SANDIA REPORT
SAND20XX-XXXX
Printed January 2020

Sandia National Laboratories

Annular Core Research Reactor (ACRR) Pulse Curve Characterization

David Hamilton Saucier, Edward Parma

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico
87185 and Livermore,
California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods/

ABSTRACT

Reactor pulse characterization at the Annular Core Research Reactor (ACRR) at Sandia Technical Area V (TA-V) is commonly done through photo conductive detection (PCD) and calorimeter detectors. Each of these offer a mode of analyzing a digital signal with different advantages/methods for determination of integrated dose or temporal based metrology. This report outlines a method and code that takes the millions of data points from such detectors and delivers a characteristic pulse trendline through two main methods: digital signal filtration and machine learning, in particular, Support Vector Machines (SVMs). Each method's endpoint is to deliver a characteristic curve for the many bucket environments of ACRR while considering other points of interest including delayed gamma removal for prompt dose metrics. This work draws and adds on previous work detailing the delayed gamma fraction contributions from CINDER simulations of the ACRR. Results from this project show a method to determine characteristic curves in a way that has previously been limited by data set size.

ACKNOWLEDGEMENTS

The author would like to thank Edward Parma for his mentorship and guidance for this project in addition to Taylor Lane for his SAND report and previous research characterizing delayed neutron fractions in ACRR. This work could also could not have been completed without the support of TA-V and Organization 1384.

CONTENTS

LIST OF FIGURES

This page left blank

ACRONYMS AND DEFINITIONS

Abbreviation	Definition
ACRR	Annular Core Research Reactor
CM	Calorimeter
DOE	Department of Energy
ENDF	Evaluated Nuclear Data File
FFT	Fast Fourier Transform
FREC-II	Fuel Ring External Cavity
GNC	Gaussian Normal Curve
LSQ	Least Squares Regression
MCNP6	Monte Carlo N-Particle Program v.6
NI	National Instruments
PPV	Point to Point Variation
RAM	Rolling Arithmetic Mean
SNL	Sandia National Laboratories
TA-V	Technical Area V
Tpk	Peak Time
TR	Transition Region

1. INTRODUCTION

Experimental data is taken at the Annular Core Research Reactor (ACRR) at Sandia Technical Area V (TA-V). ACRR is a unique research reactor in a deep swimming pool cavity that offers the capability of operation at steady-state, transient, or singular pulse mode. A schematic of ACRR given below shows the specifications of the reactor inclusive of its large experimental cavity size of nine inches in diameter. This experimental cavity spans up to the top of the reservoir to allow for placement of experimental packages in the reactor's dry cavity.

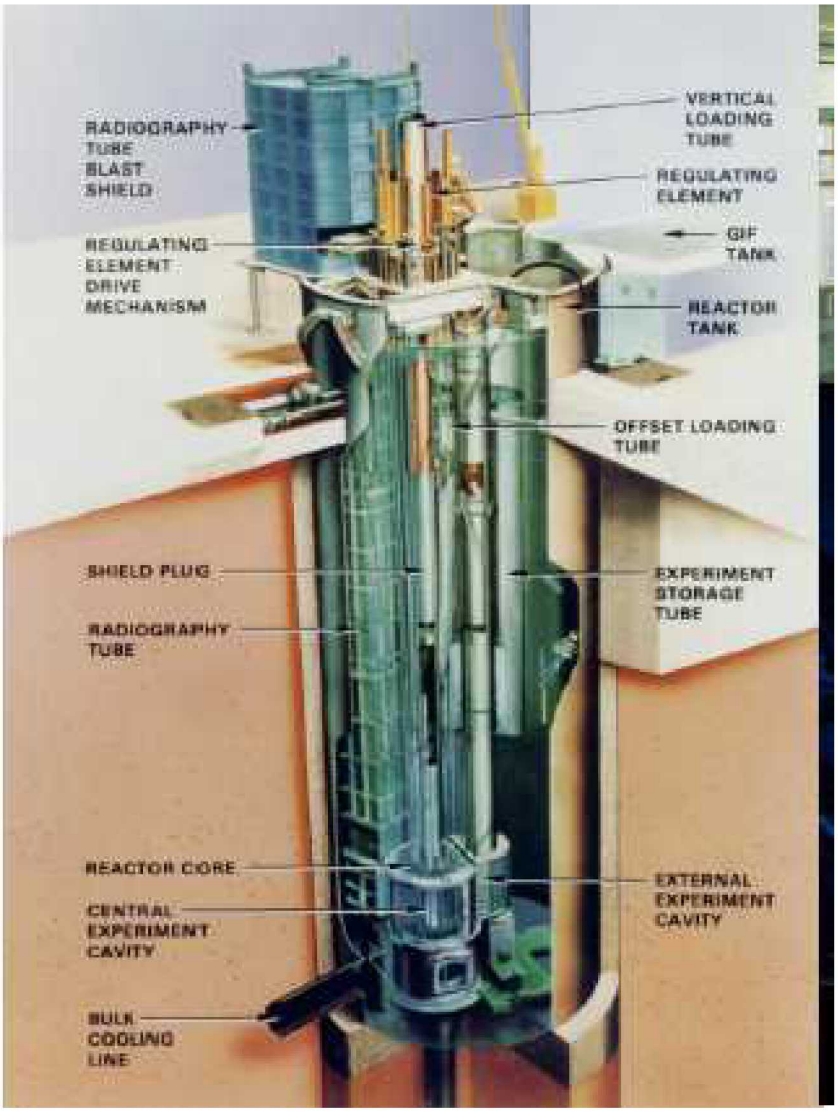


Figure 1-1. Schematic of the Annular Core Research Reactor (ACRR)

The ACRR is very in unique in the sense that the flux characteristics of the central cavity of ACRR is epithermal and may be operated at 4 MW at steady-state and 400 MJ (licensed for 500 MJ) in pulse

and transient modes, both of which are much higher than the pulse transients done at most TRIGA pulse reactors which operate in the realm of 10 MJ (Parma, 2009). Such an environment is achievable through in part to the unique Uranium Beryllium Oxide fuel that offers up favorable neutronic benefits through (γ, n) interactions in addition to high heat capacity. So beneficial is this effect that this core may be started up without use of a startup neutron source as in most other reactors (Parma, 2009) (Duderstadt & Hamilton, 1976). Neutron characteristics may be tailored to customers' needs by addition of 'buckets' to the central cavity to tune the neutron energy spectrum.

Such beneficial fuel neutronics will not be discussed in-depth in this paper, but a in depth explanation may be found in *Nuclear Reactor Analysis* (Duderstadt & Hamilton, 1976) under photo neutron source terms for the radiation transport equation.

It should be noted as well that the ACRR has an additional capability in the form of the Fuel Ring External Cavity (FREC-II) which houses a larger experimental diameter of 20 inches for larger experiments. This is shown below on Figure 1-2 while the reactor is operating at steady state. The ACRR is on the right with transient rods while the FREC is lifted into its steady-state position on the right (Parma, 2009).

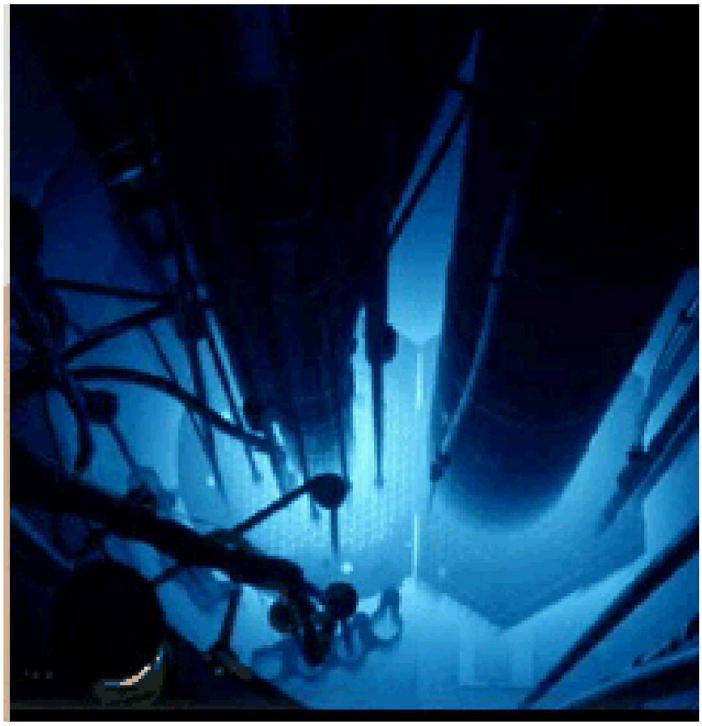


Figure 1-2. Schematic of the ACRR with FREC

The ACRR is one of the most unique reactors in the world, and it provides a key function for the mission here at Sandia Labs and this is no doubt in part to the unique neutronic and dose environment it gives for systematic interrogation of radiation dose.

One key take-away from the background of this reactor is that this fuel source contributes an artifact in two main ways: 1) dose response is over-estimated at the end of the pulse cycle slightly due to the

presence of photo neutrons and 2) ground-truth delayed critical reactor configuration is changed by the presence of the additional neutrons from the (γ, n) reaction from the beryllium. Methods to try and address this are mentioned in the methodology. Such a visualization is given below on Figure 1-3 (Bell & Glasstone, 1970) , (Duderstadt & Hamilton, 1976) ,(Parma, 2009).

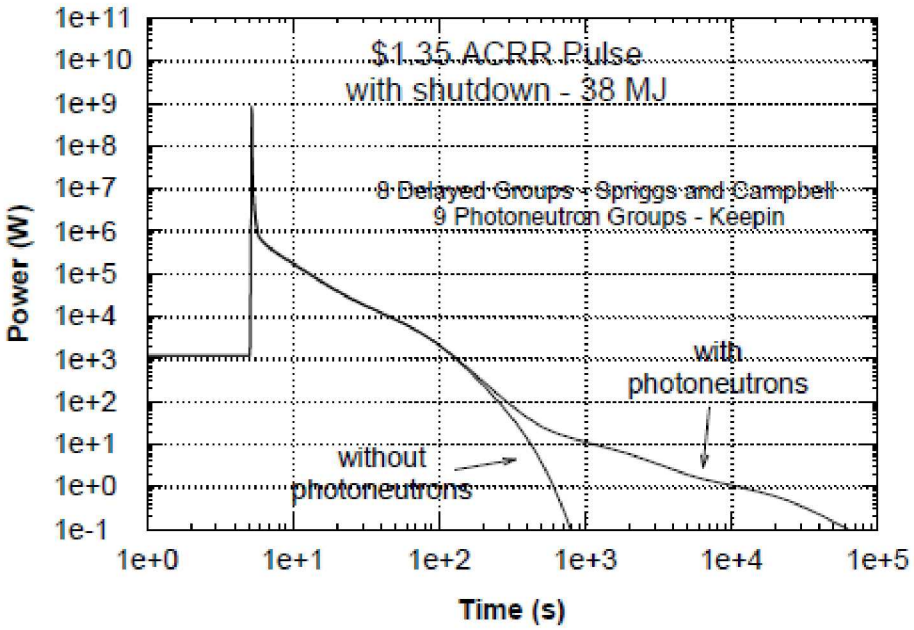


Figure 1-3. ACRR P(t) During 1.35$ Pulse Showing Importance of Photo neutrons on Dose (Parma, 2009).

Keeping that P(t) has a correlation with neutron fluence, photo neutrons influence the integrated dose given to the central cavity. How this is removed or accounted for is explained in the methodology section of this report.

1.1. Prompt and Delayed Environments in ACRR

One of the key uses for ACRR is the irradiation of electronic components. In ACRR, packages are exposed to prompt neutron and gammas with the addition of delayed neutrons and gammas because of the fission products. These prompt and delayed contributions to dose present a challenge in the form of determining each one's contributions to the electrical component, which is addressed and given in this experimental code package thanks in major part to work from Taylor Lane in his SAND report (Lane & Parma, 2015).

Prompt and delayed neutron emission environments are arenas of active study in the realm of nuclear engineering. Many software packages have been implemented to allow the user to bin which energy domains offer dose contributions to the radiation environment. In the case of ACRR, data can be obtained through the Monte Carlo N-Particle Program (MCNP6) by binning the energy domains of interest for neutrons (both delayed and prompt) (Brown, 2018). This has been a relatively new addition to the code that was implemented around 2012. These methods are readily available and necessary for the ACRR for each of the radiation environments provided by the cavity buckets.

To properly determine the dose to the item, an accurate estimate of the flux/fluence from gamma and neutrons must be made.

Dose characteristics of photon emission in reactor environments, let alone like one in the ACRR, are unfortunately very small in number. Characterization of photons, both prompt and delayed, usually are very strong functions of the fuel, and currently Evaluated Nuclear Data Files (ENDF) files usually only provide a singular value that comes from different, potentially variable experimental acquisition states (Grassberger, Hegger, Kantz, Schaffrath, & Schreiber, 1993) (Brown, 2018).Temporal interrogation of delayed gammas has remained relatively unstudied for many years and even a relatively recent ENDF/B-VII distribution had nearly 30% uncertainty with delayed gamma energy spectrum for ^{235}U (Lane & Parma, 2015). This is since been improved and the flux characterization for the function of this report looks at the delayed photon flux/fluence as a function of the data produced by the CINDER 2008 photon simulations of ACRR.

Objective: Removing the total contribution of these delayed neutrons over the entire time series of the data set to determine the purely prompt contribution. Other contributing dose factors may be easily extrapolated from prompt data.

1.2. Big Data and Filtration Methods

Big data presents challenges in the sheer large dimensionality and corresponding interpretation of data structures. While it might seem beneficial to have millions of data points, the challenge of interpretation and validation of useful structures gets exponentially more difficult. This is certainly true for the data sets taken using PCD and calorimeter (CM) detectors at the ACRR.

A typical dataset from ACRR has on the order of 1 to 3 million data points per channel with an average of about ten different detection channels for some experiments. Loading a single data

channel into Excel alone would crash from memory issues. Initially to answer this problem, the author looked to using simple data filtration methods.

1.2.1. Data Filtration/Reduction

The first alternative to reducing data dimensionality is to determine the amount of noise. To do this, most filters assume that the noise comes from a given frequency. In the case of this data, Fourier decomposition and consultation with experimenters suggested that higher frequency electronic noise was infiltrating the data. Such a problem as this can be addressed using the following: a lowpass, high pass, bandpass, or band stop filter. These are relatively simplistic heuristics of data filtration that are summarized below on Figure 1-4.

