Passive detection of radionuclides from weak and poorly resolved gamma-ray energy spectra

Paul Kump
University of Iowa

Follow this and additional works at: https://ir.uiowa.edu/etd
Part of the Electrical and Computer Engineering Commons

PASSIVE DETECTION OF RADIONUCLIDES FROM WEAK AND POORLY RESOLVED GAMMA-RAY ENERGY SPECTRA

by

Paul Kump

An Abstract

Of a thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Electrical and Computer Engineering
in the Graduate College of
The University of Iowa

July 2012

Thesis Supervisor: Professor Er-Wei Bai

ABSTRACT

Large passive detectors used in screening for special nuclear materials at ports of entry are characterized by poor spectral resolution, making identification of radionuclides a difficult task. Most identification routines, which fit empirical shapes and use derivatives, are impractical in these situations. Here I develop new, physics-based methods to determine the presence of spectral signatures of one or more of a set of isotopes. Gamma-ray counts are modeled as Poisson processes, where the average part is taken to be the model and the difference between the observed gamma-ray counts and the average is considered random noise. In the linear part, the unknown coefficients represent the intensites of the isotopes. Therefore, it is of great interest not to estimate each coefficient, but rather determine if the coefficient is non-zero, corresponding to the presence of the isotope. This thesis provides new selection algorithms, and, since detector data is undoubtedly finite, this unique work emphasizes selection when data is fixed and finite.

Abstract Approved: ______________________________
Thesis Supervisor

____
Title and Department

____
Date

PASSIVE DETECTION OF RADIONUCLIDES FROM WEAK AND POORLY RESOLVED GAMMA-RAY ENERGY SPECTRA

by

Paul Kump

A thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Electrical and Computer Engineering
in the Graduate College of
The University of Iowa

July 2012

Thesis Supervisor: Professor Er-Wei Bai

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Paul Kump

has been approved by the Examining Committee for the thesis requirement for the Doctor of Philosophy degree in Electrical and Computer Engineering at the July 2012 graduation.

Thesis Committee: ______________________________
Er-Wei Bai, Thesis Supervisor

____
Kung-Sik Chan

____
William Eichinger

____
Anton Kruger

____
Raghuraman Mudumbai

To Tessa Harmony

ACKNOWLEDGEMENTS

In addition to thanking my committee members for their time spent reading my papers, listening to my talks, and supplying me with crucial information and advice, I acknowledge the help of several others who played an important role in my completion of this thesis. My wife Amanda has been extremely supportive through this whole process. My correspondence with electrical engineering secretary Cathy Kern during the three semesters I was away from school made getting back into the classroom an easy task. Jinzheng Li provided me with shielding data, and she and Jim Niemeier helped me with the formatting of this thesis. Most importantly, I would like to thank Professor Er-Wei Bai who provided me invaluable guidance in every aspect of this research and for whom this work would not have been possible. Er-wei is one of the, presumably few, people I will meet in life who manage to dramatically impact my life for the better. To describe all the positive encounters I have had with Professor Bai that show how he can be such a great adviser, teacher, and person, would require a separate thesis. Thank you, sincerely.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF THEOREMS

CHAPTER 1 INTRODUCTION

This thesis details the development of new physics-based methods for determining the presence of spectral signatures from one or more of a set of nuclear materials. The spectral signatures addressed in this work are weak as compared to background effects and poorly resolved as to make the detection problem non-ideal and non-trivial. Unless otherwise stated, "spectrum", and its several forms, should be taken to mean "gamma-ray energy spectrum," as to avoid any confusion with measurements based on neutrons, which are also observables from nuclear materials. Neutron-based detection methods will not be analyzed in this thesis. Further, all of the detection schemes discussed here are methods of passive detection.

The original methods presented in this paper are shown to be useful tools in the area of nuclear material detection and are demonstrated to be improvements over existing passive detection schemes. Areas that are improved upon are:

- robustness: detection schemes are applicable to a broader range of situations.
- accuracy: reduction in false alarms and of materials that go undetected.
- utility: specific materials are identified instead of just anomaly warnings.

Theoretical analyses, experimental results, and a review of relevant published works support my claims. Algorithms and simulations, when applicable, are carried out exclusively using MATLAB.

1.1 Motivation for This Work

Roughly seven million containers of trade goods are entering the 361 seaports of the United States per year, with most of the activity happening at the ports of Los Angeles, Long Beach, and New York-New Jersey [18]. In fact, these ports are operating on a 24/7 basis and processing over 11,000 containers per day, or eight containers per minute. With the dissolution of the Soviet Union, the poor security associated with its nuclear stockpile, and the War on Terror, the concern is that terrorists will use any of these containers to smuggle special nuclear materials (SNM) into the country. Experts have estimated the cost to the U.S. economy of port closures due to the detonation of a weapon of mass destruction as $1 trillion. Hence, the ultimate goal is to screen every container entering the U.S. and accurately determine if the containers are housing nuclear materials, all while staying true to the fast processing times. Therefore, detection schemes should be fast and reliable, as well as safe, in order to prevent harm to those on scene.

The demand for uncovering smuggled nuclear materials has caused a flurry of activity associated with the nuclear signatures of these materials. While a number of active detection methods have been proposed which can detect quite small amounts of material, they are ultimatley deemed unsafe because they involve potentially lethal doses of some type of radiation delivered to the objects to be examined. This leaves passive sensing as the primary candidate method.

Gamma-rays and neutrons are observables from nuclear materials allowing for the possibility of passive sensing. Most neutron detectors are based off a nonradioactive isotope of helium. The global shortage of this isotope has steered research in the direction of gamma-ray detection [39]. A number of both portable and non-portable gamma-ray spectrometers are commercially available; all suitable for such applications as checkpoint monitoring and nuclear searches where the detector and potential source are close and the expected signal is relatively large so that the gamma-ray signature peaks are well-resolved [4]. Gamma-ray signature recognition is straightforward if signals are strong and high-resolution detectors can be used. Such detectors allow for unambiguous identification of radioactive nuclides using photo peak search algorithms [28]. However, these conditions cannot be reasonably assumed for detection at seaports. Nuclear materials can be buried deep inside the

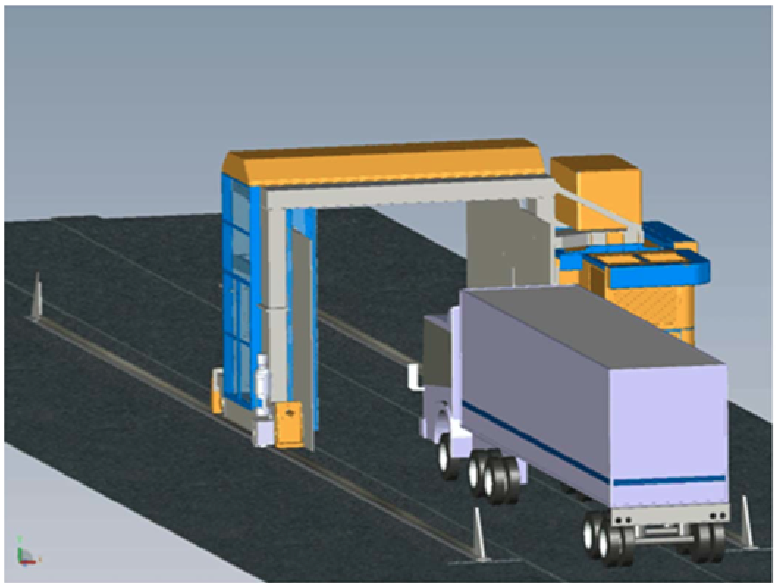


Figure 1.1: Detectors at seaports may not be close to potential sources which could be deep inside the bed of cargo trucks.

bed of cargo trucks, several meters away from detectors, as illustrated in Figure 1.1. Further, there may exist a host of materials between the detector and possible nuclear materials: the atmosphere, other cargo in the truck, and the truck itself. In this case, the signals will be weak and difficult to discern from the background radiation or from the signatures of naturally occurring radioactive materials (NORM). Moreover, large detectors are required as to allow for timely detection. Unfortunately, as the surface area of a detector increases, its spectral resolution decreases, and the defining characteristics of spectra will be challenging to recover.

1.2 Characteristics of the Detector

Detection involves using detector elements to obtain data, and then converting data to useful information via computer algorithms. An incoming photon's energy is converted to an electrical pulse that can be measured. This may rely on the photoelectric effect, discussed in Section 2.1, and is the task of the detector hardware. The pulses are sent to algorithms, and it is the algorithms' job to convert the pulses to an interpretable format.

Detectors will perform better when nuclear material signals are strong and signal to noise ratios (SNRs) are high. That is to say, the signal (gamma-ray spectrum of the nuclear material) cannot become too corrupted by the noise (random gamma-ray spectrum of the background radiation), or else detection results will be unreliable. Two concepts are central to gamma-ray detector sensitivity [27]: detection efficiency and spectral resolution. Efficiency refers to the fraction of the signal a detector actually records. The number of photons radiated by the material per unit area diminishes with distance according to the $1/r^2$ law. Since radioactive material emits radiation in a three-dimensional sphere, using a detector that is larger or that is closer to the material increases the efficiency. A more efficient detector collects data faster, reducing processing times.

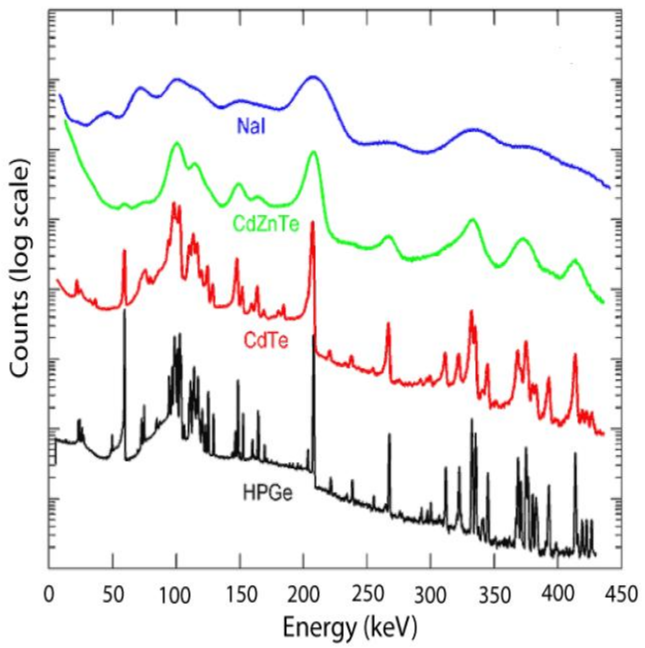


Figure 1.2: The spectrum of Pu239 as measured by different detector materials to show the differences in spectral resolution.

Spectral resolution refers to the sharpness of peaks in a spectrum. A detector with almost infinite resolution would show peaks as vertical lines with not much thickness. In reality, detectors have a finite resolution and detected peaks look more like Gaussian functions - the poorer the resolution, the wider the peak, as shown in Figure 1.2 [6]. High resolution detectors like high-purity germanium (HPGe) are expensive, heavy, and inefficient [27]. On the other hand, if a low resolution detector like sodium iodide (NaI) is used, then peaks from gamma-rays of different energies may blur together, making identification of the material a difficult task.

There are two methods for transducing the gamma-ray energy into electrical pulses [27]. One is with a scintillator material, such as sodium iodide. When a gamma-ray interacts with the scintillator, the scintillator emits a large number of photons of lower energy. A photomultiplier tube converts these photons to electrons, then multiplies the electrons to generate a measurable pulse whose voltage is proportional to the initial energy of the gamma-ray. The energy is resolved into a discrete energy bin, or channel, and the count in said bin is increased by one. The second method is with a semiconductor material. Without radiation, the semiconductor is configured to act like an insulator. When radiation strikes the material, electrons are freed from their electronic states in the atoms of the material. With the help of an electric field, the charge can be collected and then recorded.

1.3 Background Radiation

Varying background radiation that results from changes in the environment perhaps presents one of the biggest challenges to nuclear material detection. An example background spectrum is shown in Figure 1.3. Naturally occurring sources of gamma-rays can be classified into three different categories [32]: terrestrial, atmospheric, and cosmic-ray. Different minerals with various abundances of isotopes in the Th232, U238 and K40 decay chains make up the terrestrial category. Th232 and U238 have relatively long decay chains ending in lead, and K40 has a shorter decay chain but can decay in one of two branches ending in two different isotopes: Ar40 or Ca40. Radon gas, a member of the U238 decay chain, is released from the decay of radium in the Earth's soil and is classified as atmospheric background. The third category, cosmic-ray, classifies the gamma-rays produced from from muon interactions with the environment. Cosmic-ray background effects increase rapidly with altitude.

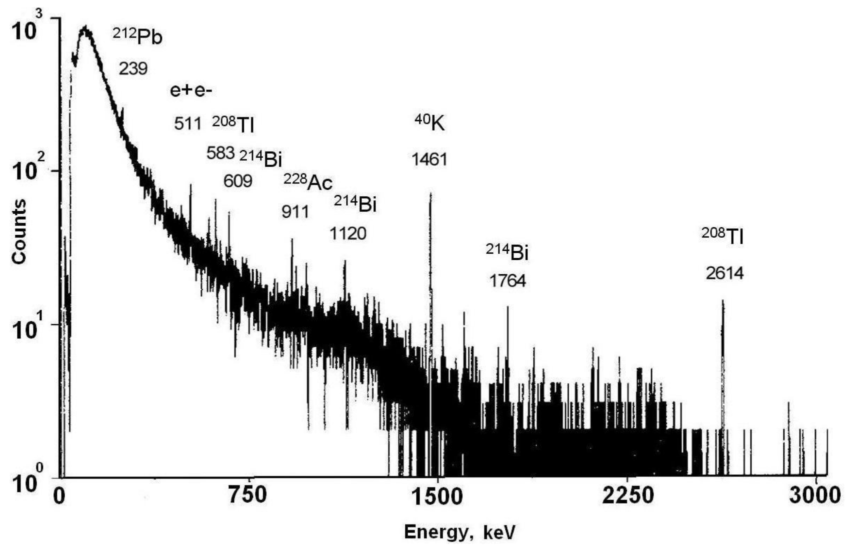


Figure 1.3: An example shape of a gamma-ray background spectrum. As time varies, the vertical scale can fluctuate due to natural and anthropological factors, but the overall shape will remain somewhat constant.

In an urban setting, man-made structures such as bridges, tunnels, buildings and roads will contain NORM [1]. It is important to note that while these variations affect the total number of counts observed over the entire range of the detector, the shape of the background spectrum remains somewhat constant over time at a fixed location. This is due to the fact that many sources of environmental radiation possess isotopes that vary proportionally to one another [1].

1.4 Overview of the Main Sections

The remainder of this thesis is organized as follows. In Chapter 2, I conduct a review of significant published results that apply to nuclear material detection. The review begins with physics concepts that explain the nature of observed spectra. The photoelectric effect justifies the use of detectors in that it shows it is indeed possible to accurately observe gamma-ray energies, and the Compton effect explains some non-ideal characteristics in the detected gamma-ray spectra. After the physics review, I provide a review of the detection techniques that are being applied today. Eventually, I recast the problem of detection as a problem of variable selection and so the literature review is concluded by examining common linear regression techniques. An in-depth look at the least absolute shrinkage and selection operator (LASSO) and its variants is conducted. Much of my original work starts with Chapter 3. This chapter begins with how to model the output of a detector as a linear equation and explains why variable selection and nuclear material detection are one in the same. The performances of traditional methods are shown to be inadequate. I propose several types of detection algorithms, most based on the methods presented in Chapter 4, and I compare them directly to the performances of the traditional methods as well as to the performances of popular variable selection methods. Gamma-ray shielding is addressed in Chapter 5, and my original methods are shown to perform well in the presence of carbon, concrete, and water shields. Lead shielding is also examined. In Chapter 6, in an attempt to make the detection methods more robust, I consider applying the method of total least squares for cases when the spectral signatures of the isotopes in question may not be known *exactly*. Chapter 7 provides summaries and conclusions and also gives recommendations for future work. I relegate technical proofs to Chapter 8 and supply my original MATLAB code in Chapter 9. Finally, all of the sources used in this thesis are listed in the references section.

CHAPTER 2
LITERATURE REVIEW

2.1 Physics Review

2.1.1 The Photoelectric Effect

Photons are packets of energy traveling at the speed of light with no rest mass and no electrical charge. Electromagnetic radiation, e.g., gamma-rays, consists of photons, and may be measured as a wavelength, frequency, or an energy. In the latter part of the 19th century, experiments showed that photons incident on certain metal surfaces caused electrons to be emitted from the surfaces. This phenomenon is known as the photoelectric effect, and the emitted electrons are called photoelectrons [35].

Figure 2.1 shows a diagram of an apparatus to observe the photoelectric effect. A quartz tube holds two metallic plates, each connected to opposite terminals of a power supply, V. In the absence of electromagnetic radiation, the ammeter reads zero. However, when radiation is incident on the plate connected to the negative terminal, a current can be detected, indicating a flow of electrons across the gap between the two plates. The electrons are photoelectrons emitted from the emitter and collected by the collector. When a large voltage is applied, the photoelectric current is saturated and reaches a maximum value, as also shown in Figure 2.1. When V is negative, the current drops to a very low value. Only those electrons having a kinetic energy greater than the magnitude of $e \cdot V$ reach the collector, where e is the elementary charge. When V is equal to or more negative than $-V_s$, no photoelectrons reach the collector and the current is zero. As the figure suggests, V_s is independent of the radiation intensity. The maximum kinetic energy of the photoelectrons is related to V_s by the relationship $K_{max} = e \cdot V_s$.

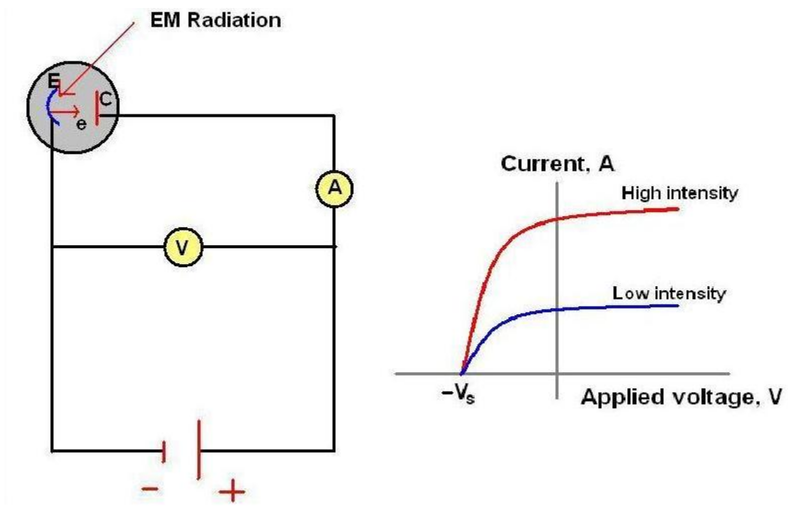


Figure 2.1: Left: Circuit diagram for observing the photoelectric effect. Light strikes the emitter (E) and photoelectrons are ejected from the plate. Electrons moving from the emitter to the collector (C) produce a current to be read by the ammeter. Right: Photoelectric current versus applied voltage for two intensities. At voltages equal to or more negative than $-V_s$, the current is zero.

Einstein showed that a photon is so localized that it gives all of its energy $E = hf$ to a single electron in the metal, where h is Planck's constant and f is the frequency of radiation. The maximum kinetic energy of a photoelectron can be

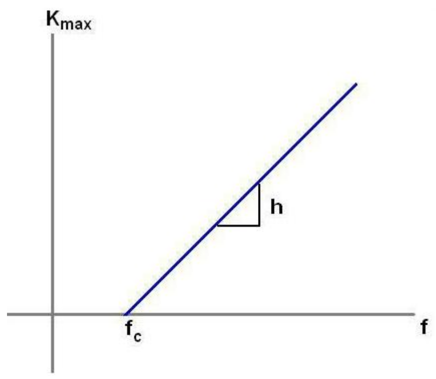


Figure 2.2: A plot of K_{max} of photoelectrons versus incident radiation frequency. Photons with a frequency less than f_c do not have enough energy to eject an electron from the metal.

written as

$$K_{max} = hf - \phi \tag{2.1}$$

where ϕ is called the work function of the metal. The work function is on the order of a few electron volts, depending on the metal, and represents the minimum energy with which an electron is bound in the metal. The graph of (2.1) versus frequency is shown in Figure 2.2. The intercept of the frequency axis gives the cut-off frequency $f_c = \phi/h$ below which no photoelectrons are emitted, regardless of the radiation intensity. The slope of the line is Planck's constant h, and if the line were to extend, the intercept of the K_{max} axis would be $-\phi$. Thus a series of graphs for different metals would be a collection of parallel lines.

The photomultiplier tube is one application of the photoelectric effect and acts like a switch in an electric circuit. It generates a current in the circuit when light of sufficiently high frequency is incident on a metallic plate but produces no current in the dark. In a nuclear material detector, the scintillator reduces the energy of the incoming gamma-ray photons to that of light (optical photons). The optical photons are then sent to the photomultiplier tube to generate electrical signals.

2.1.2 The Compton Effect

The Compton Effect was first described by Arthur Holly Compton in 1923 to explain the shift in observed wavelength (frequency) of a scattered X-ray after a collision with a free electron [8]. Compton explained that when one adopts a quantum mechanical model of light, an increase in the observed wavelength of a photon after a collision with a free electron can be predicted from conservation of energy (an elastic collision) and is a function of the scattering angle (θ in Figure 2.3). This is important to nuclear material detection as a scattered gamma-ray (photon) from an electron in the detector material would report as having lower energy than would otherwise be the case, and, if not accounted for, could produce detection errors.

To begin to explain the Compton Effect, assume the photon behaves like a particle of zero mass and energy $E_\gamma = hf = hc/\lambda$, where h is Planck's constant, c is the speed of light, and f and λ are the photon's frequency and wavelength, respectively. The photon collides elastically with a free electron which is initially at rest, as shown in Figure 2.3. Applying the principle of conservation of energy gives

$$\frac{hc}{\lambda_i} = \frac{hc}{\lambda_f} + K_e \tag{2.2}$$

where hc/λ_i is the energy of the incident photon, hc/λ_f is the energy of the scattered photon, and K_e is the kinetic energy of the electron after the collision. The speed of the electron v may be comparable to that of light, so relativity must be taken into account, i.e., $K_e = \gamma m_e c^2 - m_e c^2$ where $\gamma = 1/\sqrt{1 - v^2/c^2}$ and m_e is the mass of an electron. So

$$\frac{hc}{\lambda_i} = \frac{hc}{\lambda_f} + \gamma m_e c^2 - m_e c^2. \tag{2.3}$$

One can then apply the law of conservation of momentum. For simplicity, consider only two dimensional scattering. The magnitude of a photon's momentum is given by $p = E_\gamma/c = hf/c = h/\lambda$, and the momentum of the electron after the collision is $p_e = \gamma m_e v$. By applying conservation principles in each direction, the following expressions hold

$$x\ component: \quad \frac{h}{\lambda_i} = \frac{h}{\lambda_f}\cos\theta + \gamma m_e v \cos\phi \tag{2.4}$$

$$y\ component: \quad 0 = \frac{h}{\lambda_f}\sin\theta + \gamma m_e v \sin\phi \tag{2.5}$$

where the angles ϕ and θ are measured as in Figure 2.3.

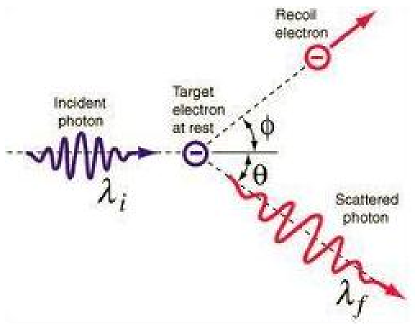


Figure 2.3: Compton scattering of a photon by an electron. The collision reduces the photon's energy, thus the scattered photon has a larger wavelength.

The angle ϕ and speed v can be eliminated in Equations (2.3)-(2.5) to obtain a single expression that relates λ_i, λ_f and θ. After some algebra, the resulting equation is known at the Compton shift equation

$$\Delta\lambda = \lambda_f - \lambda_i = \frac{h}{m_e c}(1 - \cos\theta) \tag{2.6}$$

and describes the change in wavelength as a function of the scattering angle. The quantity $h/m_e c = 0.00243$ nm in equation (2.6) is called the Compton wavelength. An illustration of this effect on intensity counts is shown in Figure 2.4.

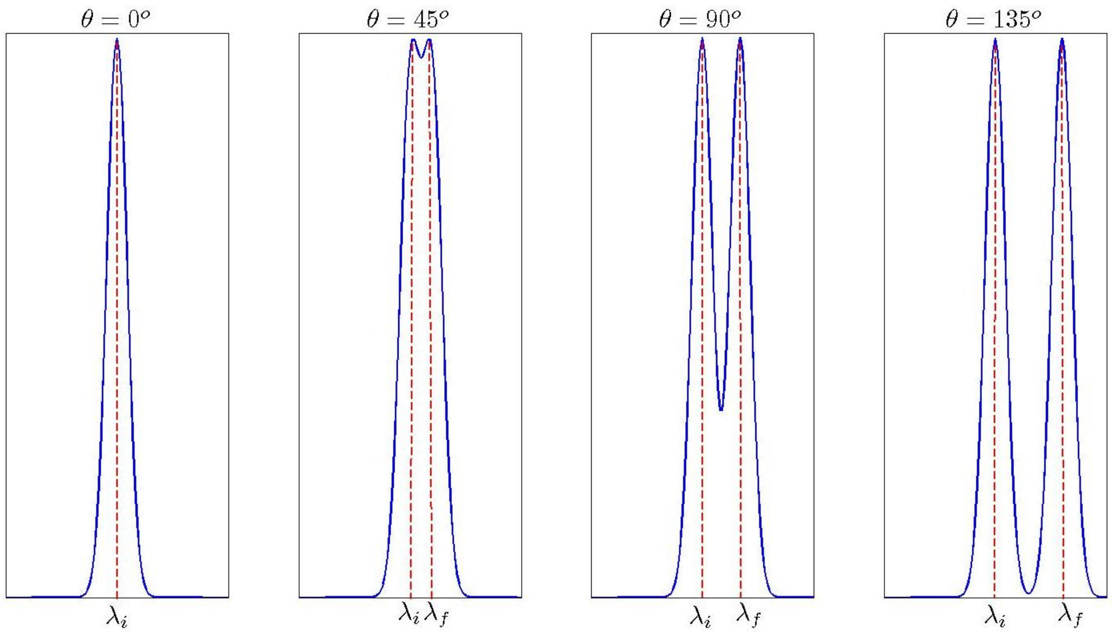


Figure 2.4: Intensity counts for a Compton-scattered photon at various scattering angles shown with the unscattered peak.

Prior to arriving at the detector, gamma-rays will naturally be scattered with various scattering angles – actually, an infinite amount of angles forming a continuum. Since the amount of energy transmitted to the recoil electron is dependent on these scattering angles, an energy continuum known as the Compton continuum will be observed by the detector. The Compton continuum is shown in Figure 2.5.

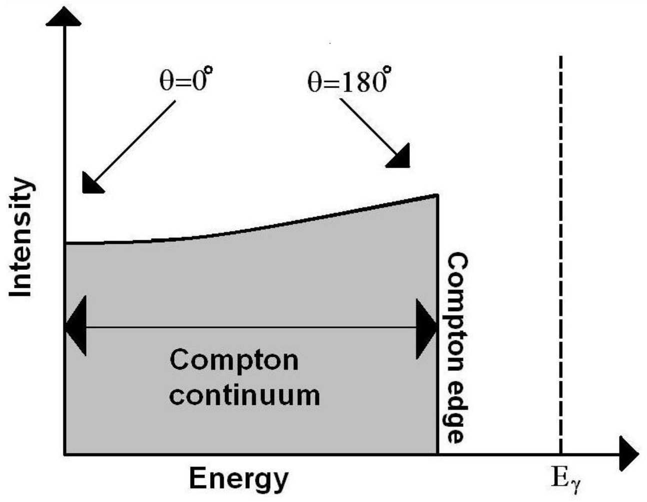


Figure 2.5: Compton continuum showing the energy intensities of recoil electrons for all possible scattering angles. The recoil electron cannot have energy more than the Compton edge.

For each scattering angle θ, one can compute the energy transferred to the electron (or, equivalently, the energy lost by the photon) K_e by the relationship

$$K_e = E_\gamma\left(1 - \frac{1}{1 + \frac{E_\gamma(1-\cos\theta)}{m_e c^2}}\right). \tag{2.7}$$

The relationship follows directly from Equations (2.2) and (2.6). Observe from equation (2.7) that the maximum kinetic energy of the recoil electron occurs when the scattering angle is $\theta = 180^0$. This value

$$K_{e,max} = \frac{E_\gamma}{1 + \frac{m_e c^2}{2E_\gamma}} < E_\gamma$$

is known as the Compton edge and is the theoretical limit to the amount of energy able to be delivered to the electron by scattering. At the same time, it is the maximum amount of energy that an incident photon can lose by scattering. It makes sense that this maximum value is less than the initial energy E_γ.

2.2 Contemporary Detection Techniques

2.2.1 Traditional Peak Detection Algorithms

Nuclear material detection is relatively straightforward when signal to noise ratio is large and high-resolution detectors can be used. In such cases, it is popular to employ peak detection algorithms as the detection scheme; peak detection algorithms like findpeaks.m, a MATLAB script developed at the University of Maryland [30], are simple, have a fast run time, and are effective. There are several variations of peak detection algorithms ([30], [37], [34], [26] [22]) but all rely on two main steps: (1) differentiating the signal to find local extrema, and (2) identifying which extrema are in fact peaks. Some algorithms go further and fit the peak to a Gaussian function to determine the peaks' heights and widths. The locations of the peaks are compared to a library – an array that stores data representing the physical characteristics of the possible threat nuclides – and a decision is made as to which, if any, materials are present. Under shielding conditions, peaks in the observed gamma-ray spectrum may not be obvious. Further, these kinds of algorithms are highly sensitive to noise and may not perform correctly under less than ideal conditions.

Any freshman calculus text will describe the procedure for finding the peaks of a function: Observe when the first derivative has a downward-going zero-crossing, or equivalently, determine the critical values of the function when the second derivative is negative. Theoretically, this idea alone works perfectly, but in a realistic experiment, however, the function will consist of random noise which can produce false zero-crossings. In an effort to help combat noise, it is common practice to first smooth the function and then look at its derivative's zero-crossings. This method was described in [26] where the derivative must be approximated by a difference because of the discrete nature of the data.

The simplest smoothing filter is an unweighted averaging smoother; it replaces each point in the signal that is within a feasible region with the average of m adjacent points. A centered smoother with odd $m > 1$ is most common and has the form

$$\hat{h}f(n) = \frac{f(n) + \sum_{j=1}^{(m-1)/2} \big(f(n-j) + f(n+j)\big)}{m} \tag{2.8}$$

with a feasible region of

$$N - \frac{m-1}{2} \geq n > \frac{m-1}{2}$$

where the signal to be smoothed is defined for $1 \leq n \leq N$. Points that fall outside of the feasible region are boundary points where the filter (2.8) is undefined. There are several methods to dealing with boundary points: zero padding, reflection, periodic repetition, even making the parameter m smaller as the boundary approaches. The best method depends on the type of data, but all of the methods produce adequate smoothing results.

Though crucial to the overall effectiveness of the smoothing filter, the choice of parameter m will not effect the location of the peak. A large m, relative to the length of the peak, will only distort the peak by reducing the peak's amplitude and increasing its width [30]. Figure 2.6 shows the peak distortion for different smoothing ratios; smoothing ratio is defined as the ratio of m to peak width.

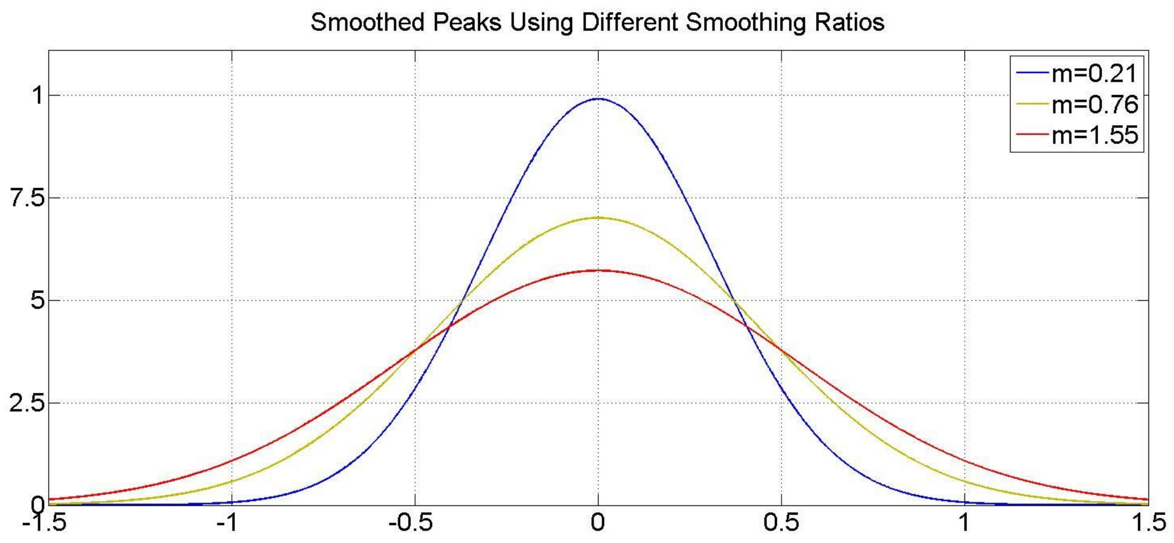


Figure 2.6: Smoothed peaks using three different smoothing ratios. Smoothing the peak does not change its location.

In the case of nuclear material detection, a larger m may be used since it is the location of the peak and not necessarily the amplitude or the width that is important. One must be careful not to increase m too large such that multiple peaks begin to overlap.

The filter (2.8) is constructed of a sum of equal weighted terms and is sometimes referred to as a rectangular filter. In fact, there is no need for the coefficients to be equal; findpeaks.m can use the more generalized triangular filter (2.9) with feasible region as in (2.10) (both shown with $m = 5$) which can increase signal to noise ratio more so than its rectangular cousin.

$$\hat{f}(n) = \frac{1}{9}\big(f(n+2) + 2f(n+1) + 3f(n) + 2f(n-1) + f(n-2)\big) \tag{2.9}$$

$$-2 \geq n > 2 \tag{2.10}$$

After the signal has been smoothed, an approximation of the derivative is applied to the now (almost) noiseless signal. An approximation, for example the centralized difference shown in (2.11), must be used because the input data is discrete. A user threshold is then used to discriminate zero derivatives (local extrema) from non-zero derivatives.

$$f^{'}(n) = \frac{f(n+1) - f(n-1)}{2}, \quad 1 < n \leq N-1 \tag{2.11}$$

The last step is to identify which local extrema are in fact peaks by observing the algebraic sign of the second derivative: those extrema which have a corresponding second derivative less than zero are indeed local maxima. Again, an approximation for the second derivative must be used. A centralized approximation used in findpeaks.m has the form

$$f^{''}(n) = f(n+1) - 2f(n) + f(n-1), \quad 1 < n \leq N-1$$

Nuclide identification is done by matching the energy location of the peak to a list of energies in a nuclide library. Automatic calibration methods make knowing the energy location of a peak equivalent to knowing its channel location found by peak detection algorithms [22]. A peak is assigned to a candidate nuclide if the library entry for that nuclide is within a threshold (typically 1keV) of the identified energy. The nuclide library is a list of gamma-ray energies with nuclide identifications such as nuclide name, atomic number, and mass.

In situations where the signal to noise ratio is low, peak detection algorithms perform poorly. An analysis of findpeaks.m was conducted in which the algorithm was tested on a noise corrupted signal with an SNR of -10 dB. The simulation details are explained in Section 3.2, and the results are put in proper context by comparing them to results of other detection algorithms for the same simulations.

2.2.2 Energy Windowing Techniques

Use of an energy windowing algorithm, as described in [1] and [11], is to compare the shape of the observed spectrum to that of the background and to quantify the similarity or difference. This method can have a better performance than traditional peak detection because the shapes of NORM spectra are very similar to that of the background, as shown in the top panel of Figure 2.7 [11], while man-made isotopes, such as SNM, have spectra with more low energy radiation, as shown in the bottom panel of Figure 2.7.

More specifically, energy windowing is to divide the spectrum into windows and add all the counts in each window to obtain a window number. The difference or similarity to the background windows can be quantified by various methods. One such

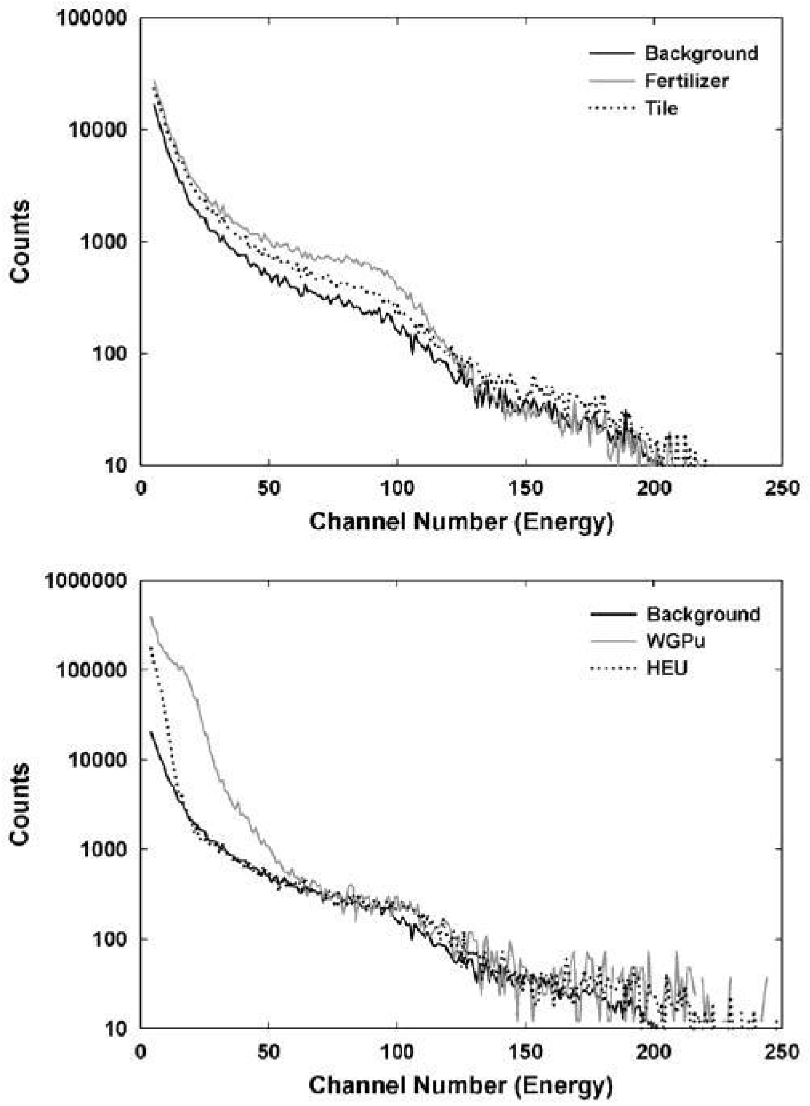


Figure 2.7: The spectral shape of gamma-ray background radiation as compared to that of NORM (top) and of nuclear threats (bottom).

method normalizes the counts in each window by the total counts in a reference window and compare that normalized count to a corresponding background normalized count. For instance, consider a source that emits only one energy and a windowing scheme dividing the spectrum into two parts: One window extending from the detection threshold to just past the Compton Edge (recall a mono-energetic photon will produce a continuum of observed energies), and the second window containing the high energy remains. One could choose the high energy window as the reference window so the normalized window counts are:

$$R_{EW} = \frac{N_{EW}}{N_H}$$

where N_{EW} is the window count of a specific window and N_H is the window count in the high energy reference window. The background windows are normalized in the same manner, with the same windows.

The next step is to compare each normalized window with the corresponding normalized background window, and [11] chooses to alarm when

$$R_{EW} > R_B + \kappa\sigma_{R_B}$$

where R_B is the normalized background window, κ determines the sensitivity, and σ_{R_B} is the standard deviation of the normalized background window given as

$$\sigma_{R_B} = R_B\sqrt{\frac{1}{N_{LB}} + \frac{1}{N_{HB}} - \frac{2\rho\sqrt{N_{LB}N_{HB}}}{N_{LB}N_{HB}}} \tag{2.12}$$

where N_{LB} and N_{HB} are the unnormalized window counts for the lower energy window and the higher energy window, respectively, and ρ is the correlation coefficient between N_{LB} and N_{HB}. Equation (2.12) is derived by linearizing $R_B = N_{LB}/N_{HB}$ using a Taylor expansion approximation, applying propagation of uncertainty techniques, and noting that the standard deviation σ is the square root of the number of counts for Gaussian statistics [11]. For their experiments, the authors of [11] assume maximal correlation, $\rho = 1$.

The spectral comparison ratio (SCR) method [1], [31] is an energy windowing technique that works to maximize threat discrimination by adjusting window parameters, e.g., window placement, to maximize a certain figure of merit. At the heart of the SCR algorithm is a residual between a window's observed count and its predicted count, with the prediction being based on a ratio of expectation values of two windows. More precisely, the residual α^{ij} is

$$\alpha^{ij} = Count^i - \frac{Expectation^i}{Expectation^j} \cdot Count^j \tag{2.13}$$

where *Count* is the number of observed counts in a window, and *Expectation* is the expected number of counts in a window which can be based on previous observations. The number of windows is a variable which needs to be chosen. If the number of windows is more than two, there will be multiple residuals. Specifically, for N windows, there will be $N-1$ linearly independent residuals. [1] goes on to develop a figure of merit based on the popular Fisher Linear Discriminant [2] to discriminate between the α of threats from the α of benign sources. The authors maximize the figure of merit using Powell's algorithm [33] to determine window placements for maximal discrimination.

As an experiment, a target was buried in soil at different depths for producing different classes of collected data sets. Soil, along with uranium ore, comprised the background. The detector was a 10x10x40cm sodium iodide detector capable of resolving spectra to 1024 energy channels ranging from 0 keV to 3260 keV. Four energy windows were used, and a particle swarm optimization (PSO) [40] method was further used to determine optimal window widths, i.e., not all window sizes were

Table 2.1: Results from an experiment in which energy windowing techniques are used to detect a target material from ore buried in soil.

Method	Target Alarm Rate	Nuisance Alarm Rate
Fixed Width	0.9405	0.2220
PSO	0.9702	0.1250

equal. The target was then replaced with a nuisance source of natural ore, also placed at different soil depths, and the experiment was repeated. The target and nuisance alarm rates are shown in Table 2.1 [40] with fixed window widths and PSO-chosen widths. Unfortunately, the authors do not put their work in context by providing any sort of signal to noise measurement. Regardless, energy windowing techniques cannot say anything regarding the type of material being detected. Further, the utility of energy windowing techniques is questionable given that the main assumption – the defining characteristic of SNM is its abundance of low energy – is arguable under modest shielding conditions, where low energy gamma-rays are easily attenuated [23].

2.3 Linear Regression Techniques

Chapter 3 describes how one can model the output of a nuclear material detector as a linear equation. The problem of detecting the material is then recast as a variable selection problem where the variables of interest are components of an unknown sparse parameter vector. Thus this section reviews the many contributions from various authors to the field of linear regression, and specifically variable selection, which will be applied in whole or in part to our nuclear material detection problem.

Given a linear equation

$$Y_n = X_n \beta^* + V_n, \tag{2.14}$$

or equivalently

$$Y_n = \sum_{j=1}^{p} \mathbf{x}_j \beta_j^* + V_n,$$

$$X_n = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_p}), \quad \beta^* = \begin{pmatrix} \beta_1^* \\ \beta_2^* \\ \vdots \\ \beta_p^* \end{pmatrix},$$

where $V_n = (V(1), ..., V(n))^T \in \Re^n$ is an independent and identically distributed (i.i.d.) random noise sequence with zero mean and finite variance, $X_n \in \Re^{n \times p}$ is the regressor matrix with regressors $\mathbf{x}_j$, $Y_n = (y(1), ..., y(n))^T \in \Re^n$ is the output sequence or the response sequence, and $\beta^* \in \Re^p$ is the unknown parameter vector; one wishes to gain some sort of knowledge about β^*, usually by estimating its parameters β_j^* and/or inferring its active set. The active set is the set $A^* = \{j \ : \ \beta_j^* \neq 0\}$, i.e., those β_j^*'s which are actively contributing to the observed data Y_n. Estimating the active set is known as variable selection.

Suppose $\hat{\beta} = (\hat{\beta}_1, \hat{\beta}_2, ..., \hat{\beta}_p)^T$ is the estimate of β^*, and $\hat{\beta}_j = 0, \ j = d+1, ..., p$, for some positive integer d. Then, one can trim off the irrelevant regressors from X_n and build a model

$$\hat{Y}_n = \sum_{j=1}^{d} \mathbf{x}_j \hat{\beta}_j$$

to predict the output.

The desirable properties of variable selection are [12],[46]:

$$\Pr\{A = A^*\} \to 1, \;\; as\; n \to \infty \quad (set\; consistency) \tag{2.15}$$

$$\Pr\{\hat{\beta}_j = \beta_j^*\} \to 1, \;\; as\; n \to \infty, \quad j = 1, ...d \quad (parameter\; consistency). \tag{2.16}$$

In other words, those non-zero β_j^*'s and zero β_j^*'s are correctly identified, and their estimates grow closer to the true values as more data is available.

