2012-05-03

Data Analysis and Double Pulse Detection for the MARE Experiment

Jonathan D. Armstrong
University of Miami, j.armstrong3@umiami.edu

UNIVERSITY OF MIAMI

DATA ANALYSIS AND DOUBLE PULSE DETECTION FOR THE MARE EXPERIMENT

By

Jonathan D. Armstrong

A DISSERTATION

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Coral Gables, Florida

May 2012

UNIVERSITY OF MIAMI

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

DATA ANALYSIS AND DOUBLE PULSE DETECTION FOR THE MARE EXPERIMENT

Jonathan D. Armstrong

Approved:

_____ Massimiliano Galeazzi, Ph.D. Professor of Physics	_____ Terri A. Scandura, Ph.D. Dean of the Graduate School
_____ Joshua Gundersen, Ph.D. Professor of Physics	_____ Thomas Curtright, Ph.D. Professor of Physics
_____ Tarek Saab, Ph.D. Professor of Physics University of Florida	

ARMSTRONG, JONATHAN D. (Ph.D., Physics)

Data Analysis and Double Pulse Detection For the MARE Experiment (May 2012)

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Professor Massimiliano Galeazzi.
No. of pages in text. (162)

With their existence first proposed in 1930, neutrinos have subsequently proven themselves as experts at avoiding detection. Until early this century, it was not even known if neutrinos were massive particles. With the results of neutrino oscillation experiments such as Super-Kamiokande and SNO, we now know that neutrinos have a non-zero mass. However, these experiments are only sensitive to the difference of the square of neutrino mass eigenstates and do not provide sufficient information to resolve the neutrino mass hierarchy. Several complimentary methods are being explored to obtain an absolute mass scale, but the most promising model-independent approach is high precision spectroscopy of the β-spectrum endpoint (Q-value). In general, all energy from the decay is detected except for that of the neutrino, which results in a correction near the endpoint of the spectrum that is related to the neutrino rest mass. To detect this difference requires excellent energy resolution. This may be obtained by utilizing a scalable approach consisting of microcalorimeter arrays with the β-decay source embedded in the absorber. Two such experiments, Troitsk and Mainz have been able to set an upper limit of 2.3 eV on the neutrino mass, but higher precision is needed. MARE (Microcalorimeter Arrays for a Rhenium Experiment) is the successor to these experiments and plans to obtain resolution in the sub-eV range.

Using an analysis program developed at the University of Miami, we have been able to verify the creation of holmium-163 which has a higher activity than rehenium-187. A landmark in the MARE project, this higher activity can provide better statistics and reduces the live time and array size requirements for a given sensitivity. One of the primary limits on the sensitivity of the MARE project, related to the source activity, is the pile-up spectrum, which is the result of unresolved double pulses. We have developed a platform to explore the efficiency of different algorithms at detecting these difficult to resolve double pulses. Using this platform, we characterize the efficiency of two different algorithms (one of which we developed for this exact purpose). The resulting analysis demonstrates that it is possible to remove a significant fraction of these events with minimal false positives. By utilizing these algorithms, MARE will be able to achieve improved sensitivity yielding a higher precision neutrino mass value.

To my family and loved ones

Acknowledgements

I would like to thank everyone who made it possible for me to pursue my passions in physics and to ultimately complete my Ph.D. at the University of Miami.

My advisor, Dr. Galeazzi, has guided my research since my undergraduate work at the University of Miami. From his patience and time grew my understanding.

I would never be where I am right now without the superhuman love and support I received from my parents. Thank you for the piano lessons, the origami paper, the fencing lessons, the countless hours invested in my education, and everything else that has made me the well-rounded, well-adjusted, and happy person I am today.

Thanks to my soon to be wife, Riesa and her family for taking me in with open arms and for all the love they have given me. My life is so much happier and richer because of them.

TABLE OF CONTENTS

List of Figures viii

List of Tables xi

1 Introduction to Neutrinos and Their Detection 1

1.1 First Detection 1

1.2 Neutrino Flavors and the Standard Model 2

1.3 Sources of Neutrino Radiation 4

1.4 Neutrino Interactions 6

1.5 The Solar Neutrino Problem and Neutrino Oscillations 8

1.6 Experimental Verification of Neutrino Oscillation 12

2 Absolute Neutrino Mass Experiments 15

2.1 Overview 15

2.2 Cosmological Methods 16

2.3 Measurement of the Neutrinoless Double Beta Decay 17

2.4 Measurement of the β/EC Endpoint 19

3 Microcalorimeters Array for a Rhenium Experiment 22

3.1 Background 22

3.2 Microcalorimeters 22

3.3 MARE Systematic Limitations 26

3.4 Rhenium-187 28

3.5 Holmium-163 32

4 Analysis Program 36

4.1.1 Extracting Pulse Energy 36

4.1.2 Optimum Filter 37

4.1.3 Optimum Filter in Practice 39

4.2.1 Analysis Program (FITSFILTER) 40

4.2.2 Preprocessing 41

4.2.3 Threshold Selection (Setting Limits) 46

4.2.4 Sweep Classification 51

4.2.5 Double Pulse Analysis 52

4.2.6 Creation of the Optimum Filter/Energy Extraction 54

4.2.7 Energy Spectrum Corrections 54

4.2.8 Energy Spectrum 56

5 Verification of Holmium Source 57

5.1 Background 57

5.2 Holmium Data Analysis 58

6 Double Pulse Detection 64

6.1 Background 64

6.2.1 Initial Results 68

6.2.2 Experimental Setup 71

6.2.3 Standard (XQC) Algorithm 73

6.2.4 Optimum Filter (OF) Algorithm 76

6.2.5 Results 80

6.3.1 Current Results, Experimental Setup 83

6.3.2 Optimum Filter Modifications 84

6.3.3 Updated Algorithms 91

6.3.4 Results, Effects of Double Pulse Contamination 94

6.3.5 Results: Effects of Detector Non-linearity 97

6.3.6 Results: Combining Both Algorithms 100

6.4 Summary 102

Appendix

A. FITS File Details 103

B. FITSFILTER, Technical Details 121

C. Contour Plots from Analysis of Chapter 6 141

References 161

List of Figures

1.1 Standard model 3

1.2 Neutrino mass hierarchy 12

2.1 $0\upsilon\beta\beta$ expected spectrum 19

3.1 Microcalorimeter model 23

3.2 Microcalorimeter readout and response 25

3.3 MARE rhenium track analytic sensitivity 31

3.4 MARE holmium track roadmap 32

3.5 Theoretical ^{163}Ho EC-spectrum 33

3.6 MARE holmium track analytic sensitivity 35

4.1 Optimum filter 38

4.2 Pulse regions 41

4.3 Pre-trigger region and pulse energy 42

4.4 Rise-time region 43

4.5 Pulse max, min, and raw energy 45

4.6 Noise limit 47

4.7 Pre-trigger voltage limit 48

4.8 Pre-trigger variation limit 49

4.9 Software trigger channel limit 50

4.10 Scatter plot 51

4.11 Sample double pulse 52

4.12 Algorithm example 53

4.13 Energy corrections 55

5.1 Holmium-163 analysis, regions of interest 58
5.2 Regions 1 and 2 sample pulse 60
5.3 Regions 3 and 4 sample pulse 61
5.4 Holmium-163 analysis, region 2 closeup 62
5.5 Holmium-163 verification 63
6.1 Pileup simulations 65
6.2 Pileup PDF examples 67
6.3 Pileup CDF examples 68
6.4 Detector nonlinearity 70
6.5 Standard algorithm 75
6.6 Optimum filter algorithm 79
6.7 Optimum filter algorithm cont. 80
6.8 2 μs contour plots 81
6.9 1 μs contour plots 82
6.10 Optimum filter improvement determination 85
6.11 Trigger channel spread 87
6.12 Optimum filter template and pileup spread 88
6.13 Optimum filter distribution sample 90
6.14 Double pulse contamination – 5 eV 95
6.15 Double pulse contamination – 2 eV 96
6.16 Nonlinear detector effects on efficiency 98
6.17 Nonlinear detector effects on false positives 99
6.18 Combined algorithm efficiency and false positive 101

(Appendix C contour plots)

C.1 Clean sets at 1μs pulse separation with 5 eV energy resolution 143

C.2 Clean sets at 2μs pulse separation with 5 eV energy resolution 144

C.3 Clean sets at 5μs pulse separation with 5 eV energy resolution 145

C.4 Dirty sets at 1μs pulse separation with 5 eV energy resolution 146

C.5 Dirty sets at 2μs pulse separation with 5 eV energy resolution 147

C.6 Dirty sets at 5μs pulse separation with 5 eV energy resolution 148

C.7 Clean sets at 1μs pulse separation with 2 eV energy resolution 149

C.8 Clean sets at 2μs pulse separation with 2 eV energy resolution 150

C.9 Clean sets at 5μs pulse separation with 2 eV energy resolution 151

C.10 Dirty sets at 1μs pulse separation with 2 eV energy resolution 152

C.11 Dirty sets at 2μs pulse separation with 2 eV energy resolution 153

C.12 Dirty sets at 5μs pulse separation with 2 eV energy resolution 154

C.13 Combined, 5 eV energy resolution, 1μs pulse separation 155

C.14 Combined, 5 eV energy resolution, 2μs pulse separation 156

C.15 Combined, 5 eV energy resolution, 5μs pulse separation 157

C.16 Combined, 2 eV energy resolution, 1μs pulse separation 158

C.17 Combined, 2 eV energy resolution, 2μs pulse separation 159

C.18 Combined, 2 eV energy resolution, 5μs pulse separation 160

List of Tables

1.1 Current neutrino mass parameters 11

2.1 Isotope half lives and Q-values 20

6.1 Double pulse combinations used 72

A.1 Sweep file format 109

A.2 PUL file format 112

A.3 FIL file format 118

B.1 FFT accounting 126

Chapter 1

Introduction to Neutrinos and Their Detections

1.1 First Detection

The existence of neutrinos was first postulated by Wolfgang Pauli in 1930 in order to reconcile experimental results with the current model of beta decay. Without including the neutrino, a single energy peak is expected for the beta spectrum (proportional to the difference of the masses of the parent and daughter particles). However, experimental results were in disagreement with this model and demonstrated a continuous spectrum with energies up to the expected peak. The existence of a new particle that possesses no charge and that caries energy undetected out of the detector system was plausible, but it took 20 years until its existence was actually verified. In 1950, Fred Reines and Clyde Cowan took on the challenge and set out to detect neutrinos in what is now known as the Cowan-Reines neutrino experiment. The detection method took advantage of the inverse beta decay reaction:

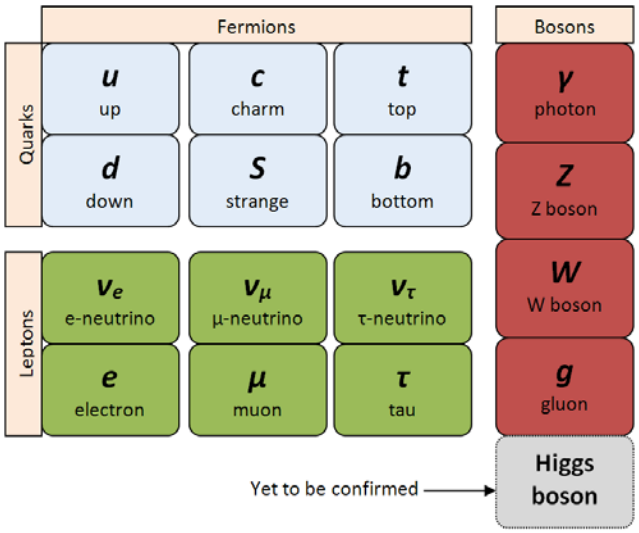
$$p^+ + \bar{\nu}_e = n^0 + e^+ \tag{1.1}$$

where a proton and antineutrino interact to form a neutron and a positron. The positron quickly reacts with an electron annihilating each other resulting in the detectable release of two gamma rays.

1.2 Neutrino Flavors and the Standard Model

With the discovery of muons in 1937 ultimately came the discovery of muon neutrinos via observations of muonic decay. More recently, in 2000, the existence of tau neutrinos was confirmed by the DONUT (Direct Observation of the NU Tau) experiment.[1] It is generally believed that there are at least three different types of neutrinos. The three experimentally verified neutrinos are each associated with a charged lepton and interact via the weak nuclear force. It is possible that there are additional neutrino flavors that are not coupled to any charged leptons. These are known as sterile neutrinos.

In the same sense that we consider atoms to be the building blocks of matter, the "elements" of the standard model are the building blocks of atoms (as well as the carriers of force, excluding gravity). The standard model, finalized in the 1960's, has been proven to be an invaluable tool for particle physicists. Many of the particles were initially theoretical and were only later experimentally verified, with The Higgs boson as the last remaining unverified element.



(Fig. 1.1) *Standard model elementary particles and gauge bosons.*

The standard model consists of groupings of fermions and bosons along with their respective antiparticles (some of which are their own antiparticle). The fermions are grouped into three columns, with the first column representing the first generation etc. Higher generations (the second and third) are unstable except in very high energy environments and rapidly decay into their first generational counterparts. Neutrinos are the exception to this. As a result, most of the matter we are familiar with is some combination of first generation fermions. The fermions can further be divided into generational pairs, or doublets, whose absolute total charge difference is equivalent to the absolute charge of an electron. For the quark pairs (u, d), (c, s), and (t, b), the first element possesses fractional charge $2/3$ and the second element possesses fractional charge $-1/3$. With the possible exception of very high energy environments (early big bang) quarks do not exist independently, but rather are bound together in composite structures (baryons corresponding to three quark combinations, and mesons corresponding to quark-antiquark pairs). For example, the neutron consists of one up and

two down quarks (udd) and thus has zero net charge and a proton consists of two up and one down quarks (uud) and has a net charge of 1. For the lepton pairs (ν_e, e), (ν_μ, μ), and (ν_τ, τ) the first element is a neutrino and has no charge, and the second elements is the associated charged lepton. Each generation of leptons can be referred to by their flavor, designated by the name of the associated charged particle. For example, the second generation neutrino has muon flavor.

The bosons are the force carriers for the electromagnetic, strong, and week force. If two given fermions can interact via a certain boson, then their interactions are characterized by the associated force. For example, fermions that interact via exchange of photons are interacting electromagnetically. In the case of neutrinos, the Z and W bosons, collectively mediating the electroweak force, are of interest. More specifically we want to know the types of interactions that produce neutrinos and the ways in which the resulting neutrinos can interact and thus be detected.

1.3 Sources of Neutrino Radiation

- **Electron Neutrino (ν_e)**

Electron neutrinos can be produced from nuclear reactions such as β-decay or electron capture. β-decay can further be divided into β^- and β^+-decay. In a fundamental sense, β^--decay is the decay of a down quark to an up quark:

$$d(du) \rightarrow u(du) + W^-; \tag{1.2}$$

$$W^- \rightarrow e^- + \bar{\nu}_e\ . \tag{1.3}$$

The neutron (ddu) decays into a proton (udu) and emits a W^- boson. The emitted W^- boson quickly decays into an electron and an anti-electron neutrino pair. This reaction occurs naturally for radioactive elements.

The β^+-decay on the other hand requires energy input to convert a proton to a neutron:

$$u(du) + Energy \rightarrow d(ud) + W^+; \quad (1.4)$$

$$W^+ \rightarrow e^+ + \nu_e. \quad (1.5)$$

Again, the emitted W^+ boson (now with positive unit charge) decays into a particle antiparticle lepton pair such that electrical charge is conserved.

The electron capture reaction occurs for proton rich elements and is the result of an inner core proton absorbing an inner core electron. Again, this reaction is mediated by a W boson and has two interpretations that are equally valid depending on whether the proton or the electron emits the W boson. The net effect is:

$$u(du) + e^- \rightarrow d(du) + \nu_e. \quad (1.6)$$

- **Muon Neutrino (ν_μ)**

Muons are not naturally created by nuclear processes because the energy required to produce one is insufficient. They may however be formed as the result of high energy cosmic radiation colliding with atmospheric particles (or similarly in a particle accelerator). The resulting particle shower is known as atmospheric radiation. As stated previously, fermions in the second generation are unstable and ultimately decay in first

generation particles. There are various possible decay modes, but the most probable result is the emission of neutrinos (depending on the muon's charge):

$$\mu^- \rightarrow \nu_\mu + W^- \rightarrow \nu_\mu + e^- + \bar{\nu}_e; \quad (1.7)$$

$$\mu^+ \rightarrow \bar{\nu}_\mu + W^+ \rightarrow \bar{\nu}_\mu + e^+ + \nu_e. \quad (1.8)$$

- **Tau Neutrino (ν_τ)**

Similar to the case of the muon neutrino, the tau neutrino is the result of a tau particle decay. These particles require even more energy than what is required to produce a muon and as a result have many available decay modes, some of which produce tau neutrinos (as well as the other two types). Unique to the charged leptons, tau particles have sufficient energy that they may occasionally decay into a hadron (composite quark particle). We are not too concerned with the specific modes of decay resulting in tau neutrinos and will consider it sufficient to note that they are the result of the tau particle decay.

1.4 Neutrino Interactions

Understanding neutrino interactions is important because they ultimately provide us with potential detection channels. Neutrinos possess no charge, are massive (as we will show soon) but very light, and only interact via the weak nuclear force. Because of this, they are extremely difficult to detect and tend to require massive detectors with long live-times. We are restricted to observing them through interactions that ultimately result

in a detectable charged particle or the nuclear metamorphosis of elements. These can be divided into two types of reactions which are (1) the result of electron scattering, and (2) the result of nucleon absorption.

Electron scattering is mediated by the electroweak bosons (W & Z), but the net effect is an exchange of momentum between the neutrino and an electron previously bound to a molecule/atom. This exchange is possible between any of the three neutrinos (and their antiparticles), but more likely for the electron flavor. In the case of an electron neutrino, the reaction is simply written as:

$$e^- + \nu_e \rightarrow e^- + \nu_e. \tag{1.9}$$

The newly freed electrons tend to have relativistic velocities, which when properly taken advantage of, allows detection.

There are different ways a nucleon may absorb a neutrino, some of which completely absorb the neutrino while the others re-emit it, known respectively as a charged current interaction (CC) or a neutral current interaction (NC). CC interactions are mediated by the charged W boson and have different energy thresholds depending on which flavor of neutrino is involved. One potential reaction may be written as:

$$\nu_\alpha + d(du) \rightarrow u(du) + l_\alpha, \tag{1.10}$$

where $\alpha = \{e, \mu, \tau\}$ and l_α is the associated charged lepton. In most cases, the emitted lepton has insufficient energy to produce radiation, but for a sufficiently pure medium, the modified elements may be counted in order to determine the integral neutrino flux that occurred over the detector live-time. Another potential reaction, involving only electron neutrinos, is the CC inverse beta decay:

$$p^+ + \bar{\nu}_e \rightarrow n^0 + e^+. \tag{1.11}$$

By taking advantage of reactions involving one or both of the resulting daughter particles, the electron neutrino may be detected

NC reactions are coupled with the electrically neutral Z boson and only involve an energy transfer from the neutrino to the nucleon. This reaction takes place at the same rate regardless of neutrino flavor. In the case of a heavy water nucleus (deutron), this interaction breaks up the nucleon into its constituent proton and neutron:

$$\nu_\alpha + D \rightarrow p^+ + n^0 + \nu_\alpha, \qquad (1.12)$$

with the same convention as above and with D representing the heavy water nucleus.

1.5 The Solar Neutrino Problem and Neutrino Oscillations

In the 1960's, the science of detecting neutrinos had advanced to the point where detectors were able to resolve neutrinos emitted from the solar core. Results showed neutrino fluxes that were one third to one half of what was expected based on calculation using the standard solar model. This discrepancy between theory and experimental results is known as the solar neutrino problem. Its resolution required revision of our understanding of neutrinos or of the solar model (or possibly both)! At the time, the physics of solar processes was much better understood than that of neutrinos and it was ultimately shown through experimentation that it was our understanding of neutrinos that was incomplete (and still is). The solution to the solar neutrino problem lies in a phenomenon known as flavor oscillations.

Simply stated, neutrino flavor oscillations refer to the ability of neutrinos emitted as one flavor to be detected as another. This is a purely quantum mechanical effect that

results from the flavor eigenstates (ν_e, ν_μ, ν_τ) not directly coinciding with the mass eigenstates (ν_1, ν_2, ν_3). This only occurs if neutrinos are massive (have a nonzero mass). Each flavor eigenstate can be represented as a linear combination of the different mass eigenstates resulting in a 3×3 unitary matrix. The probability of a neutrino emitted as one flavor and being detected as another flavor is of particular interest. The calculation of this probability for neutrinos traveling in a vacuum is outlined in the paper, "Neutrino Mass, Mixing and Flavor Change",[2] and yields the result:

$$P(\nu_\alpha \to \nu_\beta) = \delta_{\alpha\beta} - 4\sum_{i>j} \mathcal{R}\left(U^*_{\alpha i} U_{\beta i} U_{\alpha j} U^*_{\beta j}\right) \sin^2\left[1.27\Delta m^2_{ij}\left(\frac{L}{E}\right)\right] + \quad (1.13)$$

$$2\sum_{i>j} \mathcal{I}\left(U^*_{\alpha i} U_{\beta i} U_{\alpha j} U^*_{\beta j}\right) \sin^2\left[2.54\Delta m^2_{ij}\left(\frac{L}{E}\right)\right],$$

where $\Delta m^2_{ij} \equiv m^2_i - m^2_j$ is in eV^2; L, the distance the neutrino traveled from emission to detection (in the lab frame) is in km; E, the energy of the neutrino (in the lab frame) is in and GeV; $\mathcal{R}(*)$ and $\mathcal{I}(*)$ are the real and imaginary functions respectively; and $U_{\alpha i}$ is an element from the 3×3 mixing matrix mentioned above. The most important aspects of eq. 1.13 to note are its dependence on the distance traveled, its energy, the difference of the squares of its masses, and the mixing matrix. All of these dependencies must be considered when calculating the expected neutrino flux. In particular, if a detector is only sensitive to electron neutrinos, it will miss any neutrinos that have decayed to other flavors. This result must further be modified to account for neutrinos traveling through dense matter (such as the solar core). In this regime, the oscillation probabilities are altered by what is known as the Mikheyev-Smirnov-Wolfenstein effect and is the result of coherent scattering of the neutrinos.[2] Taken together these effects would later explain the observed flux deficit.

The 3×3 flavor mixing matrix is typically parameterized as follows:

$$U = \begin{pmatrix} C_{12}C_{13} & S_{12}C_{13} & S_{13}e^{-i\delta} \\ -s_{12}c_{23} - c_{12}s_{23}s_{13}e^{i\delta} & s_{12}c_{23} - s_{12}s_{23}s_{13}e^{i\delta} & s_{23}c_{13} \\ s_{12}s_{23} - c_{12}c_{23}s_{13}e^{i\delta} & c_{12}s_{23} - c_{12}c_{23}s_{13}e^{i\delta} & c_{23}c_{13} \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & e^{i\emptyset_1} & 0 \\ 0 & 0 & e^{i\emptyset_2} \end{pmatrix}, \quad (1.14)$$

where c_{ij} and s_{ij} represent $\sin(\theta_{ij})$ and $\cos(\theta_{ij})$ respectively for the mixing angles between the mas eigenstates of the same index and the rows and columns represent the neutrino flavor and mass eigenstates respectively. The three complex phases are related to CP-violation and do not affect neutrino oscillations.

In cases where one flavor transition dominates, it is possible to simplify the situation to oscillations between one neutrino flavor eigenstate and a quasi-two-neutrino (a linear combination of the remaining two flavor eigenstates). The resulting mixing matrix (ignoring irrelevant phase factors), becomes:

$$\begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\cos(\theta) & \cos(\theta) \end{pmatrix}. \quad (1.15)$$

In this regime, the analog of equation 1.13 gives:

$$P(\nu_\alpha \rightarrow \nu_\beta) = \sin^2(2\theta)\sin^2[1.27\Delta m^2(L/E)] \quad (1.16)$$

With $\beta \neq \alpha$, and

$$P(\nu_\alpha \rightarrow \nu_\alpha) = 1 - \sin^2(2\theta)\sin^2[1.27\Delta m^2(L/E)] \quad (1.17)$$

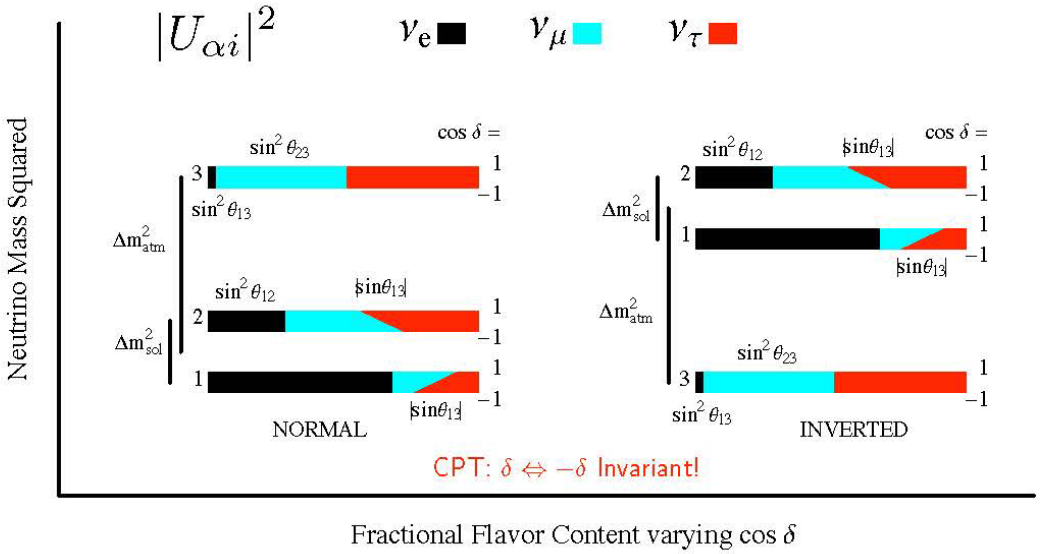
with the same conventions as used in eq. 1.13. Many neutrino experiments are parameterized in this way. In the case of atmospheric neutrino oscillations experiments, $P(\nu_\mu \rightarrow \nu_e)$ (based on null results from short baseline neutrino reactor experiments)[2] is very small so that the quasi-two-neutrino is approximately a tau neutrino. Thus, these experiments try to obtain values for $\theta_{atm} = \theta_{23}$ and $\Delta m^2_{atm} = \Delta m^2_{32}$. In the case of solar

neutrinos, we are mainly concerned with the probability of an electron neutrino oscillating to a different flavor. The quasi-two-neutrino in this case is a linear combination of tau and muon neutrinos. These experiments try to obtain values for $\theta_{\odot} = \theta_{13}$ and $\Delta m^2_{\odot} = \Delta m^2_{31}$. Currently, the most accurate results for these parameters are given in the following table:

$\sin^2(2\theta_{12}) = 0.861^{+0.026}_{-0.022}$	$\Delta m^2_{21} = (7.59 \pm 0.21) \times 10^{-5} eV^2$
$\sin^2(2\theta_{23}) > 0.92$	$\lvert\Delta m^2_{32}\rvert = (2.43 \pm 0.13) \times 10^{-3} eV^2$
$\sin^2(2\theta_{13}) < 0.15\ (C.L\ 90\%)$	$\Delta m^2_{31} = ?$

(Table 1.1)[3] *Summary of current neutrino parameter values.*

Until the neutrino mass eigenstates are resolved, there remains ambiguity as to their relative hierarchy. Based on our current knowledge of neutrinos, there remain three possibilities. If the mean mass of υ_1 and υ_2 (associated with solar neutrinos) is less than that of t υ_3 (associated with atmospheric neutrinos), then the spectrum is said to be "normal", otherwise its hierarchy is "inverted" (see fig. 1.3). An additional possibility, known as the degenerate hierarchy occurs when the mass states are approximately equal.



(Fig 1.2)[4] *The neutrino flavor density of the normal and inverted hierarchy for the neutrino mass states (assuming conservation of CPT).*

1.6 Experimental Verification of Neutrino Oscillation

The first experimental validation of neutrino oscillations (and thus massive neutrinos) came in 1998 from the Super-Kamiokande collaboration in Japan. This experiment was directionally sensitive to muon flux from atmospheric radiation. To a reasonable approximation, the atmospheric neutrino flux is homogeneous. The angle relative to the zenith was divided into bins based on the cosine of this angle. After analyzing the data resulting from 537 detector live days, a muon neutrino flux deficit was found for upwards traveling neutrinos (through the bulk of Earth). These neutrinos traveled a sufficient distance to allow the effects of neutrino oscillations to take place which resulted in the flux deficit.[5]

The most convincing evidence for neutrino oscillations came from the Sudbury Neutrino Observatory (SNO) in Canada. Similar to other neutrino detectors, SNO consisted of a large tank buried deep underground, but varied in its use of heavy water (deuterium) as the detection medium. By taking advantage of neutrino scattering interactions, SNO was capable of detecting the three neutrino flavors. Elastic scattering reactions of neutrinos with water/deuterium may produce Cherenkov radiation (as with Super-Kamiokande) whose angular distribution is peaked in the direction of the incident neutrino. Further, the use of deuterium provided two more interactions to detect neutrons: neutral current (NC) and charged current (CC) interactions. NC interactions are the result of a neutrino separating a deuterium nucleus (deuteron) into its constituent proton and neutron. The NC reaction produces a neutron and is not sensitive to direction and may be the result of any of the three neutrino flavors. Detection occurs through the absorption of the free neutron which results in a gamma ray that generates Cherenkov radiation via Compton scattering. CC interactions are the result of interactions between an electron neutrino and a deuteron and transform a neutron into a proton emitting an electron that may be detected. By combining the three detection methods with statistical analysis, SNO was able to detect all three neutrino flavors and demonstrated that the deficit of electron neutrino flux was complimented by muon and tau electron flux.

Following these experiments, it became very well accepted that neutrinos are massive particles. However, neutrino oscillation experiments are only sensitive to the difference of neutrino masses squared and thus don't give any information on actual mass of a given neutrino mass state. In order to achieve such results, a new type of approach is

needed. In the next chapter, we explore various methods and ultimately lead into the MARE experiment which provides a direct, model independent approach.

Chapter 2

Absolute Neutrino Mass Experiments

2.1 Overview

Experiments such as Super-Kamiokande and SNO have provided evidence for massive neutrinos via verification of neutrino oscillations. Even though these experiments yield valuable insight into neutrino physics they are only sensitive to the difference of the square of neutrino mass eigenstates. To obtain measurements of the absolute mass of neutrinos, a completely different approach is required. Promising possibilities for obtaining sub-eV resolution of the neutrino mass exist in several forms, each with their own pros and cons and fall into three categories:

- Cosmological methods
- Measurements on the neutrinoless double β-decay: $0\upsilon\beta\beta$
- Measurement of the β/EC endpoint

In this chapter we briefly describe all three methods with emphasis on the last, which my research falls under.

2.2 Cosmological Methods

Cosmological methods for obtaining an absolute scale on the neutrino mass involve comparing cosmic microwave background data sets (Planck, WMAP, etc.) and galaxy surveys with models that are sensitive to neutrino densities. Such analysis is capable of placing a limit on the neutrino mass. More specifically, this method is not sensitive to neutrino flavor and only places a limit on the sum of neutrino masses:

$$m_{tot} = \sum_{i}^{N} m_i \tag{2.1}$$

where the sum over N allows for the possibility of sterile neutrinos. The current results obtained from such methods are however sensitive to model assumptions. Different assumptions can lead to results that vary by factors of 2 and 4![6,7] With this in mind, the limit of $m_{tot} < 2.5eV$ (95% confidence) was obtained using constraints set by the CMB and 2Df galaxy survey along with several other large scale astrophysical data sets.[6] As our understanding of the early universe improves and we obtain improved data, so will this limit. However other means for determining the absolute scale of the neutrino mass exist and are currently being explored.

2.3 Measurement of the Neutrinoless Double Beta Decay

In general there are many double β-decay modes possible, but the existence of a neutrinoless double beta decay ($0\upsilon\beta\beta$) is still somewhat uncertain though controversially suggested by results from the Heidelberg-Moscow experiment.[8] The $0\upsilon\beta\beta$ reaction:

$$N(A,Z) \rightarrow N(A+2,Z-2) + 2e^- \tag{2.2}$$

requires that neutrinos are their own antiparticle (Majorana as opposed to Dirac). The related double β-decay ($2\upsilon\beta\beta$) has been experimentally verified and only differs by the additional emissions of two neutrinos:

$$N(A,Z) \rightarrow N(A+2,Z-2) + 2e^- + 2\bar{\nu}_e \tag{2.3}$$

Assuming the $0\upsilon\beta\beta$ exists, the absolute value of the neutrino mass states is related to the half life of elements that undergo such decay. The half life of $0\upsilon\beta\beta$ can be very large, on the order of 10^{25} years[9] requiring large amounts of ultra pure source material with very high resolution detectors and excellent understanding of systematics. As with cosmological methods, the neutrino mass obtained from $0\upsilon\beta\beta$ experiments is a sum of the mass eigenstates. In this case, the sum is weighted with elements from the 3×3 neutrino mixing matrix (often referred to as the Pontecorvo–Maki–Nakagawa–Sakata matrix or PMNS matrix for short). The resulting value is known as the effective Majorana mass and is described by the equation:

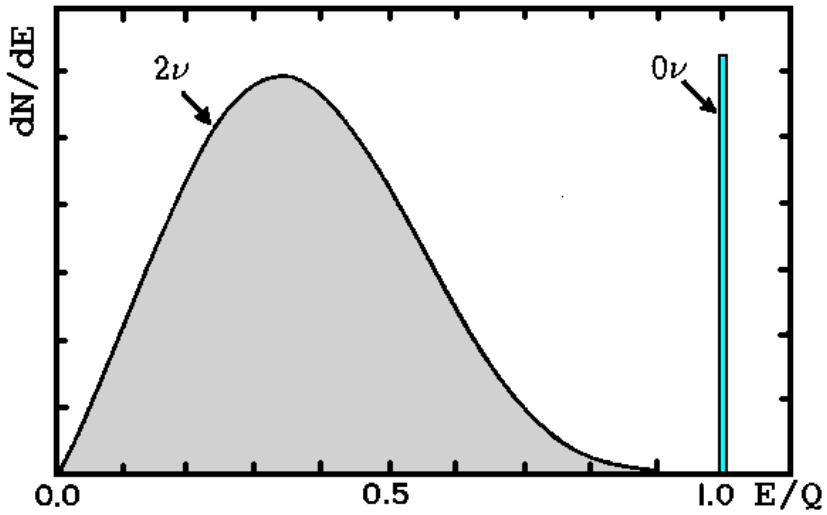
$$m_{\beta\beta} = \left| \sum_{i=1}^{3} U_{ei}^2 m_i \right| \tag{2.4}$$

where U_{ei} represents elements from the PMNS matrix. Due to the weighing of each mass term by U_{ei}^2 (more specifically from the CP phases), there is the possibility of cancelation which introduces model dependency into the results.

The half life of a potential $0\upsilon\beta\beta$ source is proportional to the effective Majorana mass described above:

$$T_{1/2}^{0\upsilon} \propto \frac{1}{\left(m_{\beta\beta}\right)^2} \tag{2.5}$$

where the constant of proportionality depends on a phase space factor and the nuclear matrix element, with the latter introducing additional model dependence. In order to obtain this half life, a calorimetric measurement may be performed with the resulting energy spectrum yielding a continuous $2\upsilon\beta\beta$ spectrum and a peak from the $0\upsilon\beta\beta$ reaction, whose width would only depend on the energy resolution (fig. 2.1). By counting events in the $0\upsilon\beta\beta$ peak, it is possible to then determine the half life of the source and thus perform a measurement on the effective Majorana mass. A successful experiment would not only allow determination of the neutrino mass hierarchy but would also verify the existence of the $0\upsilon\beta\beta$.



(Fig. 2.1)[21] *Representation of the* $0\upsilon\beta\beta$ *energy spectrum (right spike) and the* $2\upsilon\beta\beta$ *energy which has the same form as eq. 2.2 but with an additional two anti-electron neutrinos emitted.*

2.4 Measurement of the β/EC Endpoint

This class of experiments attempts to resolve the effects of massive neutrinos near the Q-value (end point) of the β/EC-spectrum. Emitted neutrinos are easily able to escape detector systems removing energy in the process. The effect this has on the observed β-spectrum is that the end point of the energy spectrum is shifted by an amount related to the rest mass of neutrinos given as:

$$m_{\nu_e}^2 = \sum_{i=1}^{3} |U_{ei}|^2 m_i^2 \tag{2.6}$$

In this case, there is no possibility for cancellation of mass terms in the sum. This means that experiments measuring $m_{\nu_e}^2$ through high precision spectroscopy are capable of determining the neutrino mass scale in a model independent way.

For such experiments, it is obviously desirable to measure the β-spectrum of a source with a high relative count rate near the Q-value. It turns out that the count rate in this area is proportional to the inverse cube of the endpoint energy. Three different isotopes are particularly favorable due to their very low endpoint energies: tritium (^3H), rhenium (^{187}Re), and holmium (^{163}Ho). Their respective half-lives and Q-values are listed in table 2.1 below.

	Half-life (years)	**Q-value (keV)**
tritium (^3H)	12.23	18.6
rhenium (^{187}Re)	4.5×10^9	2.48
holmium (^{163}Ho)	4570	2.2-2.8

(Table 2.1) *half-life and Q-value for endpoint measurement candidates.*

Tritium, with the highest decay rate was the first of the isotopes explored for neutrino mass experiments, lends itself to large scale detectors that are spatially separated from the source. Initial experiments suffered from significant systematic error due to interaction with the radiation before detection. After the systematics involved were better understood, two experiments, Troitsk and Mainz, were able to set an upper limit on the neutrino mass of 2.3 eV.[10,11] The successor to these experiments, KATRIN (Karlsruhe Tritium Neutrino experiment), plans to achieve sub-eV resolution of the neutrino mass at around 0.2 eV after three years of detector live-time. The approaches used for these experiments impose strict scaling laws for which KATRIN is the limit. This means that if

the neutrino mass (eq. 2.6) is below the resulting sensitivity achieved by KATRIN, only an upper limit will be obtained.

Due to their comparatively lower activity, the remaining two isotopes lend themselves to a different type of setup with the source embedded in the detector. This removes the systematic error problems that tritium experiments must contend with. The typical setup relies on arrays of detectors which allow for scalability of experiments. The feasibility of using Rhenium has already been demonstrated by the MANU and MIBETA experiments which were able to obtain an upper limit on the neutrino mass of about 15 eV.[12, 13] The successor to these experiments is MARE, which will be discussed in the next chapter.

Chapter 3

Microcalorimeters Arrays for a Rhenium Experiment (MARE)

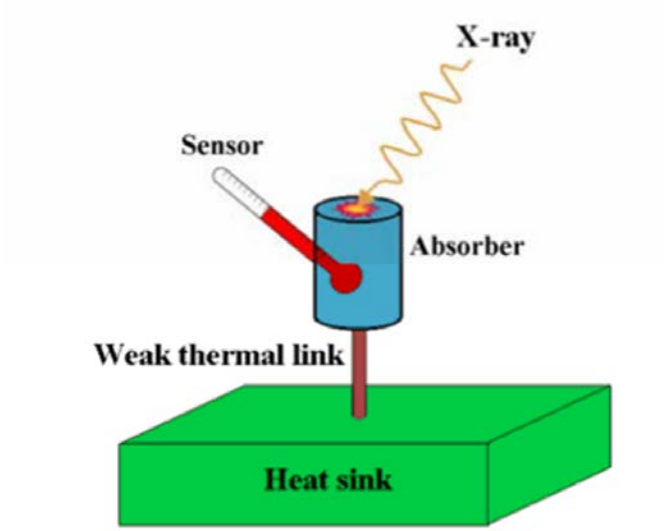
3.1 Background

The goal of MARE (Microcalorimeter Arrays for a Rhenium Experiment) is to resolve the perturbation of the β/EC endpoint caused by massive neutrinos in order to obtain the effective mass of the electron anti-neutrino and resolve the neutrino hierarchy. This experiment varies dramatically from KATRIN in its systematics but plans to achieve a similar sensitivity (around .2 eV) by using detector arrays with embedded sources. The use of detector arrays provides the scalability while the embedded source removes the spatial separation and that was the source of systematic errors in KATRIN. Despite the acronym, both holmium and rhenium isotopes are candidates for this experiment and possess different pros and cons. The feasibility of MARE is currently being explored. Once the feasibility is sufficiently demonstrated the actual experiment will begin.

3.2 Microcalorimeters

Microcalorimeters are a type of detector that take advantage of the first law of thermodynamics, using heat as a means of measuring the energy of radiation. The theory

of "ideal" microcalorimeters was developed in 1984 by Mosley, Mather, and McCammon[14] and updated in 2003 by Galeazzi and McCammon to include non-ideal effects.[15] Microcalorimeters are able to obtain the energy of a single event with very high resolution. Because of this and their scalable nature, microcalorimeters have been chosen for the MARE project. The basic model of a calorimeter is simple and consists of three main components: an absorber, a thermometer, and a heat sink (fig. 3.1). The role of the absorber, as implied by its name, is to absorb incident electromagnetic radiation (and other associated forms of energy for imbedded sources) and to convert it to thermal energy where the resulting change in temperature is proportional to the energy of the radiation. This temperature change is observed via a thermometer. The weak thermal link provides a means to dissipate the energy allowing the system to return to equilibrium.