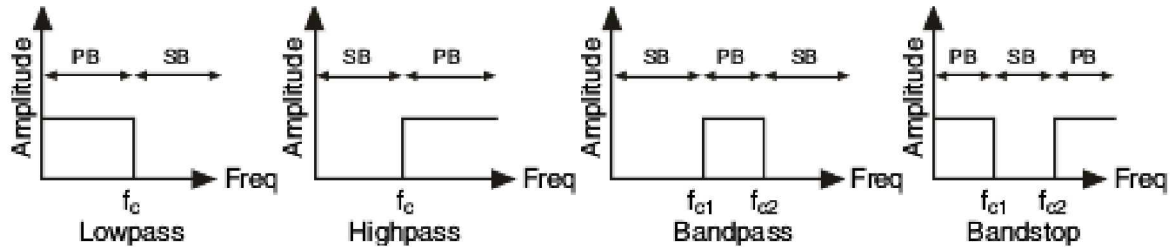


Figure 1-4. Data Filtration Heuristics for Ideal Situations (National Instruments , 2018).

In Figure 1-4 above, SB denotes the stopband or threshold where all data reads stop and PB denotes the passband threshold where all reads are kept. These above examples show a very idealistic way of data filtration that would have to be applied to what experimenter needs are (Grassberger, Hegger, Kantz, Schaffrath, & Schreiber, 1993).

Realistically, we do not know what the exact cutoff should be experimentally as that can change greatly depending on experimental setup. To address such concerns, what is commonly done is to introduce a transition region (TR) that attenuates data. The above figures are step functions and typically are too restrictive.

When trying to determine what would be the best filter to use, many times it helps to determine the frequency distribution using a Fast Fourier Transform (FFT) which is described by Equation 1.

$$X_k = \Sigma x_n e^{-\frac{i2\pi kn}{N}} \; k = 0, \ldots N-1, n = 0 \rightarrow N-1 \qquad Eqn(1)$$

What is really being computed here is a discrete Fourier transform (DFT). This helps reduce a noisy data set to a collection of frequencies. Below on Figures 1-5 and 1-6 are examples of ACRR data sets before and after filtration through Bessel functions.

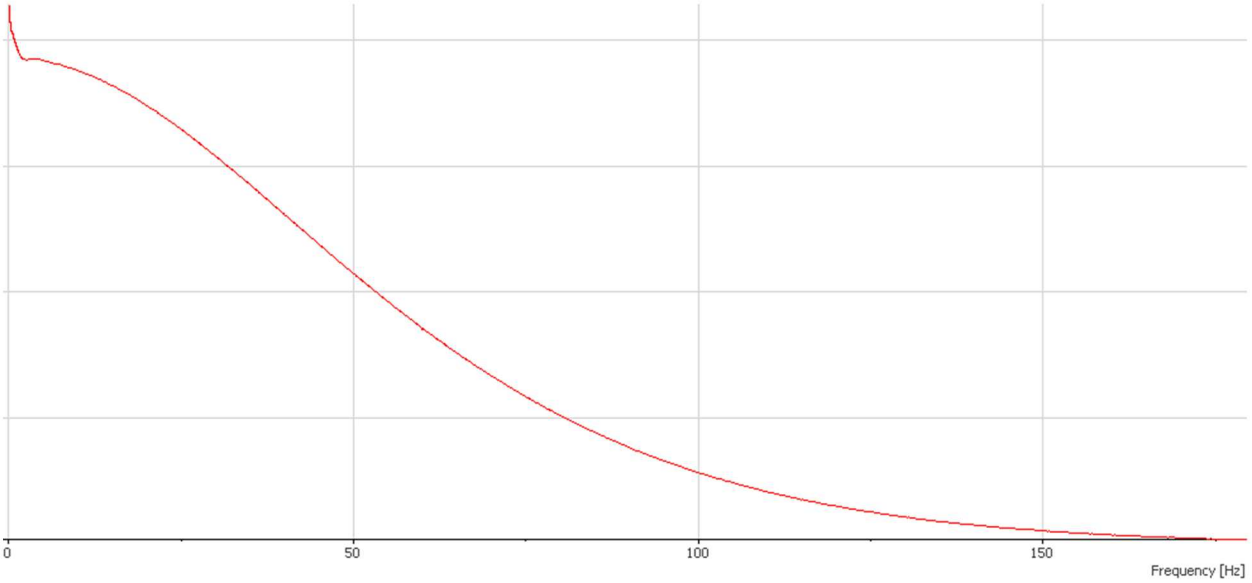


Figure 1-5. Fast Fourier Transform on PCD Noisy Data

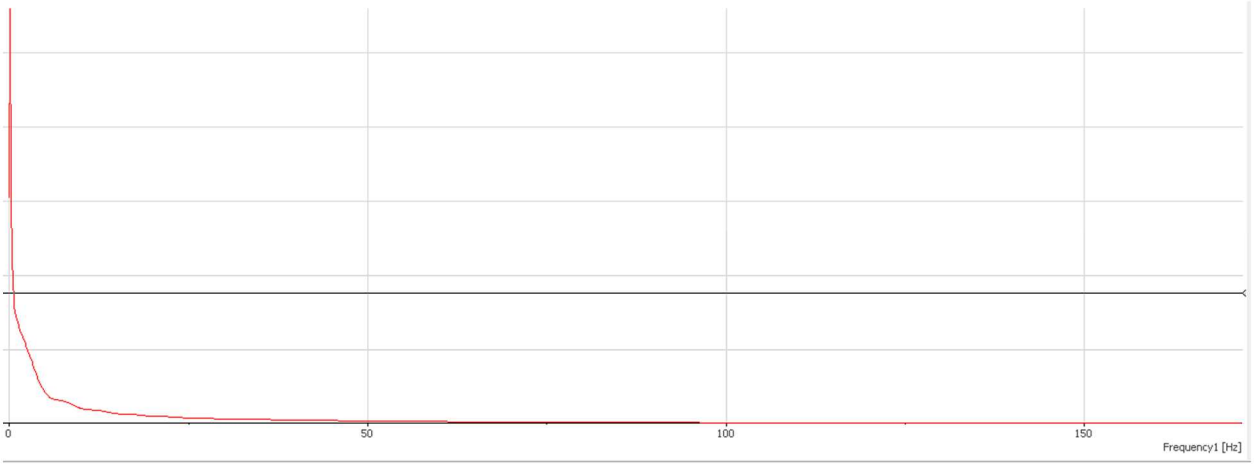


Figure 1-6. Fast Fourier Transform on PCD Data with Bessel Filtration

These curves might appear to be esoteric or unmeaningful, but the true beauty in the FFT is found through comparison of these with an FFT of a Gaussian normal curve (GNC). This is shown below on Figure 1-7.

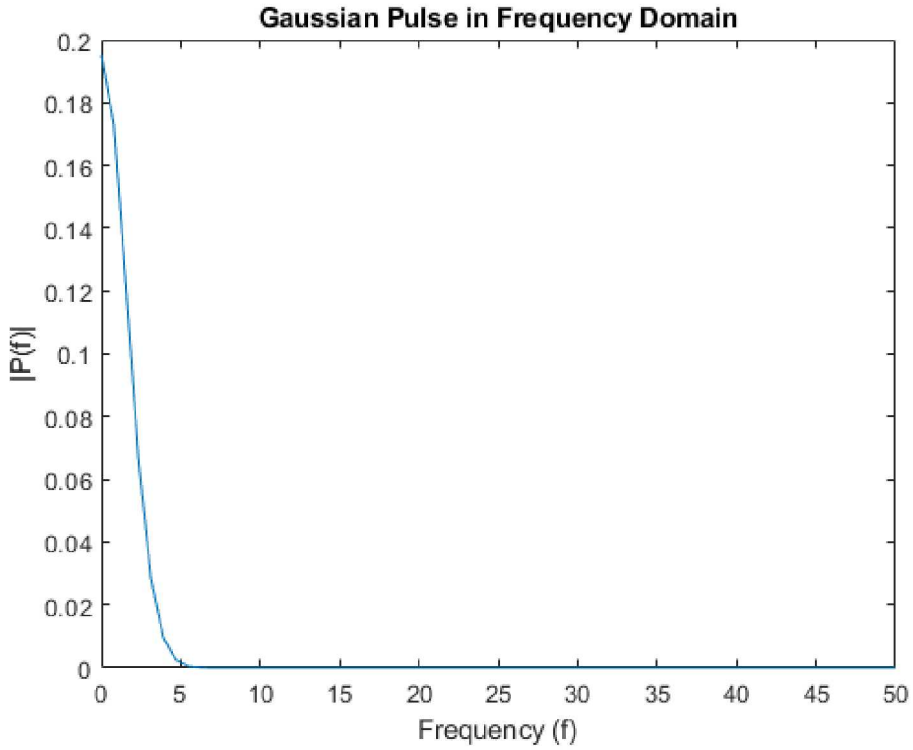


Figure 1-7. Fast Fourier Transform of Gaussian Normal Distribution

What should be noted is the shape of each of these distributions. The author of this report started the shaping of the filter PB by looking at what a characteristic pulse might look like, which was assumed to be roughly that of a GBC. It should be noted that with removal of delayed gammas, this assumption is not too unreasonable, with a rough similarity close to a Gaussian as determined through a Chi-Square test ($P<0.05$ as threshold of significance).

This data pipeline utilizes a Bessel filtration method for the data stream. The other options along with their TR behavior are given below on Figure 1-8.

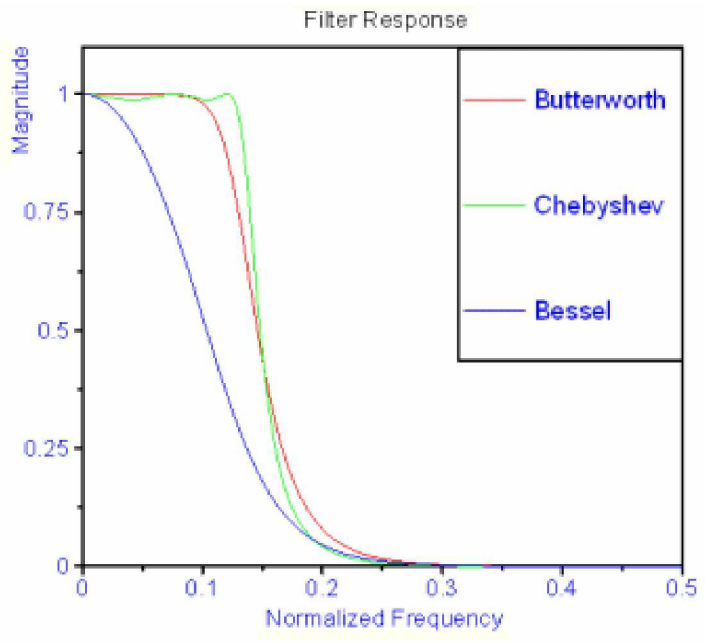


Figure 1-8. Digital Filtration Profiles of Bessel, Chebyshev, and Butterworth Filters

Out of each of these, Bessel offers benefit by being the largest TR with non-rigid attenuation like Butterworth or Chebyshev. For a low-pass filter, it offers added benefit of being continuous and smooth on the stopband and passband.

Future developments of this detection method will likely utilize hybridized methods or qualifiers that better detection of valuable data during the experimental envelope. These methods given are ones that are currently used and available in the Diadem infrastructure. A current limitation of the Diadem software is that these more advanced detectors will need to be implemented using custom, user-created and coded methods.

1.2.2. Machine Learning Methods For Data Filtration

Machine learning is an unfortunately overused, broad term that is at times poorly defined. For the purposes of this report, there are two overarching flavors of machine learning:

- Supervised Learning: Consists of having a specific subset of paired input and output data. Commonly this is applied for use in classification of new inputs to a given output family
- Unsupervised Learning: Consisting of NO prior data labelling and only pure heuristic development using a set classification method. Typically, this is in the form of clustering analyses to find a hidden pattern in the data structures.

Data structures from ACRR offer the ability to do both alternatives, i.e.:

- Supervised Learning: Using prior dataset characteristics to predict or classify a new experimental environment as one or many buckets. (This could even be separated as a function of energy for more interesting datasets).
- Unsupervised Learning: Aggregate all ACRR data for each detector and determine if each experimental spectrum clusters with a given environment, energy or mixture of all of them. This could be done to determine which experiments have equivalent environments across all experimental data sets.

It should be noted that this was created for a limited dataset. The author had access to 3 different experimental runs and was able to determine characteristic curves using data splitting to increase statistical power by 3^N. Given more time, the author would have liked to expand this and analyze the rigor of the statistical assumptions of splitting each data channel and smoothing.

The proper conditioning of this data for the hopeful future user of these methods would be to have a dataset with around 4 test runs 3 validation runs and then as many experimental characterization runs that are required for the given requirements of the experiment. These are still lower than ideal, but for the future, when such data sets are done routinely, this is essential.

2. METHODOLOGY & RESULTS

Detector data from Self-Powered Neutron Detectors (SPND), Photo Counting Detectors(PCD), and Calorimetry based data have a great deal of signal noise when it comes to variance of the voltage signal It becomes clear that to work with a large data set with large inter-temporal variance, it is best to start with a simple, robust method. To start, we begin with a typical SPND and calorimeter channel (Figure 2-1 and 2-2 respectively) below.