2.3.1 The Ordinary Least Squares Estimate

If the noise V_n corrupting the output sequence Y_n is small, model (2.14) is approximately $Y_n \approx X_n\beta^*$. The regressor matrix X_n is usually not square, so we cannot take its inverse to solve for β^*. Instead, we can multiply from the left by X_n^T, $X_n^T Y_n \approx X_n^T X_n \beta^*$, and now take an inverse as $X_n^T X_n$ is surely square. Assuming that the inverse exists, we now have

$$\beta^* \approx (X_n^T X_n)^{-1} X_n^T Y_n = \hat{\beta}_{LS}. \tag{2.17}$$

$\hat{\beta}_{LS}$ is known as the ordinary least squares estimate, or simply the least squares estimate, and can also be written equivalently as

$$\hat{\beta}_{LS} = \beta^* + (X_n^T X_n)^{-1} X_n^T V_n.$$

Because of its closed-form solution, the popular least squares estimate is efficient computationally. Geometrically, the least squares estimate is a projection onto the linear space spanned by the regressors and satisfies

$$\hat{\beta}_{LS} = arg \min_{\beta} ||Y_n - X_n\beta||_2,$$

where $||\cdot||_2$ is the Euclidean norm.

Though the least squares estimate is convergent [14], i.e., properties (2.15) and (2.16) are satisfied at $n = \infty$, this estimate does not solve the problem of variable selection [5]. In practice, the number of data points never reaches infinity, and the least squares estimate is unlikely to force the estimates to zero for those j for which $\beta_j^* = 0$. The result is many uniformly small estimates, and, without knowing d, determining how small is actually zero is impossible.

2.3.2 Regularized Ordinary Least Squares

The least absolute shrinkage and selection operator (LASSO) [38] is a popular technique for variable selection in linear regression models because, unlike ordinary least squares, it can produce a sparse estimate. The sparsity of a LASSO-generated estimate is a result of employing an L_1-type penalty on the unknown coefficients by minimizing

$$J(\beta) = ||Y_n - X_n\beta||_2^2 + t\sum_{j=1}^{p}|\beta_j| \tag{2.18}$$

where $t \geq 0$ is the regularization parameter that works to keep a balance between the two terms in (2.18). The much involved task of selecting an appropriate t is discussed in Section 2.3.5. In an orthogonal design, it is easy to see the utility of LASSO to produce sparse estimates where the solution is [38]

$$\hat{\beta}_j = sign(\hat{\beta}_{LSj})(|\hat{\beta}_{LSj}| - t/2)^+, \quad j = 1, ..., p$$

where $\hat{\beta}_{LSj}$ is the jth component of the least squares estimate $\hat{\beta}_{LS}$, and $(x)^+ = x,\ x > 0;\ 0,\ x \leq 0$.

In 2006, it was shown [44] that the LASSO is not a so-called oracle procedure as it does not enjoy the desirable properties of model selection (2.15) and (2.16). Specifically, [44] showed that a (almost necessary and) sufficient condition for LASSO consistency is

$$|c_n(2,1)c_n^{-1}(1,1)s| \leq 1 - \epsilon_n \tag{2.19}$$

where $c_n(1,1) = \frac{1}{n}(\mathbf{x}_1, ..., \mathbf{x}_d)^T(\mathbf{x}_1, ..., \mathbf{x}_d)$, $c_n(2,1) = \frac{1}{n}(\mathbf{x}_{d+1}, ..., \mathbf{x}_p)^T(\mathbf{x}_1, ..., \mathbf{x}_d)$, $s = (sign(\beta_1^*), ..., sign(\beta_d^*))^T$, $\epsilon_n > 0$, $\epsilon_n \to 0$ as $n \to 0$, and the inequality is understood component-wise. In general, the irrepresentable condition (2.19) is not satisfied and LASSO is not consistent. Nevertheless, the simplicity of the LASSO motivates its continued use and the creation of its variants.

2.3.3 Least Angle Regression (LARS)

The question still remains as to how to solve the LASSO minimization problem (2.18). Least angle regression (LARS) [9] is an algorithm that can efficiently compute the LASSO solution path for every value of t in (2.18). The LARS algorithm begins with all coefficients equal to zero and builds a model by successively introducing one variable at a time, all while updating the least squares fit and the L_1-norm defined as

$$|\hat{\beta}_1| + |\hat{\beta}_2| + ... + |\hat{\beta}_p| = \sum_{j=1}^{p} |\hat{\beta}_j|. \tag{2.20}$$

The generated estimates are built in successive steps so that after k steps only k of the $\hat{\beta}_j$'s are non-zero. Denote the active set during the kth step as A_k. The current prediction is given by

$$\hat{\mu}_{\mathbf{k}} = \sum_{j \in A_k} \mathbf{x}_j \hat{\beta}_j$$

where $\hat{\beta} = (\hat{\beta}_1, ..., \hat{\beta}_p)^T$ is the current candidate vector of regression coefficients.

The order in which the variables are introduced depends on their respective predictors' correlation with the current residual: The variable with a corresponding predictor of the highest correlation will be introduced next. What makes LARS a more efficient algorithm than its predecessors, e.g., forward stagewise [21], is the notion that each variable is introduced with only as much of a predictor as it deserves; the coefficient of the current predictor is increased only up to the point where another predictor has as much correlation with the current residual. This new predictor's coefficient is introduced and both coefficients are increased along the path that bisects the angle formed by the two predictors. The process is repeated until all of the predictors have been introduced.

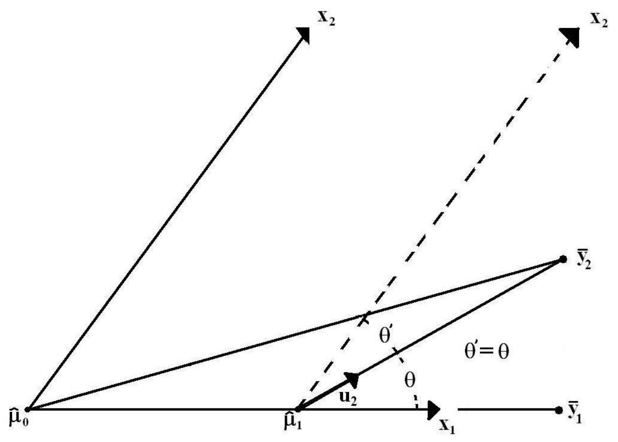


Figure 2.8: Geometric interpretation of the LARS algorithm with $p = 2$ coefficients.

For clarity, consider an example with $p = 2$ coefficients. The algorithm begins with $\hat{\mu}_0 = 0$, as the response has been standardized to have zero mean. Project the response, Y_n, onto the two dimensional space spanned by $\mathbf{x}_1$ and $\mathbf{x}_2$. Figure 2.8 has the projection $\bar{\mathbf{y}}_2$ closer to vector $\mathbf{x}_1$ than $\mathbf{x}_2$, i.e., $\bar{\mathbf{y}}_2 - \hat{\mu}_0$ makes a smaller angle with $\mathbf{x}_1$ than with $\mathbf{x}_2$. LARS then augments the initial prediction $\hat{\mu}_0$ along the direction of $\mathbf{x}_1$ to $\hat{\mu}_1 = \hat{\mu}_0 + \gamma_1 \mathbf{x}_1$.

Algorithm 2.1 Least Angle Regression Algorithm

```
1. Standardize the predictors to have zero mean and variance 1. Start with the
   residual $r = Y_n - \bar{\mathbf{y}}$, $\beta_1, ..., \beta_p = 0$.
2. Find the predictor $\mathbf{x}_j$ most correlated with $r$.
3. Move $\beta_j$ from 0 towards its least-squares coefficient $\langle \mathbf{x}_j, r \rangle$, until some other
   competitor $\mathbf{x}_k$ has as much correlation with the current residual as does $\mathbf{x}_j$.
4. Move $(\beta_j, \beta_k)$ in the direction defined by their joint least squares coefficient
   of the current residual on $\langle \mathbf{x}_j, \mathbf{x}_k \rangle$, until some other competitor $\mathbf{x}_l$ has as much
   correlation with the current residual.
5. Continue in this way until all $p$ predictors have been entered. After $p$ steps,
   the full least squares solution is obtained.
```

Forward stagewise would choose γ_1 equal to some small user-defined parameter and repeat many times. This would result in a staircase path typical of the forward stagewise algorithm. Classic forward selection algorithms would choose γ_1 large enough for $\hat{\mu}_1$ to equal $\bar{\mathbf{y}}_\mathbf{1}$, the projection of Y_n onto $\mathbf{x}_1$ [9]. LARS chooses an intermediate value that makes $\bar{\mathbf{y}}_\mathbf{2} - \hat{\mu}_1$ equally correlated with $\mathbf{x}_1$ and $\mathbf{x}_2$. That is to say that LARS chooses γ_1 such that θ, the angle $\bar{\mathbf{y}}_\mathbf{2} - \hat{\mu}_1$ makes with $\mathbf{x}_1$, is equal to θ', the angle $\bar{\mathbf{y}}_2 - \hat{\mu}_1$ makes with $\mathbf{x}_2$. Then the estimate is updated as

$$\hat{\mu}_2 = \hat{\mu}_1 + \gamma_2 \mathbf{u}_2$$

where $\mathbf{u}_2$ has unit length. In this case γ_2 is increased such that $\hat{\mu}_2 = \bar{\mathbf{y}}_2$, but when $p > 2$, γ_2 would only be increased until the next predictors equally correlated, and so on. Algorithm 2.1 [20] provides a summary.

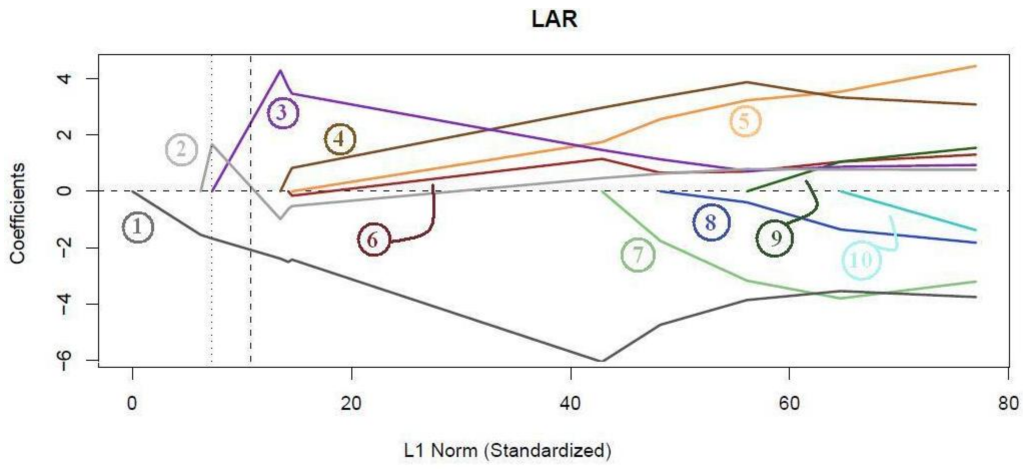


Figure 2.9: The coefficient profile generated by LARS in an example concerning ten variables. The predictors are standardized to have mean zero and variance 1. The coefficients are given on their original scale, on which the details are more clear.

Algorithm 2.2 LASSO Modification to the LARS Algorithm

```
5a. If a non-zero coefficient hits zero, drop it from the active set and recompute
    the current joint least squares direction.
```

As described above, the algorithm introduces one variable at a time, all while keeping track of the L_1-norm (2.20). One can then construct a graph of the magnitudes of the coefficients versus the L_1-norm. In a simulated example of ten variables, [20] arrives at the coefficient profile shown in Figure 2.9 [20]. Upon inspection of the profile, one could make the conclusion that variables 1 and 2 are important, i.e., they greatly affect the observed data, because they are the first variables to be introduced, and variables 3 through 10 are unimportant. A different person might conclude that variables 1 through 3 are important and variables 4 through 10 are unimportant. Yet another person may observe the large gap between the introductions of variables 6 and 7 and conclude that variables 1 through 6 substantially contribute to the data and variables 7 through 10 do not. The first person would have limited the L_1-norm by the dotted line shown in Figure 2.9, the second person with the dashed line shown in the same figure, and the last person would have limited the L_1-norm somewhere around 40 (between the introductions of variables 6 and 7). If the L_1-norm is unlimited, then the LARS estimate approaches the ordinary least squares estimate. The idea of limiting the L_1-norm for the purpose of variable selection is the very idea behind LASSO, and the LARS method can generate the entire solution path of the

LASSO with one simple modification outlined in Algorithm 2.2 [20].

2.3.4 More Selection Operators

2.3.4.1 Adaptive LASSO

LASSO (2.18) can be efficiently solved for every parameter t using the LARS algorithm. Nevertheless, whether it can capture the true index set A^* depends on condition (2.19). Observe that the condition depends only on the data X_n and, in general, is not satisfied. However, a non-negative weighting vector w may be introduced into the LASSO minimization

$$J(\beta) = \min_{\beta} ||Y_n - X_n\beta||_2^2 + t\sum_{j=1}^{p} w_j|\beta_j| \tag{2.21}$$

or equivalently,

$$J(\beta) = \min_{\beta} ||Y_n - \sum_{j=1}^{p} \mathbf{x_j}\beta_j||_2^2 + t\sum_{j=1}^{p} w_j|\beta_j|$$

where

$$0 \leq w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{pmatrix} = \begin{pmatrix} w_{11} \\ \vdots \\ w_{1d} \\ w_{21} \\ \vdots \\ w_{2(p-d)} \end{pmatrix} \in \Re^p,$$

and d is unknown. Now, the irrepresentable condition (2.19) becomes

$$|a_{nj}| \leq w_{2j} - \epsilon_n, \;\; j = 1, ..., p-d$$

where $a_n = c_n(2,1)c_n^{-1}(1,1)\hat{s}$ and $\hat{s} = (w_{11}sign(\beta_1^*), w_{12}sign(\beta_2^*), ..., w_{1d}sign(\beta_d^*))^T$. It is interesting to observe that now the condition depends on the choice of the weights w_{1j} and w_{2j} and can be satisfied if the weights were chosen in such a way that w_{1j}'s are small and w_{2j}'s are large. Of course, the problem is that it is not known which $\beta^* = 0$ and which $\beta^* \neq 0$. To overcome this, the adaptive LASSO [46] uses the idea of data-dependent weights.

For any $\gamma > 0$, define the components of the weight vector as $w_j = 1/|\hat{\beta}_{LS_j}|^{\gamma}$. As the sample size grows, the weights for the zero coefficients w_{2j} grow without bound, whereas the weights for the non-zero coefficients w_{1j} are bounded. Then, under the convergence rate assumptions that $t/\sqrt{n} \to 0$ and $tn^{(\gamma-1)/2} \to \infty$ as $n \to \infty$, the set estimated by the adaptive LASSO, A, is one such that $A = A^*$ in probability [46].

Just like LASSO, the adaptive LASSO is a convex optimization problem; it does not suffer from issues arising from local minima, and the global minimizer can be efficiently solved. The efficient algorithms used to solve the LASSO (e.g., LARS) can be used to solve the adaptive LASSO, i.e., Algorithm 2.3.

Algorithm 2.3 LARS Algorithm for the Adaptive LASSO

1. Define $\mathbf{x_j}^{**} = \mathbf{x_j}/w_j,\ j = 1, ..., p$
2. Solve the LASSO problem for all t,

   $\hat{\beta}^{**} = arg\min_{\beta} \left\{ ||Y_n - \sum_{j=1}^{p} \mathbf{x_j}^{**}\beta_j||_2^2 + t\sum_{j=1}^{p} |\beta_j| \right\}$.
3. The jth component of the adaptive LASSO estimate is $\hat{\beta}^{**}/w_j,\ j = 1, ..., p$

2.3.4.2 Group LASSO

Often times in variable selection it is desirable to select variables that are grouped together. Such cases arise naturally in many situations, nuclear material detection included. In these situations, the mechanics of any variable selection algorithm should be such to encourage the selection of entire groups of variables instead of the selection of individual ones. Thus variable selection methods like LASSO and LARS are sub-optimal when variables are grouped [42].

In the case of nuclear material detection, the unknown intensity vector β^* in (2.14) describes the strength of some candidate isotopes which may or may not be present. Each candidate isotope may have more than one relevant spectrum as decided by its branching ratios. These are its sub-spectra, as described in Section 5.1, and their unknown intensities are included in β^*. Thus a group variable selection method may be applied with a group consisting of all sub-spectra of a parent isotope.

All of the regression analysis thus far can be viewed as regression with p groups, each group consisting of exactly one element. Now consider each group to have K_j elements, $j = 1, ..., p$. The linear model (2.14) can be written as

$$Y_n = \sum_{j=1}^{p} X_{jn}\beta_j^* + V_n \tag{2.22}$$

where $X_{jn} \in \Re^{n\times K_j}$ and β_j^* is a vector of length K_j. Furthermore, denote $||\nu||_D = \sqrt{\nu^T D\nu}$ for a vector ν of length K_j and a symmetric matrix $D > 0$. Assume the response has had its mean subtracted off and that each X_{jn} is orthonormalized. Then, given matrices $D_1, ..., D_p > 0$, the group LASSO [42] estimate is the estimate that minimizes

$$J(\beta) = \frac{1}{2}||Y_n - \sum_{j=1}^{p} X_{jn}\beta_j||_2^2 + t\sum_{j=1}^{p}||\beta_j||_{D_j}. \tag{2.23}$$

If D_j's are identity matrices of dimension K_j, the solution to (2.23) has the components

$$\hat{\beta}_j = \Big(1 - \frac{t}{||S_j||_I}\Big)^+ S_j,$$

where $S_j = X_{jn}^T(Y_n - X_n\hat{\beta}_{-j})$, with $\hat{\beta}_{-j} = (\hat{\beta}_1^T, ..., \hat{\beta}_{j-1}^T, 0^T, \hat{\beta}_{j+1}^T, ..., \hat{\beta}_p^T)^T$.

The penalty function in (2.23) is an intermediate one between the L_1 penalty used in LASSO and the L_2 penalty used in other regressions (e.g. ridge regression). This is illustrated in Figure 2.10 [42] where the D_j's are identity matrices. Note that use of the L_2 penalty in regularized linear regression problems is known to yield all non-zero estimates. Consider the case when there are $p = 2$ groups. The first group, $\beta_1 = (\beta_{11}, \beta_{12})^T$, is of length two. The second group is of length one and is denoted as β_2. The top row in Figure 2.10 shows the contours of the penalty functions: the left corresponds to the L_1 penalty $|\beta_{11}| + |\beta_{12}| + |\beta_2| = 1$, the center corresponds to the group LASSO penalty $||\beta_1||_2 + |\beta_2| = 1$, and the right corresponds to the L_2 penalty $||(\beta_1^T, \beta_2)^T||_2 = 1$. Cross sections are shown in the last three rows. As shown in Figure 2.10, the L_1 penalty treats the three coordinate directions differently from other directions, and this encourages sparsity. The L_2 penalty treats all directions equally and does not encourage sparsity. The group LASSO encourages spareness at the group level.

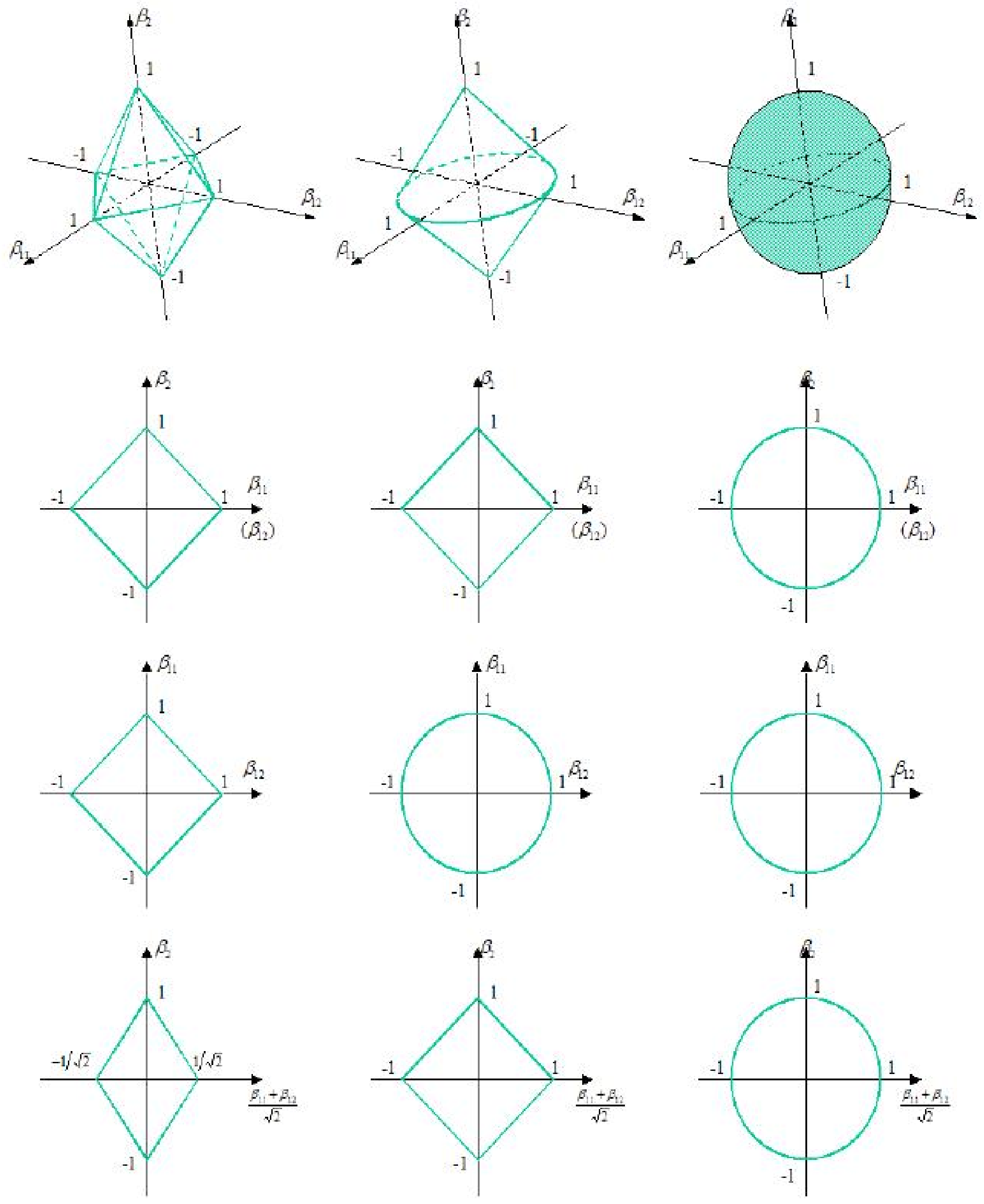


Figure 2.10: The L_1 penalty (left), the group LASSO penalty (center) and the L_2 penalty (right).

2.3.4.3 Positive LASSO

In nuclear material detection, where the unknown intensities cannot possibly be negative, the performance of LASSO can be much improved by incorporating the a priori information $\beta_j^* \geq 0$ directly into the LASSO minimization, resulting in the positive LASSO [9]:

$$J(\beta) = \min_{\beta \geq 0} \left\{ ||Y_n - X_n\beta||_2^2 + t\sum_{j=1}^{p} \beta_j \right\} \tag{2.24}$$

where $\beta \geq 0$ stands for $\beta_j \geq 0,\ j = 1, ..., p$. The positive LASSO is briefly mentioned in [9] where it is explained how to modify the LARS algorithm to conveniently solve the positive LASSO problem for all t, though the authors fail to work through an example. In fact, though the positive LASSO (2.24) can be solved efficiently, it has yet to be thoroughly explored.

2.3.4.4 Non-Negative Garrote

Another technique for variable selection, the non-negative garrote, produces a scaled version of the least squares estimate [43]. The scaling factor $d = (d_1, ..., d_p)^T$ is given as the minimizer to

$$||Y_n - Zd||_2^2 + t\sum_{j=1}^{p} d_j,\ d_j \geq 0,\ j = 1, ..., p,$$

where $Z = (Z_1, ..., Z_p)$, $Z_j = X_{jn}\hat{\beta}_{LSj}$. The non-negative garrote estimate is then $\hat{\beta}_j^{NG} = d_j\hat{\beta}_{LS_j}$, for $j = 1, ..., p$. In the orthogonal case when $X_n^TX_n = I$, the components of the scaling vector d have the explicit form

$$d_j = \left(1 - \frac{t}{\hat{\beta}_{LS_j}^2}\right)^+,\ j = 1, ..., p.$$

Solving the non-negative garrote for every value of $t \geq 0$ may be done efficiently by modifying the LARS algorithm. A concern about the nonegative garrote is its explicit reliance on the full least squares estimate. With a small sample size, the least squares may perform poorly, and the non-negative garrote will suffer too.

2.3.5 Determining the Model Dimension

As we have seen, the LARS algorithm is able to produce the entire solution path of LASSO for every value of t in (2.18), see Figure 2.9. The variables are introduced one at a time until all variables are non-zero and the least squares estimate is obtained. Consider a correct solution path to be one in which the LASSO profile generated by LARS contains the true underlying model. That is to say, if the true underlying model is $A^* = \{j \ : \ \beta_j^* \neq 0\}$, then the first variables to be introduced by LARS are of the set A^*. Then, in the presence of a correct solution path, the best choice of the regularization parameter t_{opt} would be one that hopefully generates an estimated set $A = A^*$. Further, t_{opt} could change with the number of data points n. If the regularization parameter were to be held constant as the number of data points increased, the first term in (2.18) would dominate, and the LASSO solution would be very close to that of the least squares estimate. If t is too large, the estimate gets forced to all zero components. Thus, the selection of the regularization parameter is crucial to LASSO's success. The same argument can be made for adaptive LASSO, group LASSO, positive LASSO and the non-negative garrote.

Cross validation or an information-based criterion like AIC or BIC [36] may be effective in determining t_{opt}. The former is a technique that is used to determine how well a model will generalize to an independent data set, while the latter is a measure of the goodness of fit of a model. As illustrated in Figure 2.11, M-fold cross validation is the process of partitioning the data set into M data sets, where one of the sets is used for testing and the others are used for training. Modeling prediction errors are calculated from the test data, and the process is repeated M times with a new training set each time. The results are averaged together, and the t corresponding to the model with the smallest average cross validation prediction error is selected as t_{opt}.

Specifically, let the partitioned data set pairs be represented as $(X_n^{(1)}, Y_n^{(1)})$, $(X_n^{(2)}, Y_n^{(2)})$,..., $(X_n^{(M)}, Y_n^{(M)})$. Let the pair $(X_n^{(-k)}, Y_n^{(-k)})$ be the regressor matrix X_n with data $X_n^{(k)}$ removed and the output sequence Y_n with data $Y_n^{(k)}$ removed, respectively. For a given t, the generated LASSO estimate with the kth partition removed is

$$\hat{\beta}^{(-k)} = \min_{\beta} \left\{ ||Y_n^{(-k)} - X_n^{(-k)}\beta||_2^2 + t\sum_{j=1}^{p} |\beta_j| \right\}.$$

Then the estimate $\hat{\beta}^{(-k)}$ produces the cross validation prediction error

$$CV_k(t) = ||Y_n^{(k)} - X_n^{(k)}\hat{\beta}^{(-k)}||_2^2, \; k = 1, ..., M.$$

The best t is the one that minimizes the average cross validation prediction error

$$t_{opt} = arg \min_{t} \frac{1}{M} \sum_{k=1}^{M} CV_k(t).$$

The same analysis applies to LASSO's variants and the non-negative garrote.

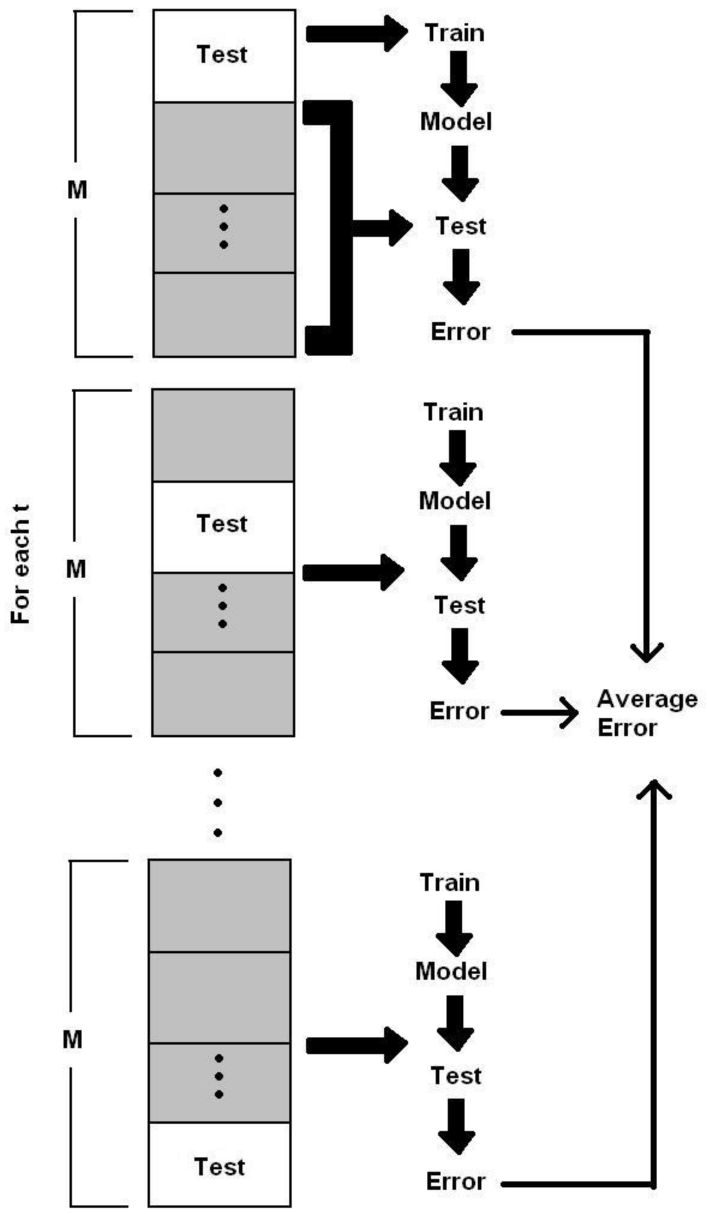


Figure 2.11: M-fold cross validation for determining the regularization parameter t. Each t leads to an average cross validation prediction error. Choose the t that minimizes this error.

AIC and BIC may also be used to determine the true dimension of the underlying model. For a given t, suppose the estimate generated by LASSO or the non-negative garrote is $\hat{\beta}$ and the number of non-zero components of that estimate is q. The best t according to AIC and BIC is

$$t_{opt} = arg\min_{t} \left\{ n \cdot \ln \frac{||Y_n - X_n\hat{\beta}||_2^2}{n} + \alpha q \right\} \tag{2.25}$$

where $\alpha = 2$ for AIC and $\alpha = \ln(n)$ for BIC [36]. BIC is consistent in the sense that it was designed to identify the true dimensionality of the underlying model [41]. AIC is not consistent but has lower mean squared error [41]. Therefore, one must make a decision between consistency and minimizing prediction errors.

CHAPTER 3
DETECTION AS LINEAR REGRESSION AND THE LASSO APPROACH

3.1 Modeling Gamma-Ray Emission

Radioactive sources randomly emit gamma-rays, and a Poisson distribution characterizes the number of gamma-ray counts, k, registered by a detector per unit time and per unit amount of source material [5] according to $((\lambda)^k)/k!)e^{-\lambda}$. The distribution parameter λ mainly depends on the type of radioactive decay of the source, the characteristics of the detector material, and some environmental effects such as temperature and the position of the detector relative to the source. When β^* units of radioactive source material are present, the number of counts continues to follow a Poisson distribution

$$z \sim ((\lambda\beta^*)^k)/k!)e^{-\lambda\beta^*}.$$

The source material is considered absent when $\beta^* = 0$. A convenient property of Poisson random variables ensures the mean $E(z)$ be equal to the variance $E((z - \lambda\beta^*)^2)$, where E is the expectation operator. The mean and variance of z is $\lambda\beta^*$. When more than one radioactive source is present, the number of counts recorded by the detector is the sum of the contributions from all of the sources, including the background.

When p radioactive sources are present, each modeled as a Poisson random variable

$$((\lambda_i\beta_i^*)^k)/k!)e^{-\lambda_i\beta_i^*},\ i = 1, ..., p,$$

it may be reasonably assumed that the contributions of each individual source is statistically independent. Then, yet another convenient property of Poisson random variables [3] suggests that the total gamma-ray count is also Poisson distributed

$$z \sim \frac{(\sum_{i=1}^{p} \lambda_i \beta_i^*)^k}{k!} e^{-\sum_{i=1}^{p} \lambda_i \beta_i^*}, \tag{3.1}$$

where the mean and variance of (3.1) are both $\sum_{i=1}^{p} \lambda_i \beta_i^*$.

The above analysis applies to each range of energy (keV) of the gamma-ray spectra. If a detector has n energy channels, then the gamma-ray counts $Z = (z_1, z_2, ...z_n)'$ is a random vector, and the gamma-ray counts in the ith channel, z_i, follows a Poisson distribution

$$z_i \sim \frac{(\lambda_{i1}\beta_1^* + \lambda_{i2}\beta_2^* + ... + \lambda_{ip}\beta_p^*)^k}{k!} e^{-(\lambda_{i1}\beta_1^* + \lambda_{i2}\beta_2^* + ... + \lambda_{ip}\beta_p^*)}, \; i = 1, 2, ..., n \tag{3.2}$$

where the vector $(\lambda_{1i}, \lambda_{2i}, ..., \lambda_{ni})^T$ is the spectrum of the ith radioactive source.

Define

$$\lambda = \begin{pmatrix} \lambda_{11} & \dots & \lambda_{1p} \\ \vdots & \ddots & \vdots \\ \lambda_{n1} & \dots & \lambda_{np} \end{pmatrix}, \; \beta^* = \begin{pmatrix} \beta_1^* \\ \vdots \\ \beta_p^* \end{pmatrix} \tag{3.3}$$

where $(\lambda_{i1}\beta_1^* + \lambda_{i2}\beta_2^* + ... + \lambda_{ip}\beta_p^*)$ is the mean gamma-ray counts per unit time received by the detector at the *ith* spectrum channel from all radioactive sources. With a slight abuse of notation, the random gamma-ray counts vector Z may be expressed as

$$Z = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix} \sim \frac{(\lambda\beta^*)^k e^{-\lambda\beta^*}}{k!}$$

where $e^{-\lambda\beta^*}$ and $(\lambda\beta^*)^k$ are calculated component-wise. When the numbers of gamma-ray counts at different channels are independent the mean value and the variance of

Z are

$$EZ = \lambda\beta^* = \begin{pmatrix} \lambda_{11} & \dots & \lambda_{1p} \\ \vdots & \ddots & \vdots \\ \lambda_{n1} & \dots & \lambda_{np} \end{pmatrix} \begin{pmatrix} \beta_1^* \\ \vdots \\ \beta_p^* \end{pmatrix}$$

and

$$E(Z-\lambda\beta^*)(Z-\lambda\beta^*)^T = \begin{pmatrix} \lambda_{11}\beta_1^* + ... + \lambda_{1p}\beta_p^* & 0 & \dots & 0 \\ 0 & \lambda_{21}\beta_1^* + ... + \lambda_{2p}\beta_p^* & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_{n1}\beta_1^* + ... + \lambda_{np}\beta_p^* \end{pmatrix}.$$

Let Z_i, $i = 1, 2, ..., l$, be l independent observations of Z. Define the empirical average of Z by

$$\bar{Z} = \frac{1}{l}\sum_{i=1}^{l} Z_i$$

that satisfies

$$\bar{Z} = \lambda\beta^* + \underbrace{\frac{1}{l}\sum_{i=1}^{l} Z_i - EZ}_{\bar{V}} = \lambda\beta^* + \bar{V} \tag{3.4}$$

which is cast in a linear regression form and can be used to estimate the strength coefficients vector β^* where $\bar{Z}$ is available from measurements, the relative magnitude matrix λ is known a priori for given radioactive sources, and $\bar{V}$ is a random variable which can be considered as noise.

It is easy to verify the mean and variance of $\bar{V}$ as

$$E\bar{V} = \frac{1}{l}\sum_{i=1}^{l} EZ^i - EZ = 0,$$

and

$$E\bar{V}\bar{V}^T = \frac{1}{l}\sum_{i=1}^{l} E(Z^i - \lambda\beta^*)(Z^i - \lambda\beta^*)^T$$

$$= \frac{1}{l}\begin{pmatrix} \lambda_{11}\beta_1^* + ... + \lambda_{1p}\beta_p^* & 0 & \dots & 0 \\ 0 & \lambda_{21}\beta_1^* + ... + \lambda_{2p}\beta_p^* & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_{n1}\beta_1^* + ... + \lambda_{np}\beta_p^* \end{pmatrix}.$$

Notice that $\sum_{i=1}^{l} Z^i$ is also Poisson. If the gamma-ray counts are high, the Poisson $\sum_{i=1}^{l} Z^i$ is very close to a Gaussian distribution [3]. In other words, the noise $\bar{V}$ is close to a multivariate Gaussian of zero mean. Moreover, by the properties of the Gaussian distribution, the components of $\bar{V}$ are independent if $\bar{V}$ are Gaussian.

For detection, equation (3.4) could be used to determine the unknown source strength coefficient vector β^* where $\bar{Z}$ is available from detector measurements and λ can be determined a priori. Detection now is to find out which β_i^*'s are zero and which are not; $\beta_i^* = 0$ indicates that the *ith* radioactive source or material is absent, otherwise, the material is considered to be present. In general, the available background radiation information can and will be included in λ and β^*. Although the components of the noise $\bar{V}$ are of zero mean and almost Gaussian and independent, their variances are not identical, and having i.i.d. noise is preferable. If $\lambda\beta^*$ were available, we could scale $\bar{Z}$ by

$$\begin{pmatrix} \frac{1}{\sqrt{\lambda_{11}\beta_1^*+\ldots+\lambda_{1p}\beta_p^*}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\sqrt{\lambda_{n1}\beta_1^*+\ldots+\lambda_{np}\beta_p^*}} \end{pmatrix} \bar{Z} = \begin{pmatrix} \frac{1}{\sqrt{\lambda_{11}\beta_1^*+\ldots+\lambda_{1p}\beta_p^*}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\sqrt{\lambda_{n1}\beta_1^*+\ldots+\lambda_{np}\beta_p^*}} \end{pmatrix} \lambda\beta^*$$

$$+ \begin{pmatrix} \frac{1}{\sqrt{\lambda_{11}\beta_1^*+\ldots+\lambda_{1p}\beta_p^*}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\sqrt{\lambda_{n1}\beta_1^*+\ldots+\lambda_{np}\beta_p^*}} \end{pmatrix} \bar{V}$$

and note that the noise $\begin{pmatrix} \frac{1}{\sqrt{\lambda_{11}\beta_1^*+\ldots+\lambda_{1p}\beta_p^*}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\sqrt{\lambda_{n1}\beta_1^*+\ldots+\lambda_{np}\beta_p^*}} \end{pmatrix} \bar{V}$ would have identical variance for each component. Though $\lambda\beta^*$ is unobtainable, its estimate $\bar{Z}$ is however available. To reduce the effect of non-constant variance, equation (3.4) is scaled by

$$\underbrace{\begin{pmatrix} \frac{1}{\sqrt{\bar{z}_1}} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \frac{1}{\sqrt{\bar{z}_n}} \end{pmatrix} \begin{pmatrix} \bar{z}_1 \\ \vdots \\ \bar{z}_n \end{pmatrix}}_{Y_n} = \underbrace{\begin{pmatrix} \frac{1}{\sqrt{\bar{z}_1}} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \frac{1}{\sqrt{\bar{z}_n}} \end{pmatrix} \lambda}_{X_n} \beta^* + \underbrace{\begin{pmatrix} \frac{1}{\sqrt{\bar{z}_1}} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \frac{1}{\sqrt{\bar{z}_n}} \end{pmatrix} \bar{V}}_{V_n},$$

or simply

$$Y_n = X_n\beta^* + V_n$$

where

$$Y_n = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \; X_n = \begin{pmatrix} x_{11} & \dots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix} = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix}, \; \beta^* = \begin{pmatrix} \beta_1^* \\ \vdots \\ \beta_p^* \end{pmatrix}, \; V_n = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix},$$

which is exactly equation (2.14).

Y_n and X_n are available, as they are scaled versions of the gamma-ray counts $\bar{Z}$ and λ, respectively. V_n is noise of zero mean and almost i.i.d. Gaussian components. Equation (2.14) is the basic equation for detection.

3.2 Peak Detection Performance

Several versions of peak detection algorithms exist, but they all rely on the same basic steps: locate peaks; fit each peak to a Gaussian function with some width, position, and height; and then match the energy of the photo peak to a list of energies in a nuclide library [22]. The key is a peak location algorithm that has some advanced smoothing functions with varying smooth windows to combat data fluctuations due to noise, as discussed in Section 2.2.1. The size of the smoothing window serves the purpose of balancing the peak detection sensitivity and the sensitivity to the noise.

If signals are strong and high-resolution detectors can be used, gamma-ray signature recognition is straightforward by peak detection algorithms [28]. However, in a number of applications in which the detectors cannot reasonably be expected to be close to the sources, the detectors generally have large surface area and poor resolution. The signals from these detectors will be weak and difficult to separate from the background radiation or from the signatures of commonly used radioactive materials. In such cases, peak detection algorithms do not work well, as illustrated in the next example.

Consider a semi-real experiment where a germanium type of detector with $n = 1024$ channels is used and the background is a real measurement which consists primarily of trace amounts of radiation from local sources and cosmic rays. The gamma-ray counts of the 11 isotopes shown in Table 3.1 are synthetically generated according to nuclear physics posted on the web-site of the National Institute of Standards and Technology or equivalently by the handbook of [22]. The background is considered as the 12th isotope in the simulation. For a germanium type of detector capable of resolving spectra to 1024 channels, the mean gamma-ray counts for the isotopes in Table 3.1 are well documented which provides λ in equation (3.4) as shown in Figure 3.1.

Figure 3.2 illustrates the scheme under which detection performances will be tested. In simulation, the coefficients β^*_{12} of the background and β^*_9 of I131 are set to be one, and the coefficient β^*_1 for Pu239 is equal to 0.2. All of the other β^*_i's are zero. Simply put, the background and I131 are present and dominant. Pu239, a special nuclear material, is also present, but its low energy lines are hiding in the low energy lines of the more dominant I131. The top diagram of Figure 3.3 shows

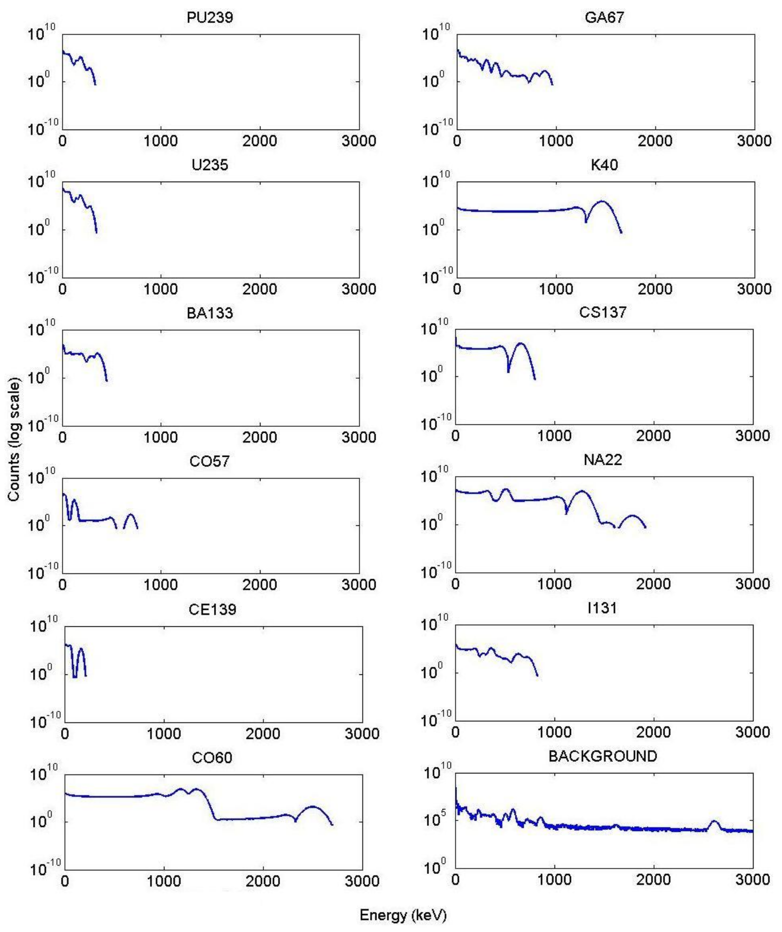


Figure 3.1: Gamma-ray emission spectra of the 12 isotopes used in testing as resolved by a germanium type of detector with 1024 channels.

the mean gamma-ray counts of the background+I131 per unit time for channels one through 1024 and the mean gamma-ray counts of the background+I131+0.2Pu239. The addition of Pu239 is hardly visible, as the plutonium barely contributes to the observed gamma-ray spectrum. The bottom diagram of Figure 3.3 is a close-up one together with the mean gamma-ray counts of Pu239 per unit time. The signal to noise (background) ratio is calculated by summing up the energy at all frequencies

Table 3.1: The 11 isotopes involved in testing of a semi-real experiment (background radiation is considered the 12th isotope).

