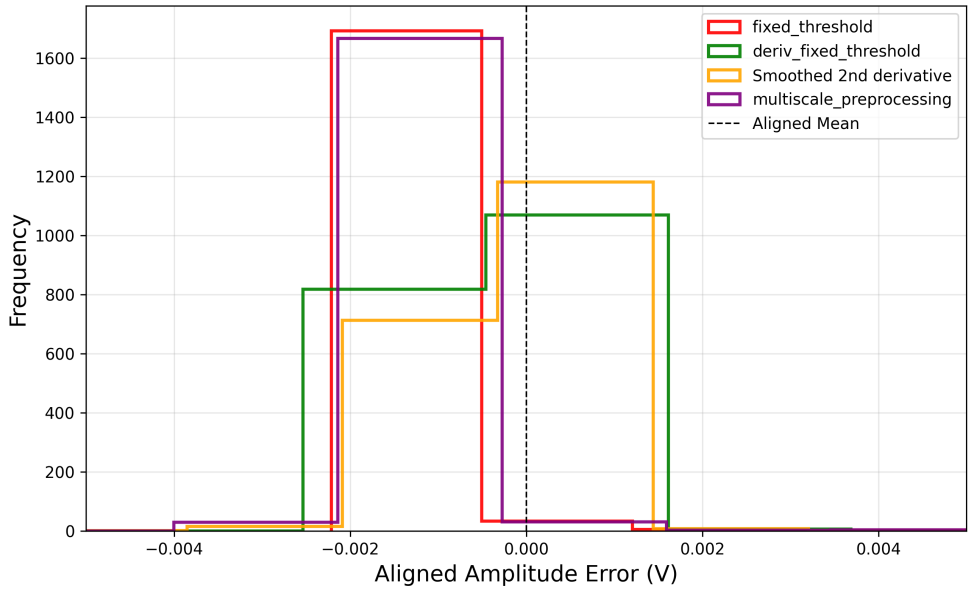


Figure 10: Aligned histograms of amplitude errors for all the methods.

Method	FP	FN	des time	ess time	time SD (ns)	des amp	ess amp	amp SD (V)
Fixed Threshold	0.00%	9.60%	13.27%	80.59%	0.374	1.44 %	95.69 %	$9.32 * 10^{-3}$
Derivative Threshold	0.88%	3.90%	47.45%	94.90%	0.716	58.64%	98.49%	$5.33 * 10^{-3}$
Smoothed 2nd derivative	0.00%	3.00%	61.08%	98.56%	0.327	71.03%	98.35%	$2.46 * 10^{-3}$
Pyramid	0.00%	9.55%	13.99%	82.37%	0.372	1.49%	95.58%	$9.32 * 10^{-3}$

Table 6: False positive (FP) percentage and false negative percentage, the empirical percentage of time and amplitude (amp) errors included in the desired (des) and essential (ess) resolutions and the corresponding standard deviation (SD) from a Gaussian fit to the error distributions, for each of the methods.

6 Numerical Performance on Real Data

In this section, we present the results for each of the methods on real data. Specifically, we test our methods on datasets from three detectors: EMU Instrument (Subsection 6.3), HiFi Instrument (Subsection 6.4), and Super-MuSR Detector on MuSR Instrument (Subsection 6.5). For each detector, the methods' parameters had to be tuned separately (see Subsection 6.1).

To assess the performance of each method, per-channel histograms of detected peak times and amplitudes were computed. From the amplitude histogram, the amplitude threshold was found using the same method as in Section 5. Using the amplitude threshold, the histogram of detected peak times was filtered to reduce noisy peaks. Finally, the filtered histogram of detected peak times was utilised to find the fitted muon lifetime as explained in Subsection 6.2.

6.1 Parameter Tuning

In this subsection, we report the tuned parameters that were utilised for each type of data. Tables 7, 8, and 9, show the tuned parameters for EMU Instrument, HiFi Instrument, and Super-MuSR Detector on MuSR instrument respectively.

Method	parameters
Fixed Thresholding	$\theta = 0.02$ V, $d = 4$, $c = 0$
1st Derivative Thresholding	$\theta = 0.009 \frac{\text{V}}{\mu\text{s}}$, $d = 4$, $c = 0$
Smoothed 2nd Derivative	$p = 99$, $\sigma = 3$, $k = 10$, $m = 5$
Pyramid Approximation Preprocessing	denoise layers $1 - 4$, threshold 0.002 V

Table 7: Methods' parameters for EMU.

Method	parameters
Fixed Thresholding	$\theta = 0.01$ V, $d = 4$, $c = 0$
1st Derivative Thresholding	$\theta = 0.004 \frac{\text{V}}{\mu\text{s}}$, $d = 4$, $c = 0$
Smoothed 2nd Derivative	$p = 99$, $\sigma = 3$, $k = 4$, $m = 5$
Pyramid Approximation Preprocessing	denoise layers $1 - 4$, threshold 0.002 V

Table 8: Methods' parameters for HiFi.

Method	parameters
Fixed Thresholding	$\theta = 120$ LSB, $d = 10$, $c = 0$
1st Derivative Thresholding	$\theta = 14\ \frac{\text{LSB}}{\mu\text{s}}$, $d = 100$, $c = 0$
Smoothed 2nd Derivative	$p = 90$, $\sigma = 3$, $k = 10$, $m = 5$
Pyramid Approximation Preprocessing	denoise layers $1 - 4$, threshold $[2, 2, 4, 4]$ LSB

Table 9: Methods' parameters for Super-MuSR. Note that the amplitude units are LBS. For the Pyramid Approximation Preprocessing, we use different per-layer denoising threshold.

6.2 Dead Time Analysis (Muon lifetime fit)

In this subsection, we elaborate on the muon lifetime fit that is later used to assess the performance of the different methods that we tested. For each method, and for each channel, the following process is performed. The observed muon decay histogram is modelled as an exponential decay with a constant background contribution. Let t denote time (in μs), $N(t)$ the measured count rate, and N_{frames} as the number of frames. The data are first normalised by the number of frames and bin width, yielding a count rate in units of counts per microsecond:

$$N_{\text{norm}}(t) = \frac{N(t)}{N_{\text{frames}} \cdot \Delta t}. \tag{1}$$

To characterise the decay, the normalised data are fitted over a restricted time window $[t_{\min}, t_{\max}]$ using the model