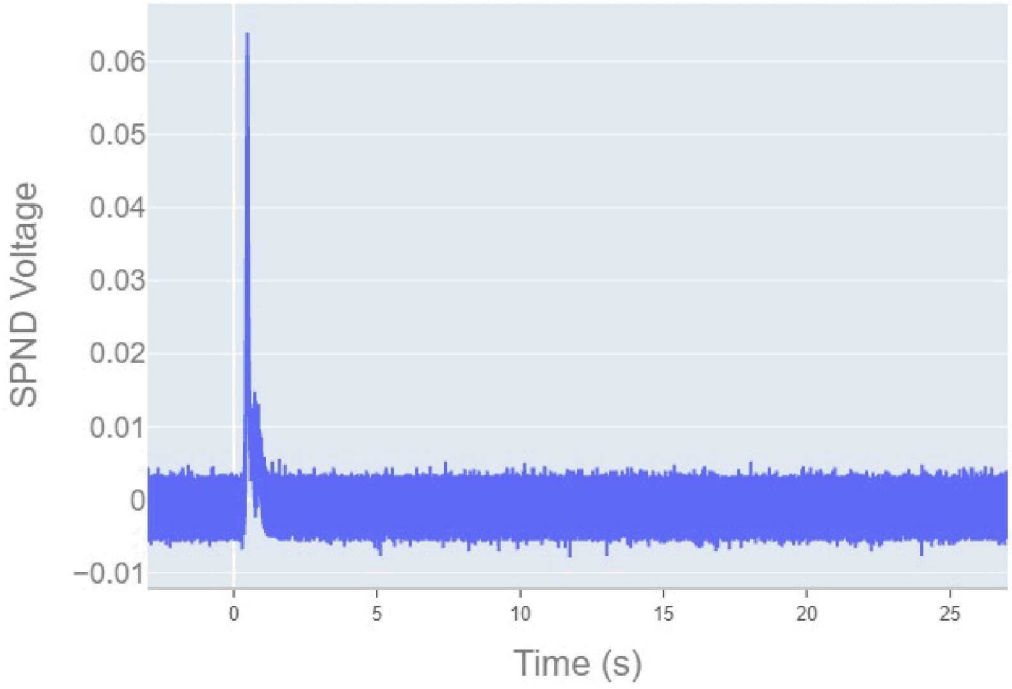


Figure 2-1. SPND Raw Data Signal from 40 MJ Pulse

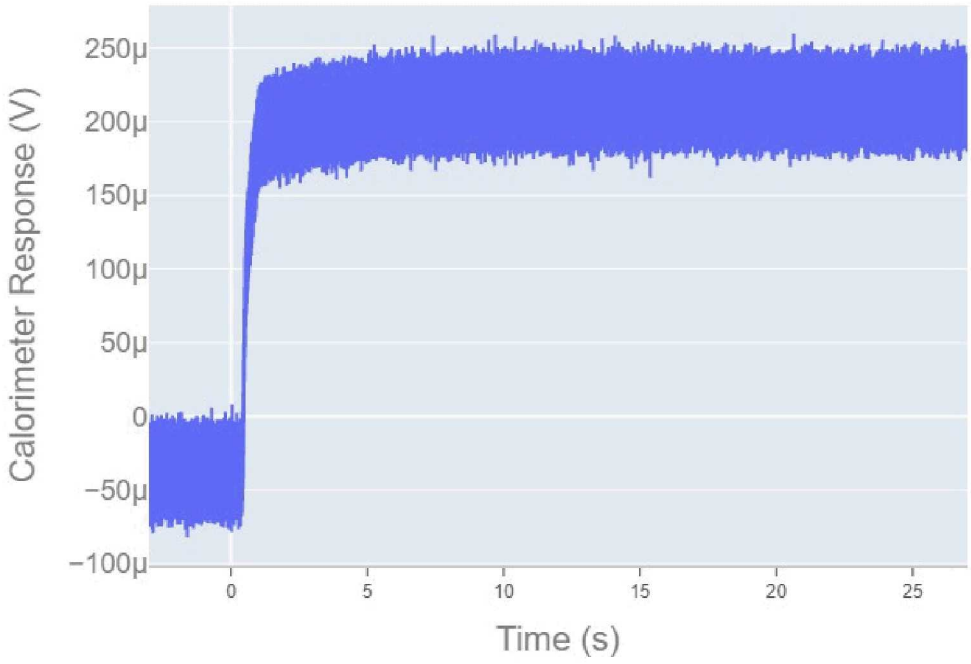


Figure 2-2. Silicon Calorimeter Raw Data From 40 MJ Pulse

These detection methods are displayed because they show two different contributions to radiation dose. SPNDs generate their current by neutron activation and corresponding beta decay of the rhodium material in the central portion of the detector. They require no signal amplification and are used as a method of active neutron flux measurements. Calorimeters show the integrated dose via heat deposition, which ultimately helps determine how much dose was given to the sample after calculating parameters from dosimetry. Different materials such as bismuth or tin will give different information to the integrated dose for a sample.

The point-to-point variance (PPV) is quite large. This presents immediate difficulty when trying to do line regressions or trendlines because PPV makes it nearly impossible to determine where the ground truth characterization might lie. An unavoidable data artifact found after conferring with experimenters is the presence of high-frequency noise, which is from the experimental equipment itself. One of the goals of this project was to find a way to filter out these high frequencies with minimal experimental data loss in addition to other forms of signal noise introduction from detection instruments.

2.1. PCD and Calorimeter Filtration

To begin to filter the data, it must be stated that there are two primary issues facing the data:

1) A large data set consisting of millions of data points overwhelms commonly used data analytic software.
2) PPV Noise is incredibly high and we must remove the variance while preserving the general shape or characteristic of the dose curve.

These are first addressed through use of the filtration methods mentioned in the previous introduction section. After optimization and some trial and error, the filtration method chosen for this was the Bessel filtration method. Butterworth and Chebyshev filters were ultimately dismissed due to their over conservative SB (Grassberger, Hegger, Kantz, Schaffrath, & Schreiber, 1993).

Bessel edge filters were chosen after a quick comparison of chi-square values between filtered pulse shapes, and a gaussian. The underlying assumption here that a pulse has the rough characteristics of a gaussian during peak time. There are better distributions to model this such as a log-normal distribution, but the limitation of time lead to the choice of the normal curve being the optimization parameter.

Filtered data sets were input in National Instruments (NI) Diadem and then altered using digital filtration lowpass filters with Bessel edges. The results of this filtration are shown below on Figures 2-3 and 2-4. Logarithmic amplification (log-amp) is done on this PCD to detect with fine temporal resolution when the ACRR pulse peak contributions from gammas are given as well as the dose response at this time due to the prompt gamma emissions. Working in the log-log space allows these rapid events to be captured. The plot below on Figure 2-3 is inverted due to the experimental acquisition setup. The data may be retransformed into an upright log-amp curve using a conversion table for the Yokogawa data. Such an example is given on Figure 2-4.

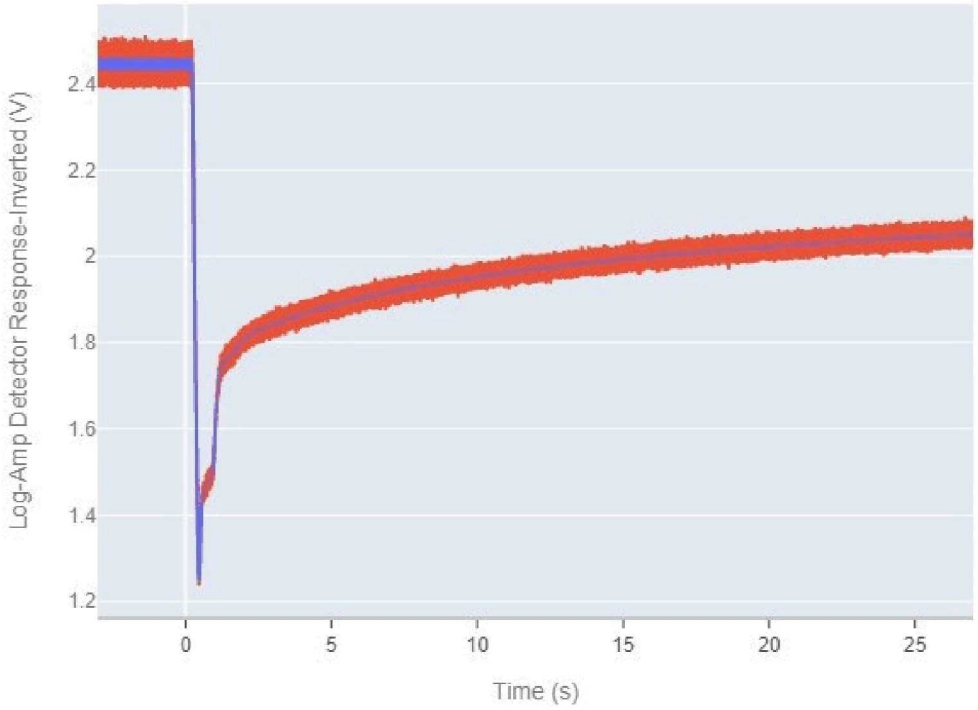


Figure 2-3. Bessel Low Pass Filtered PCD Detector Data (Log-Log, Red-Raw Data, Purple-Filtered Data)

This noise reduction was speculated to be further be improved through a moving average to reduce the data load. This can be pre-conditioned in the code by inputting the number of data points that you would like to 'skip,' but the user is taking a million and more data points and putting it in a data format that could be transferred to a different IDE platform.

With this initial filtration, it can be further improved with smoothing using the rolling arithmetic mean (RAM). This is a computationally inexpensive process for big data. This greatly improves the fidelity of the data that is being used. A comparison is given below on Figure 2-5.

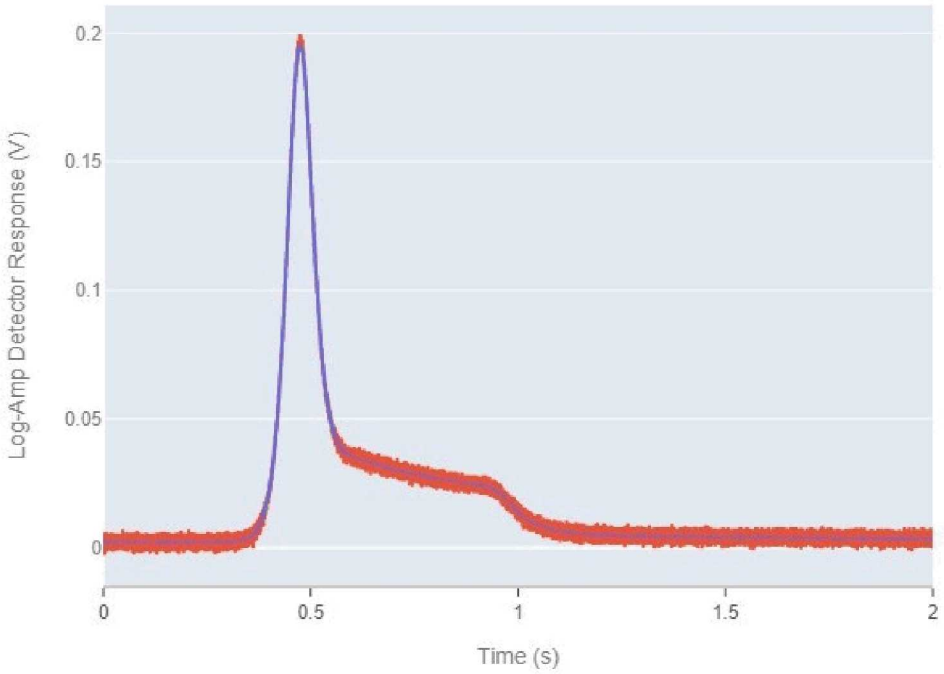


Figure 2-4. Reference Log Amp Data with Filtered Data (Red- Raw Data, Purple- Filtered Data)

At this scale, there is a difference in the absolute minima of the smoothed curve as well as the raw data. Being in logarithmic space, this presents a problem in that this difference is further exaggerated outside of log space. This is speculated anecdotally to be improved through use of tuning the window length of the moving average post smoothing. A smaller window compresses the data less, but in turn would encapsulate the shape of the spectrum better. This is the inherent trade-off in this question, and best practices to optimize are in the hands of the experimenter to determine which is best: more data compression or a shape closer to the experimental data.

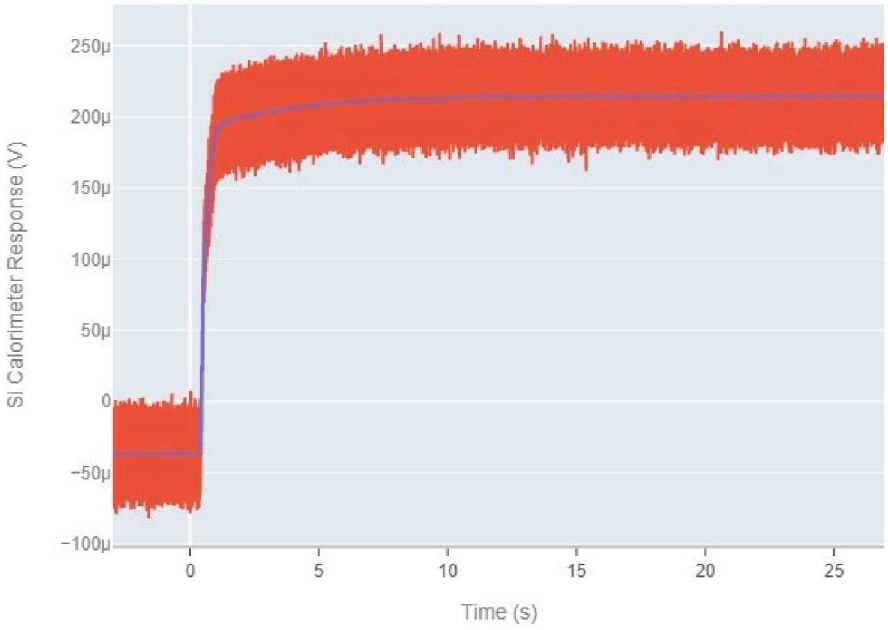


Figure 2-5. Comparison of Raw and Smoothed Silicon Calorimeter Data (Red-Raw Data, Purple-Filtered Data)

These filtration methods are an important starting point for the overarching aim of this code: to generate characteristic curves of dose environment that are manageable. These filtration steps are used in all analyses of this tool box: regression, derivation analysis of peak regions, logarithm transformations, and finally power analyses (both with delayed gammas and without).

2.2. Filtration and Beyond: Regression and Derivative Peak Analysis

2.2.1. Derivative Peak Analysis

After the filtration, the experimenter wants to know about the transient data of the peak, so skipping data points in this region offers little value and can even remove valuable data during transients. To address this, this code does the following method:

$$T_{critical} = \frac{d}{dt}(V_i) = 0 \quad Eqn\ (2).$$

The largest maxima is the one found from the transient with a few fluctuations just before the pulse. Taking the derivative of the smoothed data we get the following diagram (Figure 2-6):

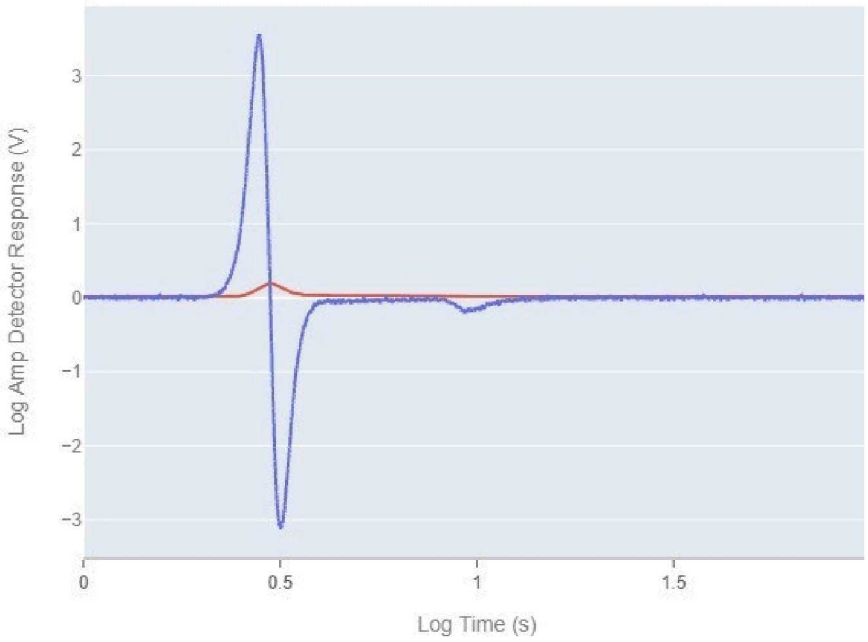


Figure 2-6. Pulse Time Detection Using Derivative of Detection Response: Log-Amp Channel (Purple- Derivative of Filtered Channel, Red- PCD Log Amp Channel)

The data show that the $T_{critical}$ has a slight lag from the true peak time from the data. For the purposes of this code, it serves as a good peak time indicator that is not sensitive to the slight perturbations seen and gets the main global maxima. The total time envelope in which we will create an exclusion region for the data averaging and smoothing is the following:

$$\Delta T_{peak} = FWHM\left(\frac{d}{dt}(V_{PCD})\right) Eqn(3).$$

In practice, this should work well as we are looking at a large full-width at half-maximum (FWHM) of the derivative of the time dependent voltage. This helps to ensure that the data is preserved liberally at the area around the transient peak, and the derivative is the most robust global maxima detector currently available.