(Fig. 3.1) *Schematic representation of a calorimeter. Note that in the case of MARE, the source is embedded in the absorber.*

Despite its apparent simplicity, in practice much care must be taken with the development of a microcalorimeter. When considering detection of single electron or x-ray (or any

other charged particle radiation) the thermal noise of the system must be very low. In order to achieve this, the detector must be maintained at a temperature in the 100 μK temperature range. By examining the well known relationship:

$$Q = C\Delta T \tag{3.1}$$

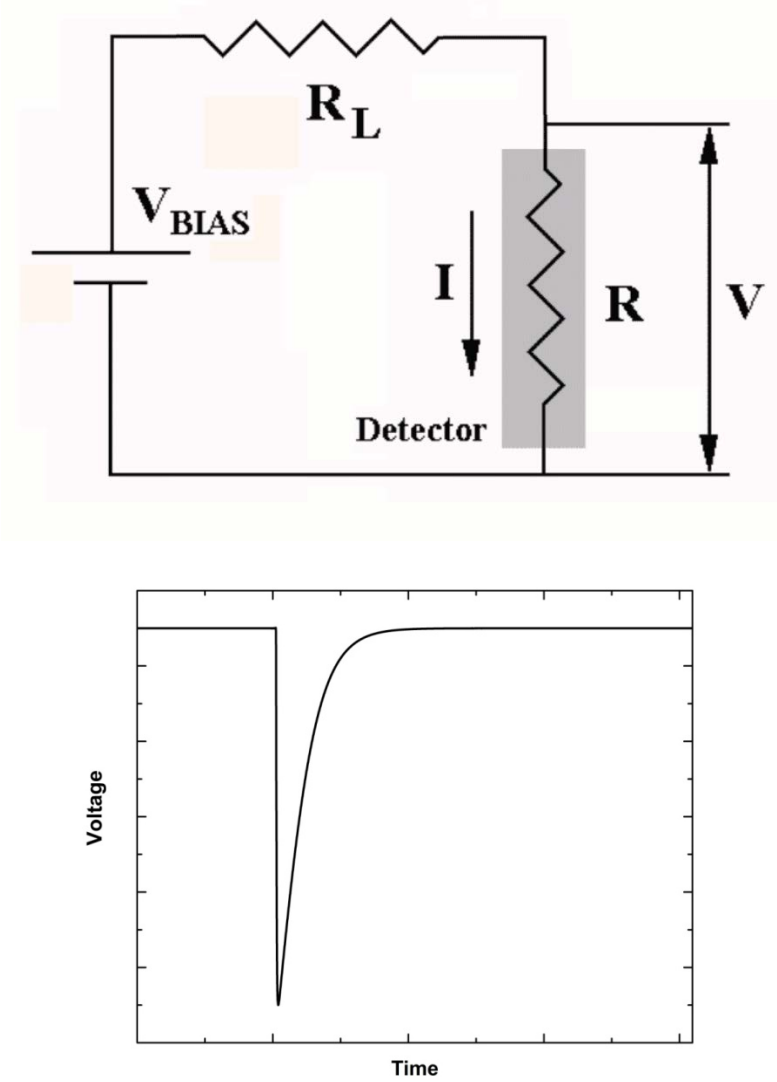
we see that in order to maximize the change of temperature for a given energy, the heat capacity must be very small. This may be achieved by using a small absorber and by carefully selecting the material used (considering its thermal properties at low temperatures).

The term thermometer broadly includes any type of sensor that is sensitive to changes in temperature. Two potential thermometers for the MARE project are magnetic and TES (Transition Edge Sensors) microcalorimeters. Magnetic microcalorimeters incorporate a paramagnet whose magnetization is inversely proportional to temperature. TES microcalorimeters are detectors biased at a temperature corresponding to the phase transition between normal and superconducting where the resistance is very sensitive to small changes in temperature. The unitless sensitivity of a detector is defined as:

$$\alpha = \frac{T}{R}\frac{dR}{dT} \tag{3.2}$$

which is clearly very large in the phase transition region for a TES microcalorimeter. Typically such detectors are amplified by a SQUID (Superconducting Quantum Interference Device); a high gain, low noise signal transducer. Similar to the working principals of a voltmeter or ammeter, the typical detector readout circuit (fig 3.2 - top) may be voltage or current biased by using a load resistor with a high or low resistance compared to that of the sensor. The response in both cases (only varying by a y-axis flip)

is a sudden change in voltage/current resulting from the absorption of an event followed by a slow decay back to equilibrium (fig. 3.2 - bottom). This shape is what we call a "pulse". The change in voltage/current for a pulse is related to the energy that will need to be extracted from each pulse (this process is described in the next chapter).



(Fi.g 3.2) *Top: tyical readout circuit for a TES sensor (voltage biased with $R_L \gg R$). Bottom: Typical response of a current biased detector ($R_L \ll R$, with the current converted to a voltage by the readout electronics).*

3.3 MARE Systematic Limitations

In order to show the feasibility of MARE, we need to understand the limits on detector sensitivity and how to mitigate them. Three primary limiting factors have been identified and are: energy resolution, pulse pileup, and detector statistics. In this section these factors are introduced with a more quantitative description given in the next sections.

- **Energy Resolution**

The noise of the detector sets the limit on energy resolution for a microcalorimeter. Detector optimization with considerations for a broad range of noise sources has been performed, but to first approximation, the Johnson and thermal (phonon) noise dominate.[16] The noise of a detector is parameterized as the noise equivalent power (NEP), which is the amount of power a detector needs to absorb to equal the noise output signal. If the NEP for a detector is known, its energy resolution is given as:

$$\Delta E_{RMS} = \frac{1}{\sqrt{\int_0^\infty \frac{2d\omega}{\pi NEP^2(\omega)}}} \tag{3.3}$$

which gives:

$$\Delta E_{RMS} \approx \sqrt{\frac{4k_B T^2 C}{\alpha}} \tag{3.4}$$

For the Johnson and thermal noise contributions, where k_B is the Boltzmann constant, T is the working temperature of the detector, C is the heat capacitance, and α is the detector sensitivity given in equation 3.2.[17] Note eq. 3.4 was derived assuming the optimum filter described above, is used. Simpler methods for extracting the pulse energy would result in decreased energy resolution.

- **Pulse Pileup**

The next limiting factor for MARE is due to the presence of pulses that occur sufficiently close together that they become unresolvable as separate events. The resulting shape resembles a single pulse with total energy roughly equal to the sum of its constituent pulses. The result of this is a false spectrum which is added to the β/EC spectrum. This pileup spectrum may be accurately calculated from the actual β/EC spectrum, but since these events are rare, there is significant variation in the experimental spectrum making it difficult to remove directly. Additionally, because of its additive nature, the pileup spectrum is shifted to higher energies relative to the actual β/EC spectrum. These two effects, despite the rarity of unresolvable pileup events, contribute significantly to decreased sensitivity because of the low statistics achieved near the endpoint of the spectrum.

The fraction of unresolvable pileup events is related to response time (rise-time) of the detector. This time, in principal, is due to the time lag caused by the rate of heat diffusion in the absorber and the thermal coupling between the absorber and thermometer. In practice, this occurs sufficiently fast that the rise-time is often set by

response time of the readout electronics. The phenomenon and detection of double pulses will be explored in more detail in chapter 6.

- **Detector Statistics**

With the source embedded directly in the detector, there is no way of filtering out unwanted energies. The fraction of total events that are near the Q-value is on the order of 10^{-5} which implies that a very large number of events are required to obtain the desired sensitivity for MARE. A reasonable balancing of the source activity, the number of detectors, and the detector acquisition time is required (with consideration for the effects of increased source activity on the pile-up spectrum). This balance is different for rhenium and holmium. A more quantitative description is given in the next two sections for the specific cases of rhenium, then holmium.

3.4 Rhenium-187

In pure rhenium, the isotope ^{187}Re occurs with a natural abundance of 62.2%. This convenience means that natural rhenium may be used without any enrichment. The β-decay reaction is given as:

$$^{187}Re \rightarrow {}^{187}Os + e^{-} + \bar{\nu}_e \tag{3.5}$$

with rhenium decaying to osmium. In order to accurately observe the perturbation caused by massive neutrinos near the Q-value, the theoretical spectrum must be well known. The available energy to the electron-neutrino system (E_0) is the energy obtained from the mass difference of the parent and daughter elements:

$$E_0/c^2 = Mass(^{187}Re) - Mass(^{187}Os) \tag{3.5}$$

Because the source is embedded, the nuclear recoil is also detected, but simple conservation principles give this energy (ignoring the neutrino) as:

$$E_R = \frac{E_\beta}{2Mc^2} \tag{3.6}$$

where M is the mass of recoiled body, c is the speed of light, and the energies correspond to that of the recoil and the energy of the emitted radiation. In the case of rhenium, for the largest emitted energy, the associated recoil energy is on the order of 10^{-9}, negligible compared to the expected energy resolution of the MARE project (~2 eV). Ignoring the recoil energy, the most general β-spectrum of emitted electrons is given as:

$$N_\beta(Z, E_\beta, m_{\nu_e}) = \tag{3.7}$$

$$\left\{p_\beta E_\beta (E_0 - E_\beta)\sqrt{(E_0 - E_\beta)^2 - m_{\nu_e}c^4}\right\} F(Z, E_\beta 0) S(E_\beta)[1 + \delta_R(Z, E_\beta)]$$

where p_β and E_β are the momentum and energy of the emitted electron; the portion in curly brackets is the phase space contribution for a three body decay (ignoring nuclear recoil); $F(Z, E_\beta 0)$ is the Fermi function correction due to the interactions between the nuclear charge and the wave function of the emitted electron; $S(E_\beta)$ is the "form factor" which accounts for electro-weak interactions; and $\delta_R(Z, E_\beta)$ is the radiative electromagnetic correction (usually neglected due to its minimal contribution). In the case

of rhenium crystals, an oscillatory correction must be applied due to interference between the electron wave and it reflections off of the regular lattice structure (analogous to the extended x-ray fine structure effect).[18]

$N_\beta\left(Z, E_\beta, m_{\nu_e}\right)$ may be integrated strictly near the endpoint of the β-spectrum to yield the approximate percentage of counts in the region of interest

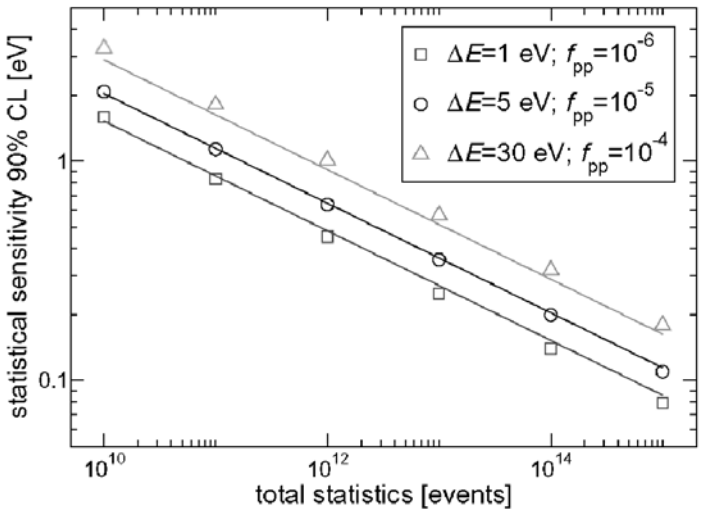
$$\left(\frac{\Delta E_0}{E_0}\right)^3 \tag{3.7}$$

where ΔE_0 is the size of the region near the endpoint being considered. A reasonable value for this would be $3m_{\nu_e}$, but is generally limited by the detectors energy resolution (on the order of sing digits). Taking ΔE_0 as 5 eV and using the Q-value of 2.48 keV for rhenium, this fraction is of the order 10^{-9}, implying the need to collect a significant number of events. However, a high activity must be balanced with the effects of the pileup spectrum on the sensitivity.

In order to optimize the MARE experiment, we need a quantitative understanding of the systematics involved. A systematic and Monte Carlo approach at determining a calorimetric neutrino mass experiments sensitivity to double pulses was explored by A. Nucciotti, O. Cremonesi and E. Ferri.[19] In their analysis, they demonstrated that even a very small population (~0.01-0.0001% of all events – see fig. 3.3 below) can have significant effects on the experimental sensitivity, which demonstrates the importance of detecting pileup events for the MARE project. In their paper, they derive an equation for the statistical sensitivity (at 90% confidence level) of determining m_{ν_e} in a calorimetric experiment as:

$$\sum(m_\nu)_{90} = 1.13\left(\frac{E_0^3\,\Delta\mathrm{E}}{A_\beta t_M N_{det}} + 0.3\frac{\tau_R E_0^5}{t_M N_{det}\Delta\mathrm{E}}\right)^{1/4} \tag{3.8}$$

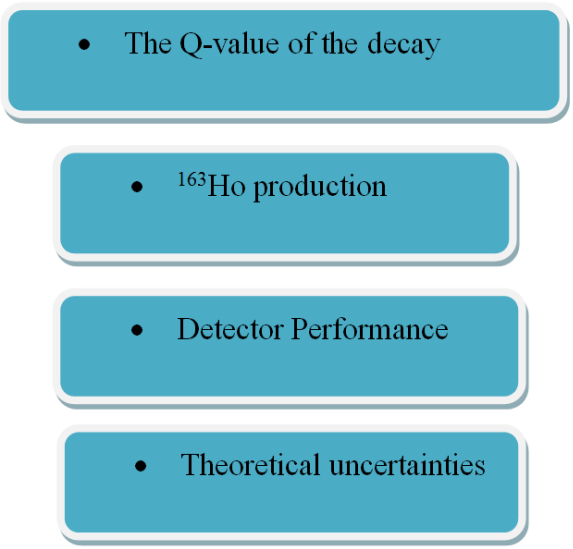
where E_0 is the endpoint energy; ΔE is the interval below the end point energy being considered; t_M is the measuring time; τ_R is the detector rise-time; A_β is the single detector source activity; and N_{det} is the total number of detectors. In general a smaller ΔE is desirable, but use restricted by the detector energy resolution. Because of the dependence on $N_{det}^{-1/4}$ we can get a sense of how many detectors are required for a given sensitivity. The dependence of the sensitivity on τ_R is related to the effect of rise-time double pulses, which are very difficult to detect and provide one of the biggest sources of error to calculations of m_{ν_e}.



(Fig. 3.3) *Sensitivity of a calorimetric experiment as a function of total statistics for three different pileup fractions.*

3.5 Holmium-163

The holmium isotope, ^{163}Ho, does not exist naturally in nature and thus required production. This added complication is offset by the higher activity which has the potential to improve the statistics of MARE (lower live-time and fewer arrays required for a given sensitivity). A roadmap for the holmium track of MARE has been outlined with four landmarks:



(Fig. 3.4) *MARE holmium track roadmap landmarks.*

The sensitivity a ^{163}Ho experiment depends strongly on the Q-value. To understand why this is the case, we need to obtain the EC-spectrum. The EC reaction:

$$^{163}Ho + e^{-} \rightarrow {}^{163}Ds + \nu_e \tag{3.9}$$

is caused by the absorption of an inner shell electron in the nucleus. The resulting electron hole is filled by an outer shell electron which emits an x-ray in the process. The decay rate is thus a sum over the possible shells of the electron capture:

$$\lambda_{EC} = \frac{{G_\beta}^2}{4\pi^2}\sum_i n_i\, C_i \beta_i^2 B_i (Q - E_i)[(Q - E_i)^2 - {m_{\nu_e}}^2]^{1/2} \tag{3.10}$$

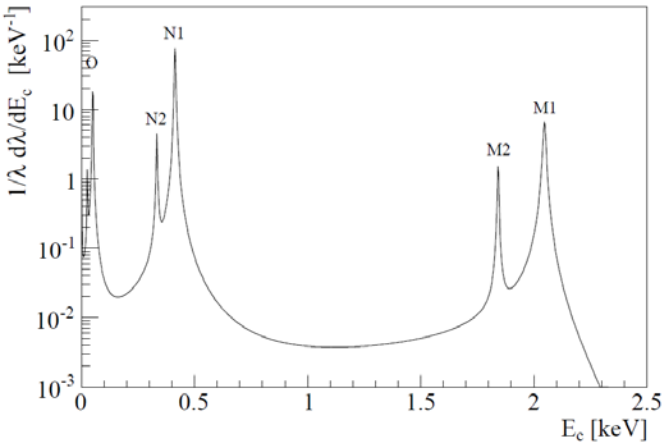
where $G_\beta = G_F \cos(\theta_c)$ is the fraction of occupancy of the i-th atomic shell; C_i is the atomic shape factor; β_i is the Coulomb amplitude of the electron radial wave; and B_i is the atomic correction for the electron exchange and overlap.[20]

From this, the expected energy distribution detectable by a microcalorimeter is given as:

$$\frac{\lambda_{EC}}{dE_c} = \frac{{G_\beta}^2}{4\pi^2}(Q - E_C)\left[(Q - E_C)^2 - {m_{\nu_e}}^2\right]^{\frac{1}{2}} \times \tag{3.11}$$

$$\sum_i n_i\, C_i \beta_i^2 B_i \frac{\Gamma_i}{2\pi} \frac{1}{(E_C - E_i)^2 + \Gamma_i^2/4}$$

where E_c is the energy detected by the calorimeter.[20] The resulting theoretical spectrum is represented in fig. 3.5.



(Fig 3.5)[20] *Theoretical ^{163}HoEC spectrum. The resonance peaks allow for self calibration which is another advantage of using a holmium source.*

Typically, EC reactions have a poor count rate near their Q-value. However, in the case of ^{163}Ho, the count rate in this region is amplified by the M peak. To better understand the potential sensitivity of a calorimetric ^{163}Ho experiment, an accurate measurement of

the Q-value is required. A straightforward approach using a single microcalorimeter with about 5×10^5 counts is sufficient to determine the Q-value to better than 20 eV accuracy.[20]

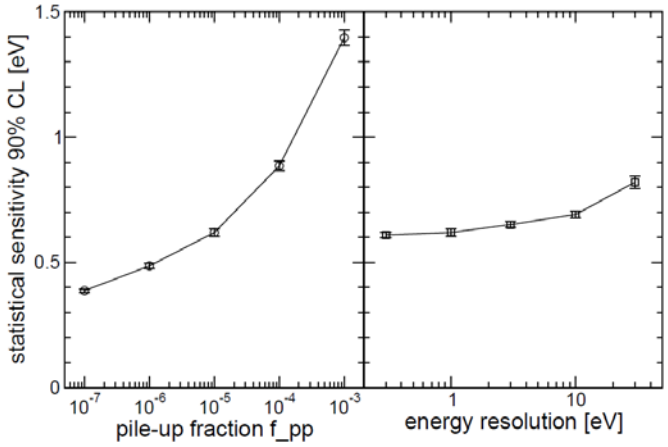
The second landmark for the holmium track is the production of the isotope. There are several different methods that exist for producing holmium of which we identify three ideal candidates:[20]

- Neutron activation in a nuclear reactor of ^{162}Er
- α-particle bombardment of ^{165}Ho target
- γ-particle bombardment of ^{165}Ho target

A reliable production of ^{163}Ho is required with minimal long-lived contamination, and with consideration of ease of production and reproducibility. This requires investigation which is in the preliminary stages. Verification of the production method using ^{162}Er has been confirmed by the University of Genoa and the University of Miami (this will be described in more detail in ch. 5).

Alongside identifying a reliable production method for ^{163}Ho, the detector performance must be explored. There are many variables, though many are similar in nature to the rhenium track. One difference arises from the way the source is used in the detector. Unlike rhenium, where the source is the absorber, holmium is implanted in an absorber. With a comparatively higher count rate there is more freedom in selecting the capacitance of the absorber and the activity of the source. This aspect is currently being explored at the University of Miami. Additionally, just as with rhenium, the effects of pileup spectrum affect the potential sensitivity of this experiment (fig. 3.5). Methods to

reduce the occurrence of these events are thus fundamental to improving the potential sensitivity of this experiment.



(Fig. 3.6) [20] *Effects of pileup fraction and detector energy resolution on the sensitivity of a calorimetric holmium experiment.*

The final landmark for the holmium track is to better understand theoretical uncertainties. The bulk of these arise from the derivation of the EC-spectrum. There are still atomic and nuclear aspects which are not completely understood. To maintain this model independence, additional effects that alter the EC spectrum must be identified an implemented in an improved version of eq. 3.11.

Chapter 4

Analysis Program

4.1.1 Extracting Pulse Energy

In order to obtain a high resolution energy spectrum, it is fundamentally important to extract the energy of a pulse as accurately as possible. In principal, this can be done by just measuring the change in voltage/current, but the accuracy of this method can strongly be affected by the presence of noise. An improvement can be achieved by integrating the signal, but such an operation has the negative feature that it treats all regions of the pulse equally, particularly the baseline which may be noisy in comparison to the pulse amplitude. Clearly the presence of noise has a negative effect of our ability to extract the energy of a pulse. Because of this, a method that is capable of limiting the noise contribution is desirable. This can be done by modifying the integration method with the incorporation of a weighting function. This method is called optimum filtering. All that needs to be determined then is the weighting function.

4.1.2 Optimum Filter

In the linear regime, the pulse shape of a calorimetric measurement is independent of the energy of the absorbed radiation, where that energy is simply a scaling factor:

$$V(t) = E_0 g(t) \tag{4.1}$$

We must also account for the noise, which we assume has a known spectral density: $e_n(f)$. If we assume that the noise spectrum is not altered in the presence of a pulse, which in most cases is a reasonable assumption, it follows that the noise amplitude in different frequency bins is uncorrelated. To take advantage of this assumption, we perform the derivation of our optimum filter in the frequency domain. We will let s_i and n_i respectively denote the i^{th} frequency bin of the (noise-free) signal and the noise. Following from the properties of Fourier transforms, each frequency bin of s_i is proportional to E_0 and thus allows an independent estimation of the pulse's energy.

In our case, we can identify that we want to maximize the signal to noise ratio. Thus considering a weighting function given by w_i, we may write the energy and noise fluctuations as sums over the frequency bins:

$$E = \sum_i w_i s_i \tag{4.2}$$

$$\Delta E_{rms} = \left(\sum_i |w_i n_i|^2 \right)^{1/2} \tag{4.3}$$

The ratio of these sums is what we would like to optimize. By taking the derivative of their ratio with respect to some arbitrary w_k and setting the result equal to zero, we can

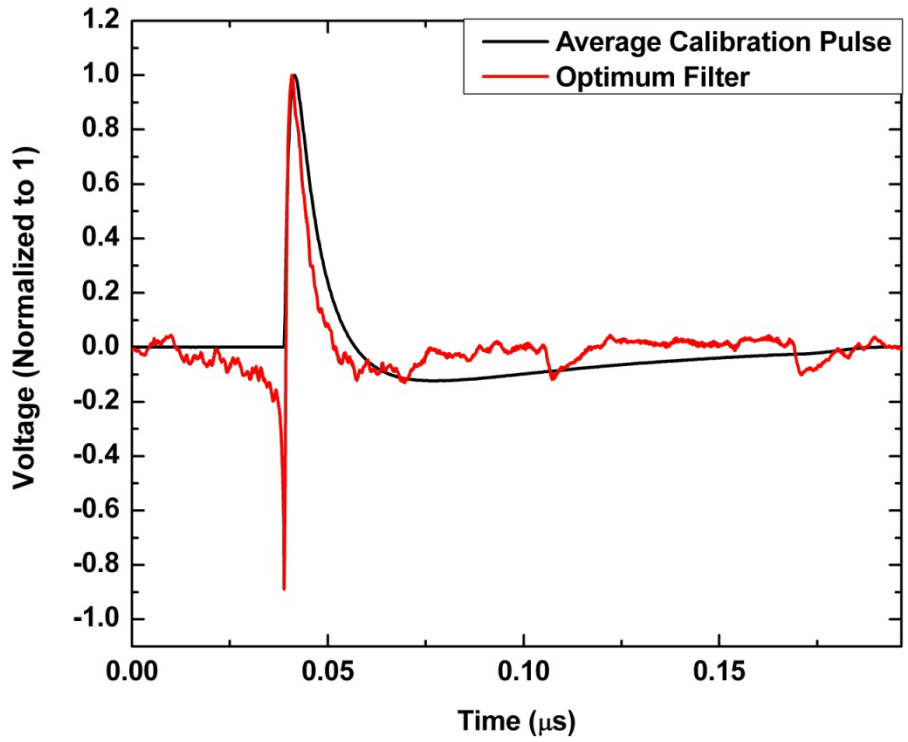
solve for w_k and thus solve for the optimum filter weighting function (in the frequency domain). The result obtained is:

$$w_k = \frac{s_k}{|n_k|^2}\left(\frac{(\Delta E_{rms})^2}{E}\right) \tag{4.4}$$

Since the factor in parentheses is constant for any given element, we simply drop it (scaling will be accounted for later). Thus, the optimum filter in the frequency domain is:

$$w_i = \frac{s_i}{|n_i|^2} \tag{4.5}$$

This function may be converted back to the time domain via a Fourier transform. The relative phase between the weighting function and the pulse may be out of phase, so a convolution can be performed to locate the local maximum of the weighted integral.



(Fig. 4.1) *Example of a pulse and a weighting function for an optimum filter.*

4.1.3 Optimum Filter in Practice

There are two ingredients required to produce the optimum filter: a noiseless pulse and the noise spectrum. A perfectly noiseless pulse is impossible to obtain from experimental data, but a reasonable approximation may be obtained by averaging a sufficiently large number of pulses. The relative signal to noise ratio is better for higher energy pulses, so the best results may be obtained by using a subset of all of the pulses with higher energies. Typically, this subset corresponds to calibration pulses which from a source with known energy (the K-α line from ^{55}Fe with an energy of about 5.9 keV is a very common calibration source). The noise spectrum is obtained by averaging noise samples in the frequency domain. With these two ingredients, we may now build the optimum filter and extract the energy of the pulses.

We still need to determine the scaling factor to obtain the actual energy spectrum. This requires calibration pulses with a known energy. Using the optimum filter created above along with the average of the calibration pulses (to minimize the noise contribution), we have all the ingredients required to find this scaling factor. By relating the energy of the calibration sweeps to their weighted integral we obtain:

$$E_{cal} = A \sum_i V_{i,cal} W_i \tag{4.6}$$

which we can solve for the scaling factor, A, where W_i is the time domain weighting function of the optimum filter, $V_{i,cal}$ is the averaged calibration pulses, and E_{cal} is the energy of the calibration pulses. The optimum filtered energy for any pulse can then be calculated as:

$$E_{sweep} = \left(\frac{E_{cal}}{\sum_i V_{i,cal} W_i}\right) \sum_i V_{i,pulse} W_i = A \sum_i V_{i,pulse} W_i \tag{4.7}$$

These extracted energies may then be used to create an energy spectrum.

4.2.1 Analysis Program (FITSFILTER)

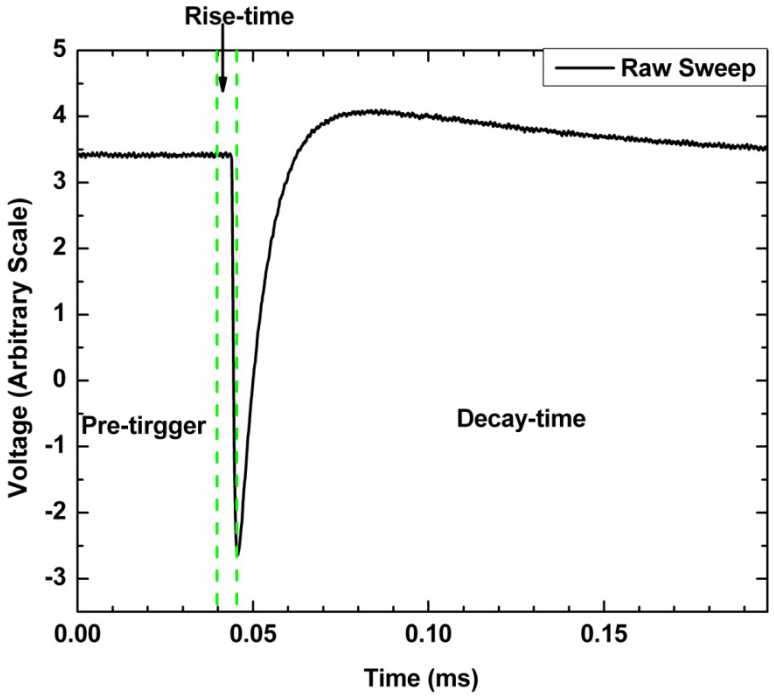
We have identified that it is important to be able to identify undesirable events (e.g. pileup events) and to use an optimum filter to extract pulse energies. The next logical step is to develop an analysis program which performs these functions. Such a program was developed at the University of Miami in conjunction with the XQC (X-ray Quantum Calorimeter) team headed by Dan McCammon. The analysis program was dubbed FITSFILTER due to its use of the FITS file format and incorporation of the optimum filtering technique (more information on this file format is located in the appendix). Through development, we focused on maintaining good portability across Linux systems and overall usability.

The input for FITSFILTER consist of collections of what we call sweeps. Sweeps are the results of converting a continuous data stream into smaller pieces (the sweeps) which contain potentially relevant data. This is done via a triggering mechanism which identifies events above a threshold then saves the local region. To include noise in the sweeps (required to construct the optimum filter) a random trigger is also employed. The desirable sweeps contain single events or flat baseline noise. It is important to reject sweeps that were incorrectly triggered by glitches in the electronics, noise spikes, or any other method.

FITSFILTER is built around subroutines which perform all of the analysis required to reject undesirable sweeps, build the optimum filter, extract the pulse energies, and plot the results. The subroutines are sequential requiring the analysis of the previous ones. The steps in the analysis chain and their purpose are described in the next sections.

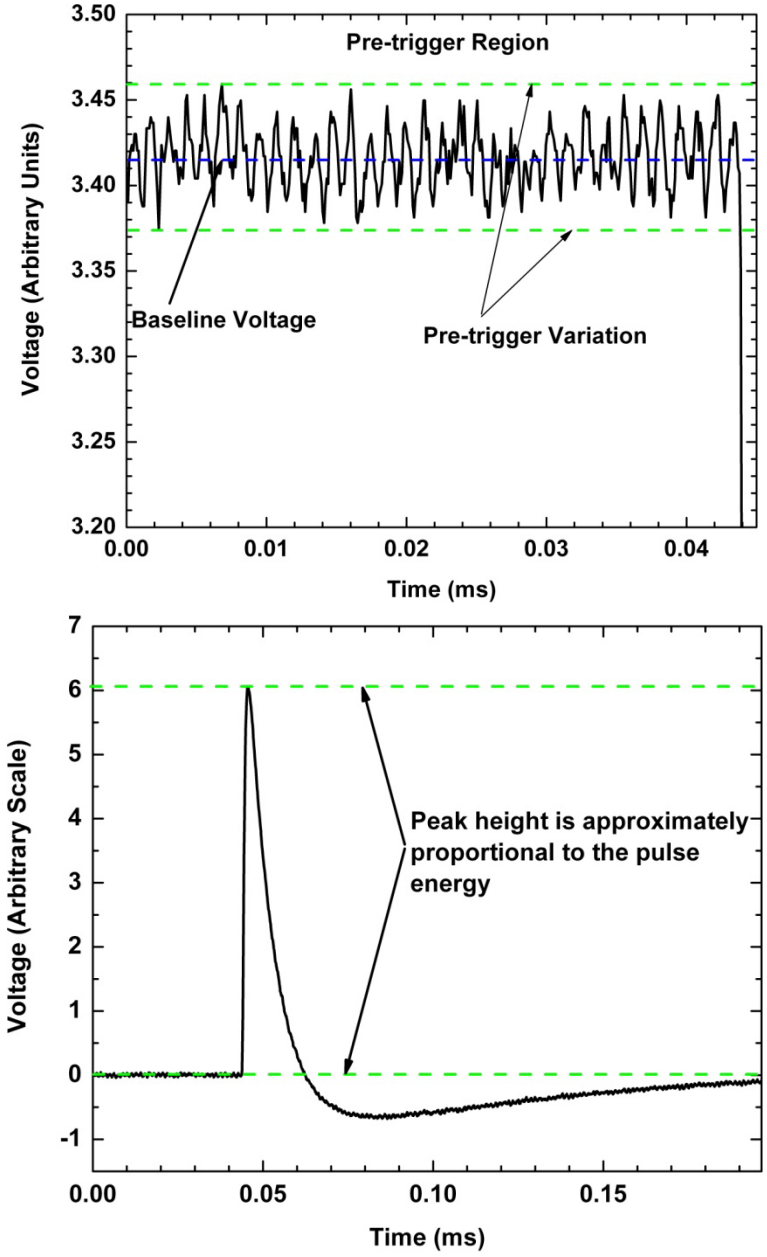
4.2.2 Preprocessing

The first step in the analysis chain is preprocessing of the data sweeps. In general, a sweep is divided into three regions: pre-trigger, rise-time, and decay-time (see fig. 4.1). These three qualitatively different regions are used to extract characteristic parameters from each sweep.



(Fig. 4.2) *Sub- regions of a raw sweep. Note that because the detector is voltage biased, the pulse is "upside down". Also, there is some distortion of the decay time region due to the response of the amplifier used (in general it decays smoothly back to the baseline and does not overshoot it). Because the energy scale will be calibrated in a later step, we are not concerned with the actual scale, only its proportionality.*

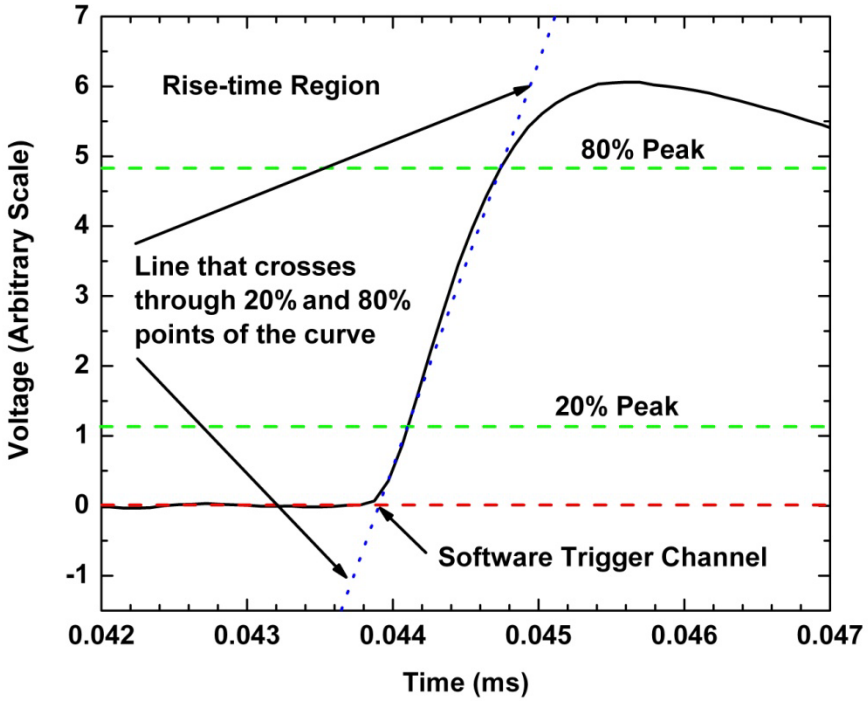
In the pre-trigger region, the baseline voltage and variation are calculated for each sweep (fig. 4.3 - top). The baseline voltage is usually removed from each sweep so that the voltage of their maxima is the difference in voltage (fig. 4.3 - bottom). The baseline corresponds to the DC level and variation corresponds to the noise.



(Fig. 4.3) *Top: visual of parameters extracted from the pre-trigger region of a sweep. Bottom: Raw sweep with its baseline voltage removed and inverted so that its maximum voltage also relates to its energy.*

In the event that a sweep is triggered on the tail of another pulse the baseline will be skewed. Because the extraction of pulse energies involves an integral over the entire sweep, this will modify the energy of the pulse that actually triggered the sweep. These parameters allow identification and rejection of such events.

The rise-time is calculated in the rise-time region along with the software trigger (fig. 4.4). The calculations for these parameters are rough but sufficient for our purposes. The rise-time is based on the time constant for an exponential curve and is reasonably independent of the pulse amplitude. This is calculated from the line that crosses through the first points within the region corresponding to 20 and 80% of the pulses maximum. The software trigger calculated as the x-intercept of this line.



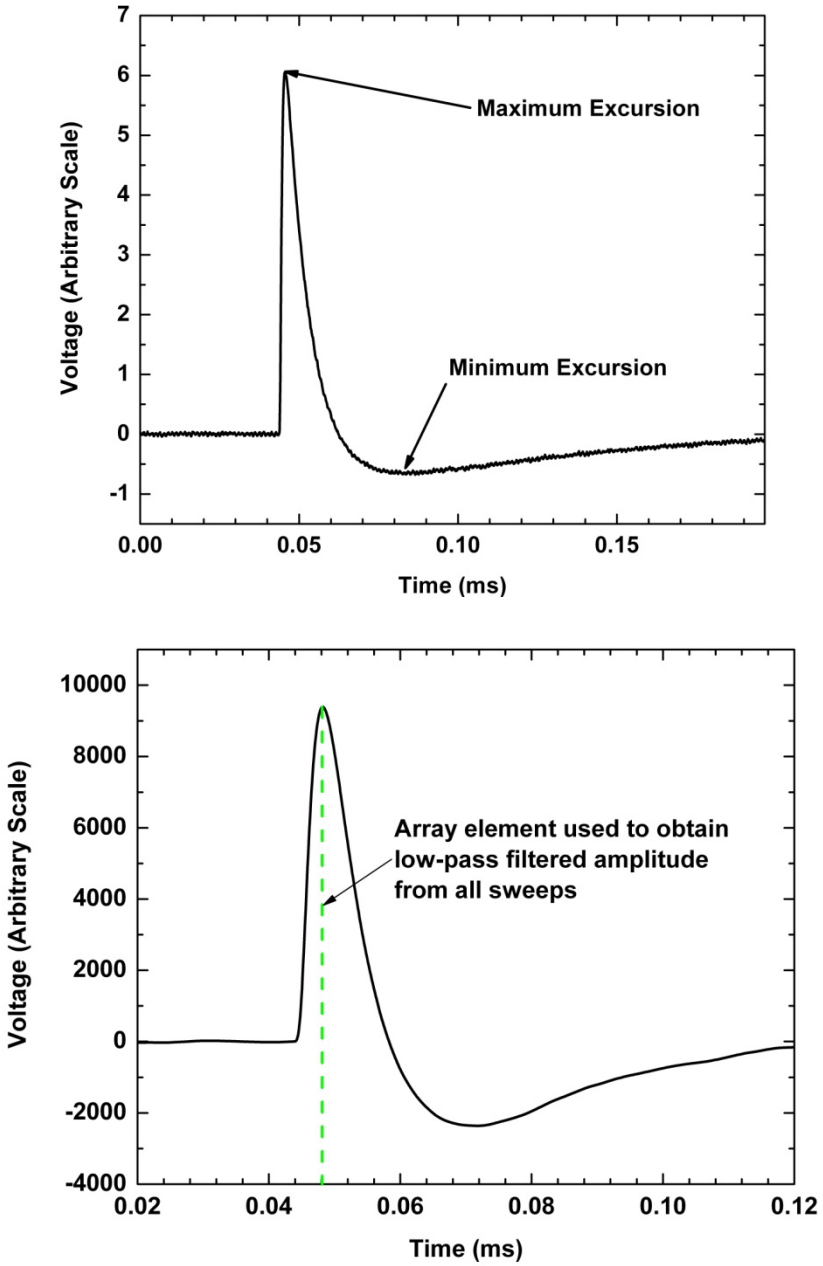
(Fig. 4.4) *Rise-time region of a pulse with the baseline removed and inverted. A line is built based on the points of the curve that are approximately at 20 and 80% of the pulse height. The intercept of this line with zero gives the software trigger.*

One of the most important parameters, the rise-time is related to the absorption of energy in the absorber. Strong variations from the norm of this value indicate problems with the absorption of radiation or other related issues and indicate regions of interest which are qualitatively different. The software trigger channel identifies the start of a pulse which allows them to be properly lined up for averaging and comparison throughout the analysis process. As with the rise-time, the decay time is calculated in the decay-time region and proceeds very similarly as above to obtain a value (but with no software trigger channel analog).

In addition to these parameters, the maximum and minimum excursions are calculated for an entire sweep (fig. 4.4). These values can come from any array element where the maximum/minimum occurs. The minimum generally should be similar to a noise excursion but if the signal is high-pass filtered (generally as a result of the readout electronics used) it may be large due to the dip caused by the amplifier response. The maximum amplitude allows us to distinguish between sweeps with noise and sweeps with pulses. The minimum amplitude is not as important but may identify undesirable negative noise spikes.