$$f(t; A, \tau, B) = Ae^{-t/\tau} + B, \tag{2}$$

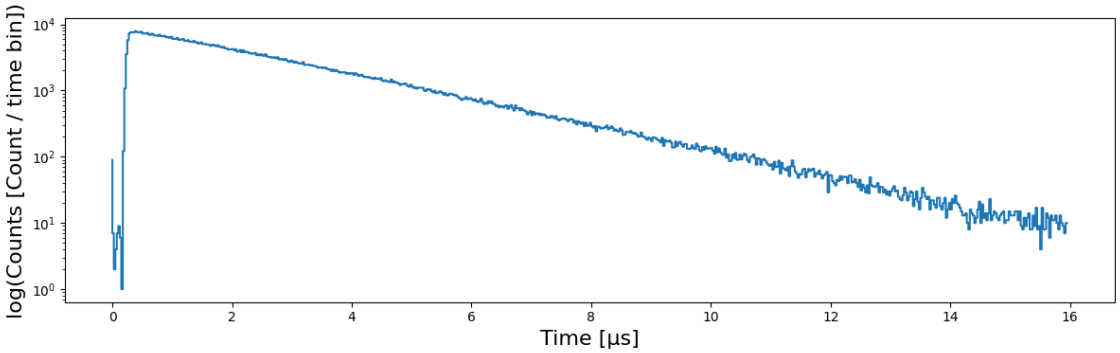
where A is the amplitude, τ is the decay constant, and B represents a constant non-negative background. The fitted decay constant τ_{fit} is compared to the known muon lifetime $\tau_\mu = 2.197$ μs.

6.3 EMU Instrument Datasets

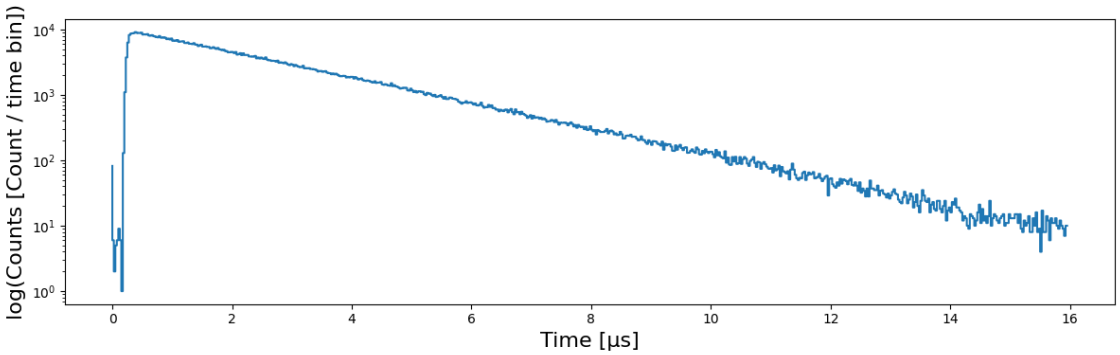
In this subsection, we present the results for the Emu Instrument Datasets. First, technical details for the datasets are given in Table 10. In Fig. 11 and Fig. 12, examples of time and amplitude histograms are presented. Finally, in Fig. 13, for each method, the mean fitted muon lifetime across channels is presented as a function of the slit separation, and the error bars represent the standard deviation of the fitting across channels. Then, a linear fit is fitted to the mean values as a function of the slit separation. From Fig. 13, it can be seen that the derivative-based method outperforms the other two methods. Specifically, the smoothed second-derivative thresholding method reaches the same level of deadtime distortion at two times higher event rates (80 slits) than the fixed thresholding baseline method and the multiscale method (40 slits). Similarly, the derivative thresholding method at 40 slits gets to the level of deadtime distortion that the fixed thresholding baseline method and the multiscale method have at approximately 10 slits (four times higher event rates).

Slit separation	Frames	Channels	Samples	dt (ns)
4.6	64,659	4	20,000	0.8
8	15,417	4	20,000	0.8
15	14,815	4	20,000	0.8
20	19,756	4	20,000	0.8
40	20,223	4	20,000	0.8
80	13,816	4	20,000	0.8

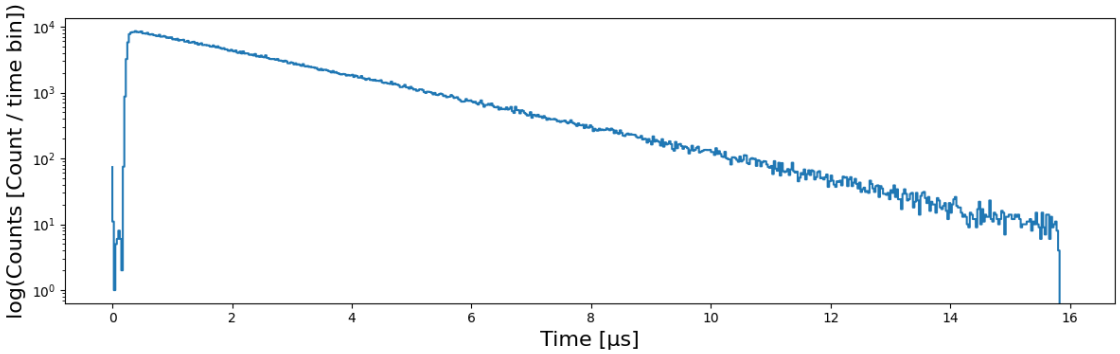
Table 10: EMU datasets specifications: slit separation, number of frames, number of channels, number of time samples, time sample interval (dt).



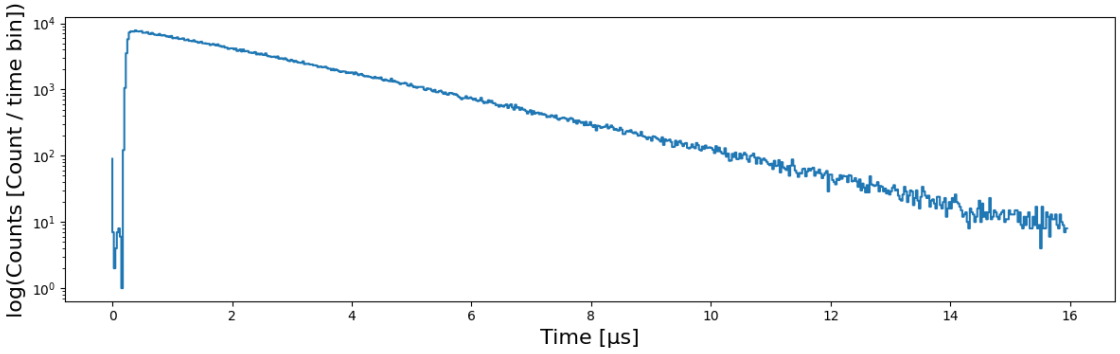
(a) Detected peak times histograms - Fixed Threshold Discrimination.



(b) Detected peak times histograms - 1st Derivative Fixed Threshold Discrimination.