Derivative peak analysis presented an initial challenge to calorimeter data. This was done using a similar method, but rather focused on finding the critical time on the second derivative of the peak data or when the rate was increasing the most. Or in equation format:

$$T_{critical_{Cal}} = \max\left(\frac{d^2}{dt^2}V_{cal}(t) = 0\right) \quad Eqn(4).$$

The time envelope could not be calculated using FWHM, so for calorimeters it is purely based on when the maximum second derivative begins to when the first derivative is zero. This time envelope is something that could be readily improved through either experimenter input or more rigorous mathematical methods.

2.2.2. Data Regression

Unlike using the derivative, which only applies to the data from PCD detectors, proper data regression is central to determining the characteristic of experimental data. The predominant method of regression here is logarithm transformation and then linear regression. There are better methods of regression such as Least Squares (LSQ) or even some machine learning methods that will be discussed in future sections. For the sake of brevity, this can be done using a logarithm regression in Diadem directly or by fitting in Excel after running the script to reduce the data. Such a graph is shown below on Figure 2-7.

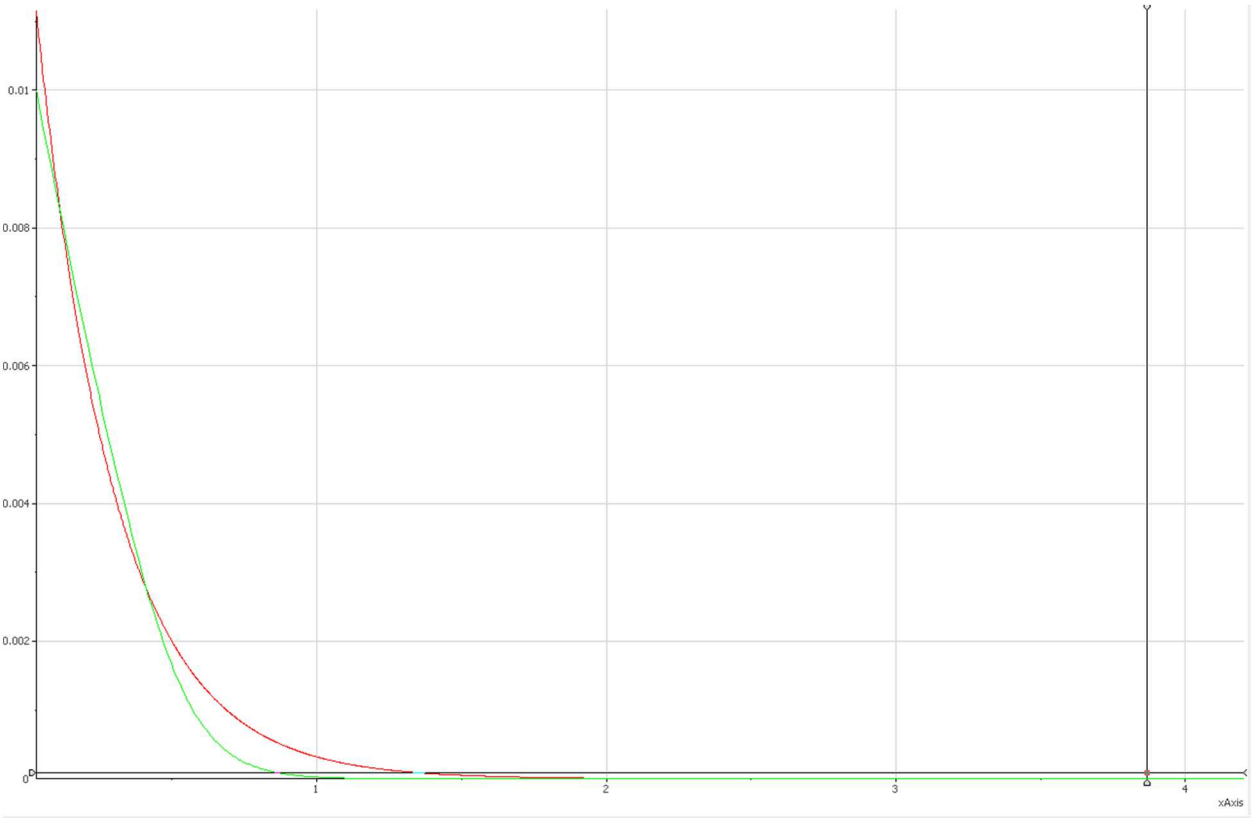


Figure 2-7. Diadem Logarithmic Data Regression on Log-transformed Data (G-Log Regression, R-PCD)

Diadem is not adept at regression. For this, the author highly recommends the use of Python, MATLAB or other scientific software that would make use of more advanced methods. (Which can be easily done after compressing the data in this code!)

2.3. Delayed Gamma Removal

Delayed gamma removal was one of the main goals of this project. To do this, the fractional energy distributions were needed for ^{235}U. Raw data for these were taken from (Parma, 2009), (Lane & Parma, 2015), and relevant parameters are given below on Figure 2-8.

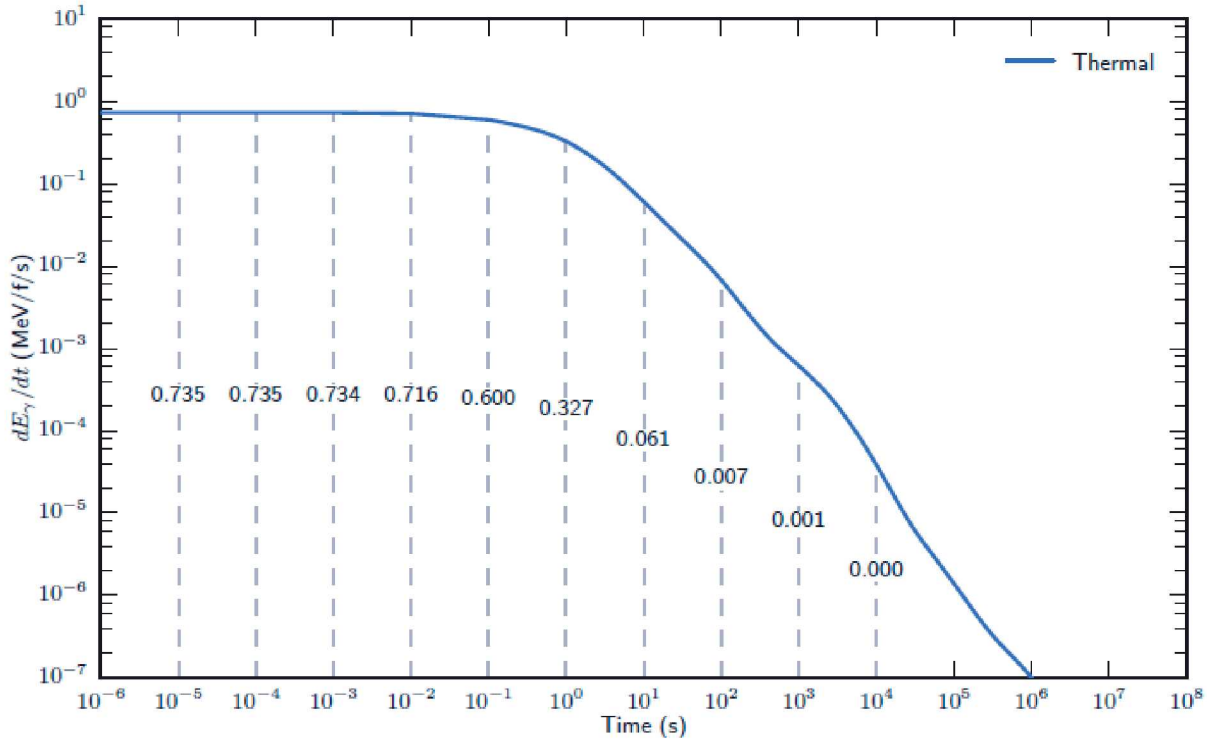


Figure 2-8. Differential Delayed Gamma Energy for ^{235}U Thermal Fission (Lane & Parma, 2015)

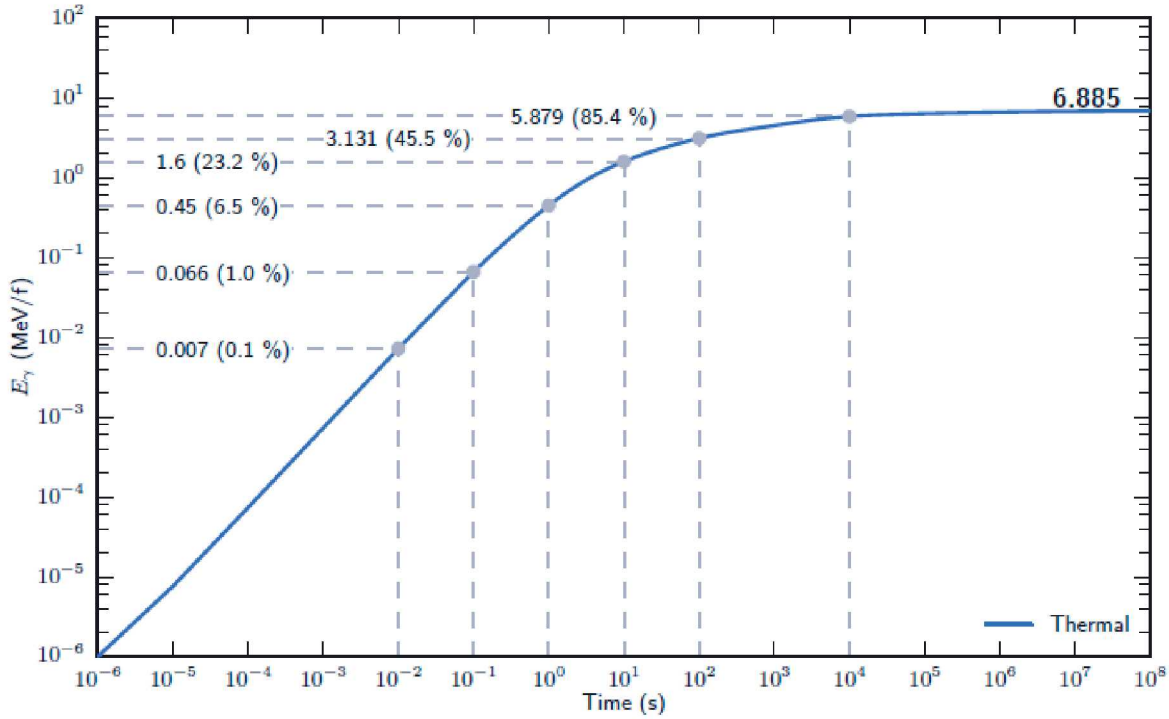


Figure 2-9. Delayed Gamma Energy Curve for ^{235}U at Thermal Temperatures Fission (Lane & Parma, 2015).

These curves' data was tabulated and parsed for input into Diadem. These data show the expected amount of delayed gamma energy deposited after a given time (Lane & Parma, 2015). This would mean that for experimental packages left in for a very long time, they would have a substantial additional dose. Using this data, we calculate the delayed gamma fraction to subtract from our gamma detectors to get the pure contributions of dose from prompt gammas. This is given below on Figure 2-10.

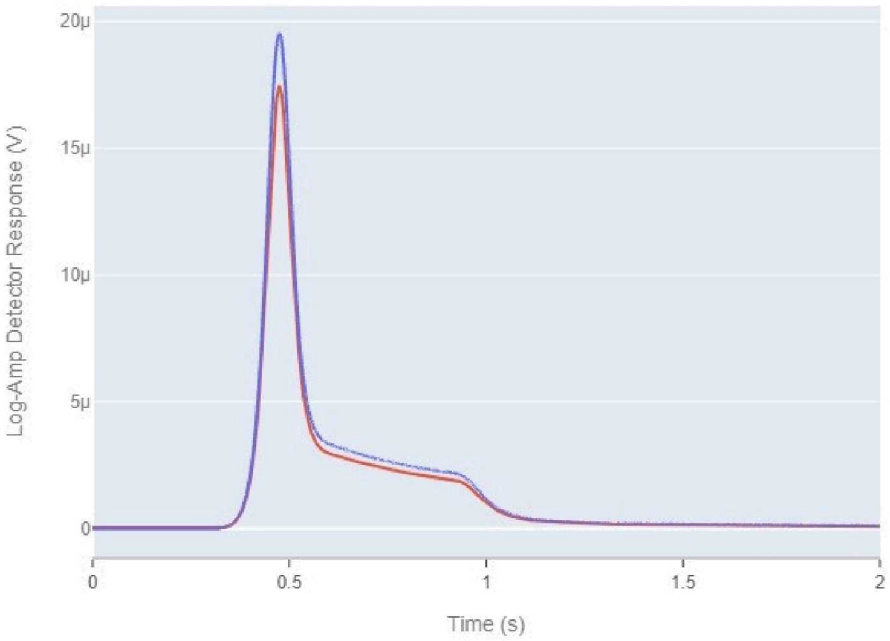


Figure 2-10. PCD Log-Amp Detector Channel Data with Subtracted Delayed Gamma Dose (Red- Total Detector Response, Purple- Prompt Gamma Response)

One item of interest is the peak difference after delayed gamma contributions. It is not currently known as to the exact reason for this depression in data. This was unexpected and not what we would predict from other methods. After researching into the reasoning, it is believed to be tied to the approximation made for the dE/dt distribution given from the Lane report. This approximation is a rather simple linear regression, that doesn't encapsulate the finer nuances of the data. In the case of the peak, where fine temporal and fit resolution is necessary, a small residual in the regression could explain this larger depression. This data could be improved greatly by improving this regression. Outside of peak data, there is decrease in the total detector response from delayed gamma subtraction, but this method has yet to be compared to other methods or experimental methods. Once that is completed, it would offer more valuable insight into what is the pure prompt gamma contributions to the dose environment.