The most important parameter collected is a raw un-scaled approximation of the pulse energy (the low-pass filtered amplitude). Unlike the previous parameters this value is taken from the same array element of each sweep. Because of the triggering mechanism, in well behaved sweeps, all of the peaks should be commensurate yielding a parameter that is a good approximation of the pulses energy. In order to minimize the effects of noise, the sweeps are first low-pass filtered before this parameter is saved. Because it provides a good estimate for the energy, this provides important information

about every sweep and can be used to construct a rough energy spectrum (up to a scale factor). Once all of the parameters have been extracted from sweeps we continue to the next step

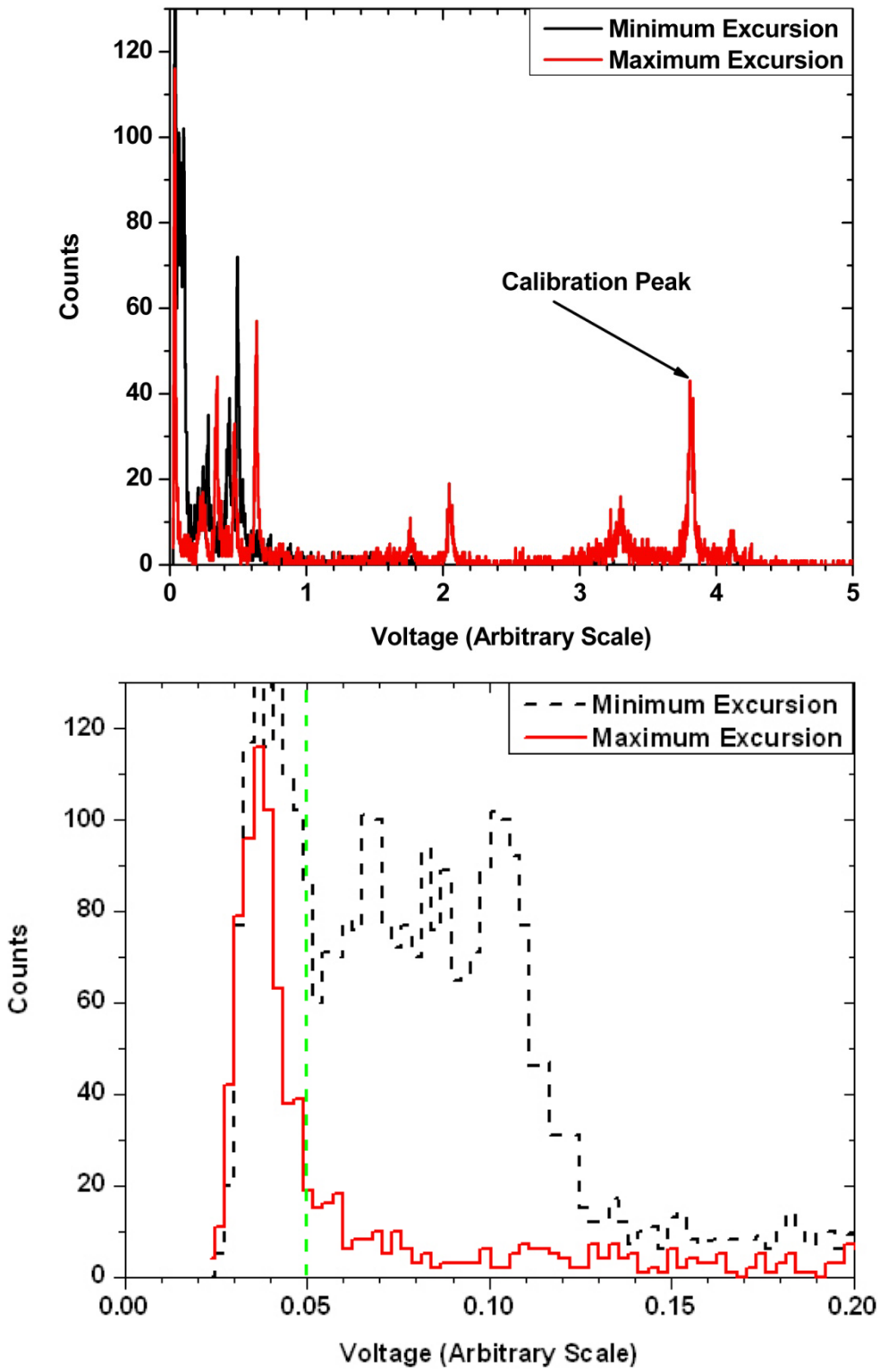


(Fig. 4.5) *Top: Example of maximum and minimum excursion for a pulse. Bottom: The average of all low-pass filtered sweeps (pole = 400Hz) used to select the array element that will determine the low-pass amplitude. This value plays the role of a first order approximation for the un-scaled energy.*

4.2.3 Threshold Selection (Setting Limits)

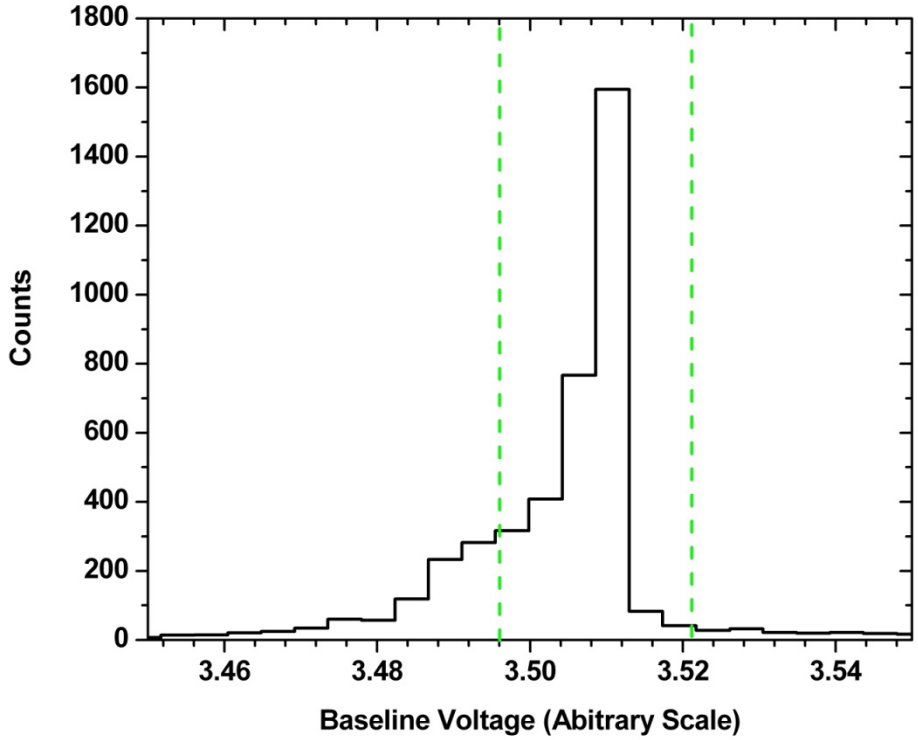
The second step in the analysis chain is to set limits on the previously acquired parameters. For reasonably well behaved data sets (and even for many unruly sets), the parameters tend to be clustered together with outliers corresponding to some sort of problem. By identifying the clustered regions, thresholds can be set that allow the rejection of bad sweeps.

The first histograms created are built from the maximum and minimum excursion parameters. Since these values are comparatively lower for noise sweeps than for sweeps with a pulse they allow us to distinguish between the two cases. The histogram derived from the maximum excursions is a rough estimate of the un-scaled energy spectrum and allows identifications of spectral features such as a calibration peak (fig. 4.6 - top). The other histogram's "spectrum" is a result of the decay-time overshoot and is roughly a compressed version of the real spectrum.



(Fig 4.6) *Histogram of maximum and Minimum excursions. The top plot shows the entire spectrum where the calibration peak from* ^{55}Fe *can be seen. The bottom plot shows the threshold set on noise.*

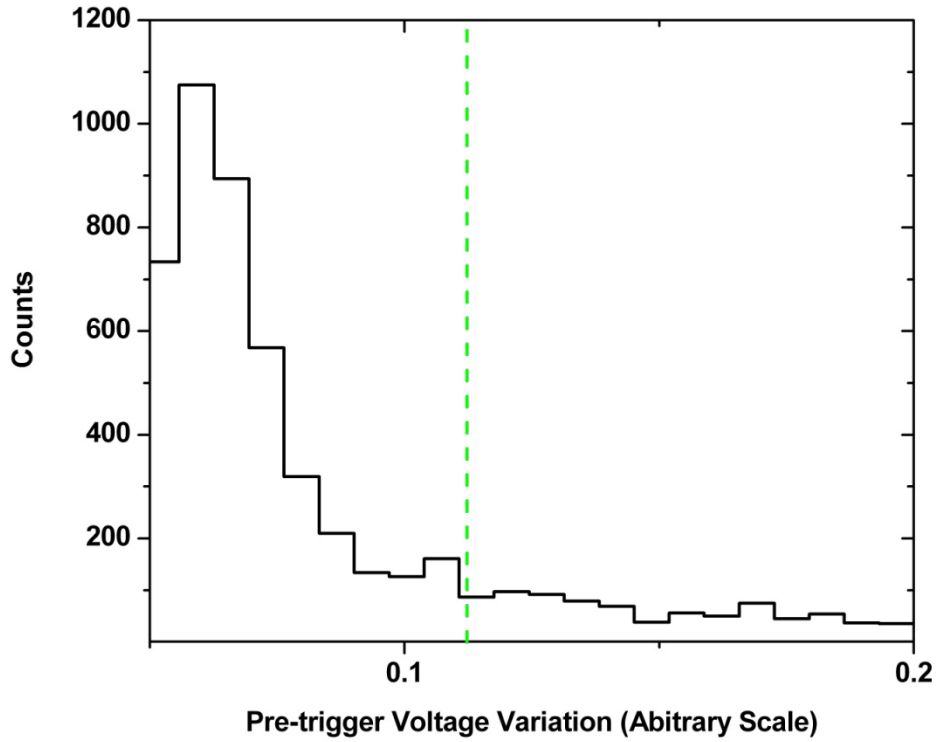
The next limit set is on the average pre-trigger voltage which establishes the acceptable baseline voltage (fig. 4.7). Assuming no problems with the detector this value should be fairly constant. This set of data had a fairly high activity which means there are many double and even triple pulses. Because of this many sweeps are on the end of a pulse resulting in a bad baseline. Because of the decay-time overshoot, there is a tendency for bad events to have a lower value. This can be seen by the fatter left side of the histogram. Noise spikes also have the potential to widen the main feature. This possibility is accounted for with the next limit.



(Fig. 4.7) *Limits set on the acceptable pre-trigger voltage (pulse baseline); everything outside of the green lines is rejected.*

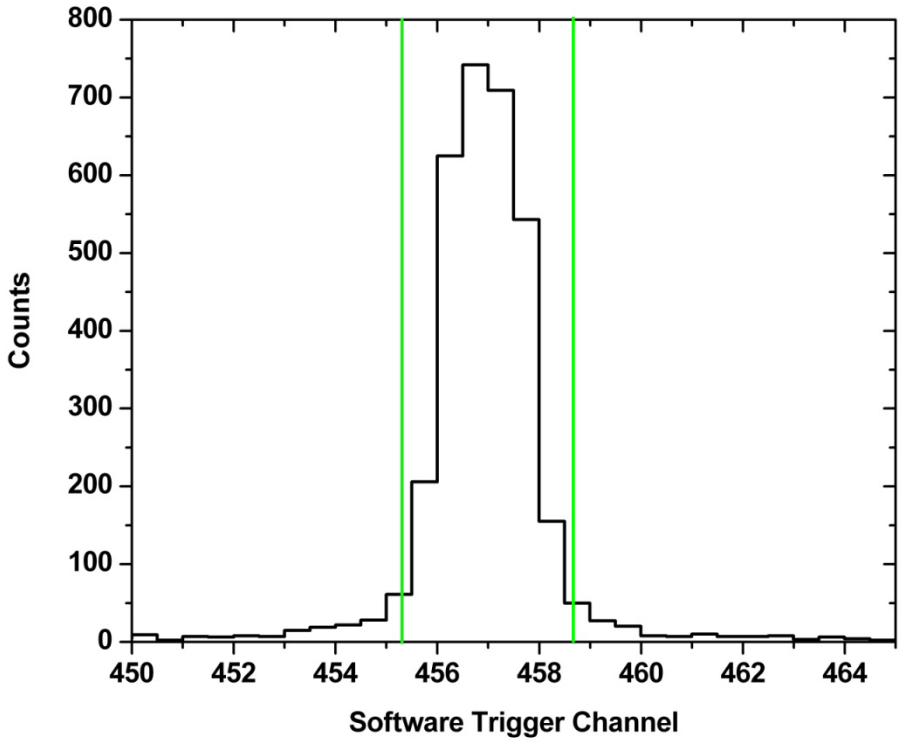
Next, we look for any abnormal features of the baseline voltages. This is done by calculating the maximum and minimum excursions (restricted to the pre-trigger region),

and taking their difference (fig. 4.8). For good sweeps, it is consistently small compared to sweeps with problems in the pre-trigger region.



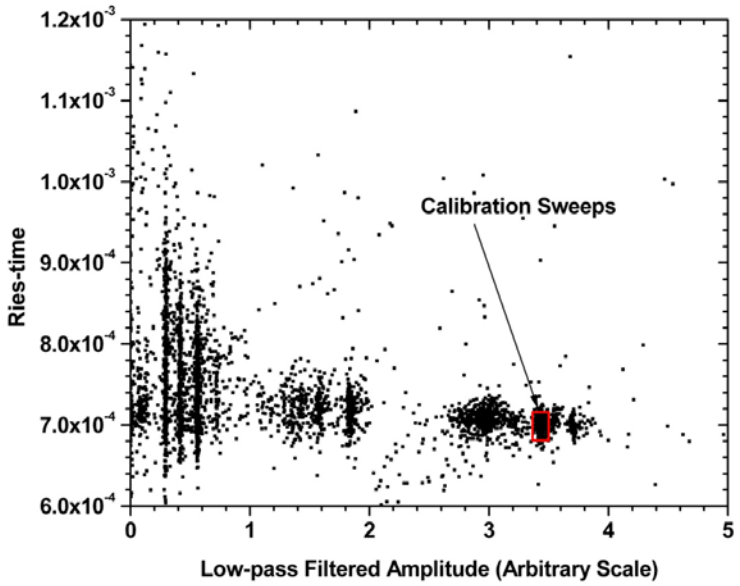
(Fig. 4.8) *Limit set on the allowed pre-trigger variation; everything to the right of the green line is rejected.*

The last histogram used to set limits is on the software trigger channel (fig. 4.9). This identifies abnormalities in the rise-time region which alter the effective rise-time. Because this trigger channel is set from the largest peak, cases where there is an additional pulse of large amplitude are also excluded here because their trigger will occur at a location too far up in the array to be in the acceptable range.



(Fig. 4.9) *Limits on the software trigger channel; everything outside of the green lines is rejected.*

The final and most important plot allows regions of interest to be indentified and investigated. It is a scatter plot of the rise-time vs. the low-pass filtered amplitude, which are the two most descriptive characteristics of a sweep (fig. 4.10). This plot can be used to identify regions of interest to verify that the detector and its setup are working as intended. This is where the calibration pulses (windowed signals) are selected. Because the x-axis provides a rough estimate of the energy spectrum, spectral features can be seen in the form of bands which aid in locating the calibration window.



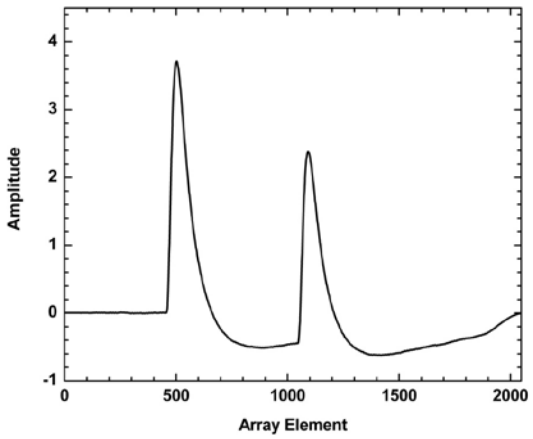
(Fig. 4.10) *Scatter plot of the rise-time vs. the low-pass filtered amplitude. The red square denotes the selection of calibration pulses. These events tend to be the most tightly clustered.*

4.2.4 Sweep Classification

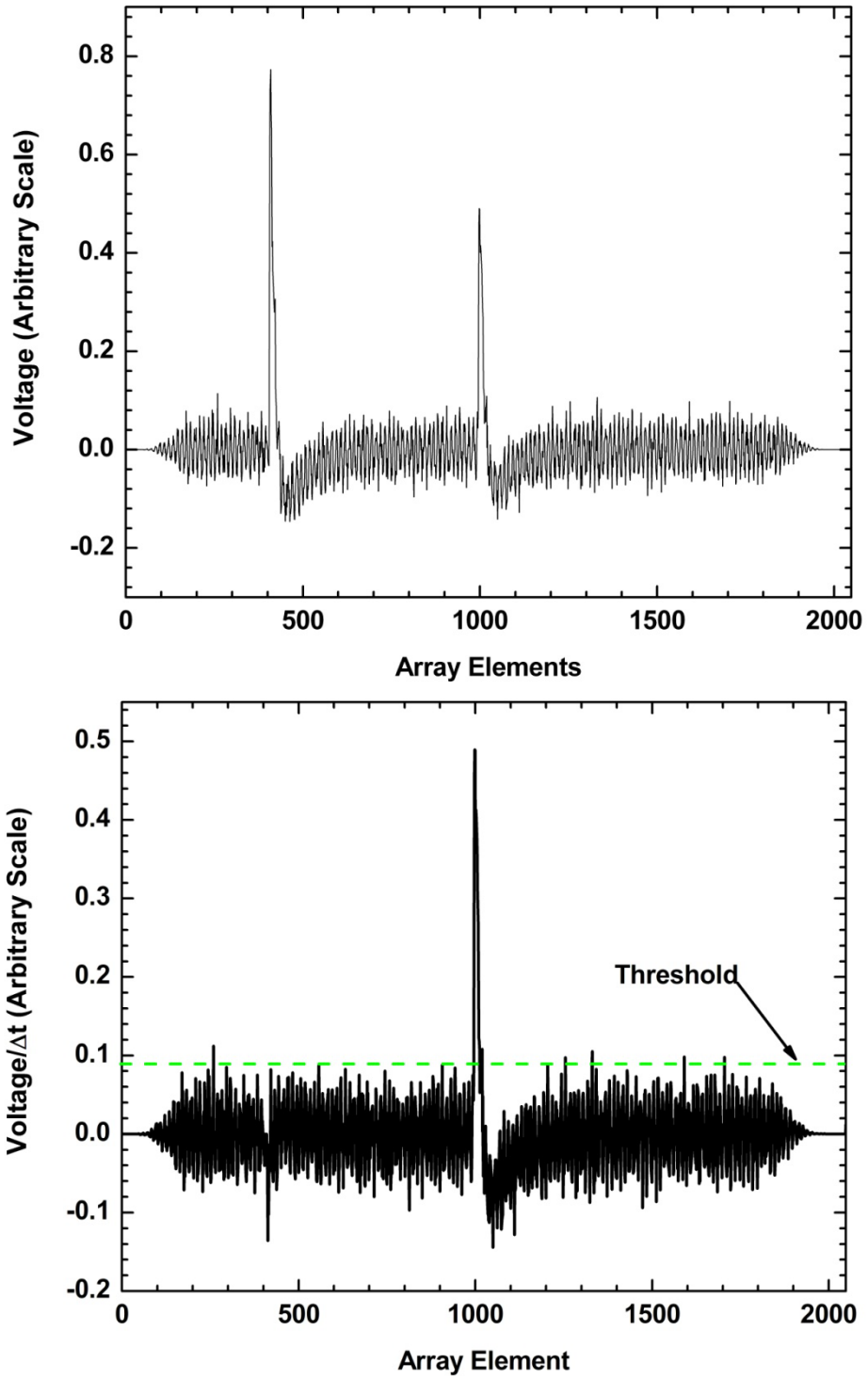
The third step in the analysis chain is to classify events based on the thresholds previously set. In general, four main classes can be identified: noise, pulses, calibration pulses, and bad pulses. The first three collectively are considered good pulses, and are used to build the energy spectrum. Throughout this thesis, the term "good pulses/sweeps" will refer to these classes. The bad pulses are collectively any sweep with a parameter that is not in an acceptable range set by the thresholds. These can be the result of detector saturation, noise spikes, false triggers, baseline jumps, or any other failure due to electronics or detector problems. Many of these events contribute to detector dead time. A high fraction of such events generally indicates a problem with the detector or readout electronics.

4.2.5 Double Pulse Analysis

The fourth step in the analysis chain is to search for double pulses. The algorithm used is based on a method devolved by the XQC team that has proven itself effective. This algorithm assumes a detector with a linear response which is not the case for the microcalorimeters used for MARE. The modifications to correct for this are covered in significant detail in the next chapter. Here, we cover the simple version which assumes a detector with a linear response. In this case, a template pulse is built from the average of the calibration sweeps. When scaled and subtracted from a sweep with two pulses, only the second pulse remains. This pulse will typically have an amplitude larger than the noise and thus may be identified (the threshold is typically based on a multiple of the RMS voltage of the baseline). This procedure is improved by taking the time derivative of the template and the pulse being analyzed before taking their difference. This reduces the algorithms dependence on the pulse shape and provides sharper features increasing its sensitivity.



(Fig 4.11) *Example of a double pulse.*



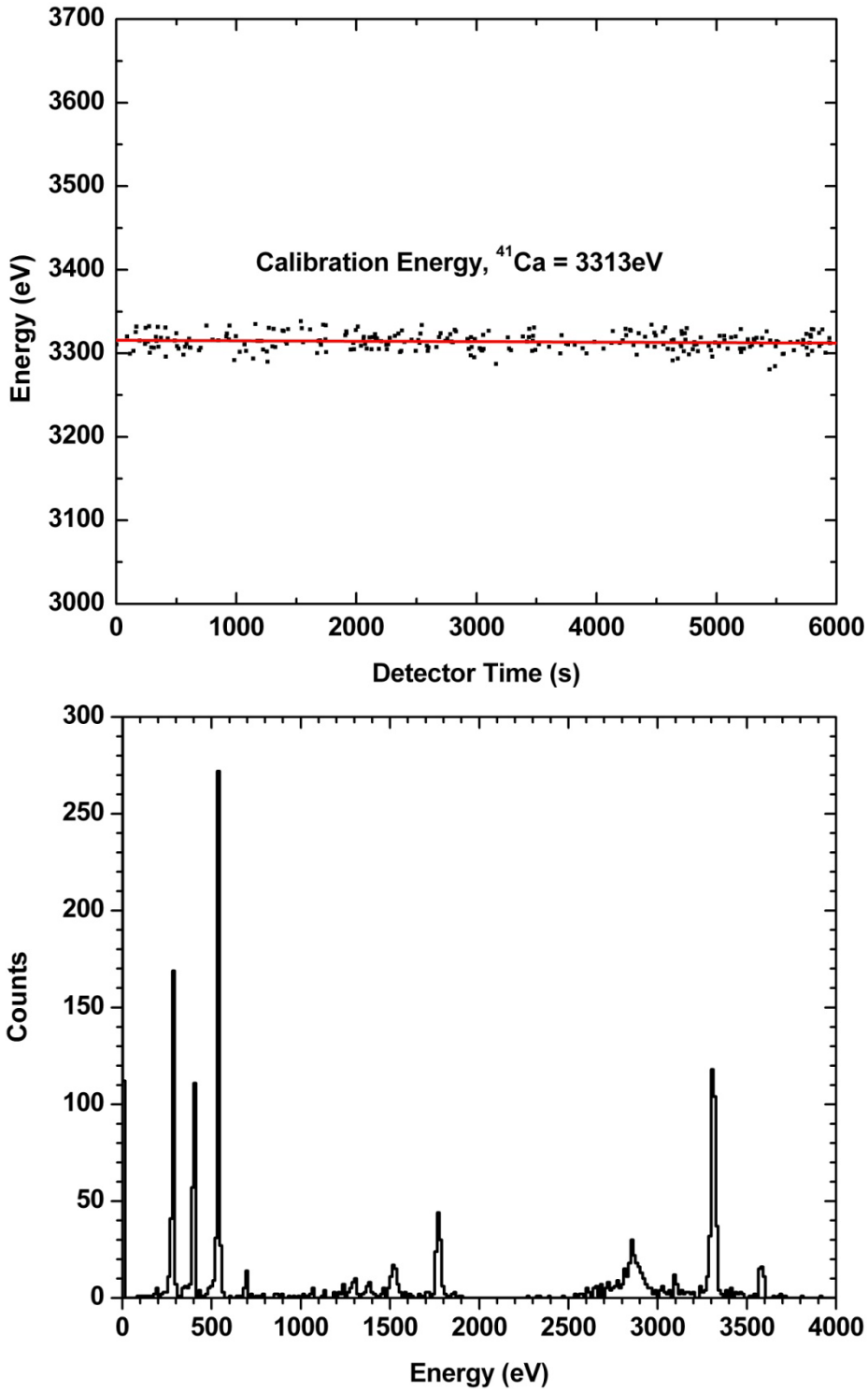
(Fig. 4.12) *Top: time derivative of double pulse in figure 4.11. Bottom: Above double pulse with template subtracted. The threshold is shown in green.*

4.2.6 Creation of the Optimum/Energy Extraction

The following two steps of the analysis chain build the optimum filter then extract the energies from the pulses as described early in this chapter.

4.2.7 Energy Spectrum Corrections

The seventh step in the analysis chain corrects for gain drift and nonlinearity of the detector (parabolic correction). The gain drift correction accounts for a linear change in the voltage baseline over time. The gain drift correction is achieved by fitting a line to the scatter plot of good calibration pulses as a function of their trigger time (fig. 4.14 - top). The second correction requires a second calibration peak of known energy and makes a second order polynomial correction using the baseline and the new and original calibration lines (fig. 4.14 - bottom). Considerations of the quality and quantity of data as well as the detector used should be considered when applying any of these corrections.



(Fig. 4.13) *Top: Visualization of the gain drift correction. Bottom: Nonlinear energy correction; a second calibration peak of known energy is selected.*

4.2.8 Energy Spectrum

The final step in the analysis chain is a plotting routine that allows histograms and scatter plots of the parameters obtained in previous subroutines. The plot which is most important is the energy spectrum obtained from a histogram of the extracted energy with any corrections applied in the previous subroutine. In the case where a nonlinear correction is not applied (as with this data set), the lower plot of figure 4.13 is an example of what the energy spectrum looks like.

Chapter 5

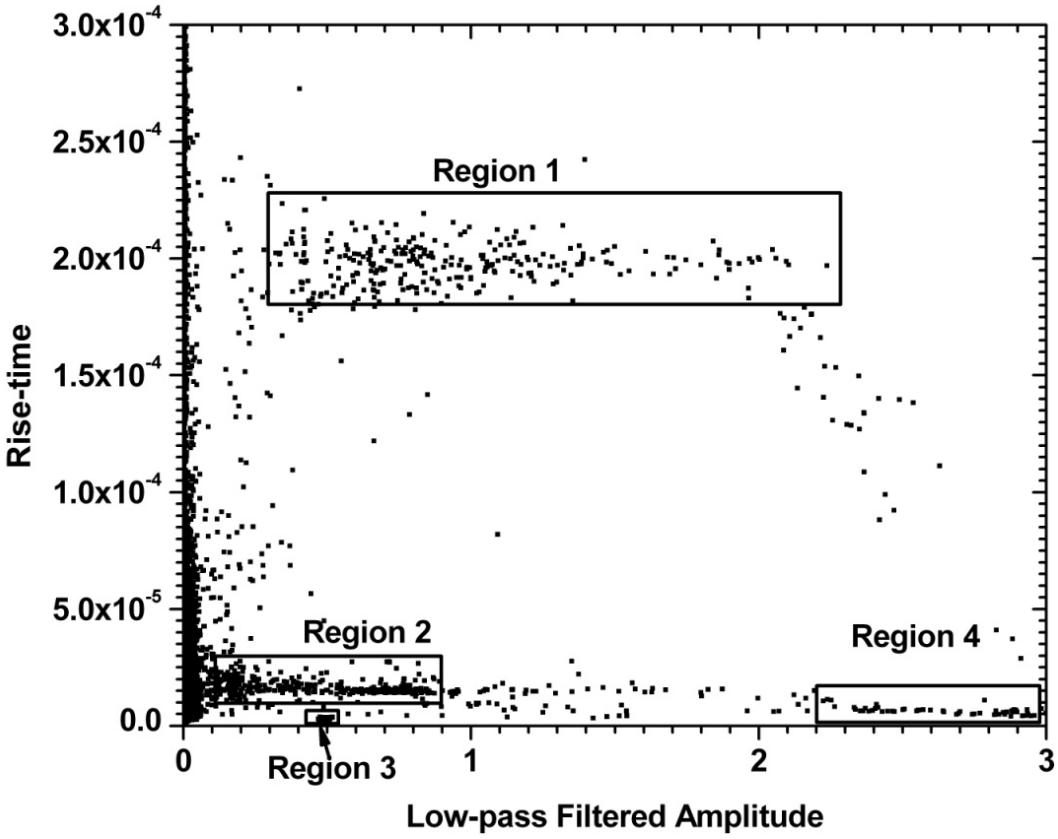
Verification of Holmium Source

5.1 Background

As described earlier, identifying the best method for producing ^{163}Ho is one of the landmarks for the holmium track of the MARE experiment. This process is in its initial stages where the most important requirements are to first verify the absence of long-lived radioactivity and to then actually verify that ^{163}Ho was produced. The University of Genoa obtained a sample produced by neutrino activation of ^{162}Er in a nuclear reactor. The sample was isolated for a month to allow the dissipation of short life-time contaminants. The absence of long-lived radioactivity was confirmed with a proportional counter. The x-rays from the ^{163}Ho EC-reaction are very low energy and are quickly attenuated by the atmosphere. To verify the presence of ^{163}Ho, the source was dissolved in hydrochloric acid, then deposited by hand onto a microcalorimeter. Once evaporated, the presence of ^{163}Ho can then be verified via microcalorimetric measurements. Because the source is not embedded in the absorber the results will not be optimal, but should be sufficient. Once data was acquired, it was analyzed by FITSFILER at the University of Miami.

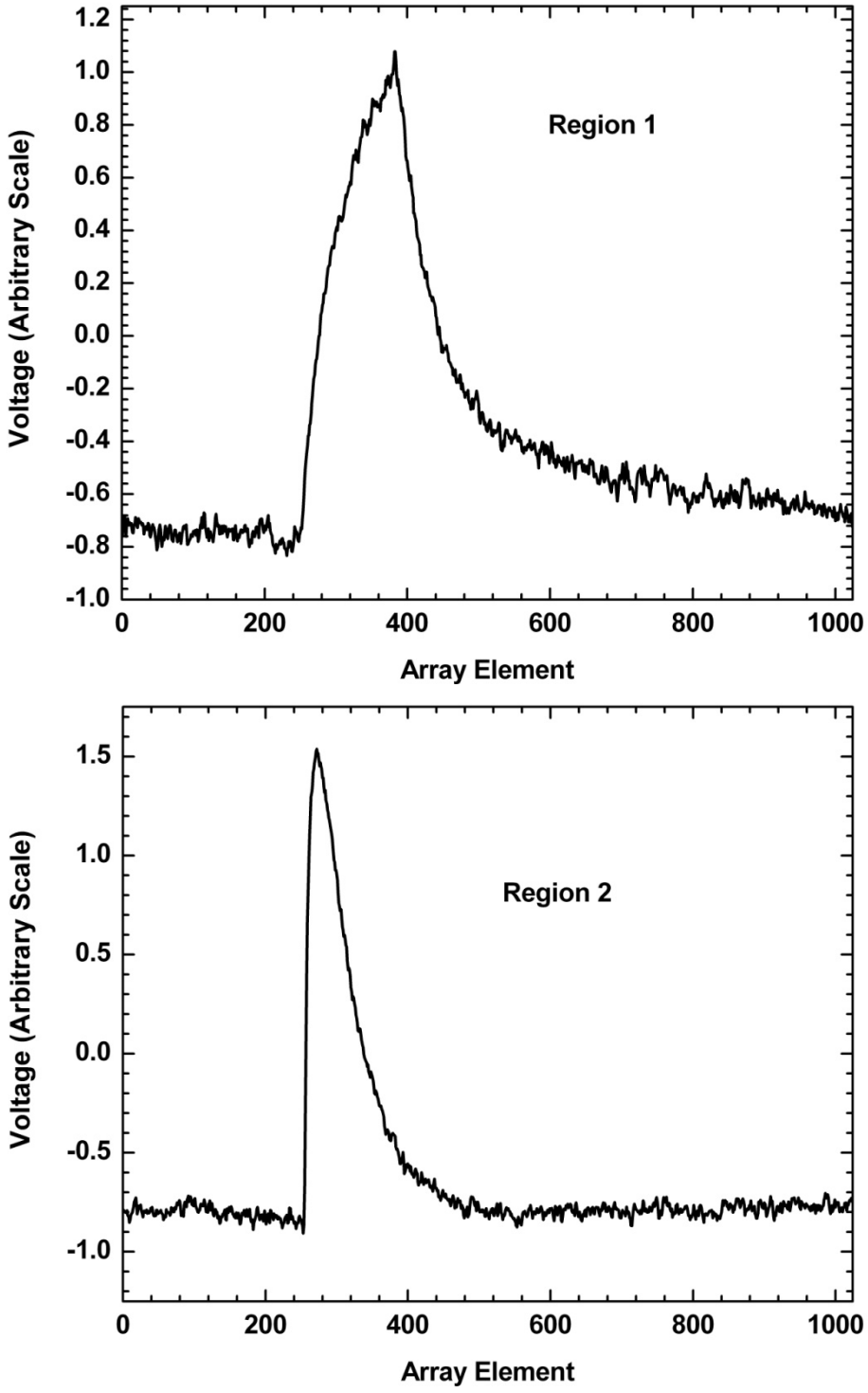
5.2 Holmium Data Analysis

We first examine the scatter plot of the rise-time vs. the low-pass filtered amplitude (1kHz low-pass filter) to identify regions of interest. In fig 5.1 below, 4 regions of interest can be identified. Most likely the points that are grouped with tighter rise-time are acceptable, but this assumption must be verified. In order to do this, sweeps from each region must be explored. Included are representative pulses from each region (fig. 5.2 and 5.3).

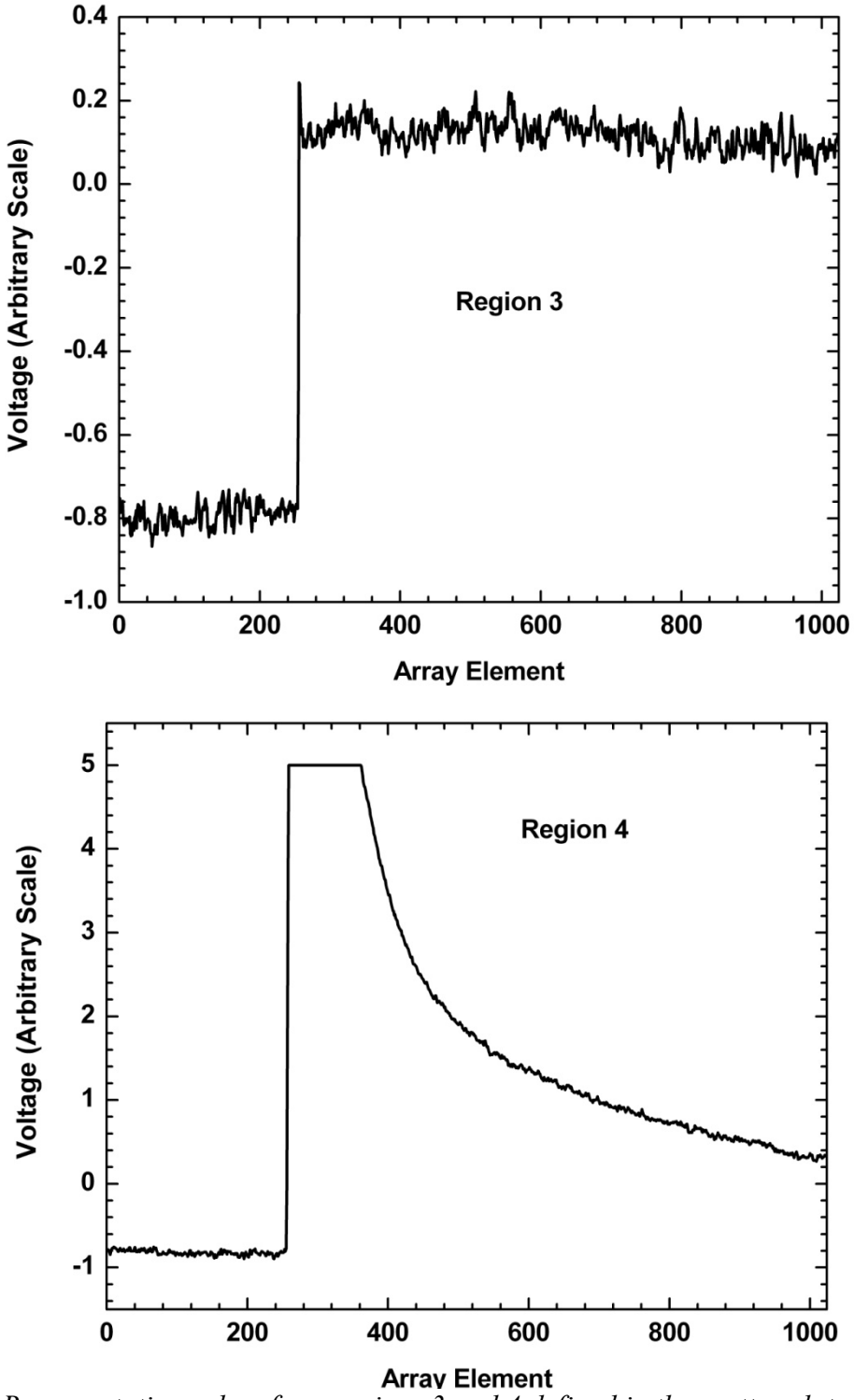


(Fig. 5.1) *Scatter plot of rise-time vs. the low-pass filtered amplitude. In this plot, there are four distinct regions to consider.*

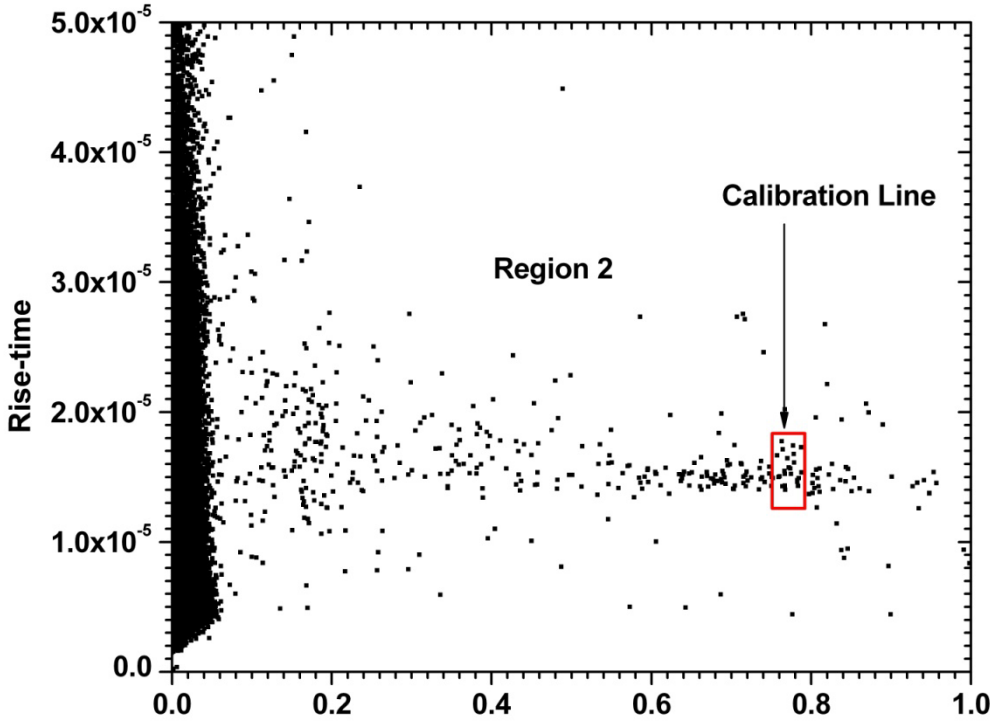
All the data points near the left end correspond to noise or incompletely absorbed radiation and are ignored. In region 1 (fig. 5.2 - top), pulses are consistently oddly shaped like s stretched out pulse. This is likely caused by the non-ideal thermal connection between the source and absorber, or by some other thermal problem with the detector. Pulses in region 2 (fig. 5.2 - bottom) seem to be well behaved and are a good candidate for the ^{163}Ho spectrum. We will examine these in more detail shortly. The small cluster which constitutes region 3 (fig. 5.3 - top) are all voltage jumps, easily identified by FITSFILTER via their low rise-time and long decay-time. These jumps are the result of the SQUID malfunctioning and shifting the baseline by one flux quantum. Region 4 (fig. 5.3 - bottom) consists of high energy pulses that saturate the detector. Only region 2 seems to have viable pulses, so we explore it in more detail.



(Fig. 5.2) *Representative pulses from regions 1 and 2 defined in the scatter plot above.*

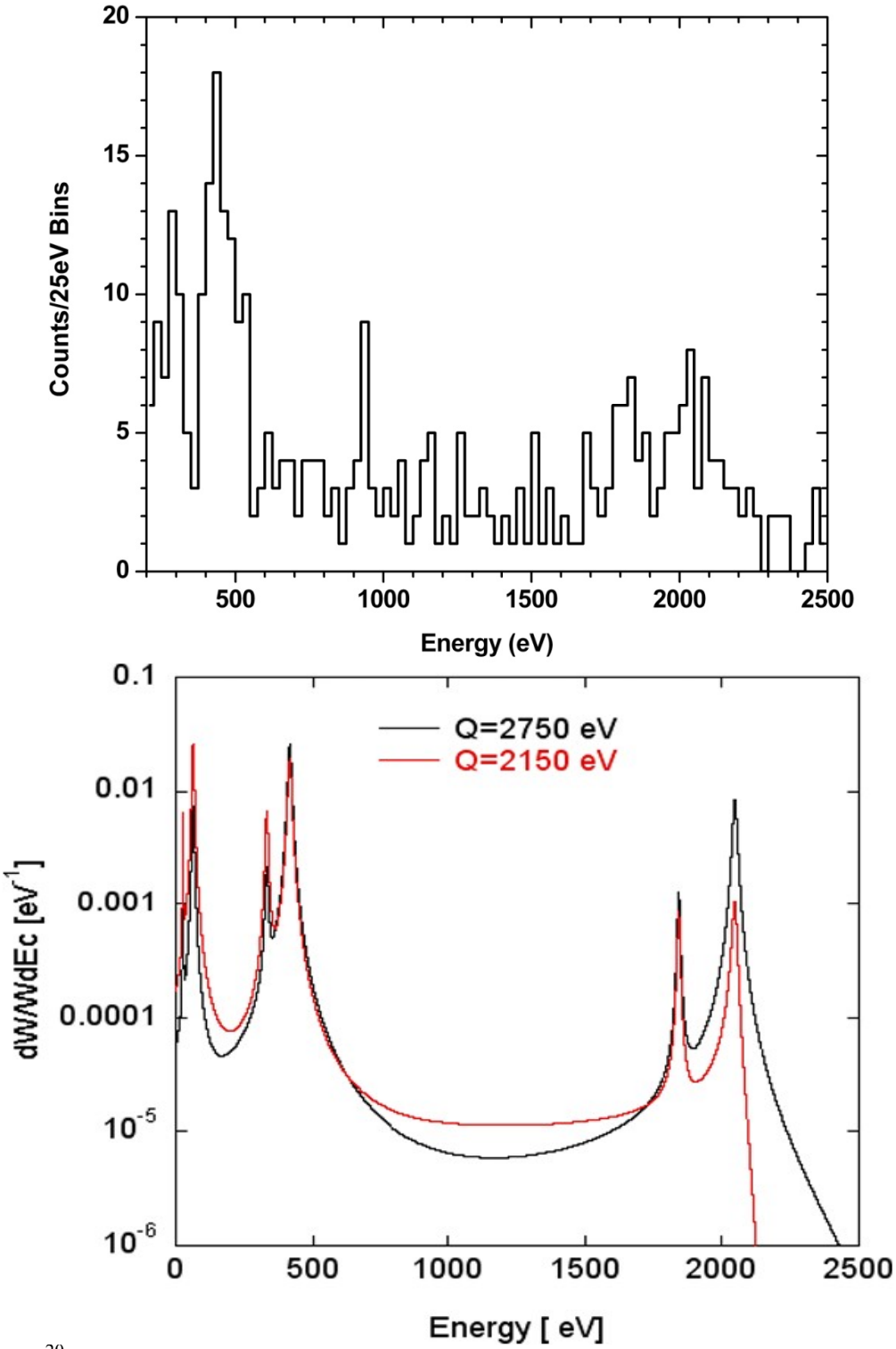


(Fig. 5.3) *Representative pulses from regions 3 and 4 defined in the scatter plot above.*



(Fig. 5.4) *Region 2 with the calibration sweeps selected. We are assuming that these pulses correspond to the resonant peak of* ^{163}Ho*, near the end of its spectrum at 2.039 eV.*

In the close-up of region 2 above, we get a closer look at our ^{163}Ho β-spectrum candidate. Week bands are visible and look like they correspond to the resonance peaks from the electron capture reaction. Using this region, we continue through the analysis steps, but ignore the double pulse classification and energy spectrum corrections (recall we are just verifying that we are detecting the β-spectrum of ^{163}Ho). After this is done, we obtain the energy spectrum by taking the histogram with bins of size 25 eV, which when compared to a theoretical spectrum is a very close match (fig. 5.4). Thus by using FITSFILTER, we were able to verify the presence and detection of ^{163}Ho using the detectors fabricated in Italy.