	Isotope	Half-life	Notes
1	Pu239	24,000 years	special nuclear material
2	Ga67	3 days	uses include medical imaging
3	Cs137	30 years	sometimes used in cancer treatments
4	U235	7×10^8 years	special nuclear material
5	K40	1×10^9 years	used in potassium-argon dating
6	Na22	3 years	used for instrument calibration
7	Ba133	10 years	used as a radioactive tracer
8	Ce139	138 days	used for instrument calibration
9	I131	8 days	treatment for hyperthyroidism
10	Co57	272 days	used for instrument calibration
11	Co60	5 years	disinfectant of surgical equipment
12	Background	n/a	NORM and cosmic rays

by varying α,

$$SNR(\alpha) = 10 \log \frac{\sum_{i=1}^{1024} \alpha(0.2Pu239(i) + I131(i))}{\sum_{i=1}^{1024} Background(i)}. \tag{3.5}$$

The detection problem is to find which β_i^* are non-zero or equivalently which isotopes are present. It is obvious that all the signature peaks generated by Pu239 are buried in the background or in I131 and are invisible which implies that traditional peak detection methods will not work well. To test, I used an advanced peak detection algorithm developed at the University of Maryland [30] called findpeaks.m in MATLAB. Figure 3.4 shows the result of traditional peak based detection methods at an SNR of -10 dB. The top panel in Figure 3.4 shows the peaks detected (black circles) when Pu239 is absent, and the bottom panel shows the peaks detected when Pu239 is present. The peaks detected with and without Pu239 remain the same. This

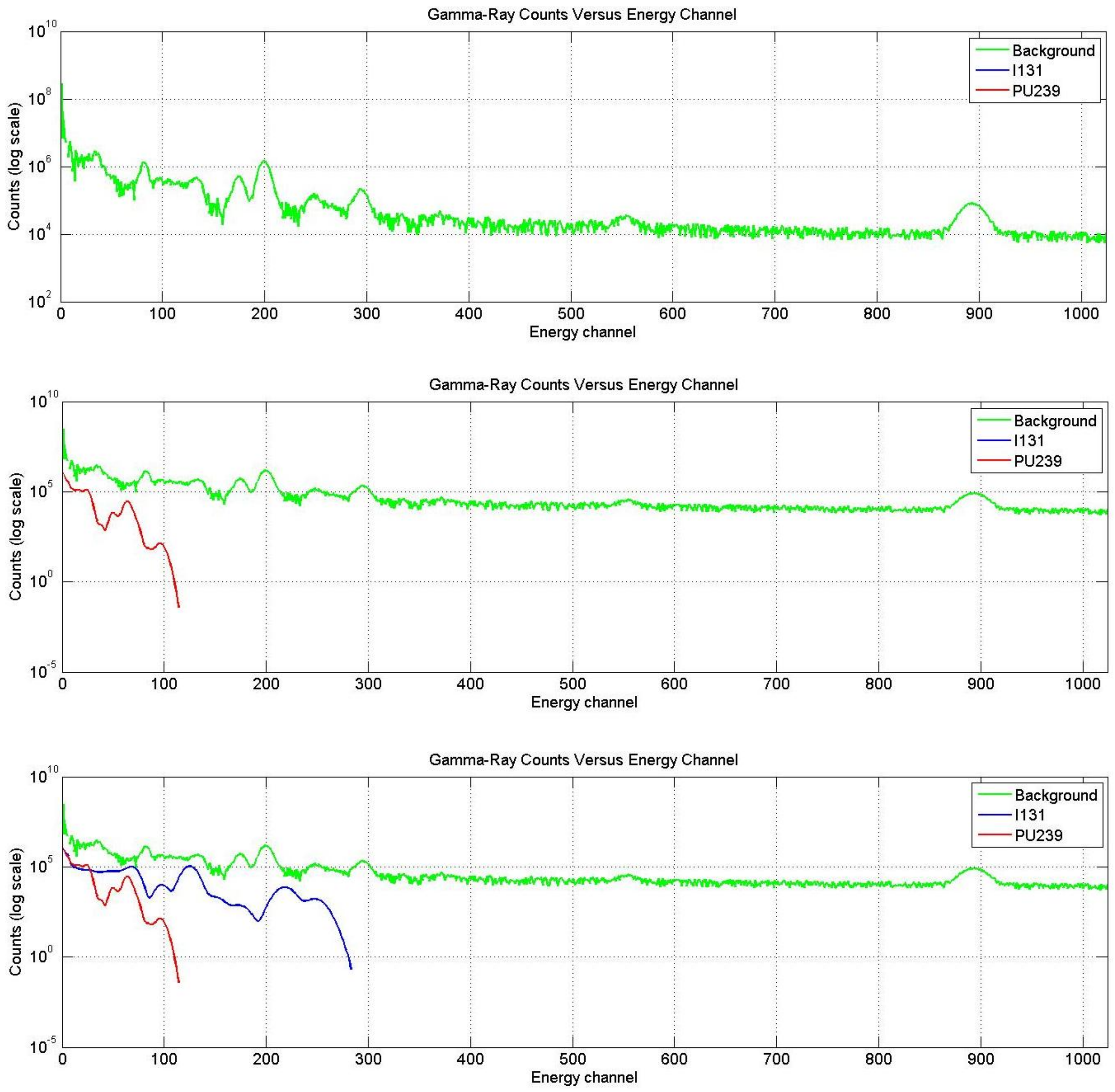
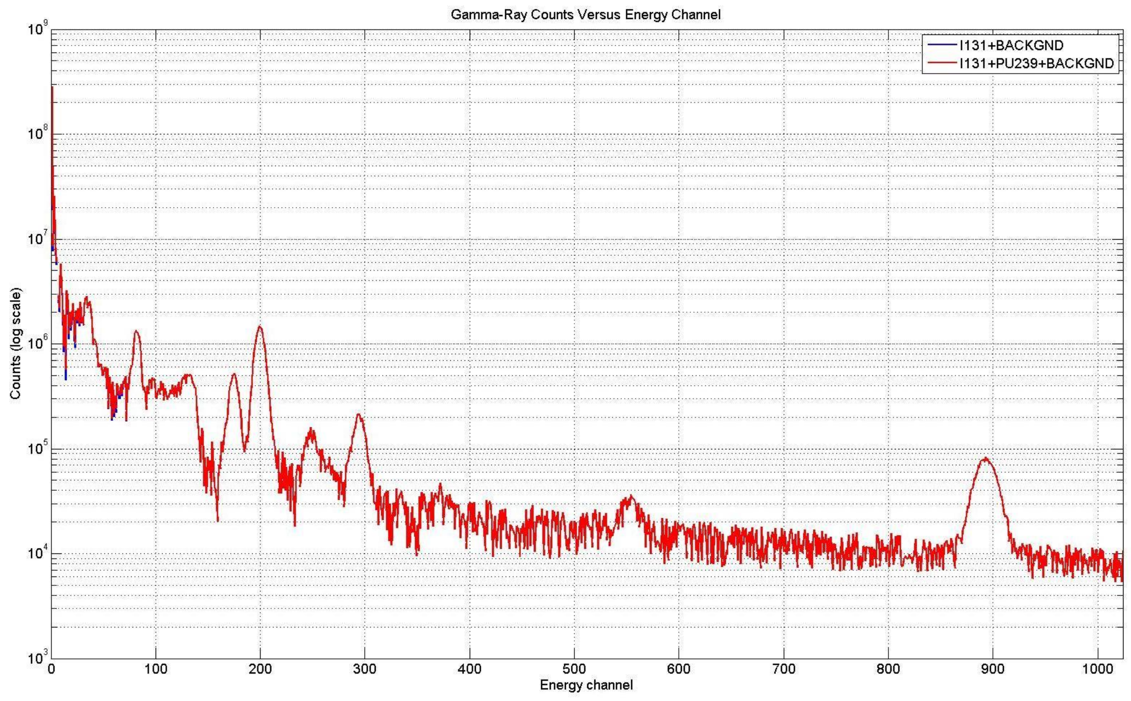


Figure 3.2: The testing scheme is as follows: A relatively dominant background is established (top panel). Then, a trace amount of the special nuclear material Pu239 is introduced (middle panel). Finally, the addition of I131, which has a low energy signature, attempts to mask the low energy signature of Pu239 (bottom panel).



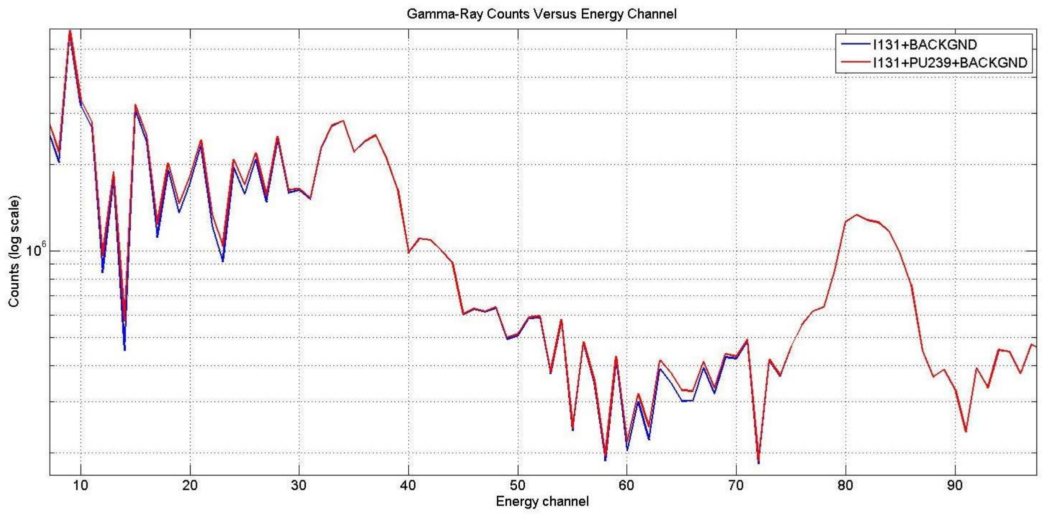


Figure 3.3: A visualization of just how little the addition of Pu239 contributes to the overall received gamma-ray spectrum. The received spectrum of I131 + backgound shown together with the received spectrum of I131 + background + Pu239 (top). The two spectra seem almost indistinguishable until a closer inspection of the low energy channels is given (bottom).

implies the traditional methods completely fail to find Pu239 at an SNR of -10 dB in this experiment. The simulation is repeated with a less sensitive peak detector, shown in Figure 3.5), but the conclusion is the same.

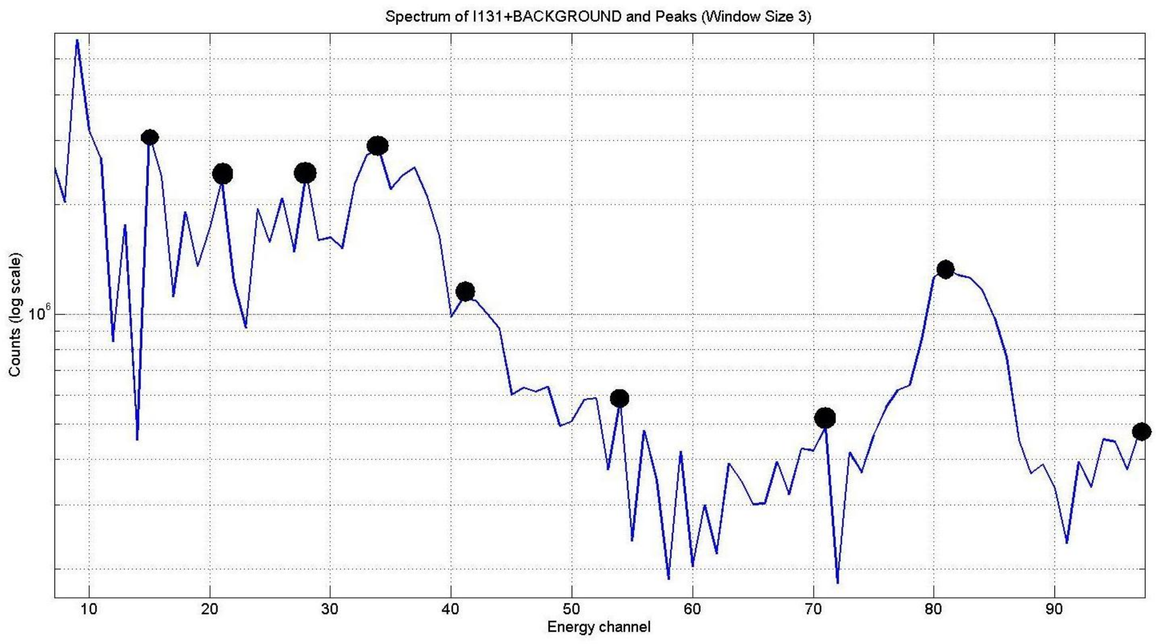
3.3 LASSO Detection Performance

To overcome the problem of detecting invisible peaks, variable selection methods may be used based on model (2.14). Recall that the primary goal of detection is not to estimate the magnitude of each component β_i^*, $i = 1, 2, ..., p$, but to find out if β_i^* is zero or not – referred to as variable selection in the literature.

Equation (2.14) is in a standard linear equation form for the unknown vector β^*. Let $\hat{\beta}$ denote the estimate of β^*. It is well known that the least squares estimate (2.17) does not solve the problem of variable selection. Instead, I consider the LASSO (2.18) due to its convenience and simplicity, as well as its range of applicability.

For the LASSO method, the choice of the tuning parameter t is critical. In general, the choice of t is based on the well established AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) [36]. To emphasize the dependence on t, write the estimate derived from equation (2.18) for a given t as $\hat{\beta}(t)$. For each t and $\hat{\beta}(t)$, let $q(t)$ be the number of non-zero components of $\hat{\beta}(t)$. If no component of $\hat{\beta}(t)$ is zero, $q(t) = p$, the dimension of β^*, and $q(t) = 0$ if all the components of $\hat{\beta}(t)$ are zero.

Using $\hat{\beta}(t)$ and $q(t)$, the optimal t according to the AIC or BIC is given by equation (2.25): The LASSO-AIC estimate is the estimate derived from equation (2.18)



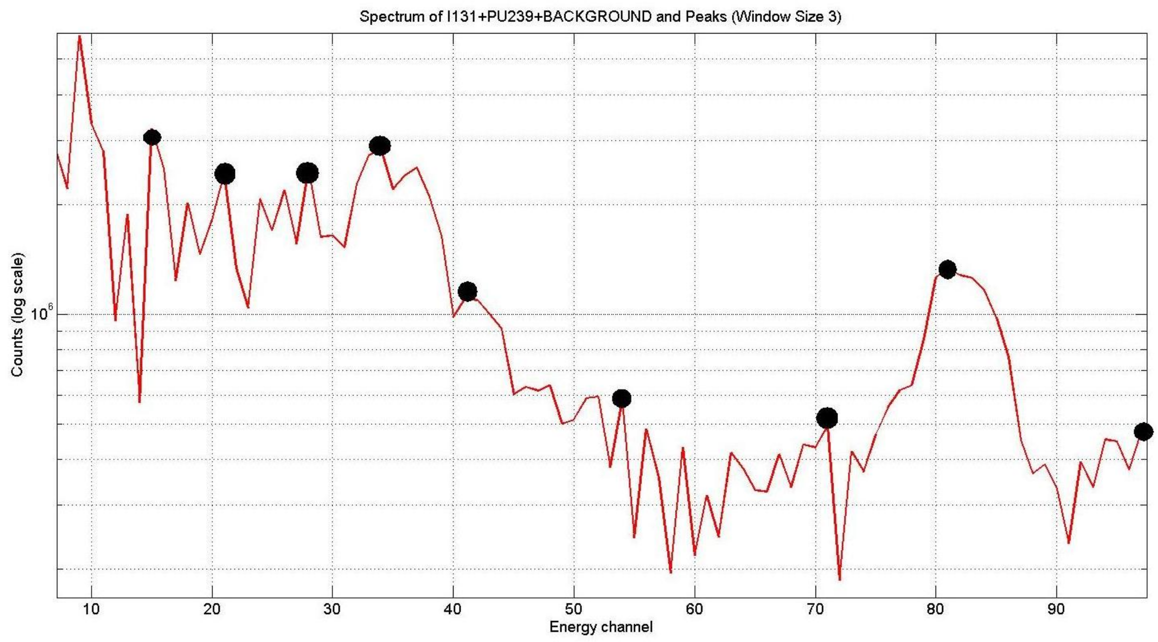
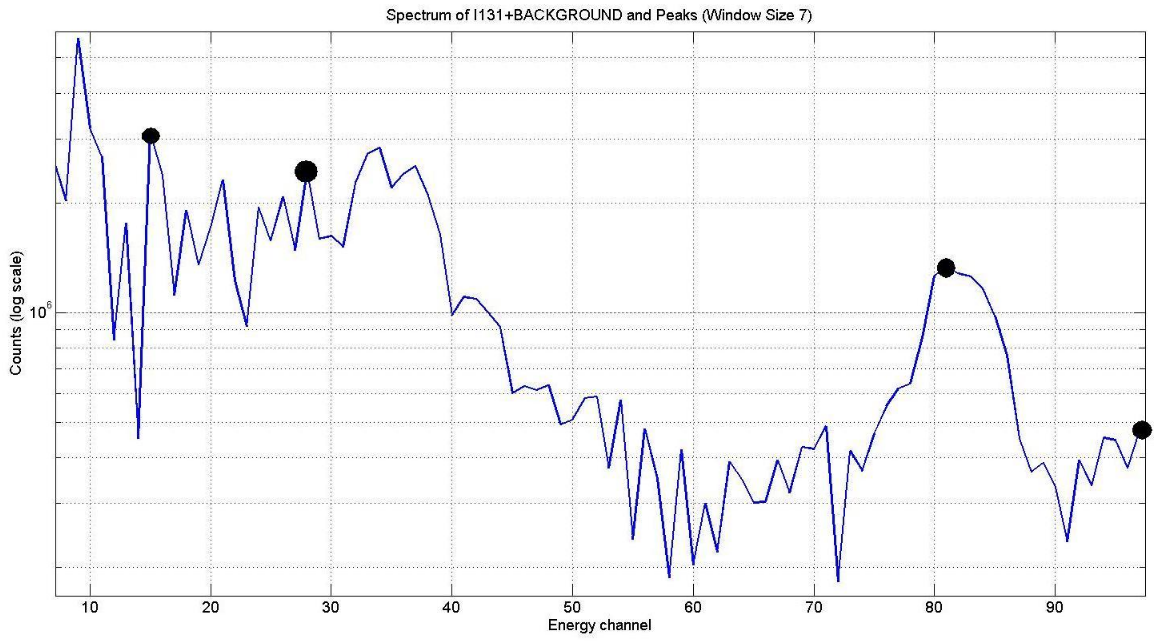


Figure 3.4: Top panel: The detected peaks in the received spectrum of I131 + background for a window size of three are shown as black circles. Introducing Pu239 (bottom panel) does not change the location or amount of detected peaks. This result suggests peak detection algorithms fail to find the Pu239, even with a modest SNR of -10 dB.



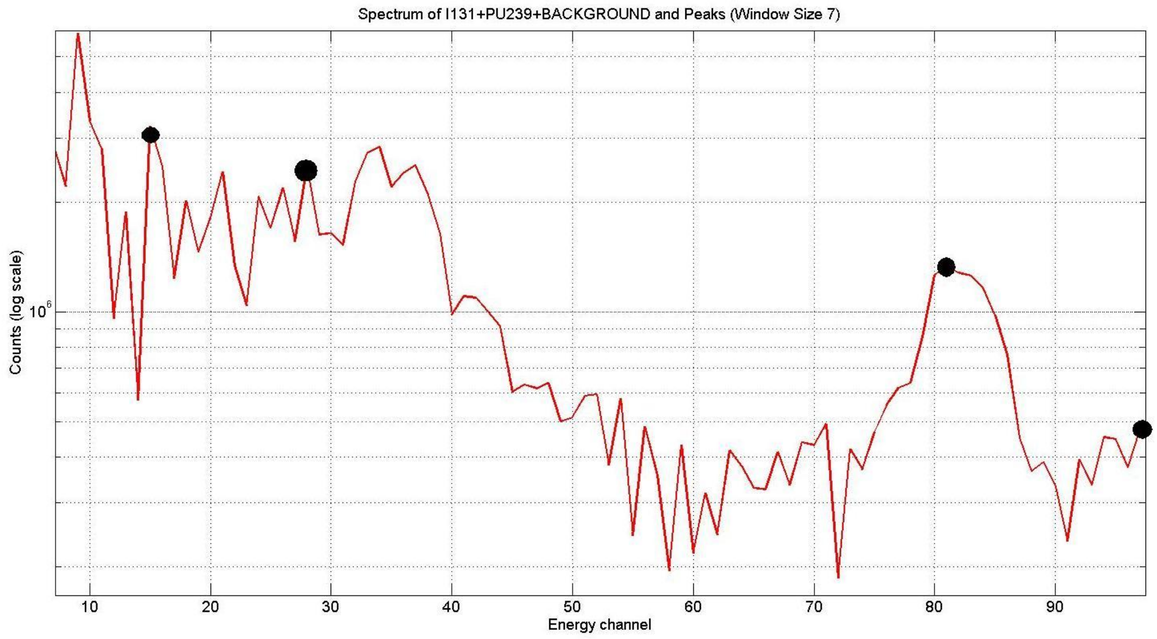


Figure 3.5: Top panel: The detected peaks in the received spectrum of I131 + background for a window size of seven are shown as black circles. Introducing Pu239 (bottom panel) does not change the location or amount of detected peaks. This result suggests peak detection algorithms fail to find the Pu239, even with a modest SNR of -10 dB.

when t is chosen from equation (2.25) for $\alpha = 2$ while the LASSO-BIC estimate is the one derived from equation (2.18) when t is chosen from equation (2.25) for $\alpha = \ln(n)$. Both AIC and BIC estimates can be efficiently calculated [47].

To illustrate the performances of LASSO-AIC and LASSO-BIC, consider the same simulation study discussed previously where the mean gamma-ray counts provide λ in equation (3.4) as shown in Figure 3.1 and further in equation (2.14) after scaling. The simulation is based on model (2.14) with $l = 5$ and $n = 1024$. In simulation, the number of gamma-ray counts are generated according to Poisson distribution (3.2) for each isotope and each spectrum channel to have Z and $\bar{Z}$. Then, λ and $\bar{Z}$ are scaled by $\bar{Z}$ resulting in X_n and Y_n as in equation (2.14). The resulting random noise acting on model (2.14) is V_n. As before, the coefficients β^*_{12} of the background and β^*_9 of I131 are set to be one, and the coefficient β^*_1 for Pu239 is equal to 0.2. All other β^*_i's are zero.

The LASSO-AIC and LASSO-BIC are implemented in MATLAB with one modification. In the application to nuclear material detection, the values of the unknown β^*_i's are always non-negative. If the ith isotope is present, the corresponding β^*_i can be small but is positive. If the *ith* isotope is absent, the corresponding β^*_i is zero but cannot be negative. Thus, in numerical simulations, if $\hat{\beta}_i$ of the LASSO-AIC or the LASSO-BIC estimate is negative, $\hat{\beta}_i$ is set to zero and the ith isotope is considered absent.

To quantify the performance of the estimates, I define two types of errors:

False negative: Isotope is present and the algorithm fails to find it.

False positive: Isotope is absent and the algorithm falsely identifies it.

The simulation consists of 10,000 Monte Carlo runs. Since for each run, I131, Pu239, and the background are always there and no other isotopes are present, the column labeled "present" in Table 3.2 shows the total numbers of appearances of each isotope in 10,000 runs. The last two columns of Table 3.2 show the number of times each isotope was identified by LASSO-AIC and LASSO-BIC. The corresponding error rates are shown in Table 3.3.

From the simulation results, it is obvious that both the LASSO-AIC and the LASSO-BIC are very good in identifying the radioactive sources, but their false positive rates are concerns. To reduce the false positive errors, it will be helpful to analyze the performance limitations of the LASSO estimates in the context of this nuclear material application.

Two questions are particularly interesting and important. First, for a given measurement data, does there always exist at least one tuning parameter t in equation (2.18) that provides the correct estimate $\hat{\beta}$? By correct, I mean the LASSO estimates $\hat{\beta}_i \neq 0$ if $\hat{\beta}_i^* \neq 0$ and $\beta_i = 0$ if β_i^* for all i. Secondly, if such an optimal t exists, does AIC or BIC always find the optimal t? Since the analysis is similar, I focus on AIC here.

The first question was studied in the literature [25], and the answer is no. In fact, the probability that no such t exists could be very high. To study the second question, consider a scalar equation

$$y(k) = \beta^* + v(k), \;\; k = 1, 2, ..., n \tag{3.6}$$

where $v(k) = \xi(k) - \lambda$ and $\xi(k)$'s are independently Poisson distributed with mean and variance λ. In other words, $v(k)$'s are i.i.d. with zero mean and identical variance λ. Following the same procedures as in [47], the LASSO estimate of equation (2.18) for a given t assumes the form of

$$\hat{\beta} = sign(\hat{\beta}_{LS})(|\hat{\beta}_{LS}| - \frac{t}{2n})^+$$

where $\hat{\beta}_{LS}$ is the least squares estimate of equation (2.17) with $X_n = 1$, $Y_n = y(k)$, and $V_n = v(k)$.

Two cases need to be calculated to find the optimal t based on AIC and the corresponding LASSO estimate $\hat{\beta}$: (1) $|\hat{\beta}_{LS}| \leq t/2n$ or $\hat{\beta} = 0$ and $q(t) = 0$ as in equation (2.25) which implies $\ln(s(t)/n) + \alpha(q(t)/n) = \ln \sum y(k)^2/n$, and (2) $|\hat{\beta}_{LS}| > t/2n$ or $|\hat{\beta}| > 0$, $q(t) = 1$ and

$$\min_{\beta} \left\{ \ln \sum \frac{(y(k) - \beta)^2}{n} + \frac{2}{n} \right\} = \ln \sum \frac{(y(k) - \hat{\beta}_{LS})^2}{n} + \frac{2}{n}.$$

Clearly, based on AIC, the LASSO estimate $\hat{\beta} = 0$ if

$$\ln \sum \frac{y(k)^2}{n} < \ln \sum \frac{(y(k) - \hat{\beta}_{LS})^2}{n} + \frac{2}{n}$$

and $|\hat{\beta}| > 0$ if

$$\ln \sum \frac{y(k)^2}{n} > \ln \sum \frac{(y(k) - \hat{\beta}_{LS})^2}{n} + \frac{2}{n}.$$

Now, suppose the actual $\beta^* = 0$, $\lambda = 10$, and $n = 1024$ in equation (3.6). The probability that the LASSO estimate based on AIC is not zero is

$$\Pr\{|\hat{\beta}| > 0\} = \Pr\left\{ \ln \sum \frac{y(k)^2}{n} > \ln \sum \frac{(y(k) - \hat{\beta}_{LS})^2}{n} + \frac{2}{n} \right\} \approx 0.16.$$

This demonstrates that even in the case of $\beta^* = 0$ or the nuclear material is absent and an optimal $t > 2n|\hat{\beta}_{LS}|$ exists that produces a correct estimate $\hat{\beta} = 0$, AIC gives rise to a tuning parameter t that results in incorrect estimates $|\hat{\beta}| > 0$ with a substantial probability. This explains why the false positive error of the LASSO method is so high. The above analysis clearly illustrates the problem of the LASSO, i.e., the LASSO or similar types of variable selection algorithms including the LARS, the Forward Stepwise, the Backward Stepwise and other variable selection methods [47] alone are unable to solve the radionuclide detection problem reliably and accurately.

3.4 Sub-Sampling with LASSO

On one hand, we want to utilize the nice property of the LASSO that produces very small false negative errors. On the other hand, we need to reduce or eliminate its large false positive errors. Observe that the problem with the false positive error rate is that though the *ith* isotope is absent, the LASSO-AIC or the LASSO-BIC estimate $\hat{\beta}_i$ is a random variable that could be positive for a particular experiment or a Poisson realization. Notice that individual classification methods are recently challenged by combined classification systems which often show better performance. A motivation for ensembles is that a combination of a decision made by individual classification produces powerful committee [21]. Based on the idea of majority voting, LASSO can be combined with an estimation of a tight interval in which the true but unknown β^* lies. To this end, the least squares (LS) estimate and the probability distribution associated with the LS estimate is useful. Although the LS estimate is not good for variable selection in general because it usually gives a small non-zero value even if the true value is zero, its asymptotical distribution does provide a way to estimate an interval in which the true β^* lies.

Let the LS estimate $\hat{\beta}_{LS}$ be defined as in equation (2.17). Its mean value and variance are given by

$$E(\hat{\beta}_{LS}) = \beta^*$$

$$E(\hat{\beta}_{LS} - \beta^*)(\hat{\beta}_{LS} - \beta^*)^T = \frac{\sigma^2}{n}(\frac{1}{n}X_n^T X_n)^{-1} = \frac{\sigma^2}{n}(\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T)^{-1}.$$

If $n \to \infty$, the asymptotic distribution of $\hat{\beta}_{LS}$ is given by [14] as

$$\sqrt{n}(\hat{\beta}_{LS} - \beta^*) \sim N\Big(0,\ \sigma^2(\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T)^{-1}\Big),$$

or for large n, $\hat{\beta}_{LS}$ is asymptotically Gaussian distributed

$$\sim N\Big(\beta^*,\ \frac{\sigma^2}{n}(\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T)^{-1}\Big)$$

and each component $\hat{\beta}_{LSi}$ obeys

$$\sim N\Big(\beta_i^*,\ \frac{\sigma^2}{n}(\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T)_{i,i}^{-1}\Big) \tag{3.7}$$

where $((1/n)\sum_{i=1}^{n} x_i x_i^T)^{-1}$ are computable and x_{ii} is the iith element of the matrix X_n.

Let *std* stand for the standard deviation of the distribution (3.7). By a property of the Gaussian distribution, with probability 0.999 or higher, $\beta_i^* - 3 \cdot std \leq \hat{\beta}_{LSi} \leq \beta_i^* + 3 \cdot std$, or equivalently

$$\hat{\beta}_{LSi} - 3 \cdot std \leq \beta_i^* \leq \hat{\beta}_{LSi} + 3 \cdot std.$$

Also let $\epsilon > 0$ be a small positive number. If

$$\hat{\beta}_{LSi} + 3 \cdot std < \epsilon$$

we can say with a high confidence that $\beta_i^* \leq 0$ of the ith isotope is absent independent of the results of LASSO-AIC or LASSO-BIC. This would provide a way to reduce the false positive rates.

The actual LS distribution is unknown, and only a large enough n ensures (3.7) is an appropriate approximation. For the application of nuclear material detection, n is fixed and may not be large. In such a case, the approximation performance cannot be guaranteed because no error bound is available. A different approach must be developed that possesses a bound for finite n. A sub-sampling approach may be used to find the LS distribution $F_{\hat{\beta}_{LS}}$. Rewrite equation (2.14) as

$$y_i = x_i^T \beta^* + v_i, \quad i = 1, 2, ..., n$$

and let N_s and m be two positive numbers such that

$$0 < N_s < n, \quad m = n - N_s + 1$$

Define

$$X(i) = \begin{pmatrix} x_i^T \\ \vdots \\ x_{i+N_s-1}^T \end{pmatrix}, \quad Y(i) = \begin{pmatrix} y_i \\ \vdots \\ y_{i+N_s-1} \end{pmatrix}, \quad V(i) = \begin{pmatrix} v_i \\ \vdots \\ v_{i+N_s-1} \end{pmatrix}, \quad i = 1, 2, ..., m.$$

Further, let $\hat{\beta}(i)_{LS}$ be the LS estimate of β^* based on the data $X(i)$ and $Y(i)$ of length N_s,

$$\hat{\beta}(i)_{LS} = \left(X^T(i)X(i)\right)^{-1} X^T(i)Y(i)$$

$$= \beta^* + \Big(\frac{1}{N_s}\sum_{j=i}^{i+N_s-1} x_i x_i^T\Big)^{-1}\Big(\frac{1}{N_s}\sum_{j=i}^{i+N_s-1} x_i v_i\Big), \quad i = 1, 2, ..., m.$$

Let $F_{\hat{\beta}_{N_s}}(b) = \Pr\{\hat{\beta}(i) \leq b\}$ be the probability distribution function of $\hat{\beta}(i)$, where the vector inequality is taken component-wise. The idea is to approximate $F_{\hat{\beta}_{LS}}$ of $\hat{\beta}$ by $F_{\hat{\beta}_{N_s}}$ of $\hat{\beta}(i)$.

Let

$$\mathbf{1}_{[\hat{\beta}(i)\leq b]} = \begin{cases} 1, & \hat{\beta}(i) \leq b \\ 0, & \hat{\beta}(i) > b \end{cases}$$

be the indicator function, and define the empirical probability function $\hat{F}_{\hat{\beta}_{N_s}}(b)$ as

$$\hat{F}_{\hat{\beta}_{N_s}}(b) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}_{[\hat{\beta}(i)\leq b]}$$

Then, the following theorem is in place:

Theorem 3.1 (Sub-Sampling Error Bound). *Assume the y_i's are stationary and $(X^T(i)X(i))^{-1}$ exists for each i. Then,*

1. If the v_i's are i.i.d., then for each m,

$$E\Big(\hat{F}_{\hat{\beta}_{N_s}}(b) - F_{\hat{\beta}_{N_s}}(b)\Big)^2 \leq \frac{12}{m}$$

2. If the v_i's are generated by a stable linear filter derived by i.i.d. noise, then

$$E\Big(\hat{F}_{\hat{\beta}_{N_s}}(b) - F_{\hat{\beta}_{N_s}}(b)\Big)^2 \leq 12\frac{N_s - 1}{n - N_s}\Big(2 - \frac{N_s}{n - N_s}\Big) + \frac{24}{n - N_s}\cdot\frac{\rho}{1-\rho}$$

for some $0 < \rho < 1$.

The proof of Theorem 3.1 can be found in Chapter 8.

Figure 3.6 demonstrates the improvement of the sub-sampling approach over the asymptotic approximation by showing the actual distribution of $\hat{\beta}_{LSi}$ and approximate distributions by asymptotic approximation and sub-sampling. Clearly, the distribution by sub-sampling is closer to the actual one than the one by asymptotic approximation.

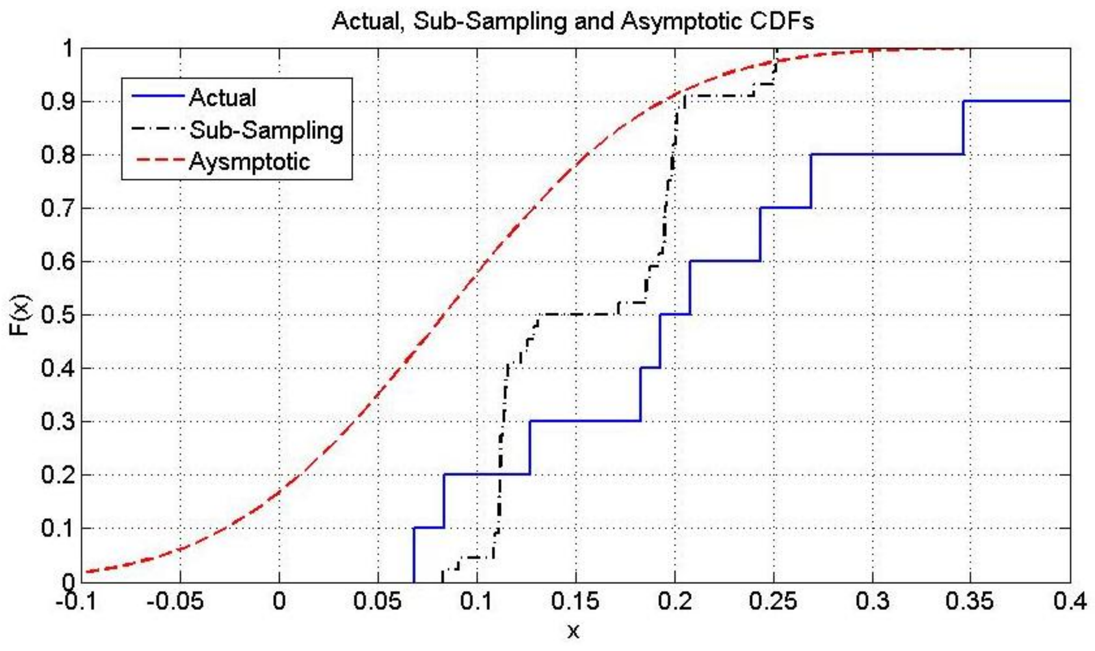


Figure 3.6: Actual distribution and approximate distributions by asymptotic and sub-sampling approaches.

Now, the error bound given by Theorem 3.1 is a non-asymptotic bound which provides an error quantification for modest n which is more useful than the asymptotic bound for this application. Notice what really is of interest, however, is the LS estimate distribution $F_{\hat{\beta}}$ of $\hat{\beta}$ that is the estimate by taking all the data $i = 1, 2, ..., n$. On the other hand, $F_{\hat{\beta}_{N_s}}$ is the LS distribution of $\hat{\beta}(i)$ by taking the data length $N_s < n$. Further, $F_{\hat{\beta}_{N_s}}$ is unknown and is approximated by $\hat{F}_{\hat{\beta}_{N_s}}$. Intuitively, when

N_s is close to n, $F_{\hat{\beta}_{N_s}}$ is close to $F_{\hat{\beta}}$. However, N_s close to n implies $m = n - N_s + 1$ is small and the error between $F_{\hat{\beta}_{N_s}}$ and its estimate $\hat{F}_{\hat{\beta}_{N_s}}$ is large. N_s and m have to be adjusted carefully.

Algorithm 3.1 LASSO with Sub-Sampling Algorithm

1. Given a detector and possible radioactive sources, construct the matrix λ as in equation (3.4).
2. Observe the number of gamma-ray counts and construct equation (2.14).
3. Apply the LASSO-AIC or the LASSO-BIC as discussed previously. Set $\hat{\beta}_i$ to zero if $\hat{\beta}_i < 0$.
4. Set the threshold $\epsilon > 0$ and $m,\ N_s$. Determine the sample distribution $F_{\hat{\beta}_{N_s}}$ for each $\hat{\beta}_i$ using the sub-sampling technique. Calculate the sample mean value $\hat{\mu}_i$ and the sample standard deviation $\hat{\sigma}_i$ from the sample distribution $F_{\hat{\beta}_{N_s}}$ for each $\hat{\beta}_i$.
5. For both the LASSO-AIC and the LASSO-BIC, if $\hat{\beta}_i > 0$ and $(\hat{\mu}_i + 3\hat{\sigma}_i) \geq \epsilon$, we say the *ith* isotope is present. Otherwise, the *ith* isotope is absent.

Now the detection algorithm is in place by combining the LASSO-AIC or LASSO-BIC with sub-sampling. Table 3.4 shows the results from 10,000 Monte Carlo simulations of the above detection algorithm with $\epsilon = 0.05$, $m = 44$ and $N_s = n-m =$

Table 3.2: The numbers of appearances and detections by the LASSO-AIC (LA) and the LASSO-BIC (LB) for each isotope in 10,000 simulations.

Isotope	Present	Detected (LA)	Detected (LB)
Pu239	10000	10000	10000
Ga67	0	6579	8467
Cs137	0	6728	7911
U235	0	9725	9792
K40	0	5607	6172
Na22	0	6101	7347
Ba133	0	6467	8534
Ce139	0	5791	7340
I131	10000	10000	10000
Co57	0	6113	7734
Co60	0	4054	2768
Background	10000	10000	10000

980 at an SNR of -10 dB, and Table 3.5 shows the results at different SNRs. By slightly sacrificing excellent false negative rates, the false positive rates have been drastically reduced from 0.6352 to 0.00078 for the LASSO-AIC and from 0.7341 to 0.00098 for the LASSO-BIC. At the same time the false negative rates remain very small at 0.0027 for both the LASSO-AIC and LASSO-BIC. Clearly, the combined algorithm of the LASSO and sub-sampling outperforms both the LASSO-AIC and the LASSO-BIC significantly.

An interesting question is how the performance of the combined algorithm is compared to that of sub-sampling only. To this end, the false negative and false positive errors of the sub-sampling algorithm from the same simulation with the same threshold ϵ are shown in the last row of Table 3.5. The combined algorithm improves the false negative error of the sub-sampling algorithm by approximately (0.0002/0.0027=) 7% and the false positive

Table 3.3: False negative and false positive detection error rates of LASSO-AIC and LASSO-BIC.

	False negative rate	False positive rate
LASSO-AIC	0.0000 (0/30000=0)	0.6352 (57165/90000=0.6352)
LASSO-BIC	0.0000 (0/30000=0)	0.7341 (66065/90000=0.7341)

Table 3.4: The numbers of appearances and detections of each isotope by the LASSO-AIC (LA) and the LASSO-BIC (LB) together with sub-sampling at an SNR of -10 dB.

Isotope	Present	Detected (LA)	Detected (LB)
Pu239	10000	9932	9932
Ga67	0	68	87
Cs137	0	0	0
U235	0	0	0
K40	0	0	0
Na22	0	0	0
Ba133	0	2	2
Ce139	0	0	0
I131	10000	10000	10000
Co57	0	0	0
Co60	0	0	0
Background	10000	10000	10000

Table 3.5: Detection errors of the two stage algorithms and of sub-sampling only at different SNRs.

	False negative rate				
Method	-15dB	-18 dB	-20dB	-22dB	-24dB
LASSO-AIC with sub-sampling	0.0000	0.0000	0.0027	0.0079	0.2170
LASSO-BIC with sub-sampling	0.0000	0.0000	0.0027	0.0079	0.2170
Sub-sampling only	0.0000	0.0000	0.0029	0.0091	0.2690
	False positive rate				
Method	-15dB	-18dB	-20dB	-22dB	-24dB
LASSO-AIC with sub-sampling	0.0007	0.0007	0.0008	0.0010	0.0078
LASSO-BIC with sub-sampling	0.0007	0.0009	0.0010	0.0012	0.0078
Sub-sampling only	0.0012	0.0010	0.0011	0.0024	0.0091

error by approximately (0.00032/0.00078=) 40%. The MATLAB code, PoiLandSub.m, for these simulations may be found in Section 9.7.

CHAPTER 4
REMOVING IRRELEVANT VARIABLES AMIDST LASSO ITERATIONS (RIVAL)

The number of detector channels n is, of course, finite, and as we have seen, this renders the asymptotical results of popular variable selection methods in Chapter 2 impractical. Though these results do provide a theoretical benchmark, it is desirable to have results that could apply to a large but fixed n. A new algorithm for detection, called positive RIVAL (removing irrelevant variables amidst LASSO iterations), is presented here and in [24] and is shown to be especially effective for such an n. The algorithm assumes that all unknown intensities are non-negative, and thus positive RIVAL is related to the positive LASSO (2.24). As such, a comprehensive analysis of positive LASSO is performed. Sufficient and necessary conditions for positive LASSO's asymptotic convergence are supplied, and an algorithm for solving the positive LASSO is included. Moreover, an adaptive positive LASSO is proposed and analyzed to have both parameter consistency (2.16) and set consistency (2.15) as $n \to \infty$. Positive RIVAL's ability of finding the correct model is highlighted in examples where it outperforms adaptive (positive) LASSO, non-negative garrote, and LARS. Because the non-negativity assumption is not crucial to the technical development of the positive RIVAL algorithm, this assumption is later relaxed so it may have a broader set of applications, and the more general RIVAL is tested against these same variable selectors.

Consider the positive LASSO with a weighting vector $w = (w_1, ..., w_p)^T$, $w_j \geq 0$, $j = 1, ..., p$, applied to the penalty term, i.e.,

$$J(\beta) = \min_{\beta \geq 0} \Big\{ ||Y_n - X_n\beta||_2^2 + t \sum_{j=1}^{p} w_j \beta_j \Big\}. \tag{4.1}$$

Positive RIVAL generates a sequence of weights $w(k) > 0$ and corresponding solutions to (4.1), $\hat{\beta}(k)$, and the hope is that $0 < \eta_1 \leq \hat{\beta}_j(k) \leq \eta_2 < \infty$, $j = 1, ...d$ and $\hat{\beta}_j(k) = 0$, $j = d+1, ..., p$ as $k \rightarrow \infty$. That is to say, the estimated set $A = A^*$ as $k \rightarrow \infty$. If n is allowed to be large, we may use the set generated by positive RIVAL to trim off the irrelevant regressors from X_n and the corresponding irrelevant variables from $\hat{\beta}$ to estimate the unknown intensities using the ordinary least squares estimate. Note that the iteration of $\hat{\beta}(k)$ is with respect to k while the number of data n is held fixed. This is completely different from the positive LASSO or the adaptive LASSO where only when a new observation is made or $n \rightarrow n+1$, a new $\hat{\beta}$ is recalculated. To fully understand positive RIVAL, we first need a comprehensive analysis of the positive LASSO.

4.1 Positive LASSO Algorithm

The weighted positive LASSO (4.1) can be numerically calculated by modifying the LARS algorithm discussed in Section 2.3.3. The modified algorithm is very efficient but sensitive in terms of identifying the correct variables if the regressors are highly correlated, which is the case in our nuclear material detection problem. The algorithm works by identifying the regressor with the most correlation with the output. It introduces the corresponding variable and increases its magnitude only until a new regressor has as much correlation with the current residual. The algorithm then introduces the second variable, and so on. If two regressors are highly correlated and the problem is very noisy, the algorithm may find a wrong regressor in which to proceed, thus giving an incorrect solution path. To avoid this problem, Algorithm 4.1, a coordinate descent-type of algorithm, is proposed. Later, this algorithm is compared with the modified LARS algorithm. The MATLAB code pLasso.m implements Algorithm 4.1 and may be found in Section 9.5.