2.4. Power Conversion

Putting voltage into a relevant unit of power is necessary for an endpoint deliverable. Admittedly, this was fairly tricky to do because of detection of the peak and its timepoint. Integration over noise would lead to a false normalization and incorrect power. This is accomplished by using a relatively simple power scale conversion as well as a first-derivative slope-based detector to determine the peak time or (Tpk). This power conversion is given below on Equation 5 and 6 for logarithm power. These equations are taken from a conversion based on dosimetry data (primarily Nickle foils). These results have been compiled by a worksheet generated by Elliott Pelfrey, and offer the following means as a conversion to power. It should be noted that ideally, this should be done on a

experimental basis, matching the reactor foil dosimetry to generate the constants given below. In the case of Equations 5 and 6, these values are an average of those used for a 40 MJ pulse (the data presented in this report). These values will change dependent upon the Peak power as well as other experimental parameters.

$$P = \frac{\Sigma(\{V_i - \sigma_{noise}\})}{\Sigma V_{i_{peak}}} * -0.0299(T_{peak} + 1) + 0.8077 \qquad Eqn\ (5)$$

$$P_{log} = \frac{\Sigma\{\log(V_i) - \log(\sigma_{noise})\}}{\Sigma\left(\log\left(V_{i_{peak}}\right)\right)} * \log(-0.0299(T_{peak} + 1) + 0.8077) \qquad Eqn(6)$$

2.5. Machine Learning Methods

Machine learning methods are a unfortunately vague term that often do not include the central methodology of the method. To avoid such misgivings here, the underlying proposed methodology is a supervised learning based Support Vector Machine (SVM). For clarity this means the following:

- Training data sets are required to set limits for distribution features
- Final output is a classification
    - ACRR Bucket Environment was used as a first example
    - Relied on conserved distribution shapes as determinants
- Qualitative data output could be quantified as % likelihoods.
    - Ex) 80% most like PbB bucket 20% like no bucket in ACRR cavity
- Offer a quick diagnostic for anomalous data for experimenters

Using these basic parameters, a challenge was to generate a training data set to try these methods. Data was limited for ACRR pulses, so ultimately the choice was to use one of the prior weakness of too much data in our favor. Simply put, make three data sets for one distribution environment by taking every first, second, and third data point repeatedly and use a moving average characteristic and other smoothing techniques to determine three curves for one data set.

Keep in mind, this method is fundamentally due to the lack of data the author had. With more data, this would be a very powerful method to train a simple SVM to classify experiments with more confidence. For a visual depiction of an SVM, please refer to Figure 2-11 below (Mountrakis, Im, & Ogeole, 2010). This shows a SVM hyperplane, denoted in this report as a Decision plane, along with two populations of data, represented by 'x' symbols and 'o' symbols. The hyperplane is generated by observing where most of a certain data type falls and trends. The more points that are along this path, the stronger that support vector is, and this in-turn helps set the linear threshold for classification. The width between the support vector for both 'x' data and 'o' data is called the margin with the middle being set as the SVM hyperplane.

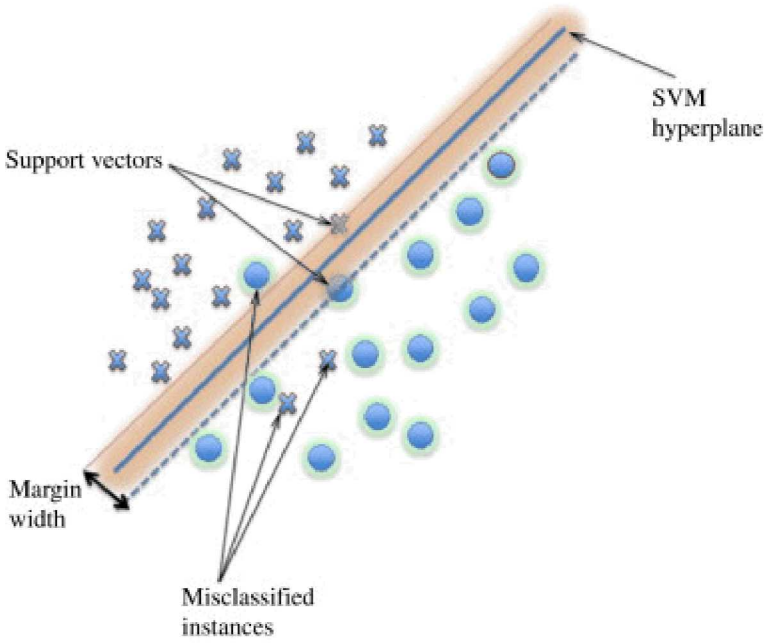


Figure 2-11. SVM Machine Learning Classifier Schematic

It is important to mention that data or distributions can be misclassified. In this case, these misclassified instances are where our data has additional complexities that aren't captured by a linear generalization. Fundamentally we are reducing the dimensionality of our data to something more general, so data may be classified. Data points inside of the margin can be rejected or, better yet, chosen as a validation set for when more training set data is available. At its essence, it's a well-informed Least Squares like algorithm that doesn't rely purely on fitting the shape but the features of the data set, which are much less limiting (Bishop, 2006).

When applied to ACRR data, this was tested using the different CM material detectors: Sn, Bi, and Zr. However, this method could be applied to any data set that have roughly similar environments with enough historical data to serve as a training set. For the scope of this report, features were determined based on histogram average values. Such a plot is given below on figure 2-13.

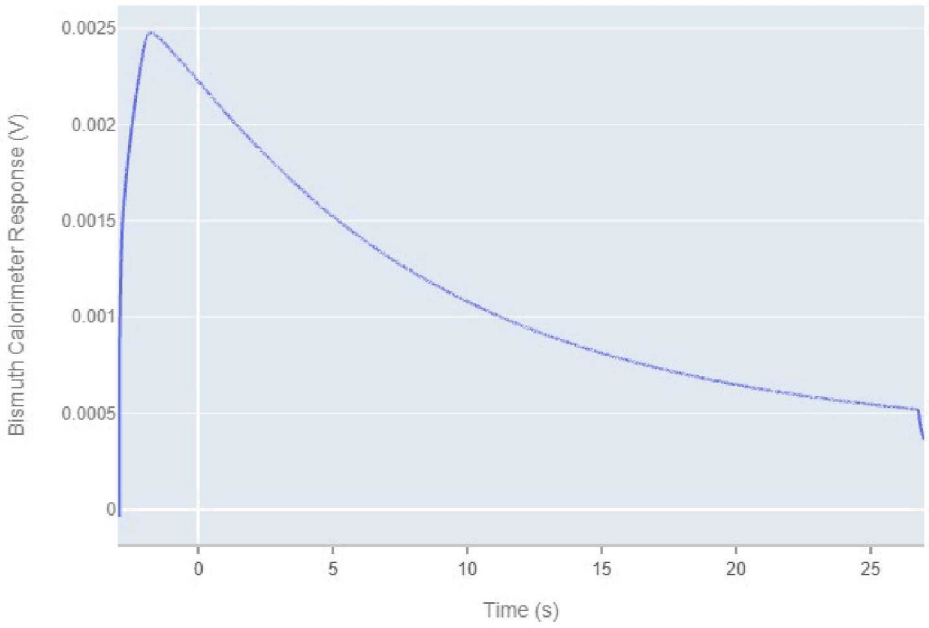


Figure 2-12. Bismuth Detector Response as a Function of Time

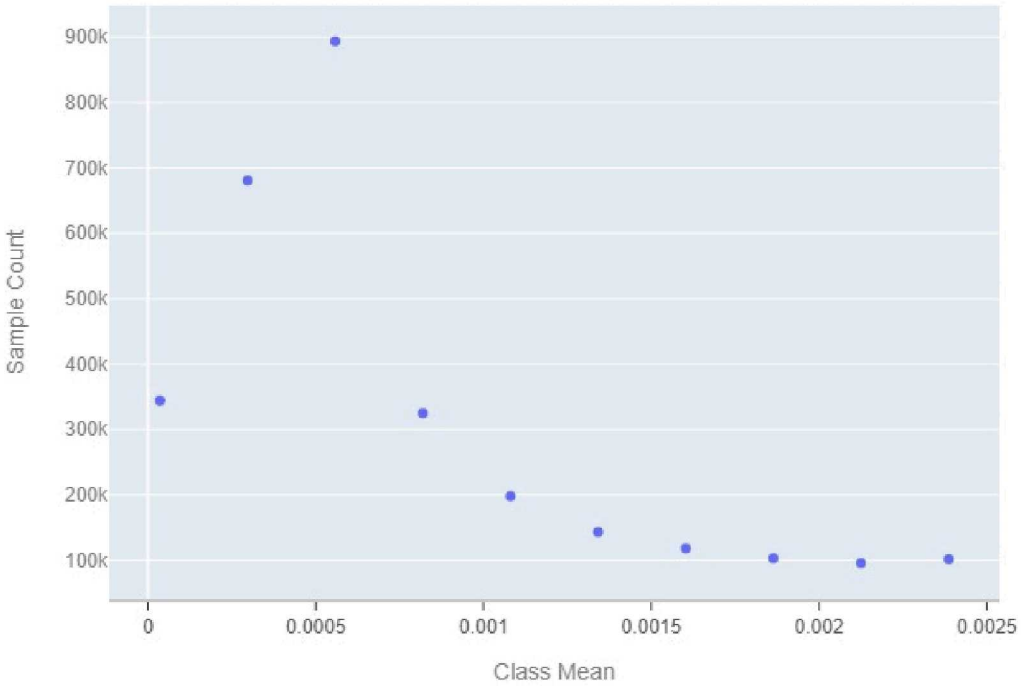


Figure 2-13 Bismuth Detector Mean Classification Histogram

The salient point from Figures 2-12 and 2-13 is that to classify a dataset and spectrum of nearly 3 million data points, the dimensionality of the data needs to be reduced to classify. This is done by taking means of small windows of data and compiling their frequencies to serve as the fingerprint for a detector. The first attempts at creating an SVM based classifier on ACRR Pulse Data is given below on Figure 2-14.

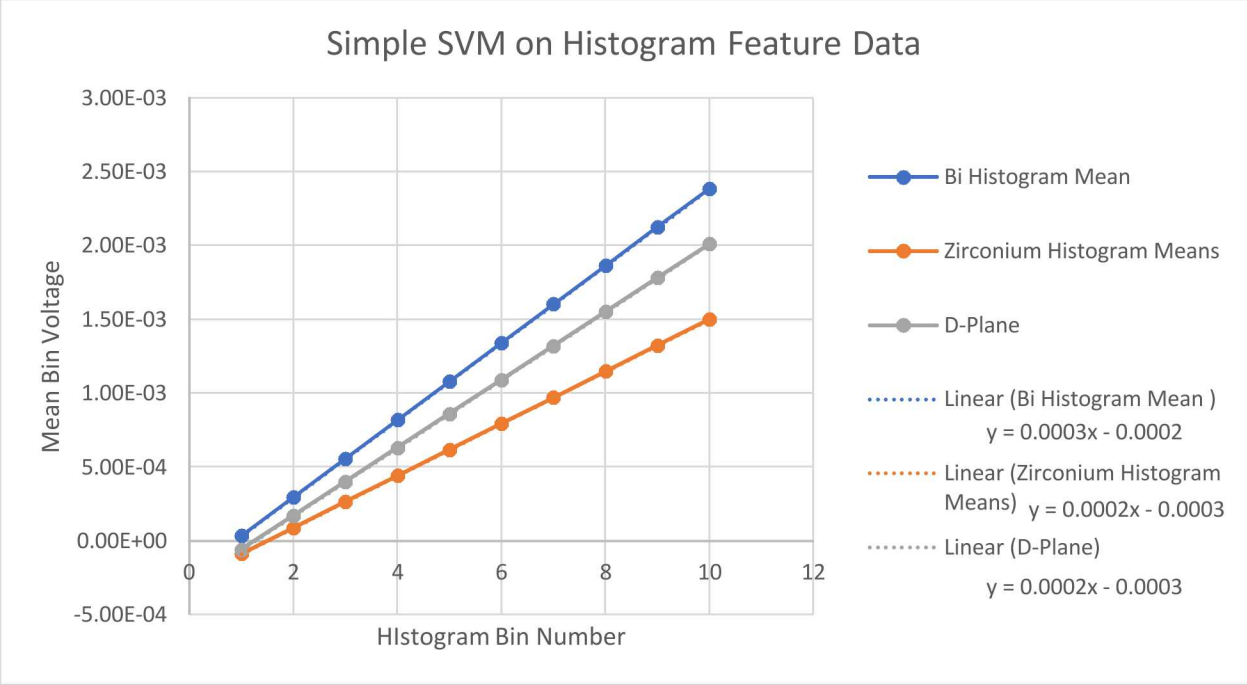


Figure 2-14 SVM Classification of ACRR: Bi (Blue) and Zr (Orange) Detector

This very simplified method of determining a hyperplane was specifically chosen for its quick diagnostic approach for experimenters needing an immediate result.

Figure 2-14 is a visual diagram of a simplistic machine learning algorithm. The grey line, denoted as D or Decision Line, is created through generation of a support vector, which through iteration, determines this line of demarcation based on the data points classified by data set 1 and 2 (blue and orange lines respectively). Simply put, it is the classification line where if a new experiment data point were to be placed above the grey line, it would be classified as part as part of blue group data. It can make this assumption after observing where the data points in the blue and orange groups fell.

How does each data point on a data spectrum get dimensionally reduced to a series of features? Or alternatively, how does an entire spectrum get represented by 10 different features? In the case of this report, this is done through generating histograms of mean values of data in a set elapsed time. This ideally makes the data comparable between different experimental conditions, thus making this useful for determination of experimental classifiers. In the y-axis above, these are mean averages for every 100 data points with a N=10 histogram bins set. Each of these parameters can be tuned or increased as necessary based on the requirements needed for the experiment (Bishop, 2006).

This is best streamlined in a Python environment but can be done relatively simply in a Microsoft Excel Spreadsheet using good prior data in terms of what to classify features based on. For most data sets, Diadem can offer a rolling mean characterization on 10 different cluster groups in the ACRR data based off of mean value. These bins then offer a very flat line as seen above that can be quickly optimized for a differentiation between Bi and Zr detection data.

It is currently unknown if this would work once larger datasets are given, but at that point, much more sophisticated methods could be used as a point of entry for feature/shape detection.

This presents a great improvement for data analytics at ACRR as it offers a new way to compare detector responses over different experimental data sets. Because this method looks at overall averages, it could be used to determine if a detector is responding as a given type, or better yet, consider with how much confidence its spectrum falls into what previous testing data shows. The addition of more and more testing data allows for more iteration and trial on the classification line (support vector) and thus would improve the fidelity and confidence in stating a classification for a detector response (Mountrakis, Im, & Ogeole, 2010). Future work in this arena could even lead to statistical testing against other discriminators to ensure quality of data and dose that extend beyond subtraction of delayed gammas and truly take the entire experiment into consideration.