(Fig. 5.5)[20] *Observed spectrum and a theoretical spectrum for comparison.*

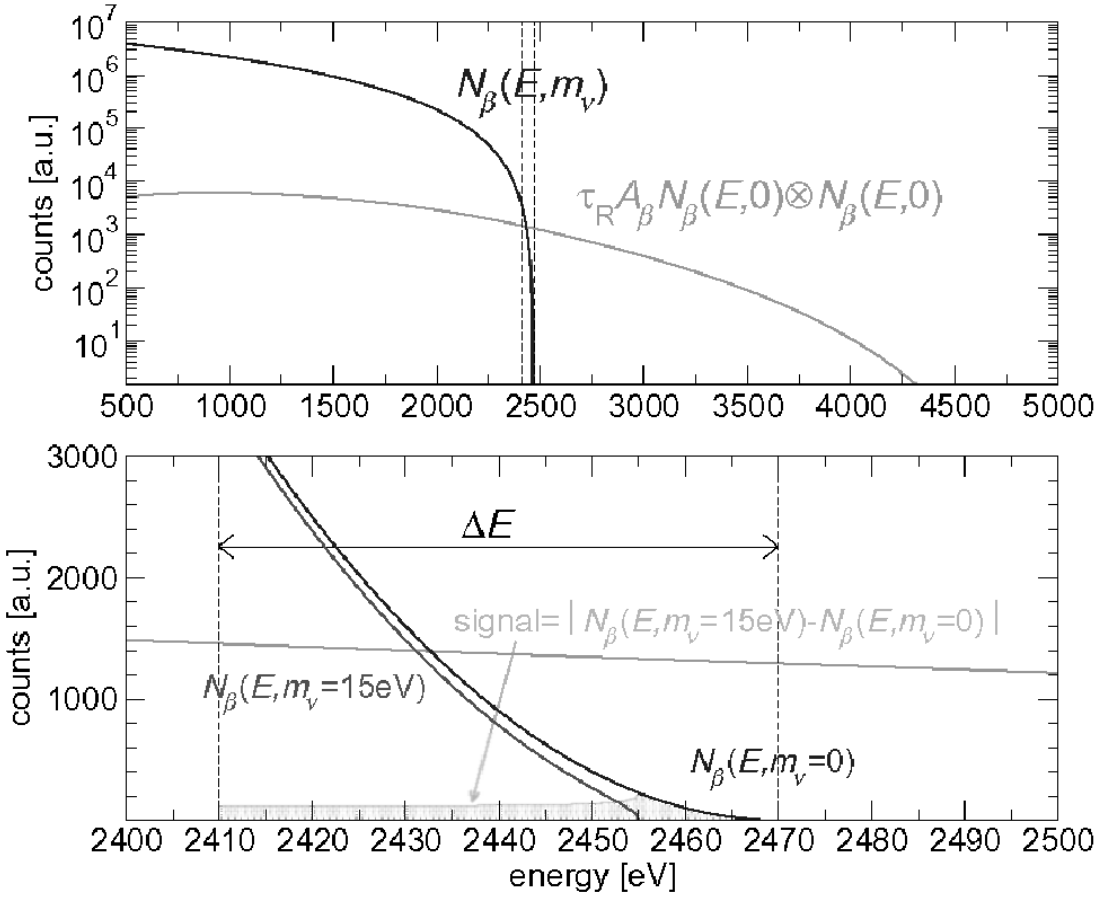
Chapter 6

Double Pulse Detection

6.1 Background

Double pulses are the result of two pulses appearing in a single sweep. Pulses are characterized by their energy, rise-time, and decay-time and spend most of their short life decaying back to equilibrium. Because of this, most cases of double pulses consist of the second pulse occurring on the tail of the first. These are generally easy to detect with any reasonable double pulse algorithm and are not considered here. Alternately, the rise-time is very short compared to the length of the sweep and to the decay-time. Thus a small percentage of double pulses can be classified as rise-time double pulses. As the maximum of the two pulses approach each other, the resulting double pulse looks more and more like a single pulse with energy corresponding roughly (depending on how linear the detector response is) to the sum of the two constituent pulses. This property makes rise-time double pulses very difficult to detect and sets the limit on algorithm efficiency and on the potential sensitivity of MARE. Unless otherwise stated, the term double pulse in this section chapter will correspond to rise-time double pulses.

Because we are most interested in the Q-point of the energy spectrum where the count rate is very low, the double pulse spectrum can easily become the most significant source of error for our experiment (fig. 6.1). Thus it is very important to understand its effects and to minimize the possibility of rise-time double pulses.



(Fig. 6.1) [19] *Simulations of the pileup spectrum and its effects on the Q-value. Top: the β-spectrum and its associated pileup spectrum. Notice how sharply the true β-spectrum ends. Bottom: the β-spectrum with and without the effects of a double pulse. Note how the endpoint loses its sharpness with the addition of the pileup events.*

If we let A denote the average count rate for a pixel, let τ_R denote the resolving time, and assume a Poisson distribution, we may estimate the fraction of double pulses. In general, the probability of k occurrences within a given interval with average occurrence λ, we have:

$$P(k,\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}, \lambda = A\tau_R \tag{6.1}$$

It is easier to consider the probability of not having a double pulse in which case k is either zero or one.

$$P(0,\lambda) = e^{-\lambda} \tag{6.2}$$

$$P(1,\lambda) = \lambda e^{-\lambda}$$

Since the sum of all probabilities is one, we have

$$P(unresolved\ double\ pulse) = 1 - (1+\lambda)e^{-\lambda} \tag{6.3}$$

In the case where $\lambda \ll 1$, this can further be simplified to:

$$P(unresolved\ double\ pulse) \cong \lambda^2 \tag{6.4}$$

up to second order.

We can further calculate the double pulse spectrum if the spectrum is known. Let $N_\beta(E)$ denote the energy spectrum, normalized so that its integral over its energy domain is 1. Up to a normalization constant, a, the probability of a double pulse occurring with energy E' is given by:

$$a\int_0^{E'} N_\beta(E'')N_\beta(E'-E'')\,dE'' \tag{6.5}$$

And the cumulative density function is given by

$$F(E) = a\int_0^{E}\left(\int_0^{E'} N_\beta(E'')N_\beta(E'-E'')\,dE''\right)dE' \tag{6.6}$$

with

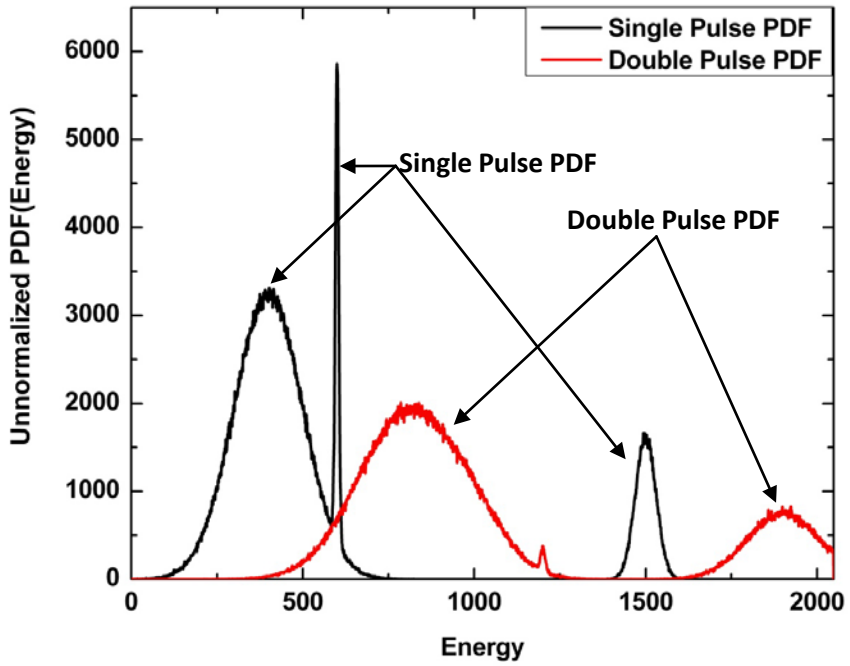
$$a = \frac{1}{\int_0^{\infty}\left(\int_0^{E'} N_\beta(E'')N_\beta(E'-E'')\,dE''\right)dE'} \tag{6.7}$$

Thus the observed spectrum assuming $\lambda \ll 1$ is:

$$N_{\beta,obs.}(E) = (1-\lambda^2)N_\beta(E) + \lambda^2 a \int_0^E N_\beta(E')N_\beta(E'-E)\,dE \tag{6.8}$$

where we have ensured that

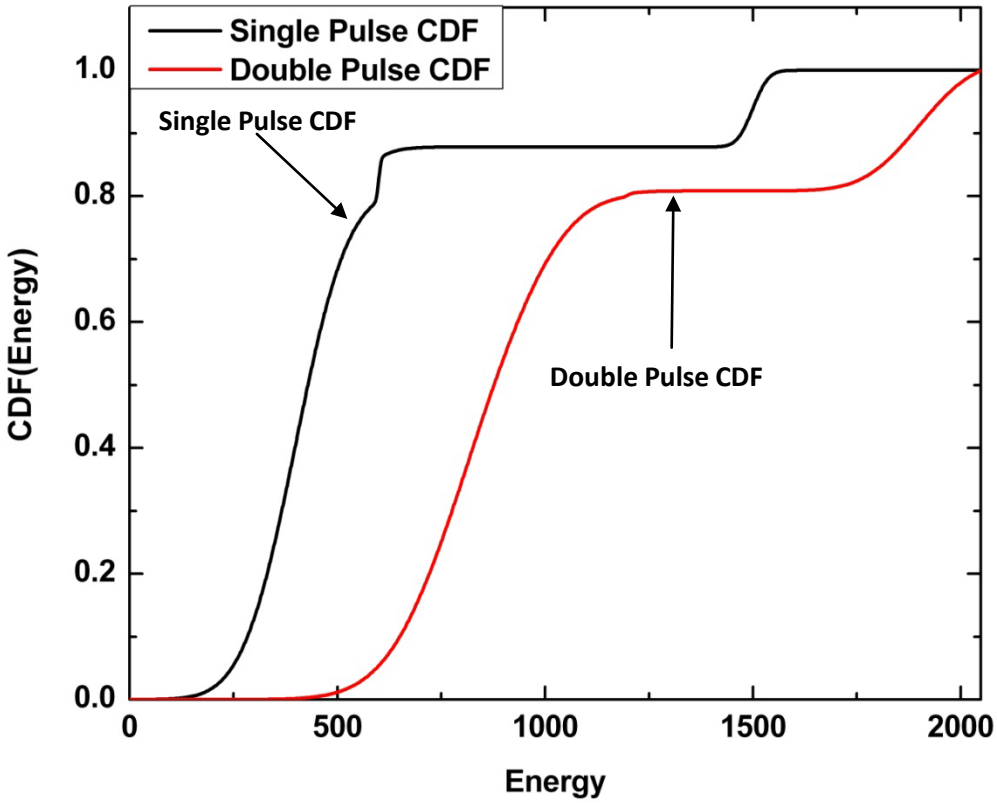
$$\int_0^\infty N_{\beta,obs.}(E'')dE'' = 1 \tag{6.9}$$



(Fig. 6.2) *A simulated energy spectrum and its associated double pulse spectrum.*

In order to get a sense of how the double pulse energy spectrum is dependent on the true spectrum, we created a toy program that generates a double pulse spectrum from an input spectrum using a simple Monte Carlo simulation. An input spectrum consisting of the sum of three normal distributions was used: $\{(400,100),(600,5),(1500,30)\}$ for pairs of (μ,σ) and with relative scaling rations of 0.6,1, and 0.3 respectively. The resulting spectrums play the role of the energy histograms that would be extracted from data using the analysis program and are plotted against an arbitrary energy scale. The

main features to note are how the double pulse spectrum resembles the single pulse spectrum smeared out over higher energies, and how narrow peaks have a limited effect whereas wide peaks have a significant effect. The smearing effect is particularly apparent in the cumulative density functions (CDFs) associated with the spectrums (fig. 6.2). This is why the Q-value, which is at the end of the β-spectrum has such a strong effect.



(Fig. 6.3) *The CDFs for the single pulse spectrum and its associated double pulse spectrum.*

6.2.1 Initial Results

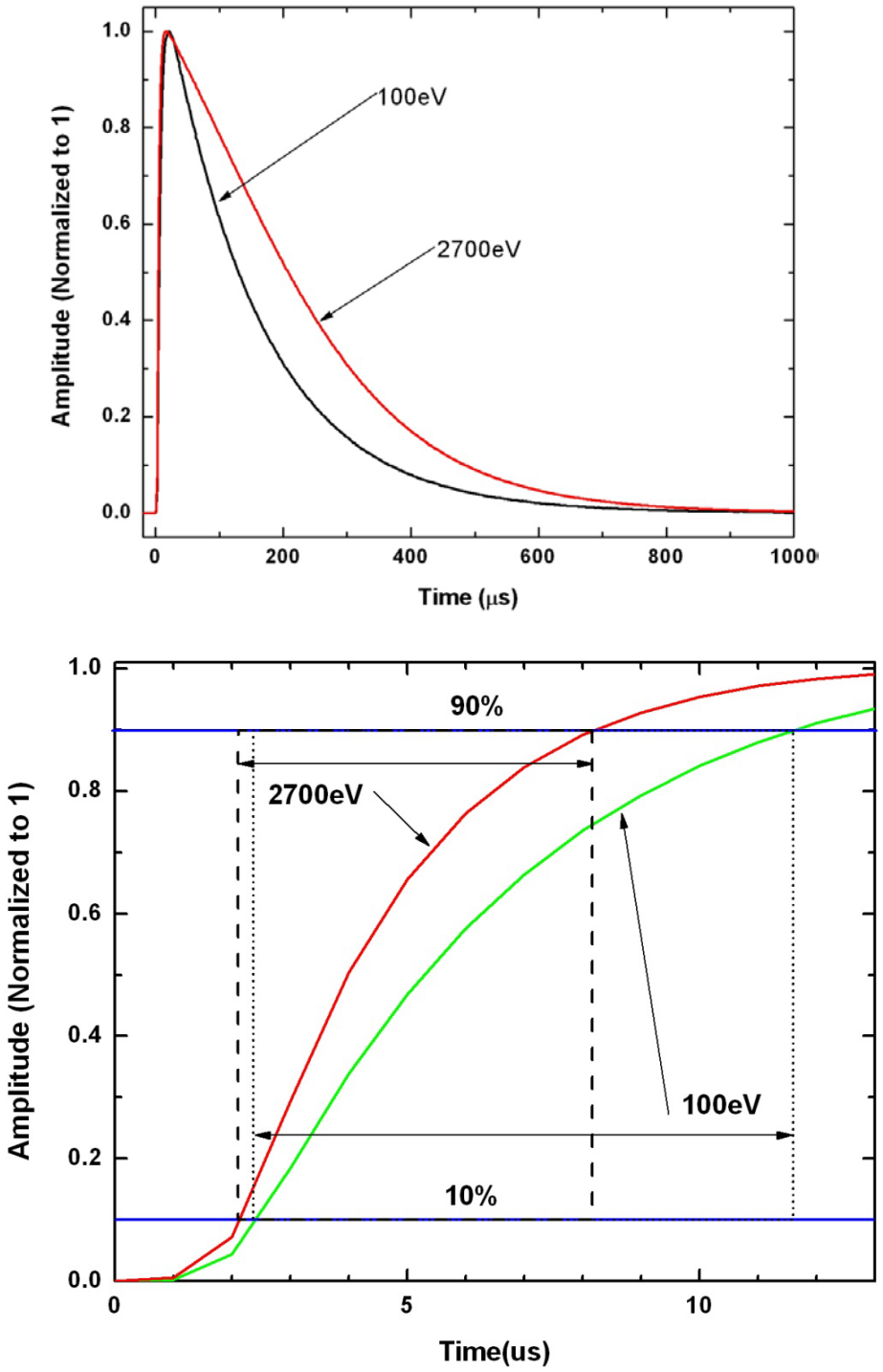
First, we note that often times the goal is not only to detect double pulses but to resolve the energies of their constituent pulses. Due to the low count rate for MARE, the probability of rise-time double pulses is sufficiently low that simply rejecting them has

no significant effect on the experiment and dead time calculations. Additionally, due to the nonlinearity of microcalorimeters, we would not be able to extract their energies if we wanted to! This research then strictly focuses on double pulse identification and rejection with no concern for extracting their energies.

The double pulse algorithms explored rely on templates created from averaging good sweeps and are thus very sensitive to their accuracy. For detectors with a very linear response, the template is constructed from the calibration sweeps and then used to filter the other sweeps searching for double pulses. However, for the detectors we are considering, the energy response is nonlinear and requires modification of the above process. The nonlinear response of the detector is reflected in varying pulse shapes as a function of energy. Thus we may approximate pulses as:

$$Pulse = f(t, E) \neq Ef(t). \tag{6.10}$$

To correct for this, the energy spectrum is divided into subsections (based on their low-pass filtered energy extracted early on in FITSFILTER) where the variation of the sweeps in each subsection is acceptably negligible (this is addressed further in the current results subsection of this chapter). For each sub-region, all good pulses are averaged to create a template which is then used to filter its constituent pulses. If the particular algorithm detects any double pulses, this indicates that those bad pulses were used in the creation of the template. After the double pulses are classified, the remaining good pulses are used to create a more accurate template and the process repeats iteratively until no double pulses are detected in the given energy division.



(Fig. 6.4) *Example of a nonlinear detectors response to pulses of different energies (both scaled so that their maxima are equivalent). Top: Note how strongly the decay time is effected by pulse energy. Bottom: the rise-time is similarly affected by nonlinearity.*

6.2.2 Experimental Setup

In order to properly characterize the efficiency of double pulse detection algorithms, sweeps with known parameters were generated at the University of Florida. They developed a set of tools to simulate the MARE experiment, done in two steps: first, a Geant[21] Monte Carlo simulation is used to generate a list of events; second, this list is used as input for a numerical solver which realistically models a TES detector's noise and nonlinearity. [22] The detector parameters in the simulation are based on microcalorimeters developed at the NASA/Goddard Space Flight Center, [23] which are the current baseline for a holmium MARE experiment. In our case, the first step is simplified by using a user generated set of input values. The sets generated for the preliminary results have an energy resolution of 2 eV with an acquisition rate of 1 μs. Double pulses were created using pair-wise combinations of the ten energies from the following set with separations times of 0.5, 1, 2, and 5μs:

$$Energies(eV) = \{50,100,200,400,700,1000,1400,1800,2200,2700\}. \quad (6.11)$$

Calibration pulses were created with energies ranging from 100 to 2700 eV in steps of 100 eV. The range of energies corresponds to a region of interest valid for most calorimetric neutrino experiments. For each point in our parameter space described above 500 sweeps were generated.

Corresponding to the 27 different calibration energies used, we divided the energy spectrum into 27 subsections. Then each section was analyzed separately using the linear version of the detector with the calibration pulses classified as windowed signals (used for the algorithm template). This allowed many double pulses to be analyzed with few

calibration pulses. This method only becomes unacceptable when a significant fraction of pulses are pileup events. The resulting template may be sufficiently corrupted that the algorithms efficiency is diminished (this effect is explored in the current results subsection of this chapter). This situation requires sources with very high activity which is not the case for MARE and thus is not considered in the preliminary analysis.

The boundaries for the 27 subdivisions of the energies (based on the 27 energy calibrations sets created) were chosen to lie halfway between the average low-pass filtered energies of the calibration pulses. The lower limit used is 50 eV and the upper limit used is about 2750 eV. The average low-pass filtered energies for the doubles pulses was similarly averaged and the resulting value was used to determine which region a particular double pulse set would be grouped with. Any combinations that resulted in energies above 2750 eV were excluded. The following table shows the groupings used in the analysis. A total of 73 double pulse pairs were used while the remaining 23 were excluded.

Regions (eV)	Associated double pulses (eV)
100	{(50,50),(50,100),(100,50)}
200	{(100,100),(50,100),(100,50)}
300	{(100,200),(100,200)}
400	{(200,200),(50,400),(400,50)}
500	{(100,400),(400,100)}
600	{(200,400),(200,400)}
700	{(50,700),(700,50)}
800	{(400,400),(100,700),(700,100)}
900	{(200,700),(700,200)}
1000	{(50,1000),(1000,50)}
1100	{(400,700),(700,400),(100,1000),(1000,100)}
1200	{(200,1000),(1000,200)}
1300	{None}
1400	{(50,1400),(1400,50),(400,1000),(1000,400)}
1500	{(100,1400),(1400,100)}
1600	{(200,1400),(1400,200)}
1700	{(700,1000),(100,700)}

1800	{(400,1400),(1400,400),(50,1800),(1800,50)}
1900	{(100,1800),(1800,100)}
2000	{(1000,1000),(200,1800),(1800,200)}
2100	{(700,1400),(1400,700)}
2200	{(400,1800),(1800,400)}
2300	{(50,2200)*,(2200,50)*,(100,2200),(2200,100)}
2400	{(400,1000),(1000,400),(200,2200),(2200,200)}
2500	{(700,1800),(1800,700)}
2600	{(400,2200),2200,400)}
2700	{(50,2700),(2700,50),(1400,1400), (1000,1800),(1800,1000)}

(Table 6.1) *These values are consistent for 0.5, 1 ,2 , and 5 μs. The exception is the two symmetric pairs denoted with an asterisk. These pairs lie in the 2200 eV range for the 5 μs separation.*

6.2.3 Standard (XQC) Algorithm

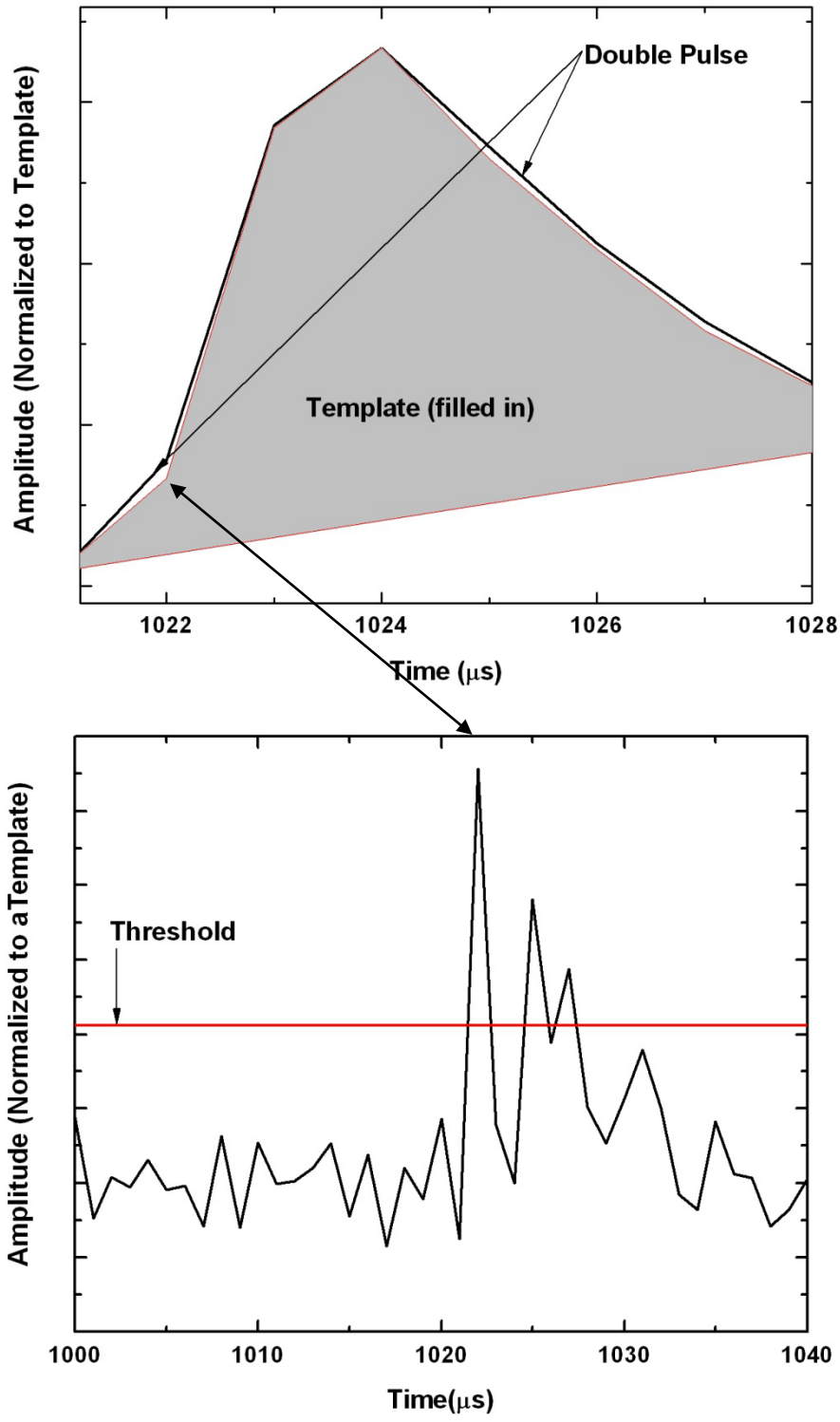
Here, we go over the standard algorithm briefly introduced in the previous chapter in more detail. The first portion of analysis for the standard algorithm (XQC algorithm for short) involves obtaining a threshold from the noise and is independent of the energy subdivision being considered. To reduce the wordiness of this subsection the prefix d_t will be used to denote the first derivative with respect to time of the following object

- Calculate d_t-noise for good noise sweeps, compute the associated RMS value, and average this for all noise sweeps: σ;
- The threshold, θ, is set as $2^a N\sigma$, where a is initialized with a value of 4, and N is the minimum multiple of σ to use as a threshold (for all analysis we use $N = 5$);
- All good d_t-noise sweeps are then scanned for excursions above θ;
- If any such excursions are found their associated noise sweep is reclassified and the process repeats in order to calculate a “cleaner” value for σ;
- If no such excursion are found, then a is reduced by 1;

- The process is repeated until $a = 0$ in which case, we have the threshold: $\theta = N\sigma$.

Now that we have established a threshold, we analyze each energy division separately as follows:

- Average all calibration sweeps (shifted to line up based on their software trigger) and calculate its time derivative to build a template: T;
- Each sweep being analyzed is shifted (again, based on the software trigger) and its time derivative is taken giving us: P_i;
- The template is normalized so that its maximum corresponds to that of P_i, and we take their difference;
- This difference is scanned for excursions beyond the threshold, $\theta = N\sigma$, already established by the noise sweeps;
- The sweep associated with any such excursions is reclassified as a double pulse.



(Fig. 6.5)[24] *Top: comparison of the template and the first derivative of a double pulse with pulse energies of 1000 eV and 100 eV and a separation of 1 μs. Bottom: difference between the two curves with the threshold in red (horizontal line).*

6.2.4 Optimum Filter (OF) Algorithm

The optimum filter algorithm (OF algorithm for short) was our attempt at creating an algorithm fine tuned to detecting double pulses. The pulses we are considering have two characteristically sharp features corresponding to the first instance of the pulse and the rise-time to decay-time transitions (near the pulse maximum). Because of this, the second derivative with respect to time is very sensitive to pulses. As with the previous section, to reduce the wordiness of this subsection the prefix d_{tt} will be used to denote the second derivative with respect to time of the following object. If we construct a template from averaged d_{tt}-pulses and convolve this with its individual constituents (d_{tt}-pulses), decay-time double pulses will generally be characterized by two peaks at different phases, and rise-time double pulses will be characterized by a distorted, slightly fatter peak. Because it is the latter we are interested in detecting, we need to be able to distinguish the difference in shape of the convolutions corresponding to single and double pulses.

It would be beneficial to use a template d_{tt}-pulse that is modified to maximize the signal to noise ratio. This can be achieved similarly to the optimum filter used in the analysis program to extract the energies of pulses in FITSFILTER. The derivation assumed a function of the form:

$$V(t) = E_0 g(t). \tag{6.12}$$

The second derivative of a pulse also follows this form:

$$V_{tt}(t) = \frac{d^2V(t)}{dt^2} = E_0 \frac{d^2g(t)}{dt^2} = E\, g_{tt}(t). \tag{6.13}$$

So long as pulses in a given range do not vary significantly in shape, this is valid. However, the noise response varies on a pulse due to the nonlinearity and results in correlation of noise in the frequency domain. The resulting optimum filter does not maximize the signal to noise ratio as well as possible. Improving this requires considering the noise correlations, but this is not practical for our considerations as it would significantly increase the analysis time for a single pulse. Thus we consider the resulting optimum filter sufficient.

Returning to the original goal of discerning rise-time double pulses, we first create a template convolution. This is the result of creating an optimum filter from averaged d_{tt}-pulses (along with d_{tt}-noise average in the frequency domain) and convolving it with the averaged d_{tt}-pulses. The resulting convolution template is normalized so its maximum is 1. Similarly, the pulse being analyzed is convolved with the optimum filter and normalized to one and shifted so that their peaks coincide. All that is required at this point is a measure of the difference of the convolutions and a threshold on this measure. A simple measure could be described as:

$$m(c_T(t_n), c_S(t_n)) = \max\{c_T(t_n) - c_S(t_n)\} \tag{6.14}$$

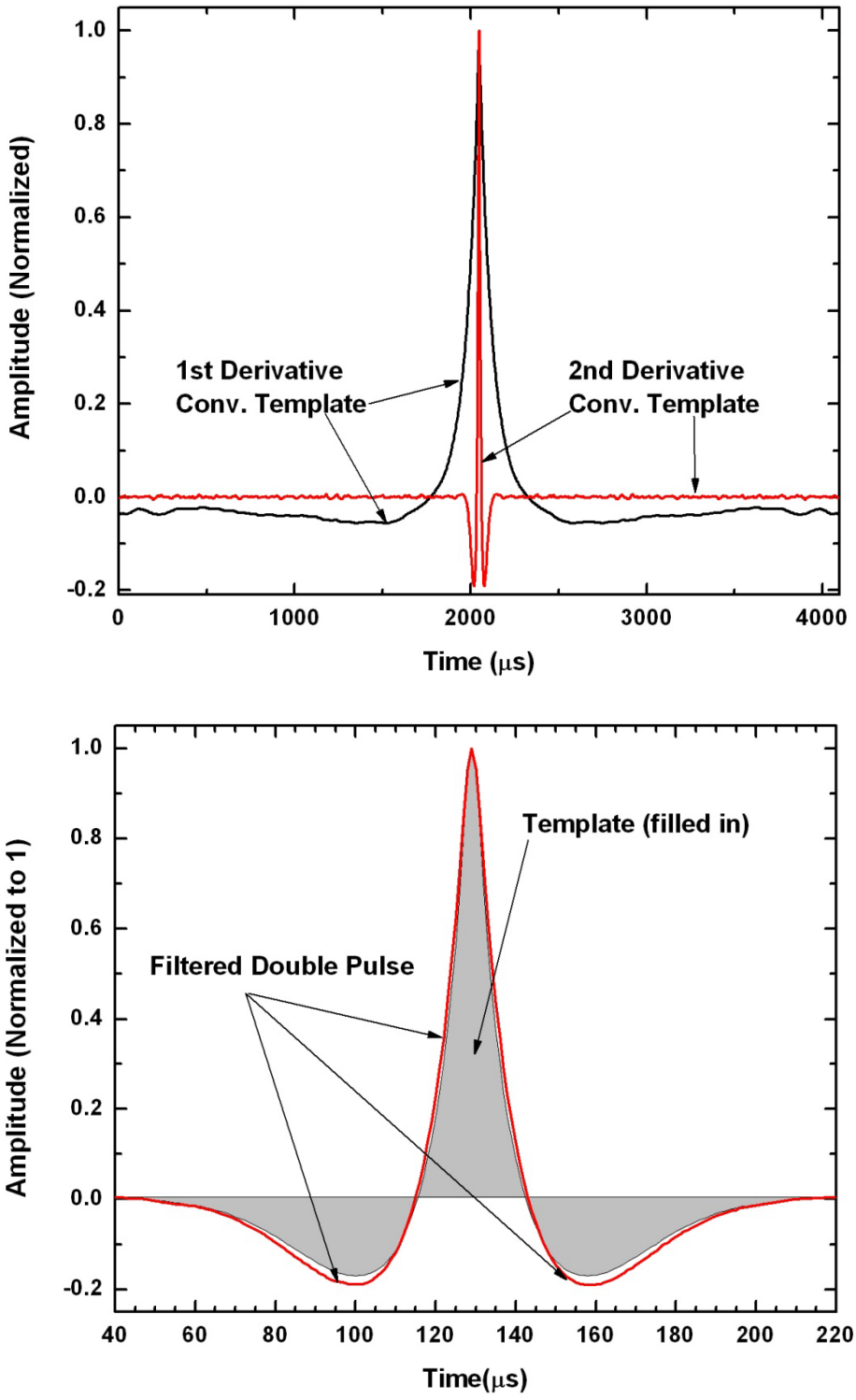
Where C_T is the template convolution and C_S is the optimum filtered pulse (this is analogous to the measure used for the standard algorithm). However, we are more interested in differences near the peak and do not care as much about the other regions. Several other measures were explored, but the best results were obtained using:

$$m(c_T(t_n), c_S(t_n)) = \max\{c_T^2(t_n) - c_s^2(t_n)\} \tag{6.15}$$

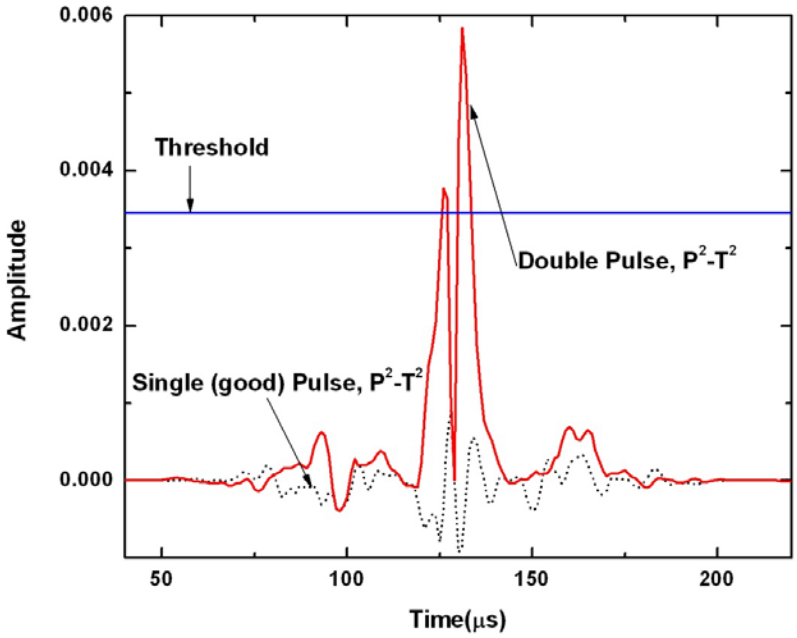
This measure exaggerates differences between the two convolutions, particularly near the peak as desired. The threshold is chosen via a histogram of the resulting values. If we let

characters with a tilde represent Fourier transformed functions (in the frequency domain), the process may be outlined as follows (for a given subdivision of the energy):

- Average good pulses in the time domain, calculate its second time derivative and transform to the frequency domain: $\tilde{P}_{tt,ave.}(f_n)$;
- Compute d_{tt}-noise and average in the frequency domain: $\tilde{N}_{tt,ave.}(f_n)$;
- Create the optimum filter as: $\tilde{F}_{opt}(f_n) = \frac{\tilde{P}_{tt,ave.}(f_n)}{\tilde{N}_{tt,ave.}(f_n)}$;
- Create the convolution template as: $\tilde{C}_T(f_n) = \left(\tilde{F}_{opt}(f_n)\right)\left(\tilde{P}_{tt,ave.}(f_n)\right)$, and convert this to the time domain: $C_T(t_n)$;
- Normalize $C_T(t_n)$ so that its maximum is 1: $c_T(t_n)$;
- For each pulse, $\tilde{P}_{tt,i}(f_n)$, to be analyzed, create a template as: $C_{p,i}(t_n) = \left(\tilde{F}_{opt}(f_n)\right) * \left(\tilde{P}_{tt,i}(f_n)\right)$ (this is done in the frequency domain as with the template above);
- Normalize $C_S(t_n)$ so that its maximum is 1, and shift so that its peak corresponds to that of the template convolution: $c_S(t_n)$;
- Calculate: $m\left(c_T(t_n), c_{S,i}(t_n)\right) = \max\left\{c_T^2(t_n) - c_{S,i}^2(t_n)\right\} = m_i$
- Calculate the average (μ) and variance (σ) for the resulting set of m_i and set the threshold equal to: $\mu + 3\sigma$;
- Pulses corresponding to an m_i above this threshold are classified as double pulses



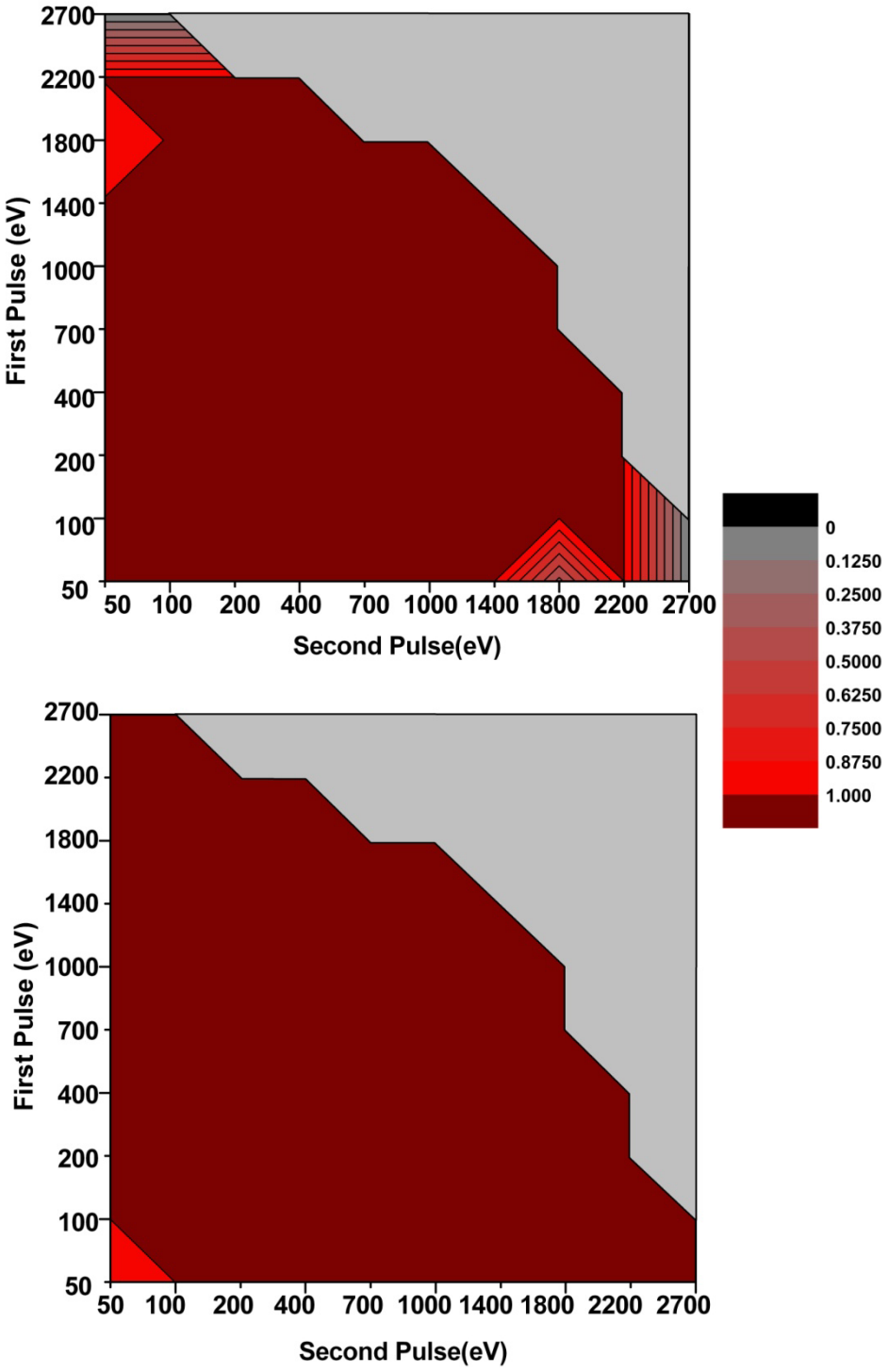
(Fig. 6.6)[24] *Top: Comparison between the resulting convolutions for first and second time derivatives of an average pulse and their respective optimum filters. Bottom: comparison of the convolution template and a double pulse convolution with pulse energies of 1000 eV and 100 eV and a separation of 1 μs.*



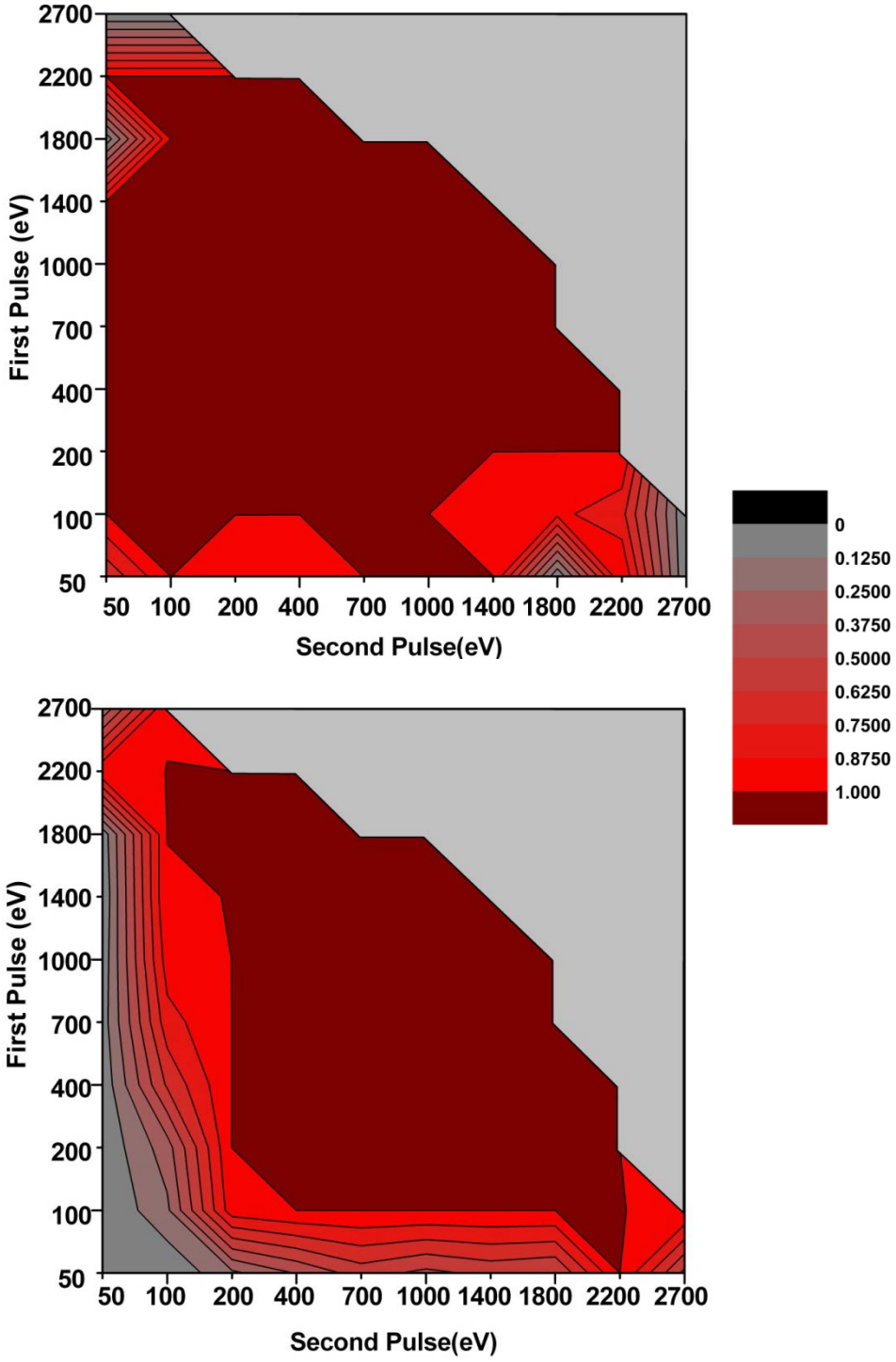
(Fig. 6.7)[5D] *Difference between the two curves from the bottom plot of fig. 6.6 with the lower signal result for a single pulse sweep. The threshold is the horizontal blue line.*

6.2.5 Results

Analysis using both algorithms resulted in similar results for the data sets with separation of 0.5 and 5 μs catching roughly nothing and everything respectively. The following results are thus restricted to the more interesting 1 and 2 μs sets. In the case of holmium, where the Q-value is somewhere in the 2.5 keV range,[5D] the efficiency of an algorithm is critical in this region (and similarly for rhenium). In the four contour plots (fig 6.8,9), this range corresponds roughly to the diagonal from the top left to the bottom right. In general the OF algorithm is much more sensitive to double pulses with a low energy contribution, but is not quite as efficient as the XQC algorithm in the critical region. In the next subsection of this chapter we explore improvements to the setup as well as combining the two algorithms to maximize efficiency.