Algorithm 4.1 Weighted Positive LASSO Coordinate Descent Algorithm

```
1. Set $k = 0$.
2. At each $k$, let $\gamma(k) = \hat{\beta}(k) = (\hat{\beta}_1(k), ..., \hat{\beta}_p(k))^T$. Set $l = 1$.
   2.1. For each $l$, calculate $\gamma_l = \max(\frac{C_l - \sum_{j \neq l} q_{lj}\gamma_j(k)}{q_{ll}}, 0)$.
   2.2. Let $\gamma_l(k) = \gamma_l$. If $l < p$, set $l = l + 1$ and go to Step 2.1. If $l = p$, go
        to Step 3.
3. Set $\hat{\beta}(k+1) = \gamma(k),\ k = k + 1$ and go to Step 2. (This step may be modified
   to add the stopping criterion. For instance, terminate the iteration if
   $||\hat{\beta}(k+1) - \hat{\beta}(k)||_2/||\hat{\beta}(k)||_2$ is smaller than a prescribed small number).
```

Let the initial estimate be $\hat{\beta}(0) \geq 0$, and let

$$Q = X_n^T X_n = \begin{pmatrix} q_{11} & \cdots & q_{1p} \\ \vdots & \ddots & \vdots \\ q_{p1} & \cdots & q_{pp} \end{pmatrix},$$

$$C^T = (C_1, ..., C_p) = Y_n^T X_n - 1/2tw^T.$$

Then, we can solve (4.1) from Algorithm 4.1. To establish convergence, observe the following:

- $\text{argmin}_{\beta \geq 0} \left\{ ||Y_n - X_n\beta||_2^2 + tw^T\beta \right\}$ $=\text{argmin}_{\beta \geq 0}(\beta - \bar{\beta})^T X_n^T X_n(\beta - \bar{\beta})$with $\bar{\beta} = (X_n^T X_n)^{-1}(X_n^T Y_n - \frac{1}{2}tw)$. Thus, the weighted positive LASSO problem is a quadratic optimization over a convex set and so the solution is unique if $Q = X_n^T X_n > 0$ and there is no local minimum issue.
- From the level sets $\{\beta \geq 0 | (\beta - \bar{\beta})^T X_n^T X_n(\beta - \bar{\beta}) = c\}$, it is clear that if $\beta \geq 0$ is not the minimizer, there always exists some direction $(0, ..., 0, 1, 0, ..., 0)^T$ such that

for some ρ,

$$\beta + \rho(0, ..., 0, 1, 0, ..., 0)^T \geq 0$$

and

$$J(\beta + \rho(0, ..., 0, 1, 0, ..., 0)^T) < J(\beta)$$

as illustrated for a 2-dimensional case in Figure 4.1.

- For each l,

$$J = Y_n^T Y_n + q_{ll}\beta_l^2 + \sum_{i \neq l} q_{ii}\beta_i^2 + \sum_{i \neq j} q_{ij}\beta_i\beta_j - 2\sum_i \beta_i C_i$$

which is in a quadratic form of β_l, and $J \to \infty$ as $\beta_l \to \infty$. Further,

$$0 = \frac{\partial J}{\partial \beta_l} = 2q_{ll}\beta_l + 2\sum_{l \neq j} q_{lj}\beta_j - 2C_l$$

implies that if $\beta_1, ..., \beta_{l-1}, \beta_{l+1}, ..., \beta_p$ are fixed, the optimal $\beta_l \geq 0$ that minimizes J is given by

$$\beta_l = \max\left(\frac{C_l - \sum_{j \neq l} q_{lj}\beta_j}{q_{ll}}, 0\right).$$

- The generated sequence $\beta(k)$ is bounded and $J(\beta(k+1)) < J(\beta(k))$ if $\beta(k)$ is not the minimizer.

Now it should be clear that $\hat{\beta}(k)$ is a convergent sequence and converges to a local minimum that is also the global minimum. The algorithm performs well even when two regressors are highly correlated. In the two dimensional case, two highly correlated regressors make the ellipses in Figure 4.1 extremely narrow, and noise jitters their centers. This has very little effect on changing the optimal point from lying on one axis to the other. Summarizing these observations, we have:

Theorem 4.1 (Weighted Positive LASSO Algorithm Convergence). *Consider the weighted positive LASSO for given* $t \geq 0$, $w = (w_1, ..., w_p)^T$, $w_j \geq 0$, $j = 1, ..., p$, Y_n *and*

X_n. Assume $Q = X_n^T X_n > 0$. Then the sequence $\hat{\beta}(k)$ generated by the above algorithm converges to the solution of the weighted positive LASSO.

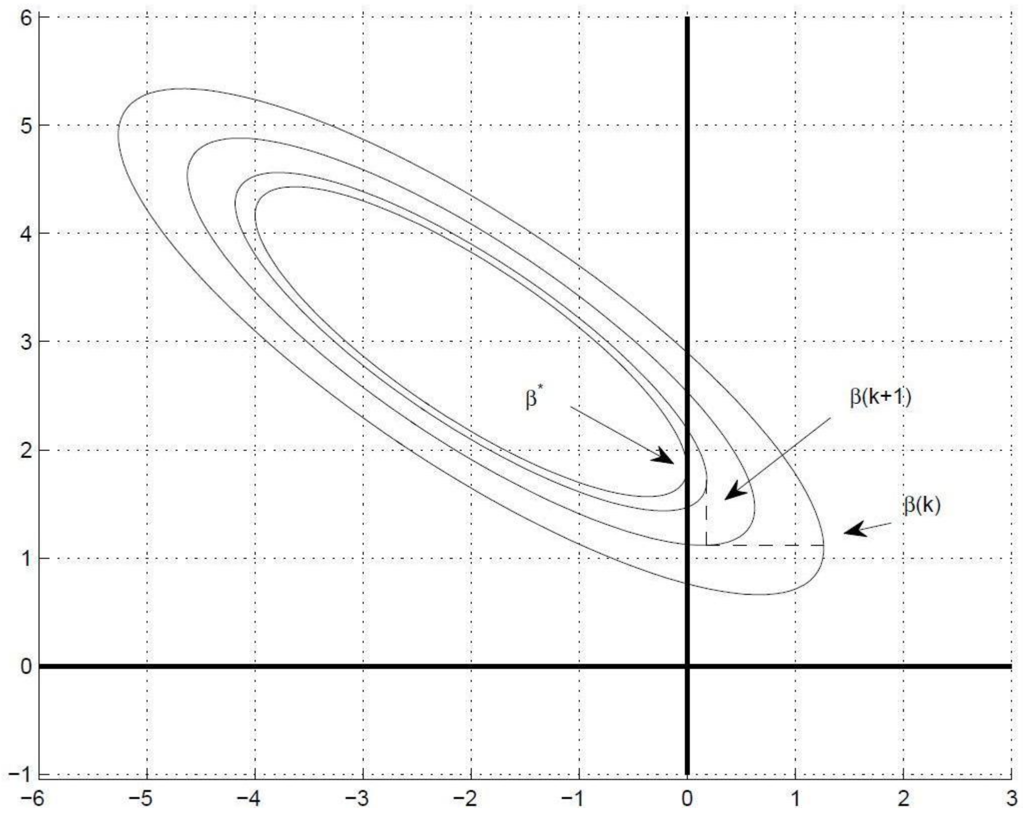


Figure 4.1: Illustration of Algorithm 4.1 in two dimensions. One complete iteration includes moving in both dimensions separately.

In a comparison with the popular LARS algorithm, consider a noisy two-dimensional example with both regressors being highly correlated with one another:

$$X_n = \begin{pmatrix} 10 & 20 \\ 10 & 22 \end{pmatrix}, \quad \beta^* = \begin{pmatrix} 2 \\ 0 \end{pmatrix}.$$

Let the noise components of V_n be i.i.d. normal with zero mean and variance 225, and t

is chosen as 0.01 for the coordinate descent algorithm. After 1,000 iterations, the modified LARS algorithm found the correct set six times, and Algorithm 4.1 found it 369 times. On three occurrences, LARS identified the correct set and the coordinate descent algorithm failed. On 366 occurrences, the coordinate descent algorithm found the correct set and LARS failed. I am not claiming one algorithm to be better than the other. Recall the duty of the algorithm is to find the estimate which produces the minimum J value in (4.1), and I have said nothing about this value. It is obvious from this example, however, that the algorithms differ from each other and can lead to a different index set. Both algorithms should be considered when solving the weighted positive LASSO.

4.2 Positive LASSO Consistency

The consistency results of [44] for the LASSO can be extended to the weighted positive LASSO. Partition X_n as

$$X_n = \begin{pmatrix} X_{n,11} & X_{n,12} \\ X_{n,21} & X_{n,22} \end{pmatrix}$$

where

$$X_{n,11} \in \Re^{d\times d}, X_{n,12} \in \Re^{d\times(p-d)}, X_{n,21} \in \Re^{(n-d)\times d},$$

$$X_{n,22} \in \Re^{(n-d)\times(p-d)}$$

Define C_n as

$$\alpha_2 I \geq \frac{1}{n} X_n^T X_n = C_n = \begin{pmatrix} c_n(1,1) & c_n(1,2) \\ c_n^T(1,2) & c_n(2,2) \end{pmatrix} \geq \alpha_1 I > 0$$

where the α_i's are independent of n. As before, define the weight vector as

$$0 \leq w = \begin{pmatrix} w_1 \\ \vdots \\ w_p \end{pmatrix} = \begin{pmatrix} w_{11} \\ \vdots \\ w_{1d} \\ w_{21} \\ \vdots \\ w_{2(p-d)} \end{pmatrix}.$$

Also define

$$a_n = c_n^T(1,2)c_n^{-1}(1,1)\begin{pmatrix} w_{11} \\ \vdots \\ w_{1d} \end{pmatrix} \in \Re^{p-d}.$$

Then it can be shown that when $\frac{n}{t} \to \infty$ and $\frac{t}{\sqrt{n}} \to \infty$ as $n \to \infty$, a sufficient condition for the weighted positive LASSO to have set consistency (2.15) and parameter consistency (2.16) is

$$a_{nj} \leq w_{2j} - \epsilon_n, \; j = 1, ..., p-d \tag{4.2}$$

and a necessary condition is

$$a_{nj} \leq w_{2j} + O_p(\frac{\sqrt{n}}{t}), \; j = 1, ..., p-d$$

where $\epsilon_n > 0$ is a positive sequence satisfying

$$\epsilon_n \to 0, \epsilon_n/(\frac{t}{n}) \to \infty, \; \epsilon_n/(\frac{\sqrt{n}}{t}) \to \infty$$

as $n \to \infty$. The proof has been omitted due to similarty to that of [44].

On one hand, the weighted positive LASSO is convex and can be solved efficiently, however whether it can capture the index set A^* depends on the sufficient condition (4.2). Notice that the condition depends on the data X_n and the weight w. In general, the condition is not satisfied. Just as the adaptive LASSO could automatically satisfy the LASSO condition (2.19), I propose the adaptive positive LASSO, Algorithm 4.2, which will automatically satisfy condition (4.2) as $n \to \infty$.

By a similar argument as in [46] for adaptive LASSO, we arrive at the following result:

Theorem 4.2 (Adaptive Positive LASSO Convergence). *Consider the adaptive positive LASSO and assume* $\frac{n}{t} \to \infty$ *and* $\frac{t}{\sqrt{n}} \to \infty$ *as* $n \to \infty$. *Then, in probability as* $n \to \infty$, $A \to A^*$ *and* $\hat{\beta} \to \beta^*$.

Algorithm 4.2 Adaptive Positive LASSO Algorithm

```
1. Given $Y_n$ and $X_n$, calculate the ordinary least squares estimate $\hat{\beta}_{LS}$.
2. Set weights $w_j = 1/|\hat{\beta}_{LSj}|$.
3. Apply the weighted positive LASSO as in (4.1) for the given $t$, $w$, $Y_n$ and $X_n$.
   Denote the solution as $\hat{\beta} = (\hat{\beta}_1, ..., \hat{\beta}_d, ...\hat{\beta}_p)^T$.
4. With the new data $Y_{n+1}$, replace $Y_n$ and $X_n$ by $Y_{n+1}$ and $X_{n+1}$ respectively, and
   set $n \to n+1$. Go to Step 1.
```

4.3 Positive RIVAL for Detection

In the previous sections, a method for solving the weighted positive LASSO problem was demonstrated and the (adaptive) positive LASSO asymptotic convergence results were supplied. Now consider the weighted positive LASSO for a given t, n, Y_n, and X_n. Assume $\frac{1}{n}X_n^T X_n > 0$. Let $0 \le q(k) \le 1$ be a positive sequence satisfying $\sum_{k=1}^{\infty} q(k) = \infty$. Now, Algorithm 4.3 describes the positive RIVAL method and Theorem 4.3 justifies its use. The positive RIVAL MATLAB code for a fixed regularization parameter MAdPosiLasso.m is available in Section 9.4.

Theorem 4.3 (Positive RIVAL Convergence). *Consider the positive RIVAL algorithm. Define* $a = \frac{n}{t}$, $\hat{d} = (\hat{d}_1, ..., \hat{d}_p) = \frac{2V_n^T X_n}{\sqrt{n}} \frac{\sqrt{n}}{t}$, $c_j = \hat{d}_j/\xi_j$, *and* $b_j = \beta_j^* + \frac{\hat{d}_j}{2a\xi_j}$. *Assume* n/t, $t \to \infty$ *as* $n \to \infty$. *Further assume* X_n *is orthogonal, i.e.,* $\frac{1}{n}X_n^T X_n = \begin{pmatrix} \xi_1 & 0 & \cdots & 0 \\ 0 & \xi_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \xi_p \end{pmatrix} >$ 0. *Then, there is an integer* $n_0 > 0$. *For any* $n \ge n_0$, *there exists an integer* $k_0 > 0$ *such* *that when the non-zero conditions*

$$b_j > \delta_j = \frac{b_j - \sqrt{b_j^2 - 2/a}}{2} > 0,\ 1 \leq 2ab_j - \frac{1}{b_j - \delta_j} \tag{4.3}$$

$$j = 1, ..., d$$

and the zero condition

$$a \geq \frac{(c_j + \delta)^2}{8}\ \textit{for any}\ \delta > 0,\quad j = d+1, ..., p \tag{4.4}$$

are satisfied, the sequence $\hat{\beta}(k)$ *generated by positive RIVAL satisfies*

$$0 < \eta_1 \leq \hat{\beta}_1(k), ..., \hat{\beta}_d(k) \leq \eta_2 < \infty,\ \forall k \geq k_0$$

$$\hat{\beta}_{d+1}(k) = \hat{\beta}_{d+2}(k) = ... = \hat{\beta}_p(k) = 0,\ \forall k \geq k_0$$

or equivalently

$$A(k) = A^*\ \forall k \geq k_0$$

where $A(k)$ *is the estimated index set at stage* k*.*

The proof of Theorem 4.3 may be found in Chapter 8.

For the positive RIVAL algorithm, the increasing rate of t as n gets larger is specified which guarantees the set consistency asymptotically. For our nuclear material application, however, n is fixed and how to choose an optimal t becomes an issue. Conceptually, positive RIVAL will generate a correct sequence of weights that identifies and eliminates irrelevant variables and the corresponding regression vectors at each iteration but requires a properly chosen t for fixed n. Too small t would not be useful for identification of irrelevant predictors and too greedy t would likely incorrectly identify variables. To this end, the optimal t_{opt} can be determined by AIC or BIC as discussed in Section 2.3.5., but with a twist. The analysis is similar for both AIC and BIC, so just BIC is analyzed here. To emphasize the dependence of A on t, write A as $A(t)$ and $\hat{\beta}_{ALS}$ as $\hat{\beta}_{A(t)LS}$. The subscript ALS indicates that the least squares estimate $\hat{\beta}_{ALS}$ is dependent on A as a result of positive RIVAL. Further, let $q(t)$ be the number of non-zero components of $\hat{\beta}_{A(t)LS}$ and $X_{A(t)n}$ denote X_n after the irrelevant regressors are removed according to $A(t)$. The optimal t_{opt}, according to BIC, is

$$t_{opt} = arg\min_t \Big\{ n \ln \frac{||Y_n - X_{A(t)n}\hat{\beta}_{A(t)LS}||_2^2}{n} + \ln(n)q(t) \Big\}.$$

The idea is that with a number of candidate t's, the weighted positive LASSO can be solved for each t, and positive RIVAL generates the corresponding subset of regressors. Trim off the irrelevant regressors from X_n and corresponding $\hat{\beta}_j$'s from the estimate. Then, the ordinary least squares estimate is computed and used to determine the estimate's BIC value. The candidate t which minimizes the BIC value is the optimal t.

Figure 4.2 illustrates the difference in using a selection operator's estimate (e.g. positive LASSO estimate) to directly calculate BIC versus the method of using the positive RIVAL-generated set to trim off irrelevant regressors and using the dimensionally-reduced ordinary least squares estimate to calculate BIC. In both cases, the t that minimizes the curves is the optimal t. From Figure 4.2, there exists only one such t for the positive LASSO scenario, however, any $t \in [121, 131]$ for the RIVAL scenario can be chosen, as these t's produce the same minimal BIC value. The reason is because the model dimension of the RIVAL-generated set is not changing over this range. Though the estimate's parameters are changing magnitudes, this information is not used, as it is the parameters of the reduced ordinary least squares that ultimately decides the BIC value. This gives the RIVAL BIC plot its rectangular shape, where the discontinuities occur when the model dimension changes (as is also the case with the positive LASSO BIC discontinuities).

Algorithm 4.3 Positive RIVAL Algorithm

1. Let $\bar{w}(1) = (\bar{w}_1(1), ..., \bar{w}_p(1))^T = (1, ..., 1)^T$, $w_j(1) = \xi_j \bar{w}_j(1)$, $j = 1, ..., p$ and set $k = 1$, where ξ_j is the norm of the jth column of the matrix $\frac{1}{n} X_n^T X_n$.
2. Apply the weighted positive LASSO as in (4.1) and denote $\hat{\beta}(k) = (\hat{\beta}_1(k), \hat{\beta}_2(k), ..., \hat{\beta}_p(k))^T$ the solution of the weighted positive LASSO with respect to the weight vector $w(k)$.
3. If $\hat{\beta}_j(k) = 0$, set $\hat{\beta}_j(k+i) = 0$ for all $i \geq 0$. Remove $\hat{\beta}_j$ and w_j from $\hat{\beta}$ and w respectively,
$$\begin{pmatrix} \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_{j-1} \\ \hat{\beta}_j \\ \hat{\beta}_{j+1} \\ \vdots \\ \hat{\beta}_p \end{pmatrix} \rightarrow \begin{pmatrix} \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_{j-1} \\ \hat{\beta}_{j+1} \\ \vdots \\ \hat{\beta}_p \end{pmatrix}, \begin{pmatrix} w_1 \\ \vdots \\ w_{j-1} \\ w_j \\ w_{j+1} \\ \vdots \\ w_p \end{pmatrix} \rightarrow \begin{pmatrix} w_1 \\ \vdots \\ w_{j-1} \\ w_{j+1} \\ \vdots \\ w_p \end{pmatrix}$$
Also remove the corresponding jth column from X_n so the dimension of the optimization is reduced by one. If $\hat{\beta}_j(k) > 0$, let
$\bar{w}_j(k+1) = q(k) \cdot \frac{1}{\hat{\beta}_j(k)} + (1 - q(k))\bar{w}_j(k),\ \ w_j(k+1) = \xi_j \bar{w}_j(k+1)$.
Repeat this processes for all $j = 1, ..., p$.
4. Set $k = k + 1$ and go back to Step 2. The dimension could be reduced if some of $\hat{\beta}_j = 0$ at Step 3. (This step may be modified to add the stopping criterion using the standard one in numerical analysis, e.g., the iterations stop if $||\hat{\beta}(k+1) - \hat{\beta}(k)||_2/||\hat{\beta}(k)||_2$ is smaller than the prescribed threshold).

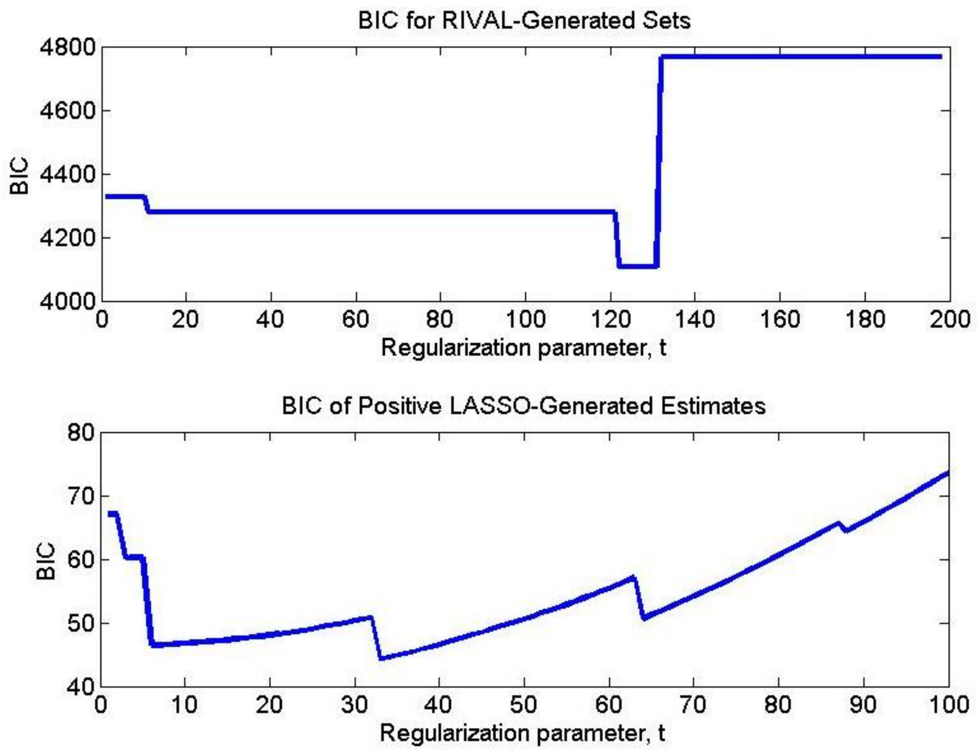


Figure 4.2: The discrete nature of a BIC plot generated with positive RIVAL (top) is due to the two-stage BIC calculation of using positive RIVAL to identify the active set of regressors and then using a dimensionally-reduced ordinary least squares estimate. The BIC plot generated with positive LASSO (bottom) will not have this characteristic because the positive LASSO estimates are used directly in the BIC calculation.

I would like to make a few comments regarding the positive RIVAL algorithm:

- Conceptually, the weights $w_j(k)$'s generated by positive RIVAL corresponding to the non-zero coefficients β_j^*, $j = 1, ..., d$ are uniformly bounded for all k and the weights $w_j(k)$'s corresponding to zero coefficients β_j^*, $j = d+1, ..., p$ grow monotonically until the corresponding estimate $\hat{\beta}_j = 0$.

- The initial estimate $\hat{\beta}(1)$ is generated by the uniform weights $w_j(1) = 1$, $j = 1, ..., p$. To increase the algorithm's speed, the inverse of the least squares could be used $w_j(1) = 1/|\hat{\beta}_{LSj}|$ if n is large.
- At least in theory, positive RIVAL has the perfect ability to find the index set if $\frac{1}{n}X_n^T X_n > 0$ is diagonal and n is large enough to satisfy the non-zero conditions (4.3) and the zero condition (4.4). Simulations suggest that the results also hold for a much larger class including when matrix $X_n^T X_n$ is diagonally dominate.
- If $A = A^*$ we can apply the ordinary least squares $\hat{\beta}_{ALS} = (\hat{\beta}_{ALS1}, ..., \hat{\beta}_{ALSd})^T$ to estimate $(\beta_1^*, ..., \beta_d^*)^T$ by trimming off zero coefficients $\hat{\beta}_j = 0$, $j = d+1, ..., p$ and the corresponding regression vectors. If n is allowed to be large, the convergence of $\hat{\beta}_{ALS}$ to $(\beta_1^*, ..., \beta_d^*)^T$ and of A to A^* is guaranteed by the properties of the least squares estimate.

Positive RIVAL is similar to non-negative garrote and adaptive LASSO in the sense that weights are adjusted based on a convergent sequence of estimates. All three have asymptotical set consistency with proper choices of the regularization parameter. However, for a fixed and finite n, the performances are different. Because of the self-adjusting ability for a fixed n which neither adaptive LASSO nor non-negative garrote possesses, positive RIVAL seems to work better. Though it is hard to quantify the improvement theoretically, a large number of simulations seem to suggest a significant improvement. I provide two examples here.

Example 1:

$$Y_n = \begin{pmatrix} 10 & 20 \\ 10 & 22 \end{pmatrix} \begin{pmatrix} 10 \\ 0 \end{pmatrix} + V_n, \ V_n \sim \mathcal{N}(0, \sqrt{5}^2 \cdot I)$$

Table 4.1: Correctly identified rates for finding the correct index set when n is small and the regressors are highly correlated (Example 1) and orthogonal (Example 2).

	LARS	Non-negative garrote	Adaptive LASSO	Positive RIVAL
Example 1	0.0004	0.0360	0.0020	0.9930
Example 2	-	0.0000	0.0413	0.9994

Example 2:

$$Y_n = \begin{pmatrix} 10 & -30 \\ 10 & 30 \end{pmatrix} \begin{pmatrix} 10 \\ 0 \end{pmatrix} + V_n,\ V_n \sim \mathcal{N}(0, 15^2 \cdot I)$$

Both are two dimensional for a small $n = 2$. This makes achieving set consistency a tough task. In Example 1, the two regressors are highly correlated, and in Example 2, the two are orthogonal.

To guarantee asymptotic consistency, the regularization parameters were chosen according to their theoretically specified values. Adaptive LASSO's regularization parameter $t = n^{1/3} = 2^{1/3}$ according to [46], and non-negative garrote's parameter $t = 3\sqrt{\log(n)/n} = 3\sqrt{\log(2)/2}$ according to [43]. In accordance with Theorem 4.3, positive RIVAL's parameter was chosen as $t = n^{1/3} = 2^{1/3}$. For Example 1, 10,000 simulations were carried out for positive RIVAL, adaptive LASSO, non-negative garrote, and LASSO implemented by LARS. The top portion of Table 4.1 shows the correctly identified rates (CIRs) for the four methods for finding the correct index set. The CIR is defined as $1-$(false positive rate + false negative rate).

Adaptive LASSO only found the correct index set 20 times out of 10,000 tries. Non-negative garrote also performed poorly, as it found the correct index set only 360 times. The reason for the poor performance in each case is the same: $n = 2$ is just too small for weight by a converging sequence of estimates to take effect. On the other hand, since positive RIVAL has an adjusting ability for a fixed n, it performed very well by identifying the correct index set 9,930 times in 10,000 tries. LARS did not fare well as expected because the two regressors are highly correlated. Thus, LARS finds a wrong regressor to start with. Figure 4.3 shows a typical path of LASSO solutions for Example 1 generated by LARS. The LARS algorithm chooses the second regressor to start with, when in fact the second regressor does not contribute to the output.

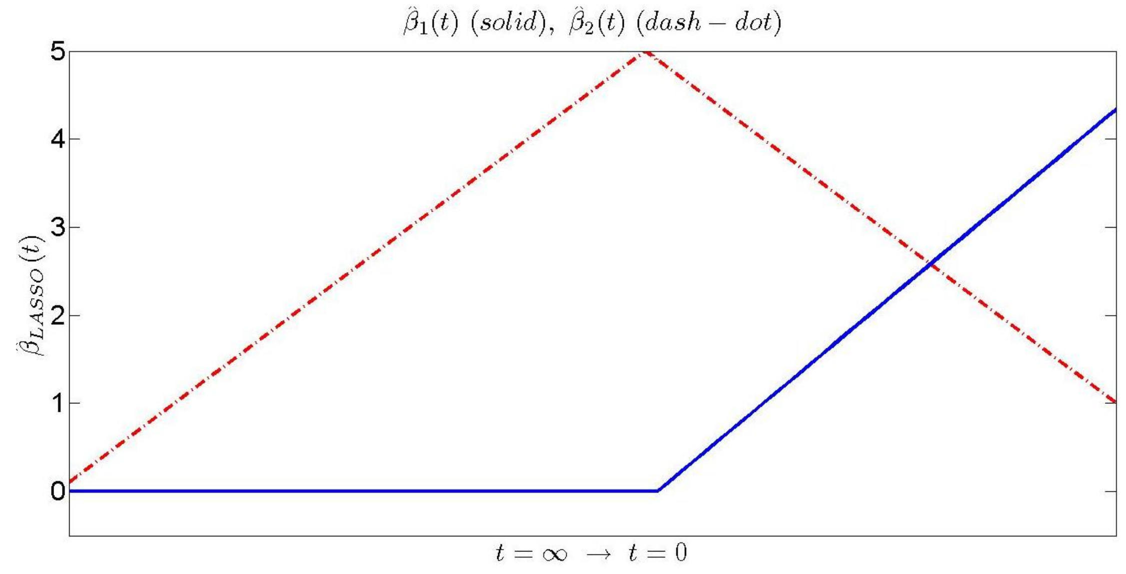


Figure 4.3: The LASSO solution path as found by the LARS algorithm often will not contain the correct model when two regressors are highly correlated. The correct model $\hat{\beta}_1 \neq 0$ and $\hat{\beta}_2 = 0$ cannot be selected for any value of t.

In a sense it is not fair to compare LARS with the other three methods because the other three methods have an advantage of adjusting weights based on a convergent sequence of estimates for each n, e.g., by the least squares estimate. LARS does not have this advantage. However, it is fair to compare positive RIVAL with adaptive LASSO and non-negative garrote. To this end, Example 2 is used to test how the three methods perform when the regressors are orthogonal. 10,000 more simulations were run with the same choices of regularization parameters. The bottom portion of Table 4.1 shows the results. Again, non-negative garrote and adaptive LASSO did not perform well because of a small n, but positive RIVAL worked almost perfectly. This demonstrates the utility of positive RIVAL: it achieves set consistency for a smaller number of data than is required by its competitors.

Now, consider the nuclear material problem presented in Section 3.2. Recall the true $\beta^* = \alpha \cdot (0.2, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1)^T$ and the signal to noise ratio defined in equation (3.5). Also recall the 11 isotopes involved in testing found in Table 3.1. For this problem, $n = 1024$ is large but fixed, so positive RIVAL should be applied. The average error rates of 500 simulations with $q(k) \equiv 1$ are shown in Table 4.2. For comparison, the results of the adaptive LASSO and non-negative garrote are provided with two different choices of t. One choice is according to the asymptotic value for convergence when $n \to \infty$, and the other, whose results are shown in parentheses, is based on AIC/BIC for the fixed $n = 1024$. The results of standard LASSO using AIC and BIC are shown in Table 3.3 for an SNR of -10 dB. Clearly, positive RIVAL outperforms traditional methods such as LASSO and peak detection algorithms, whose detection performances were presented in Section 3.2.

4.4 RIVAL without Positivity

Somewhat unrelated to nuclear material detection, this section gives the performance results of positive RIVAL when the unknown parameters are allowed to be negative. Relaxing the positivity constraint allows the modified algorithm, called RIVAL, to be useful in more general applications. I directly compare RIVAL to popular variable selectors

Table 4.2: False positive and false negative rates of the positive RIVAL algorithm (AIC and BIC), and error rates for adaptive LASSO and the non-negative garrote with regularization parameters chosen according to theoretical values and by AIC/BIC (in parentheses).

Method	SNR (dB)	False positive rate	False negative rate
Positive RIVAL	-10	0.0000	0.0000
	-20	0.0005	0.0013
	-30	0.0010	0.0015
	-32	0.0017	0.0065
Adaptive LASSO	-10	0.0624 (0.0164)	0.0000 (0.0000)
	-20	0.0771 (0.0482)	0.1387 (0.1353)
Non-negative garrote	-10	0.0000 (0.0803)	0.6667 (0.0000)
	-20	0.0000 (0.0649)	0.6667 (0.1360)

using well known simulated examples presented in other papers. Further, I abandon the use of AIC and BIC to select model dimensions in favor of cross validation as to conform to these papers. The results suggest that indeed RIVAL can be applied confidently in situations where the scientist or statistician cannot guarantee unknown parameters to be non-negative.

Just as positive RIVAL uses the weighted positive LASSO, RIVAL makes use of the weighted LASSO (2.21). The convergence results of the weighted LASSO can be established by generalizing the irrepresentable condition for weighted positive LASSO (4.2) to

$$|a_{nj}| \leq w_{2j} - \epsilon_n, \;\; j = 1, ..., p-d.$$

This condition we've seen before; it is the condition (2.19) that motivated the creation of the adaptive LASSO.

The RIVAL algorithm is the same as positive RIVAL only with a few modifications.

Instead of solving the weighted positive LASSO, solve the weighted LASSO (2.21). Also, use $|\hat{\beta}_j(k)|$ to update the weights. Algorithm 5.3 summarizes the RIVAL method.

Algorithm 4.4 RIVAL Algorithm

1. Same as positive RIVAL Step 1.
2. Same as positive RIVAL Step 2, but solve the weighted LASSO (2.21) instead of the positive LASSO.
3. Same as positive RIVAL Step 3, but update the weights as $\bar{w}_j(k+1) = q(k) \cdot \frac{1}{|\hat{\beta}_j(k)|} + (1 - q(k))\bar{w}_j(k),\ w_j(k+1) = \xi_j \bar{w}_j(k+1).$
4. Same as positive RIVAL Step 4.

The use of RIVAL is justified by the following result:

Theorem 4.4 (RIVAL Convergence). *Consider the RIVAL algorithm. Define* $a = \frac{n}{t}$, $\hat{d} = (\hat{d}_1, ..., \hat{d}_p) = \frac{2V_n^T X_n}{\sqrt{n}} \frac{\sqrt{n}}{t}$, $c_j = \hat{d}_j/\xi_j$, *and* $b_j = \beta_j^* + \frac{\hat{d}_j}{2a\xi_j}$. *Assume* $n/t,\ t \to \infty$ *as* $n \to \infty$. *Further assume* X_n *is orthogonal, i.e.,* $\frac{1}{n} X_n^T X_n = \begin{pmatrix} \xi_1 & 0 & \cdots & 0 \\ 0 & \xi_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \xi_p \end{pmatrix} > 0$. *Then, there is an integer* $n_0 > 0$. *For any* $n \geq n_0$, *there exists an integer* $k_0 > 0$ *such that when the non-zero conditions*

$$b_j > \delta_j = \frac{|b_j| - \sqrt{b_j^2 - 2/a}}{2} > 0,\ 1 \leq 2a|b_j| - \frac{1}{|b_j| - \delta_j} \tag{4.5}$$

$$j = 1, ..., d$$

and the zero condition

$$a \geq \frac{(c_j+\delta)^2}{8} \; for \; any \; \delta > 0, \quad j = d+1, ..., p \tag{4.6}$$

are satisfied, the sequence $\hat{\beta}(k)$ generated by RIVAL satisfies

$$0 < \eta_1 \leq |\hat{\beta}_1(k)|, ..., |\hat{\beta}_d(k)| \leq \eta_2 < \infty, \;\; \forall k \geq k_0$$

$$\hat{\beta}_{d+1}(k) = \hat{\beta}_{d+2}(k) = ... = \hat{\beta}_p(k) = 0, \;\; \forall k \geq k_0$$

or equivalently

$$A(k) = A^* \; \forall k \geq k_0$$

where $A(k)$ is the estimated index set at stage k.

This result is similar to that of Theorem 4.3 and so the proof has been omitted.

RIVAL works just like positive RIVAL, though it should be expected to require a larger n for convergence than positive RIVAL since the positivity constraint has been relaxed. The performance of the algorithm can be illustrated in a few examples from the literature. In [38], the author provides three examples to test the performance of his LASSO method. These examples were also used in [46] to test the adaptive LASSO against the original LASSO. I shall begin with these three examples and compare the performances of general RIVAL to those of LASSO, adaptive LASSO, and non-negative garrote in the context of variable selection. The MATLAB code rival.m in Section 9.6 is the implementation of Algorithm 5.3 for a fixed regularization parameter.

For Example 1, $Y_n = X_n\beta^* + V_n$, $\beta^* = (3, 1.5, 0, 0, 2, 0, 0, 0)^T$. The noise components are i.i.d. normal with zero mean and standard deviation σ. $X_n = (X_{1n} \; ... \; X_{8n})$ and the correlation between X_{in} and X_{jn} is $\rho^{|i-j|}$ with $\rho = 0.5$. For testing, $n = 20, 60, 100$ and

Table 4.3: Correctly identified rates of RIVAL, LASSO, adaptive LASSO and non-negative garrote for Example 1.

		LASSO	Adaptive LASSO	RIVAL	Non-negative garrote
$n = 20$	$\sigma = 1$	0.470	0.485	0.850	0.695
	$\sigma = 2$	0.300	0.320	0.410	0.285
$n = 60$	$\sigma = 1$	0.740	0.770	0.910	0.880
	$\sigma = 2$	0.750	0.760	0.785	0.700
$n = 100$	$\sigma = 1$	0.830	0.835	0.930	0.895
	$\sigma = 2$	0.800	0.780	0.835	0.770

Table 4.4: Correctly identified rates of RIVAL, LASSO, adaptive LASSO and non-negative garrote for Example 2.

		LASSO	Adaptive LASSO	RIVAL	Non-negative garrote
$n = 20$	$\sigma = 1$	0.400	0.445	0.930	0.460
	$\sigma = 2$	0.020	0.040	0.475	0.235
$n = 60$	$\sigma = 1$	0.990	0.990	0.995	0.975
	$\sigma = 2$	0.460	0.445	0.880	0.655
$n = 100$	$\sigma = 1$	1.000	1.000	1.000	1.000
	$\sigma = 2$	0.760	0.795	0.925	0.873

standard deviations of 1 and 2 were considered for each n. 200 simulations were run for each combination of n and σ with 5-fold cross validation to choose the model. The average correct identification rates are shown in Table 4.3. From the results, we see the utility of RIVAL in action: RIVAL requires a smaller n than its competitors. When $n = 20$, RIVAL is nearly 0.15 better than the second best competitor. It is only when n becomes larger do the other methods start relatively improving. But even when $n = 100$, RIVAL performs the best. From the table, it is obvious that RIVAL is the preferred method for Example 1.

Example 2 is the same as Example 1, but with $\beta_j^* = 0.85$, $j = 1, ..., 8$. RIVAL really

Table 4.5: Correctly identified rates of RIVAL, LASSO, adaptive LASSO and non-negative garrote for Example 3.

		LASSO	Adaptive LASSO	RIVAL	Non-negative garrote
$n = 20$	$\sigma = 1$	0.875	0.890	0.940	0.855
	$\sigma = 2$	0.886	0.875	0.810	0.815
$n = 60$	$\sigma = 1$	0.984	0.988	0.965	0.915
	$\sigma = 2$	0.980	0.980	0.865	0.875
$n = 100$	$\sigma = 1$	0.988	0.992	0.975	0.930
	$\sigma = 2$	0.990	0.992	0.905	0.890

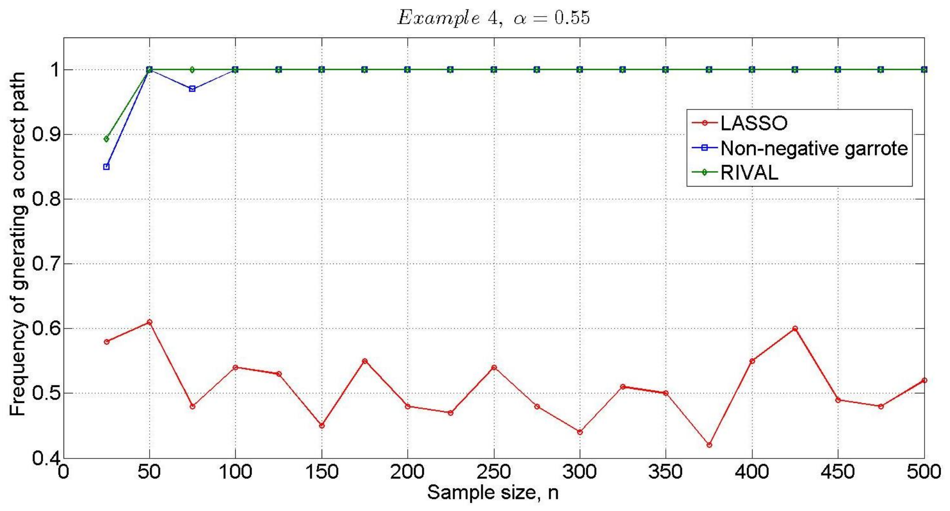
does well here; it outperforms its competitors in every scenario, as shown in Table 4.4. RIVAL is especially good again when $n = 20$.

Example 3 is the same as Example 2, however $\beta^* = (5, 0, 0, 0, 0, 0, 0, 0)^T$. RIVAL performs well again, but so do the others. Notice from Table 4.5 that RIVAL's best relative performance comes again when $n = 20$.

The next example, Example 4,

$$Y_n = X_n\beta^* + V_n,\ X_n = (X_{1n}\ X_{2n}\ X_{3n}),\ \beta^* = (1, 1, 0)^T,\ V_n \sim \mathcal{N}(0, I),$$

is supplied from a paper which analyzes the solution path of the non-negative garrote [43]. The claim is that the non-negative garrote is path consistent, i.e., the probability that its solution path contains the correct model approaches 1 as $n \to \infty$. The paper compares the frequency of correct paths produced by non-negative garrote to that of LASSO. Recall that a correct path is one that contains the correct model. This is an interesting example, because, unlike the previous three, this example does not attempt to pick the correct model. Thus, blame for any errors falls solely on the variable selection method and not on the preferred method for estimating the model dimension, e.g., AIC, BIC, or cross validation. I use the data from this paper and compare it with RIVAL's performance.



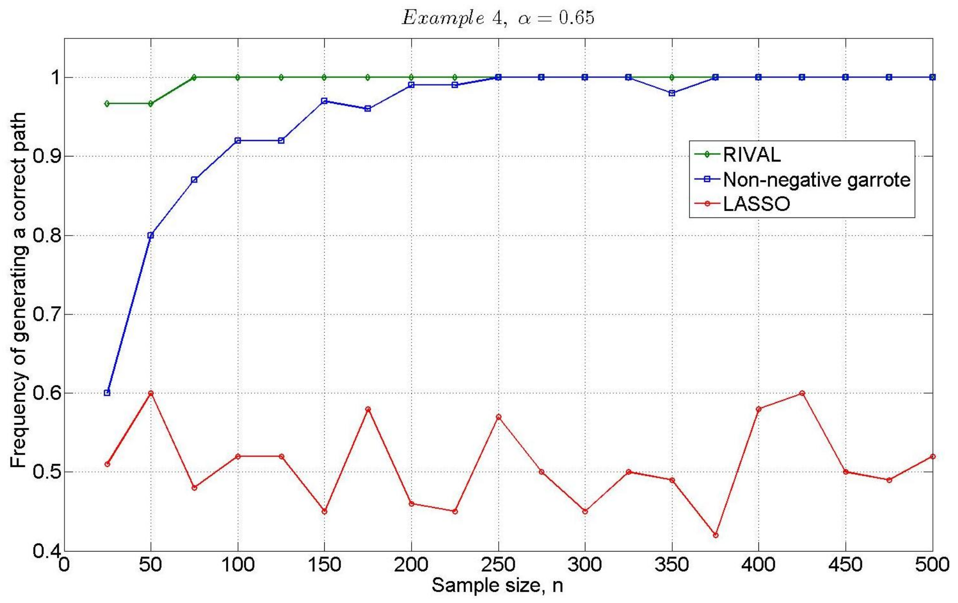


Figure 4.4: Frequency of generating correct paths for RIVAL, non-negative garrote, and LASSO in Example 4. For $\alpha = 0.55$, LASSO is inconsistent and non-negative garrote begins to falter. For $\alpha = 0.65$, RIVAL clearly outperforms LASSO and non-negative garrote.

The two active regressors X_{1n} and X_{2n} were independently simulated from a standard normal distribution. X_{3n} was generated from a normal distribution with mean $\alpha(X_{1n}+X_{2n})$ and variance $1-2\alpha^2$. Four different α's were considered: 0.35, 0.45, 0.55, 0.65. For each α, 20 equally spaced sample sizes were considered: 25, 50, ..., 500. For each scenario, 100 data sets were simulated, and the rates of generating a correct path were recorded. In this example, a correct path is one in which $\hat{\beta}_1 \neq 0$, $\hat{\beta}_2 \neq 0$ and $\hat{\beta}_3 = 0$.

For $\alpha = 0.35$, all three methods were consistent, though RIVAL performed better than LASSO at small sample sizes. When $\alpha = 0.45$, RIVAL and non-negative garrote were consistent. LASSO does in fact approach a frequency of 1, however it is not monotonically increasing. The performances of the three methods for $\alpha = 0.55$ and $\alpha = 0.65$ are shown in Figure 4.4. When $\alpha = 0.55$, both RIVAL and non-negative garrote performed well, but LASSO is inconsistent. Finally, when $\alpha = 0.65$, LASSO remains inconsistent and RIVAL clearly outperforms non-negative garrote.