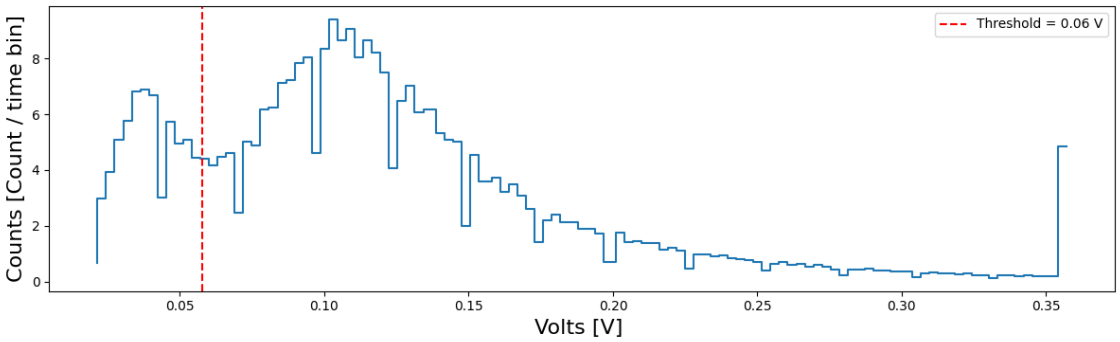


(c) Detected peak times histograms - Smoothed 2nd Derivative Thresholding Discrimination.

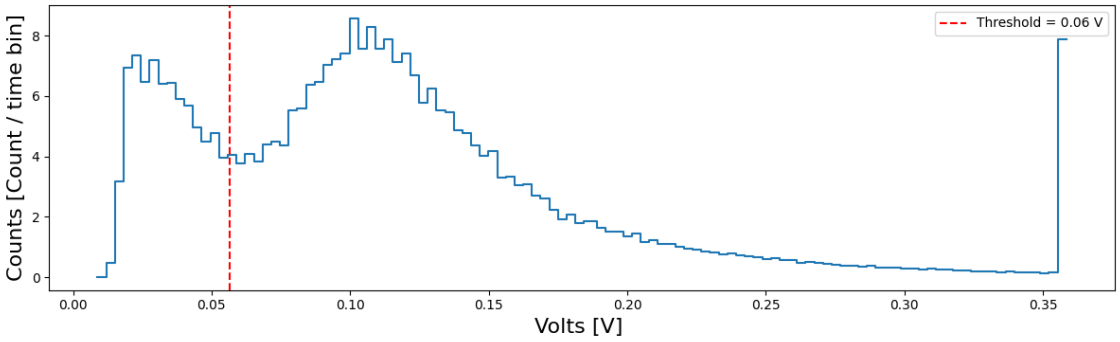


(d) Detected peak times histograms - Pyramid Approximation Preprocessing.

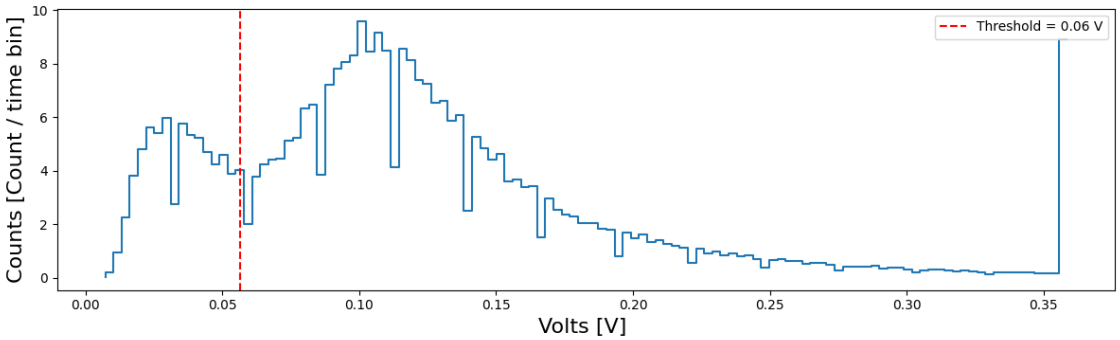
Figure 11: An example of detected peak time histograms for each of the methods applied to the slit setting 40 EMU dataset, of channel 0.



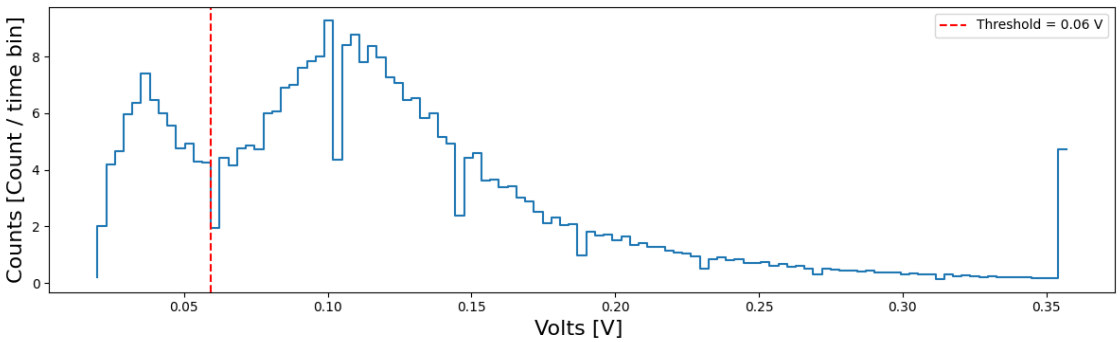
(a) Detected peak amplitudes histograms - Fixed Threshold Discrimination.



(b) Detected peak amplitudes histograms - 1st Derivative Fixed Threshold Discrimination.



(c) Detected peak amplitudes histograms - Smoothed 2nd Derivative Thresholding Discrimination.



(d) Detected peak amplitudes histograms - Pyramid Approximation Preprocessing.

Figure 12: An example of detected peak amplitude histograms for each of the methods applied to the slit setting 40 EMU dataset, of channel 0. The red dashed line is the amplitude threshold value for true peaks.