(Fig. 6.8) *2 μs contour plots for the optimum filter (top) and standard algorithm (bottom). The grey region on the top right corners represents energy combinations not considered (above 2750 eV).*



(Fig. 6.9) *1 μs contour plots for the optimum filter (top) and standard algorithm (bottom).*

6.3.1 Current Results, Experimental Setup

The goal of this set of data and experiments is to perform the analysis as realistically as possible and to verify the effectiveness of the OF algorithm. Again for this setup, we use the same energy set to create double with separations of 1, 2, and 5 μs (excluding 0.5 now, with the same energy combinations, eq. 6.11). Double pulses with a combined energy above 2750 eV are excluded. The templates for both algorithms in the initial results were built from single valued calibration pulses. This is unrealistic and may result in an artificially increased efficiency and is undesirable. To correct this, a new set of calibration pulses were constructed with energies ranging from 50-2750 eV with 500 sweeps for every 100 eV increment. As before, each sweep consists of 4096 array elements with a sampling time of 1 μs. Here we consider parameter space described above for 2 eV and additionally for 5 eV resolution. Due to the spread out calibration pulses energies, we can also explore the effects of using different numbers of divisions.

To allow testing of worst case scenarios, 50 files for both energy resolutions were generated each with 12000 calibration pulses representing the entire energy range considered, 2190 double pulses (10 of each double pulse combination) , and 500 noise sweeps for a total of 14590 sweeps per file. The resulting double pulse fraction of about 15% is significantly higher than anything we would actually obtain from MARE experiments, but provides an excellent test for the efficiency of the algorithms and can definitely be considered a worst case scenario!

6.3.2 Optimum Filter Modifications

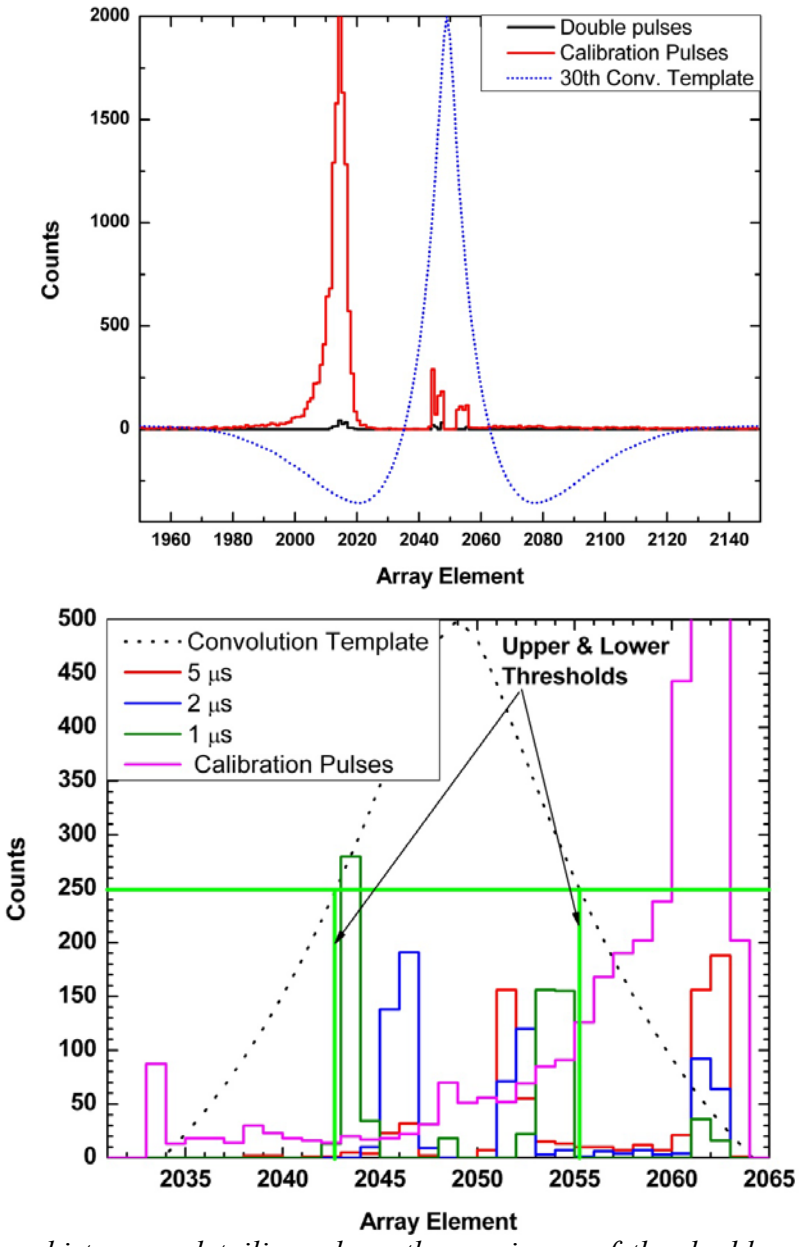
The original form of the OF algorithm results in different thresholds for each energy subdivision considered and uses a nonlinear measure (difference of squares). Ideally, we would like a measure that is linear and constant for all energy divisions. The benefit of using the original measure (eq. 6.13) was that it biased the results to values from the central peak of the convolution where it is most sensitive to rise-time double pulses. To get a similar effect with a linear measure, we now consider the measure:

$$m(c_T(t_n), c_S(t_n)) = \max\{c_T(t_n) - c_s(t_n)\}, \tag{6.16}$$

where the maximum is taken over a sub-region of the entire convolution. This will be referred to as the difference measure in this chapter. For the following development, we consider dividing the energy spectrum in to 30 subdivisions.

Our first task is to decide what subset of the domain to consider. An obvious starting point is to consider elements around the convolution curve before it goes to zero on both sides. We first would like to get a sense of where the new measure is maximal for calibration and double pulses. To do this, we plot a histogram detailing which array element, relative to the convolution template, the maximum value of $m(c_T(t_n), c_S(t_n))$ occurs. From the top plot of fig. 6.10, we see that most calibration pulses are triggered in the left lobe of the convolution. By restricting ourselves to the main central peak, we increase the relative difference between double and calibration pulses with double pulses tending to have larger measure values (since most calibration pulses don't have maximum measures in this range). In the second plot of fig. 6.10, we see that the calibration pulses are maximal on the right wing of the central convolution peak, whereas

most double pulses are maximal in the central region. This indicates that we should further restrict ourselves to the central FWHM region.



(Fig. 6.10) *Top: histogram detailing where the maximum of the double pulse measure occurs (for all divisions) relative to the main region of the convolution template. The convolution template included is for reference. Bottom: Similar histogram as top figure but with the domain restricted to the central peak. The upper and lower thresholds are set at half of the maximum of the convolution template.*

The construction of the convolution template involves lining up the pulses based on their software trigger. However, the software trigger is not perfect and has some amount of spread (see fig. 6.11). When averaging pulses this leads to a slight "blurring" of the final average so that it is slightly larger in shape when compared to a single pulse. Additionally, the presence of noise in the pulse being filtered its shape relative to that of the convolution template. The net result of these effects is a general hierarchy (not a rule) of the convolutions being considered (which will further be demonstrated shortly) for:

$$Good\ Pulse\ Convolution \leq Convolution\ Template \leq Double\ Pulse\ Convolution \tag{6.17}$$

which is reasonably true in general except for the noisier, low energy pulses where the left inequality is more often violated. Because of this general hierarchy, we do not consider the absolute value of the difference measure when testing sweeps to determine if they are double pulses. In general, we would like this hierarchy to be true of the difference measure as well. Suppose we know the cumulative density function for values from the difference between the pulse and convolution template, $N(x)$. If we set:

$$y = \max\{x_1, \dots, x_2\} \tag{6.18}$$

Then the CDF of y is:

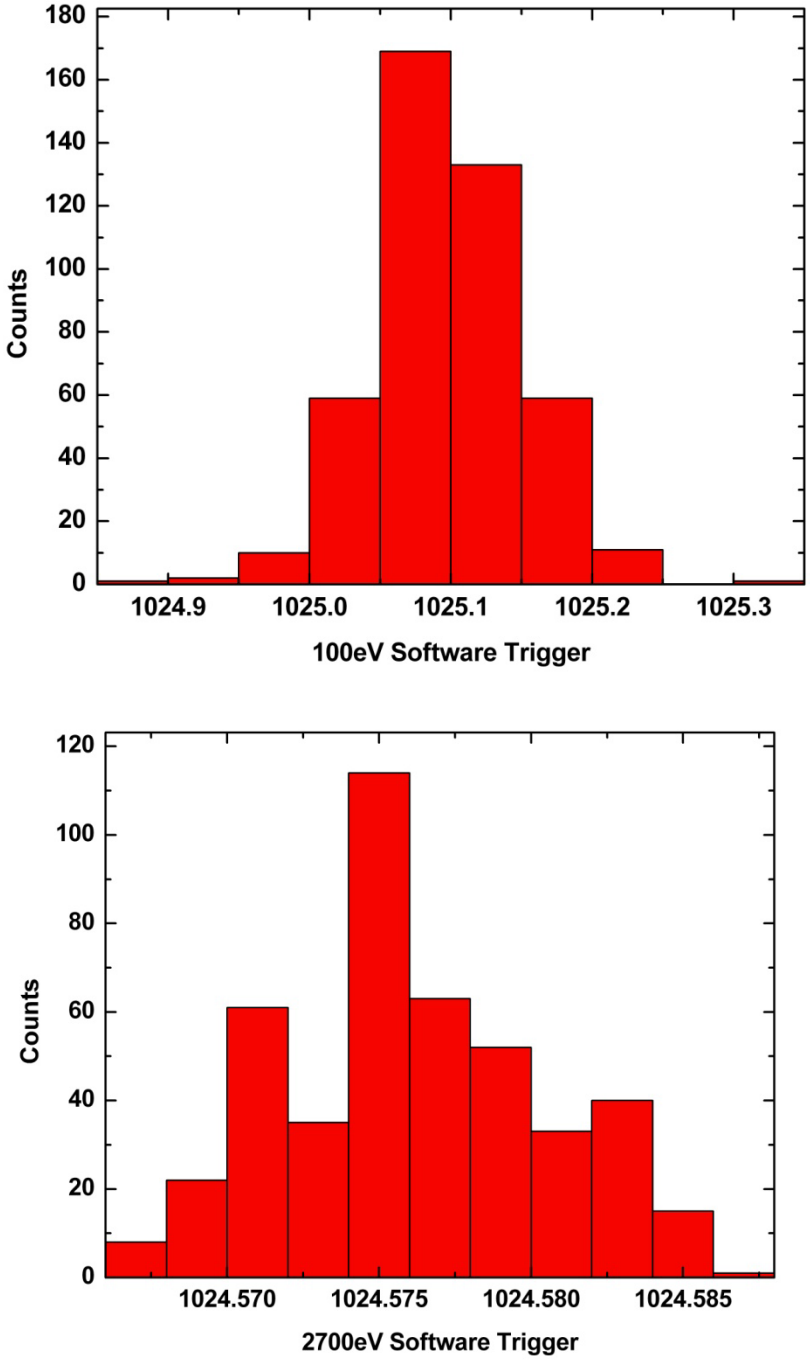
$$M(y) = \mathbb{P}(x_1 < y) \cup \dots \cup P(x_n < y) = N(x)^n \tag{6.19}$$

With a probability density function obtained from its derivative:

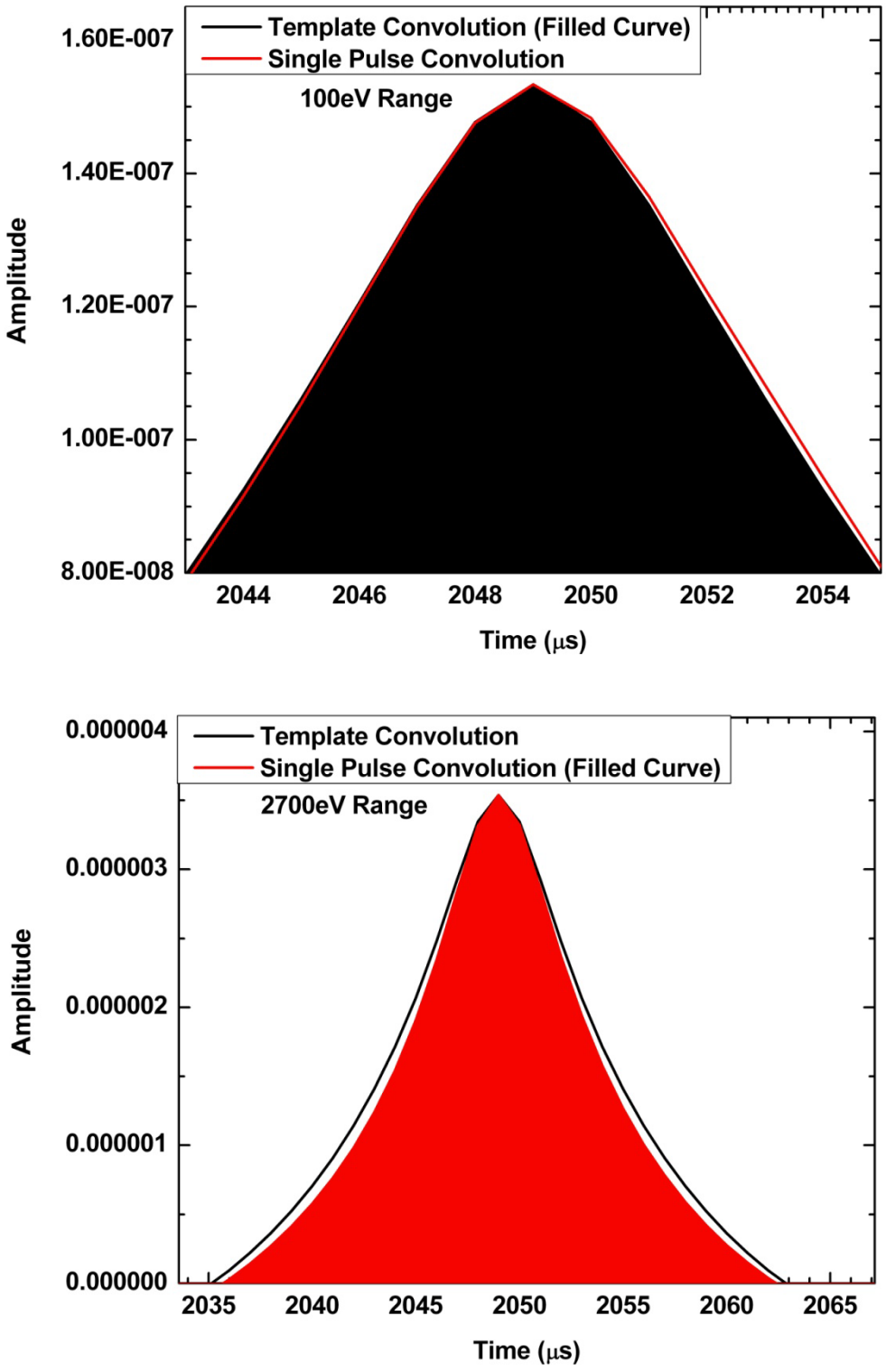
$$m(y) = nN(x)^{n-1}n(x) \tag{6.20}$$

Where $n(x)$ is the PDF of x. Fitting intuition, such a distribution tends towards higher values the as n gets larger. In the case of the difference measure, where a maximum of

the difference between a pulse convolution and a template convolution a similar effect takes place which hides difficult to resolve double pulses.



(fig. 6.11) *Histograms of the software trigger channel for the 5 eV energy resolution set at two different energies ranges (100 and 2700 ± 50 eV). Note the dramatically lower range for the higher energy plot.*



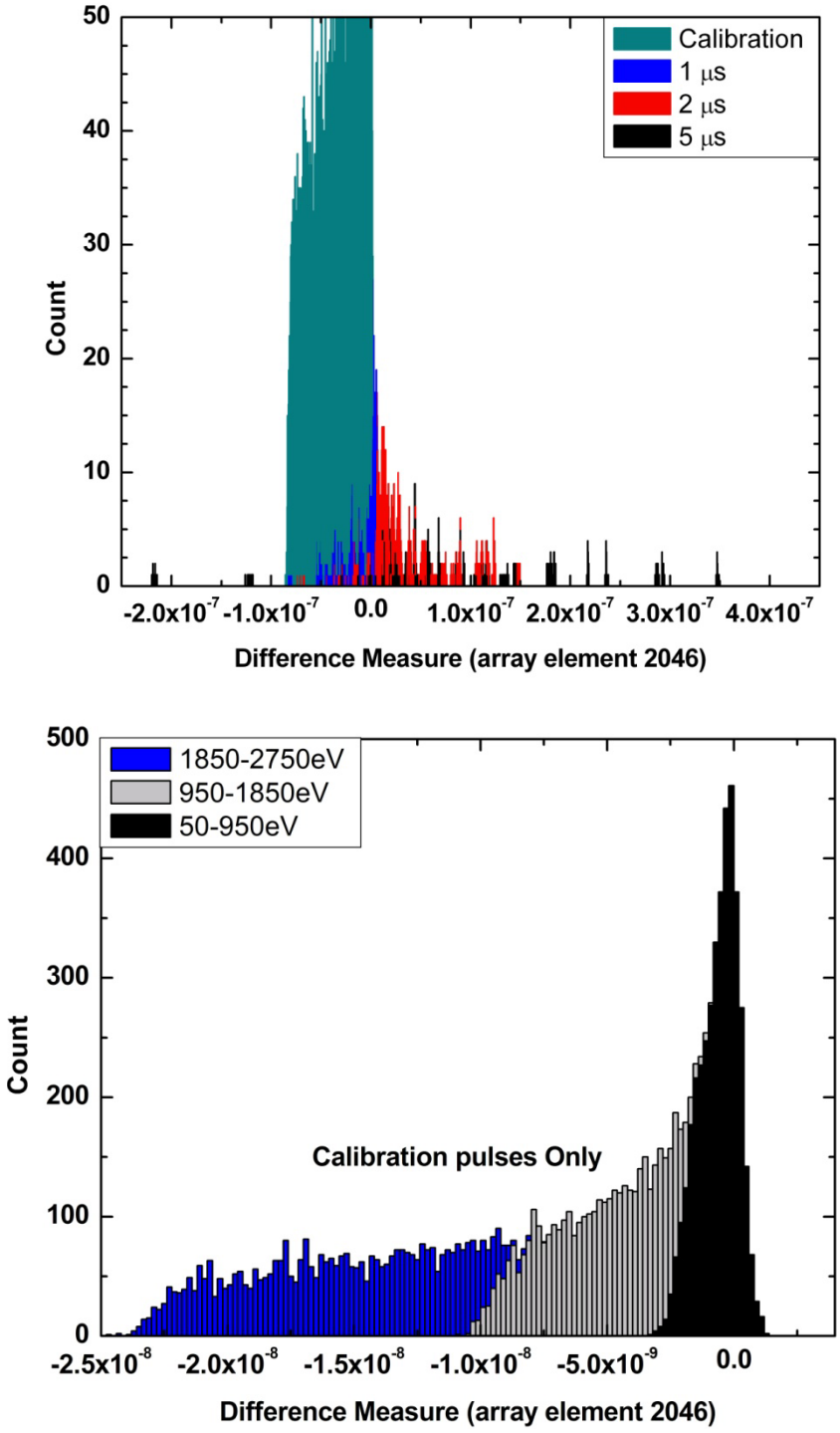
(Fig. 6.12) *Comparison of a good filtered pulse and the convolution template at the extremes of the energies being considered.*

To correct the problems involved in taking the maximum of a set of differences, we consider each array element in the domain being considered separately. This new difference measure can be thought of as a vector now:

$$\boldsymbol{m} = \{c_T(t_1) - c_s(t_1), \ldots, c_T(t_n) - c_s(t_n)\} \tag{6.21}$$

where the indices are restricted to array elements within the FWHM region of the central peak of the convolution template. Note that due to the variance in shape of convolution templates, some subdivisions will have more elements than others. In order to ensure consistent results and good statistics, we exclude regions where the number of measure values is less than 20% of the total number of pulses being analyzed. An example of histograms from array element 2046 is available in figure 6.13 below which demonstrates the rough hierarchy described previously.

Now that we have an acceptable double pulse measure, all that remains is to establish a method for obtaining a threshold. In general it would be ideal to establish a single value to use for the histogram associated with each array element considered, but each spectrum has a slightly different shape. Note how sharply the calibration pulse spectrum ends near zero (fig. 6.13). This feature is common to each spectrum and can be taken advantage of. To obtain a threshold, we locate the bin with the highest count rate (which is the calibration peak near zero) and then locate the first bin to the right where a count that is less than 10% of the maximum. The results using this threshold have very good efficiency and a false positive fraction around 0.5%.



(Fig. 6.13) *Histograms showing the distribution of the double pulse measure for array element 2046. Top: Note how the calibration peak tends towards values below zero whereas most double pulses have positive measure values. Bottom: the calibration pulse spectrum was broken up into its three subdivisions to demonstrate how higher energies have lower values.*

Note that in obtaining the threshold above, where it was set by locating the first bin to the right of the maximum that is less than 10% of this maximum, this percentage is artificially high due to the large double pulse fraction in each double set. In general this fraction should be closer to 1%. In the case where the double pulse threshold is set to zero, there is a significant number of false positives (around 15%) but are strictly caused by low energy pulses. In cases where the energy spectrum being analyzed does not have a high density of low energy pulses, or where there is no concern for low energy pulses (as is the case with MARE) this threshold may be ideal. From the analysis performed on this double pulse algorithm, we determined that it has a very low false positive fraction. In principal this can help when deciding how to set the double pulse threshold for an experiment with real data. Assuming the double pulse density is fairly low (as with MARE), any threshold that results in a false positive fraction higher than 1% (or lower percentages) should definitely be rejected.

6.3.3 Updated Algorithms

Because of the more realistic data format used, modifications to the algorithms originally described in the initial results subsection. Here the new algorithms are listed with the altered portions in bold font for convenience.

Updated Standard (XQC) Algorithm:

- Calculate d_t-noise for good noise sweeps, compute the associated RMS value, and average this for all noise sweeps: σ;

- The threshold, θ, is set as $2^a N\sigma$, where a is initialized with a value of 4, and N is the minimum multiple of σ to use as a threshold (for all analysis we use $N = 5$);
- All good d_t-noise sweeps are then scanned for excursions above θ;
- If any such excursions are found their associated noise sweep is reclassified and the process repeats in order to calculate a "cleaner" value for σ;
- If no such excursion are found, then a is reduced by 1;
- The process is repeated until $a = 0$ in which case, we have the threshold: $\theta = N\sigma$.

Now that we have established a threshold, we analyze each energy division separately as follows:

- Average all good sweeps (shifted to line up based on their software trigger) and calculate its time derivative to build a template: T;
- Each sweep being analyzed is shifted (again, based on the software trigger) and its time derivative is taken giving us: P_i;
- The template is normalized so that its maximum corresponds to that of P_i, and we take their difference;
- **This difference is scanned for excursions beyond the threshold, $2^a N\sigma$;**
- **If any excursions are found, they are reclassified, the template is reconstructed from the remaining good pulses, and the process repeats**
- **If no excursions are found in any of the sweeps being analyzed, then a is reduced by 1;**
- **The process is repeated until $a = 0$ in which case, we have the threshold: $\theta = N\sigma$.**

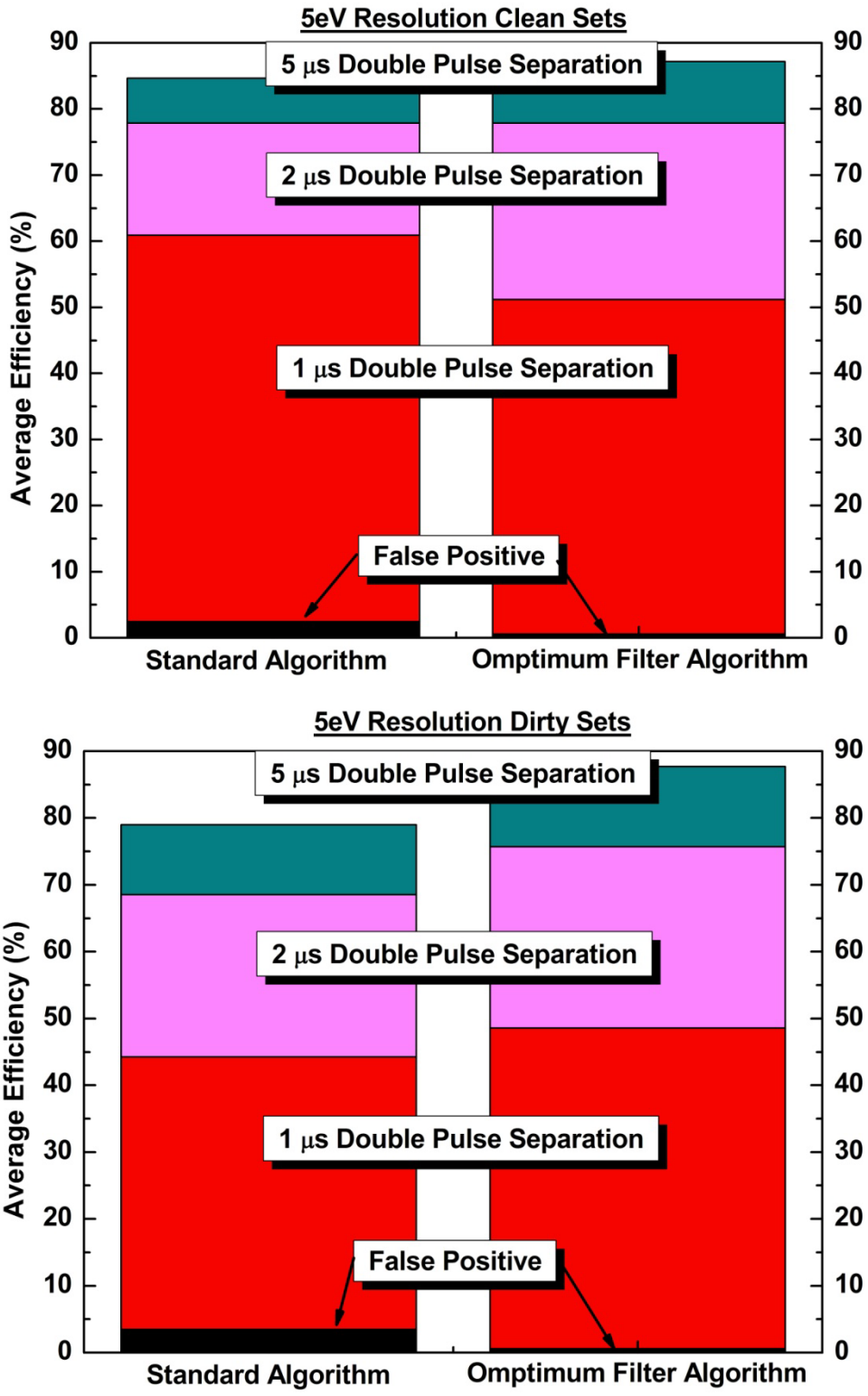
Updated Optimum Filter (OF) Algorithm:

- Average good pulses in the time domain, calculate its second time derivative and transform to the frequency domain: $\tilde{P}_{tt,ave.}(f_n)$;
- Compute d_{tt}-noise and average in the frequency domain: $\widetilde{N}_{tt,ave.}(f_n)$;
- Create the optimum filter as: $\tilde{F}_{opt}(f_n) = \frac{\tilde{P}_{tt,ave.}(f_n)}{\widetilde{N}_{tt,ave.}(f_n)}$;
- Create the convolution template as: $\tilde{C}_T(f_n) = \left(\tilde{F}_{opt}(f_n)\right)\left(\tilde{P}_{tt,ave.}(f_n)\right)$, and convert this to the time domain: $C_T(t_n)$;
- Normalize $C_T(t_n)$ so that its maximum is 1: $c_T(t_n)$;
- For each pulse, $\tilde{P}_{tt,i}(f_n)$, to be analyzed, create a template as: $C_{p,i}(t_n) = \left(\tilde{F}_{opt}(f_n)\right) * \left(\tilde{P}_{tt,i}(f_n)\right)$ (this is done in the frequency domain as with the template above);
- Normalize $C_S(t_n)$ so that its maximum is 1, and shift so that its peak corresponds to that of the template convolution: $c_S(t_n)$;
- **Calculate: $\boldsymbol{m}$ (eq. 6.21)**
- **Create a histograms from the resulting $\boldsymbol{m_i}$ values to obtain a threshold**
- **Pulses corresponding to an $\boldsymbol{m_i}$ above this threshold are classified as double pulses**

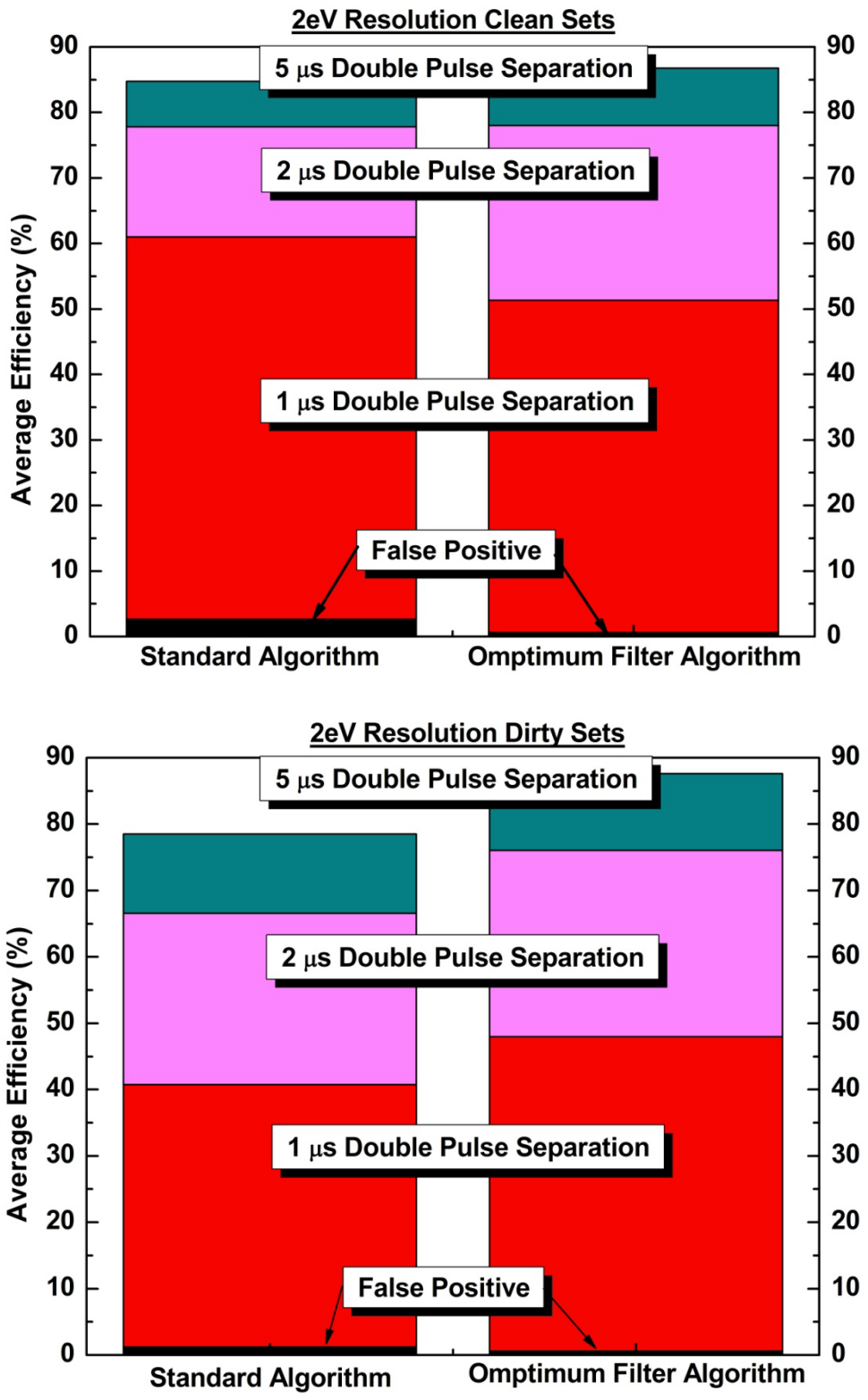
6.3.4 Results: Effects of Double Pulse Contamination

As noted earlier, it was expected that the single valued energy pulses used to construct the templates for the initial results would artificially increase the efficiency of the two algorithms considered. This expectation is verified here. To aid in readability the resulting contour plots are summarized via their average efficiency (over the double pulse amplitudes considered) and their false positive fraction. All contour plots relating to this data may be found in the appendix.

Here we consider the effects of double pulse contamination on the efficiency by dividing analysis into two categories: clean and dirty. In clean sets, only calibration pulses are used to construct the double pulse algorithm templates, whereas for dirty sets, no distinction is made between double and calibration pulses (this is a worst case scenario approximation of real data). The left sides of the plots represent results from the XQC algorithm and the right side represents results from the OF algorithm. We see that in general, the OF algorithm is much less sensitive to the presence of double pulse contamination than the XQC algorithm. For clean sets, they perform similarly except for the 1μs separation double pulses and lower false positive fractions where the XQC algorithm outperforms for the former and the OF algorithm outperforms in the latter (a lower double pulse threshold for the OF algorithm would close this gap by increasing its false positive fraction and low energy double pulse efficiency).



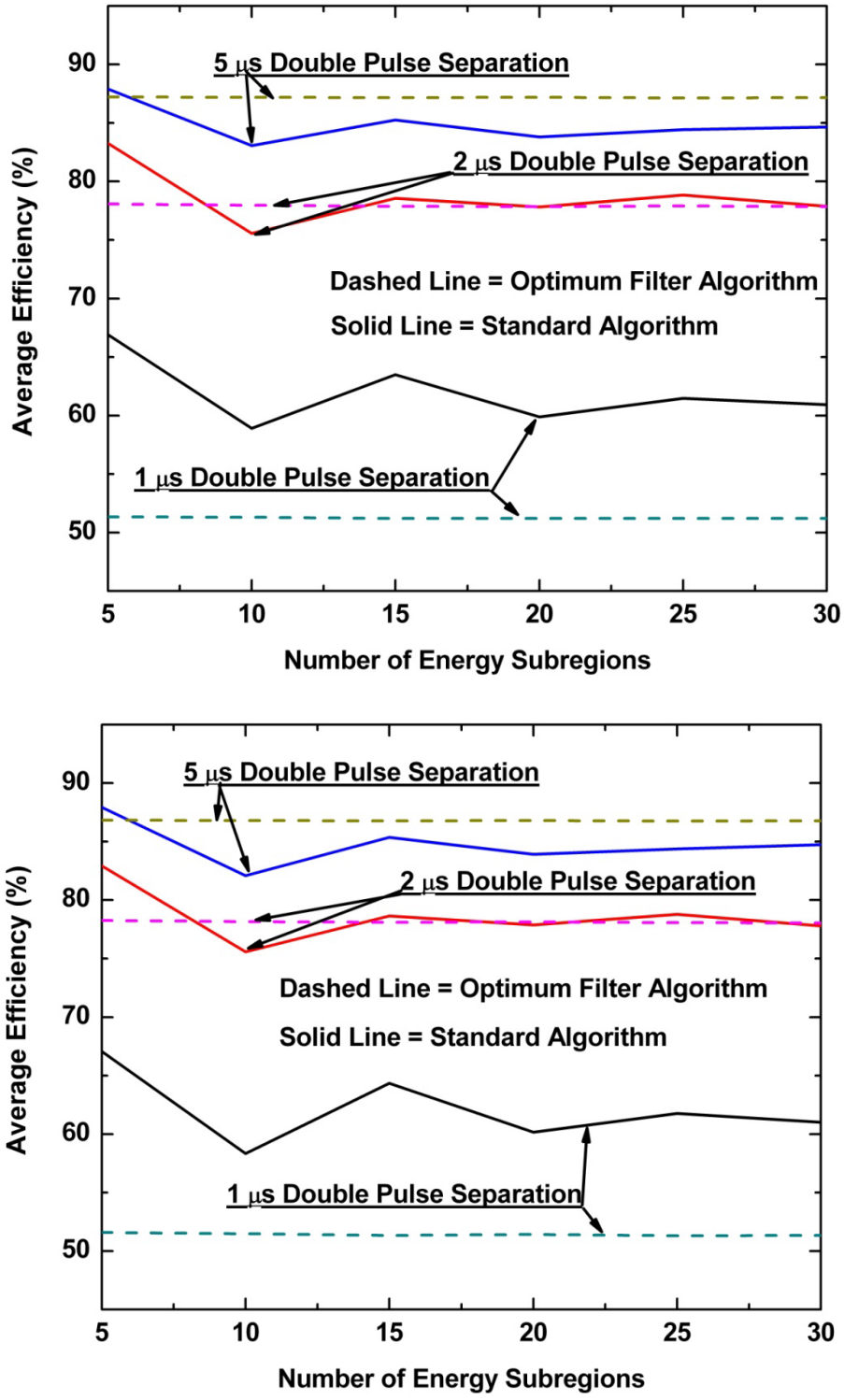
(Fig. 6.14) *Summary of results for clean and dirty sets and 5 eV Energy Resolution.*



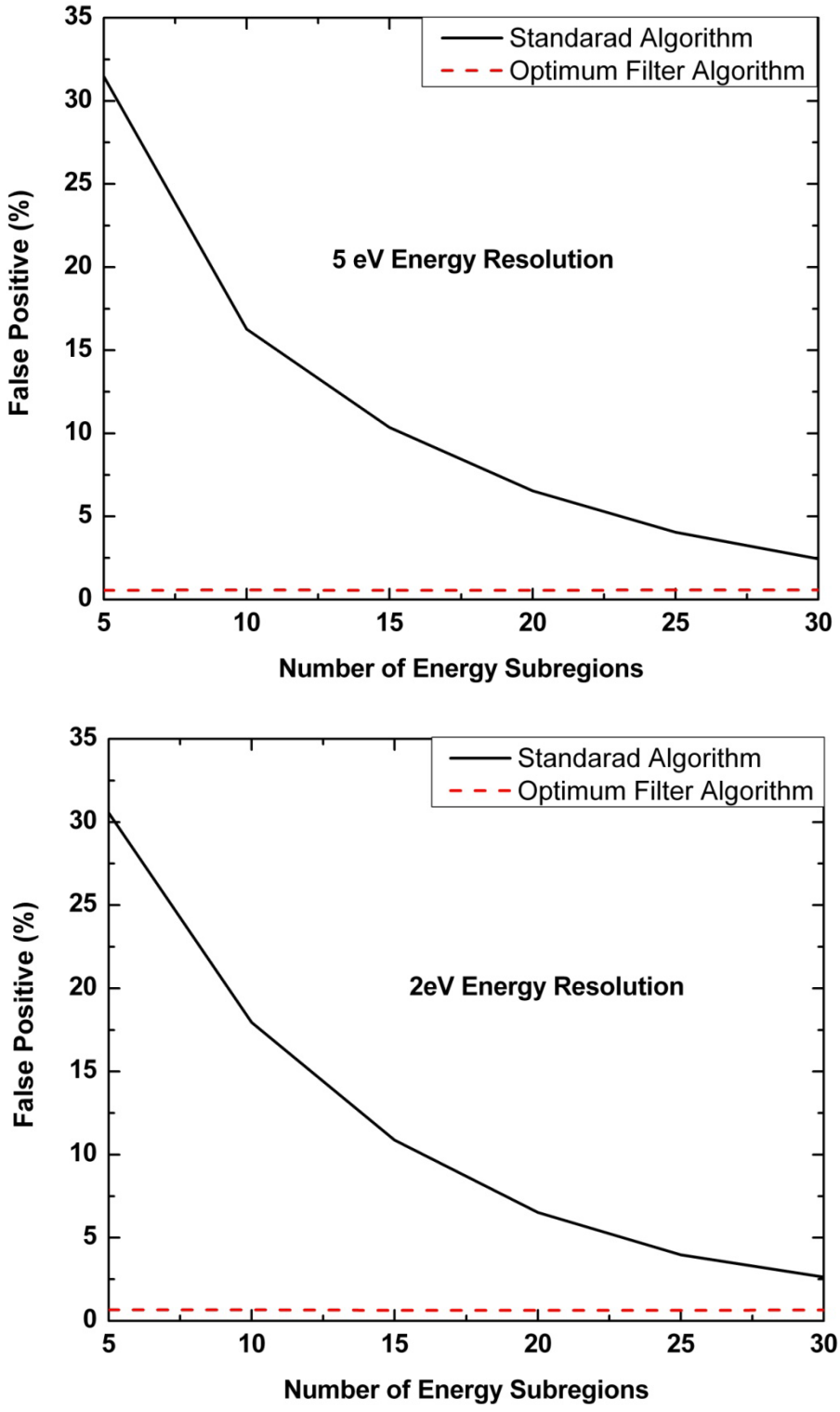
(Fig. 6.15) *Summary of results for clean and dirty sets and 2 eV Energy Resolution.*

6.3.5 Results: Effects of Detector Non-linearity

Because of the spread out distribution of calibration pulses used, the number of energy subdivisions becomes an additional parameter to explore. Here, we consider only clean sets with 5,10,15,20,25, and 30 energy subdivisions (equally spaced). From fig. 6.16, we see that overall, both algorithms are fairly stable over the different numbers of energy subdivisions considered with little difference between the 2 and 5 eV energy resolution sets considered. Differences do appear though, when we look at how the false positive fraction depends on the number of energy divisions (fig 6.17). Of particular note is that the optimum filter algorithm is extremely stable whereas the false positive fraction of the standard algorithm increases quickly as the number of energy divisions is reduced. This is largely due to the way in which the algorithms are constructed. By construction, the OF algorithm was made to be sensitive to the sudden changes in voltage associated with a pulse. Recall that for a single pulse, there are only two such regions that contribute strongly to the second derivative. Other regions do not contribute as strongly making this algorithm very sensitive to the separation and sharpness of these two regions and with little sensitivity to the overall pulse shape (which changes due to the nonlinearity and to which the standard algorithm is more sensitive).