CHAPTER 5
SHIELDING

When a material is placed between a source of radiation and the detector, thus decreasing the gamma-ray intensity reaching the detector, that material is said to be a shield. Shielding nuclear materials can be an effective method for deceiving a detector, hence a useful detection algorithm should be able to perform adequately under modest shielding conditions. It is well known from physics that shielding materials such as lead, concrete, carbon and water attenuate lower energy radiation more so than radiation of higher energy, and thereby change the characteristic shapes of the isotope spectra in Figure 3.1. To manage this problem, it will be convenient to express each isotope's spectrum as a superposition of its characteristic peaks - its sub-spectra. The introduction of sub-spectra requires new selection algorithms since it is counterproductive for an algorithm to select between sub-spectra of the same group (isotope). As it happens, the RIVAL algorithm can be modified to incorporate variables with a group structure [23].

To demonstrate the need for the selection of grouped variables, consider an example from the literature [13] with $n = 200$ observations and $p = 100$ predictors, in blocks of ten. The number of non-zero coefficients in the first six blocks are 10, 8, 6, 4, 2 and 1 respectively, with coefficients equal to ± 1, the sign chosen at random. The next four blocks are all zero. The regressors are standard normal with correlation of 0.2 within a group and zero otherwise. Finally, Gaussian noise with zero mean and variance 4^2 is added to each observation.

The average number of misclassifications by RIVAL in 500 simulations is shown in Figure 5.1 for a grid of regularization parameters. From the figure, it appears that RIVAL can do no better than twelve misclassifications on average. Figure 5.2 shows the performance in one simulation with a regularization parameter of $t = 1.4$. There are errors in all but two groups. In fact, RIVAL classifies two of the four irrelevant groups as non-zero. The reason for the large number of errors is that each parameter has its own weight and this encourages sparsity at the individual parameter level. If a group structure is known a priori, then this information should be incorporated into the algorithm to encourage sparsity at the group level.

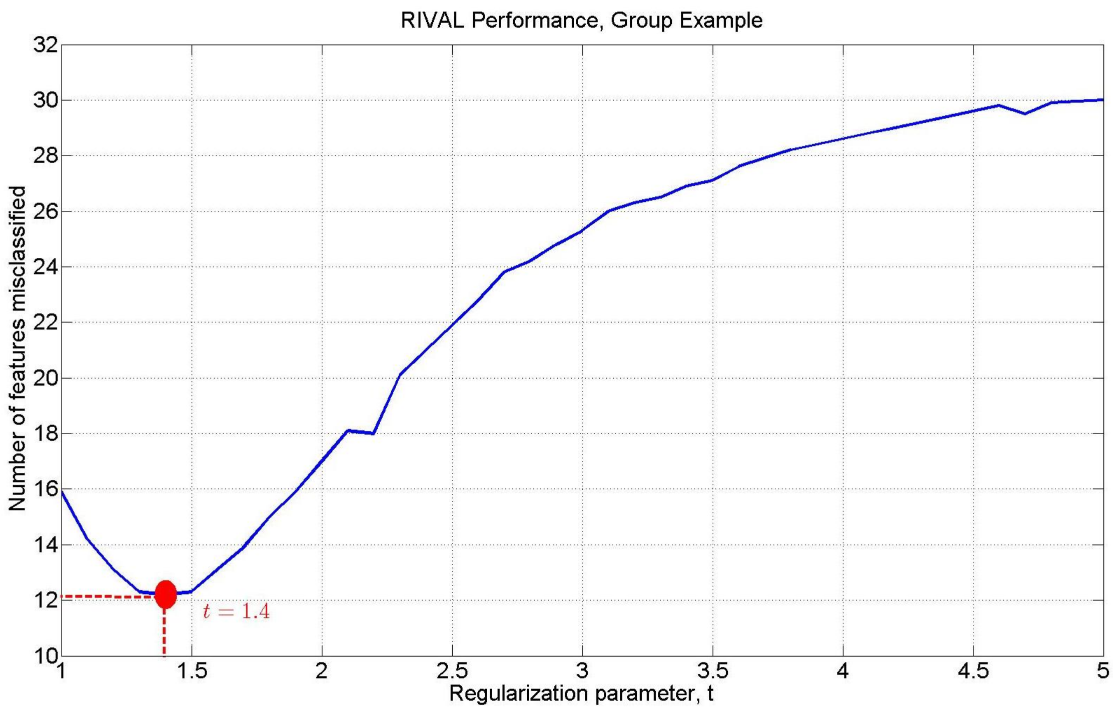


Figure 5.1: Results for the group example showing the number of variables misclassified by RIVAL changes with the regularization parameter.

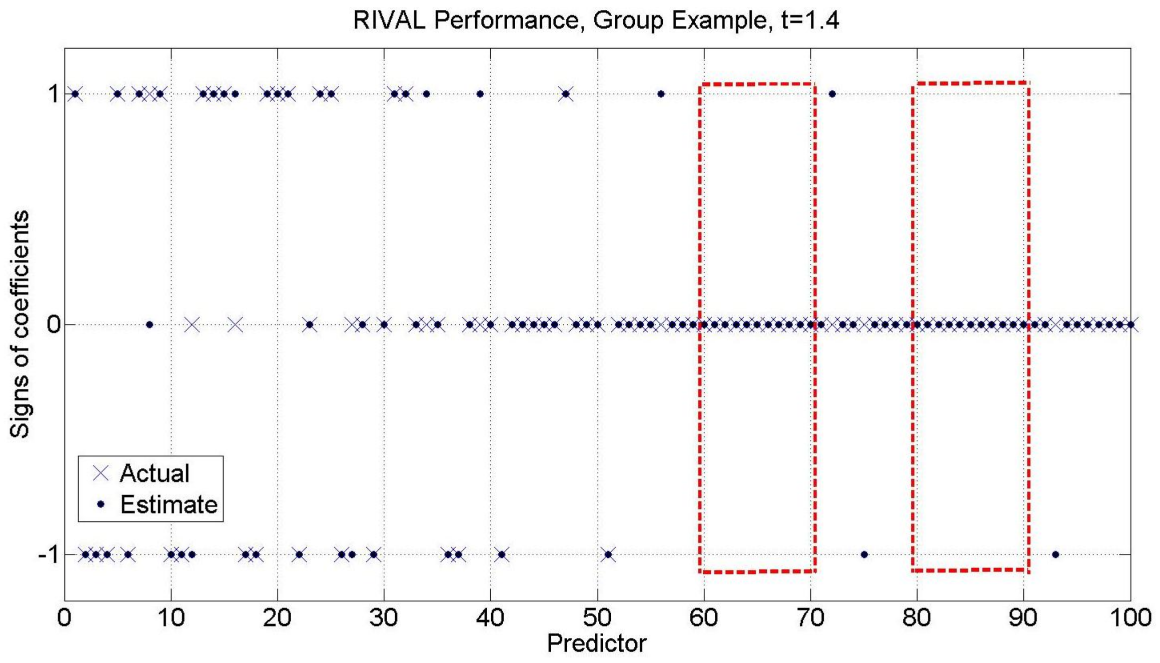


Figure 5.2: Result of one simulation of the group example for $t = 1.4$. The actual coefficients are shown as X's and the signs of the estimated coefficients are shown with filled in circles. The seventh and the ninth groups are the only ones to be classified completely correctly.

5.1 Modeling Shielded Intensitites

If the spectrum of the jth isotope has K_j characteristic peaks, or sub-spectra, it may be expressed as a superposition of the K_j sub-spectra. In this case, the spectrum of the jth radioactive source is a matrix

$$X_j = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1K_j} \\ x_{21} & x_{22} & \cdots & x_{2K_j} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nK_j} \end{pmatrix}.$$

Each column of X_j gives the mean gamma-ray counts per unit source material and per unit time of each sub-spectrum comprising the jth isotope. Now, each sub-spectrum is considered mono-energetic and the attenuation by a shield is assumed constant over the entire sub-spectra. I131 has $K_j = 5$ energy peaks with a branching ratio larger than 0.005, and thus its spectrum can be decomposed into five sub-spectra, with five column vectors comprising X_j. The decomposition of I131 is illustrated in Figure 5.3 and the decomposition of Pu239 into its $K_j = 10$ sub-spectra is shown in Figure 5.4. Branching ratio is defined as the fraction of particles which undergo a certain type of decay with respect to the total number of particles that decay.

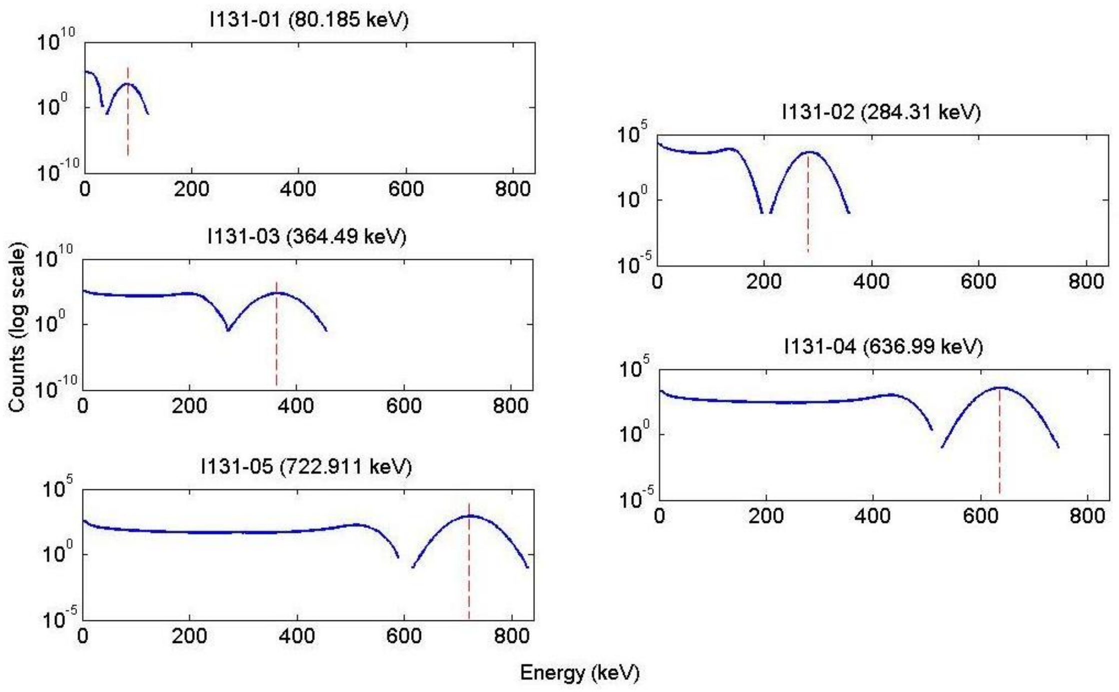


Figure 5.3: The decomposition of I131 into its five sub-spectra with branching ratios larger than 0.005.

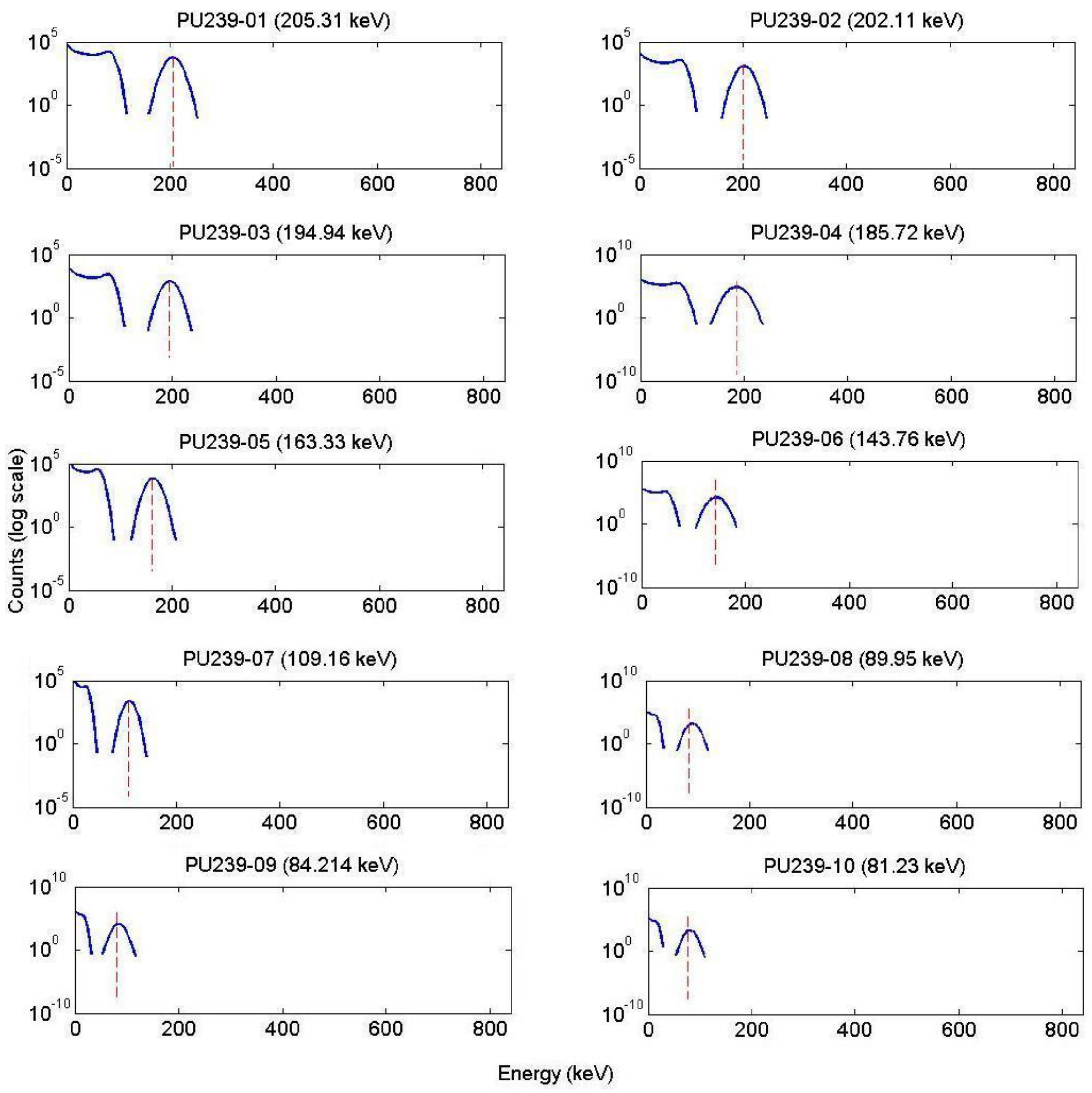


Figure 5.4: The decomposition of Pu239 into its ten sub-spectra with branching ratios larger than 0.005.

Define

$$X_n = (X_1 X_2 \cdots X_p),\ \beta^0 = \begin{pmatrix} \beta_1^0 \\ \beta_2^0 \\ \vdots \\ \beta_p^0 \end{pmatrix}$$

where the vector $\beta_j^0 \in R^{K_j}$ is the unshielded relative intensities of the sub-spectra from the jth isotope. All of the components of β_j^0 are equal due to the fact that each sub-spectra in group j has the same parent isotope. Further, let the received gamma-ray counts at the channel i be $Y(i)$. Then,

$$Y = \sum_{j=1}^{p} X_j \beta_j^0 + V,$$

or equivalently,

$$\begin{pmatrix} Y(1) \\ \vdots \\ Y(n) \end{pmatrix} = X\beta^0 + V$$

where, as before, V is a random vector describing the difference between the actual received gamma-ray counts and its statistical average, and it can be assumed that V is Gaussian with zero mean. However, since the variance at each channel is different, each component of V does not have the same variance. The effect of non-homogeneous variance at different channels may be reduced by normalizing the model by $\bar{Y}$

$$\underbrace{\bar{Y}\begin{pmatrix} Y(1) \\ \vdots \\ Y(n) \end{pmatrix}}_{Y_n} = \underbrace{\bar{Y}X}_{X_n}\beta^0 + \underbrace{\bar{Y}V}_{V_n}, \qquad \bar{Y} = \begin{pmatrix} \frac{1}{\sqrt{Y(1)}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\sqrt{Y(n)}} \end{pmatrix}.$$

Now, if $X_{jn} = \bar{Y}X_j$, we have

$$Y_n = \sum_{j=1}^{p} X_{jn}\beta_j^0 + V_n,$$

or simply

$$Y_n = X_n\beta^0 + V_n.$$

Define l as the product of the shielding material's density and its thickness. This product l is called the mass thickness and has units of grams per square centimeter. The amount of attenuation caused by a shielding material of mass thickness l at energy level i is an exponential function [29], $e^{-\mu_i l}$, where μ_i, having units of square centimeters per gram, is the mass attenuation coefficient at the ith energy level that depends on the type of shielding material. The mass attenuation coefficients are well known for various shielding materials [29] and depend on the energy of the incoming gamma-rays, as shown in Figure 5.5 for lead. For a given shielding material and mass thickness l, define the attenuation matrix

$$D = \begin{pmatrix} D_1 & 0 & \cdots & 0 \\ 0 & D_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_p \end{pmatrix}, \; D_j = \begin{pmatrix} e^{-\mu_{j,1}l} & 0 & \cdots & 0 \\ 0 & e^{-\mu_{j,2}l} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{-\mu_{j,K_j}l} \end{pmatrix}, \tag{5.1}$$

where $\mu_{j,i}$ is the attenuation coefficient at the energy level corresponding to the i sub-spectra peak of the jth isotope. For example, if j is the index corresponding to I131, values of $\mu_{j,i}$ for lead, $i = 1, ..., 5$ can be read off of Figure 5.5 for the five sub-spectra of I131. In this case $\mu_{j,1} = 1.86$, $\mu_{j,2} = 0.367$, $\mu_{j,3} = 0.231$, $\mu_{j,4} = 0.106$, and $\mu_{j,5} = 0.093$ (all in units of cm^2/g). Then, the intensities after the shield has been applied are

$$\beta^* = \begin{pmatrix} \beta_1^* \\ \beta_2^* \\ \vdots \\ \beta_p^* \end{pmatrix} = D\beta^0, \; \beta_j^* = (\beta_{j1}^*, \beta_{j2}^*, ..., \beta_{jK_j}^*)^T,$$

where the K_j components of vector β_j^* are the shielded intensities of the sub-spectra from the jth isotope. Now the system is modeled as

$$Y_n = \sum_{j=1}^{p} X_{jn}\beta_j^* + V_n, \tag{5.2}$$

or

$$Y_n = X_n\beta^* + V_n. \tag{5.3}$$

In the analysis that follows, it is assumed that the normalized spectrum is given by (5.2), or equivalently, (5.3), with the components of V_n being i.i.d. of zero mean and constant variance. The validity of this assumption is demonstrated in Section 5.3 where nuclear materials are simulated as Poisson random variables, but detection is based on the Gaussian model (5.3).

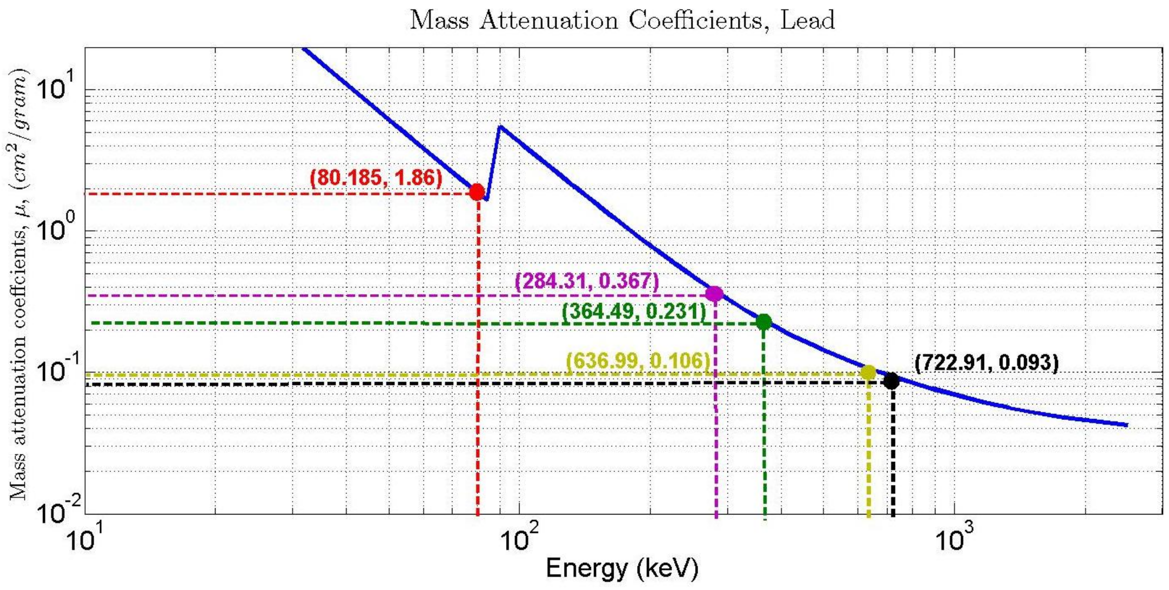


Figure 5.5: The mass attenuations coefficients for lead plotted against energy. The labeled points are those applying to I131 and its five sub-spectra.

5.2 Group Positive LASSO Consistency

The system (5.3) is assumed to be sparse, that is to say that some of the unknown groups of coefficients β_j^* are exactly zero, corresponding to isotopes that are absent. Further, for the application of nuclear material detection, the non-zero coefficients must be positive when the material is present.

Suppose there are an unknown number $0 < d < p$ of non-zero groups corresponding to the d isotopes which are present. Without loss of generality, we may re-arrange indices such that

$$\beta^* = \begin{pmatrix} \beta_1^* \\ \beta_2^* \\ \vdots \\ \beta_d^* \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \ ||\beta_1^*||_2 > 0, ..., ||\beta_d^*||_2 > 0, \tag{5.4}$$

$$||\beta_{d+1}^*||_2 = ... = ||\beta_p^*||_2 = 0.$$

Now, consider the true but unknown index set to be

$$A^* = \{j \ : \ ||\beta_j^*||_2 > 0\}. \tag{5.5}$$

The problem of nuclear material detection is again one of variable selection: identify the index set A^* and remove those irrelevant groups of regressors as well as the corresponding isotopes, thus identifying which materials are present and which are absent. It would be nice to be able to use positive RIVAL for this problem, but recall that at the heart of the positive RIVAL algorithm is the positive LASSO, and both LASSO and positive LASSO encourage selection among individual parameters instead of grouped parameters [42], and thus, for this application, the positive RIVAL method leaves more to be desired. To combat this problem, I introduce the group positive LASSO.

Let an estimate of β^* be

$$\hat{\beta} = \begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \vdots \\ \hat{\beta}_p \end{pmatrix}, \ \hat{\beta}_j = (\hat{\beta}_{j1}, \hat{\beta}_{j2}, ..., \hat{\beta}_{jK_j})^T, \ j = 1, 2, ..., p$$

then the group positive LASSO estimate is the β that solves

$$\min_{\beta \geq 0} \left\{ ||Y_n - X_n\beta||_2^2 + t \sum_{j=1}^{p} w_j \sum_{i=1}^{K_j} \beta_{ji} \right\} \tag{5.6}$$

where $\beta \geq 0$ is taken component-wise, $t \geq 0$ is a regularization parameter, and $w = (w_1, w_2, ..., w_p)^T$ is a non-negative weighting vector. Group positive LASSO encourages grouping by applying the same weight w_j to each parameter in the jth group. The role of t is to balance the two terms in (5.6): a $t = 0$ produces the least squares estimate, and a very large lambda forces every group estimate to zero.

For a given Y_n, X_n and $w = (w_1, w_2, ..., w_p)^T$, one can efficiently solve the group positive LASSO (5.6) in a similar fashion to the method for solving the adaptive LASSO [46]. Algorithm 5.1 describes the method in detail. Indeed, this algorithm solves the group positive LASSO for all t. The optimal regularization parameter t_{opt} can be determined by cross-validation or AIC/BIC.

Algorithm 5.1 Group Positive LASSO Algorithm

1. Define $X_{jn}^{**} = X_{jn}/w_j, \ j = 1, ..., p$.
2. Modify the popular LARS [9] algorithm to incorporate the positive LASSO.

   Use it to solve the positive LASSO problem for all t,

   $$\hat{\beta}^{**} = \begin{pmatrix} \hat{\beta}_1^{**} \\ \hat{\beta}_2^{**} \\ \vdots \\ \hat{\beta}_p^{**} \end{pmatrix} = arg \ \min \left\{ ||Y_n - \textstyle\sum_{j=1}^{p} X_{jn}^{**}\beta_j||_2^2 + t \sum_{j=1}^{p} \sum_{i=1}^{K_j} \beta_{ji} \right\}, \ \beta \geq 0$$
3. The group positive LASSO solution for the jth group is $\hat{\beta}_j^{**}/w_j, \ j = 1, ..., p$.

For a given data set Y_n and X_n, define the estimated set A to be the estimate of A^* in (5.5). To establish the consistency of group positive LASSO, recall that for a variable selector to be consistent, it should have the properties (2.15) and (2.16), namely, those groups of parameters which have non-zero norms are correctly identified, and the estimates of the unknown parameters grow closer to the true values as more data is available.

Let z be the total number of components in β^* across all non-zero groups, i.e., $z = \sum_{j=1}^{d} K_j$, and let m be the length of vector β^*, i.e., $m = \sum_{j=1}^{p} K_j$. Partition the regressor matrix X_n as

$$X_n = \begin{pmatrix} X_{n,11} & X_{n,12} \\ X_{n,21} & X_{n,22} \end{pmatrix}$$

where

$$X_{n,11} \in \Re^{z\times z}, X_{n,12} \in \Re^{z\times(m-z)}, X_{n,21} \in \Re^{(n-z)\times z}, X_{n,22} \in \Re^{(n-z)\times(m-z)}.$$

Define C_n and assume

$$\alpha_2 I \geq \frac{1}{n} X_n^T X_n = C_n = \begin{pmatrix} c_n(1,1) & c_n(1,2) \\ c_n(2,1) & c_n(2,2) \end{pmatrix} \geq \alpha_1 I > 0$$

where the α_i's are independent of n. Also define

$$W_1^T = (W_{11}, W_{12}, ..., W_{1d}),$$

$$W_{1j} = w_j \cdot \underbrace{(1, ..., 1)}_{K_j}, \quad j = 1, ...d$$

so that the length of W_1 is z, and similarly define

$$W_2^T = (W_{21}, W_{22}, ..., W_{2(p-d)}),$$

$$W_{2j} = w_{d+j} \cdot \underbrace{(1, ..., 1)}_{K_{d+j}}, \quad j = 1, ...p-d$$

so that the length of W_2 is $m - z$. Let

$$a_n = c_n^T(1,2) c_n^{-1}(1,1) W_1 \in \Re^{m-z}$$

then, in an analysis similar to that of [44] for LASSO, it can be shown that when $\frac{n}{t} \to \infty$ and $\frac{t}{\sqrt{n}} \to \infty$ as $n \to \infty$, a sufficient condition for the group positive LASSO to have set consistency (2.15) and parameter consistency (2.16) is

$$a_{nj} \le (0, ..., 0, \underbrace{1}_{jth}, 0, ..., 0)W_2 - \epsilon_n,\ j = 1, ..., m - z \tag{5.7}$$

and a necessary condition is

$$a_{nj} \le (0, ..., 0, \underbrace{1}_{jth}, 0, ..., 0)W_2 + O_p(\frac{\sqrt{n}}{t}),\ j = 1, ..., m - z \tag{5.8}$$

where $\epsilon_n > 0$ is a positive sequence satisfying

$$\epsilon_n \to 0,\ \epsilon_n / \Big(\frac{t}{n}\Big) \to \infty,\ \epsilon_n / \Big(\frac{\sqrt{n}}{t}\Big) \to \infty$$

as $n \to \infty$.

It would be convenient to be able to modify the group positive LASSO to guarantee that condition (5.7) is satisfied and the correct set A^* identified, at least for when n is large. In fact, the condition is automatically satisfied when the weights w_j's, $j = 1, ..., d$ are small and the weights w_j's, $j = d + 1, ..., p$ are large. Of course, d is unknown, but, as was the case with adaptive positive LASSO, the idea of data-dependent weights may be used. Properly choosing the relationship between the w_j's and the β_j's will give small weights for those groups whose estimates are non-zero and give large weights for those groups whose estimates are close to zero. Thus, it is natural to propose the adaptive group positive LASSO as described in Algorithm 5.2.

By a similar argument as in [46] for adaptive LASSO, the following theorem is in place:

Theorem 5.1 (Adaptive Group Positive LASSO Convergence). *Consider the adaptive group positive LASSO and assume $\frac{n}{t} \to \infty$ and $\frac{t}{\sqrt{n}} \to \infty$ as $n \to \infty$. Then, in probability as $n \to \infty$, $A \to A^*$ and $\hat{\beta} \to \beta^*$.*

Algorithm 5.2 Adaptive Group Positive LASSO Algorithm

1. Given Y_n and X_n, calculate the ordinary least squares estimate $\hat{\beta}_{LS} = (\hat{\beta}_{LS1}^T, \hat{\beta}_{LS2}^T, ..., \hat{\beta}_{LSp}^T)^T$, where $\hat{\beta}_{LSj}^T = (\hat{\beta}_{LSj1}, \hat{\beta}_{LSj2}, ..., \hat{\beta}_{LSjK_j})$.
2. Set weights $w_j = 1/||\hat{\beta}_{LSj}||_1,\ j = 1, ..., p$, where $|| \cdot ||_1$ is the 1-norm.
3. Apply the group positive LASSO as in (5.6) for the given w, Y_n, and X_n. Denote the solution corresponding to t as $\hat{\beta} = (\hat{\beta}_1^T, \hat{\beta}_2^T, ..., \hat{\beta}_p^T)^T$.
4. Only when a new measurement is taken and new data is gathered, replace Y_n and X_n by Y_{n+1} and X_{n+1}, respectively, and set $n \to n + 1$. Go to Step 1.

5.3 Group Positive RIVAL for Detection

As was the case with adaptive positive LASSO, the asymptotic convergence results of group adaptive positive LASSO are not extremely useful for detecting nuclear materials because the number of detector channels is finite. With a large but fixed n, it should be possible to, by a properly chosen sequence of weights $w(k)$'s, generate a corresponding sequence of group positive LASSO solutions $\hat{\beta}(k) = (\hat{\beta}_1^T(k), \hat{\beta}_2^T(k), ..., \hat{\beta}_p^T(k))^T$ where $0 < \eta_1 \leq ||\hat{\beta}_j(k)||_2 \leq \eta_2 < \infty,\ j = 1, ..., d$ and $||\hat{\beta}_j(k)||_2 = 0,\ j = d + 1, ..., p$ as $k \to \infty$.

Positive RIVAL, as presented in Chapter 4, may be modified to encourage grouping, and the resulting algorithm is named group positive RIVAL. Consider the group positive LASSO as described in (5.6) for given n, Y_n, and X_n. Assume $\frac{1}{n}X_n^T X_n > 0$. Let $0 \leq q(k) \leq 1$ be a sequence satisfying $\sum_{k=1}^{\infty} q(k) = \infty$. Then, group positive RIVAL is described by Algorithm 5.3.

The following theorem justifies the use of group positive RIVAL:

Theorem 5.2 (Group Positive RIVAL Convergence). *Consider the group positive RIVAL algorithm. Assume* $\frac{n}{t} \to \infty$, $t \to \infty$ *as* $n \to \infty$ *and* X_n *is block diagonal, i.e.,*

$$\frac{1}{n}X_n^T X_n = \begin{pmatrix} \mathbf{H_1} & 0 & \cdots & 0 \\ 0 & \mathbf{H_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{H_p} \end{pmatrix}, \ \mathbf{H_j} = \xi_j I \in \Re^{K_j \times K_j},\ \xi_j > 0$$

and I *is the identity matrix. Then, there is an integer* $n_0 > 0$*. For any* $n \geq n_0$*, there exists an integer* $k_0 > 0$ *such that the sequence* $\hat{\beta}_j(k)$ *generated by group positive RIVAL satisfies*

$$0 < \eta_1 \leq ||\hat{\beta}_j(k)||_2 \leq \eta_2 < \infty,\ \forall k \geq k_0,\ \ j = 1, ..., d$$

$$||\hat{\beta}_j(k)||_2 = 0,\ \forall k \geq k_0,\ \ j = d+1, ..., p$$

or equivalently,

$$A(k) = A^* \ \forall k \geq k_0$$

where $A(k)$ *is the estimated index set at stage* k*.*

The proof of Theorem 5.2 may be found in Chapter 8.

The sequence of weights $w_j(k)$'s generated by group positive RIVAL determines which groups of parameters, if any, are forced to zero and which remain non-zero. Applying the same weight w_j to all the estimates in the jth group encourages the selection of groups, as the weights $w_j(k)$'s corresponding to the non-zero groups $||\beta_j^*|| \neq 0$ are uniformly bounded for all k and the weights $w_j(k)$'s corresponding to zero groups of coefficients β_j^*, $j = d + 1, ..., p$ grow monotonically without bound until the corresponding group estimate $||\hat{\beta}_j||_2 = 0$. One could choose to initialize the weighting vector as $w_j(1) = 1/||\hat{\beta}_{LSj}||_1$ instead of setting $w_j(k) = 1,\ j = 1, ..., p$ to potentially increase the algorithm's speed, but only when n is large enough to ensure a reliable least squares estimate. In such a case, and if group positive RIVAL can capture the correct index set A^*, one can estimate $\beta^* =$

$(\beta_1^{*T}, ..., \beta_d^{*T})^T$ by applying a dimensionally-reduced ordinary least squares estimate $\hat{\beta}_{ALS} = (\hat{\beta}_{ALS1}^T, ..., \hat{\beta}_{ALSd}^T)^T$ by trimming off the zero groups $||\hat{\beta}_j||_2 = 0$, $j = d+1, ..., p$. Recall the subscript A indicates that the estimate $\hat{\beta}_{ALS}$ is A dependent and a result of group positive RIVAL. Then, the convergence of the estimates to their true values is guaranteed by the properties of the least squares estimate.

If $\frac{1}{n}X_n^T X_n > 0$ is diagonal with equal diagonal elements within a group and n is large enough, group positive RIVAL, in theory, has the perfect ability to find the correct index set. The use of group positive RIVAL, however, should not be limited to just this case; simulations show the algorithm may be applied successfully to a much larger class of X_n, as will be demonstrated in our nuclear material application where $\frac{1}{n}X_n^T X_n > 0$ is not diagonal and $n = 1024$. In fact, the utility of group positive RIVAL is easily demonstrated for a fixed and finite n, when the unique, self-adjusting ability of the algorithm is made apparent. Not to be outdone, group positive RIVAL has asymptotical set consistency with proper choices of t as n gets larger, and this is a result of the algorithm adjusting weights based on a convergent sequence of estimates. Note that the choice of $q(k)$ is not a critical one; a $q(k)$ close to 1 increases the speed of the algorithm, whereas a $q(k)$ close to 0 decreases sensitivity to noise.

Consider the nuclear detection experiment described in Section 3.2, i.e., a germanium type of detector with $n = 1024$ channels and a background consisting of radiation from local sources and cosmic rays. The 10 isotopes considered are listed in Table 5.1, along with their sub-spectra energy levels. An isotope's sub-spectrum is considered only if its branching ratio is larger than 0.005. The coefficients of the background and of I131's sub-spectra are set to 1, and the coefficients of Pu239's sub-spectra are set to 0.2. All other nuclear materials are absent and their group coefficients are set to 0. For such an experiment, the true unshielded parameter vector has the group form

$$\beta^0 = \Big(\underbrace{0,...,0}_{K_1=7}, \underbrace{0}_{K_2=1}, \underbrace{0,0}_{K_3=2}, \underbrace{0,...,0}_{K_4=4}, \underbrace{0,...,0}_{K_5=4}, \underbrace{0,...,0}_{K_6=6}, \underbrace{1,...,1}_{K_7=5}, \underbrace{0}_{K_8=1}, \underbrace{0,0}_{K_9=2}, \underbrace{0.2,...,0.2}_{K_{10}=10}, 1\Big)^T$$

so that the dimension of the problem is $m = 43$. The signal to noise ratio is calculated in a way similar to (3.5), but across all energy channels of all sub-spectra, i.e.,

$$SNR(\alpha) = 10\log \frac{\alpha\Big(\sum_{i=1}^{1024}(\sum_{s=1}^{10} 0.2Pu239_s(i) + \sum_{s=1}^{5} I131_s(i))\Big)}{\sum_{i=1}^{1024} Background(i)}. \quad (5.9)$$

I consider $\alpha = 1, 10, 20$ corresponding to unshielded SNRs of about $-11, -0.9, 2$ dB, though a shield will of course lower the SNR. Values of $\mu_{j,i}$ in (5.1) for carbon, concrete, lead, and water for the energy levels in Table 5.1 are retrieved from [29]. Mass thicknesses of $l = 10, 20, 30\ g/cm^2$ are considered, as well as $l = 0$, corresponding to the absence of a shield. After shielding, the true parameter vector is

$$\beta^* = D\beta^0 = \left(0^T, 0^T, 0^T, 0^T, 0^T, 0^T, \beta_7^{*T}, 0^T, 0^T, \beta_{10}^{*T}, 1\right)^T, \quad (5.10)$$

where

$$\beta_7^* = 1 \cdot \left(e^{-\mu_{7,1}l}, e^{-\mu_{7,2}l}, e^{-\mu_{7,3}l}, e^{-\mu_{7,4}l}, e^{-\mu_{7,5}l}\right)^T$$

and

$$\beta_{10}^* = 0.2 \cdot (e^{-\mu_{10,1}l}, e^{-\mu_{10,2}l}, e^{-\mu_{10,3}l}, e^{-\mu_{10,4}l}, e^{-\mu_{10,5}l},$$

$$e^{-\mu_{10,6}l}, e^{-\mu_{10,7}l}, e^{-\mu_{10,8}l}, e^{-\mu_{10,9}l}, e^{-\mu_{10,10}l})^T.$$

Notice that the background is left unshielded. The dimension of the zero vectors in (5.10) should be obvious from context. The SNR after the application of a shielding material with mass attenuation coefficients μ and a mass thickness l is

$$SNR(\alpha) = 10\log \frac{\alpha\Big(\sum_{i=1}^{1024}(\sum_{s=1}^{10} 0.2e^{-\mu_{10,s}l}Pu239_s(i) + \sum_{s=1}^{5} e^{-\mu_{7,s}l}I131_s(i))\Big)}{\sum_{i=1}^{1024} Background(i)}. \quad (5.11)$$

Table 5.1: The isotopes and corresponding sub-spectra energies involved in a shielding experiment.

Isotope	Sub-spectra energy peaks considered (keV)
Ba133	53.15, 79.6, 81, 276.4, 302.85, 356, 383.8
Ce139	165.86
Co57	122.06, 136.47
Co60	346.9, 1170, 1330, 2500
Cs137	31.8, 37.3, 283.5, 661.657
Ga67	91.27, 93.3, 184.58, 208.95, 300.22, 393.5
I131	80.185, 284.31, 364.489, 636.99, 722.911
K40	1460.83
Na22	511, 1274.53
Pu239	81.229, 84.214, 89.95, 109.16, 143.8, 163.3, 185.7, 194.9, 202.1, 205.3
Background	N/A

Though group positive RIVAL encourages grouping, it is certainly possible for the algorithm to find some sub-spectra and not others belonging to the same parent isotope. That is to say, for a given j, group positive RIVAL may produce a group estimate such that $\hat{\beta}_{ji} = 0$ and $\hat{\beta}_{jk} \neq 0$ for some i and k. It is clear that, with the addition of sub-spectra, some ambiguity exists in interpreting the results as present or absent. For this experiment, an isotope is considered present if just one of its sub-spectra is detected. As before, false positive error rate, false negative error rate, and correctly identified rate quantifies the algorithm's performance, and the optimal regularization parameter, t_{opt}, is chosen by AIC/BIC.

1000 simulations were run without shielding for several SNRs, and the average of the performances is illustrated in Figure 5.6. The algorithm works well for an SNR as low as -24 dB, but does not perform as well as in the non-group case, which performed well at -32 dB. The reason is because X_n becomes ill-conditioned as more sub-spectra are considered. Sub-spectra whose branching ratios are larger than 0.005 are included in X_n;

Algorithm 5.3 Group Positive RIVAL Algorithm

1. Let $w(1) = (w_1(1), ..., w_p(1))^T = (1, ..., 1)^T$. Set $k = 1$.

2. Apply the group positive LASSO with respect to the weight vector $w(k)$ and denote the solution corresponding to t as $\hat{\beta}(k) = (\hat{\beta}_1^T(k), \hat{\beta}_2^T(k), ..., \hat{\beta}_p^T(k))^T$, where $\hat{\beta}_j^T(k) = (\hat{\beta}_{j1}, \hat{\beta}_{j2}, ..., \hat{\beta}_{jK_j})$.

3. For any j such that $||\hat{\beta}_j(k)||_2 = 0$, set $\hat{\beta}_j(k+\kappa) = 0$ for all $\kappa \geq 0$. Remove $\hat{\beta}_j$ and w_j from β and w, respectively,
$$\begin{pmatrix} \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_{j-1} \\ \hat{\beta}_j \\ \hat{\beta}_{j+1} \\ \vdots \\ \hat{\beta}_p \end{pmatrix} \to \begin{pmatrix} \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_{j-1} \\ \hat{\beta}_{j+1} \\ \vdots \\ \hat{\beta}_p \end{pmatrix}, \begin{pmatrix} w_1 \\ \vdots \\ w_{j-1} \\ w_j \\ w_{j+1} \\ \vdots \\ w_p \end{pmatrix} \to \begin{pmatrix} w_1 \\ \vdots \\ w_{j-1} \\ w_{j+1} \\ \vdots \\ w_p \end{pmatrix}.$$
Also remove the corresponding group of regressors X_{jn} from X_n so the dimension of the optimization is reduced by K_j. If $||\hat{\beta}_j(k)||_2 > 0$, update the corresponding weights as
$w_j(k+1) = q(k) \cdot \frac{1}{\sum_{i=1}^{K_j} \hat{\beta}_{ji}(k)} + (1 - q(k))\frac{w_j(k)}{\xi_j}$,
where ξ_j is the average of the norms of each column of $\frac{1}{n}X_{jn}^T X_{jn}$.

4. Set $k = k + 1$ and go back to Step 2. The dimension could be reduced if some of $\hat{\beta}_j = 0$ at Step 3. (This step may be modified to add the stopping criterion using the standard one in numerical analysis, e.g., the iterations stop if $||\hat{\beta}(k+1) - \hat{\beta}(k)||_2/||\hat{\beta}(k)||_2$ is smaller than the prescribed threshold).

this number could be raised to improve performance and omit certain sub-spectra, but in reality, shielding always exists, and hence, it is important to consider a reasonable amount of sub-spectra. Thus, the absence of a shield is purely a theoretical situation and is only considered to provide a best case scenario for comparison purposes.

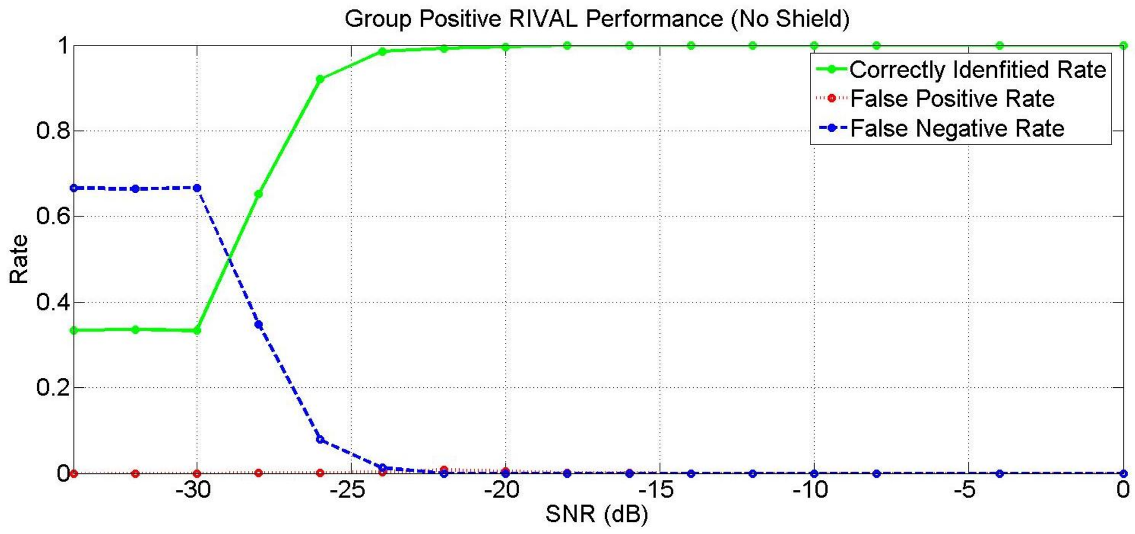


Figure 5.6: Group positive RIVAL performance versus various SNRs for the unshielded case.

Simulations were run with different combinations of shielding materials, thicknesses, and nuclear material strengths. The simulations were carried out using the MATLAB code master.m (see Section 9.1) which calls applyShield.m (see Section 9.2) to apply a shield to the nuclear materials as according to the above analysis. For each combination of shield parameters, 100 simulations were run, and the average of the results are shown in Table 5.2. The algorithm works well for carbon, concrete, and water shielding materials. However, the

Table 5.2: Group positive RIVAL performance under various shielding schemes: FPR = false positive rate, FNR = false negative rate, and CIR = correctly identified rate.