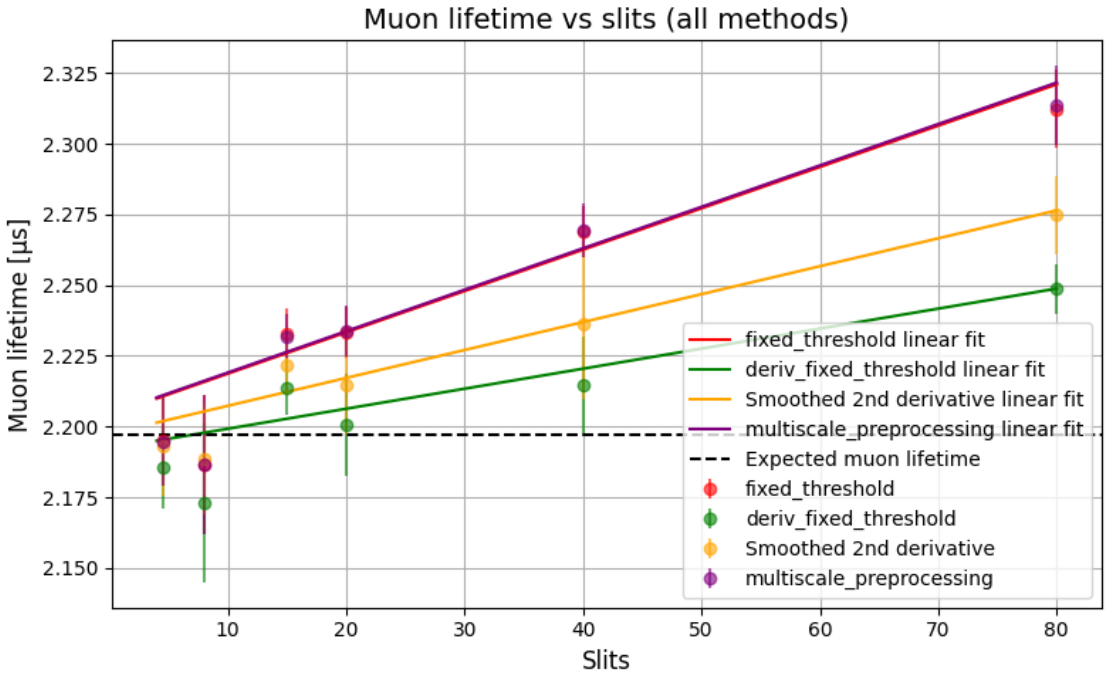
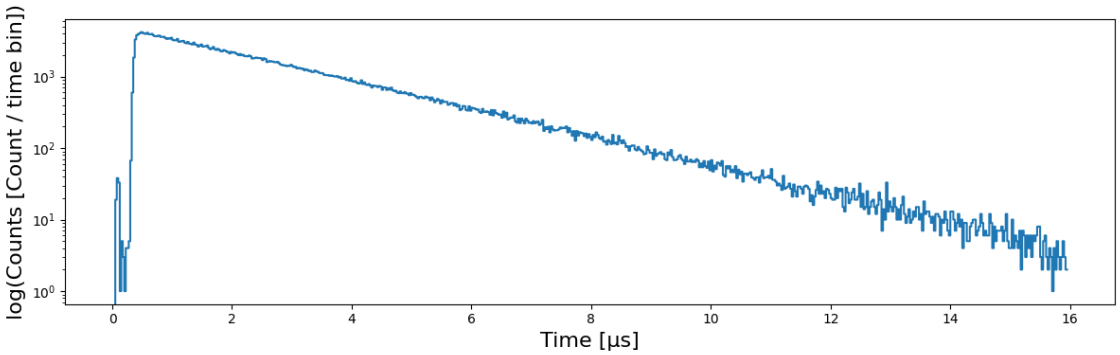


Figure 13: Fitted muon lifetime as a function of slit setting for each of the methods.

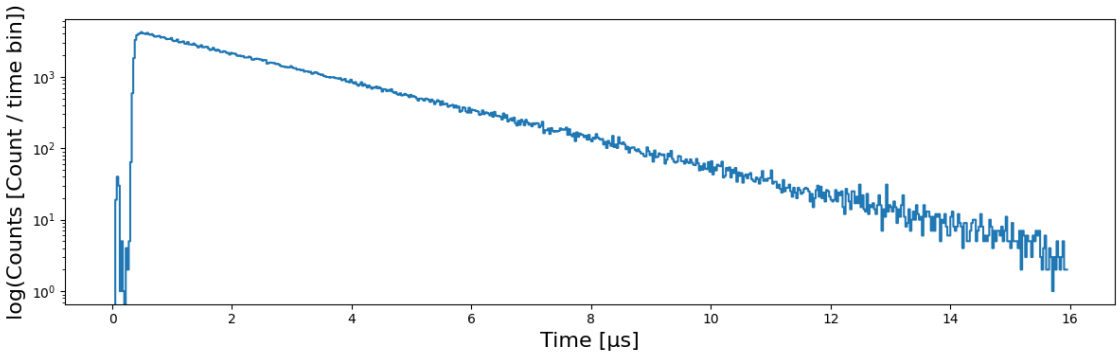
6.4 HiFi Instrument Dataset

In this subsection, we present the results for the HiFi Instrument Dataset. The dataset[4] consists of 38810 frames, 4 channels, 20000 time samples, and with a time sample interval of 0.8 ns. It was collected with a slit setting of 20, which is a typical setting used on the instrument. In Fig. 14 and Fig. 15, examples of the time and amplitude histograms are presented. Finally, in Table 11, for each method we provide the muon lifetime fit, and its standard deviation across channels.

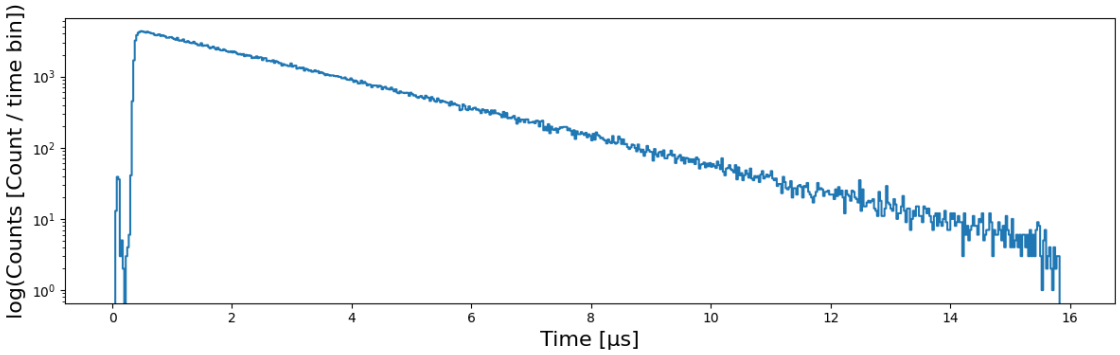
[4]HiFi_17Sept20_Slit20_ZF.traces



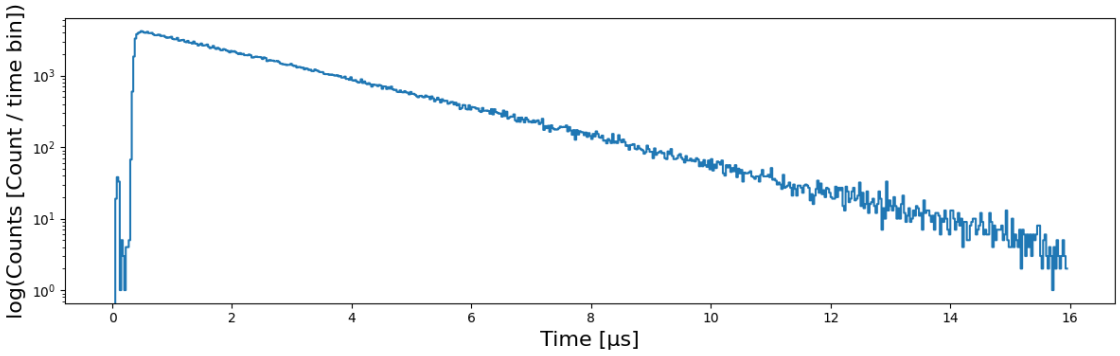
(a) Detected peak time histograms - Fixed Threshold Discrimination.