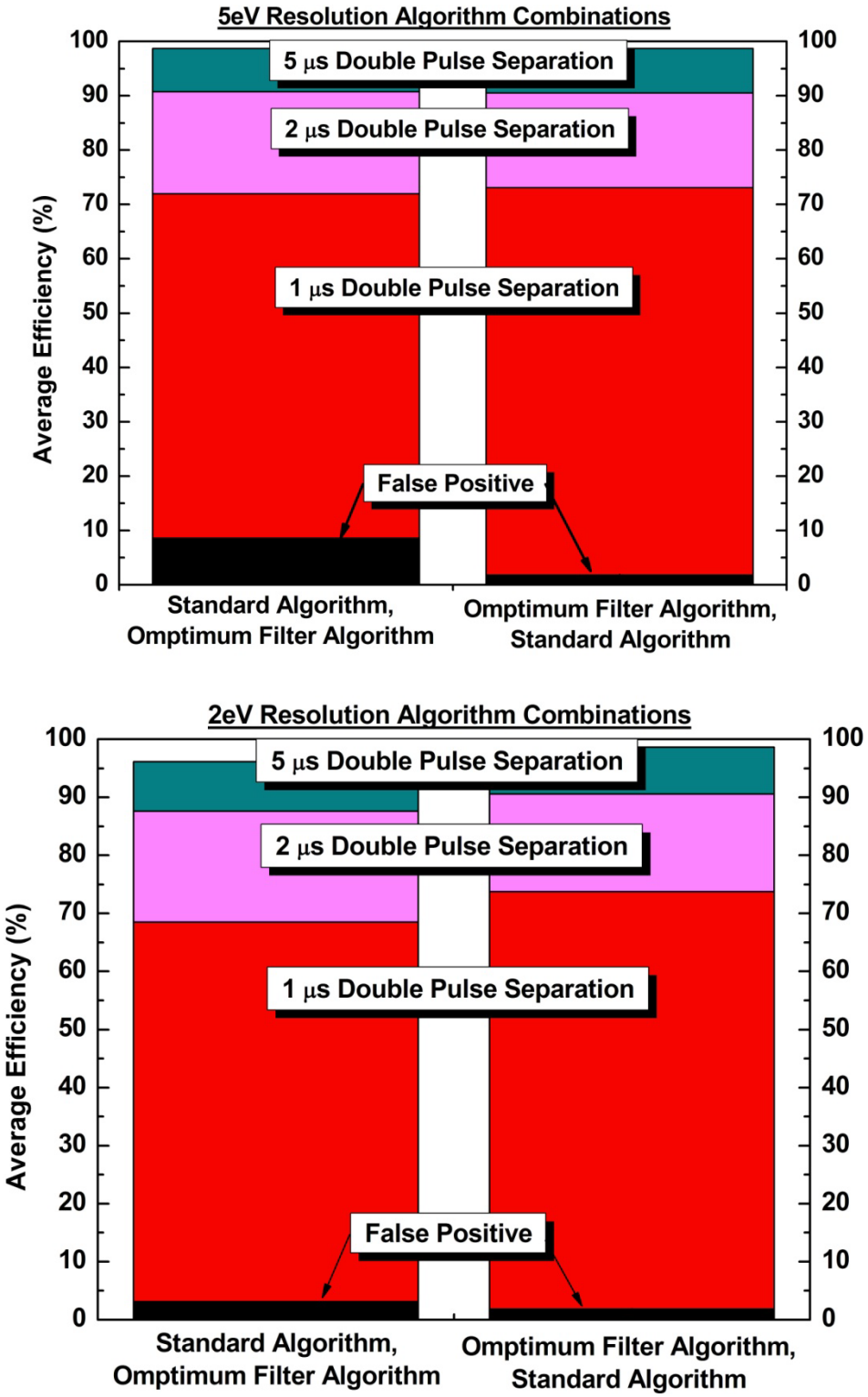
(Fig. 6.16) *Comparison of efficiency for both double pulse algorithms at 5 eV (top), and 2 eV (bottom) energy resolution.*



(Fig. 6.17) *Comparison of false positive fraction for both double pulse algorithms at 5 eV (top), and 2 eV (bottom) energy resolution.*

6.3.6 Results: Combining Both Algorithms

Here we explore the efficiency resulting from combining both algorithms (in both orders) on dirty sets. In general, their contour plots show a complimentary efficiency where the XQC algorithm is more sensitive to low energy second pulses and the OF algorithm is more sensitive to low energy first pulses (see appendix for contour plots). Because of this, the resulting efficiency is dramatically increased and become comparable to the initial results. From fig. 6.18, we see that the best results are obtained when starting with the OF algorithm. This is due in large part to its reduced sensitivity to the presence of double pulse contamination in its template. Comparable results are obtained in the reverse order, but are slightly less efficient with a higher false positive. The strong difference in false positive fractions is caused by the double pulse threshold set in the OF filter. When double pulses are removed by first running the XQC algorithm, subsequent analysis with the OF algorithm results in a threshold that is a little too low. This can easily be changed, but would result in an even lower efficiency relative to the reverse order for the algorithms.



(Fig. 6.18) *Summary of the results from combining the two algorithms in both orders for 5 eV (top) and 2 eV energy resolution. The left column represents first running the XQC algorithm then following with the OF algorithm. The right column represents the reverse order.*

6.4 Summary

Our initial results were very positive, but more realistic analysis is needed. After modifications of the experimental setup and the optimum filter algorithm, we were able to analyze both algorithms and obtain more accurate results. While some care must be taken with setting the threshold for the OF algorithm, it is very tough and stable in the presence of high double pulse template contamination and is highly insensitive to detector nonlinearity. The XQC algorithm on the other hand, is sensitive to detector nonlinearity, but does comparatively better when there is very little double pulse contamination in its template. In general, we have found that we can achieve very good efficiency for rise-time double pulse detection with both algorithms separately, but the best results are obtained by running the two algorithms in series which results in almost perfect detection for 5 μs separation double pulses and excellent efficiency for the other two separations considered. This reduces the effective rise-time used ultimately results in a higher sensitivity for any calorimetric experiment.

Appendix A

FITS File Details

A problem that often arises in custom created software created in academia is that the File format used is custom made for the specific purposes of the project (and thus unfamiliar to anyone else). This is acceptable but causes problems when collaborating with others. In order to facilitate collaboration a standard File format is preferable. We chose the FITS (Flexible Image Transport System) because it is well documented and has a library available (CFITSIO) which is compatible with FORTRAN code

The FITS File format was developed in the 1970's to provide a standard for data exchange between astronomical observatories. Since that time FITS has become the standard File format for most astrophysical data analysis software packages. FITS Files are comprised of a Primary Header Data Unit (HDU) along with a header. The Primary HDU contains an N-dimensional array of pixels which may be null (which is the case for all of our FITS Files). To this, additional HDUs and their associated headers may be added (called extensions). These extensions may be of three types:

- Image Extension: N-dimensional array of pixels (as with the Primary HDU)
- ASCII Table Extension: Rows and columns of data in an ASCII character format
- Binary Table Extension: Row and columns of data in binary form

The header of each HDU may have any number of comments (up to 80 characters), or header entries (each with a descriptive comment up to 80 characters long) which may be either integers, floating point numbers, character strings, or logical (Boolean), each denoted by a keyword (8 characters, all in caps). The following are some of the main

FITSIO calls used in FITSFILTER (all arguments to the right of the ">" are returned by the routine). The "status" parameter is used as an error check for FITS routines. It is initialized as zero and returned as some other value depending on what type of error occurred.

[xbijkefdgcmls]: This is not a routine. Some routines have different names based on the type of parameter they read/write. The characters stand for:

x bit

b character*1 (unsigned bit)

i short integer (I*2)

j integer (I*4, 32-bit integer)

k long long integer (I*8, 64-bit integer)

e real exponential floating point (R*4)

f real fixed-format floating point (R*4)

d double precision real floating point (R*8)

g double precision fixed-format floating point (R*8)

c complex reals (pair of R*4 values)

m double precision complex (pair of R*8 values)

l logical (L*4)

s logical (L*4)

Routines to Create a FITS File

- **FTGIOU(> unit, status):** This routine returns an unused integer (unit) which may be used to open a FITS File.
- **FTMAHD(unit, nhdu, > hdutype, status):** Assuming the unit is associated with an open File, this routine moves to a selected HDU (nhdu, starting with 1 for the Primary HDU) and returns its type (not used for our purposes).
- **FTCOPY(iunit, ounit, morekeys, > status):** This routine is used to copy the current HDU from iunit to the current (necessarily empty) HDU of ounit. The morekeys parameter is not used for our purposes, but allows additional space for extra keywords.
- **FTPKY[JKLS](unit, keyword, keyval, comment, > status):** This routine adds a new keyword to the current HDU associated with the open unit along with a comment up to 80 characters long.

- **FTPKY[JKLS](unit, keyword, keyval, decimals, comment, > status):** Same as above with an additional parameter, decimals, as the integer number spaces to the right of the decimal to use.
- **FTICOL(unit, colnum, ttype, tform, > status):** Each column in a FITS file HDU must be initialized by ins column number, type, and form. The parameter ttype declares the name of the column (up to eight characters) and tform declares the variable type and length. For example a real exponential floating point array

of size 2048 would be declared as a character string: ‘2048E’; whereas a single integer entry would be decalared as a character string: ‘1I’.

Routines to Read a FITS File

- **FTGCV[SBIJKEDCM](unit, colnum, frow, felem, nelements, nullval, > values, anyf, status):** This routine reads the data from a given row (frow) and column (colnum) from the current HDU and return its value. In most cases this will be a single parameter, but sometimes an array is read (nelements = ‘size of array to be read’)
- **FTOPEN(unit, Filename, rwmode, > blocksize, status):** This routine open an existing FITS Files with either readonly (rwmode = 0) or readwrite (rwmode = 1) access. The returned blocksize argument is obsolete and should be ignored.
- **FTGKEY(unit, keyword, > value, comment, status):** This routine is used to read a given keyword from the current HDU and returns its value and associated comment. Note that this routine does not have different forms associated each type of possible variable format, so care must be made to know in advance what type of keyword you are reading.

Routines to Edit a FITS File

- **FTGCNO(unit, casesen, coltemplate, > colnum, status):** This routine searches the current HDU for a data column with a given name (coltemplate), and returns it

number (starting with 1 from the leftmost column). This routine is used extensively throughout FITSFILTER so that no assumptions need to be made about what data is in a given column (as long as one exists with the given name). This allows additional columns to be inserted in the future without requiring any rewriting of the code.

- **FTMKY[JKLS](unit, keyword, keyval, comment, > status):** It is possible to have multiple entries in a given HDU with the same keyword name. Caution must be taken to ensure that any keyword is created in an HDU only once. To modify a given keyword, this routine must be called instead.
- **FTMKY[EDFG](unit, keyword, keyval, decimals, comment, > status):** Same as above with an additional parameter, decimals, as the integer number spaces to the right of the decimal to use.
- **FTPCL[SLBIJKEDCM](unit, column, frow, felem, nelements, values, > status):** This is the workhorse of FITSFILTER and is used to write data to the unit associated with the current HDU. Colnum and frow are the column and row number for the entry to be modified. Nelements declares the integer size of the array, usually just one unless in contains a sweep array in which case it is a power of 2. This call has a useful property that if the row does not exist, it will create it along with any missing intermediate rows.
- **FTGKY[EDJKLS](unit, keyword, > keyval, comment, status):** This routine reads a keyword from the header of the current HDU and returns its value in keyval along with its associated comment (usually ignored for out purposes).

The above list is not exhaustive and some lesser used calls occur in FITSFILTER. To find a complete list and description of cfitsio routines, you may visit http://heasarc.gsfc.nasa.gov/fitsio/ and download the "Fortran Programmer Reference Guide".

The following details cover the files used in fitsfitlter:

Files:

- ***_SWP.FITS (SWP File)**: This is the main input File for FITSFILTER and contains important data in the header as well as all of the sweeps and related information for at least one pixel.
- ***_PUL.FITS (PUL File)**: This File is created by FITSFILTER and holds parameters calculated from the sweeps of analyzed pixels such as rise-time, rough amplitude, and trigger channel.
- ***_FIL.FITS (FIL File)**: This File holds templates created for each pixel analyzed. It saves many unused templates for future reference. The main purpose of this File is to save the filter template, which maximizes the signal to noise ratio. With the aid of a calibration peak, the energy of each sweep is extracted.
- **PARAMETER.FITS**: This File holds many customizable parameters used in FITSFILTER (especially if it is run in auto mode). This File allows the user to change parameters without editing the code and recompiling.

***_SWP.FITS (SWP File)**:

This File is the output of trigger (or other File type conversion programs we have created for various purposes) and is where most of the content of the subsequent PUL and

FIL File are obtained. The '*' indicates the base name of the File which will be used for the associated PUL and FIL Files. Note that when using FV, the HDUs are indexed starting from 0 rather than 1, which is the convention used when moving between HDUs with the ftmahd call. The header keywords listed are strictly user added and required for FITSFILTER to read it properly. Any support keywords which are automatically created for the FITS File are ignored (including the keywords associated with the binary data column declarations). To avoid confusion, note that the columns are listed in rows – this is strictly to allow the entries to be more readable and does not imply that columns have been switched with rows on the FITS Files. The SWP File has the following format:

Primary HDU (HDU 1)
Header: Empty
Image: Empty

HDU 2
Header:

- SWPLEN: array size of each sweep
- DELTIME: timescale
- DELVLT: voltage scale
- NCHST: trigger channel
- NSWEEP: total number of sweeps (from all pixels on File)
- PIXCOUNT: total number of pixels on the SWP File

Binary Data Columns (one row for each pixel):

Column	Name	Type	Units
1	PIXNUM	1J	None
2	NSWEEP	1J	None
3	ACGAIN	1E	None
4	SQUID	1E	None
5	ELCF	1E	Hz
6	UCF	1E	Hz
7	ELP	1J	None
8	UCP	1J	None
9	FWIN	1E	None

HDU 3
Header: Empty

Binary Data Columns (number of rows is equal to NSWEEP keyword in HDU 2):

Column	Name	Type	Units
1	Pixel_Number	1J	None
2	Pixel_Count	1J	None
3	PULse_Time	1D	sec_of_day
4	Flag	1J	bits
5	PULse_Data	(SWPLEN)E	ampout_volts

(Table A.1) *Sweep file format.*

The first HDU is left empty, since it cannot contain a binary table. The first extension (HDU 2) contains parameter data for each pixel. Originally many of these values were strictly in the header, but in order to increase flexibility and allow for different pixels to have different parameter values associated with them, they were converted to a table. The columns names and descriptions are as follows:

1. **PIXNUM:** the pixel associated with the row. This is needed since some formats used have pixels unevenly distributed over more than one File resulting in skips in the list (e.g. pixels; 1,2,3,4,5,6, 13,14,15,16,17,18 may occur in order with 7,8,9,10,11,12 on a different File)
2. **NSWEEP:** the total number of sweeps for the associated pixel
3. **ACGAIN:** the DC gain (amplification) applied to the signal before it is digitized.
4. **SQUID:** the amplification of the signal from the SQUID and occurs before the gain is applied. When used in subroutine of FITSFILTER, the two are multiplied (the null entry should be 1, but this is corrected if it is set to 0 for some reason) and stored in the subroutines acgain parameter.

5. **ELCF:** the lower corner frequency of the bandpass filter response due to the amplifier.
6. **UCF:** the upper corner frequency of the bandpass filter response due to the amplifier. Obviously, we expect UCF > ELCF.
7. **UCP:** the pole (power) of the upper corner frequency and generally has a value of 1 or 2 (0 if no upper corner frequency is given).
8. **LCP:** the pole (power) of the lower corner frequency and has similar (though not necessarily equal) values as UCP.
9. **FWIN:** a parameter used when windowing a sweep. It is a real number between 0 and 0.5 and indicates the fraction of each side that will be attenuated.

HDU 3 posses the bulk of the data because it contains the actual sweeps for each pixel associated with the File. For data that is hardware triggered, there is no particular order for the pixel associated with each successive sweep. This requires additional columns for proper 'book keeping'. The columns names and descriptions are as follows:

1. **Pixel_Number:** the pixel number for which all data in the row is associated with.
2. **Pixel_Count:** the count (starting with 1) of the number of sweeps for a given pixel.
3. **PULse_Time:** the time associated with the given pixel.
4. **Flag:** a hardware flag which is not currently used for the main functions of FITSFILTER.

5. **PULse Data:** the sweep associated with current pixel. This contains all triggered including ones that were randomly triggered to collect noise.

All together the SWP File can become quite large and bulky. It would be convenient to possess a 'lighter', more portable File that contains characteristic values for each sweep rather than the sweep itself. This is exactly what the analysis performed by FITSFILTER does and the resultant File is called a PUL File.

***_PUL.FITS (PUL File)**

This File, associated with the input SWP File and shares the same base name. If this File does not exist, it will be created upon running FITSFILTER with HDU 2 of the SWP File copied to HDU 2 of the PUL File along with additional columns. Using the same conventions as with the SWP File, the PUL File has the following format:

HDU 1
Header:

- FLOWPASS: Lowpass filter applied in FILCHR.F
- THRESH1: Fraction of max. to use as a threshold in GNUSEL.F
- THRESH2: Fraction of max. to use as a threshold in GNUSEL.F
- THRESH3: Fraction of max. to use as a threshold in GNUSEL.F
- THRESH4: Fraction of max. to use as a threshold in GNUSEL.F
- THRESH5: Fraction of max. to use as a threshold in GNUSEL.F
- THRESH6: Fraction of max. to use as a threshold in GNUSEL.F
- CALEN: Calibration energy for K-alpha
- NSIGMA: Multiple of sigma used for min threshold in DBLFIL.F
- LPPOLE: Pole of lpfilter applied to the optimum filter
- LPFREQ: Frequency of lpfilter applied to the optimum filter
- A: Gain drift correction:
- B: Gain drift correction:
- C: Nonlinear correction:

- D: Nonlinear correction:
- WINFRAC: Parameter used in window subroutine (utility.f)
- PRTRG: Fraction used to determine ilzero for pretrigger
- TRIGGER: F: use hardware trigger; T: use software trigger

Image: Empty

HDU 2
Header:

- SWPLEN: array size of each sweep
- DELTIME: timESCALE
- DELVLT: voltage scale
- NCHST: trigger channel
- NSWEEP: total number of sweeps (from all pixels on File)
- PIXCOUNT: total number of pixels on the SWP File

Binary Data Columns (one row for each pixel):

Columns	Name	Type	Units
1	PIXNUM	1J	None
2	NSWEEP	1J	None
3	ACGAIN	1E	None
4	SQUID	1E	None
5	ELCF	1E	Hz
6	UCF	1E	Hz
7	ELCP	1J	None
8	UCP	1J	None
9	FWIN	1E	None
10	NCHST	1E	
11	LOWPASS	1E	
12	NMAX	1J	
13	RTUL	1E	
14	RTLL	1E	
15	RNPKUL	1E	
16	RNPKLL	1E	
17	BASEUL	1E	
18	BASELL	1E	
19	TRIGUL	1E	
20	TRIGLL	1E	
21	ZERLIM	1E	
22	DBASEL	1E	
23	SUMFIL	1E	
24	CALEN	1E	
25	CONVFACT	1E	
26	STNR	1E	
27	FWHM	1E	

28	SUMSIG	1E	
29	ELC_A	1D	
30	ELC_B	1D	
31	ENLC_C	1E	
32	ENLC_D	1E	
33	FILTPROG	1J	
34	WINSTART	1E	
35	WINEND	1E	
36	prtrg	1E	
37	winfrac	1E	

HDU 3

Header:

- SWPLEN: array size of each sweep
- DELTIME: timescale
- DELVLT: voltage scale
- NCHST: trigger channel
- NSWEEP: total number of sweeps (from all pixels on File)
- PIXCOUNT: total number of pixels on the SWP File

Binary Data Columns (number of rows is equal to NSWEEP keyword in HDU 2):

Columns	Name	Type	Units
1	Pixel	1J	None
2	Sweep_Num	1J	None
3	Pixel_Num	1J	None
4	Pusse_Time	1D	sec_of_day
5	Flag		bits
6	yinit	1E	V
7	rawppk	1E	V
8	rawnpk	1E	V
9	dbase	1E	V
10	dectim	1E	V
11	ristim	1E	V
12	tfb	1E	V
13	lpamp	1E	V
14	iclass	1I	Class
15	Eraw	1E	eV
16	Egdc	1E	eV
17	Enl	1E	eV
18	xmax	1E	sec

(Table A.2) *PUL file format.*

The header of the primary HDU consists of elements copied from the PARAMETER.FITS File and will be covered shortly in that section. HDU 2 copied exactly from HDU 2 of the SWP File with lots of additional columns. All of the elements in each row can be considered as characterizing the entire sweep data set for their associated pixel whereas the binary data array in HDU 3 contains elements that characterize each sweep individually. The columns names and descriptions are as follows (excluding those covered in the description of the SWP File):

10. **NCHST:** the trigger channel indicates the array element where the signal first exceeded a threshold i.e. triggered. It is technically an integer and starts out as so, but may be replaced by the peak (based on a histogram) of the software triggers calculated in FILCHR.F
11. **LOWPASS:** the lowpass frequency of a lowpass filter applied to each sweep in FILCHR.F to increase the signal to noise ratio.
12. **NMAX:** the sweep array element where the maximum of the average of all sweeps occurs.
13. **RTUL:** upper limit on lpamp in k-alpha window.
14. **RTLL:** lower limit on lpamp in k-alpha window.
15. **RNPKUL:** upper limit on risetime in k-alpha window.
16. **RNPKLL:** lower limit on risetime in k-alpha window.
17. **BASEUL:** upper limit on pretrigger average voltage.
18. **BASELL:** lower limit on pretrigger average voltage.
19. **TRIGUL:** upper limit on PULse trigger.
20. **TRIGLL:** lower limit on PULse trigger.

21. **ZERLIM:** maximum excursion allowed for noise sweeps.
22. **DBASEL:** maximum allowed pre-trigger variation.
23. **SUMFIL:** integral of the optimum filter without the time factor.
24. **CALEN:** energy of the calibration peak in eV.
25. **CONVFACT:** calen/sumfil, used as a conversion factor to convert the voltage of sweeps to their actual energy.
26. **STNR:** best signal to noise ratio for a given sweep.
27. **FWHM:** full width half maximum value.
28. **SUMSIG:** un-normalized maximum of the average windowed signals and the optimum FILter used to calculate calen.
29. **ELC_A:** gain drift correction parameter.
30. **ELC_B:** gain drift correction parameter.
31. **ENLC_C:** nonlinear correction parameter.
32. **ENLC_D:** nonlinear correction parameter.
33. **FILTPROG:** progress of analysis for a given pixel.
34. **WINSTART:** lower time limit of pulse classification limiting range.
35. **WINEND:** upper time limit of pulse classification limiting range.
36. **Prtrg:** value from parameter File used to determine the pre-trigger region.
37. **winfrac:** parameter used in windowing.

HDU 3 has all but the sweep array column from the SWP File along with a number of additional parameters which characterize each sweep. The columns names and descriptions are as follows:

1. **Pixel:** the pixel number for a given row.

2. **Sweep_Num:** the running count for all pixels.
3. **Pixel_Num:** the running count for each pixel separately.
4. **PULse_Time:** the trigger time for each pulse in terms of the original stream of data.
5. **Flag:** is not currently used.
6. **yinit:** is the baseline voltage of each sweep.
7. **rawppk:** is the maximum positive excursion.
8. **rawnpk:** is the maximum negative excursion.
9. **dbase:** is the maximum variation in the pre-trigger region.
10. **dectim:** is the decay-time.
11. **ristim:** is the rise-time
12. **tfb:** is the software calculated trigger channel.
13. **lpamp:** is the amplitude of the low-pass filtered sweep at the array element corresponding to the average maximum position.
14. **iclass:** contains information on all classifications of each sweep.
15. **eraw:** is the energy calculated using the optimum filter and scaling to the energy of the calibration pulses.
16. **egdc:** is the gain drift correction applied to eraw.
17. **enl:** is the nonlinear correction applied to egdc.
18. **xmax:** is the real position of the maximum excursion.

***_FIL.FITS (FIL File)**

The FIL File is the last FITS File created by FITSFILTER that is unique to a given base name. Its main purpose it to hold the optimum filter template. Additional HDUs hold other relevant data arrays relevant to the optimum filter for good measure. The columns names and descriptions are as follows:

HDU 1
Header: Empty
Image: Empty

HDU 2
Header:

- NSIG: Number of windowed signals used to create this array

Binary Data Columns (one row for each pixel):

Columns	Name	Type	Units
1	Pixel	1J	None
2	signal	2048E	Voltage

HDU 3
Header:

- **NNOI:** Number of noise sweeps used to create this array

Binary Data Columns (one row for each pixel):

Columns	Name	Type	Units
1	Pixel	1J	None
2	Noise	2048E	voltage

HDU 4
Header: Empty

Binary Data Columns (one row for each pixel):

Columns	Name	Type	Units
1	Pixel	1J	1J
2	S**2/N**2	2048E	voltage

HDU 5
Header: Empty

Binary Data Columns (one row for each pixel):

Columns	Name	Type	Units
1	Pixel	1J	None
2	S/N**2,time doma	2048E	voltage

HDU 6
Header: Empty

Binary Data Columns (one row for each pixel):

Columns	Name	Type	Units
1	Pixel	1J	None
2	S**2/N**2, time	2048E	voltage

HDU 7
Header: Empty

Binary Data Columns (one row for each pixel):

Columns	Name	Type	Units
1	Pixel	1J	None
2	S/N**2, freq do	2048E	voltage

(Table A.3) *FIL file format.*

HDU 5 contains the template created in FILAVG.F which is used to filter each sweep in FILFIL.F to ultimately obtain the energy spectrum. This File is particularly useful for sets of data that have poor statistics because the user has the option to filter the sweeps using a template from another FIL File (presumably sharing the same characteristics as the one being analyzed).

PARAMETER.FITS

PARAMETER.FITS is the only FITS File that is common to all data sets. It contains parameters for many of the automatic features of FITSFILTER and can be altered directly, eliminating the need to modify hardcoded parameters and recompile.

This File is read when creating the PUL File for the first time and its header elements are copied into the primary HDU of the PUL File. Thus, if something is modified in the parameter File, the update option in FITSFILTER must be used to overwrite the values in the PUL Files header. The format is very simple, containing just one HDU with header values. The header keywords and descriptions are as follows:

- **FLOWPASS:** Low-pass filter applied in FILCHR.F
- **THRESH[1,2,3,4,5,6]:** Fraction of max. to use as a threshold in GNUSEL.F
- **CALEN:** Calibration energy for K-alpha
- **NSIGMA:** Multiple of sigma used for min threshold in DBLFIL.F
- **LPPOLE:** Pole of lpfilter applied to the optimum filter
- **LPFREQ:** Frequency of lpfilter applied to the optimum filter
- **A:** Gain drift correction:
- **B:** Gain drift correction:
- **C:** Nonlinear correction:
- **D:** Nonlinear correction:
- **WINFRAC:** Parameter used in window subroutine (utility.f)
- **PRTRG:** Fraction used to determine ilzero for pre-trigger
- **TRIGGER:** F: use hardware trigger; T: use software trigger

All together these are the four primary Files used for FITSFILTER. Before we continue with a deeper description of FITSFILTER and its subroutines, we will take a small detour and cover some concepts which will clarify the reader's understanding.

Appendix B

FITSFILTER, Technical Details

This appendix chapter provides an overview of the technical aspects and code for FITSFILTER that may be useful for users of the program.

FITSFITLER and its Subroutines

- **FITSFILTER.F**: This is the main program which controls access to the subroutines. It takes the SWP File as input and generates an associated PUL File if it does not already exist. The user if given the option of running in manual mode (pick a pixel and perform any menu option available), or in auto mode which analyzes all pixels in a given SWP File.
- **FILCHR.F**: This is the first subroutine that the user may run. It calculates parameters from each sweep in the pixel being analyzed and saves them in the associated PUL File.
- **GNUSEL.F**: This routine reads the parameters created in FILCHR.F and plots histograms to select limits on the parameters. This is also where the user selects the calibration sweeps (windowed sweeps) and their related energy.
- **FILWIN.F**: This program uses the limits selected in GNUSEL.F and classifies pulses accordingly.

- **DBLFIL.F**: This program analyzes pulses to identify pulses in sweeps classified as noise and double pulses and reclassifies them accordingly. At this point, no further automatic changes to classification will occur.
- **PLOTSWP.F**: This subroutines may be run at any time. It allows the user to plot the sweeps, and if they were classified in FILWIN.F, they may be plotted based on class. The user is given the option of custom class changes if desired
- **FILAVG.F**: This subroutine reads in noise and calibration sweeps to generate an optimum FILter (saved into the FIL file).
- **FILFIL.F**: This subroutine read the FIL file (user has the option of using one from a different file) and filters all sweeps and scales the results (based on the values obtained for the windowed sweeps of known energy) and saves them in the PUL file.
- **ESCALE.F**: This subroutine provides corrections to the energy spectrum. First it performs a gain drift correction by fitting the windowed signals in time to a line. The user may also apply a second order polynomial correction if a second calibration peak of known energy exists
- **FILPLT.F**: This subroutine allows the user to plot any combination of parameters saved into the PUL File as either a histogram or scatter plot. The user may save the plots or data. The histogram of the energy is ultimately the final result desired, but the other options allow the user to check for problems and to identify their cause.
- **Utilit.f**: This holds most of the subroutines and functions called from FITSFILTER and its subroutines such as low-pass filters and linear regressions.

- **Plot.f**: This holds the subroutines that allow for plotting and includes the routine that generates histograms.

Each subroutine exists independently, which facilitates swapping of subroutines as required by the user. For example, DBLFIL.F, the double pulse detection subroutine, may be swapped out with a File that does the same task, but uses a different possibly more effective algorithm. In general, this allows other users to easily modify the program for their own purposes.

Units

There are three main stages encountered during the acquisition process of sweeps which are all related by a constant factor.

$$Real\ Signal \rightarrow Amplified\ Signal \rightarrow Digital\ Signal$$

The real signal is the actual output of the detector. This is amplified by a SQUID (or some other integrated signal transducer) and then a low noise amplifier. That signal is then digitized and becomes the starting point of our analysis. The digital signal is recorded in scope units which are generally integers. In order to get the Amplified signal from this, you must multiply the Digital signal by the sampling voltage. Assuming the SQUID and amplifier gain are known, we may convert the amplified signal to the real signal. The three stages are thus related as follows:

$$Real\ Signal = (Amplified\ Signal)/(Total\ Gain)$$

$$Amplified\ Signal = (Digital\ Signal) \times (Sampling\ Voltage)$$

It is important to note that because these three forms are related only by a constant factor, in principal we could just pick one and use it exclusively because the final results for the energy spectrum are normalized based on a calibration spike of known energy. In reality, the digital signal is not used because it is an integer array and analysis is performed which requires a real array as input. Thus, either the Real or Amplified signal are used. The only time the real signal need be calculated is when we are looking at plots.

Windowing

Fourier transforms decompose functions into sums of sines and cosines. Because these functions are periodic, considerations need to be made when applying a Fourier transform to a non-periodic function such as the data being analyzed by FITSFILTER. It is ideal to modify the original function as little as possible without causing discontinuities (which are handled very poorly in this regime). The standard approach is to use a windowing function applied to the data being transformed. The windowing function only alters the data at its boundaries where it smoothly pinches to zero. We use a half potato shaped function which is just the first half of a sine wave of amplitude one with the maximum stretched out.

Fast Fourier Transform (FFT)

The Fast Fourier Transform, or FFT, is an efficient algorithm for performing the Fourier Transform on discreet, complex data sets. The version of FFT we use comes from

the book, 'Numerical Recipes in Fortran 77', where the call to the function is: four1(data,N,isign). 'Data' is a complex array assumed to be a size that is a power of 2, 'N' is the size of the complex array, and 'isign' is +1 for a forward FFT (time to frequency domain) and -1 for a reverse FFT (frequency to time domain). Note that you must manually multiply the output of a reverse FFT by a factor of 1/N because the routine does not account for normalization. Doing so ensures that applying both and FFT and a reverse FFT to a data array will return the original.

It is important to understand which elements of the output correspond to which frequency/time element. For the analog case, if we let h(t) be the signal in the time domain and H(f) be the Fourier Transform of h(t), we have:

$$h(t) = \int_{-\infty}^{\infty} H(f) e^{-2\pi i f t} df$$

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f t} dt$$

In the discrete case, with a sampling rate of Δ and N samples, we define the discrete Fourier transform as foloows:

$$h_k = \frac{1}{N} \sum_{n=1}^{N} H_n e^{-2\pi i k n / N}$$

$$H_n = \sum_{k=1}^{N} h_k e^{2\pi i k n / N}$$

With the following relationship between the discrete and continuous case:

$$h(t_k) = \frac{1}{\Delta} h_k$$

$$H(f_n) = \Delta H_n$$

The following chart shows how the time and frequency elements are related for the input and output of four1():

Element #	time	frequency
1	0	0
2	Δ	1/NΔ
…	…	…
N/2	(N/2-1)Δ	(N/2-1)/(NΔ)
N/2+1	(N/2)Δ	±1/(2Δ)
N/2+2	(N/2+1)Δ	-(N/2-1)/(NΔ)
…	…	…
N-1	(N-2)Δ	-2/(NΔ)
N	(N-1)Δ	-1/(NΔ)

(Table B.1) *FFT accounting.*

In general, time and frequency can be written as a function the array element number, 'n', as follows:

$$t(n) = (n-1)\Delta$$

$$f\left(n \le \frac{N}{2}\right) = \frac{n-1}{N\Delta}$$

$$f\left(n \ge \frac{N}{2}\right) = \frac{N-(n-1)}{N\Delta}$$

For the discrete case of the reverse FFT, note that the summation can be written a constant plus the sum over positive frequencies plus the sum over negative frequencies. In the case of the Nyquist frequency (1/2Δ) for both the sum over positive and negative frequencies, the exponential becomes:

$$e^{\pm \pi i t(k)/\Delta}$$

Because t(k)/Δ is an integer it follows that only the even cosine part survives. Thus the sign of the Nyquist frequency is ambiguous. In FITSFILTER for all forward FFTs, the input is strictly real (the complex portion of the input array is set to zero), so the result has the following symmetry:

$$H(-f_n) = [H(f)_n]^*$$

Because the input array has N real elements and the output array has N real and N complex elements, it is implied that some of the information in the output array is redundant. The above symmetry shows that the negative frequencies are related to the positive frequencies. Specifically, each frequency element is split equally between the positive and negative frequencies so when plotting, it suffices to plot the first N/2+1 elements where all of them except for the 0 and Nyquist frequency are doubled.

FITSFILTER.F

FITSFILTER.F is the first code called by the FITSFILTER program. It has been completely rewritten because the FITS Files data format differs so substantially from the original FILter program. When FITSFILTER is initialized it first performs a check to ensure that support Files required for the printing routine exists in the working directory. If everything is in order, the user is prompted to enter the base name for the sweep File and if a File of the form *_SWP.FITS (* denotes the base name) does not exist, the user may retry other base names, or exit. When a base name associated with an existing File is entered, the user has the option of running in one of two different possible modes: batch or interactive. Listed on the console are the existing pixels on the sweep File. By selecting the fictional pixel, '-1' (technically any negative integer) batch mode initialized and all existing pixels are automatically analyzed without further user input. If an existing pixel is selected, FITSFILTER runs in interactive mode strictly for the selected pixel. Should a PUL File associated with the base name not exist, one is created before

continuing. In general interactive mode should be the first option exercised on a new data set to check that automatically generated parameters are acceptable (and tweak them accordingly). Interactive mode is also beneficial for diagnosing problems, which will be covered in a later section. The subsequent descriptions of subroutines will assume interactive mode.

The analysis subroutines exist in a hierarchical order where most will not run out of order. This process is controlled by the 'filtprog' (filter progress) variable which is updated after successful completion at each step of analysis. At this point, in interactive mode, the console lists menu options to call subroutines based on the value of filtprog. The only subroutines which may be called at any point is the first plotting routine which allows any sweep contained on the File to be plotted. Note that filtprog is not regressively updated, so any changes resulting from rerunning a subroutine will not automatically propagate to the most recently run subroutine with rerunning the intermediate analysis as well. The only acceptation to this occurs in the double pulse detection subroutine when no windowed signals survive since this will cause misbehavior of subsequent analysis.

Note that option: " (u) update parameters;" is not associated with a subroutines and runs within FITSFILTER.F. It checks that the values saved in the primary HDU of the PUL Files correspond to the values in the parameter File. If this is not the case the values in the HDU are updated with the values in the parameter File. This allows fine tuning of parameters without the requiring the user to delete the FIL File and rerun the analysis.

FILCHR.F

FILCHR.F is the first of the analysis subroutines required for FITSFILTER. Characterizing parameters for each sweep are calculated and added to the PUL File which will be refined and ultimately contribute to obtaining an energy spectrum. Before analysis proceeds, the console lists some options which allow the user to manually fine tune the parameter calculation process. The options given are:

- (0) Another pulse
- (1) Set new position of maximum
- (2) Change low-pass frequency
- (3) Enter pulse number to jump to
- (4) Proceed with analysis

In addition, the first sweep is plotted with a line indicating which array element contains the average maximum of all the sweeps. Option (0) and (3) are used to explore the sweeps to make sure that the maximum array element is reasonable. This may not be the case if there were a significant number of sweeps triggered by noise or if there is microphonic contamination, in which case a new position may be selected by option (1). Depending on the amount of noise associated with the data set being analyzed, option (2) can be called to change the low-pass frequency to a larger value if there is significant noise or a lower value is there is negligible noise. Note that changing the lowpass frequency significantly will likely require a new maximum position to be selected since

the newly filtered pulses may have a comparably different shape. Once everything is satisfactory, all that is left is to proceed with analysis.

GNUSEL.F

Originally named idlsel.f in the original filter program, GNUSEL performs the same tasks but uses gnuplot rather than IDL to provide plots (as do all subroutines in FITSFILTER). The main purpose of this subroutine is to plot histograms of the values calculated in FILCHR.F to allow limits to be set on will be considered acceptable values. The first time this routine runs, the limits are selected automatically (based on values in the PARAMETER.FITS File), but each histogram is still plotted allowing further modification of the limits. The first four plots have very similar user options:

- (1) Change binning ('# of bins used');
- (2) Mark limits;
- (3) save data;
- (4) save plot;
- (9) continue [default];

Option (1) allows the resolution to be increased or decreased depending on the amount of statistics available. If the current limits displayed are not acceptable, option (2) may be selected. Options (3) and (4) save the data (into a *.dat File) or save the plot displayed (in postscript format) respectively. The last option will proceed to the next plot.