α	Shield material	l (g/cm^2)	Thickness (cm)	SNR (dB)	FPR	FNR	CIR
20	Carbon	10	3.82	-3.108	0.0000	0.0000	1.0000
		20	7.63	-8.057	0.0000	0.0000	1.0000
		30	11.5	-12.81	0.0000	0.0000	1.0000
	Concrete	10	4.35	-3.491	0.0000	0.0000	1.0000
		20	8.70	-8.601	0.0000	0.0000	1.0000
		30	13.0	-13.42	0.0000	0.0000	1.0000
	Lead	10	0.882	-11.06	0.0000	0.3333	0.6667
		20	1.76	-19.71	0.0000	0.3333	0.6667
		30	2.65	-26.69	0.0000	0.3333	0.6667
	Water	10	10.0	-3.709	0.0000	0.0000	1.0000
		20	20.0	-9.179	0.0000	0.0000	1.0000
		30	30.0	-14.40	0.0000	0.0000	1.0000
10	Carbon	10	3.82	-6.118	0.0000	0.0000	1.0000
		20	7.63	-11.07	0.0000	0.0000	1.0000
		30	11.5	-15.82	0.0000	0.0000	1.0000
	Concrete	10	4.35	-6.501	0.0000	0.0000	1.0000
		20	8.70	-11.61	0.0000	0.0000	1.0000
		30	13.0	-16.43	0.0000	0.0000	1.0000
	Lead	10	0.882	-14.07	0.0000	0.3333	0.6667
		20	1.76	-22.72	0.0000	0.3333	0.6667
		30	2.65	-29.70	0.0000	0.3333	0.6667
	Water	10	10.0	-6.719	0.0000	0.0000	1.0000
		20	20.0	-12.19	0.0000	0.0000	1.0000
		30	30.0	-17.41	0.0000	0.0000	1.0000
1	Carbon	10	3.82	-16.12	0.0000	0.0000	1.0000
		20	7.63	-21.07	0.0000	0.0000	1.0000
		30	11.5	-25.82	0.0000	0.3333	0.6667
	Concrete	10	4.35	-16.50	0.0000	0.0000	1.0000
		20	8.70	-21.61	0.0013	0.0100	0.9887
		30	13.0	-26.43	0.0000	0.3333	0.6667
	Lead	10	0.882	-24.07	0.0000	0.3333	0.6667
		20	1.76	-32.72	0.0000	0.6667	0.3333
		30	2.65	-39.70	0.0000	0.6667	0.3333
	Water	10	10.0	-16.72	0.0000	0.0000	1.0000
		20	20.0	-22.19	0.0000	0.0633	0.9367
		30	30.0	-27.41	0.0000	0.3333	0.6667

algorithm struggles to find the materials under lead shielding for an SNR as high as -11 dB. I took a closer look at lead by simulating with a finer grid of smaller mass thicknesses; this time with $l = 1, 2, 3, 4, 4.25, 4.5, 4.75, 5$ and $5.25\ g/cm^2$ for an intermediate signal strength of $\alpha = 10$. As shown in Figure 5.7, there is a steep drop off of performance at $l = 5\ g/cm^2$ most likely due to lead's large density.

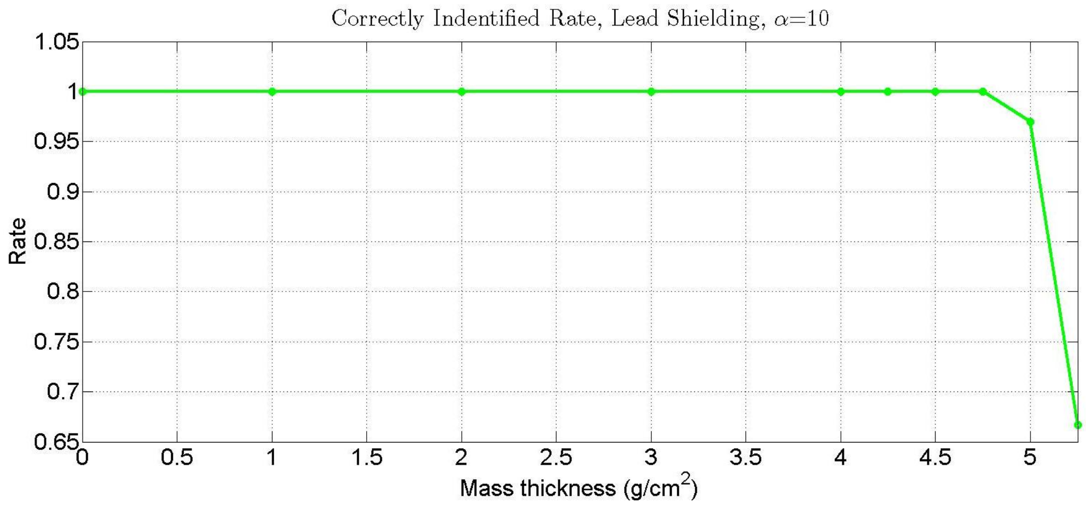


Figure 5.7: Correctly identified rates of group positive RIVAL for lead shielding of various mass thicknesses.

CHAPTER 6
DETECTION WITH AN INEXACT LIBRARY

Until this point, it has been assumed that the library X_n is known perfectly; that is, the template of isotope spectra and sub-spectra is exact. In some respects, this is true; branching ratios are known to at least 99.99% accuracy, and it is not uncommon for a detector to have its own library to account for specific detector characteristics [10]. However, in the less than perfect world, there exist errors caused by so-called detector drift, and these errors can be modeled as an uncertainty in X_n. Detectors require high voltages to operate, and so electronics get hot, and the net result is an inaccurate energy reading. For instance, a 1.33 MeV photon may be reported as 1.30 MeV or 1.40 MeV. Yet another possible source of library error is from fluctuations in background radiation.

Consider the non-group case, and call the uncertainty in the library ΔX_n, with unknown regressors $\mathbf{\Delta x}_j$. Then the detection model (2.14) becomes

$$Y_n = (X_n + \Delta X_n)\beta^* + V_n,$$

or equivalently

$$Y_n = \sum_{j=1}^{p}(\mathbf{x}_j + \mathbf{\Delta x}_j)\beta_j^* + V_n.$$

Rearrange indices and partition $\Delta X_n = (\Delta X_{n1}\ \Delta X_{n2})$, where $\Delta X_{n1} = (\mathbf{\Delta x}_1\ ...\ \mathbf{\Delta x}_d)$ and $\Delta X_{n2} = (\mathbf{\Delta x}_{p-d}\ ...\ \mathbf{\Delta x}_p)$. For large enough n and a fixed ΔX_n, the irrepresentable condition for positive LASSO (4.2), using a binomial expansion, (approximately) becomes

$$a_{nj} \leq w_{2j} - \epsilon_n,\ j = 1, ..., p - d \tag{6.1}$$

where

$$a_n = c_n^T(1,2)\left[c_n(1,1) + \frac{\Delta X_{n1}\Delta X_{n2}}{n}\right]^{-1}\begin{pmatrix} w_{11} \\ \vdots \\ w_{1d}\end{pmatrix} \in \Re^{p-d},$$

and C_n and ϵ_n are as defined as before.

Condition (6.1) is not extremely useful, as ΔX_n may or may not be fixed and is unknown, hence a new method is needed. For unknown ΔX_n, total least squares (TLS) ([15], [19], [45]) is a least squares data modeling technique that takes both the errors ΔX_n and V_n into account. In one dimension, TLS works to minimize the perpendicular distance between the model and the data points, while ordinary least squares minimizes the distance parallel to the y-axis, as illustrated in Figure 6.1. Note that the TLS distance is perpendicular only when x and y have equal variances.

Specifically, the TLS estimate is the triple $(\hat{\beta}, \hat{Er}, \hat{e})$ that solves

$$\min_{\beta, Er, e} ||[Er, e]||_F^2 \tag{6.2}$$

$$s.t. \;\; Y_n + e = (X_n + Er)\beta,$$

where $[Er, e]$ represents matrix Er augmented with vector e, and $||\cdot||_F$ is the Frobenius norm [16]. Much like ordinary least squares, the TLS estimate generally gives all non-zero elements and thus will need to be regularized for variable selection applications like nuclear material detection.

6.1 Regularized Total Least Squares

Consider the TLS minimization problem (6.2) with a positive LASSO-type regularization, i.e.,

$$\min_{\beta \geq 0, Er, e} \left\{ ||[Er, e]||_F^2 + t\sum_{j=1}^{p} w_j\beta_j \right\} \tag{6.3}$$

$$s.t. \;\; Y_n + e = (X_n + Er)\beta.$$

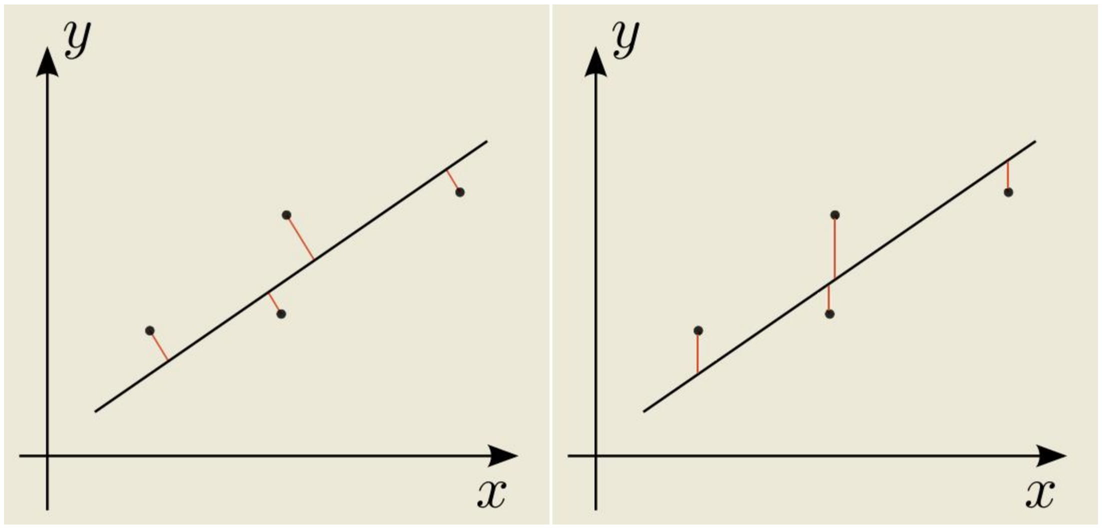


Figure 6.1: TLS accounts for errors in the independent variable as well as the dependent variable by minimizing the perpendicular distance between the model and the data (left). This is different than ordinary least squares which only accounts for error in the dependent variable (right).

It should be expected that the model produced by (6.3), with a properly chosen regularization parameter t, is sparse and less sensitive to library noise than positive LASSO. The regularized TLS problem (6.3) is non-convex, and so no efficient convex solver can guarantee to achieve the global minimum. Notice that e follows a fixed relationship with Er and β, so finding the optimum triple $(\hat{\beta}, \hat{Er}, \hat{e})$ is equivalent to finding the optimum pair $(\hat{\beta}, \hat{Er})$. Also notice that for a given Er, problem (6.3) reduces to the positive LASSO problem – a convex problem that can be solved efficiently with Algorithm 4.1 or by modifying Algorithm 2.2. Further, when β is fixed, problem (6.3) is convex in Er. This type of problem is called a biconvex problem [17], and, though no convergence results can be guaranteed, it can be solved using an alternating convex search algorithm [45]. Categorized as such, Algorithm 6.1 is specific to solving the regularized TLS problem.

Algorithm 6.1 Total Least Squares with Positive LASSO Algorithm

1. Let $\hat{Er}(k)$ and $\hat{\beta}(k)$ denote the estimates of ΔX_n and β^*, respectively, at iteration k. Initialize the algorithm with $\hat{Er}(1) = 0_{n\times p}$. Set $k = 1$.
2. For the given $\hat{Er}(k)$, X_n, Y_n, w and t, use a convex solver, e.g., Algorithm 4.1, to solve the positive LASSO problem
   $\hat{\beta}(k) = arg\min_{\beta\geq 0}\left\{||Y_n - [X_n + \hat{Er}(k)]\beta||_2^2 + t\sum_{j=1}^{p} w_j\beta_j\right\}$.
3. Update $\hat{Er}$ according to $\hat{Er}(k+1) = [Y_n - X_n\hat{\beta}(k)]\hat{\beta}^T(k)[I + \hat{\beta}(k)\hat{\beta}^T(k)]^{-1}$.
4. If the stopping criteria is met, stop. If not, set $k = k+1$, and go back to Step 2.

Similarly, TLS may be combined with positive RIVAL, for which Algorithm 6.2 provides a description.

I would like to make a few comments regarding Algorithms 6.1 and 6.2:

- The update $\hat{Er}(k+1)$ in Step 3 is from substituting the constraint $Y_n + e = (X_n + Er)\hat{\beta}$ back into the cost function for given
  $hatbeta$, and setting the first-order derivative of the cost function with respect to Er to zero.
- One full iteration includes updating $\hat{\beta}$ while holding Er fixed, and updating Er while holding β fixed. Both updates either maintain or improve, but do not worsen, the cost function in (6.3). Thus, the monotonic convergence of the bounded, non-negative

Algorithm 6.2 Total Least Squares with Positive RIVAL Algorithm

```
1. Let $\hat{Er}(k)$ and $\hat{\beta}(k)$ denote the estimates of $\Delta X_n$ and $\beta^*$, respectively, at iteration $k$. Initialize the algorithm with $\hat{Er}(1) = 0_{n\times p}$. Set $k = 1$.
2. Let $X_n^{**}(k) = X_n + \hat{Er}(k)$. Replace $X_n$ with $X_n^{**}(k)$ in Algorithm 4.3, and use it to find the positive RIVAL estimate $\hat{\beta}(k)$.
3. Update $\hat{Er}$ according to $\hat{Er}(k+1) = [Y_n - X_n\hat{\beta}(k)]\hat{\beta}^T(k)[I + \hat{\beta}(k)\hat{\beta}^T(k)]^{-1}$.
4. If the stopping criteria is met, stop. If not, set $k = k + 1$, and go back to Step 2.
```

cost function is established.

- The convergence of β and Er to their true values β^* and ΔX_n, respectively, is not guaranteed, and will most likely depend on the initialization of Er. Setting $Er(1) = 0_{n\times p}$ is a good choice in that the first iteration gives the positive LASSO solution.
- The stopping criterion may be one common to numerical analysis, e.g., stop the iteration if $||\hat{\beta}(k+1) - \hat{\beta}(k)||_2/||\hat{\beta}(k)||_2$ is smaller than a prescribed threshold.

6.2 Positive RIVAL with TLS for Detection

With uncertainty in the library, it is expected that the combination of positive RIVAL and total least squares will perform better than positive RIVAL alone. To test this, uncertainty is added to the library, and Algorithms 6.2 and 4.3 are applied to the usual scenario, i.e., the isotopes in Table 3.1 are considered, and I131 and Pu239, along with the background, are present. Pu239 has a strength one-fifth that of I131 and the background. Two methods to model the uncertainty in the library are considered. The first is a case where the uncertainty of each library component is an independent random variable whose variance is related to the corresponding unperturbed library component. The second case has a more structured uncertainty; the uncertainty is such that the perturbed library is a compressed version of the unperturbed one. The latter case arises due to the aforementioned detector drift, while the former case is meant as a catch-all.

For the first case, recall from Section 3.1 that the library X_n with components x_{ij} is the mean gamma-ray counts per unit time and per unit source material *that has been scaled* by detector measurements to reduce the effect of inhomogeneous noise variance. The scaled uncertainty in the spectrum of the jth isotope is modeled as

$$\Delta x_{ij} \sim N(0, \gamma \cdot x_{ij}), \ i = 1, ..., n$$

The parameter $\gamma \geq 0$ represents the fractional change in the scaled library. Then, the unscaled uncertainty $\Delta\tilde{x}_{ij}$ follows

$$\Delta\tilde{x}_{ij} \sim N(0, \gamma \cdot \tilde{x}_{ij} \cdot \sqrt{z_i}), \ i = 1, ..., n$$

where $\tilde{x}_{ij}$ is the unscaled mean gamma-ray counts at the ith channel of the jth isotope, and z_i is from the detector measurements. Thus, the perturbed spectrum of the jth isotope is a random vector that follows

$$\tilde{x}_{ij} + \Delta\tilde{x}_{ij} \sim N(\tilde{x}_{ij}, \gamma \cdot \tilde{x}_{ij} \cdot \sqrt{z_i}), \ i = 1, ..., n.$$

The unperturbed spectrum of Pu239 and three realizations of its perturbed spectrum with a fractional change $\gamma = 0.10$ are shown in Figure 6.2. The noise is more prominent when the counts are smaller because the figure is plotted on a semi-log scale. A small disturbance is easily observed when the counts are around 10^2, but this same disturbance is not observable when counts are high, i.e., 10^4 and above.

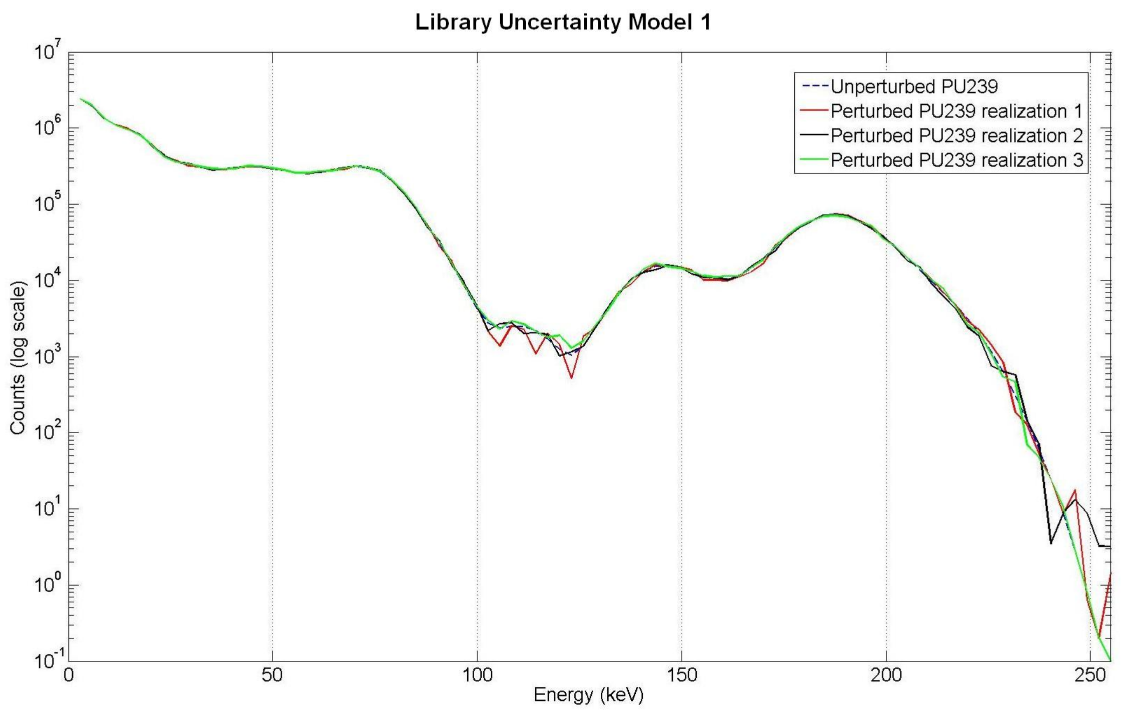


Figure 6.2: The first of two methods of modeling the uncertainty in X_n illustrated on the spectrum of Pu239 for an error of 10%. Each point on the perturbed spectrum is an independent random variable with a mean equal to the unperturbed spectrum at that point.

For the second case, $\gamma \geq 0$ is the factor in which the spectrum gets compressed along the energy (horizontal) axis. In generating the perturbed spectra, the compressed energy channels are rounded to the nearest integer value. Then, the compressed spectra are scaled so that each spectrum has the same total energy as its unperturbed counterpart. The unperturbed spectrum of Pu239 and its perturbed spectrum with a 10% compression are shown in Figure 6.3.

1000 simulations were run with $\alpha = 10$ in (3.5) so that the SNR is about -7 dB. The

SNR is still as defined as before, i.e., it is not a function of the library uncertainty. For each simulation, and using the MATLAB code TLSlassoNuclear2.m available in Section 9.8, both positive RIVAL (Algorithm 4.3) and TLS with positive RIVAL (Algorithm 6.2) are executed over a grid of regularization parameters t, and the optimal t is chosen by minimizing BIC. Recall that positive RIVAL does not have parameter convergence, but one can use the set A generated by positive RIVAL to trim off the irrelevant regressors and then perform a least squares estimate. The set A is, of course, a function of the regularization parameter t, and so in the analysis to come, let $X_{A(t)n}$, $\hat{Er}_{A(t)}$, and $\hat{\beta}_{A(t)LS}$ be the reduced X_n, $\hat{Er}$, and ordinary least squares estimate, respectively, based on the set found by positive RIVAL. Algorithm 6.2 has the added benefit of estimating the library uncertainty which can be used in calculating the BIC value. More precisely, the algorithm estimates the library uncertainty ΔX_n as $\hat{Er}$ and generates the model estimate $\hat{\beta}$ based on the regressor matrix $X_n + \hat{Er}$. Therefore, the BIC value is calculated as

$$BIC(t) = n \cdot \ln \frac{||Y_n - (X_{A(t)n} + \hat{Er}_{A(t)})\hat{\beta}_{A(t)LS}||_2^2}{n} + \ln(n) \cdot q(t).$$

The optimal t according to BIC is then

$$t_{opt} = arg \min_t \left\{ n \cdot \ln \frac{||Y_n - (X_{A(t)n} + \hat{Er}_{A(t)})\hat{\beta}_{A(t)LS}||_2^2}{n} + \ln(n) \cdot q(t) \right\}.$$

To measure the performance, use the old metrics of false positive (FPR), false negative (FNR), and correctly identified (CIR) rates, and define a new metric:

Frequency of Correct Model (FoCM): The algorithm finds all the isotopes that are present and none which are absent.

Table 6.1: The performance of total least squares with positive RIVAL as compared to that of positive RIVAL alone for different library uncertainties modeled as in case 1.

γ_I	γ_B	Method	FPR	FNR	CIR	FoCM
0.01	0.05	Positive RIVAL	0.0024	0.0030	0.9946	0.9830
		TLS + Positive RIVAL	0.0010	0.0023	0.9967	0.9910
	0.10	Positive RIVAL	0.0084	0.0023	0.9893	0.9330
		TLS + Positive RIVAL	0.0045	0.0033	0.9922	0.9630
	0.20	Positive RIVAL	0.0207	0.0030	0.9763	0.8380
		TLS + Positive RIVAL	0.0087	0.0037	0.9876	0.9270
	0.30	Positive RIVAL	0.0260	0.0010	0.9730	0.8050
		TLS + Positive RIVAL	0.0099	0.0030	0.9871	0.9190
0.05	0.10	Positive RIVAL	0.0201	0.0040	0.9759	0.8320
		TLS + Positive RIVAL	0.0082	0.0050	0.9868	0.9330
	0.20	Positive RIVAL	0.0320	0.0030	0.9635	0.7630
		TLS + Positive RIVAL	0.0120	0.0050	0.9830	0.9020
	0.30	Positive RIVAL	0.0368	0.0033	0.9599	0.7240
		TLS + Positive RIVAL	0.0177	0.0050	0.9773	0.8600
0.10	0.20	Positive RIVAL	0.0493	0.0040	0.9467	0.6350
		TLS + Positive RIVAL	0.0176	0.0070	0.9754	0.8590
	0.30	Positive RIVAL	0.0417	0.0023	0.9560	0.6870
		TLS + Positive RIVAL	0.0184	0.0080	0.9736	0.8520
0.15	0.20	Positive RIVAL	0.0524	0.0030	0.9446	0.6070
		TLS + Positive RIVAL	0.0206	0.0070	0.9724	0.8340
	0.30	Positive RIVAL	0.0546	0.0037	0.9417	0.6080
		TLS + Positive RIVAL	0.0209	0.0083	0.9708	0.8400

Table 6.2: For the second noise model, the number of data n decreases as a result of compressing the spectra.

	$\gamma = 0.01$	$\gamma = 0.03$	$\gamma = 0.05$	$\gamma = 0.10$
$n * (1-\gamma)$	1013.76	993.28	972.8	921.6
n after truncating	1014	993	973	922

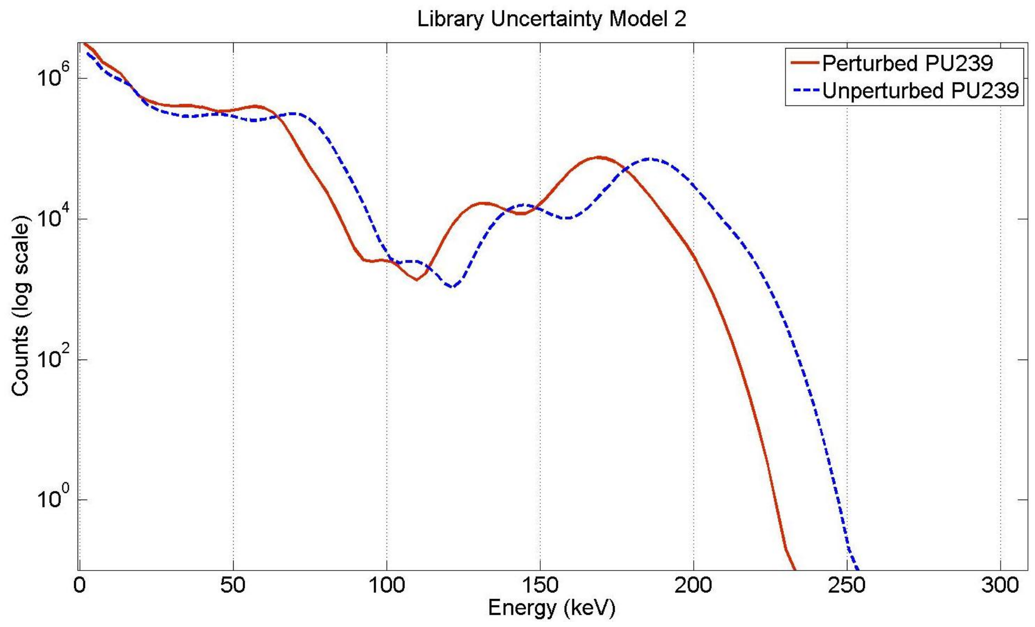


Figure 6.3: The second of two methods of modeling the uncertainty in X_n illustrated on the spectrum of Pu239 for an error of 10%. The perturbed spectrum is a compressed version of the original spectrum. Such an error is a characteristic of detector drift.

CIR and FoCM provide different ways or measuring the correctness of the estimated model. To illustrate this, consider a simple example where the true $\beta^* = (1, 0, 1, 0)^T$ and $A^* = \{1, 3\}$. Suppose the estimate is $\hat{\beta} = (0.9, 0.1, 0.9, 0)^T$ and $A = \{1, 2, 3\}$. Then, the FPR is $1/2 = 0.50$, the FNR is $0/2 = 0$, and the CIR is $1 - (FPR + FNR) = 0.50$. The FoCM in this case is zero because the estimated set A does not match the true set A^*.

Table 6.1 shows the average performance results of both algorithms for different fractional uncertainties γ when the uncertainty is modeled as in the first case. For such a case, one might expect more uncertainty in the background part of the library than that in the isotope part. To distinguish between the two uncertainties, a subscript has been added to γ: The isotope uncertainty is γ_I and the background uncertainty is γ_B. For the harshest condition simulated, $\gamma_I = 0.15$ and $\gamma_B = 0.30$, TLS with positive RIVAL finds the correct model almost 25% more often that positive RIVAL alone.

For the second noise model, one would expect the error in the background part of the library to be the same as the error in the isotope part. After all, the noise model is meant to mimic detector drift, and detector drift is due to non-ideal electronics inside the detector hardware that should produce the same error regardless of the source of radiation. Therefore, simulations were only considered for $\gamma_I = \gamma_B = \gamma$. Further, since compressing the spectra has the effect of leaving the high energy-ends undefined, each spectrum was truncated after the last defined energy bin, leaving the number of data $n < 1024$ as shown in Table 6.2.

The number of detections of each isotope in 1,000 simulations of the second noise model are shown in Table 6.3. Careful inspection of Table 6.3 reveals the unique characteristic that, for a given γ, the same model was selected for every run of each group of 1,000 simulations. This can easily be explained in the following way. At -7 dB, the SNR was set relatively high in order to isolate the effect that the library uncertainty has on Algorithms 4.3 and 6.2. Recall from Chapter 4 that positive RIVAL would otherwise perform flawlessly at this SNR. For each group of 1,000 simulations, γ is fixed, and the library is not a random variable as it was in the first case. Therefore, for a given γ, the algorithms will always find the same model. The FoCM will therefore either be 0 or 1, depending on γ, as shown in Table 6.4.

Both methods perform perfectly when $\gamma = 0.01,\ 0.03$. When $\gamma = 0.05$, it is clear that detection benefits from the addition of TLS to the positive RIVAL algorithm because

Table 6.3: The number of isotope detections by positive RIVAL and TLS + positive RIVAL in 1,000 simulations for different library uncertainties modeled as case 2.

γ	Isotope	Present	Positive RIVAL	TLS + Positive RIVAL
0.03	Pu239	1000	1000	1000
	Ga67	0	0	0
	Cs137	0	0	0
	U235	0	0	0
	K40	0	0	0
	Na22	0	0	0
	Ba133	0	0	0
	Ce139	0	0	0
	I1317	1000	1000	1000
	Co57	0	0	0
	Co60	0	0	0
	Background	1000	1000	1000
0.05	Pu239	1000	0	1000
	Ga67	0	0	0
	Cs137	0	0	0
	U235	0	1000	0
	K40	0	0	0
	Na22	0	0	0
	Ba133	0	1000	0
	Ce139	0	0	0
	I1317	1000	1000	1000
	Co57	0	0	0
	Co60	0	0	0
	Background	1000	1000	1000
0.10	Pu239	1000	0	1000
	Ga67	0	1000	0
	Cs137	0	1000	1000
	U235	0	1000	0
	K40	0	0	1000
	Na22	0	0	0
	Ba133	0	0	0
	Ce139	0	0	0
	I1317	1000	1000	1000
	Co57	0	0	0
	Co60	0	0	0
	Background	1000	1000	1000

Table 6.4: The performance of total least squares with positive RIVAL as compared to that of positive RIVAL alone for different library uncertainties modeled as in case 2.

γ	Method	FPR	FNR	CIR	FoCM
0.01	Positive RIVAL	0.0000	0.0000	1.0000	1.0000
	TLS + Positive RIVAL	0.0000	0.0000	1.0000	1.0000
0.03	Positive RIVAL	0.0000	0.0000	1.0000	1.0000
	TLS + Positive RIVAL	0.0000	0.0000	1.0000	1.0000
0.05	Positive RIVAL	0.2222	0.3333	0.4444	0.0000
	TLS + Positive RIVAL	0.0000	0.0000	1.0000	1.0000
0.10	Positive RIVAL	0.3333	0.3333	0.3333	0.0000
	TLS + Positive RIVAL	0.2222	0.0000	0.7778	0.0000

positive RIVAL alone fails to find the correct model, while TLS with positive RIVAL is successful. All is not lost when $\gamma = 0.10$. Though TLS with positive RIVAL has a FoCM of zero, the algorithm did, in fact, generate correct paths for the simulations; AIC and BIC failed to select the correct model. This is more than what can be said for positive RIVAL, which never generated correct paths. Perhaps cross validation or some other method could prove to be useful in selecting a model from a path generated by TLS with positive RIVAL, but there is no hope for paths generated by positive RIVAL.

The total least squares method can also be used in conjunction with group positive LASSO and group positive RIVAL. If the jth isotope has K_j sub-spectra, $j = 1, ..., p$, i.e., an estimate of β^* is

$$\hat{\beta} = \begin{pmatrix} \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_p \end{pmatrix}, \; \hat{\beta}_j = (\hat{\beta}_{j1}, ..., \hat{\beta}_{jK_j})^T, \; j = 1, ..., p$$

Algorithm 6.3 Total Least Squares with Group Positive LASSO Algorithm

1. Let $\hat{Er}(k)$ and $\hat{\beta}(k)$ denote the estimates of ΔX_n and β^*, respectively, at iteration k. Initialize the algorithm with $\hat{Er}(1) = 0_{n\times p}$. Set $k = 1$.
2. For the given $\hat{Er}(k)$, X_n, Y_n, w and t, use a convex solver, e.g., Algorithm 5.1, to solve the group positive LASSO problem
   $\hat{\beta}(k) = arg\min_{\beta\geq 0}\left\{||Y_n - [X_n + \hat{Er}(k)]\beta||_2^2 + t\sum_{j=1}^{p} w_j \sum_{i=1}^{K_j}\beta_{ji}\right\}$.
3. Update $\hat{Er}$ according to $\hat{Er}(k+1) = [Y_n - X_n\hat{\beta}(k)]\hat{\beta}^T(k)[I + \hat{\beta}(k)\hat{\beta}^T(k)]^{-1}$.
4. If the stopping criteria is met, stop. If not, set $k = k+1$, and go back to Step 2.

then total least squares with a group positive LASSO-type penalty is the minimization

$$\min_{\beta\geq 0, Er, e}\left\{||Er, e||_F^2 + t\sum_{j=1}^{p} w_j \sum_{i=1}^{K_j}\beta_{ji}\right\} \tag{6.4}$$

$$s.t. \;\; Y_n + e = (X_n + Er)\beta.$$

Notice that problem (6.4) is once again a biconvex minimization and may be solved with an alternating convex search algorithm. Algorithm 6.3 provides the details for solving TLS with group positive LASSO. Further, Algorithm 6.4 describes TLS with group positive RIVAL.

To test Algorithm 6.4, 100 simulations were run with an SNR of -15 dB as in (5.9). All other simulation details were the same as the non-group simulations, including the two cases in which library uncertainty was added to the model. Algorithms 5.3 and 6.4 were implemented with the MATLAB code TLSMaster.m in Section 9.9, and the results for case 1 are shown in Table 6.5. The total least squares method works better in every scenario,

Algorithm 6.4 Total Least Squares with Group Positive RIVAL Algorithm

1. Let $\hat{Er}(k)$ and $\hat{\beta}(k)$ denote the estimates of ΔX_n and β^*, respectively, at iteration k. Initialize the algorithm with $\hat{Er}(1) = 0_{n\times p}$. Set $k = 1$.
2. Let $X_n^{**}(k) = X_n + \hat{Er}(k)$. Replace X_n with $X_n^{**}(k)$ in Algorithm 5.3, and use it to find the group positive RIVAL estimate $\hat{\beta}(k)$.
3. Update $\hat{Er}$ according to $\hat{Er}(k+1) = [Y_n - X_n\hat{\beta}(k)]\hat{\beta}^T(k)[I + \hat{\beta}(k)\hat{\beta}^T(k)]^{-1}$.
4. If the stopping criteria is met, stop. If not, set $k = k + 1$, and go back to Step 2.

especially when $\gamma_I = 0.02$ and $\gamma_B = 0.10$; in this case, the total least squares method finds the correct model more than 35% more than group positive RIVAL alone.

The algorithm did not perform as strongly for the second case, where the library uncertainty had to be drastically reduced to get any interpretable results. As shown in Table 6.6, group positive RIVAL and TLS with group positive RIVAL both perform well for $\gamma = 0.001$. When $\gamma = 0.003$, the TLS algorithm finds the correct set every time, while the group positive RIVAL never finds it. Neither algorithm finds the correct set when $\gamma = 0.005$. There are many possible explanations for this outcome, but I believe the cause to be a numerical issue. Decomposing spectra into sub-spectra increases the dimension of the problem and, given the fact that the algorithm must be implemented for many different values of t, this challenges computer processing power to the point where the grid of t's may not have been simulated as fine as it needed to be without having processing issues. It was always assumed that the grid of t's was sufficient enough to contain the correct model. This may have been the case for the first model of library uncertainty but is not the case for

Table 6.5: The performance of total least squares with group positive RIVAL as compared to that of group positive RIVAL alone for different library uncertainties modeled as in case 1: FPR = false positive rate, FNR = false negative rate, CIR = correctly identified rate, and FoCM = frequency of correct model.

γ_I	γ_B	Method	FPR	FNR	CIR	FoCM
0.01	0.01	Group positive RIVAL	0.0013	0.0000	0.9988	0.9900
		TLS + Group positive RIVAL	0.0000	0.0000	1.0000	1.0000
	0.02	Group positive RIVAL	0.0013	0.0000	0.9988	0.9900
		TLS + Group positive RIVAL	0.0000	0.0000	1.0000	1.0000
	0.05	Group positive RIVAL	0.0413	0.0033	0.9554	0.7200
		TLS + Group positive RIVAL	0.0125	0.0000	0.9875	0.9000
0.02	0.10	Group positive RIVAL	0.0688	0.0067	0.9246	0.5200
		TLS + Group positive RIVAL	0.0125	0.0000	0.9875	0.8800

Table 6.6: The performance of total least squares with group positive RIVAL as compared to that of group positive RIVAL alone for different library uncertainties modeled as in case 2.

γ	Method	FPR	FNR	CIR	FoCM
0.001	Group positive RIVAL	0.0000	0.0000	1.0000	1.0000
	TLS + Group positive RIVAL	0.0000	0.0000	1.0000	1.0000
0.003	Group positive RIVAL	0.0000	0.3333	0.6667	0.0000
	TLS + Group positive RIVAL	0.0000	0.0000	1.0000	1.0000
0.005	Group positive RIVAL	0.2500	0.3333	0.4167	0.0000
	TLS + Group positive RIVAL	0.2500	0.3333	0.4167	0.0000

the second model. Recall for the first model, the library uncertainty needed to be reduced from that of the non-group case. Even for some fixed t's, the group algorithm was unable to be simulated because of computer memory issues. Thus, the results of when $\gamma = 0.005$ should be viewed as inconclusive.

CHAPTER 7
CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

7.1 Summary and Conclusions

Contemporary radioactive isotope detection algorithms in less-than-ideal situations, such as at U.S. ports of entry where detectors are large and far away from the materials in question, are not reliable means of detection due to the poor resolution characteristic of large detectors, and from the corruption of the isotopes' nuclear signatures (i.e., their respective gamma-ray spectra) by background radiation from naturally occurring radioactive materials. The methods of peak detection, energy windowing, and LASSO were analyzed and demonstrated to give lackluster performances, especially in an experiment where a trace amount of special nuclear material used for nuclear weapons is present and in the shadow of a more dominant, but less threatening, radioactive isotope such as I131. From the literature survey I conducted in Chapter 2, it is obvious that the energy windowing technique fails to separate the special nuclear material from the less threatening isotope, because the energy windowing technique, by definition, only produces anomaly warnings and fails to classify the isotopes. I demonstrated in Chapter 3 that, even with a modest signal to noise ratio of -10 dB, traditional peak detection methods also fail to separate the special nuclear material from the I131, and LASSO has an impractically high false positive error rate.

In Section 3.1, I began the development of new physics-based detection algorithms by describing how to model detection as a linear equation. Physics provides the regressor library X_n – a template describing the expected gamma-ray energy counts for a collection of isotopes per unit time and per unit source material – but the actual counts are random and follow a Poisson distribution. Then, the difference between the actual counts and the expected counts is considered random noise V_n, with noise at each energy channel independent from the next. The model

$$Y_n = X_n\beta^* + V_n$$

is the basic model used for detection, where β^* is a vector representing the true but unknown isotope intensities, and Y_n is known from the detector. Thus, detection is essentially a linear regression problem for which it is desirable to determine which components of β^* are zero and which are not, corresponding to isotopes being absent or present. Since LASSO was created for just this purpose, and a substantial amount of research has been conducted on LASSO accordingly, it was an attractive option upon which to build detection algorithms, though LASSO, by itself, fails to solve the specific problem of nuclear material detection when energy spectra are weak and poorly resolved.

There are two problems concerning the application of LASSO to nuclear material detection. The first is that the detector data may or may not satisfy the irrepresentable condition described in Section 2.3.2, hence, LASSO may or may not have set consistency. The second is that most of the LASSO results established from various authors in various works are asymptotical, i.e., the results hold only when the number of detector channels n goes to infinity. The implication of these problems is that few LASSO results hold for the detection problem when n is fixed and finite, and LASSO may have a high false alarm rate.

With an emphasis placed on data being fixed and finite, several new detection algorithms were introduced in this thesis and were shown to outperform other linear regression variable selectors. The first original detection algorithm introduced here was a two stage algorithm that combined LASSO with sub-sampling to estimate the parameter distribution and provide a tight confidence interval to reduce false positive errors. An error bound for the estimated distribution was established for finite data length, and, since an asymptotic distribution may or may not be a good approximation when data is finite, it was shown that the estimation based on the sub-sampling technique was sometimes a better approximation than that of the asymptotic method. The performance of the two stage algorithm of LASSO and sub-sampling was compared to that of LASSO alone, and, as expected, the addition of the sub-sampling stage decreased the false positive error rate.

Positive RIVAL was the next new detection algorithm to be introduced, and all detection methods that followed were based on this one algorithm. Positive RIVAL, which stands for removing irrelevant variables amidst LASSO iterations, works by iteratively solving the weighted positive LASSO problem while strategically updating crucial parameters inside the weighted positive LASSO cost function. The positive modification to LASSO is motivated by the application to nuclear material detection where all of the unknown parameters (isotope intensities) are to be non-negative. The idea of positive RIVAL is similar to some other popular variable selectors, e.g., non-negative garrote and adaptive LASSO, in that weights are adjusted based on a convergent sequence of estimates. All these methods have asymptotical set consistency but perform differently in real situations when the the number of data is fixed and finite. In this case, the unique, self-adjusting ability of positive RIVAL allows for the algorithm to perform better than its competitors by requiring a smaller number of data for set convergence. Quantifying these improvements theoretically proved to be difficult, but also unnecessary, as a large number of simulations were run showing the significant improvement of positive RIVAL over the non-negative garrote, LASSO as solved by LARS, and the adaptive LASSO.

In an effort to make positive RIVAL more applicable for a wider range of problems, the non-negativity assumption of the unknown parameters was relaxed, and the modified algorithm was dubbed RIVAL. This algorithm is the same as positive RIVAL, except each iteration involves solving the weighted LASSO (not weighted positive LASSO). RIVAL was not tested on nuclear material detection, but was instead tested on several popular variable selection problems from various published works. The performance of RIVAL, along with the performances of LASSO, adaptive LASSO, and non-negative garrote, were supplied in Section 4.4, and to no surprise, RIVAL performed well, especially when the number of data points was small. The reason is the same as it was for positive RIVAL: RIVAL has a self-adjusting ability that allows for convergence with a smaller n.

With the utility of RIVAL having been established, the focus was directed back to nuclear material detection when it was considered how shielding would affect detection and, in particular, the positive RIVAL algorithm. When nuclear materials are shielded, isotopes' nuclear signatures change, and this presents a significant problem to any detection algorithm, positive RIVAL included. It was explained in Chapter 5 that realistically *all* situations will have some degree of shielding due to truck cargo, the truck itself, the atmosphere, etc., and so another modification to positive RIVAL was necessary. As it happens, shielding attenuates lower frequency signals more so than higher ones, and, as a result, the characteristic shapes of the isotopes' gamma-ray spectra change. Therefore, it was necessary to decompose each isotopes' spectrum into a linear combination of its sub-spectra, where an isotope's entire group of sub-spectra completely described its main spectrum. Each sub-spectrum featured exactly one characteristic peak of the main spectrum, and so the attenuation by a shield was assumed constant for each entire mono-energetic sub-spectrum. It was the grouped nature of the data that was the real motivation for the new algorithm, called group positive RIVAL, since it is counterproductive to have an algorithm select between sub-spectra of the same group (isotope). Instead, what was needed was an algorithm to encourage grouping and determine which groups of sub-spectra are present, thereby identifying which isotopes are present. The group positive LASSO – a hybrid of group LASSO and positive LASSO – was introduced in this thesis, and its convergence results were supplied. The new group positive RIVAL algorithm iteratively solved the group positive LASSO and encouraged the selection of groups by applying the same weights to each parameter within a group.

Group positive RIVAL was initially tested without the presence of shielding materials, and, to my surprise, the results were not as good as the non-group case; positive RIVAL performed well at -32 dB, whereas group positive RIVAL could go no lower than -24 dB. I hypothesized that the reason for this was due to the library becoming ill-conditioned as more and more sub-spectra were considered. I could choose to omit several sub-spectra by raising the minimum branching ratio, but this defeats the purpose of the experiment. In fact, since shielding exists everywhere and in all situations, the comparison between positive RIVAL and group positive RIVAL, as applied to nuclear material detection, is purely for theoretical purposes and really has no practical interpretation. The algorithm was then tested with different combinations of shielding materials, shield thicknesses, and nuclear material strengths. The algorithm worked well for concrete, water, and carbon shielding materials, but struggled to detect accurately under lead shielding. A modest amount of nuclear material was practically invisible under a lead shield thinner than a pack of cigarettes – no doubt a consequence of lead's large density.

Finally, a more robust detection scheme was proposed that paired (group) positive

RIVAL with the method of total least squares for when uncertainty may exist in the library X_n. Total least squares is a linear regression technique that takes errors into account from both the dependent variable and the independent variable; however, much like the ordinary least squares estimate, the total least squares estimate generally produces all non-zero estimates, making it insufficient for variable selection. When a regularization is applied to the total least squares estimate, i.e., the regularization found in the positive LASSO minimization, this new method, called regularized total least squares, can be useful in the variable selection problem. The positive RIVAL algorithm was modified to incorporate regularized total least squares, and this new algorithm was shown in Section 6.2 to perform better than ordinary positive RIVAL under different library uncertainties. In fact, for the case where the library uncertainty was the largest simulated, the addition of total least squares helped positive RIVAL find the correct nuclear materials more than 35% more often than without total least squares. Unfortunately, as of the time of writing this thesis, I do not have any theoretical convergence proofs for this new algorithm.