(b) Detected peak time histograms - 1st Derivative Fixed Threshold Discrimination.

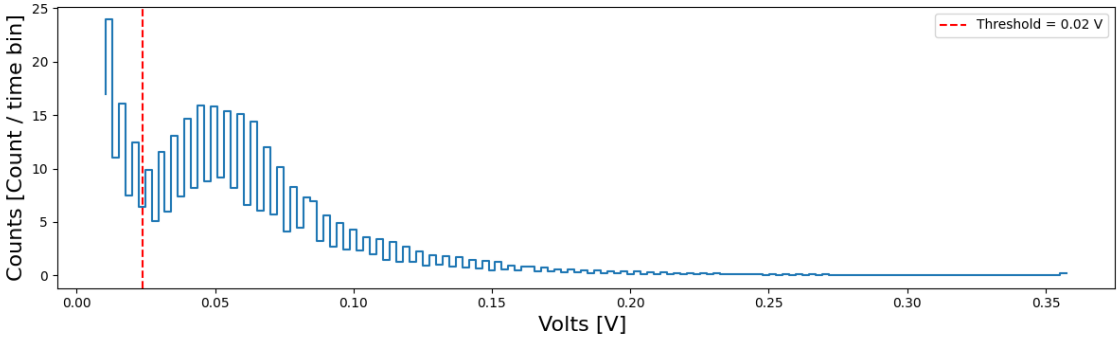


(c) Detected peak time histograms - Smoothed 2nd Derivative Thresholding Discrimination.

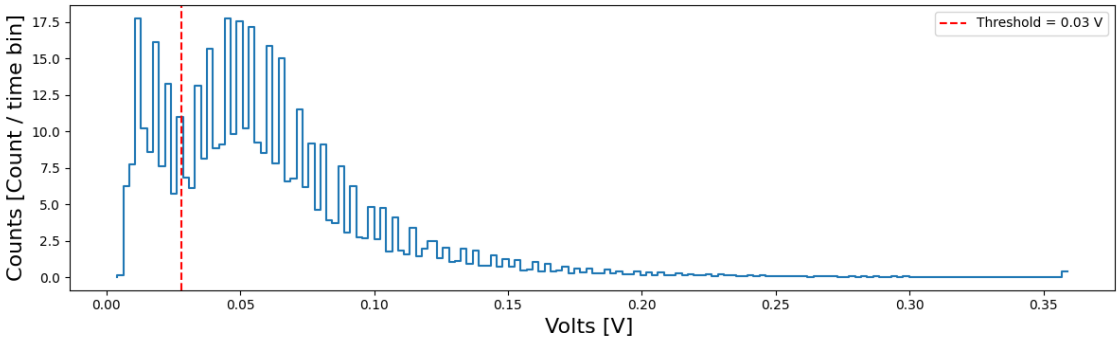


(d) Detected peak time histograms - Pyramid Approximation Preprocessing.

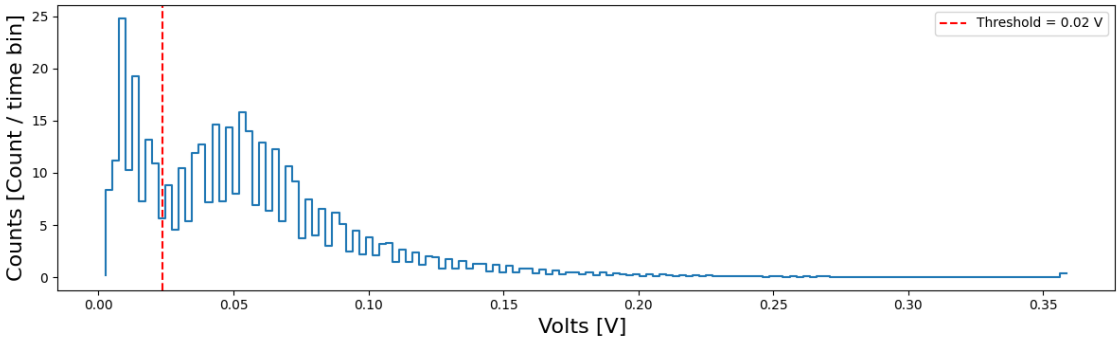
Figure 14: An example of detected peak time histograms for each of the methods for the HiFi dataset, of channel 0.



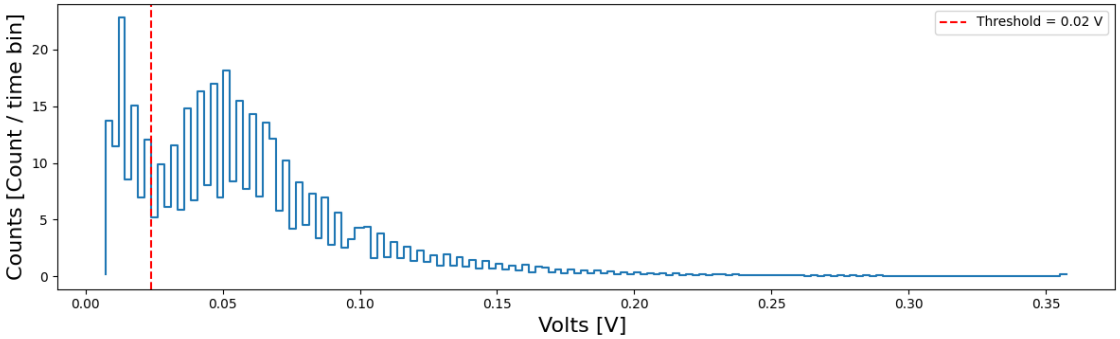
(a) Detected peak amplitude histograms - Fixed Threshold Discrimination.



(b) Detected peak amplitude histograms - 1st Derivative Fixed Threshold Discrimination.



(c) Detected peak amplitude histograms - Smoothed 2nd Derivative Thresholding Discrimination.



(d) Detected peak amplitude histograms - Pyramid Approximation Preprocessing.