3. Concluding Statements

In conclusion, this code offers great benefits and a toolbox of resources for interrogation of voltage data. The goal of this project was to be able to generate a smoothed characteristic curve for each detector for nearly 12 million different data points across 10+ channels. These analyses can easily become overwhelming, so this offers a one-stop shop to get these analyses done. This pipeline offers:

- Data Compression- Millions down to thousands of data
- Logarithmic Transforms-Made specifically for our log-amps
- Delayed Gamma Subtraction- Determine Prompt Dose for Customers
- Power Conversions- Allow for further data analyses
- Novel Peak Detection Methods and Data Compression
    - Using derivatives and expectation values to determine time envelopes
- Data Filtration- Taking very noisy data to something manageable
    - Performing regressions and other transforms
- Begin to build a pipeline for machine learning determination through SVM bootstrap

For future work, what needs to be applied is a central compendium of all pulse data to apply a rigorous machine learning mechanic. There is a weakness that can only be addressed by exposing the SVM algorithm to many different features from ACRR pulse data. The higher fidelity the algorithm, the quicker and more reliable it becomes to give an experimenter a definitive classification of their data, which ultimately will greatly cut down on the post-processing work for the customer. This pipeline also works well with previous work (Saucier, Parma 2016) in determining features between Transient Rod time points in the ACRR (Appendix B.4).

Going forward, a suggestion to the user/reader that Diadem is great at reducing and handling larger data sets, but outside of that is incredibly limited. Ad-hoc SVM training was done outside of Diadem simply cause the methods are too archaic or very limited. Python and other languages offer much more in terms of useful machine learning libraries as well as other user developed tools in the community.

4. REFERENCES

Bell, G., & Glasstone, I. (1970). *Nuclear Reactor Theory.* Washington DC: US Atomic Energy Commission.

Bishop, C. (2006). *Pattern Recognition and Machine Learning.* Springer.

Brown, F. (2018). *MCNP6 Users Manual.* Los Alamos : Los Alamos National Laboratory.

Duderstadt, J., & Hamilton, L. (1976). *Nuclear Reactor Analysis* . Ann Arbor: Wiley.

Grassberger, P., Hegger, R., Kantz, H., Schaffrath, C., & Schreiber, T. (1993). On noise reduction methods for chaotic data. *Chaos: An Interdiscplinary Journal of Nonlinear Science*, 127.

Lane, T., & Parma, E. (2015). *Delayed Fission Gamma-ray Characteristics for 232Th, 233U, 235U, 238U, and 239Pu.* Albuquerque: Sandia National Laboratories.

Mountrakis, G., Im, J., & Ogeole, C. (2010). Support vector machines in remote sensing: A Review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 247-259.

Parma, E. J. (2009). *Photoneutron Effects on Pulse Reactor Kinetics for the Annular Core Research Reactor (ACRR).* Albuquerque: Sandia National Labs.