7.2 Directions for Future Research

7.2.1 Determining the Shield Material

If nuclear material is detected, it is possible, in theory, to identify the type of shielding material and also estimate its thickness by observing the ratio of estimated within-group sub-spectra intensities. To perform such an analysis, two things are required: (1) the detected nuclear material must have more than two sub-spectra (see Table 5.1), and (2) a reliable estimate of the detected intensities must be available. Regarding (2), recall that group positive RIVAL does not estimate intensities, rather, it identifies if intensities are zero or not.

For now, assume a reliable estimate is available. To simplify the analysis, also assume that the detected material consists of exactly three sub-spectra. Let the true sub-spectra intensities of the jth isotope be

$$\beta_{j1}^* = \beta_j^0 e^{-\mu_{j,1} l}$$

$$\beta_{j2}^* = \beta_j^0 e^{-\mu_{j,2} l}$$

$$\beta_{j3}^* = \beta_j^0 e^{-\mu_{j,3} l},$$

where β_j^0 is the unknown unshielded intensity of the jth nuclear material, $\mu_{j,i},\ i = 1, 2, 3$, is the attenuation coefficient of the jth nuclear material at its ith energy peak, and l is the mass thickness of the shielding material – all of which are discussed in Section 5.1. If $\beta_{ji}^*,\ i = 1, 2, 3$, were available, the first two expressions could be divided to eliminate the unknown unshielded intensity β_j^0,

$$\frac{\beta_{j1}^*}{\beta_{j2}^*} = e^{l(\mu_{j,2} - \mu_{j,1})},$$

and this leads to $\ln(\beta_{j1}^*/\beta_{j2}^*) = l(\mu_{j,2} - \mu_{j,1})$. This process can be repeated for β_{j2}^* and β_{j3}^* to get $\ln(\beta_{j2}^*/\beta_{j3}^*) = l(\mu_{j,3} - \mu_{j,2})$. These two expressions can be divided to eliminate the unknown mass thickness l

$$\frac{\ln(\beta_{j1}^*/\beta_{j2}^*)}{\ln(\beta_{j2}^*/\beta_{j3}^*)} = \frac{\mu_{j,2} - \mu_{j,1}}{\mu_{j,3} - \mu_{j,2}}.$$

The intensities $\beta_{ji}^*,\ i = 1, 2, 3$, are not available, but their estimates $\hat{\beta}_{ji}$ are. Thus, we choose the one of four possible shielding materials, whose attenuation coefficients are known, that minimizes

$$\left| \frac{\ln(\hat{\beta}_{j1}/\hat{\beta}_{j2})}{\ln(\hat{\beta}_{j2}/\hat{\beta}_{j3})} - \frac{\mu_{j,2} - \mu_{j,1}}{\mu_{j,3} - \mu_{j,2}} \right|.$$

Once the shielding material is found, finding the mass thickness l is trivial.

A reliable estimate is crucial to the identification of the shielding material. For the estimates used in this thesis, I used the set as found by group positive RIVAL to trim off irrelevant regressors from the matrix X_n and then perform a least squares estimate. What I found was that with the inclusion of much more data into X_n brought forth by the introduction of sub-spectra, the pseudo-inverse required of the least squares estimate was near singular. This makes the least squares estimate unreliable. Different estimates should be considered, if not a completely different approach. For reference, I have included the MATLAB code determineShielding.m in Section 9.3, which is my attempt at determining the shield material and thickness.

7.2.2 Generalizing Theorem 5.2

It was required for the convergence proof of Theorem 5.2 that $\mathbf{H_j}$ was a scaled identity matrix, even though simulations showed the convergence of group positive RIVAL for more general cases. Generalizing this theorem by requiring only that $\mathbf{H_j}$ be diagonal (and positive) would further strengthen the argument in favor of group positive RIVAL. As it is now, Algorithm 5.3 instructs us to update weights by dividing by ξ_j, the average of the norms of each column of X_{jn}. Of course, when $\mathbf{H_j}$ is a scaled identity matrix, the scaling constant is ξ_j, and so essentially we are normalizing by the diagonal element.

The problem is that, for each iteration k, it is not known which elements within the jth group are contributing to N_k (see the proof of Theorem 5.2 in Chapter 8), and so the normalization in the weight update procedure must be one that takes all of the columns of X_{jn} into account, e.g., the average of the norms of each column of X_{jn}, the minimum of the norms of each column of X_{jn}, or the maximum of the norms of each column of X_{jn}. When $\mathbf{H_j}$ is a scaled identity matrix, these three normalization examples are all equivalent and reduce to normalizing column by column.

If $\mathbf{H_j}$ is allowed to be a general diagonal matrix (with positive elements), and the update procedure is changed as to normalize by the maximum of the norms of each column of X_{jn}, Lemma A.3 can be proven, but Lemma A.4 cannot. On the other hand, when the update procedure is to normalize by the minimum of the norms of each column of X_{jn}, Lemma A.4 can be proven, but Lemma A.3 cannot.

7.2.3 Algorithm 6.2 Parameter Convergence

It was established in Section 6.2 that Algorithms 6.1 and 6.2 converge to the global minimum of the cost functions of the positive LASSO regularized total least squares and the positive RIVAL total least squares, respectively, by stating that each update never worsens the positive, convex cost functions. No convergence results were ever stated regarding the estimates $\hat{\beta}$ and $\hat{Er}$ to their true values β^* and ΔX_n. Recall, that these two algorithms fall into a class of algorithms known as alternating convex search algorithms and can be used to solve the minimization of biconvex cost functions. It was shown in [17] that it is not possible to prove the parameter convergence of an alternating convex search algorithm applied to a biconvex minimization in the general case. One could hope that the specific form of the positive LASSO regularized total least squares would lend itself well to the mathematics and allow for the possibility of proving parameter convergence. The authors of [45] use an alternating convex search algorithm to solve total least squares with a LASSO-type of regularization for a compressive sensing application. They state that the algorithm at least converges to a stationary point and the limit point depends on the initialization, but they make no attempt at a rigorous proof.

CHAPTER 8
PROOFS OF SELECTED THEOREMS AND LEMMAS

8.1 Proof of Theorem 3.1

The second part of the theorem follows directly from [14]. For the first part, $F_{\hat{\beta}_{N_s}}(b) = E\mathbf{1}_{[\hat{\beta}(i)\leq b]}$. Observe $W_i = \mathbf{1}_{[\hat{\beta}(i)\leq b]} - F_{\hat{\beta}_{N_s}}(b)$ is zero mean and stationary because y_i and v_i are stationary. Following [14], we have

$$E\Big(\hat{F}_{\hat{\beta}_{N_s}}(b) - F_{\hat{\beta}_{N_s}}(b)\Big)^2 = E\Big(\frac{1}{m}\sum_{i=1}^{m} W_i\Big)^2 = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} W_i W_j$$

$$= \frac{1}{m^2}\sum_{\tau=-m}^{m}(m-|\tau|)\gamma(\tau)$$

where $\gamma(\tau) = EW_iW_{i+\tau}$. Notice v_i is i.i.d. and its strong mixing coefficient satisfies

$$\alpha(\tau) = \sup_{A,B}\{|\Pr\{AB\} - \Pr\{A\}\Pr\{B\}|\big|A \in A_0,\ B \in A^{\tau}\}$$

$$= \begin{cases} \leq 1, & \tau = 0 \\ 0, & \tau > b \end{cases}$$

where A_0 and A^{τ} are the σ–algebras generated by v_i, $i \leq 0$ and v_{τ}, $\tau \geq 0$ respectively. Now, it follows from [14] that $|\gamma(\tau)| \leq 12\alpha(|\tau|)$. Thus,

$$E\Big(\hat{F}_{\hat{\beta}_{N_s}}(b) - F_{\hat{\beta}_{N_s}}(b)\Big)^2 = \frac{1}{m^2}\sum_{\tau=-m}^{m}(m-|\tau|)\gamma(\tau) \leq \frac{12}{m}\ .$$

□

8.2 Proof of Theorem 4.3:

Minimizing

$$J = (Y_n - X_n\beta)^T(Y_n - X_n\beta) + t\cdot\sum_{j=1}^{p} w_j\beta_j, \quad \beta_j \geq 0$$

is equivalent to minimizing

$$J_2 = -2n\beta^{*T} C_n \beta - 2V_n^T X_n \beta + n\beta^T C_n \beta + t \cdot \sum_{j=1}^{p} w_j \beta_j, \quad \beta_j \geq 0$$

which is equivalent to minimizing

$$J_3 = -2\frac{n}{t}\beta^{*T} \begin{pmatrix} \xi_1 & 0 & \cdots & 0 \\ 0 & \xi_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \xi_p \end{pmatrix} \beta + \frac{2V_n^T X_n}{t}\beta^* - \frac{2V_n^T X_n}{t}\beta$$

$$+\frac{n}{t}\beta^T \begin{pmatrix} \xi_1 & 0 & \cdots & 0 \\ 0 & \xi_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \xi_p \end{pmatrix} \beta + \frac{n}{t}\beta^{T*} \begin{pmatrix} \xi_1 & 0 & \cdots & 0 \\ 0 & \xi_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \xi_p \end{pmatrix} \beta^* + \sum_{j=1}^{p} w_j \beta_j$$

$$= a(\beta^* - \beta)^T \begin{pmatrix} \xi_1 & 0 & \cdots & 0 \\ 0 & \xi_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \xi_p \end{pmatrix} (\beta^* - \beta) + \hat{d}(\beta^* - \beta) + \sum_{j=1}^{p} w_j \beta_j$$

where $a = \frac{n}{t}$, $\hat{d} = \frac{2V_n^T X_n}{t} = \frac{2V_n^T X_n}{\sqrt{n}} \frac{\sqrt{n}}{t} = (\hat{d}_1, ..., \hat{d}_p)$, and $\beta_j \geq 0$. Therefore, minimizing J_3 is achieved by minimizing J_4 and J_5 separately, where

$$J_4 = \sum_{j=1}^{d} (a\beta_j^2 - 2ab_j\beta_j + \frac{w_j}{\xi_j}\beta_j), \; b_j = \beta_j^* + \frac{\hat{d}_j}{2a\xi_j}$$

where $\beta_j \geq 0$, and

$$J_5 = \sum_{j=d+1}^{p} (a\beta_j^2 - c_j\beta_j + \frac{w_j}{\xi_j}\beta_j), \; \beta_j \geq 0.$$

Since $a = \frac{n}{t}$, $\hat{d} = \frac{2V_n^T X_n}{\sqrt{n}} \frac{\sqrt{n}}{t}$, $c_j = \frac{\hat{d}_j}{\xi_j}$, $b_j = \beta_j^* + \frac{\hat{d}_j}{2a\xi_j}$, there always exists a $\delta > 0$ for large enough n so the the conditions of Lemmas 8.1 and 8.2 are simultaneously satisfied.

□

Lemma 8.1 (Zero Identification). *Consider a sequence of scalar minimization problems*

$$J = \min_{\beta(k) \geq 0} \left\{ a\beta^2(k) - c\beta(k) + w(k)\beta(k) \right\}, \; a > 0, \; w(1) = 1.$$

Assume there exists a constant $\delta > 0$ such that

$$a \geq \frac{(c+\delta)^2}{8}.$$

Let $\beta(k)$ be the solution of J for the given $w(k)$. Construct $\beta(k+1)$ as follows. If $\beta(k) = 0$, set $\hat{\beta}(k+i) = 0$ for $i \geq 0$ and stop the algorithm. If $\beta(k) > 0$, let $\bar{w} = 1/\beta(k)$ and

$$w(k+1) = Q(k)\bar{w}(k) + (1 - Q(k))w(k),$$

where $0 \leq Q(k) \leq 1$ is a sequence satisfying $\sum_{k=1}^{\infty} Q(k) = \infty$. Denote $\beta(k+1)$ the solution of J for given $w(k+1)$. Then, there exists a finite integer $k_0 \geq 1$ such that the sequence generated from the above satisfies

$$\hat{\beta}(k) = 0, \ \forall k \geq k_0$$

Proof: If $1 = w(1) \geq c$, the minimum $\beta(1) = 0$ and this implies $k_0 = 1$ and $\beta(k) = 0$, $k \geq k_0 = 1$. If $c > w(1)$, the minimum $\beta(1)$ is achieved at some $\beta(k) > 0$. The first order necessary condition

$$\frac{\partial J}{\partial \beta} = 2a\beta - (c - w) = 0$$

implies

$$\beta(1) = \frac{c - w(1)}{2a} > 0 \ \rightarrow \ \bar{w}(1) = \frac{2a}{c - w(1)}.$$

From the hypothesis, we have

$$2a - \frac{(c-\delta)^2}{4} \geq \delta c \ \rightarrow \ \left[w(1) - \frac{(c-\delta)}{2}\right]^2 - \frac{(c-\delta)^2}{4} + 2a \geq \delta c$$

Thus,

$$\frac{2a - cw(1) + w^2(1)}{c - w(1)} \geq \delta \ or \ \bar{w}(1) \geq w(1) + \delta$$

and

$$w(2) = Q(1)\bar{w}(1) + (1 - Q(1))w(1)$$

$$\geq Q(1)w(1) + Q(1)\delta + (1 - Q(1))w(1) = w(1) + Q(1)\delta$$

By induction, if $w(k) < c$, $w(k+1) \geq w(k) + Q(k)\delta \geq w(1) + \sum_{i=1}^{k} Q(i) \cdot \delta$. Equivalently, there is an integer $k_0 > 0$ such that $w(k_0) > c$ and the corresponding solution of J is $\beta(k_0) = 0$ and $\beta(k_0 + i) = 0$, $i \geq 0$.

□

Lemma 8.2 (Non-Zero Identification). *Consider a sequence of scalar minimization problems*

$$J = \min_{\beta(k) \geq 0} \left\{ a\beta^2(k) - 2ab\beta(k) + w(k)\beta(k) \right\}, \ a > 0, \ b > 0, \ w(1) = 1.$$

Assume there exists a constant $\delta > 0$ such that

$$\delta = \frac{b - \sqrt{b^2 - 2/a}}{2} > 0$$

and

$$w(1) \leq 2ab - \frac{1}{b - \delta}.$$

Let $\beta(k)$ be the solution of J for the given $w(k)$. Construct $\beta(k+1)$ as follows. If $\beta(k) = 0$, set $\beta(k+i) = 0$ for $i \geq 0$ and stop the algorithm. If $\beta(k) > 0$, let $\bar{w} = 1/\beta(k)$ and

$$w(k+1) = Q(k)\bar{w}(k) + (1 - Q(k))w(k),$$

where $0 \leq Q(k) \leq 1$ is a sequence satisfying $\sum_{k=1}^{\infty} Q(k) = \infty$. Denote $\beta(k+1)$ the solution of J for given $w(k+1)$. Then, the sequence $\beta(k)$ is uniformly bounded from above and from below

$$0 < \eta_1 \leq \beta(k) \leq \eta_2 < \infty, \ \forall k.$$

Proof: The idea of the proof is to show that the $w(k)$'s are bounded. $b > 0$ and $w(1) < 2ab$ imply that the minimum is achieved at some $\beta(1) > 0$. The first order necessary condition

$$\frac{\partial J}{\partial \beta} = 2a\beta - 2ab + w = 0$$

implies

$$\beta(1) = b - \frac{w(1)}{2a} > 0 \ \rightarrow \ \bar{w}(1) = \frac{1}{b - \frac{w(1)}{2a}} > 0.$$

Further, $w(1) \leq 2ab - \frac{1}{b-\delta}$ leads to

$$(b-\delta)w(1) \leq 2ab(b-\delta) - 1 \ \rightarrow \ \frac{\bar{w}(1)}{2a} - b$$

$$= \frac{1}{2ab - w(1)} - b \leq -\delta$$

or

$$\bar{w}(1) \leq 2ab - 2a\delta.$$

On the other hand, from the definition of δ, it is easilty verified that

$$\delta^2 - \delta b + 1/(2a) = 0 \ \rightarrow \ 2a\delta = 1/(b-\delta).$$

Hence,

$$0 < w(1) \leq 2ab - \frac{1}{b-\delta} \ \rightarrow \ 0 < \bar{w}(1) \leq 2ab - \frac{1}{b-\delta}$$

and this implies

$$0 < w(2) = Q(1)\bar{w}(1) + (1 - Q(1))w(1) \leq 2ab - \frac{1}{b-\delta}.$$

By the induction, we have for all $k \geq 1$,

$$0 < w(k) \leq 2ab - \frac{1}{b-\delta}$$

and

$$\beta(k) = b - \frac{w(k)}{2a} \geq \delta > 0.$$

This shows that $\beta(k)$ is bounded away from zero. The upper bound can be derived easily,

$$\beta(k) \leq b + |w(k)/(2a)| \leq 2b - \delta < \infty, \ \forall k.$$

$\square$

8.3 Proof of Theorem 5.2

Minimizing

$$J_1 = (Y_n - X_n\beta)^T(Y_n - X_n\beta) + t\sum_{j=1}^{p} w_j \sum_{i=1}^{K_j} \beta_{ji},\ \beta_{ji} \geq 0$$

is equivalent to minimizing

$$J_2 = -2n\beta^{*T}\Big(\frac{1}{n}X_n^TX_n\Big)\beta - 2V_n^TX_n\beta + n\beta^T\Big(\frac{1}{n}X_n^TX_n\Big)\beta + t\sum_{j=1}^{p} w_j \sum_{i=1}^{K_j} \beta_{ji},\ \beta_{ji} \geq 0$$

which is equivalent to minimizing

$$J_3 = -2\frac{n}{t}\beta^{*T}\begin{pmatrix} \mathbf{H}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{H}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{H}_p \end{pmatrix}\beta + \frac{2V_n^TX_n}{t}\beta^* - \frac{2V_n^TX_n}{t}\beta$$

$$+\frac{n}{t}\beta^T\begin{pmatrix} \mathbf{H}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{H}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{H}_p \end{pmatrix}\beta + \frac{n}{t}\beta^{*T}\begin{pmatrix} \mathbf{H}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{H}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{H}_p \end{pmatrix}\beta^*$$

$$+\sum_{j=1}^{p} w_j \sum_{i=1}^{K_j} \beta_{ji},\ \beta_{ji} \geq 0.$$

Because of the form of $\frac{1}{n}X_n^TX_n$, we may do the minimization of each group separately. Consider the jth group. The minimization of the group is

$$J = a(\beta_j^* - \beta_j)^T\mathbf{H}_j(\beta_j^* - \beta_j) + \hat{d}(\beta^* - \beta) + w_j\sum_{i=1}^{K_j}\beta_{ji},\ \beta_{ji} \geq 0$$

$$= a\xi_j(\beta_j^* - \beta_j)^T(\beta_j^* - \beta_j) + \hat{d}(\beta^* - \beta) + w_j\sum_{i=1}^{K_j}\beta_{ji},\ \beta_{ji} \geq 0$$

where $a = \frac{n}{t}$, and $\hat{d} = \frac{2V_n^TX_{jn}}{\sqrt{n}}\frac{\sqrt{n}}{t} = (\hat{d}_1, ..., \hat{d}_{K_j})$. For $j = 1, ...d$, the minimization can be written as

$$J_4 = \sum_{i=1}^{K_j}\Big[a\beta_{ji}^2 - 2ab_i\beta_{ji} + \frac{w_j}{\xi_j}\beta_{ji}\Big],\quad b_i = \beta_{ji}^* + \frac{d_i}{2a\xi_j}$$

where $\beta_{ji} \geq 0$, and for $j = d+1, ..., p$, the minimization is

$$J_5 = \sum_{i=1}^{K_j} \Big[a\beta_{ji}^2 - c_i\beta_{ji} + \frac{w_j}{\xi_j}\beta_{ji} \Big], \quad c_i = \hat{d}_i/\xi_j$$

where $\beta_{ji} \geq 0$. Since $a = \frac{n}{t}$, $\hat{d} = \frac{2V_n^T X_{jn}}{\sqrt{n}} \frac{\sqrt{n}}{t}$, $c_i = \frac{\hat{d}_i}{\xi_j}$, $b_i = \beta_{ji}^* + \frac{d_i}{2a\xi_j}$, there always exists a constant $\delta > 0$ for large enough n so that the conditions of Lemmas (8.3) and (8.4) are simultaneously satisfied. Then, the conclusions follow from those two lemmas.

□

Lemma 8.3 (Group Non-Zero Identification).

$$J = \min_{\beta_i \geq 0} \{a\beta_i^2(k) - c_i\beta_i(k) + \frac{w(k)}{\xi}\beta_i(k)\}, \ a > 0, \ w(1) = 1, \ i = 1, ..., K.$$

Assume there exists a constant $\delta > 0$ such that

$$a \geq \frac{(\hat{C} + \delta)^2}{8}$$

where $\hat{C} = \max_E \{\sum_{i \in E} c_i\}, \ E \subseteq \{1, ..., K\}$. Let $\beta_i(k), \ i = 1, ..., K$, be the solution of J for the given $w(k)$. Construct $\beta_i(k+1)$ as follows. If $\sum_i \beta_i(k) = 0$, set $\beta_i(k+\kappa) = 0$ for $\kappa \geq 0$ and stop the algorithm. If $\sum_i \beta_i(k) > 0$, update the weight as

$$w(k+1) = q(k) \cdot \frac{1}{\sum_i \beta(k)} + (1 - q(k))\frac{w(k)}{\xi}$$

where $0 \leq q(k) \leq 1$ is a sequence satisfying $\sum_{k=1}^{\infty} q(k) = \infty$. Denote $\beta_i(k+1)$ the solution of J for given $w(k+1)$. Then, there exists a finite integer $k_0 \geq 1$ such that the sequence generated above satisfies

$$\sum_{i=1}^{K} \beta_i(k) = 0, \ \forall k \geq k_0,$$

or equivalently, since $\beta_i \geq 0$,

$$||\beta(k)||_2 = 0, \ \forall k \geq k_0, \ \textit{where} \ \beta(k) = \begin{pmatrix} \beta_1(k) \\ \vdots \\ \beta_K(k) \end{pmatrix}.$$

Proof: Assume there are $0 \le N_k \le K$ elements that satisfy the following:

$$1 = w(1) < c_i \xi.$$

If $N_k = 0$, then $w(1) \ge \max(c_i\xi)$, $i = 1, ...K$ and $\beta_1(1) = ... = \beta_K(1) = 0$. Thus $\sum_i \beta(k) = ||\beta(k)||_2 = 0$ and $k_0 = 1$. If $N_k > 0$, without loss of generality, re-arrange indices such that $w(1) < c_i\xi$, $i = 1, ..., N_k$. Then, the first order necessary condition

$$\frac{\partial J}{\partial \beta_i} = 0 \rightarrow \ \beta_i(1) = \frac{c_i\xi - w(1)}{2a\xi} > 0, \ i = 1, ..., N_k$$

and

$$\sum_{i=1}^{K} \beta_i(1) = \frac{\sum_{i=1}^{N_k} c_i - w(1)N_k/\xi}{2a} > 0.$$

Let

$$\bar{w}(1) = \frac{1}{\sum_{i=1}^{K} \beta_i(1)} = \frac{2a}{\sum_{i=1}^{N_k} c_i - w(1)N_k/\xi} > 0.$$

From the hypothesis, it must be the case that

$$a \ge \frac{(\sum_{i=1}^{N_k} c_i + \delta)^2}{8}$$

since it is guaranteed that $\hat{C} \ge \sum_{i=1}^{N_k} c_i$. This implies

$$2a - \frac{(\sum_{i=1}^{N_k} c_i - \delta)^2}{4} \ge \delta \sum_{i=1}^{N_k} c_i$$

$$\rightarrow \ \left[w(1)N_k/\xi - \frac{(\sum_{i=1}^{N_k} c_i - \delta)}{2}\right]^2 - \frac{(\sum_{i=1}^{N_k} c_i - \delta)^2}{4} + 2a \ge \delta \sum_{i=1}^{N_k} c_i$$

$$\rightarrow \ w^2(1)N_k^2/\xi^2 - w(1)N_k/\xi\Big(\sum_{i=1}^{N_k} c_i - \delta\Big) + 2a \ge \delta \sum_{i=1}^{N_k} c_i$$

$$\rightarrow \ w(1)N_k/\xi\Big[w(1)N_k/\xi - \sum_{i=1}^{N_k} c_i\Big] + 2a \ge \delta\Big[\sum_{i=1}^{N_k} c_i - w(1)N_k/\xi\Big]$$

$$\rightarrow \ \frac{2a - w(1)N_k/\xi\Big[\sum_{i=1}^{N_k} c_i - w(1)N_k/\xi\Big]}{\sum_{i=1}^{N_k} c_i - w(1)N_k/\xi} \ge \delta \rightarrow \ \bar{w}(1) - w(1)N_k/\xi \ge \delta$$

$$\rightarrow \ \bar{w}(1)\delta + w(1)N_k/\xi \geq \delta + w(1)/\xi.$$

From the described weight update procedure,

$$w(2) = q(1)\bar{w}(1) + (1-q(1))\frac{w(1)}{\xi} \geq \delta \cdot q(1) + \frac{w(1)}{\xi}.$$

By the induction, if $w(k) < c_i\xi$, $w(k+1) \geq \frac{w(k)}{\xi} + \delta \cdot q(k) \geq \frac{w(1)}{\xi} + \delta \cdot \sum_{l=1}^{k} q(l)$ or equivalently, there is an integer $k_0 > 0$ such that $w(k_0) > c_i\xi$ and the corresponding solution of J is $\beta_i(k_0) = 0$ and $\beta_i(k+\kappa) = 0,\ \kappa \geq 0$. Then $\sum_i \beta(k) = ||\beta(k)||_2 = 0$.

□

Lemma 8.4 (Group Zero Identification). *Consider a sequence of minimization problems*

$$J = \min_{\beta_i \geq 0}\{a\beta_i^2(k) - 2ab_i\beta_i(k) + \frac{w(k)}{\xi}\beta_i(k)\},\ a > 0,\ w(1) = 1, b_i > 0,\ i = 1, ..., K.$$

Re-arrange indices and define $1 \leq N_k \leq K$ such that $w(1)/\xi < 2ab_i,\ i = 1, ..., N_k$. Assume there exists a constant $\delta > 0$ such that

$$\delta = \frac{B - \sqrt{B^2 - \frac{2N_k}{a}}}{2} > 0, \quad \frac{w(1)}{\xi} \leq \frac{2ab_i}{N_k} - \frac{1}{B-\delta},\ i = 1, ..., N_k$$

where $B = \sum_{i=1}^{N_k} b_i$. Let $\beta_i(k),\ i = 1, ..., K$, be the solution of J for the given $w(k)$. Construct $\beta_i(k+1)$ as follows. If $\sum_i \beta_i(k) = 0$, set $\beta_i(k+\kappa) = 0$ for $\kappa \geq 0$ and stop the algorithm. If $\sum_i \beta_i(k) > 0$, update the weight as

$$w(k+1) = q(k) \cdot \frac{1}{\sum_i \beta(k)} + (1-q(k))\frac{w(k)}{\xi}$$

where $0 \leq q(k) \leq 1$ is a sequence satisfying $\sum_{k=1}^{\infty} q(k) = \infty$. Denote $\beta_i(k+1)$ the solution of J for given $w(k+1)$. Then, the sequence $||\beta(k)||_2,\ \beta(k) = (\beta_1(k), ..., \beta_K(k))^T$, is uniformly bounded from above and below

$$0 < \eta_1 \leq ||\beta(k)||_2 \leq \eta_2 < \infty,\ \forall k > k_0.$$

Proof: The idea of the proof is to show that $w(k)$'s are bounded. The condition on $w(1)$ ensures $w(1)/\xi < 2ab_i$ for $i = 1, ..., N_k$, and thus the first order necessary condition

$$\frac{\partial J}{\partial \beta_i} = 2a\beta_i - 2ab_i + w/\xi \rightarrow \ \beta_i(1) = b_i - \frac{w(1)}{2a\xi} > 0, \ i = 1, ..., N_k.$$

So

$$\sum_{i=1}^{K} \beta_i(1) = \frac{2aB - N_k w(1)/\xi}{2a} > 0.$$

Define $\bar{w}(1) = 1/\sum_{i=1}^{K} \beta_i(1)$

$$\bar{w}(1) = \frac{2a}{2aB - N_k w(1)/\xi} > 0.$$

Now, from the hypothesis,

$$\frac{w(1)}{\xi} \le \frac{2ab_i}{N_k} - \frac{1}{B-\delta} \rightarrow \ N_k(B-\delta)w(1)/\xi \le 2ab_i(B-\delta) - N_k$$

$$\rightarrow \ B\Big[N_k w(1)/\xi - 2ab_i\Big] + N_k \le \delta\Big[N_k w(1)/\xi - 2ab_i\Big]$$

$$\rightarrow \ B + \frac{N_k}{N_k w(1)/\xi - 2ab_i} \ge \delta, \ i = 1, ..., N_k$$

$$\rightarrow \ \frac{N_k}{2ab_i - N_k w(1)/\xi} - B \le -\delta \rightarrow \ \frac{N_k}{2aB - N_k w(1)/\xi} - B \le -\delta$$

and

$$\frac{\bar{w}(1)N_k}{2a} - B \le -\delta$$

So

$$\bar{w}(1) \le \frac{2a}{N_k}B - \frac{2a}{N_k}\delta.$$

From our choice of δ, we see that $2a\delta/N_k = 1/(B-\delta)$, so then

$$\bar{w}(1) \le \frac{2a}{N_k}B - \frac{1}{B-\delta}$$

and

$$w(1)/\xi \le \frac{2ab_i}{N_k} - \frac{1}{B-\delta} \le \frac{2a}{N_k}B - \frac{1}{B-\delta}.$$

This implies

$$w(2) = q(1)\bar{w}(1) + (1 - q(1))\frac{w(1)}{\xi} \leq \frac{2a}{N_k}B - \frac{1}{B-\delta}.$$

By the induction, we have for all $k \geq 1$,

$$0 < w(k) \leq \frac{2a}{N_k}B - \frac{1}{B-\delta}, \; and \; \beta_i(k) = b_i - \frac{w(k)}{2a\xi}, \; i = 1, ..., N_k.$$

So,

$$2ab_i - 2a\beta_i(k) \leq \frac{2a}{N_k}B - \frac{1}{B-\delta}, \; i = 1, ..., N_k.$$

Noting that the minimum is achieved at $\beta_i(k) = 0$ for $i = N_k, ..., K$, we have

$$-2a\sum_{i=1}^{K}\beta_i(k) \leq \frac{-N_k}{B-\delta} \to \sum_{i=1}^{K}\beta_i(k) \geq \frac{N_k}{2a}\cdot\frac{1}{B-\delta} = \delta > 0.$$

This is implies there exists some η_1 such that $||\beta(k)||_2 \geq \eta_1 > 0$. The upper bound can be derived easily,

$$\sum_{i=1}^{K}\beta_i(k) = B - \frac{N_k w(k)}{2a\xi} \to \sum_{i=1}^{K}\beta_i(k) \leq B + \frac{N_k w(k)}{2a\xi} \leq B + B - \frac{N_k}{2a(B-\delta)}$$

$$\to \sum_{i=1}^{K}\beta_i(k) \leq 2B - \delta < \infty.$$

Therefore, $||\beta(k)||_2$ must be bounded by some $\eta_2 < \infty$.

□

CHAPTER 9
MATLAB CODE

9.1 Master.m

```
%The following MATLAB code, written by Paul Kump and Er-wei Bai,
%uses the group RIVAL algorithm to detect simulated nuclear
%materials. It does so by applying RIVAL with a grid of
%candidate lambdas, trimming off the irrelevant variables,
%computing least squares with new set, then computing AIC/BIC
%for each of the candidate lambdas.  The code selects that
%lambda that minimizes this AIC/BIC and selects the estimate
%given from this lambda.
%
%Shielding can be simulated by calling the function
%applyShield.m.  This function simulates shielding as the
%library is scaled by exponentials with powers of mass thickness
%times shielding coefficients.  The shielding material and
%thickness can be estimated by calling determineshielding.m.
%%

clear

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    %SETUP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[num,txt,raw]=xlsread('isotopesNS.xls');

BA133=num(:,1:7);
CE139=num(:,8);
CO57=num(:,9:10);
CO60=num(:,11:14);
CS137=num(:,15:18);
GA67=num(:,19:24);
I131=num(:,25:29);
K40=num(:,30);
NA22=num(:,31:32);
PU239=num(:,33:42);
BACKGND=1/10*num(:,43);

numberOfMaterials=11;  %%materials plus background

```

```

BB=[BA133 CE139 CO57 CO60 CS137...
    GA67 I131 K40 NA22 PU239 BACKGND];

[mb nb]=size(BB);
[BA133rows, BA133columns]=size(BA133);
[CE139rows, CE139columns]=size(CE139);
[CO57rows, CO57columns]=size(CO57);
[CO60rows, CO60columns]=size(CO60);
[CS137rows, CS137columns]=size(CS137);
[GA67rows, GA67columns]=size(GA67);
[I131rows, I131columns]=size(I131);
[K40rows, K40columns]=size(K40);
[NA22rows, NA22columns]=size(NA22);
[PU239rows, PU239columns]=size(PU239);
[BACKGNDrows, BACKGNDcolumns]=size(BACKGND);

subLengths=[BA133columns CE139columns CO57columns...
    CO60columns CS137columns GA67columns I131columns...
    K40columns NA22columns PU239columns BACKGNDcolumns];

subSum=cumsum(subLengths);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                   %CHOOSE SHIELD AND APPLY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


[BBprime,shield]=applyShield(BB,subSum,4,20);  %%(data getting
    %%shielded, subSum, shielding material, mass thickness)
                 %%shielding material: enter 1 for carbon
                 %%(graphite), enter 2 for concrete, enter 3 for
                 %%lead, and enter 4 for water.


 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                   %CHOOSE SNR OR UNCOMMENT AND CHOOSE ALPHA
   %ALPHA WILL BE DETERMINED BY SNR OR SNR WILL BE DETERMINED
   %BY ALPHA
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


 alpha=1;
 SNR=10*log10(alpha*(sum(sum(.2*BBprime(:,33:42)))+sum(sum(...
    BBprime(:,25:29))))/(sum(sum(BBprime(:,end)))));
```

```
% SNR=-10;
% alpha=10^(SNR/10)*sum(sum(BACKGND))/(sum(sum(.2*PU239))+...
%       sum(sum(I131)));   %%signal strength.

beta=[alpha*[zeros(subLengths(1),1);zeros(subLengths(2),1); ...
    zeros(subLengths(3),1);zeros(subLengths(4),1); ...
    zeros(subLengths(5),1); zeros(subLengths(6),1);...
    1*ones(subLengths(7),1); zeros(subLengths(8),1); ...
    zeros(subLengths(9),1); 0.2*ones(subLengths(10),1)];...
     ones(subLengths(11),1)];


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                         %DEFINE THRESHOLDS
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

t1=0.0000009; %threshold for zero
t2=0.00001;  %stopping criterion for iteration

for s=1:numberOfMaterials
    thresholdArray(s)=t1*sqrt(subLengths(s));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    %BEGIN SIMULATIONS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

simulations=1;
er=zeros(1,numberOfMaterials);

for ii=1:simulations %# of Monte Carlo simulations
    ii
     bh=[];
     YY=poissrnd(BBprime*beta,[mb,1]);
     for i=1:mb
        YY1(i,1)=YY(i)*sqrt(1/YY(i));
         BB1(i,:)=BB(i,:)*sqrt(1/YY(i));
     end
     bh(:,1)=abs(inv(BB1'*BB1)*BB1'*YY1);
     e1=1;
     i1=1;

     index=1;
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                        %BEGIN MAIN RIVAL LOOP
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
  for lambda=.03:.001:.05

  while e1 > t2
     ind=find(bh(:,i1) >= t1); %find > 0 elements in beta
     lind=length(ind);
     phi=[];
     bb=[];
     w=[];
     weight=zeros(numberOfMaterials,1);
     count=zeros(numberOfMaterials,1);


       for jj=1:lind
           test=false;
           ww=1;
           while test==false && ww < numberOfMaterials+1
                   if ind(jj)<1+subSum(ww)
                       group(jj,i1)=ww;
                       test=true;
                   end
                   ww=ww+1;
           end
          weight(group(jj,i1)) = weight(group(jj,i1))+...
               bh(ind(jj),i1);
          count(group(jj,i1))=count(group(jj,i1))+1;
       end

       for k=1:lind
         phi(:,k)=BB1(:,ind(k));  %corr column of beta >0
       end
       for k=1:lind
         correction=0;
         for j=1:lind
             if group(j,i1)==group(k,i1)
                 correction=correction+sum(phi(:,j));
             end
         end
         %weights
         w(k,1)=lambda*1/(weight(group(k,i1)))*correction;
       end

     A=phi'*phi;
     C=(phi'*YY1-1/2*w);
     i=1;
     e=1;
      bb(:,1)=abs(inv(A)*phi'*YY1);
```

```

        while e > t2  %solve positive lasso
           bhat1=bb(:,i);

          for k=1:lind
            bhat1(k)=max((C(k)−A(k,:)*bhat1+A(k,k)*...
              bhat1(k))/A(k,k),0);
          end
          i=i+1;
           bb(:,i)=bhat1;
           e=norm(bb(:,i)−bb(:,i−1))/norm(bb(:,i−1));
        end

        for k1=1:lind  %construct new estimate beta
           bh(ind(k1),i1+1)=bb(k1,end);
        end
        %stoping criterion
        e1=norm(bh(:,i1+1)−bh(:,i1))/norm(bh(:,i1));
        i1=i1+1;
     end

     bhat(:,ii)=bh(:,end);
     best(:,index)=bhat(:,ii);

     normArray(1,ii)=norm(bhat(1:subSum(1),ii));
     for x=2:numberOfMaterials
         normArray(x,ii)=norm(bhat(1+subSum(x−1):subSum(x),ii));
     end

     least=[];
     pass=0;
     for xx=1:numberOfMaterials
         if normArray(xx,ii)>=thresholdArray(xx)
             for xxx=1:subLengths(xx)
                 if xx==1
                     least(:,xxx+pass)=BB1(:,xxx);
                 else
                     least(:,xxx+pass)=BB1(:,subSum(xx−1)+xxx);
                 end
             end
             pass=xxx+pass;
         end
     end

     leastSquares=abs(inv(least'*least)*least'*YY1);

```

```

  AIC(index)=informationCriterion(YY1,least,leastSquares,0);
  %BIC(index)=informationCriterion(YY1,least,leastSquares,1);

  index=index+1;
   end    %%%%%%%%%%%%%%end main RIVAL loop%%%%%%%%%%%%%%%%%%

 [optimalAIC,optimalIndex]=min(AIC);
 %[optimalBIC,optimalIndex]=min(BIC);
 rivalBeta=best(:,optimalIndex);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
           %WHICH MATERIALS WERE FOUND AND DETERMINE THE SHIELD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    index=find(rivalBeta>0);
    lindex=length(index);
    least=[];
    for h=1:lindex
        hh=1;
        test=false;
        if index(h)<subSum(hh)+1
            grouping(h)=hh;
            test=true;
            elseif index(h)>subSum(numberOfMaterials -1)
                grouping(h)=numberOfMaterials;
                test=true;
        end
        while hh<numberOfMaterials && test==false
            if index(h)>subSum(hh)&& index(h)<subSum(hh+1)+1
                grouping(h)=hh+1;
                test=true;
            end
            hh=hh+1;
        end
    end

    least=[];
    pass=0;
    for xx=1:numberOfMaterials
        if normArray(xx,ii)>=thresholdArray(xx)
            er(xx)=er(xx)+1;
            for xxx=1:subLengths(xx)
                if xx==1
                    least(:,xxx+pass)=BB1(:,xxx);
                else
```

```
                                least(:,xxx+pass)=BB1(:,subSum(xx−1)+...
                                    xxx);
                        end
                    end
                    pass=xxx+pass;
                end
            end

            leastSquares=abs(inv(least'*least)*least'*YY1);

            %%determineshield doesn't work yet
            [shieldMaterial(ii),massThickness(ii),cmThickness(ii)]=...
                determineshielding(rivalBeta,leastSquares,grouping,...
                count,subLengths,subSum);


        end  %%%%%%%%%%%%%END SIMULATION LOOP%%%%%%%%%%%%%%
        er
```

9.2 ApplyShield.m

```
function [shielded,shield]=applyShield(BB,subSum,a,X)
%Written by Paul Kump.
%%This function reads in library BB, shielding material a, and
%%mass thickness X and applies the appropriate shielding to BB.

shield=eye(subSum(end));

if X==0
    shielded=BB;
else
[num,txt,raw]=xlsread('Shielding.xls');


BA133attenuation=num(1:subSum(1),a+1);
CE139attenuation=num(1+subSum(1):subSum(2),a+1);
CO57attenuation=num(1+subSum(2):subSum(3),a+1);
CO60attenuation=num(1+subSum(3):subSum(4),a+1);
CS137attenuation=num(1+subSum(4):subSum(5),a+1);
GA67attenuation=num(1+subSum(5):subSum(6),a+1);
I131attenuation=num(1+subSum(6):subSum(7),a+1);
K40attenuation=num(1+subSum(7):subSum(8),a+1);
```

```
NA22attenuation=num(1+subSum(8):subSum(9),a+1);
PU239attenuation=num(1+subSum(9):subSum(10),a+1);



%   for i=1:subSum(1)
%       shield(i,i)=exp(−1*BA133attenuation(i)*X);
%   end
%   j=1;
%   for i=subSum(1)+1:subSum(2)
%       shield(i,i)=exp(−1*CE139attenuation(j)*X);
%       j=j+1;
%   end
%   j=1;
%   for i=subSum(2)+1:subSum(3)
%      shield(i,i)=exp(−1*CO57attenuation(j)*X);
%      j=j+1;
%   end
%   j=1;
%   for i=subSum(3)+1:subSum(4)
%       shield(i,i)=exp(−1*CO60attenuation(j)*X);
%       j=j+1;
%   end
%   j=1;
%   for i=subSum(4)+1:subSum(5)
%       shield(i,i)=exp(−1*CS137attenuation(j)*X);
%       j=j+1;
%   end
% j=1;
% for i=subSum(5)+1:subSum(6)
%     shield(i,i)=exp(−1*GA67attenuation(j)*X);
%     j=j+1;
% end
j=1;
for i=subSum(6)+1:subSum(7)
    shield(i,i)=exp(−1*I131attenuation(j)*X);
    j=j+1;
end
j=1;
% for i=subSum(7)+1:subSum(8)
%     shield(i,i)=exp(−1*K40attenuation(j)*X);
%     j=j+1;
% end
% j=1;
% for i=subSum(8)+1:subSum(9)
%     shield(i,i)=exp(−1*NA22attenuation(j)*X);
%     j=j+1;
% end
j=1;
for i=subSum(9)+1:subSum(10)
    shield(i,i)=exp(-1*PU239attenuation(j)*X);
    j=j+1;
end

shielded=BB*shield;
shield;
end
```

9.3 DetermineShielding.m

```

function [shieldMaterial,massThickness,cmThickness]=...
    determineshielding(beta,betaLS,group,...
    count,subLengths,subSum)
%Written by Paul Kump
%if we are only testing one subspectra of the material in
% question, shield detection is impossible.
if length(find(count>1))==0

    shieldMaterial=-1;
    massThickness=0;
    cmThickness=0;
else    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%begin detection

    threshold=.1;     %determining if there is a shield or not
    t1=40;   %realistically, maximum shielding thickness in cm

%%%%extract shielding information%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ind=find(beta>0);
    lind=length(ind);

    bin(1)=group(1);
    pass=0;
    for i=2:lind
        if group(i)~=group(i-1)
            bin(i-pass)=group(i);
        else
            pass=pass+1;
        end
    end

    bin;
    information=[];
    pass=0;
    d=0;
    for j=1:length(bin)-1
        for jj=1:subLengths(bin(j))-1
          information(jj+pass)=betaLS(jj+pass+d)/...
            betaLS(jj+pass+1+d);
          information(jj+pass)=log(information(jj+pass));
        end
        d=d+1;
        pass=jj+pass;
    end
    information;
    lindi=length(information);
%%%%%%%%%%%Is there a shield?%%%%%%%%%%%%%%%%%%%%%%%
    %no, there is not a shield
    if norm(information)/sqrt(lindi)<threshold
        shieldMaterial=0;
        massThickness=0;
        cmThickness=0;

    else                %yes, there is a shield, now find it.

        [num,txt,raw]=xlsread('Shielding.xls');
        densities=[2.62 2.3 11.34 1];


        for z=1:4
            pass=0;
            for i=1:length(bin)-1
                for j=1:subLengths(bin(i))-1
                    if bin(i)==1
                        y(j+pass,z)=num(j+1,1+z)-num(j,1+z);
                    else
                      y(j+pass,z)=num((subSum(bin(i)-1)+j+1),...
                      1+z)- num((subSum(bin(i)-1)+j),1+z);
                    end
                end
                pass=pass+j;
            end
        end

       y;
        scaling=y'*information'/(information*information');

        for z=1:4
            J(z)=(norm(scaling(z)*information'-y(:,z)))^2;
        end

        J;
        [minJ, shieldMaterial]=min(J);
        massThickness=1/scaling(shieldMaterial);
        cmThickness=massThickness/densities(shieldMaterial);

    end
end
```