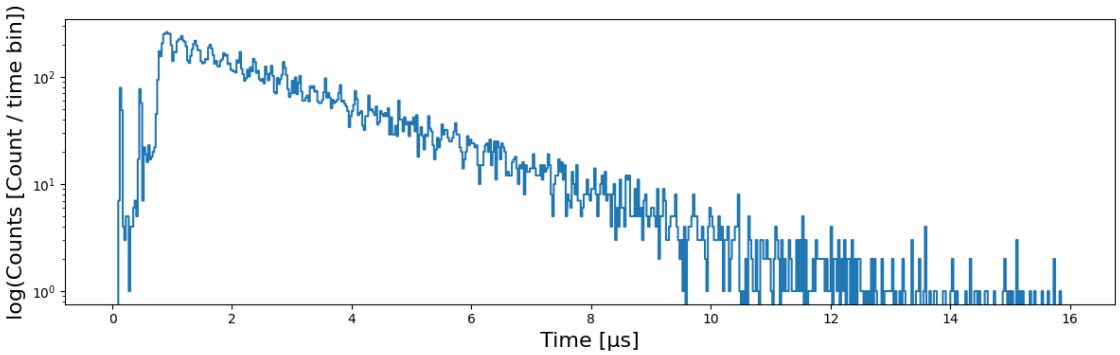
Figure 15: An example of detected peak amplitude histograms for each of the methods for channel 0 of the HiFi dataset. The red dashed line is the amplitude threshold value for true peaks.

Method	Muon lifetime fit (μs)	SD (μs)
Fixed Thresholding	2.207	0.0128
1st Derivative Thresholding	2.183	0.0174
Smoothed 2nd Derivative	2.184	0.0185
Pyramid Approximation Preprocessing	2.206	0.0109

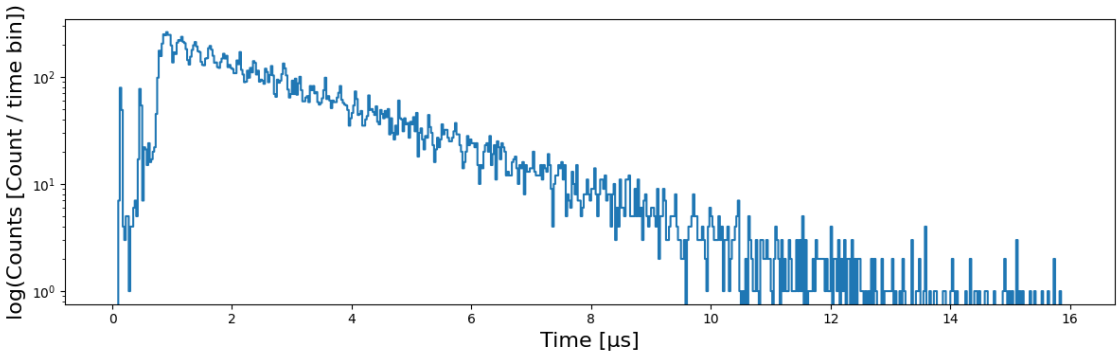
Table 11: Muon lifetime fit for each method for the HiFi dataset.

6.5 Super-MuSR Prototype Detector on MuSR

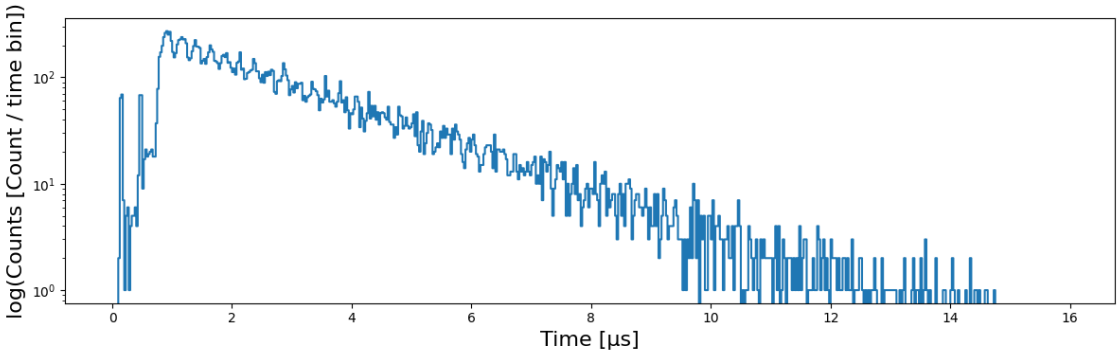
In this subsection, we present results for the analysis of data collected using a Super-MuSR prototype detector mounted on the MuSR instrument. This dataset consists of 14583 frames, 8 channels, 16500 time samples, and with a time sample interval of 1 ns. The detector was installed on the MuSR instrument for testing purposes, further from the sample than it will ultimately be installed. This leads to a lower counting rate on each tile but a realistic signal-to-noise profile and event shape. For the analysis in this work, we used only the first 4 channels. In Fig. 16 and Fig. 17, examples of the time and amplitude histograms are presented. Finally, in Table 12, for each method we provide the muon lifetime fit, and its standard deviation across channels.



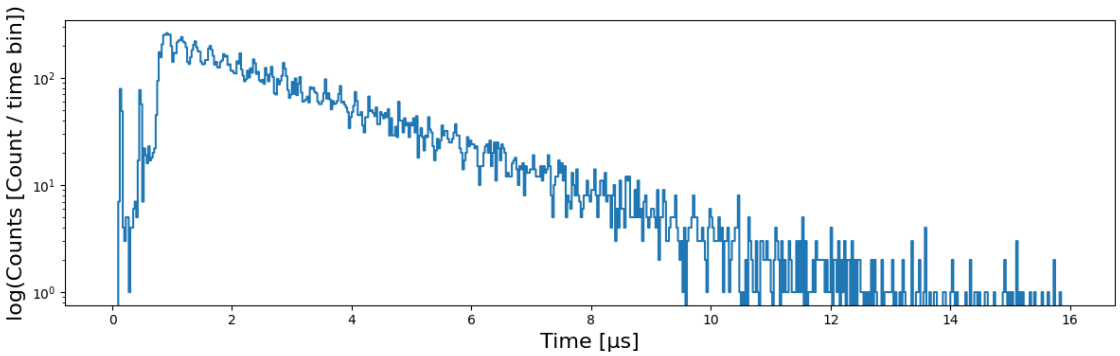
(a) Detected peak example time histogram - Fixed Threshold Discrimination.



(b) Detected peak example time histogram - 1st Derivative Fixed Threshold Discrimination.

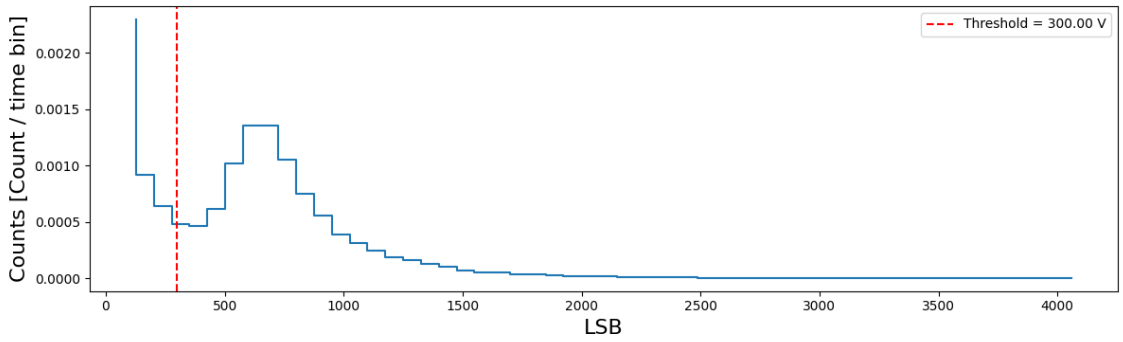


(c) Detected peak example time histogram - Smoothed 2nd Derivative Thresholding Discrimination.