The fifth and final plot displayed in this subroutine is a scatter plot rather than a histogram. Each data pair consists of the low-pass filtered raw estimate of the energy (lpamp) and the rise-time plotted on the x and y axis respectively. This pair of values strongly characterizes each sweep allowing reasonable selection of the windowed signals which will be used to calibrate the energy of all sweeps later on. There is more information required here than with the histogram plots, so there are more options available:

- (1) select k-alpha window;
- (2) select k-alpha energy; ('current energy value')
- (3) save data;
- (4) save plot;
- (5) count pulses in box;
- (6) select pulse classification limiting range;
- (9) continue [default];

Note that k-alpha just denotes the sweeps which will be used for calibration. The window is automatically calculated just as the limits are with the previous histograms, and can similarly be refined via option (1) where the window refers to selecting limiting values on both the x and y axis forming a window/box. In order to properly calibrate the sweeps, the energy of the windowed signals is required (the default/current-selection is shown to the right of this option in units of keV). Options (3) and (4) perform the same operations as with the histograms. Option (5) counts the total number of calibration

sweeps selected. Option (6) is a new addition and was added based on the fact that it may not be beneficial to use all of the windowed signals for calibration if there is a region with excessive noise (or others problems). For example, in the case of a sounding rocket, you may only want to use calibration Pulses detected after the engine has shut off. A plot showing the low-pass filtered raw energy for each sweep as a function of time allows the user to select a clean region from which to use the calibration sweeps (along with the option to remove any selections). After continuing, the option to either redo all of the selection or to continue is listed. Upon continuing, analysis with GNUSEL is complete.

FILWIN.F

Based on the limiting values chosen in GNUSEL.F, finwin.f categorizes sweeps accordingly with the following binary classification:

- **Bit 0: Good sweeps:** The default value is zero. All sweeps that are not flagged will maintain this value.
- **Bit 1: Windowed signals:** Flagged for all pulses that are within the k-alpha window selected in the scatter plot of GNUSEL.F.
- **Bit 2: Noise:** Flagged for all sweeps whose maximum and minimum excursions are below the signal threshold set in GNUSEL.F.
- **Bit 3: Baseline out of range:** Flagged for all sweeps where the average voltage of the pre-trigger region is outside of the allowed values set in GNUSEL.F.

- **Bit 4: Accidental trigger:** Flagged for all sweeps whose software calculated trigger channel is outside of the range selected in gmnusel.f.
- **Bit 5: Multiple pulses:** This is not determined until the next subroutine DBLFIL.F which identifies double pulses.
- **Bit 6: Excessive noise in pre-trigger region:** Flagged for all sweeps where the difference of the maximum and minimum excursion in the pre-trigger region (the pre-trigger variation) is outside of the range selected in GNUSEL.F
- **Bit 7: Bad pulse shape:** This is not determined until the next subroutine DBLFIL.F.

The variable that stores all of this information for each pulse is the iclass parameter. For example, a windowed signal that is a double pulse would have the value: $2^1 + 2^5 = 34$. This allows a single integer to uniquely and completely describe a given sweep. Ultimately, in regards to obtaining final results, we are only concerned with sweeps with an iclass value no larger than four. These sweeps are generally referred to as "good sweeps" whereas sweeps with a higher value are similarly referred to as "bad sweeps". The bas sweeps are useful for identifying potential issues with the detector but otherwise are generally not considered meaningful.

If in GNUSEL.F, during the scatter plot a pulse classification limiting range was selected sweeps outside of this range will be treated slightly different (the description above holds for all sweeps within the limiting range). All good windowed signals outside of the range are reclassified to zero and all good noise is reclassified to one. The net

effect of this reclassification is that these sweeps are still good (iclass is at most four), but are not used to create the optimum filter.

DBLFIL.F

The main purpose of this subroutine is to identify double pulses. The following description applies to the "standard algorithm" which is a somewhat modified version of the original filter version. It is assumed that the detector used has a relatively linear response so that all pulses are very similar to the windowed signals. The shape of pulses we are analyzing have characteristically sharp rise times so that when a pulse is detected, there is a large voltage change over a small time, thus we use the first derivative of sweeps to identify double pulses.

For consistency, all sweeps are prepared similarly. The sweeps first have their baseline removed and are inverted (so that the maximum excursion of a pulse is positive). Then, they are low-pass filtered and their derivative is calculated. The nth element of a simple discrete derivative is given as:

$$\frac{V_{n+1} - V_n}{t_{n+1} - t_n}$$

However, because the denominator is constant for all n and because the overall scale is not important we simply just use the numerator. For simplicity, in this section, it is assumed that all sweeps analyzed with this algorithm are processed as described above even if not explicitly stated.

The basic procedure in its simplest representation proceeds as follows: (1) Obtain a threshold derived from the prepared noise based on the average RMS value. (2)

Average the windowed signals to create a template. (3) Take the difference of this template with the signal being analyzed (normalized to the same maximum voltage). This difference should be relatively flat and essentially just represent the noise on the pulse so that anything above the threshold from step (1) identifies a double pulse.

However, there may be low energy pulses in the noise or double pulses in the windowed signals which may through off the algorithm. In order to avoid this possibility step (1) and (2) are performed iteratively. More specifically we read nsigma (originating from the parameter File) which denotes the lowest multiple of the RMS noise used as a threshold: cutoff. We then use 16 times this value as a threshold and search for any good noise sweeps with an excursion above this threshold (since sweeps maximum are positive). If any are found they are reclassified as a pulse (iclass is set to 0) and the RMS voltage is recalculated and the pulses are reanalyzed with the same threshold. If none are detected the threshold is halved and the process is repeated until the threshold is just cutoff (which is different than the original value if any pulses were found in the noise sweeps). The final result is a cleaner value for the cutoff. For step (2) we precede similarly but instead of refining cutoff, we are refining the template by removing double pulses in the windowed signals by performing step (3) iteratively on just the windowed signals. Finally step (3) is performed on all pulses (except for noise and windowed signals since those have already been analyzed) with the cleaner template.

PLOTSWP.F

This subroutine allows the user to view any of the sweeps on the pixel being analyzed. It has two modes: full and limited. If the pulses have been classified (finwin.f) then full mode will run otherwise option will be limited. In full mode the available options are:

- (1) Plot by class;
- (2) Select plot range;
- (3) Remove plot range;
- (9) Exit;

Option (1) allows the user to explore sweeps based on their classification. This can be done inclusively or exclusively. Inclusive will show all plots that have the selected classification whereas exclusive shows sweeps that only have the selected classification. All of these features are useful to visually confirm whether or not sweeps are being properly classified (particularly good windowed signals). Option (2) is very robust and allows the user to select the parameters plotted on the x and y-axes of a scatter plot and select a windowed region to limit what is shown in option (1). This is particularly useful for pixels with very large amounts of sweeps. Option (3) simply removes the range selected in option (2) and option (9) returns to the main FITSFILTER menu. In limited mode, the user only allowed to plot the sweeps without any additional options.

FILAVG.F

This subroutine is the union of two subroutines distinct in the original filter program. The main purpose of this routine is to create the optimum filter, stored in the FIL File, which will be used to extract the energy of each sweep. If a FIL File does not exist at this point, i.e. this is the first time running FILAVG for a File, it will be created.

To create the optimum filter, we need several ingredients. First, the windowed signals are averaged in the time domain then converted to the frequency domain. Then the noise is converted to the frequency domain, squared (absolute value squared so the result has no imaginary component), then averaged. The average signal is divided by the squared noise to obtain the template in the frequency domain. Along the way, several easily constructed arrays relating to the optimum filter and the average noise and signals are later saved into the FIL File for future reference.

Before saving the frequency domain optimum filter, the option is given to apply a low-pass filter with a user input frequency and pole. The optimum filter is displayed as a visual aid in determining whether or not to apply a low-pass filter. If one is applied, it may be saved to File. Finally, the user may display and or save the entire array created. All plots displayed may be recovered from the FIL File if desired.

FILFIL.F

The main purpose of this subroutine is to use the optimum filter created in FILAVG.F to extract the energy of sweeps. Only the maximum element of the convolution of the sweep being analyzed and the optimum filter are required so it is not necessary to calculate the entire function. In principal we already know the phase for this element, so we simply calculate the convolution explicitly. In reality we may be off slightly so we the supposed maximum of the convolution and its neighboring elements. If either of it neighbors is larger, this directly indicates that we are not at a local maximum and the calculation is performed again with the phase shifted accordingly. Once the maximum value is obtained, we have an un-scaled energy for the sweep. This value can be slightly improved by assuming it and its two neighbors lie on a parabola and taking the maximum of the parabola.

To properly scale the energy we use the un-scaled energy from the average windowed signal. Because the windowed signals have a known energy (input from GNUSEL.F), a scale factor can be created and all other un-windowed pulses, including the noise can have their energy extracted. However, further correction may be applied. These are handled in the next subroutine.

ESCALE.F

This subroutine applied corrections to the energy calculated in FILFIL.F. The first energy correction accounts for gain drift. As the data stream for a given File is read from a detector, often times over the total acquisition time, the gain (baseline) slowly drifts. This can usually be approximated as linear in nature (other possibilities are not accounted for here). In order to measure the gain drift, we examine the energy of the windowed signals as function of their time (the time they were trigger in the data stream). Via linear interpolation, we fit the data with a line whose parameters are displayed. The user may decide to apply the correction or to skip it. Either way, the gain drift energy correction column is updated (egdc) is updated since it will be used for the next correction.

The next correction that can be applied is the non-linear correction. If there is a second calibration peak of known energy, a parabolic correction may be applied. From a histogram of the gain drift correction energies, the second calibration peak may be selected and its energy inputted. Before applying the correction, a consistency check is performed to make sure the entries are physically possible (for example the second calibration peak cannot be to the right of the original with lower energy). The user may decide to apply this correction or to simply proceed without it. In both cases, the nonlinear correction column is updated with the new values or just copied from the gain drift correction. Note that in automatic mode, only the gain drift correction is applied.

FILPLT.F

This is the final analysis subroutine called by FITSFILTER.F. It allows the user to plot either a histogram or scatter plot with any of the characteristic sweep parameters. Further, these plots can be limited based on the sweeps class. In most cases the final result desired is a histogram of the nonlinear corrected energy (contains the most precise energy even if no correction were applied in which case it is the same as the raw energy). The can be plotted with any user inputted range and resolution (binning).

PLOT.F

This subroutine handles all of the plotting involved with FITSFILTER, for which only three functions are used: plotdata, plot2, and killplot. Plotdata handles plots of single functions with various options for the x and y-axis and plotting style. Plot2 only differs from plotdata in that it allows to different functions sharing an axis to be displayed on the same plot. Killplot simply closes the last open plot, but due to the method used, cannot close previous plots. Thus any call to one of the plotting functions should always be followed by killplot otherwise plots will remain even after ending FITSFILTER.

Appendix C

Contour Plots from Analysis of Chapter 6

Due to the large number of contour plots resulting from the analysis on the simulated double pulses data, the contour plots will be located here to improve readability. The parameter space explored includes two different energy resolutions (2and 5 eV energy resolution with blue and red contour plots respectively); three different double pulse time separations (1, 2, and 5μs); clean vs. dirty depending on whether double pulses are allowed to contaminate the template files. The above parameter space is considered for the standard and optimum filter algorithm. In addition the efficiency resulting from dirty sets when combining both algorithms is explored (in both possible orders). The format used for most comparisons will be to have the optimum filter plot (top) compared to the standard algorithm (bottom). The list below quickly summarizes the content of each figure

Clean with 5 eV Energy Resolution (compare efficiency of algorithms)

- Fig. 1: 1μ double pulse separation
- Fig. 2: 2μ double pulse separation
- Fig. 3: 5μ double pulse separation

Dirty with 5 eV Energy Resolution (compare efficiency of algorithms)

- Fig. 4: 1μ double pulse separation
- Fig. 5: 2μ double pulse separation
- Fig. 6: 5μ double pulse separation

Clean with 2 eV Energy Resolution (compare efficiency of algorithms)

- Fig. 7: 1μ double pulse separation
- Fig. 8: 2μ double pulse separation
- Fig. 9: 5μ double pulse separation

Dirty with 2 eV Energy Resolution (compare efficiency of algorithms)

- Fig. 10: 1μ double pulse separation
- Fig. 11: 2μ double pulse separation
- Fig. 12: 5μ double pulse separation

Comparison of efficiency using both algorithms together at 5 eV Energy resolution

- Fig. 13: 1μ double pulse separation
- Fig. 14: 2μ double pulse separation
- Fig. 15: 5μ double pulse separation

Comparison of efficiency using both algorithms together at 2 eV Energy resolution

- Fig. 16: 1μ double pulse separation
- Fig. 17: 2μ double pulse separation
- Fig. 18: 5μ double pulse separation

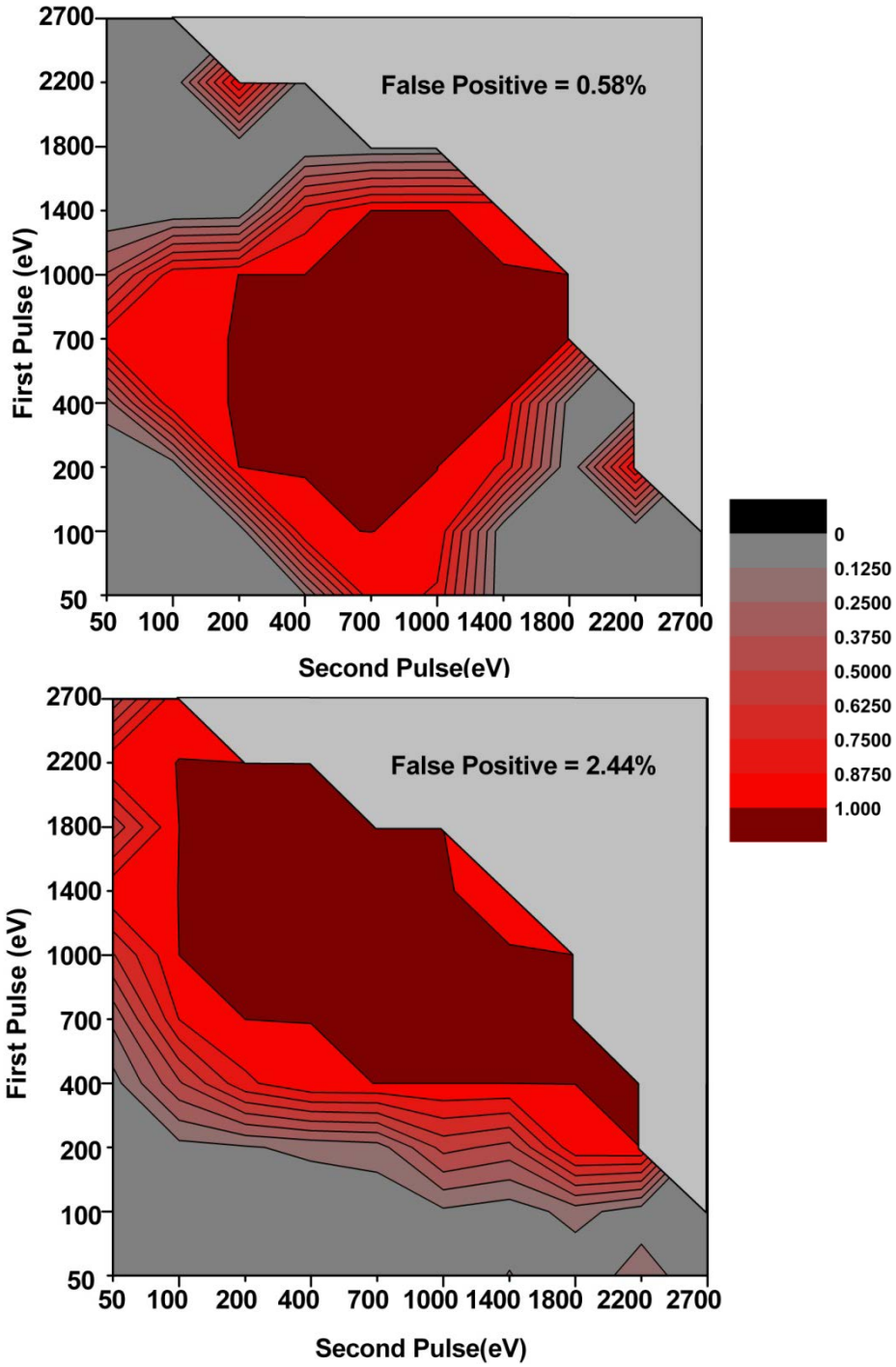


Fig. C.1; Clean sets at 1μs pulse separation with 5 eV energy resolution.

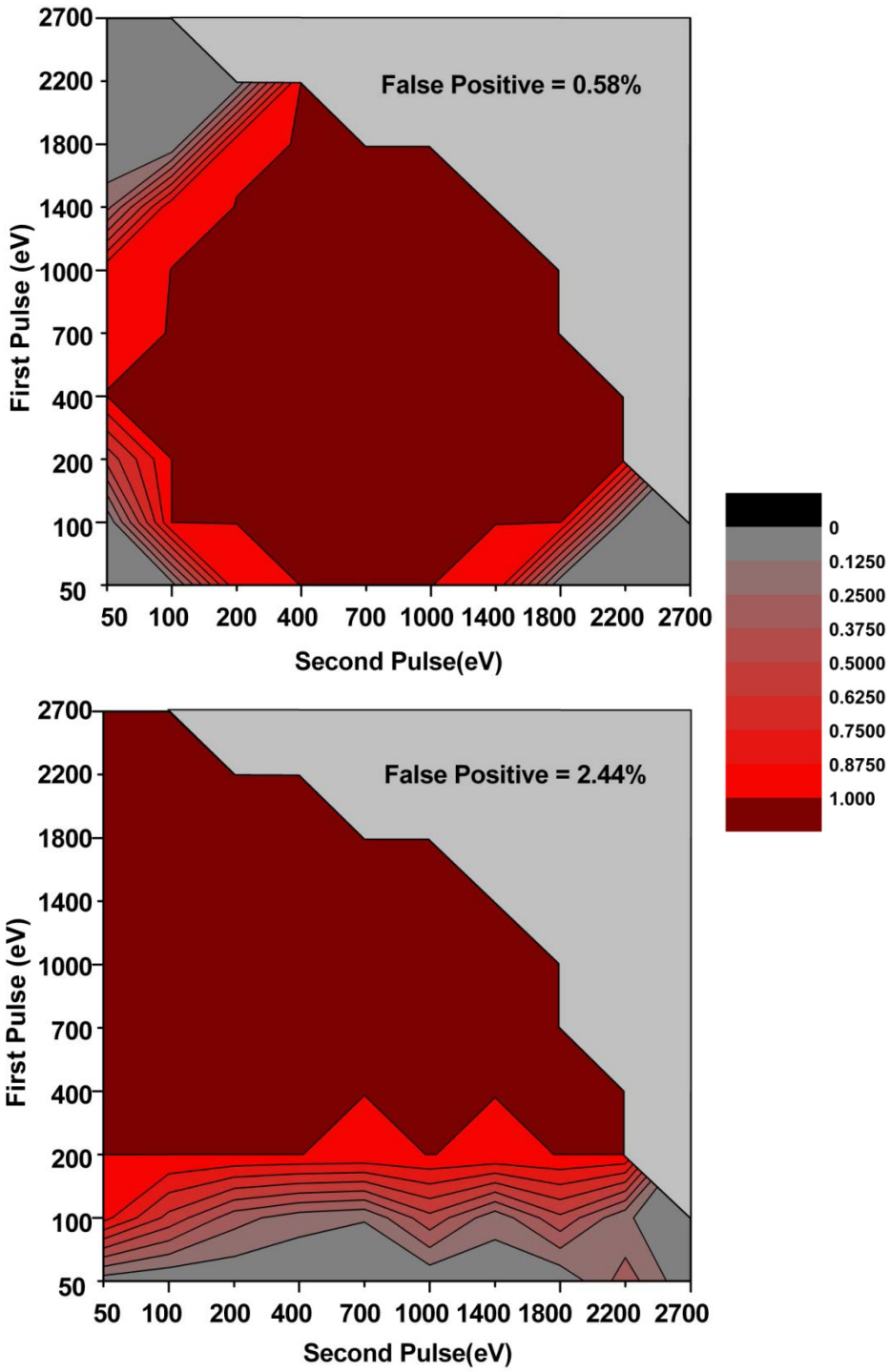


Fig. C.2; Clean sets at 2μs pulse separation with 5 eV energy resolution.

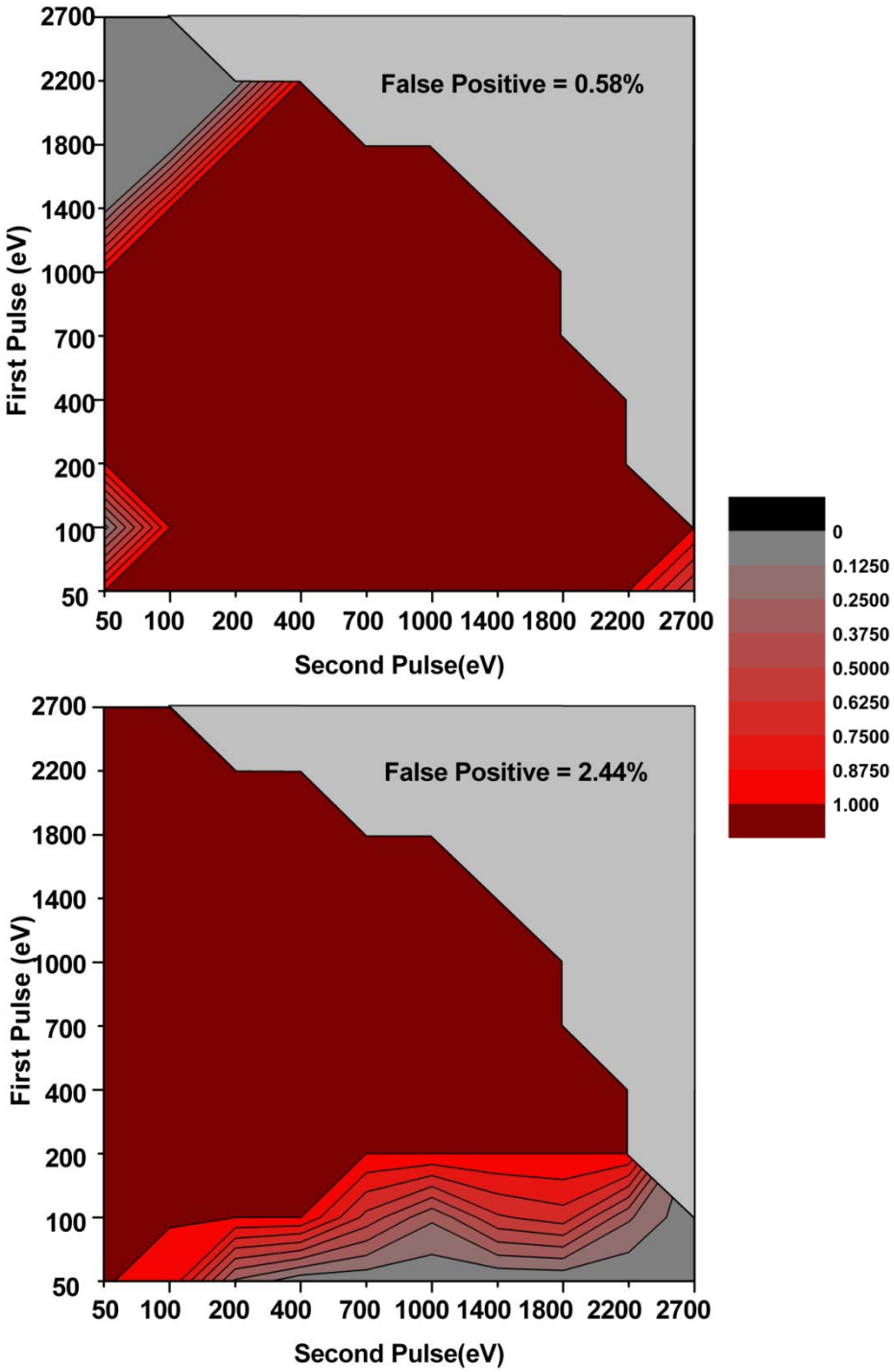


Fig. C.3; Clean sets at 5μs pulse separation with 5 eV energy resolution.

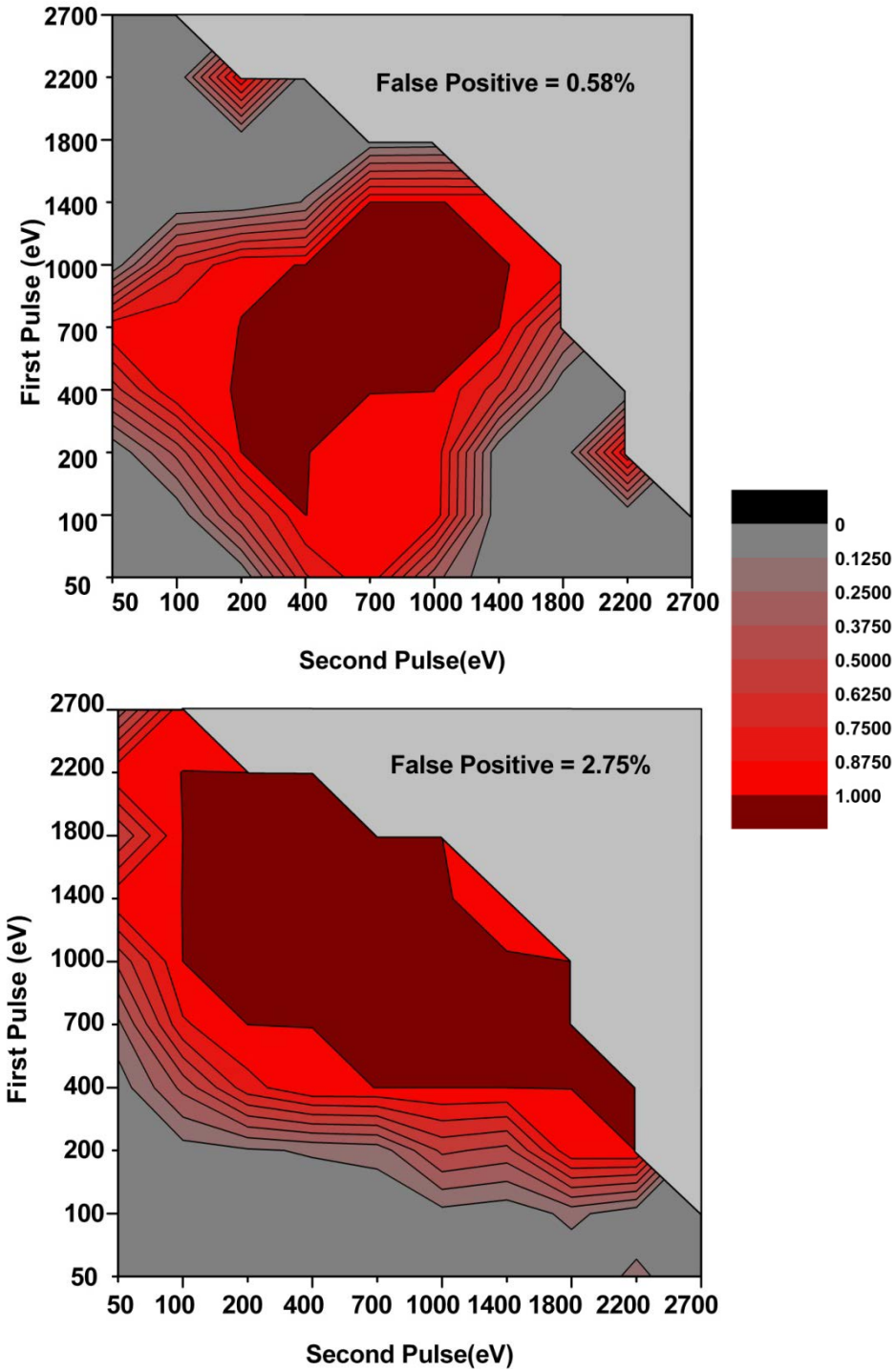


Fig. C.4; Dirty sets at 1μs pulse separation with 5 eV energy resolution.

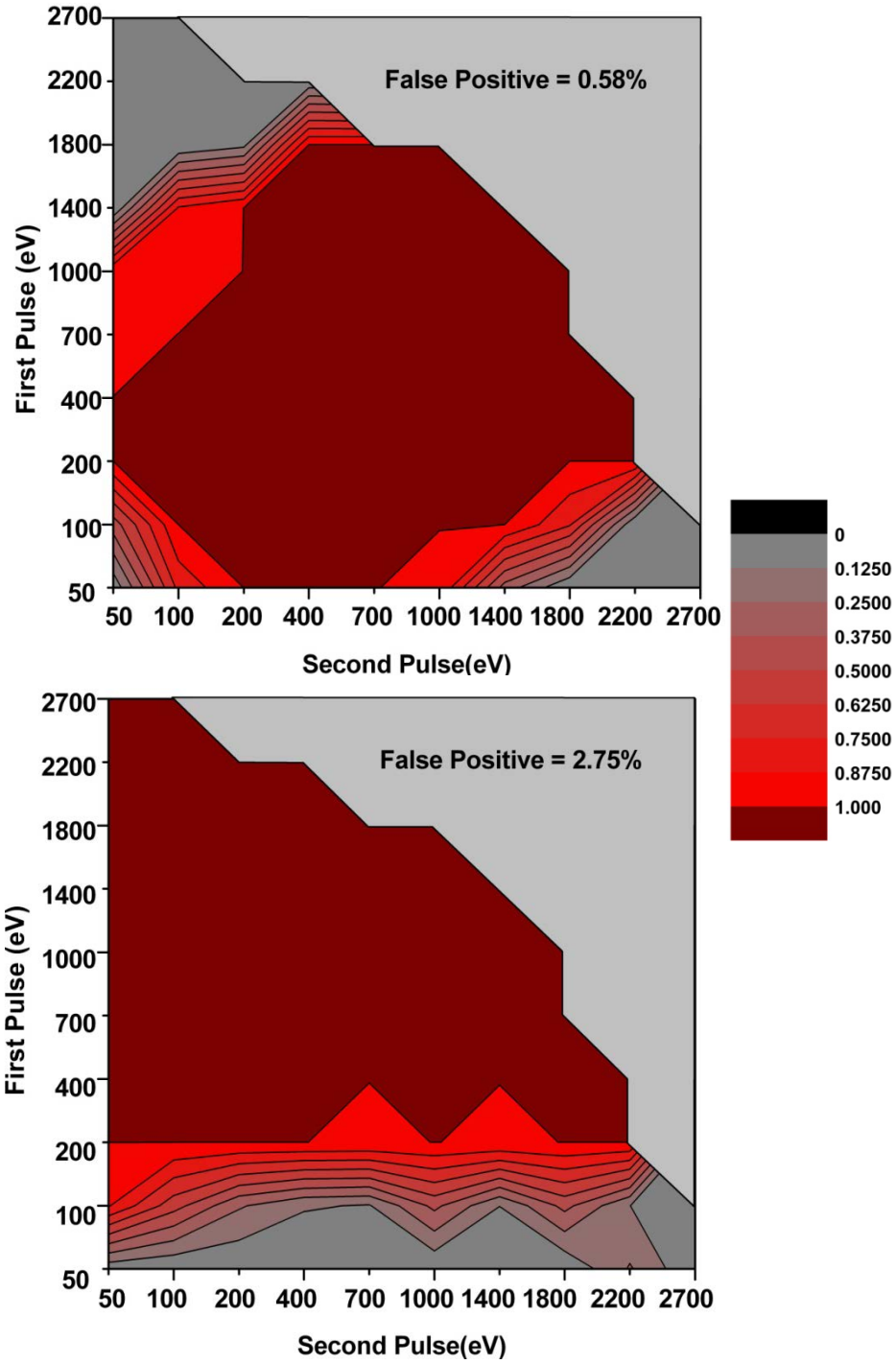


Fig. C.5; Dirty sets at 2μs pulse separation with 5 eV energy resolution.

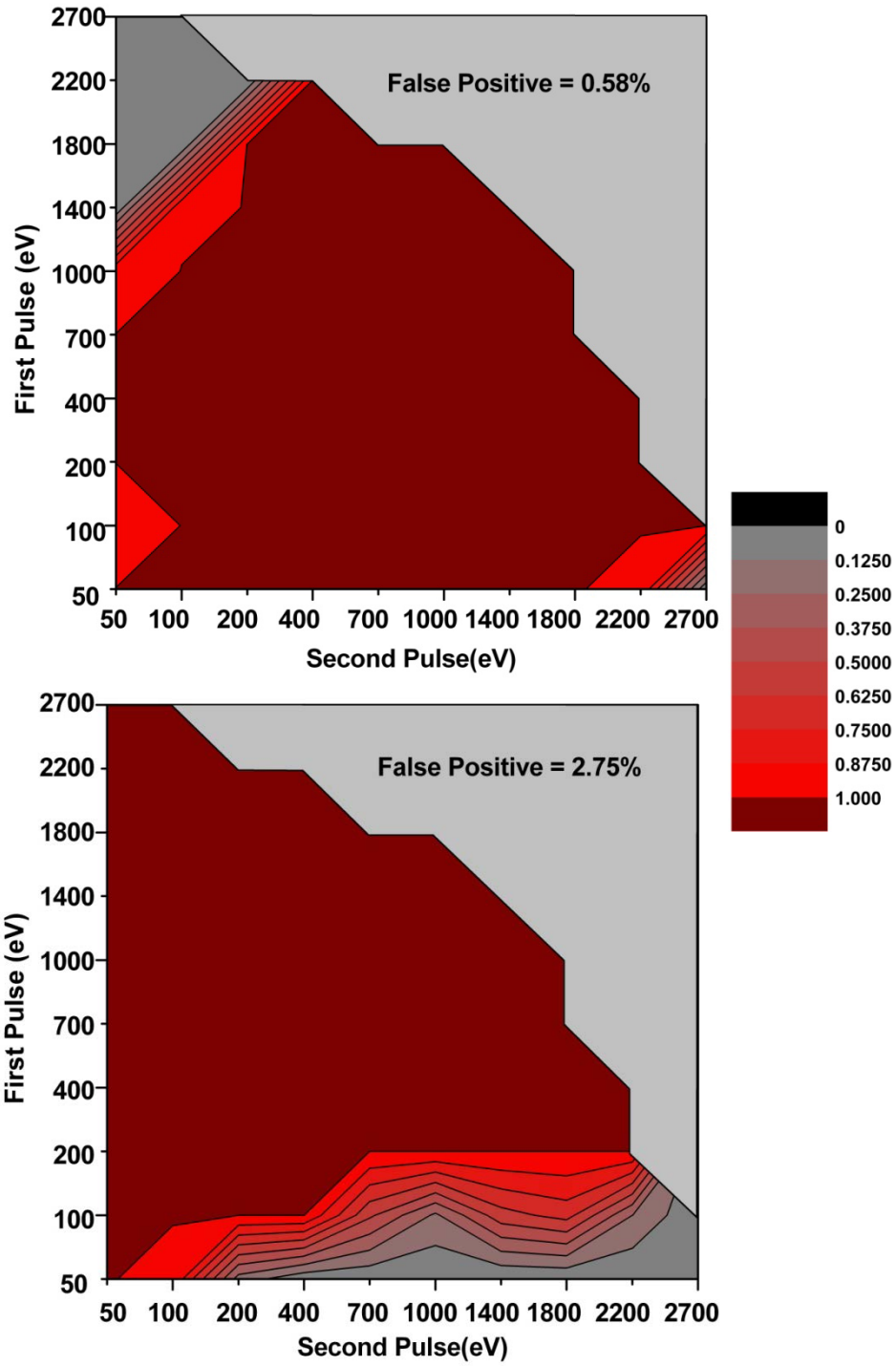


Fig. C.6; Dirty sets at 5μs pulse separation with 5 eV energy resolution.

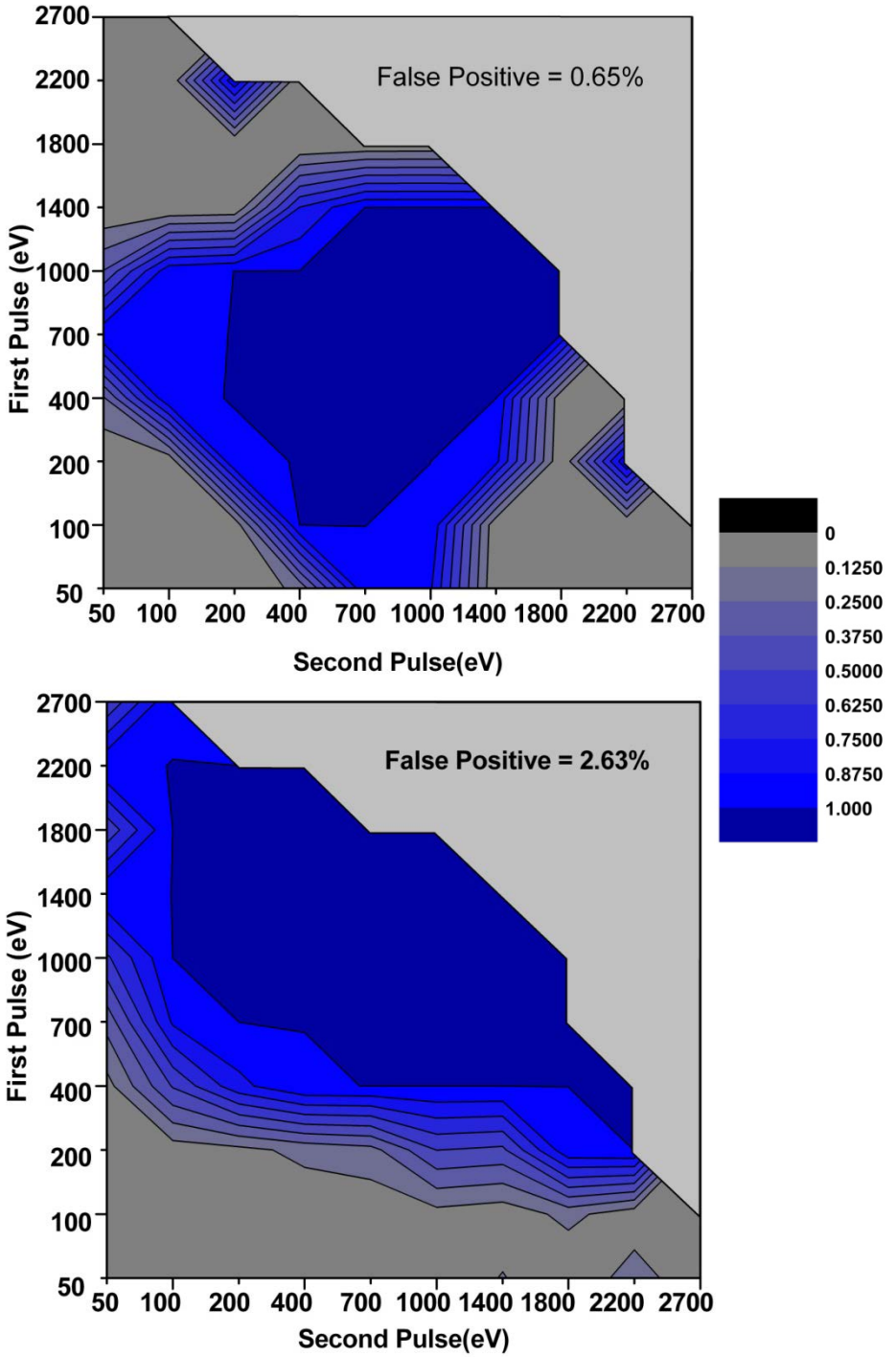


Fig. C.7; Clean sets at 1μs pulse separation with 2 eV energy resolution.

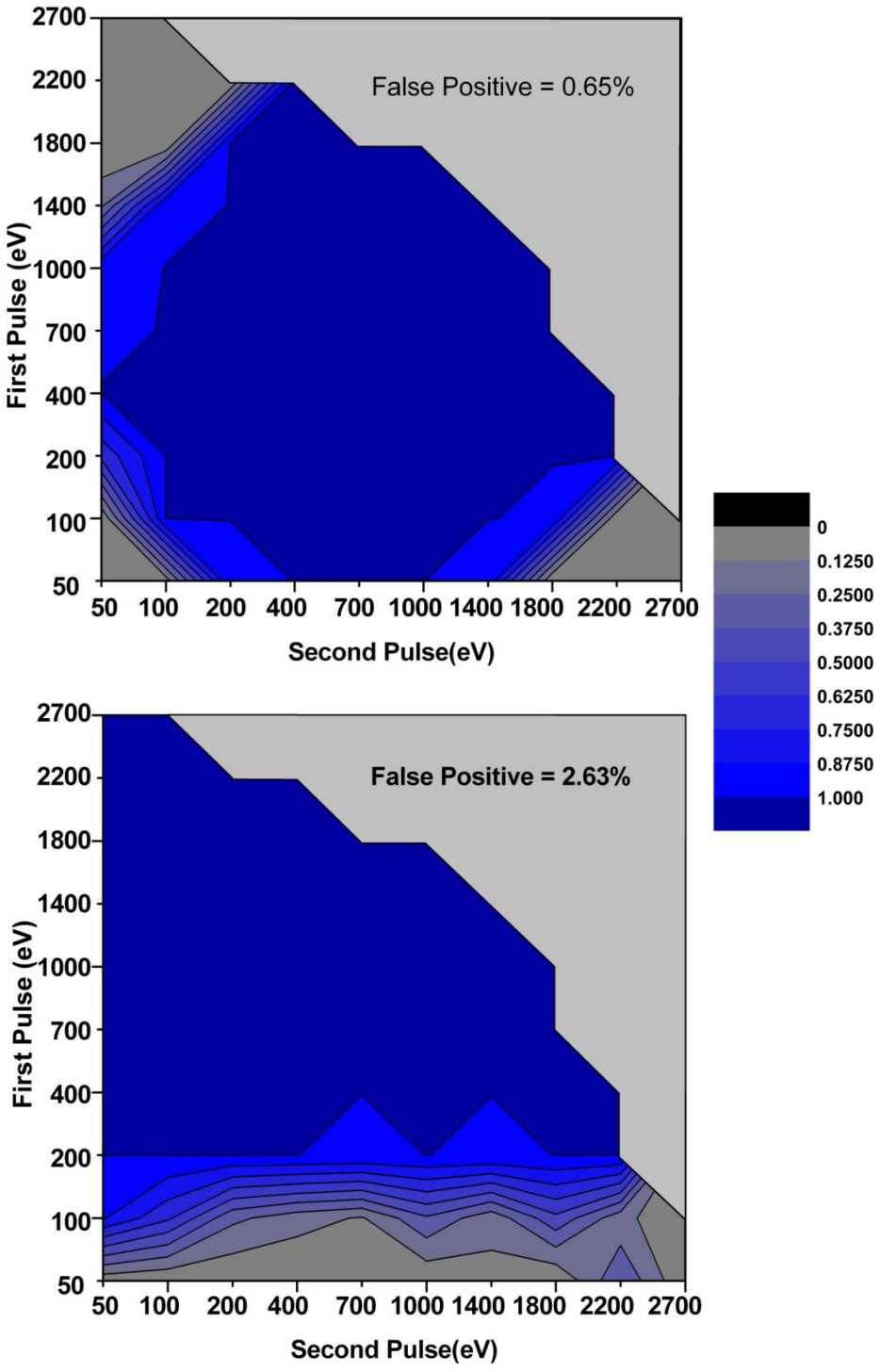


Fig. C.8; Clean sets at 2μs pulse separation with 2 eV energy resolution.