APPENDIX A. DIADEM SOURCE CODE

```
'-------------------------------------------------------------------------------
'-- VBS script file Revision 2--Yokogawa Data Being Used
'-- Author: DAVID HAMILTON SAUCIER, SANDIA NATIONAL LABS
'-- Comment: The author used advice from ANDREW TONIGAN on how to streamline this
scripting process
'---------- This code takes a Yokogawa input file and then proceeds to take a
baseline to smooth out the data
'--Version Date: 11/10/2019, REQUIRED: WDF data plugin (avaliable from NI)
' Detectors Used: All 12
'-------------------------------------------------------------------------------

Option Explicit  'Forces the explicit declaration of all the variables in a script.



Data.Root.ChannelGroups.RemoveAll

'Initiate Mapping of Network Drive-Might not be necessary.

''''Create bool to see if drive has been mapped--doing twice results in an error
'''Uncomment if you have a distant network drive (like \\snl\home)
'if snlNetDrive=
Dim snlNetDrive
Set snlNetDrive = CreateObject("WScript.Network")
'Call snlNetDrive.MapNetworkDrive("Z:","\\snl\home\dhsauci")
' NEW PATH TO TRY \\snl\home\dhsauci\MyDocuments\Active Dosimetry\Feb-2016
shots\11735-ff-43MJ-2kw-2ndshot-2-10-16 NOTE MAKE COPY TO ENSURE PRESERVATION OF DATA
'Old Path: "Z:\MyDocuments\Active Dosimetry\Feb-2016 shots\11734-ff-26MJ-2kw-1stshot-
2-10-16\RODD0004-fast.WDF"

'Initiate Variables
Set
globaldim("SPND,AA67ref,AA03log,S93308,Bi61412,Sn61212,Zr61312,SPNDacrr,S93308air,Bi61
412air,Sn61212air,Zr61312air,TRA1,TRB1,TRC1',TRIG") =
datafileload("Z:\MyDocuments\Active Dosimetry\Feb-2016 shots\11739-ff-max-2kw-3rdshot-
24hr-2-11-16\RODD0009-fast.WDF","Yokogawa_WDF","Load")

'Dim I as int 'loop iteration index
'Call SudDlgShow("TESTING INPUT", "C:\Users\dhsauci\Documents\Active Dosimetry\Feb-
2016 shots\11734-ff-26MJ-2kw-1stshot-2-10-16\moBetta.tdm")
dim pkChn1



'BEGIN DATA FILE LOADING:
'  Check on whether this path can be queried through a dialogue box or softcoded
Call datafileload("Z:\MyDocuments\Active Dosimetry\Feb-2016 shots\11734-ff-26MJ-2kw-
1stshot-2-10-16\RODD0004-fast.WDF","Yokogawa_WDF","Load")


''LOAD THE DELAYED GAMMA DATA



'BEGIN BASELINE CALCULATIONS
Dim I
Dim Z
Z = ChnLength("SPND")

'Clean up the channels.
'For i= 1 to 23
```

```
' StatSel(I)= "No"

'Next
'ArithMean'

'StatSel(6)= "Yes"

'If CNo("SPND")> 0 Then
'ChnDel("SPND")
'End If


'''''' BEGIN CLEANING ROUTINE FROM CLEANED CHANNELS ABOVE"""""""""""""

''                                                                   "

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'''
'''

'Begin Channel Filtering


Call Data.Root.ChannelGroups.Add("FilteredChannels")
Call
Data.Root.ChannelGroups("FilteredChannels").Channels.Add("SPNDclean",DataTypeFloat64)
Call
Data.Root.ChannelGroups("FilteredChannels").Channels.Add("siClean",DataTypeFloat64)
Call
Data.Root.ChannelGroups("FilteredChannels").Channels.Add("siTime",DataTypeFloat64)

Call
Data.Root.ChannelGroups("FilteredChannels").Channels.Add("AA67refClean",DataTypeFloat6
4)
Call
Data.Root.ChannelGroups("FilteredChannels").Channels.Add("AA03logClean",DataTypeFloat6
4)
Call
Data.Root.ChannelGroups("FilteredChannels").Channels.Add("Bi61412Clean",DataTypeFloat6
4)
Call
Data.Root.ChannelGroups("FilteredChannels").Channels.Add("Sn61212Clean",DataTypeFloat6
4)
Call
Data.Root.ChannelGroups("FilteredChannels").Channels.Add("Zr61312Clean",DataTypeFloat6
4)

Call
Data.Root.ChannelGroups("FilteredChannels").Channels.Add("SPNDacrrClean",DataTypeFloat
64)
Call
Data.Root.ChannelGroups("FilteredChannels").Channels.Add("S93308airClean",DataTypeFloa
t64)
Call
Data.Root.ChannelGroups("FilteredChannels").Channels.Add("Bi61412airClean",DataTypeFlo
at64)
Call
Data.Root.ChannelGroups("FilteredChannels").Channels.Add("Sn61212airClean",DataTypeFlo
at64)
```

```
Call
Data.Root.ChannelGroups("FilteredChannels").Channels.Add("Zr61312airClean",DataTypeFlo
at64)

'Begin Channel Filtering

Call ChnFiltCalc("","[1]/S93308","[3]/siClean","IIR","Bessel","Low
pass",1,60,0,0,1.2,25,"Rectangle",1,1)
Call ChnFiltCalc("","[1]/SPND","[3]/SPNDclean","IIR","Bessel","Low
pass",1,60,0,0,1.2,25,"Rectangle",1,1)
Call ChnFiltCalc("","[1]/AA67ref","[3]/AA67refClean","IIR","Bessel","Low
pass",1,60,0,0,1.2,25,"Rectangle",1,1)
Call ChnFiltCalc("","[1]/AA03log","[3]/AA03logClean","IIR","Bessel","Low
pass",1,60,0,0,1.2,25,"Rectangle",1,1)
Call ChnFiltCalc("","[1]/Bi61412","[3]/Bi61412clean","IIR","Bessel","Low
pass",1,60,0,0,1.2,25,"Rectangle",1,1)
Call ChnFiltCalc("","[1]/Sn61212","[3]/Sn61212clean","IIR","Bessel","Low
pass",1,60,0,0,1.2,25,"Rectangle",1,1)
Call ChnFiltCalc("","[1]/Zr61312","[3]/Zr61312clean","IIR","Bessel","Low
pass",1,60,0,0,1.2,25,"Rectangle",1,1)
Call ChnFiltCalc("","[1]/SPND","[3]/SPNDacrrClean","IIR","Bessel","Low
pass",1,60,0,0,1.2,25,"Rectangle",1,1)  'add acrr if labeled such
Call ChnFiltCalc("","[1]/Bi61412air","[3]/Bi61412airclean","IIR","Bessel","Low
pass",1,60,0,0,1.2,25,"Rectangle",1,1)
Call ChnFiltCalc("","[1]/Sn61212air","[3]/Sn61212airclean","IIR","Bessel","Low
pass",1,60,0,0,1.2,25,"Rectangle",1,1)
Call ChnFiltCalc("","[1]/Zr61312air","[3]/Zr61312airclean","IIR","Bessel","Low
pass",1,60,0,0,1.2,25,"Rectangle",1,1)
Call ChnFiltCalc("","[1]/S93308air","[3]/S93308AirClean","IIR","Bessel","Low
pass",1,60,0,0,1.2,25,"Rectangle",1,1)


'Begin making cleaned channels
dim cleanArray(),cleanChannels(0)
Call Data.Root.ChannelGroups.Add("CleanedChannels")
Call
Data.Root.ChannelGroups("CleanedChannels").Channels.Add("SPNDclean",DataTypeFloat64)
Call
Data.Root.ChannelGroups("CleanedChannels").Channels.Add("siClean",DataTypeFloat64)
Call Data.Root.ChannelGroups("CleanedChannels").Channels.Add("siTime",DataTypeFloat64)

Call
Data.Root.ChannelGroups("CleanedChannels").Channels.Add("AA67refClean",DataTypeFloat64
)
Call
Data.Root.ChannelGroups("CleanedChannels").Channels.Add("AA03logClean",DataTypeFloat64
)
Call
Data.Root.ChannelGroups("CleanedChannels").Channels.Add("Bi61412Clean",DataTypeFloat64
)
Call
Data.Root.ChannelGroups("CleanedChannels").Channels.Add("Sn61212Clean",DataTypeFloat64
)
Call
Data.Root.ChannelGroups("CleanedChannels").Channels.Add("Zr61312Clean",DataTypeFloat64
)

Call
Data.Root.ChannelGroups("CleanedChannels").Channels.Add("SPNDacrrClean",DataTypeFloat6
4)
Call
Data.Root.ChannelGroups("CleanedChannels").Channels.Add("S93308airClean",DataTypeFloat
64)
```

```
Call
Data.Root.ChannelGroups("CleanedChannels").Channels.Add("Bi61412airClean",DataTypeFloa
t64)
Call
Data.Root.ChannelGroups("CleanedChannels").Channels.Add("Sn61212airClean",DataTypeFloa
t64)
Call
Data.Root.ChannelGroups("CleanedChannels").Channels.Add("Zr61312airClean",DataTypeFloa
t64)

'Time Channels

Call Data.Root.ChannelGroups.Add("TimeChannels")
Call Data.Root.ChannelGroups("TimeChannels").Channels.Add("SPNDclean",DataTypeFloat64)
Call Data.Root.ChannelGroups("TimeChannels").Channels.Add("siClean",DataTypeFloat64)
Call Data.Root.ChannelGroups("TimeChannels").Channels.Add("siTime",DataTypeFloat64)

Call
Data.Root.ChannelGroups("TimeChannels").Channels.Add("AA67refClean",DataTypeFloat64)
Call
Data.Root.ChannelGroups("TimeChannels").Channels.Add("AA03logClean",DataTypeFloat64)
Call
Data.Root.ChannelGroups("TimeChannels").Channels.Add("Bi61412Clean",DataTypeFloat64)
Call
Data.Root.ChannelGroups("TimeChannels").Channels.Add("Sn61212Clean",DataTypeFloat64)
Call
Data.Root.ChannelGroups("TimeChannels").Channels.Add("Zr61312Clean",DataTypeFloat64)

Call
Data.Root.ChannelGroups("TimeChannels").Channels.Add("SPNDacrrClean",DataTypeFloat64)
Call
Data.Root.ChannelGroups("TimeChannels").Channels.Add("S93308airClean",DataTypeFloat64)
Call
Data.Root.ChannelGroups("TimeChannels").Channels.Add("Bi61412airClean",DataTypeFloat64
)
Call
Data.Root.ChannelGroups("TimeChannels").Channels.Add("Sn61212airClean",DataTypeFloat64
)
Call
Data.Root.ChannelGroups("TimeChannels").Channels.Add("Zr61312airClean",DataTypeFloat64
)



Call Data.Root.ChannelGroups("CleanedChannels").Channels.Add("t1",DataTypeFloat64)
Call Data.Root.ChannelGroups("CleanedChannels").Channels.Add("t2",DataTypeFloat64)
Call Data.Root.ChannelGroups("CleanedChannels").Channels.Add("t3",DataTypeFloat64)


Call Data.Root.ChannelGroups.Add("TransformedChannels")
Call
Data.Root.ChannelGroups("TransformedChannels").Channels.Add("SP1Log",DataTypeFloat64)
Call
Data.Root.ChannelGroups("TransformedChannels").Channels.Add("xAxis",DataTypeFloat64)
'' Generate Time Channel
Call ChnFromWFxGen("[1]/S93308","[1]/siTime","WfXRelative")

dim SPNDclean, siClean
dim j
j=1

'Query User to Enter Step Size
dim N
```

```
N = InputBox("Enter How Many Data Points to Skip","SaucierSmooth.vbs User Query",100)

dim t
t=1
dim maxt1
Call ChnPeakDetect("[1]/t","[1]/TR-A1","maxaP1","maxaA1","PEAKS",1,3)
Call ChnPeakDetect("[1]/t","[1]/TR-B1","maxbP1","maxbA1","PEAKS",1,3)
Call ChnPeakDetect("[1]/t","[1]/TR-C1","maxcP1","maxcA1","PEAKS",1,3)
dim c1, c2, c3
c1 =1
c2 =1
c3 =1
dim bool

'''''' BEGIN TRANSIENT ROD PROCESSING''''''''''''''''''''''''''''''''''''''''

''                                                                    "

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'For t=1 to Z

'b= Lg(Ch([1]/TR-B1))
'c= Lg(Ch([1]/TR-C1))

'  If Data.GetChannel("[1]/TR-A1").Values(t) >=
0.2*Data.GetChannel("[2]/maxaA1").Maximum     then
'     Data.GetChannel("[3]/t1").Values(t) = 0.0002 '(Chd(t,"[1]/TR-A1"))/40000
'     if c1 < 7 then
'      Data.GetChannel("[3]/t1").Values(t+c1) = 0.000 '(Chd(t,"[1]/TR-A1"))/40000

'      c1= c1+1

'      if c1 = 6 then c1 = 1 and t=t+6
'     End if

'    Else Data.GetChannel("[3]/t1").Values(t) =  0 '(Chd(t,"[1]/TR-A1"))/80000
'  End if


'  If Data.GetChannel("[1]/TR-B1").Values(t) >=
0.2*Data.GetChannel("[2]/maxbA1").Maximum then
'     Data.GetChannel("[3]/t2").Values(t) = 0.0002 '(Chd(t,"[1]/TR-B1"))/40000
'       if c2 < 7 then
'      Data.GetChannel("[3]/t2").Values(t+c2) = 0.000 '(Chd(t,"[1]/TR-A1"))/40000

'      c2= c2+1

'      if c2 = 6 then c2 = 1 and t=t+6
'     End if

'    else Data.GetChannel("[3]/t2").Values(t) =  0 '(Chd(t,"[1]/TR-B1"))/80000
'  End If


'  If Data.GetChannel("[1]/TR-C1").Values(t) >=
0.2*Data.GetChannel("[2]/maxcA1").Maximum  then
'     Data.GetChannel("[3]/t3").Values(t) = 0.0002 ' (Chd(t,"[1]/TR-C1"))/40000
'     if c3 < 7 then
'      Data.GetChannel("[3]/t3").Values(t+c3) = 0.000 '(Chd(t,"[1]/TR-A1"))/40000

'      c3= c3+1
```

```
'      if c3 = 6 then c3 = 1 and t=t+6
'     End if
'    Else Data.GetChannel("[3]/t3").Values(t) =  0 '(Chd(t,"[1]/TR-C1"))/80000
'  End if




'Next



''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'''''' BEGIN SMOOTHING ROUTINE FROM CLEANED CHANNELS ABOVE""""""""""""""

''                                                                          "

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'Initiate Statistical Calculations for each channel
''''Check the method-- the average smoothing function is acting oddly(dropping values)
'Call Statblockcalc("Channel","0-"& Z,"[4]/SPNDclean")
'Call Statblockcalc("Channel","0-"& Z,"[4]/siClean")

'Begin calculating the aritmetic means across the channel
'Call FormulaCalc("Ch('CleanedChannels/SPNDclnAvg') :=" & "Ch('[4]/SPNDclean')" & "-
StatArithMean")
'Call FormulaCalc("Ch('CleanedChannels/siClnAvg') :=" & "Ch('[4]/siClean')" & "-
StatArithMean")


'Begin Smoothing Routines
'' Note the generate average channels for each yield the same shape but at a lower
magnetude.

'''Normal Smoothing Block, no averages
Call
ChnSmooth("FilteredChannels/SPNDclean","FilteredChannels/SPNDsmooth",Data.Root.Propert
ies.Count,"maxNumber")
Call
ChnSmooth("FilteredChannels/siClean","FilteredChannels/siSmoothedr2",Data.Root.Propert
ies.Count,"maxNumber")

Call
ChnSmooth("FilteredChannels/AA67refClean","FilteredChannels/AA67refSmooth",Data.Root.P
roperties.Count,"maxNumber")
Call
ChnSmooth("FilteredChannels/AA03logClean","FilteredChannels/AA03logClean",Data.Root.Pr
operties.Count,"maxNumber")
Call
ChnSmooth("FilteredChannels/Bi61412Clean","FilteredChannels/Bi61412Smooth",Data.Root.P
roperties.Count,"maxNumber")
Call
ChnSmooth("FilteredChannels/Sn61212Clean","FilteredChannels/Sn61212Smooth",Data.Root.P
roperties.Count,"maxNumber")
```

```
Call
ChnSmooth("FilteredChannels/Zr61312Clean","FilteredChannels/Zr61312Smooth",Data.Root.P
roperties.Count,"maxNumber")

Call
ChnSmooth("FilteredChannels/SPNDacrrClean","FilteredChannels/SPNDacrrSmooth",Data.Root
.Properties.Count,"maxNumber")
Call
ChnSmooth("FilteredChannels/S93308airClean","FilteredChannels/S93308airSmooth",Data.Ro
ot.Properties.Count,"maxNumber")
Call
ChnSmooth("FilteredChannels/Bi61412airClean","FilteredChannels/Bi61412airSmooth",Data.
Root.Properties.Count,"maxNumber")
Call
ChnSmooth("FilteredChannels/Sn61212airClean","FilteredChannels/Sn61212airSmooth",Data.
Root.Properties.Count,"maxNumber")
Call
ChnSmooth("FilteredChannels/Zr61312airClean","FilteredChannels/Zr61312airSmooth",Data.
Root.Properties.Count,"maxNumber")


'''Average Smoothing Block, statistical averages used and must be calculated in
earlier block
'Call
ChnSmooth("CleanedChannels/SPNDclnAvg","CleanedChannels/SPNDsmoothed",Data.Root.Proper
ties.Count,"maxNumber")
'Call
ChnSmooth("CleanedChannels/siClnAvg","CleanedChannels/siSmoothed",Data.Root.Properties
.Count,"maxNumber")



'''''' BEGIN CHANNEL PEAK SEARCH AND BRANCHING FOR LOOPS'"""""""""""""""

''                                                                      "
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''


'Begin Searching For Peaks

'Call ChnPeakDetect("[1]/t","[1]/SPND","maxpSPND","maxaSPND","PEAKS",0.01,3)
'Call ChnPeakDetect("[1]/t","[1]/S93308","maxpSI","maxaSI","PEAKS",0.01,3)
'Call ChnPeakDetect("[1]/t","[1]/TR-C1","maxcP1","maxcA1","PEAKS",1,3)
Call
ChnPeakFind("","[1]/SPND","[3]/maxSPNDx","[3]/maxSPNDa",1,"Max.Peaks","Amplitude")
  'BEGIN new subroutine
dim spndInd
spndInd = PnO("[1]/SPND",CMax("[1]/SPND"))

dim aaRefInd
aaRefInd = PnO("[1]/AA67ref",CMAX("[1]/AA67ref"))







Call
ChnPeakFind("","[1]/AA67ref","[3]/maxAA67refX","[3]/maxAA67refA",1,"Max.Peaks","Amplit
ude")
```

```
''TYPE 2 PEAKS
'Note: for most type 2 peaks, channel peaks are relatively useless. Differentiate and
find where the delta is greatest.
Call ChnDifferentiate("","[3]/siClean","[3]/siDiffX","[3]/siDiffY")
Call ChnPeakFind("","[3]/siDiffY","[3]/maxSiX","[3]/maxSiA",1,"Max.Peaks","Amplitude")
dim siInd
siInd= PnO("[3]/siDiffY",CMAX("[3]/siDiffY"))


call ChnDifferentiate("","[3]/Bi61412Clean","[3]/Bi61412diffX","[3]/Bi61412diffY")
Call
ChnPeakFind("","[3]/Bi61412diffY","[3]/maxBi61412X","[3]/maxBi61412A",1,"Max.Peaks","A
mplitude")
dim biInd
biInd= PnO("[3]/Bi61412diffY",CMAX("[3]/Bi61412diffY"))

call ChnDifferentiate("","[3]/Sn61212Clean","[3]/Sn61212diffX","[3]/Sn61212diffY")
Call
ChnPeakFind("","[3]/Sn61212diffY","[3]/maxSn61212X","[3]/maxSn61212A",1,"Max.Peaks","A
mplitude")
dim snInd
snInd = PnO("[3]/Sn61212diffY",CMax("[3]/Sn61212diffY"))

call ChnDifferentiate("","[3]/Zr61312Clean","[3]/Zr61312diffX","[3]/Zr61312diffY")

dim zrInd
zrInd = PnO("[3]/Zr61312diffY",CMax("[3]/Zr61312diffY"))

call
ChnDifferentiate("","[3]/S93308airClean","[3]/S93308airdiffX","[3]/S93308airdiffY")

dim siAirInd
siAirInd = PnO("[3]/S93308airdiffY",CMax("[3]/S93308airdiffY"))

call
ChnDifferentiate("","[3]/Bi61412airClean","[3]/Bi61412airdiffX","[3]/Bi61412airdiffY")

dim biAirInd
biAirInd = PnO("[3]/Bi61412airdiffY",CMax("[3]/Bi61412airdiffY"))




'Peak width of types 1 and 2
dim dt1
dt1= 500000  '5 seconds
dim dt2
dt2 = 1000000 '10 seconds

i=1

''''Begin new algorithm Technique

dim spndChanNo, timeChanNo, AA67refChanNo
spndChanNo = CNo("[3]/SPNDsmooth")
timeChanNo = Cno("[1]/t")
AA67refChanNo= CNo("[3]/AA67refSmooth")

Call ChnAlloc("[4]/SPNDclean",Z)

Do While i< Z
 ' Redim Preserve cleanArray(0,j)
```

```
  Data.GetChannel("[5]/SPNDclean").Values(j) = chD(i, "[1]/t")


  '''''''''''''''''''
  'Type I Peaks
  if i < spndInd - dt1 or i > spndInd + dt1 Then
      Data.GetChannel("[4]/SPNDclean").Values(j) = chdx(i,spndChanNo)
      i=i+N

  else     Data.GetChannel("[4]/SPNDclean").Values(j) = chdx(i,spndChanNo)

  end If

  j= j+1
i=i+1

Loop
i= 1
j=1

DO while i<z

 Data.GetChannel("[5]/AA67refclean").Values(j) = chD(i, "[1]/t")


   if i < aaRefInd - dt1 or i > aaRefInd+ dt1 Then
      Data.GetChannel("[4]/AA67refClean").Values(j) = ChdX(i, AA67refChanNo)
      i=i+N
     else Data.GetChannel("[4]/AA67refClean").Values(j) = ChDX(i,AA67refChanNo)

    end IF
  j= j+1
  i=i+1

Loop
i=1
j=1


   'SPND ACRR has a different type of peak structure
    'Data.GetChannel("[4]/SPNDacrrClean").Values(j) = Chd(i, "[3]/SPNDacrrClean")


Do while i<z
   'Type 2 Peaks
   Data.GetChannel("[5]/siClean").Values(j) = chD(i, "[1]/t")

  if i < siInd or i > siInd +dt2 then
  Data.GetChannel("[4]/siClean").Values(j) = Chd(i, "[3]/siClean")
  i=i+N

  else  Data.GetChannel("[4]/siClean").Values(j) = Chd(i, "[3]/siClean")
  'Call MsgBox("IF Tripped")
  end if

j= j+1
i=i+1

Loop
i=1
j=1
```

```
Do while i<z

 Data.GetChannel("[5]/Bi61412clean").Values(j) = chD(i, "[1]/t")
  'ADD THESE PEAKS TO TYPE 2 IF ROUTINE
    'if i < biInd -dt2 or i > biInd +dt2 then
     if i <biInd or i > biInd + 1500000 then
    Data.GetChannel("[4]/Bi61412Clean").Values(j) = Chd(i, "[3]/Bi61412Smooth")
    i=i+N
  else  Data.GetChannel("[4]/Bi61412Clean").Values(j) = Chd(i, "[3]/Bi61412Smooth")

  end if



j= j+1
i=i+1

Loop
  i=1
  j=1
  Do while i<z

 Data.GetChannel("[5]/Sn61212clean").Values(j) = chD(i, "[1]/t")
  'ADD THESE PEAKS TO TYPE 2 IF ROUTINE
    'if i < biInd -dt2 or i > biInd +dt2 then
     if i <snInd or i > snInd + 1500000 then
    Data.GetChannel("[4]/Sn61212Clean").Values(j) = Chd(i, "[3]/Sn61212Clean")
    i=i+N
  else  Data.GetChannel("[4]/Sn61212Clean").Values(j) = Chd(i, "[3]/Sn61212Clean")

  end if
  j= j+1
i=i+1

Loop

 i=1
  j=1

 Do while i<z

   Data.GetChannel("[5]/Zr61312clean").Values(j) = chD(i, "[1]/t")

     if i <zrInd or i > zrInd + 1500000 then
   Data.GetChannel("[4]/Zr61312clean").Values(j) = Chd(i, "[3]/Zr61312Clean")
    i=i+N
  else  Data.GetChannel("[4]/Zr61312clean").Values(j) = Chd(i, "[3]/Zr61312Clean")

  end if

j= j+1
i=i+1

Loop
  i=1
  j=1


  '''''''''''''''''''
```

```
'''''' BEGIN LOG TRANSFORM ROUTINE''''''''''''''''''''''''''''""""""""""""""

''  Taken From Billy Martin, SP1 data- All should be similar          "

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Call dataFileLoad("C:\Users\dhsauci\Documents\LogAmp_062016DSrevs.csv","CSV","Load")
'Call
ChnMapLinCalc("[7]/IIN","[7]/VOUT","[1]/AA03log","[6]/SP1Log",True,"f[bound.slope]",No
Value,"analogue")
Call datafileload("Z:\MyDocuments\delayedGammaData.xlsx","Excel Example","Load")
  'LOG Type Peak

  'Data.GetChannel("[4]/AA03logClean").Values(j) = Chd(i, "[3]/AA03logClean")









 '  cleanArray(0,j) = ChV(i,"Date Time_1")

   'cleanArray(1,j) = "1" ' RTT(cleanArray(1,j),"#mm/dd/yyyy hh:nn:ss.ff")


'Loop
'FIX THIS CHANNEL GET ERROR
'dim chnCopyList
'set chnCopyList = Data.GetChannels("[4]/SPNDclean" & "[4]/siTime")

 'ChnCopy("[4]/SPNDclean","[5]/spndClean")
 'ChnCopy("[4]/siTime","[5]/Time")


'Call DataFileSaveSel("C:\Users\dhsauci\Documents\dataForTommy\dataOut"
,"CSV","'[4]/siTime','[4]/SPNDclean'")


'''''' BEGIN OPTIONAL CODING BLOCKS """""""""""""""""""""""""""""""""""""

''                                                                    "

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'For calorimeter data--- Convert to millivolts to use for calculations

'Call MatScalMul("CalSmoothed",1e3,True)



''''''''THIS SUBROUTINE ADDS THE ANALYSIS OF AIR DETECTORS''''''''''''''''''''''



'Do while i<z
'     Data.GetChannel("[5]/S93308airClean").Values(j) = chD(i, "[1]/t")
```

```
'     if i <siInd or i > siInd + 1500000 then
'    Data.GetChannel("[4]/S93308airClean").Values(j) = Chd(i, "[3]/S93308airClean")
'     i=i+N
'  else  Data.GetChannel("[4]/S93308airClean").Values(j) = Chd(i,
"[3]/S93308airClean")

'  end if
'  j= j+1
'  i=i+1

'Loop
'  i=1
'  j=1

'Do while i<z
'    Data.GetChannel("[5]/Bi61412airClean").Values(j) = chD(i, "[1]/t")

'     if i <biAirInd or i > biAirInd + 1500000 then
'    Data.GetChannel("[4]/Bi61412airClean").Values(j) = Chd(i, "[3]/Bi61412airClean")
'     i=i+N
'  else  Data.GetChannel("[4]/Bi61412airClean").Values(j) = Chd(i,
"[3]/Bi61412airClean")

'  end if
' j= j+1
'  i=i+1
'Loop
'  i=1
'  j=1

'Do while i<z
'    Data.GetChannel("[5]/Sn61212airClean").Values(j) = chD(i, "[1]/t")

'     if i <snAirInd or i > snAirInd + 1500000 then
'    Data.GetChannel("[4]/Sn61212airClean").Values(j) = Chd(i, "[3]/Sn61212airClean")
'     i=i+N
'  else Data.GetChannel("[4]/Sn61212airClean").Values(j) = Chd(i,
"[3]/Sn61212airClean")

'  end if
 ' j= j+1
 ' i=i+1
'Loop
'  i=1
'  j=1

'Do while i<z
'    Data.GetChannel("[5]/Zr61312airClean").Values(j) = chD(i, "[1]/t")

'     if i <zrAirInd or i > zrAirInd + 1500000 then
'     Data.GetChannel("[4]/Zr61312airClean").Values(j) = Chd(i, "[3]/Zr61312airClean")
'     i=i+N
'  else   Data.GetChannel("[4]/Zr61312airClean").Values(j) = Chd(i,
"[3]/Zr61312airClean")

  'end if

 ' j= j+1
 ' i=i+1
```