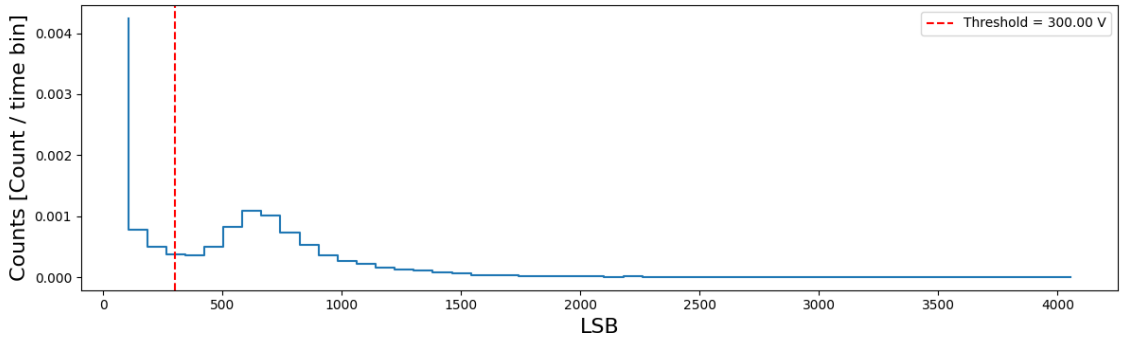


(d) Detected peak example time histogram - Pyramid Approximation Preprocessing.

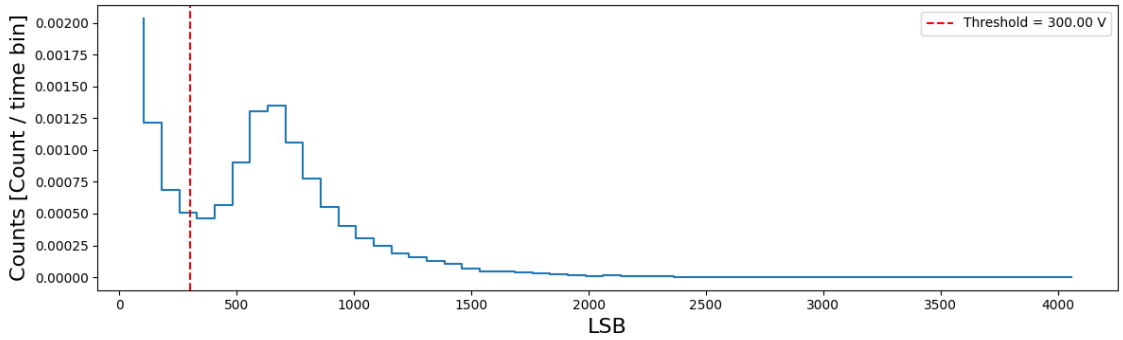
Figure 16: An example of detected peak time histograms for each of the methods for the Super-MuSR Detector on MuSR Instrument dataset, of channel 0.



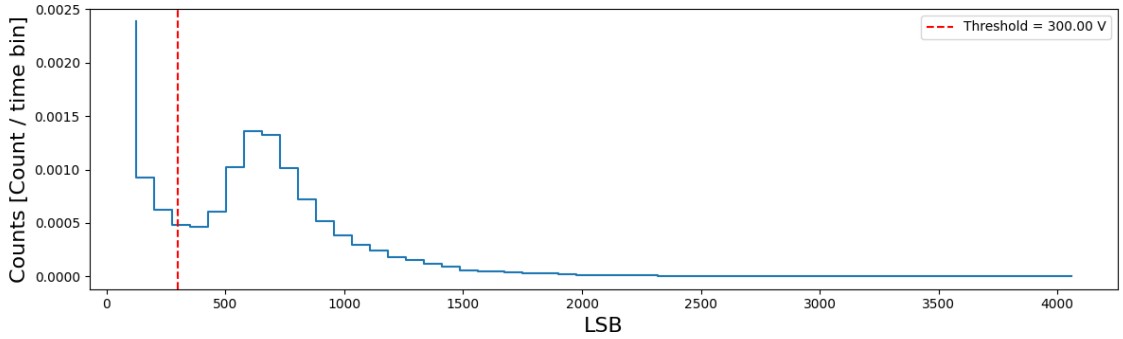
(a) Detected peak example amplitude histogram - Fixed Threshold Discrimination.



(b) Detected peak example amplitude histogram - 1st Derivative Fixed Threshold Discrimination.



(c) Detected peak example amplitude histogram - Smoothed 2nd Derivative Thresholding Discrimination.



(d) Detected peak example amplitude histogram - Pyramid Approximation Preprocessing.

Figure 17: An example of detected peak amplitude histograms for each of the methods for channel 0 of the prototype Super-MuSR Detector installed on MuSR dataset. The red dashed line is the amplitude threshold value for true peaks.

Method	Muon lifetime fit (μs)	SD (μs)
Fixed Thresholding	2.178	0.0885
1st Derivative Thresholding	2.192	0.0882
Smoothed 2nd Derivative	2.166	0.0903
Pyramid Approximation Preprocessing	2.179	0.0885

Table 12: Muon lifetime fit for each method for the prototype Super-MuSR Detector installed on MuSR.

7 Conclusion

In this work, we investigated a range of event identification methods for muon detector traces, with the goal of improving timing accuracy, pulse-pair resolution, and robustness to noise in high-rate environments. Several classes of approaches were reviewed, including direct methods, data filtering methods, and combined filtering–detection methods, from which four computationally efficient algorithms were selected for detailed study.

The numerical results on simulated data demonstrate that derivative-based methods provide improved temporal resolution and pulse-pair separation compared to standard fixed-threshold discrimination, while maintaining competitive amplitude accuracy. In particular, the smoothed second-derivative approach achieves strong performance across multiple metrics, although at the cost of increased sensitivity to false positive peaks in the pulse pair resolution simulation results. However, note that the method control parameters are not optimal, and with more careful tuning, the sensitivity to false positive peaks could be decreased. The Pyramid Approximation Preprocessing method showed very similar results to the Fixed Threshold Discrimination method, with occasional mild improvements in some of the metrics. Nevertheless, this method was not fully characterised in this work, and with more exploration, it could potentially enhance robustness to noise and baseline fluctuations, particularly for derivative thresholding. Note that Fixed Threshold Discrimination is only sensitive to noise that is above the threshold. Thus, performing the Pyramid Approximation Preprocessing before one of the derivative threshold would have a more significant effect.