9.4 MAdPosiLasso.m

```
%Modified adaptive possive lasso
%Written by Paul Kump and Erwei Bai.
clear

load WEdata\CO60
load WEdata\CO57
load WEdata\NA22
load WEdata\CS137
load WEdata\I131
load WEdata\K40
load WEdata\U235
load WEdata\PU238
load WEdata\BA133
load WEdata\CE139
load WEdata\GA67
load WEdata\PU239
load WEdata\BACKGND

BB=[PU239(:,end) GA67(:,end) CS137(:,end) U235(:,end)...
    K40(:,end) NA22(:,end) BA133(:,end) CE139(:,end)...
    I131(:,end) CO57(:,end) CO60(:,end) BACKGND(:,end)];

[mb nb]=size(BB);
beta=[1/30*[0.2;0;0;0;0;0;0;0;1];0;0;1];
% beta=[0.1;0;0;0.1;0;0;0;0;0;0;0;1];
```

```
er=zeros(1,nb);
t1=0.00009;  %threshold for zero
t2=0.00001;  %stopping criterion for iteration
lambda=.001;

for ii=1:1  %# of Monte Carlo simulations
    ii;
     bh=[];
     p1=1;  %number average
     YY=zeros(mb,1);
     for i=1:p1
        YY=YY+poissrnd(BB*beta,[mb,1])/p1;
     end
     for i=1:mb
        YY1(i,1)=YY(i)*sqrt(1/YY(i));
         BB1(i,:)=BB(i,:)*sqrt(1/YY(i));
     end

     bh(:,1)=abs(inv(BB1'*BB1)*BB1'*YY1);
     e1=1;
     i1=1;

   while e1 > t2
      ind=find(bh(:,i1) >= t1); %find > 0 elements in beta
      lind=length(ind);
      IND(:,i1)=zeros(nb,1);
      for kk=1:lind
          IND(kk,i1)=ind(kk);
      end
      phi=[];
      bb=[];
      w=[];
        for k=1:lind
          phi(:,k)=BB1(:,ind(k));  %corr column of beta >0
          w(k,1)=lambda*1/bh(ind(k),i1)*sum(phi(:,k));%weights
        end
      A=phi'*phi;
      C=(phi'*YY1-1/2*w);
      i=1;
      e=1;
       bb(:,1)=abs(inv(A)*phi'*YY1);

      while e > t2  %solve positive lasso
         bhat1=bb(:,i);

        for k=1:lind
```

```
          bhat1(k)=max((C(k)-A(k,:)*bhat1+A(k,k)*...
            bhat1(k))/A(k,k),0);
        end
        i=i+1;
         bb(:,i)=bhat1;
         e=norm(bb(:,i)-bb(:,i-1))/norm(bb(:,i-1));
      end

      for k1=1:lind  %construct new estimate beta
         bh(ind(k1),i1+1)=bb(k1,end);
      end
      %stoping criterion
      e1=norm(bh(:,i1+1)-bh(:,i1))/norm(bh(:,i1));
      i1=i1+1;
   end
   bhat(:,ii)=bh(:,end);
   for i2=1:nb
     if bhat(i2,ii) >= t1
         er(i2)=er(i2)+1;
     end
   end
end
SNR=10*log10(1/15*(.2*sum(sum(PU239))+sum(sum(I131)))...
    /sum(sum(BACKGND)))
er
```

9.5 PLasso.m

```

function [ estimate ] = pLasso( BB1, YY1, lambda )
%Solve the positive LASSO for fixed regularization parameter
%   Algorithm is a coordinate ascent algorithm
%Written by Erwei Bai and Paul Kump.
    [m p]=size(BB1);
    t1=0.00009;
    A=BB1'*BB1;
    C=(BB1'*YY1-1/2*lambda*ones(p,1));
    h=1;
    e1=1;
    bb(:,1)=abs(inv(A)*BB1'*YY1);
    while e1 > t1  %solve positive lasso
        bhat1=bb(:,h);

        for k=1:p
            bhat1(k)=max((C(k)−A(k,:)*bhat1+A(k,k)...
                *bhat1(k))/A(k,k),0);
        end
        h=h+1;
        bb(:,h)=bhat1;
        e1=norm(bb(:,h)−bb(:,h−1))/norm(bb(:,h−1));
    end
    estimate=bb(:,end);

end
```

9.6 Rival.m

```

function beta=rival(X,y,lambda,Q)
%%RIVAL with no positivity
%Written by Paul Kump and Erwei Bai.

t1=0.000009;  %threshold for zero
t2=0.00001;  %stopping criterion for iteration

if Q==0
   bls=inv(X'*X)*X'*y;
   w=1./bls.^2;
   bh=LassoShootingAdapt(X,y,lambda,w);
else
   bh=[];
   bh(:,1)=abs(inv(X'*X)*X'*y);
   e1=1;
   i1=1;
   weight=[];
   while e1 > t2
   %find > 0 elements in beta
         ind=find(abs(bh(:,i1)) >= t1);
         lind=length(ind);
         if lind==0
             e1=0;
         else
             phi=[];
             bb=[];
```

```
                w=[];
                if i1==1
                   for k=1:lind
                       phi(:,k)=X(:,ind(k));
                       w(k,1)=1/(abs(bh(ind(k),i1)).^2)   %weights
                       weight(ind(k),i1)=w(k,1);
                   end
                else
                   for k=1:lind
                       phi(:,k)=X(:,ind(k));
                       w(k,1)=(Q/(abs(bh(ind(k),i1)).^2)+(1-Q)...
                           *weight(ind(k),i1-1));  %weights
                       weight(ind(k),i1)=w(k,1);
                   end
                end

                bb=LassoShootingAdapt(phi,y,lambda,w);

                for k1=1:lind  %construct new estimate beta
                    bh(ind(k1),i1+1)=bb(k1,end);
                end
                e1=norm(bh(:,i1+1)-bh(:,i1))/norm(bh(:,i1));
                i1=i1+1;
            end
     end
end
beta=bh(:,end);
```

9.7 PoiLandSub.m

```

%The following code, written by Er-wei Bai, uses the method of
%Lasso (AIC, BIC) + Subsampling or Bootstrap or Asymptotic
%approximation to detect nuclear materials.  The Lasso is
%solved using the LARS algorithm as opposed to simulating over
%a grid of candidate lambdas.  Both the AIC and BIC methods are
%used to choose correct model dimension.

clear
close all

load WEdata\CO60
load WEdata\CO57
load WEdata\NA22
load WEdata\CS137
load WEdata\I131
load WEdata\K40
load WEdata\U235
load WEdata\PU238
load WEdata\BA133
load WEdata\CE139
load WEdata\GA67
load WEdata\PU239
load WEdata\BACKGND
% load WEdata\Whatisthis2

% BB=[10*PU238(:,end) PU239(:,end) GA67(:,end) CS137(:,end)...
% U235(:,end) K40(:,end) NA22(:,end) BA133(:,end) ...
% CE139(:,end) I131(:,end) CO57(:,end) CO60(:,end) ...
% BACKGND(:,end) ones(1024,1)];
BB=[PU239(:,end) GA67(:,end) CS137(:,end) U235(:,end)...
    K40(:,end) NA22(:,end) BA133(:,end) CE139(:,end)...
    I131(:,end) CO57(:,end) CO60(:,end) BACKGND(:,end)];

[M nb]=size(BB);

  err=zeros(6,2);
%1st row=AIC, 2nd=BIC 3rd=AIC+B or S
%4th=BIC+B or S, 5th=AIC+A, 6th=BIC+A
  FE=zeros(1,nb-1);
  eA=zeros(2,nb-1); %error of Lasso(AIC)
  eB=zeros(2,nb-1); %error of Lasso(BIC)
  esub=zeros(2,nb-1);  %subsampling or asymptotic only
  eAB=zeros(2,nb-1); %error of Lasso(AIC)+B or S
  eBB=zeros(2,nb-1); %error of Lasso(BIC) +B or S
%   eAA=zeros(2,nb-1); %error of Lasso(AIC)+A
%   eBA=zeros(2,nb-1); %error of Lasso(BIC) +A
%   beta(nb,1)=0;
%   beta(nb-1)=1;
beta=[1*[0.2 0 0 0 0 0 0 0 1] 0 0 1]';
FR=[1 0 0 0 0 0 0 0 1 0 0 1 ];

for ii=1:10
      ii

%  s=rand(1,nb-2);
% for k1=1:nb-2
%       if s(k1) > 0.6
%           beta(k1)=rand+0.05;
%           FE(1,k1)=FE(1,k1)+1;
%           FR(k1)=1;   %whether material k1 is present or not
%       else FR(k1)=0; beta(k1)=0;
%       end
% end

    p1=5;
   YY=zeros(M,1);
  for i=1:p1
   YY=YY+poissrnd(BB*beta,[M,1])/p1;
  end
    for i=1:M
      YY1(i,1)=YY(i)*sqrt(1/YY(i));
      BB1(i,:)=BB(i,:)*sqrt(1/YY(i));
    end

lassob = lars(BB1, YY1, 'lasso', 0, 0, [], 1);
normsize = sum(abs(lassob),2)/sum(abs(lassob(end,:)));

[ac al]=size(lassob);

ath3=0;
AICs=[];
BICs=[];
AICs(1)=10^(10);
BICs(1)=10^(10);
for i=2:ac
    CC=[];
    bb=lassob(i,:);
    Ibb=find(bb > ath3);
    Lbb=length(Ibb);
    for k=1:Lbb
        CC(:,k)=BB1(:,Ibb(k));
    end
    AICs(i)=M*log(norm(YY1-CC*inv(CC'*CC)*CC'*YY1)^2/M)+2*Lbb;
    BICs(i)=M*log(norm(YY1-CC*inv(CC'*CC)*CC'*YY1)^2/M)...
        +log(1024)*Lbb;
%      AICs(i)=norm(YY1-CC*inv(CC'*CC)*CC'*YY1)^2*...
%         (1+2*Lbb/(M-Lbb))/M;
%      BICs(i)=norm(YY1-CC*inv(CC'*CC)*CC'*YY1)^2*...
%     (1+log(1024)*Lbb/(M-Lbb))/M;
end
[a1,a2]=min(AICs);
[b1,b2]=min(BICs);

```

```
% % Bootstrap
% for k=1:1000
%    for i=1:length(YY1)
%      r=ceil(rand*1024);
%      B(i,:)=BB1(r,:);
%      Y(i,1)=YY1(r);
%    end
%    hb(:,k)=inv(B'*B)*B'*Y;
% end

%resampling to generate distribution of parameters
N=980;
for i=1:M-N
 B=BB1(i:i-1+N,:);
 Y=YY1(i:i-1+N);
 hb(:,i)=inv(B'*B)*B'*Y; %LS estimate
end

% %asymptotic approximation
% hba=inv(BB1'*BB1)*BB1'*YY1;
% Yhat=BB1*hba;
% sigmahat=(Yhat-YY1)'*(Yhat-YY1)/(M-nb);
% P=inv(BB1'*BB1)/M;

ath2=0.05;
for i=1:nb-1
   ab= mean(hb(i,:))+3*std(hb(i,:));  %Bootstrap or Subsampling
%    as= hba(i)+3*(sigmahat*P(i,i)/M);%asymptotic approximation
%
%     if lassob(a2,i) < ath3  & FR(i)==1  %AIC only
%         err(1,1)=err(1,1)+1; eA(1,i)=eA(1,i)+1;
%     end
%     if lassob(a2,i) >= ath3  & FR(i)==0
%         err(1,2)=err(1,2)+1; eA(2,i)=eA(2,i)+1;
%     end
%     if lassob(b2,i) <ath3 & FR(i)==1  %BIC only
%         err(2,1)=err(2,1)+1;  eB(1,i)=eB(1,i)+1;
%     end
%     if lassob(b2,i) >= ath3 & FR(i)==0
%         err(2,2)=err(2,2)+1; eB(2,i)=eB(2,i)+1;
%     end
    if (lassob(a2,i) <ath3 |  ab< ath2) & (FR(i)==1) %AIC +B
        err(3,1)=err(3,1)+1; eAB(1,i)=eAB(1,i)+1;
    end
    if (lassob(a2,i) >= ath3 &  ab > ath2) & (FR(i)==0)
        err(3,2)=err(3,2)+1; eAB(2,i)=eAB(2,i)+1;
        end
        if (lassob(b2,i) <ath3 |  ab < ath2) & (FR(i)==1) %BIC+B
            err(4,1)=err(4,1)+1; eBB(1,i)=eBB(1,i)+1;
        end
        if (lassob(b2,i) >= ath3 & ab > ath2) & (FR(i)==0)
            err(4,2)=err(4,2)+1; eBB(2,i)=eBB(2,i)+1;
        end
%       if (ab < ath2 & FR(i)==1) %B or subsampling only
%           err(5,1)=err(5,1)+1; esub(1,i)=esub(1,i)+1;
%       end
%       if (ab > ath2 & FR(i)==0)
%           err(5,2)=err(5,2)+1; esub(2,i)=esub(2,i)+1;
%       end
%       if (lassob(a2,i) <ath3 |  as< ath2) & (FR(i)==1) %AIC +A
%           err(5,1)=err(5,1)+1; eAA(1,i)=eAA(1,i)+1;
%       end
%       if (lassob(a2,i) >= ath3 &  as > ath2) & (FR(i)==0)
%           err(5,2)=err(5,2)+1; eAA(2,i)=eAA(2,i)+1;
%       end
%       if (lassob(b2,i) <ath3 |  as < ath2) & (FR(i)==1) %BIC+A
%           err(6,1)=err(6,1)+1; eBA(1,i)=eBA(1,i)+1;
%       end
%       if (lassob(b2,i) >= ath3 & as > ath2) & (FR(i)==0)
%           err(6,2)=err(6,2)+1; eBA(2,i)=eBA(2,i)+1;
%       end
end



end



% FE


eA
eB
eAB
eBB
esub
% eAA
% eBA

err
```

9.8 TLSlassoNuclear2.m

```

%The following code was written by Paul Kump and introduces
%the method of total least squares to regularization techniques
%like LASSO, positive LASSO, and RIVAL.  The performances of
%each technique can be compared with one another by commenting
%out blocks of methods.  The optimum regularization
%parameter is determined by simulating over a grid of lambdas
% and minimizing the BIC value.


clear

load CO60
load CO57
load NA22
load CS137
load I131
load K40
load U235
load PU238
load BA133
load CE139
load GA67
load PU239
load BACKGND

BB=[PU239(:,end) GA67(:,end) CS137(:,end) U235(:,end)...
    K40(:,end) NA22(:,end) BA133(:,end) CE139(:,end)...
    I131(:,end) CO57(:,end) CO60(:,end) BACKGND(:,end)];
[n p]=size(BB);
alpha=10;
Er=zeros(n,p);
er=zeros(2,p);
test=zeros(2,1);

beta=[alpha*[0.2;0;0;0;0;0;0;0;1];0;0;1];
SNR=10*log10(alpha*(.2*sum(sum(PU239))+sum(sum(I131)))/...
    sum(sum(BACKGND)))
t1=0.0009; %iteration stopping criterion
t2=0.000009; %threshold for zero

for iii=1:1
```

```
    iii
    YY=poissrnd(BB*beta,[n,1]);
    for i=1:n
        YY1(i,1)=YY(i)*sqrt(1/YY(i));
        BB1(i,:)=BB(i,:)*sqrt(1/YY(i));
    end
%%%%%%%%%%%% LASSO + TLS %%%%%%%%%%%%%%%%%%%%%%%%%%
%     %BB1=normalize(BB1);
%     %YY1=center(YY1);
%     lam=1;
%     for lambda=0:1000:10000;
%         %solve the Total Least Squares LASSO
%         E=[];
%         bb=[];
%         e=1;
%         E(:,:,1)=zeros(n,p);
%         bb(:,1)=LassoShooting(BB1+E(:,:,1),...
%              YY1,lambda);
%         ii=1;
%         while e>t1
%             E(:,:,ii+1)=(YY1-BB1*bb(:,ii))*bb(:,ii)'*...
%               inv((eye(p,p)+bb(:,ii)*bb(:,ii)'));
%             bb(:,ii+1)=LassoShooting(BB1+E(:,:,ii+1),YY1,...
%               lambda);
%             e=norm(bb(:,ii+1)-bb(:,ii))/norm(bb(ii));
%             ii=ii+1;
%         end
%
%         estimateTLS(:,lam)=bb(:,end);
%         BIC(lam)=informationCriterion(YY1, BB1+E(:,:,end),...
%            estimateTLS(:,lam),1);
%         lam=lam+1;
%     end
%
%     [minBIC,minIndex]=min(BIC);
%     best(:,iii)=estimateTLS(:,minIndex);
%
%     for jj=1:p
%         if abs(best(jj,iii))>t2
%             er(1,jj)=er(1,jj)+1;
%         end
%     end
%%%%%%%%%%%%%  END LASSO  + TLS %%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%  JUST LASSO %%%%%%%%%%%%%%%%%%%%%%

```

```
%      BB1=normalize(BB1);
%      YY1=center(YY1);
%      estimateLars=[];
%      estimateLars=lars(BB1, YY1, 'lasso', 0, 0, [], 0);
%      [m p]=size(estimateLars);
%      for kk=1:m
%          bb(:,kk)=estimateLars(kk,:)';
%          for jj=1:p
%              if abs(bb(jj:kk))<t2
%                  bb(jj:kk)==0;
%              end
%          end
%          BIC(kk)=informationCriterion(YY1,BB1,bb(:,kk),1);
%          %AIC(kk)=informationCriterion(YY1,BB1,bb(:,kk),0);
%      end
%
%
%      [minBIC,minIndex]=min(BIC);
%      best(:,iii)=bb(:,minIndex);
%
%      for jj=1:p
%          if abs(best(jj,iii))>t2
%              er(1,jj)=er(1,jj)+1;
%          end
%      end


                 %%%%%%%%%%%%%%%%%%%%%%%%%%%%
                 %        OR....           %
                 %%%%%%%%%%%%%%%%%%%%%%%%%%%%

%       BB1=normalize(BB1);
%       YY1=center(YY1);
%       lam=1;
%       for lambda=0:1:20
%            %solve the LASSO
%          bb(:,1)=LassoShooting(BB1,YY1,lambda);
%
%          estimate(:,lam)=bb(:,end);
%          AIC(lam)=informationCriterion(YY1, BB1, ...
%                estimate(:,lam),1);
%          lam=lam+1;
%      end
%
%      [minAIC,minIndex]=min(AIC);
%      best(:,iii)=estimate(:,minIndex);
```

```
%
%     for jj=1:p
%         if abs(best(jj,iii))>t2
%             er(1,jj)=er(1,jj)+1;
%         end
%     end

%%%%%%%%%%%%% END JUST LASSO %%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%Positive LASSO + TLS %%%%%%%%%%%%%%

%      BB1=normalize(BB1);
%      YY1=center(YY1);
%      lam=1;
%      for lambda=65:65;
%          %solve the Total Least Squares Positive LASSO
%         E=[];
%         bb=[];
%         e=1;
%         E(:,:,1)=zeros(n,p);
%         bb(:,1)=pLasso(BB1+E(:,:,1),YY1,lambda);
%         ii=1;
%         while e>t1
%             E(:,:,ii+1)=(YY1−BB1*bb(:,ii))*bb(:,ii)'*...
%               inv((eye(p,p)+bb(:,ii)*bb(:,ii)'));
%             bb(:,ii+1)=pLasso(BB1+E(:,:,ii+1),YY1,lambda);
%             e=norm(bb(:,ii+1)−bb(:,ii))/norm(bb(ii));
%             ii=ii+1;
%         end
%
%         estimateTLS(:,lam)=bb(:,end);
%         BIC(lam)=informationCriterion(YY1,...
%               (BB1−E(:,:,end)), estimateTLS(:,lam),1);
%         lam=lam+1;
%     end
%
%     [minBIC,minIndex]=min(BIC);
%     best(:,iii)=estimateTLS(:,minIndex);
%
%     for jj=1:p
%         if abs(best(jj,iii))>t2
%             er(1,jj)=er(1,jj)+1;
%         end
%     end

%%%%%%%%%%%%End Positive LASSO + TLS %%%%%%%%%%%%%%%%%%
```

```

%%%%%%%%%%%%%% Just Positive LASSO %%%%%%%%%%%%%%%%%%%%%%%%%
%
%      BB1=normalize(BB1);
%      YY1=center(YY1);
%      estimateLars=[];
%      estimateLars=pLars(BB1, YY1, 'lasso', 0, 0, [], 0);
%      [m p]=size(estimateLars);
%      for kk=1:m
%          bb(:,kk)=estimateLars(kk,:)';
%          BIC(kk)=informationCriterion(YY1,BB1,bb(:,kk),1);
%      end
%
%      [minBIC,minIndex]=min(BIC);
%      best(:,iii)=bb(:,minIndex);
%
%      for jj=1:p
%          if abs(best(jj,iii))>t2
%              er(1,jj)=er(1,jj)+1;
%          end
%      end
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %        OR....              %
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       BB1=normalize(BB1);
%       YY1=center(YY1);
%       lam=1;
%       for lambda=20000:20000;
%           %solve the Positive LASSO
%          bb(:,1)=pLasso(BB1,YY1,lambda);
%
%          estimate(:,lam)=bb(:,end);
%          AIC(lam)=informationCriterion(YY1, BB1, ...
%            estimate(:,lam),1);
%          lam=lam+1;
%      end
%
%      [minAIC,minIndex]=min(AIC);
%      best(:,iii)=estimate(:,minIndex);
%
%      for jj=1:p
%          if abs(best(jj,iii))>t2
%              er(1,jj)=er(1,jj)+1;
%          end
%      end
%      if abs(best(1,iii))>t2 && abs(best(9,iii))>t2 && ...
%        abs(best(12,iii))>t2&& abs(best(2,iii))<t2 && ...
%        abs(best(3,iii))<t2 && abs(best(4,iii))<t2...
%        && abs(best(5,iii))<t2 && abs(best(6,iii))<t2 &&...
%         abs(best(7,iii))<t2&& abs(best(8,iii))<t2 && ...
%         abs(best(10,iii))<t2 && abs(best(11,iii))<t2
%          test=test+1;
%      end

%%%%%%%%%%%%%%End Just Positive LASSO %%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%% RIVAL +TLS %%%%%%%%%%%%%%%%%%%%%%%%%%
    for i=1:n
        for kk=1:p-1
            Er(i,kk )=sqrt(.1*BB1(i,kk))*randn(1);
        end
            Er(i,p)=sqrt(.1*BB1(i,p))*randn(1);
    end

    BB1=BB1+Er;


     lam=1;
        for lambda=.1:.5:20.1
        e=1;
        ii=1;
        E=[];
        bb=[];
        bhat=[];
        E(:,:,ii)=zeros(n,p);
        bb=pRival(BB1+E(:,:,ii),YY1,lambda);
        bhat(:,ii)=bb;
        while e>t1 %&& ii<10
            ind=[];
            phi=[];
            EE=[];
            LS=[];
            A=[];
            ind=find(bb>t2);
            lind=length(ind);
            for k1=1:lind
                phi(:,k1)=BB1(:,ind(k1));
                EE(:,k1)=E(:,ind(k1),ii);
            end
            A=max(phi+EE,0);
            LS=abs(inv(phi'*phi)*phi'*YY1);
            EE=(YY1-phi*LS)*LS'*inv(eye(lind,lind)+LS*LS');

            for k1=1:lind
                E(:,ind(k1),ii+1)=EE(:,k1);
            end
            bb=pRival(max(BB1+E(:,:,ii+1),0),YY1,lambda);
            bhat(:,ii+1)=bb;
             e=norm(bhat(:,ii+1)-bhat(:,ii))/norm(bhat(:,ii));

%            e=norm(E(:,:,ii+1)-E(:,:,ii))/norm(E(:,:,ii));
            ii=ii+1;
        end

        estimateRival(:,lam)=bhat(:,1);
        ind1=[];
        LS1=[];
        phi1=[];
        ind1=find(estimateRival(:,lam)>t2);
        lind1=length(ind1);

        for k11=1:lind1
            phi1(:,k11)=BB1(:,ind1(k11));
        end

        LS1=abs(inv(phi1'*phi1)*phi1'*YY1);
        BICrival(lam)=informationCriterion(YY1,phi1,LS1,1);

        estimateTLS(:,lam)=bhat(:,end);
        ind2=[];
        LS2=[];
        EE2=[];
        A2=[];
        phi2=[];
        ind2=find(estimateTLS(:,lam)>t2);
        lind2=length(ind2);

        for k12=1:lind2
            phi2(:,k12)=BB1(:,ind2(k12));
            EE2(:,k12)=E(:,ind2(k12),end);
        end

        A2=max(phi2+EE2,0);
        LS2=abs(inv(A2'*A2)*A2'*YY1);
        BICTLS(lam)=informationCriterion(YY1,A2,LS2,1);

        lam=lam+1;
    end
```

```

%       for s1=1:n
%           W(s1,:,:)=E(s1,:,:);%*sqrt(YY(s1));
%       end
%
%       for k2=1:ii
%           error(k2)=norm(Er-W(:,:,k2));
%       end
%
%       plot(error)

     [minBICr, indexRival]=min(BICrival);
     bestRival=estimateRival(:,indexRival);

     [minBICt, indexTLS]=min(BICTLS);
     bestTLS=estimateTLS(:,indexTLS);

     for jj=1:p
        if abs(bestRival(jj))>t2
            er(1,jj)=er(1,jj)+1;
        end
         if abs(bestTLS(jj))>t2
            er(2,jj)=er(2,jj)+1;
         end
     end

    if abs(bestRival(1))>t2 && abs(bestRival(9))>t2 && ...
        abs(bestRival(12))>t2&& abs(bestRival(2))<t2 && ...
        abs(bestRival(3))<t2 && abs(bestRival(4))<t2...
        && abs(bestRival(5))<t2 && abs(bestRival(6))<t2 && ...
        abs(bestRival(7))<t2&& abs(bestRival(8))<t2 && ...
        abs(bestRival(10))<t2 && abs(bestRival(11))<t2
        test(1)=test(1)+1;
    end

    if abs(bestTLS(1))>t2 && abs(bestTLS(9))>t2 && ...
    abs(bestTLS(12))>t2&& abs(bestTLS(2))<t2 && ...
    abs(bestTLS(3))<t2 && abs(bestTLS(4))<t2...
    && abs(bestTLS(5))<t2 && abs(bestTLS(6))<t2 && ...
    abs(bestTLS(7))<t2 && abs(bestTLS(8))<t2 && ...
    abs(bestTLS(10))<t2 && abs(bestTLS(11))<t2
        test(2)=test(2)+1;
    end

%%%%%%%%%%%%end RIVAL +TLS stuff  %%%%%%%%%%%%%%%%%%%%%%%%

```

```

%%%%%%%%%%%%% Just RIVAL %%%%%%%%%%%%%%%%%%%%%%%%%
%
%       lam=1;
%       for lambda=2:2
%          bb(:,1)=pRival(BB1,YY1,lambda);
%
%          estimate(:,lam)=bb(:,end);
%          AIC(lam)=informationCriterion(YY1, BB1, estimate...
%            (:,lam),1);
%          lam=lam+1;
%      end
%
%      [minAIC,minIndex]=min(AIC);
%      best(:,iii)=estimate(:,minIndex);
%
%      for jj=1:p
%          if abs(best(jj,iii))>t2
%              er(1,jj)=er(1,jj)+1;
%          end
%      end
%      if abs(best(1,iii))>t2 && abs(best(9,iii))>t2 && ...
%        abs(best(12,iii))>t2&& abs(best(2,iii))<t2 &&...
%        abs(best(3,iii))<t2 && abs(best(4,iii))<t2...
%        && abs(best(5,iii))<t2 && abs(best(6,iii))<t2 && ...
%        abs(best(7,iii))<t2&& abs(best(8,iii))<t2 && ...
%        abs(best(10,iii))<t2 && abs(best(11,iii))<t2
%          test=test+1;
%      end

%%%%%%%%%%%%%end Just RIVAL stuff  %%%%%%%%%%%%%%%%%%%%%%%%%%

end
 er
%  plot(YYY)
%  hold on
%  plot(YY1,'r')
%  difference=norm(YYY-YY1)/norm(YY1)
CIR(1,1)=1-(3000-er(1,1)-er(1,9)-er(1,12))/3000 -(er(1,2)...
    +er(1,3)+er(1,4)+er(1,5)+er(1,6)+er(1,7)...
    + er(1,8)+er(1,10)+er(1,11))/9000;
CIR(2,1)=1-(3000-er(2,1)-er(2,9)-er(2,12))/3000 -(er(2,2)...
    +er(2,3)+er(2,4)+er(2,5)+er(2,6)+er(2,7)...
    + er(2,8)+er(2,10)+er(2,11))/9000;
CIR
 test/iii
```

9.9 TLSMaster.m

```

%The following MATLAB code, written by Paul Kump, uses the
%group RIVAL algorithm in conjuction with the method of total
%least squares to detect simulated nuclear materials when the
%library is uncertain.  It does so by applying RIVAL with a
%grid of candidate lambdas, trimming off the irrelevant
%variables, computing least squares with new set, then
%computing AIC/BIC for each of the candidate lambdas.  The code
%selects that lambda that minimizes this AIC/BIC and selects the
%estimate given from this lambda.

%Shielding can be simulated by calling the function
%applyShield.m.  This function simulates shielding as the
%library is scaled by exponentials with powers of mass
%thickness times shielding coefficients.  The shielding
%material and thickness can be estimated by calling
%determineshielding.m.
%%

clear

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    %SETUP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[num,txt,raw]=xlsread('isotopesNS.xls');

BA133=num(:,1:7);
CE139=num(:,8);
CO57=num(:,9:10);
CO60=num(:,11:14);
CS137=num(:,15:18);
GA67=num(:,19:24);
I131=num(:,25:29);
K40=num(:,30);
NA22=num(:,31:32);
PU239=num(:,33:42);
BACKGND=1/10*num(:,43);

numberOfMaterials=11;  %%materials plus background


BB=[BA133 CE139 CO57 CO60 CS137...
```

```
    GA67 I131 K40 NA22 PU239 BACKGND];

[n p]=size(BB);
[BA133rows, BA133columns]=size(BA133);
[CE139rows, CE139columns]=size(CE139);
[CO57rows, CO57columns]=size(CO57);
[CO60rows, CO60columns]=size(CO60);
[CS137rows, CS137columns]=size(CS137);
[GA67rows, GA67columns]=size(GA67);
[I131rows, I131columns]=size(I131);
[K40rows, K40columns]=size(K40);
[NA22rows, NA22columns]=size(NA22);
[PU239rows, PU239columns]=size(PU239);
[BACKGNDrows, BACKGNDcolumns]=size(BACKGND);

subLengths=[BA133columns CE139columns CO57columns ...
    CO60columns CS137columns GA67columns I131columns...
     K40columns NA22columns PU239columns BACKGNDcolumns];

subSum=cumsum(subLengths);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                   %CHOOSE SHIELD AND APPLY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


[BBprime,shield]=applyShield(BB,subSum,2,0);
%%(data getting shielded, subSum, shielding material,...
%%       mass thickness)
%%shielding material: enter 1 for carbon
%%(graphite), enter 2 for concrete, enter 3 for
%%lead, and enter 4 for water.


 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    %CHOOSE SNR OR UNCOMMENT AND CHOOSE ALPHA
   %ALPHA WILL BE DETERMINED BY SNR OR SNR WILL BE DETERMINED
   %BY ALPHA
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%   alpha=10;
%   SNR=10*log10(alpha*(sum(sum(.2*BBprime(:,33:42)))+...
%        sum(sum(BBprime(:,25:29))))/(sum(sum(BBprime(:,end)))));
SNR=-15;
```

```
alpha=10^(SNR/10)*sum(sum(BACKGND))/(sum(sum(.2*PU239))+...
    sum(sum(I131)));  %%signal strength.

beta=[alpha*[zeros(subLengths(1),1);zeros(subLengths(2),1); ...
    zeros(subLengths(3),1);zeros(subLengths(4),1);  ...
    zeros(subLengths(5),1); zeros(subLengths(6),1);...
    1*ones(subLengths(7),1); zeros(subLengths(8),1);...
    zeros(subLengths(9),1); 0.2*ones(subLengths(10),1)]; ...
    ones(subLengths(11),1)];


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                         %DEFINE THRESHOLDS
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

t1=0.0000009; %threshold for zero
t2=0.00001;  %stopping criterion for iteration

for s=1:numberOfMaterials
    thresholdArray(s)=t1*sqrt(subLengths(s));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    %BEGIN SIMULATIONS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

simulations=1;
er=zeros(2,numberOfMaterials);
test=zeros(2,1);
for ii=1:simulations %# of Monte Carlo simulations
    ii
     YY=poissrnd(BBprime*beta,[n,1]);
     for i=1:n
        YY1(i,1)=YY(i)*sqrt(1/YY(i));
         BB1(i,:)=BB(i,:)*sqrt(1/YY(i));
     end
     for i=1:n
        for kk=1:p-1
            Er(i,kk)=sqrt(.02*BB1(i,kk))*randn(1);
        end
            Er(i,p)=sqrt(.02*BB1(i,p))*randn(1);
     end
     BB1=BB1+Er;

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                        %BEGIN MAIN group TLS RIVAL LOOP
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
index=1;
for lambda=.03:.001:.05
     e=1;
     ii=1;
     E=[];
     bb=[];
     bhat=[];
     E(:,:,ii)=zeros(n,p);
     bb=pRivalGroup(BB1+E(:,:,ii),YY1,lambda,...
         numberOfMaterials,subSum);
     bhat(:,ii)=bb;
     normArray(1,ii)=norm(bb(1:subSum(1),1));
     for x=2:numberOfMaterials
         normArray(x,ii)=norm(bb(1+subSum(x-1):subSum(x),1));
     end
     while e>t1
         phi=[];
         EE=[];
         LS=[];
         A=[];
         pass=0;
         gg=zeros(p);
         for xx=1:numberOfMaterials
             if normArray(xx,ii)>=thresholdArray(xx)
                 for xxx=1:subLengths(xx)
                     if xx==1
                         phi(:,xxx+pass)=BB1(:,xxx);
                         EE(:,xxx+pass)=E(:,xxx,ii);
                         gg(xxx)=1;
                     else
                         phi(:,xxx+pass)=...
                             BB1(:,subSum(xx-1)+xxx);
                         EE(:,xxx+pass)=...
                             E(:,subSum(xx-1)+xxx,ii);
                         gg(subSum(xx-1)+xxx)=1;
                     end
                 end
                 pass=xxx+pass;
             end
         end
         A=max(phi+EE,0);
         [nn,pp]=size(EE);
         LS=abs(inv(phi'*phi)*phi'*YY1);
         EE=(YY1-phi*LS)*LS'*inv(eye(pp,pp)+LS*LS');
         ind=find(gg>0);
```

```

            for k1=1:pp
                E(:,ind(k1),ii+1)=EE(:,k1);
            end
            bb=pRivalGroup(max(BB1+E(:,:,ii+1),0),YY1,lambda,...
                numberOfMaterials,subSum);
            bhat(:,ii+1)=bb;
            e=norm(bhat(:,ii+1)-bhat(:,ii))/norm(bhat(:,ii));

            ii=ii+1;

            normArray(1,ii)=norm(bb(1:subSum(1),1));
            for x=2:numberOfMaterials
                normArray(x,ii)=norm(bb(1+subSum(x-1):...
                    subSum(x),1));
            end
        end

        estimateRival(:,index)=bhat(:,1);
        LS1=[];
        phi1=[];
        pass=0;
        for xx=1:numberOfMaterials
            if normArray(xx,ii)>=thresholdArray(xx)
                for xxx=1:subLengths(xx)
                    if xx==1
                        phi1(:,xxx+pass)=BB1(:,xxx);
                    else
                        phi1(:,xxx+pass)=BB1(:,...
                            subSum(xx-1)+xxx);
                    end
                end
                pass=xxx+pass;
            end
        end
        LS1=abs(inv(phi1'*phi1)*phi1'*YY1);
        BICrival(index)=informationCriterion(YY1,phi1,LS1,1);

        estimateTLS(:,index)=bhat(:,end);
        LS2=[];
        EE2=[];
        A2=[];
        phi2=[];
        pass=0;
        for xx=1:numberOfMaterials
            if normArray(xx,ii)>=thresholdArray(xx)
```

```
                 for xxx=1:subLengths(xx)
                     if xx==1
                         phi2(:,xxx+pass)=BB1(:,xxx);
                         EE2(:,xxx+pass)=E(:,xxx,end);
                     else
                         phi2(:,xxx+pass)=BB1(:,subSum(xx-1)...
                             +xxx);
                         EE2(:,xxx+pass)=E(:,subSum(xx-1)+xxx...
                             ,end);
                     end
                 end
                 pass=xxx+pass;
             end
         end

         A2=max(phi2+EE2,0);
         LS2=abs(inv(A2'*A2)*A2'*YY1);
         BICTLS(index)=informationCriterion(YY1,A2,LS2,1);

         index=index+1;
    end    %%%%%%%%%%%%%end main RIVAL loop%%%%%%%%%%%%%%%%

     [minBICr,indexRival]=min(BICrival);
     bestRival=estimateRival(:,indexRival);

     [minBICt,indexTLS]=min(BICTLS);
     bestTLS=estimateTLS(:,indexTLS);

     for xx=1:numberOfMaterials
         if normArray(xx,1)>=thresholdArray(xx)
             er(1,xx)=er(1,xx)+1;
         end
         if normArray(xx,end)>=thresholdArray(xx)
             er(2,xx)=er(2,xx)+1;
         end
     end


     if normArray(1,1)<thresholdArray(1) && normArray(2,1)...
         <thresholdArray(2) && normArray(3,1)<...
         thresholdArray(3)&& normArray(4,1)<thresholdArray(4)...
          && normArray(5,1)<thresholdArray(5) &&...
          normArray(6,1)<thresholdArray(6) && normArray(7,1)...
          >=thresholdArray(7) && normArray(8,1)...
          <thresholdArray(8) && normArray(9,1)...
          <thresholdArray(9)&& normArray(10,1)>=...
          thresholdArray(10) && normArray(11,1)>=...
          thresholdArray(11)
         test(1,1)=test(1,1)+1;
     end

     if normArray(1,end)<thresholdArray(1) && normArray(2,end)...
         <thresholdArray(2) && normArray(3,end)...
         <thresholdArray(3)&& normArray(4,end)<...
         thresholdArray(4) && normArray(5,end)<...
         thresholdArray(5) && normArray(6,end)<...
         thresholdArray(6) && normArray(7,end)>=...
         thresholdArray(7) && normArray(8,end)...
         <thresholdArray(8) && normArray(9,end)...
         <thresholdArray(9) && normArray(10,end)>=...
         thresholdArray(10) && normArray(11,end)>=...
         thresholdArray(11)
         test(2,1)=test(2,1)+1;
     end
end  %%%%%%%%%%%%%%%%%%%%%%%%%%%END SIMULATION LOOP%%%%%%%%%%%%
er
test
```

REFERENCES

[1] K.K. Anderson, K.D. Jarman, M.L. Mann, D.M. Pfund, and R.C. Runkle. Discriminating nuclear threats from benign sources in gamma-ray spectra using a spectral comparison ratio method. 276:713–718, 2008.

[2] T.W. Anderson. *An Introduction to Multivariate Statistical Analysis*. Wiley, New York, NY, 1958.

[3] R. Ash. *Basic Probability Theory*. Dover Books, 2008.

[4] R. August and R. Whitlock. Helga 2: Autonomous passive detection of nuclear weapons materials. Technical report, Naval Research Laboratory, 2005.

[5] E. Bai, K. Chan, W. Eichinger, and P. Kump. Detection of radionuclides from weak and poorly resolved spectra using lasso and sub-sampling techniques. *Radiation Measurements*, 46:1138–1146, 2011.

[6] E. Bai, K. Chan, W. Eichinger, J. Li, and P. Kump. Physics-based weak signal analysis for nuclear material detection, March 2011. Program Review.

[7] K. Chan, J. Li, W. Eichinger, and E. Bai. A new physics-based method for detecting weak nuclear signals via spectral decomposition. *Nuclear Instruments and Methods in Physics Research A*, 667:16–25, 2011.

[8] A. Compton. A quantum theory of the scattering of x-rays by light elements. *The Physical Review*, 21:483–502, 1923.

[9] B. Efron, T. Hastie, T. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32:407–499, 2004.

[10] W. Eichinger. Interview, Apr. 2012.

[11] J. Ely, R. Kouzes, J. Schweppe, E. Sicilaino, D. Strachan, and D. Weier. The use of energy windowing to discriminate snm from norm in radiation portal monitors. *Nuclear Instruments and Methods in Physics Research*, 560:373–387, 2006.

[12] J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96:1348–1360, 2001.

[13] J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso. unpublished, February 2010.

[14] S. Garatti and R. Bitmead. On resampling and uncertainty estimation in linear system identification. *Automatica*, 46:785–795, 2010.

[15] G.H. Golub, C. Hansen, and D.P. O'Leary. Tikhonov regularization and total least squares. *Siam Journal Matrix Anal. Appl.*, 21:185–194, 1999.

[16] G.H. Golub and C.F. Van Loan. *Matrix Computations.* Johns Hopkins, Balitmore, MD, 1996.

[17] J. Gorski, F. Pfeuffer, and K. Klamroth. Biconvex sets and optimization with biconvex functions - a survey and extensions. *Mathematical Methods of Operations Research*, 66:373–407, 2007.

[18] M. Grassle, B. Thomas, and C.B. Rangel. Container security: Expansion of key customs programs will require greater attention to critical success factors. Technical report, General Accounting Office, US Congress, 2003.

[19] H. Guo and R.A. Renaut. A regularized total least squares algorithm. unpublished.

[20] T. Hastie, J. Taylor, R. Tibshirani, and G. Walther. Forward stagewise regression and the monotone lasso. *Electronic Journal of Statistics*, 1:1–29, 2007.

[21] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer Verlag, New York, 2001.

[22] E. Killian and J. Hartwell. Pcgap: Users guide and algorithm description. Technical report, Idaho National Engineering and Environmental Laboratory Bechtel BWXT Idaho, LLC, 2000.

[23] P. Kump, E. Bai, K. Chan, and B. Eichinger. Detection of shielded radionuclides from weak and poorly resolved spectra using group positive rival. *Radiation Measurements*, 2012.

[24] P. Kump, E. Bai, K. Chan, B. Eichinger, and K. Li. Variable selection via rival (removing irrelevant variables amidst lasso iterations) and its application to nuclear material detection. *Automatica*, 2012.

[25] C. Leng, Y. Lin, and G. Wahba. A note on the lasso and related procedures in model selection. *Statistica Sinica*, 16:1273–1284, 2004.

[26] M.A. Mariscotti. A method for automatic identification of peaks in the presence of background and its application to spectrum analysis. *Nuclear Inst. and Measurements*, 50:309–320, 1967.

[27] Jonathan Medalia. Detection of nuclear weapons and materials: Science, technologies, observations. Technical report, Congressional Research Service, 2010.

[28] W. Murray. Gn-4, a lightweight, battery-operated gamma-ray and neutron detection system for in situ nuclear material surveys. Technical report, Los Alamos Natl. Lab., 1998.

[29] NIST. Nist: X-ray mass attenuation coefficients. http://physics.nist.gov/PhysRefData/XrayMassCoef, Sept. 2011.

[30] T. O'Haver. Peak finding and measurement, version 2. http://terpconnect.umd.edu/toh/spectrum/PeakFindingandMeasurement.htm, 2009.

[31] D.M. Pfund, R.C. Runkle, K.K. Anderson, and K.D. Jarman. Examination of count-starved gamma spectra using the method of spectral comparison ratios. *IEEE Transactions on Nuclear Science*, 54:1232–1238, 2007.

[32] G. Phillips. Strategies and sensors for detection of nuclear weapons. Technical report, Georgetown University, 2006.

[33] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C++*. Cambridge Univeristy Press, Cambridge, UK, 2002.

[34] J.T. Routti and S.G. Prussin. Photopeak method for the computer analysis of gamma-ray spectra from semiconductor detector. *Nuclear Inst. and Meth.*, 72:125–142, 1969.

[35] R. Serway and R. Beichner. *Physics For Scientists and Engineers.* Saunders College Publishing, New York, 2000.

[36] J. Shao. An asymptotic theory for linear model selection. *Statistica Sinica*, 7:221–264, 1997.

[37] Z.K. Silagadze. A new algorithm for automatic photopeak searches. *Nuclear Inst. and Meth. In Physics Research*, 376:451–454, 1996.

[38] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B, Methodological*, 58:267–288, 1996.

[39] M. Wald. Shortage slows a program to detect nuclear bombs, November 2009.

[40] W. Wei, Q. Du, and N.H. Younan. Particle swarm optimization based spectral transformation for radioactive material detection and classification. pages 1–6. IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, September 2010.

[41] Y Yang. Can the strengths of aic and bic be shared? *Biometrika*, 92:937–950, 2005.

[42] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. Technical report, Univeristy of Wisconsin, 2004.

[43] M. Yuan and Y. Lin. On the non-negative garrote estimator. *J. R. Statist. Soc. Ser.*, 69:143–161, 2007.

[44] P. Zhao and B. Yu. On model selection consistency of lasso. *Journal of Machine Learning*, 7:2541–2563, 2006.

[45] H. Zhu, G. Leus, and G.B. Giannakis. Sparse regularized total least squares for sensing applications. unpublished.

[46] H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101:1418–1429, 2006.

[47] H. Zou, T. Hastie, and R. Tibshirani. On the degrees of freedom of the lasso. *The Annals of Statistics*, 35:2173–2192, 2007.