ADDITIONAL MODULES:INTERPOLATION, CURVEFIT, SIMPLE CURVE FIT

A.1. Interpolation

```
'-------------------------------------------------------------------------------
'-- VBS script file Revision 1-- Interpolation Method/ Transform for logarithm based
channels.
'-- Created on 08/19/16
'-- Author: DAVID HAMILTON SAUCIER, SANDIA NATIONAL LABS
'-- Comment:
'--Version Date: 10/23/2018, REQUIRED: WDF data plugin (avaliable from NI) for
grabbing data, and loading CSV parsed data
' Detectors Used: Just looking at the AA03ref Log Detector
'-------------------------------------------------------------------------------
Option Explicit  'Forces the explicit declaration of all the variables in a script.


''Create Data Variables
Call chndelete("[6]/xAxis")
Call chndelete("[6]/SP1Log")
Call
Data.Root.ChannelGroups("TransformedChannels").Channels.Add("xAxis",DataTypeFloat64)
Call
Data.Root.ChannelGroups("TransformedChannels").Channels.Add("SP1Log",DataTypeFloat64)

dim i,j,slope,Y,Z, ind,N
i=1
j=1
ind= 1
slope= 1


Y = ChnLength("[7]/VOUT")

Z = ChnLength("[1]/SPND")

'Find where the peak happens in order to save time
call ChnDifferentiate("","[3]/AA03logClean","[3]/AA03logdiffX","[3]/AA03logdiffY")

dim aaLogInd
aaLogInd = PnO("[3]/AA03logdiffY",CMin("[3]/AA03logdiffY"))
Call msgbox(aaLogInd)
'Tolerance:Keeping numbers before D/dt pk
N=200

'Begin For loop interpolation method

'Outside loop--->Loop through each value in data channel
for i=aaLogInd-N to Z
ind = i
'Data.GetChannel("[6]/xAxis").Values(ind) = chD(i, "[1]/t")
'Inside Loop--->Loop through and check where to interpolate
  For j=1 to  Y 'step 1000

   if Data.GetChannel("[3]/AA03logClean").Values(i) < CHD(j,"[7]/VOUT")  and
Data.GetChannel("[3]/AA03logClean").Values(i) > Chd(j+1,"[7]/VOUT") then

     if (CHD(j+1,"[7]/VOUT")-CHD(j,"[7]/VOUT")) = 0  then

            Data.GetChannel("[6]/SP1LogInterp").Values(ind)= CHD(i-
1,"[6]/SP1LogInterp")
```

```
     else slope = (CHD(j+1,"[7]/IIN")-CHD(j,"[7]/IIN"))/(CHD(j+1,"[7]/VOUT")-
CHD(j,"[7]/VOUT"))
             Data.GetChannel("[6]/SP1LogInterp").Values(ind) =
slope*(CHD(i,"[3]/AA03logClean")-CHD(j,"[7]/VOUT"))+ CHD(j,"[7]/IIN")

        'call msgbox(slope)
     end if
    else
      Data.GetChannel("[6]/SP1LogInterp").Values(ind)= CHD(i-1,"[6]/SP1LogInterp")

     end if



  Next

Next
```

A.2. Curvefit

```
'-------------------------------------------------------------------------------
'-- VBS script file
'-- Created on 08/31/2017 17:30:11
'-- Author: David Saucier
'-- Comment:
'-------------------------------------------------------------------------------
Option Explicit  'Forces the explicit declaration of all the variables in a script.

''Create Data Variables
Call chndelete("[6]/xAxis")
Call chndelete("[6]/SP1Log")
Call
Data.Root.ChannelGroups("TransformedChannels").Channels.Add("xAxis",DataTypeFloat64)
Call
Data.Root.ChannelGroups("TransformedChannels").Channels.Add("SP1Log",DataTypeFloat64)

dim x, i, Z
Z = ChnLength("[1]/SPNDbig")
i= 1

''''''CHANGE THE BEGINNING I INDEX WHEN YOU'RE DONE POPULATING!!! THIS IS JUST TO
OBSERVE THE SHAPE AND SAVE TIME
'
' CHANGE BACK TO I=1
'

for i= 1 to Z  '1624401
'
'
'
'
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'''''''''''''''''''''''''''''''
if  Data.GetChannel("[7]/Smoothed").Values(i) < 1.85  and
Data.GetChannel("[7]/Smoothed").Values(i) => 0.0034 then
 x=Data.GetChannel("[7]/Smoothed").Values(i)

 Data.GetChannel("[6]/SP1Log").Values(i)= 10.774*exp(8.4614*x)
elseif Data.GetChannel("[7]/Smoothed").Values(i) <= 4.20  and
Data.GetChannel("[7]/Smoothed").Values(i) > 1.85 then
 x=Data.GetChannel("[7]/Smoothed").Values(i)

 Data.GetChannel("[6]/SP1Log").Values(i)= 2*x^3+9e12*x^2-2e13*x+1e13
else
 Data.GetChannel("[6]/SP1Log").Values(i)= 9999


end if



Next
```

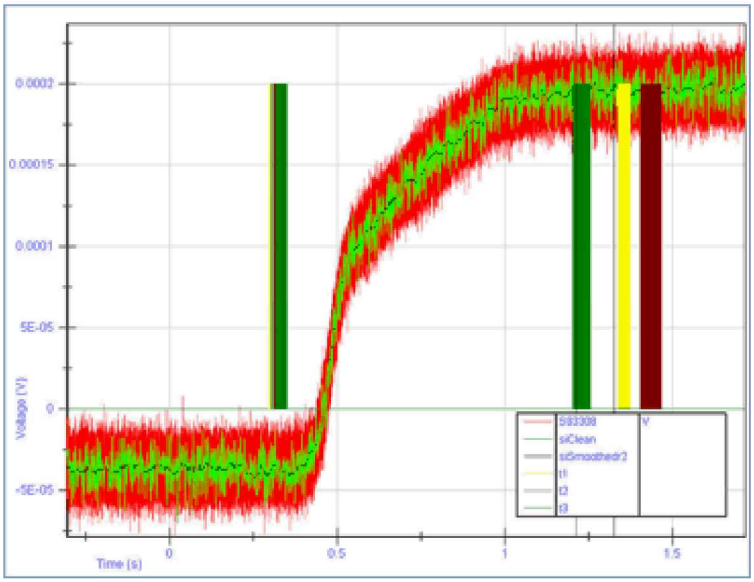
A.3. PowerConvert (voltage to MW)

```
'-- VBS script file
'-- Created on 10/06/2017 16:52:41
'-- Author: David Saucier
'-- Comment:
'--------------------------------------------------------------------------------
Option Explicit  'Forces the explicit declaration of all the variables in a script.


'NEED TO SYNC UP TIMING FOR THE LINEAR FIT, ALSO SEE IF THE FIT NEEDS TO BE PIECEWISE
CAUSE OF EARLY TIME REGION CONSTANT

' Begin Conversion to Watts
dim z
dim sum
dim j, i
z= chnlength("[1]/SPNDbig")

'Call ChnDelete("[9]/PowerChannel")
'Call ChnDelete("[9]/PowerChannel2")
'Call ChnDelete("[9]/PowerTime")
'Call ChnDelete("[9]/ConvertedPowerChannel")
'Call ChnDelete("[9]/ConvertedLogChannel")



Call Data.Root.ChannelGroups.Add("ConvertedPowerChannels")
Call
Data.Root.ChannelGroups("ConvertedPowerChannels").Channels.Add("PowerTime",DataTypeFlo
at64)
Call
Data.Root.ChannelGroups("ConvertedPowerChannels").Channels.Add("PowerTimeLog",DataType
Float64)
Call
Data.Root.ChannelGroups("ConvertedPowerChannels").Channels.Add("PowerChannel",DataType
Float64)
Call
Data.Root.ChannelGroups("ConvertedPowerChannels").Channels.Add("PowerChannel2",DataTyp
eFloat64)
Call
Data.Root.ChannelGroups("ConvertedPowerChannels").Channels.Add("PowerTime",DataTypeFlo
at64)
Call
Data.Root.ChannelGroups("ConvertedPowerChannels").Channels.Add("ConvertedPowerChannel"
,DataTypeFloat64)
Call
Data.Root.ChannelGroups("ConvertedPowerChannels").Channels.Add("ConvertedLogChannel",D
ataTypeFloat64)


Call ChnDifferentiate("","[3]/SPNDclean","[7]/spndDiffX","[7]/spndDiffY")
'Call
ChnPeakFind("","[9]/spndDiffX","[9]/spndDiffY","[3]/maxSiA",1,"Max.Peaks","Amplitude")
dim spndInd, zeroIndex, sumPulseTime, zeroPulseInd

spndInd = PnO("[3]/AA67refClean",CMax("[3]/AA67refClean"))
zeroPulseInd =  PnO("[7]/spndDiffY",CMax("[7]/spndDiffY"))
zeroIndex = spndInd - 5*(-zeroPulseInd+spndInd)
```

```
'CURRENT METHOD: POWER NORMALIZATION TO X MEGAJOULE PULSE

'Call ChnSum("[1}/SPND","[9]/SPNDintegral")
sumPulseTime= CHD(chnlength("[1]/t"),"[1]/t")-CHD(zeroPulseInd,"[1]/t")


'PULSE TIME--- Basic metric taken from DSA pg 674 15-79. Not sure if correct.
dim startIndex, pulseBool
pulseBool = FALSE

'Normalization Terms
dim sumPower, sumBG, nBG, avgBG, sumPowerLog, sumBGLog, nBGLog, avgBGLog
nBG=1
'Remove Background and calculate sum of power in peak domain

for j= 1 to z
 Data.GetChannel("[9]/PowerTime").Values(j) = chD(j, "[1]/t")


if j >= zeroIndex then
sumPower = sumPower + abs(chd(j,"[3]/AA67refclean"))
sumPowerLog = sumPowerLog + abs(chd(j,"[6]/SP1Log"))
end if

if chd(j,"[9]/PowerTime") > 22 then
sumBG = sumBG + abs(chd(j,"[3]/AA67refclean"))
sumBGLog = sumBGLog + abs(chd(j,"[6]/SP1Log"))
nBG = nBG+1
nBGLog = nBGLog +1
end if


next
avgBG = sumBG/nBG
Call MsgBox(avgBG)
for i= 1 to z

  'check these units...

if i >= zeroIndex then 'chd(i,"[9]/FilteredSignal") >  0 or
chd(i,"[9]/FilteredSignal") <  -0.000000001  then 'zeroPulseInd - 3000 then
'   if pulseBool = FALSE then
'   startIndex= i
'   pulseBool= TRUE                          MIGHT HAVE TO CHANGE TO A M*(X(T1)-
X(T0)) set up
'   end if
 'Data.GetChannel("[9]/ConvertedPowerChannel").Values(i) = ((1- (-
0.0299*(chd(i,"[9]/PowerTime")-
chd(startIndex,"[9]/PowerTime"))+0.8077)))*CHD(i,"[9]/PowerChannel")
 Data.GetChannel("[9]/PowerChannel").Values(i) = (chd(i,"[3]/AA67refclean")-
avgBG)/sumPower*1
 Data.GetChannel("[9]/PowerChannel2").Values(i) = (chd(i,"[6]/SP1Log")-
avgBGLog)/sumPowerLog*1
 Data.GetChannel("[9]/ConvertedPowerChannel").Values(i) =
(CHD(i,"[9]/PowerChannel"))*(((-0.0299*(chd(i-zeroIndex + 1,"[1]/t"))+0.8077)))
 Data.GetChannel("[9]/ConvertedLogChannel").Values(i) =
(CHD(i,"[9]/PowerChannel2"))*(((-0.0299*(chd(i-zeroIndex + 1,"[1]/t"))+0.8077)))
 Data.GetChannel("[9]/PowerTimeLog").Values(i) = chD(i, "[1]/t")

else
 Data.GetChannel("[9]/PowerChannel").Values(i) = 0
 Data.GetChannel("[9]/PowerChannel2").Values(i) = 0
```

```
 Data.GetChannel("[9]/ConvertedPowerChannel").Values(i) =  chd(i,"[9]/PowerChannel")
 Data.GetChannel("[9]/PowerTimeLog").Values(i) = chD(i, "[1]/t")
end if

next
```

A.4. ACRR Transient Rod Timing Superposition



DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
David Saucier	1384	dhsauci@sandia.gov
William Martin	1384	wjmarti@sandia.gov
Thomas Ball	1384	thoball@sandia.gov
Elliott Pelfrey	1384	eapelfr@sandia.gov
Technical Library	01977	sanddocs@sandia.gov

Hardcopy—Internal

Number of Copies	Name	Org.	Mailstop
1	David Saucier	1384	1146
1	William Martin	1384	1146
1	Thomas Ball	1384	1146
1	Elliott Pelfrey	1384	1146

This Page Left Blank

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.