Evaluation on real datasets from multiple instruments further confirms that derivative-based approaches provide more stable and accurate estimates of the muon lifetime, particularly under varying experimental conditions. The results also highlight the importance of data-driven amplitude threshold selection and careful parameter tuning for achieving reliable performance across different detector technologies. Comparing the fitted muon lifetimes between amplitude and derivative thresholding methods suggests that a two-to-four-fold improvement in count rate is possible for the same level of data distortion due to peak overlap. This could potentially reduce data acquisition times by a similar factor for count-limited μSR experiments.

Future work will focus on developing systematic and automated approaches to parameter selection, for example through Bayesian optimisation, to improve reproducibility and adaptability across experimental conditions. In addition, embedding function fitting methods within the event detection pipeline as a complementary step for other peak-finding methods presents a promising direction. While previ-

ously excluded due to computational cost, advances in optimisation and hardware may enable localised or hybrid fitting approaches that improve accuracy, particularly in the presence of overlapping pulses and low signal-to-noise ratios. Work will now be undertaken to test some of these approaches in a prototyping environment to ensure that they can be translated to the production environment to overcome the limitations identified here.

Overall, the results demonstrate that combining signal-processing techniques with physically motivated models provides a robust framework for event identification in muon detector systems, and offers a viable pathway toward scalable, high-throughput analysis for next-generation instruments such as Super-MuSR.

Acknowledgements

We would like to thank Daniel Nixon, Franz Lang, John Wilkinson, and Alex Louat for helpful discussions and comments. We would like also to thank Richard Waite for developing the Smoothed 2nd Derivative Thresholding Discrimination method.

References

[1] A. Hillier, S. J. Blundell, I. McKenzie, I. Umegaki, L. Shu, J. A. Wright, T. Prokscha, F. Bert, K. Shimomura, A. Berlie, H. Alberto, and I. Watanabe, "Muon spin spectroscopy," *Nature Reviews Primer*, vol. 2, no. 4, pp. 2662–8449, 2019.

[2] P. J. Baker, J. S. Lord, P. K. Biswas, and A. D. Hillier, "Super-musr scientific design: Progress towards a step-change in muon capabilities at isis," *J. Phys.: Conf. Ser.*, vol. 2462, p. 012032, 2023.

[3] D. E. Pooley, A. Abba, F. A. Akeroyd, P. J. Baker, A. Boston, A. Brooks, S. P. Cottrell, F. Caponio, C. Evans, and S. Franklin, "Development of a digitising data acquisition pipeline for next generation pulsed muon spectrometers," *J. Phys.: Conf. Ser.*, vol. 2462, p. 012028, 2023.

[4] G. F. Knoll, *Radiation detection and measurement*. John Wiley & Sons, 2010.

[5] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright *et al.*, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

[6] L. H. Negri, "Peakutils: Python library for peak detection," 2016. [Online]. Available: https://peakutils.readthedocs.io/en/latest/

[7] G. Cowan, *Statistical Data Analysis*, ser. Oxford science publications. Clarendon Press, 1998. [Online]. Available: https://books.google.co.uk/books?id=ff8ZyW0nlJAC

[8] O. Arnold, J.-C. Bilheux, J. Borreguero, A. Buts, S. I. Campbell, L. Chapon, M. Doucet, N. Draper, R. F. Leal, M. Gigg *et al.*, "Mantid—data analysis and

visualization package for neutron scattering and μ sr experiments," *Nuclear instruments and methods in physics research section a: accelerators, spectrometers, detectors and associated equipment*, vol. 764, pp. 156–166, 2014.

[9] W. Mattar and N. Sharon, "Pyramid transform of manifold data via subdivision operators," *IMA Journal of Numerical Analysis*, vol. 43, no. 1, pp. 387–413, 2023.

[10] N. Wiener, *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*. The MIT press, 1949.

[11] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[12] S. Tanimoto and T. Pavlidis, "A hierarchical data structure for picture processing," *Computer graphics and image processing*, vol. 4, no. 2, pp. 104–119, 1975.

[13] P. J. Burt and E. H. Adelson, "The laplacian pyramid as a compact image code," in *Readings in computer vision*. Elsevier, 1987, pp. 671–679.

[14] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.

[15] M. Mariscotti, "A method for automatic identification of peaks in the presence of background and its application to spectrum analysis," *Nuclear Instruments and Methods*, vol. 50, no. 2, pp. 309–320, 1967.

[16] S. Mallat, *A wavelet tour of signal processing*. Elsevier, 1999.

[17] P. Du, W. A. Kibbe, and S. M. Lin, "Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching," *bioinformatics*, vol. 22, no. 17, pp. 2059–2065, 2006.

[18] J. M. Gregoire, D. Dale, and R. B. Van Dover, "A wavelet transform algorithm for peak detection and application to powder x-ray diffraction data," *Review of Scientific Instruments*, vol. 82, no. 1, 2011.

[19] B. Shustin and J. Fowkes, "Event identification algorithms for muon detector traces," 2026. [Online]. Available: https://github.com/ralna/trace-fitting.git

A Additional Details for the Pyramid Approximation

The multiscale representation of a signal is constructed using a recursive subdivision-based decomposition. This produces a hierarchy of coarse approximations and detail coefficients [9], analogous to a Laplacian pyramid [13].

Let $A^{(0)}$ denote the original detector trace. At each level ℓ, the signal is decomposed into:

- a coarse approximation $A^{(\ell+1)}$ (downsampled and filtered version), and
- detail coefficients $D^{(\ell)}$ capturing the information lost during coarsening.

The decomposition consists of three main steps:

1. **Decimation (coarse approximation):** The signal is downsampled by a factor of two and convolved with a reconstruction filter γ, producing a coarse representation:
$$A^{(\ell+1)} = \gamma * \downarrow_2 A^{(\ell)}.$$
2. **Subdivision (prediction):** The coarse signal is upsampled and convolved with a subdivision mask α to reconstruct an approximation of the original signal:
$$\tilde{A}^{(\ell)} = \alpha * \uparrow_2 A^{(\ell+1)}.$$
3. **Detail extraction:** The detail coefficients are defined as the difference between the original signal and its prediction:
$$D^{(\ell)} = A^{(\ell)} - \tilde{A}^{(\ell)}.$$

This process is applied recursively to the coarse approximation $A^{(\ell+1)}$, generating a pyramid:
$$\{A^{(L)}, D^{(L-1)}, D^{(L-2)}, \ldots, D^{(0)}\},$$
where L is the number of layers.

In this implementation, all filtering operations are performed using centered convolution to preserve alignment with the original trace. The subdivision mask α determines the interpolation properties (e.g. B-spline or interpolatory schemes), while the decimation filter γ is constructed as an approximate inverse of the even part of α in the Fourier domain.

This multiscale representation separates detector signals into components corresponding to different characteristic scales, enabling targeted denoising and feature enhancement prior to event detection.