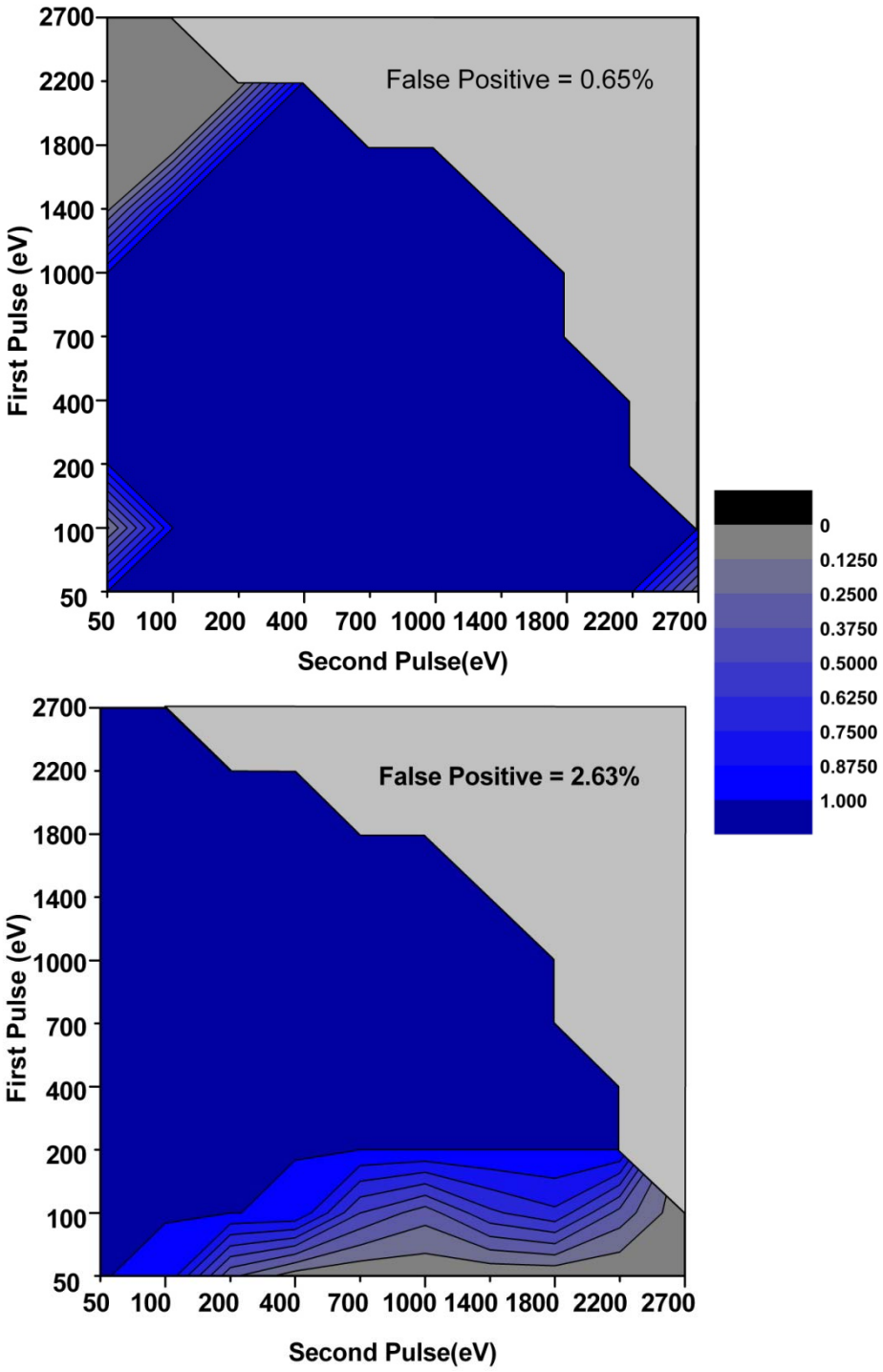


Fig. C.9; Clean sets at 5μs pulse separation with 2 eV energy resolution.

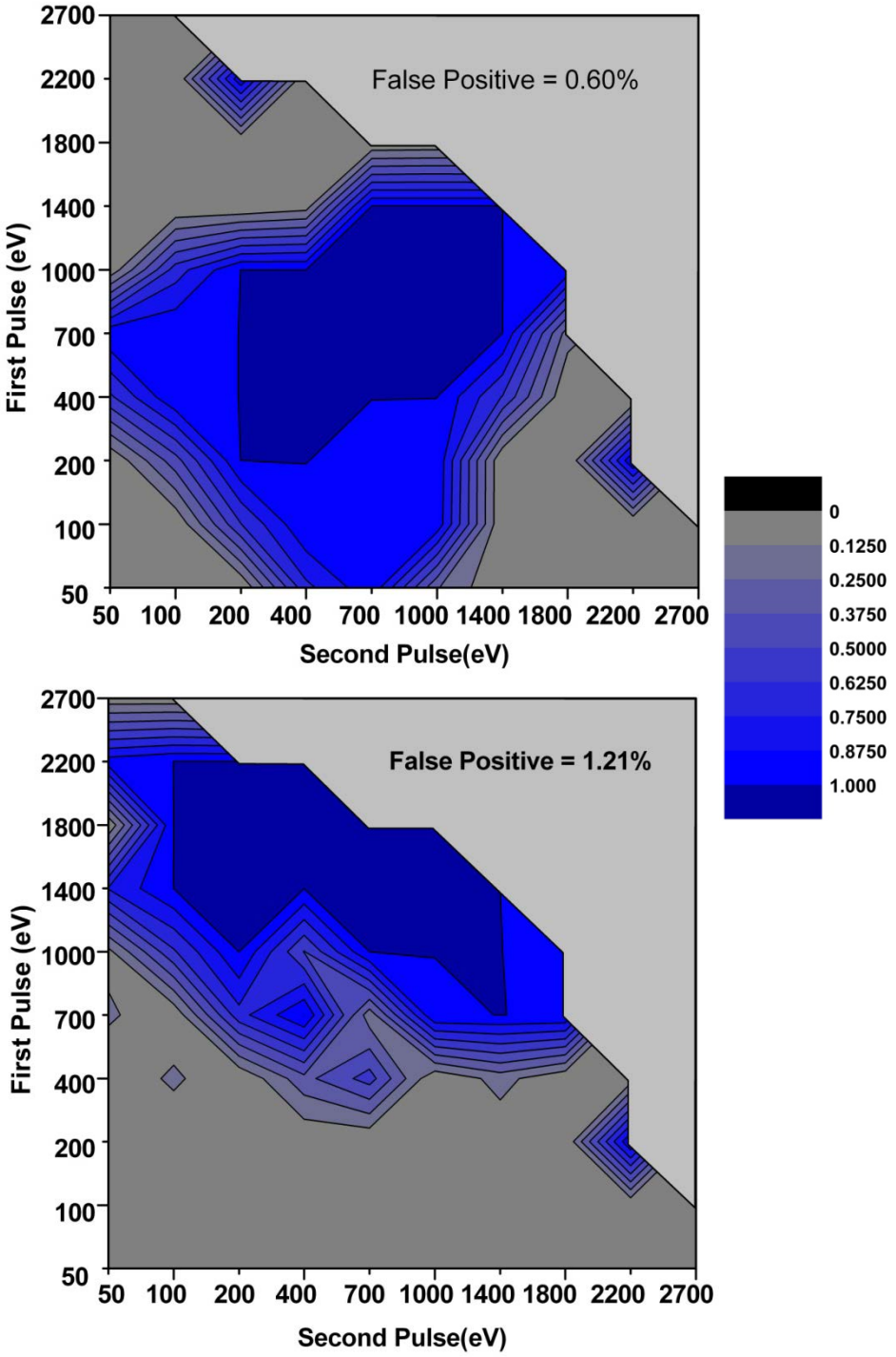


Fig. C.10; Dirty sets at 1μs pulse separation with 2 eV energy resolution.

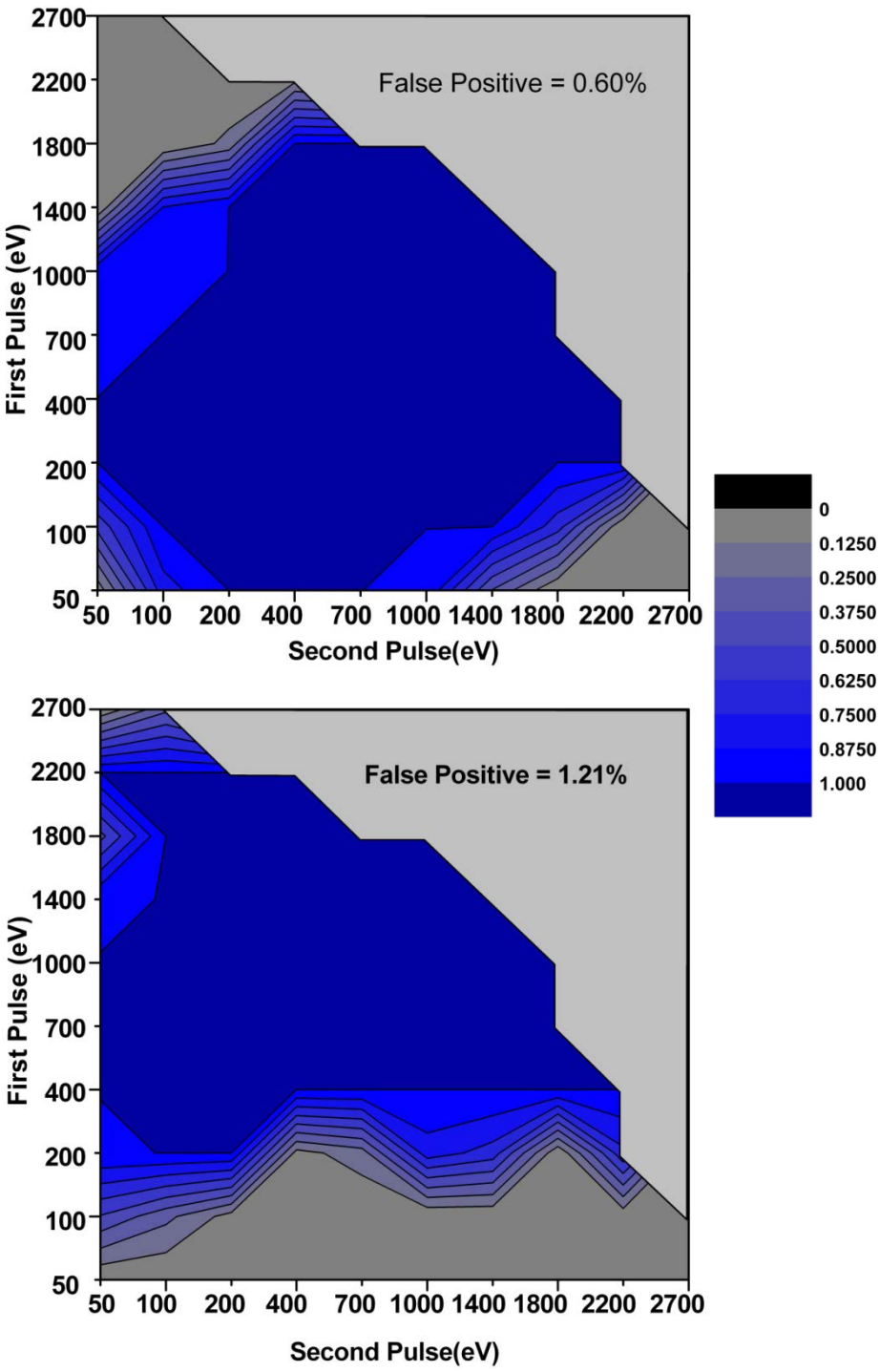


Fig. C.11; Dirty sets at 2μs pulse separation with 2 eV energy resolution.

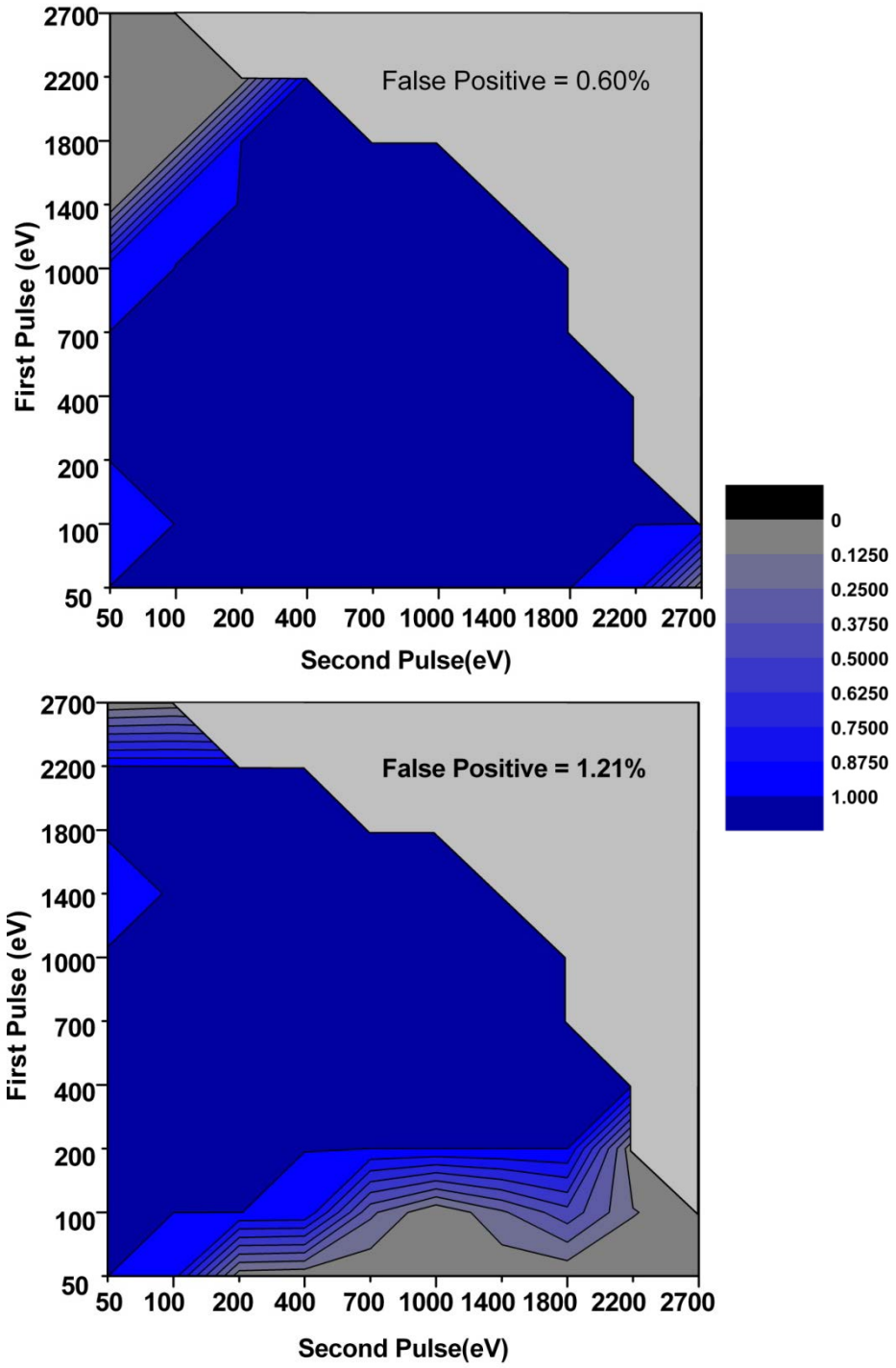


Fig. C.12; Dirty sets at 5µs pulse separation with 2 eV energy resolution.

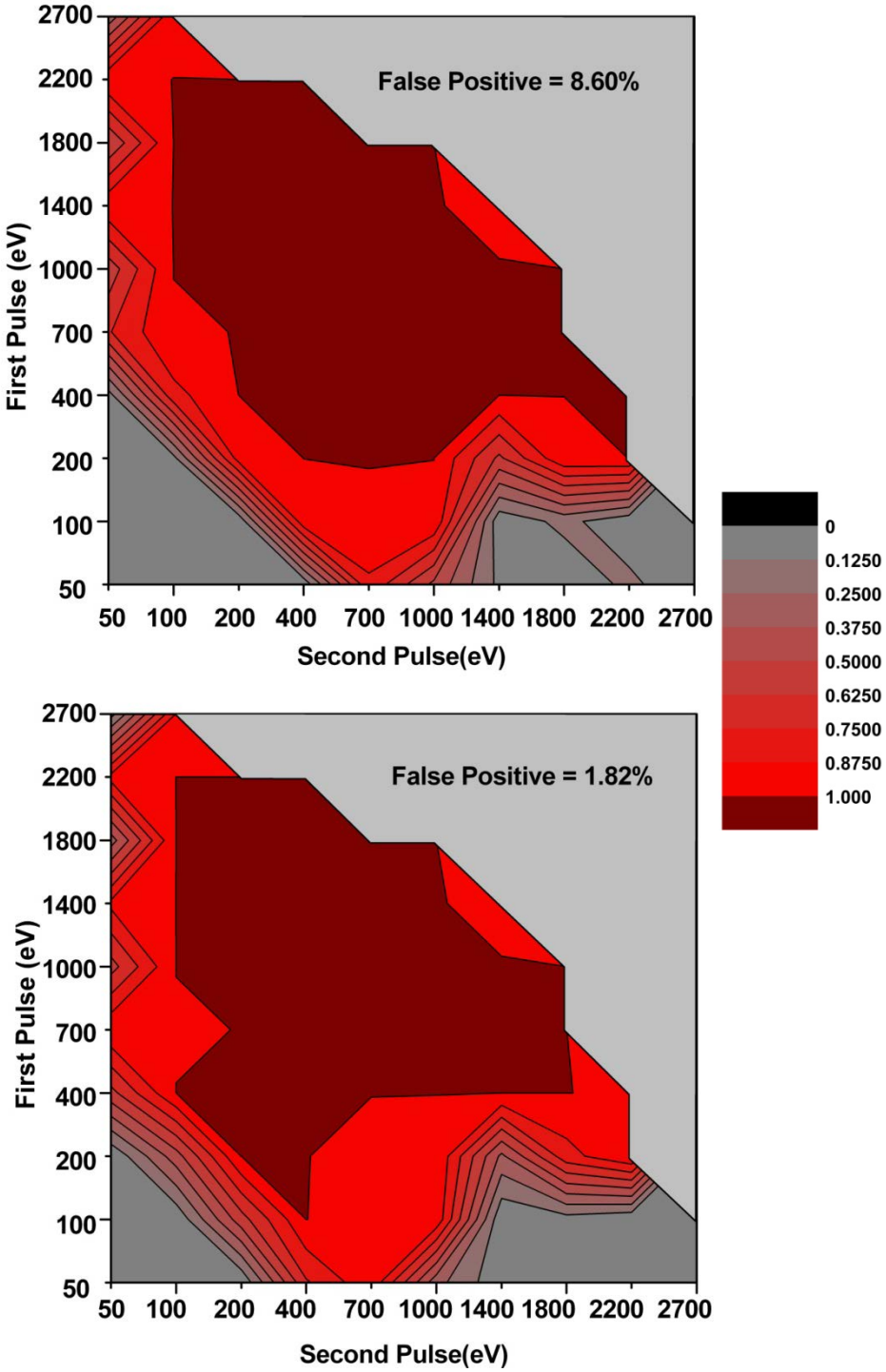


Fig. C.13; Comparison of results running both algorithms on the same data set with 1μs double pulse separation and 5 eV energy resolution: standard then optimum filter algorithm (top); optimum filter then standard algorithm (bottom).

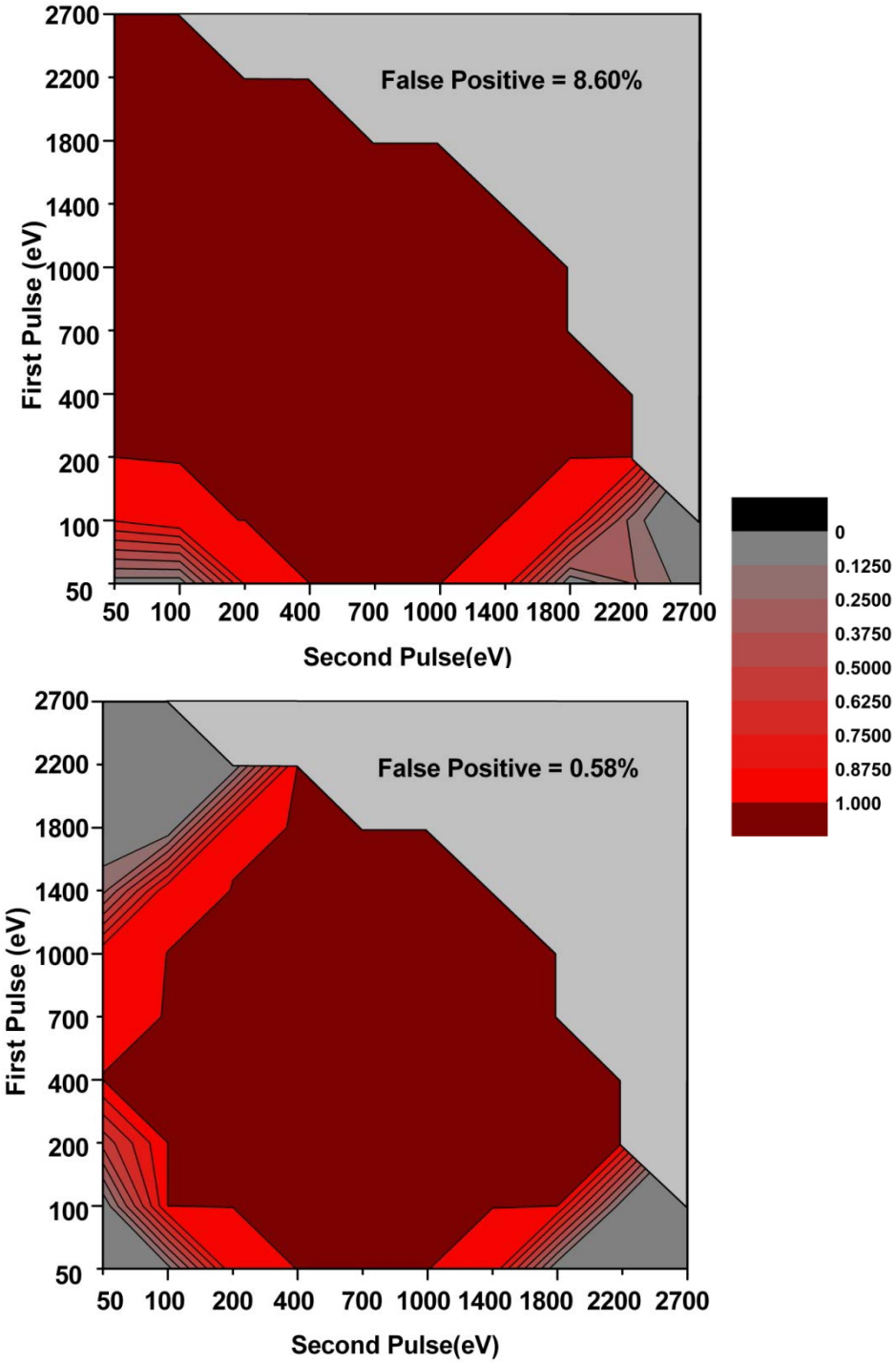


Fig. C.14; Comparison of results running both algorithms on the same data set with 2µs double pulse separation and 5 eV energy resolution: standard then optimum filter algorithm (top); optimum filter then standard algorithm (bottom).

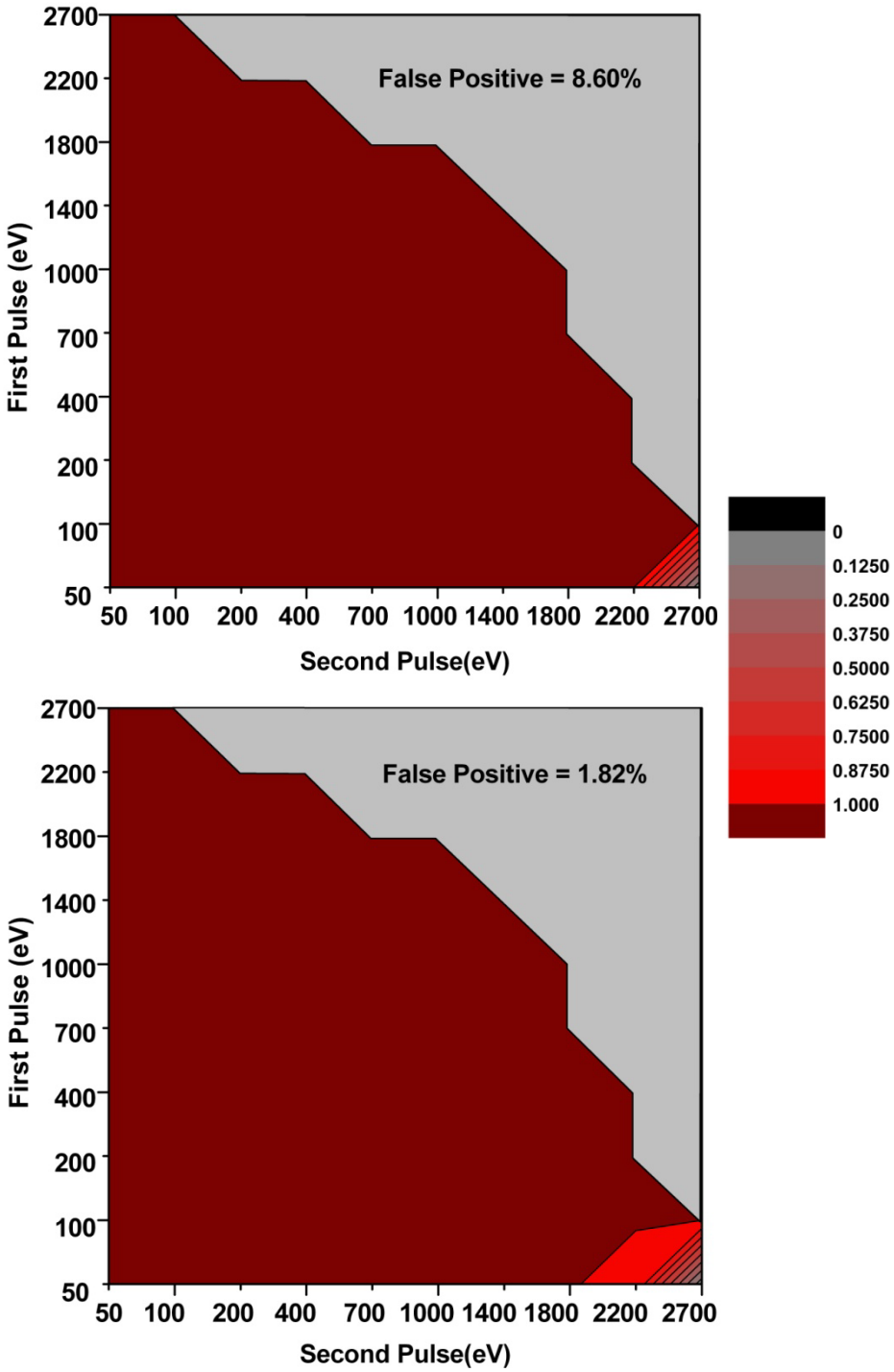


Fig. C.15; Comparison of results running both algorithms on the same data set with 5µs double pulse separation and 5 eV energy resolution: standard then optimum filter algorithm (top); optimum filter then standard algorithm (bottom).

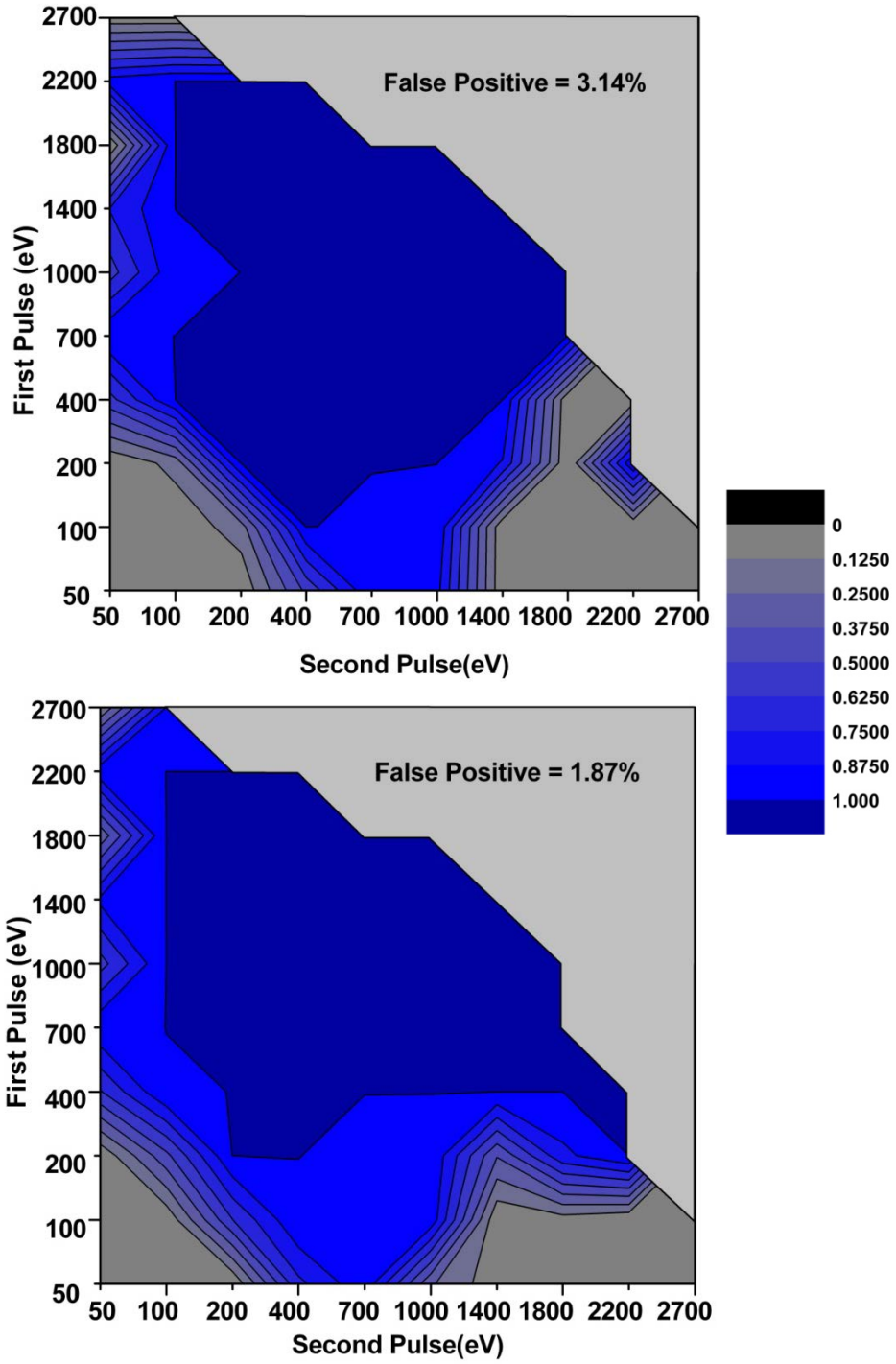


Fig. C.16; Comparison of results running both algorithms on the same data set with 1μs double pulse separation and 2 eV energy resolution: standard then optimum filter algorithm (top); optimum filter then standard algorithm (bottom).

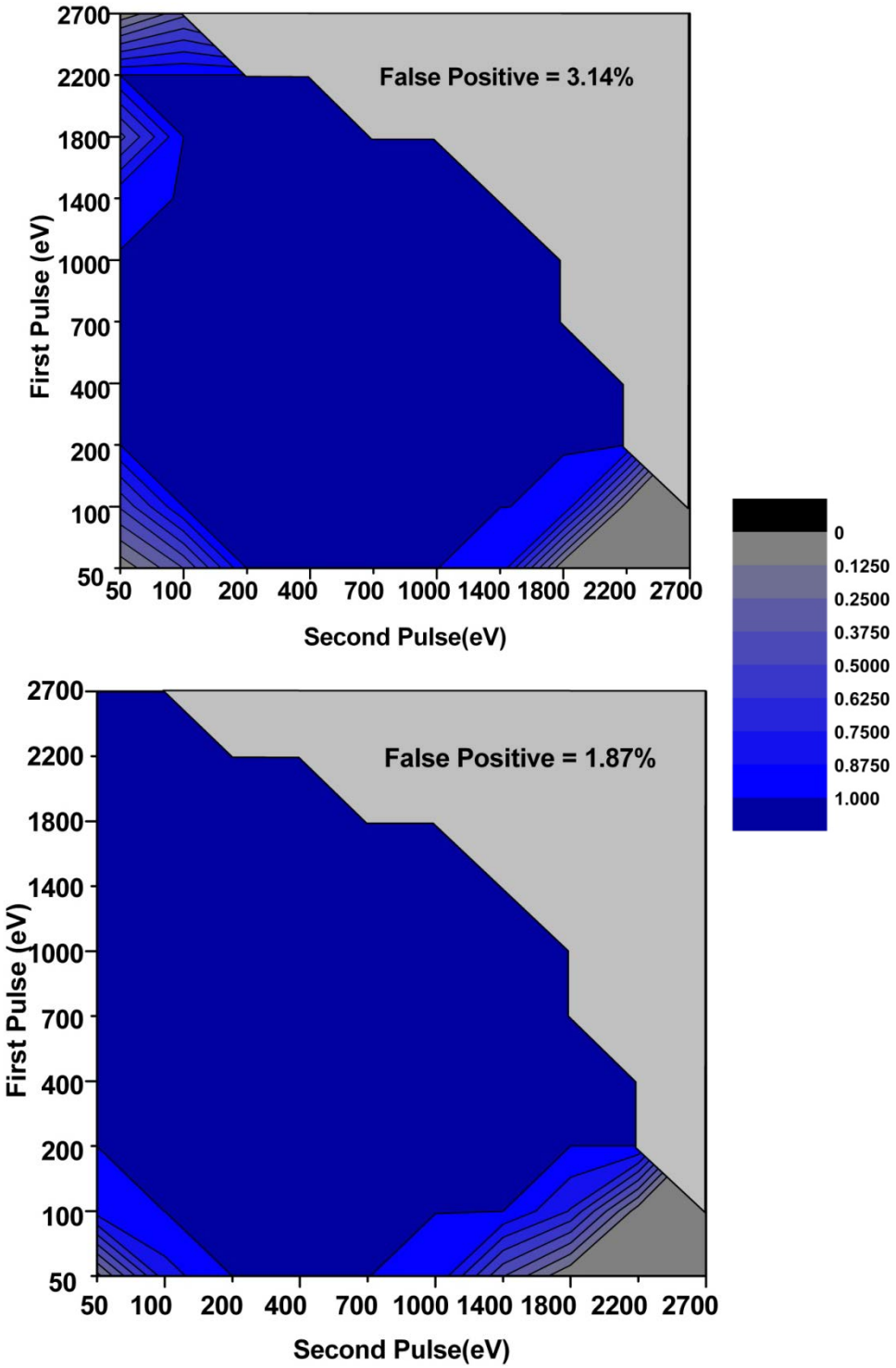


Fig. C.17; Comparison of results running both algorithms on the same data set with 2µs double pulse separation and 2 eV energy resolution: standard then optimum filter algorithm (top); optimum filter then standard algorithm (bottom).

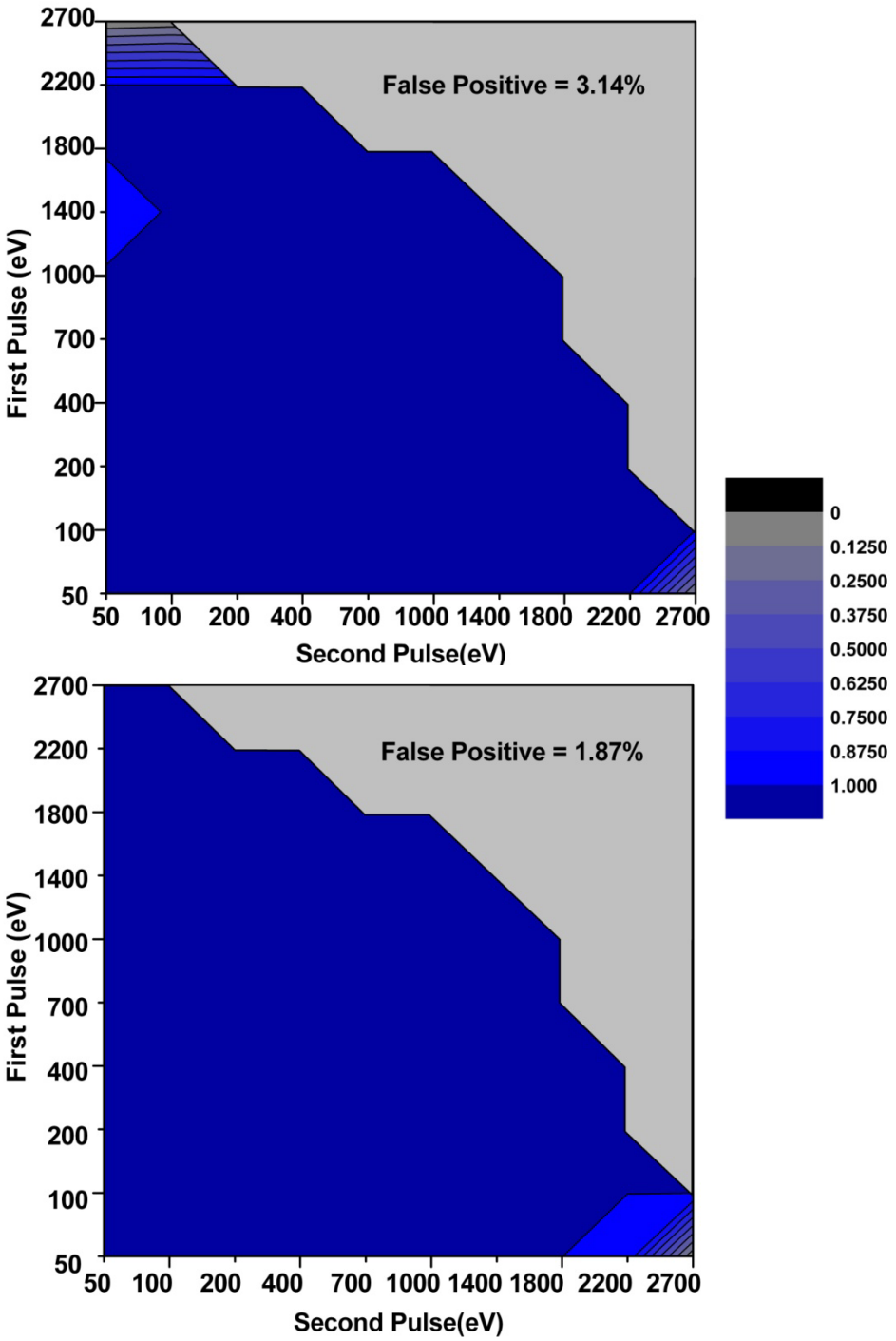


Fig. C.18; Comparison of results running both algorithms on the same data set with 5µs double pulse separation and 2 eV energy resolution: standard then optimum filter algorithm (top); optimum filter then standard algorithm (bottom).

References

[1]K. Kodama *et al*., Phys. Rev. D **78**, 052002 (2008).

[2]B. Kayser, arXiv:hep-ph/0211134v1

[3]K. Nakamura *et al.* (Particle Data Group), JP G **37**, 075021 (2010) and 2011 partial update for the 2012 edition (URL: http://pdg.lbl.gov). (Neutrino Mass, Mixing, and Oscillations)

[4]P. Stephen, *Proceedings of the 3rd International Workshop on NO-VE: Neutrino Oscillations in Venice*, Venice, FERMILAB-CONF-06-248-T (1996) pp. 115-125.

[5]Y. Fukuda *et al.*,arXiv:hep-ex/9812014v2

[6]S. Hannestad *et al*., arXiv:1004.0695.4

[7]M.C. Gonzalez-Garcia *et al*., arXiv:1006.3795

[8]H. Klapdor-Kleingrothaus, I.V. Krivosheina, Mod. Phys. Lett. **A21** (2006) 1547-1566

[9]H. Klapdor-Kleingrothaus, A. Dietz, and I.V. Krivosheina, Fnds. Of Phys. **32 no. 8** (2002)

[10]C. Kraus, *et al*., Eur. Phys. J. **C40** (2005).

[11]V. Lobashev, *et al*., Nucl. Phys. B (Proc. Suppl.) .

[12]F. Gatti, Nucl. Phys. B Proc. Suppl., **91**, 293 (2001).

[13]M. Sisti, et *al*., Nucl. Instr. and Meth., **A520**, 125 (2004).

[14]S.H. Moseley, J.C. Mather, and D. McCammon, J. Appl. Phys. **56**, 1257 (1984).

[15]M. Galeazzi, and D. McCammon, arXiv:astro-ph/0304397v1

[16]J.W. Appel and M. Galeazzi, arXiv:physics/0507011v2

[17]K. Irwin, Appl. Phys. Lett., **66**, 1998 (1995)

[18]A. Monfardini *et al*., AIP Conf. Proc., **882** (2007).

[19]A. Nucciotto, O. Cremonesi, and E. Ferri AIP Conf. Proc. **1185**, 689 (2009); doi 10.1063/1.3292435

[20]F. Gatti *et al*., J. of Low Temp. Phys., **151** (2007); doi 10.1007/s109090-008-9716-7

[21]S. Agostinelli, *et al*., Nucl Instr. andMeth. **A506**, 250 (2003); IEEE Transactions on Nuclear Science 53 No. 1,270 (2006).

[22]M. Galeazzi *et al*., AIP Conf. Proc. **1185**, 558 (2009); doi 10.1063/1.3292403

[23]T. Saab, et al., *J. Appl. Phys.* **102**, 104502 (2007).

[24]J.D. Armstrong, J. of Low Temp. Phys.; doi 10.1007/s10909-012-0570-2