To the Graduate Council:

I am submitting herewith a dissertation written by Zhong Cao entitled "New Methods for Neutron-Photon Pulse Shape Discrimination." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

Laurence F. Miller, Major Professor

We have read this dissertation
and recommend its acceptance:

Accepted for the council:

Associate Vice Chancellor and
Dean of The Graduate School

NEW METHODS FOR NEUTRON-PHOTON PULSE SHAPE DISCRIMINATION

A Dissertation
Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Zhong Cao

December, 1997

To my wife Ling Zhou, my daughter Xiaohong and Xiaoqing.

ACKNOWLEDGMENTS

I would like to specially acknowledge the encouragement and guidance of my adviser Dr. L. F. Miller and to thank the members of my dissertation committee Dr. P. G. Groer, Dr. L.W. Townsend, Dr. J. T. Mihalczo, Dr. D. J. Downing,. I would also like to thank the department head Dr. H. L. Dodds as well as Dr. R. E. Uhrig, Dr. N. Hills and Dr. M. Paulus, for their comments and assistance. The technical assistance of Nuclear Engineering Department staff, Mr. G. L. Graves and Mr. R. J. Bailey, is also very much appreciated.

The author would like to acknowledge the financial support of the Instrumentation & Controls Division and Physics Division, Oak Ridge National Laboratory from 1995 to 1997, which contributed to this work. In particular, I would like to thank Mark Buckner and Dr. D. Shapira, my supervisors, for their support and assistance.

The author would like to express his sincere gratitude to Ms. Rheba and Mr. Tom Akins, friends of his family, for their kindness and assistance during the entire period of his study.

Finally, the author gratefully thanks the former department head Dr. T. W. Kerlin for providing him with a chance to fulfill his doctoral dream at the Department of Nuclear Engineering, the University of Tennessee, Knoxville.

ABSTRACT

New methods are developed that extend the state-of -art in neutron-photon pulse shape discrimination (PSD) in the neutron low energy region (< 500 keV). These include implementing a time-of-flight selection method for acquisition of "pure" neutron spectra for calibration at low energies, establishing PSD dynamic bias using neural networks, and modifying statistical models for use in evaluating PSD scintillation materials and methods. As a result, the lower energy limit for neutron-photon PSD with commercially available equipment, is reduced to 100 keV neutron energy in this work.

First, a time-of-flight selection method using a Cf-252 neutron source is developed to calibrate the cross talk between neutrons and photons in PSD at low energies. This method permits "pure" photon and "pure" neutron spectra to be obtained, which are used to construct calibration data with known ratios of photons to neutrons. These spectra are subsequently used to construct training and testing data at 15 - 150 keV equivalent electron energies for neural network classifiers. Next, perceptron, back-propagation neural networks and Bayes statistical classifiers are developed to implement dynamic bias methods over a wide energy range. All demonstrate superior neutron-photon PSD performance to the fixed bias used in conventional methods. The perceptron is simplest to use and can be easily incorporated into an IC chip. However, the Bayes classifier performs better when neutron source conditions are significantly changed. Our tests show that implementing dynamic bias classifiers enables PSD to be successfully

performed at neutron energy down to 100 keV using a 1.5"BC501A-XP2020 PMT detector.

Finally two statistical models including the transit time spread and the electronic noise contributions are established to evaluate of scintillation materials and the PSD methods. The models are used to optimize parameters of the zero-crossing and the charge comparison methods. The calculated best CFD settings for NE213, trans-stilbene and Borexino scintillators are 0.8, 0.8 and 0.9 respectively. The best fast window settings for NE213, trans-stilbene and Borexino scintillators are 0-40, 0-22 and 0-10 ns, respectively. The effects of the transit time spread and electronic noise are evaluated. The intrinsic PSD low energy limitations for different scintillation materials are estimated. Under the optimal condition, the low energy PSD limitations of early NE213, trans-stilbene and Borexino scintillators are 100, 65 and 60 keV neutron energy, respectively. Results show that under the optimal condition the zero-crossing method is better for NE213 and the trans-stilbene crystal, and the charge comparison is better for the Borexino. Using these models to predict the PSD performance under different conditions, scintillation materials can be evaluated and parameter optimization can be completed with much less experimental effort. Further efforts to improve neutron-photon PSD at low energies should concentrate on the development of new scintillation materials.

LIST OF ABBEVIATION

BPNN	Back-Propagation Neural Network.
BPNNC	Back-Propagation Neural Network Classifier.
CFD	Constant fraction discrimination.
f	The attenuation factor in a CFD trigger.
FOM	Figure of Merit.
FWHM	Full width at the half maximum.
MCA	Multiple channel analyzer.
PMT	Photomultiplier tube.
PSD	Pulse Shape Discrimination.
PPR	Photoelectron production rate.
P/V	Ratio of the neutron peak-to-valley in a PSD spectrum.
SCA	Single channel analyzer.
SEP	Single electron peak.
SPE	Single photoelectron.
TAC	Time-to-amplitude converter.
TLD	Thermoluminescent detector.
TOF	Time of flight.
T_c	The zero-crossing time.
T_d	The delay time in a constant fraction discrimination trigger.
T_r	The rise time of a signal.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

INTRODUCTION

Pulse shape discrimination (PSD) is a well-known technique that was developed in the 1960's and 1970's. The high cost of electronics has prevented wide scale use, but recent developments of microelectronics have eliminated this barrier, and applications in fields, such as environmental protection, dosimetry and medical physics are of interest.

Although effective PSD methods are well established, improved performance is needed for low energies. For example, current PSD techniques are good for nuclear data measurements, which involve energies above 300 keV. However, for the purpose of dosimetry, good PSD performance should extend to about 100 keV, since the fluence to dose equivalent conversion function begins to increase rabidly at about this energy.

The motivation of this research is to develop new methods for neutron-photon pulse shape discrimination and to extend the state-of-the-art as follows:

1. Development of a "pure" Cf-252 neutron source by using time-of-flight selection, which enables neutron-photon PSD calibration at low neutron energies without the use of an accelerator.
2. Development of a dynamic bias technique for pulse shape discrimination by using neural network and Bayes classifiers to improve neutron-photon PSD performance at low energies.

3. Establishment of statistical models to determine theoretical low energy limits for various scintillation materials in combination with specific PSD methods. Also, parameters for optimal performance of specific PSD systems are obtained from model calculations.

Chapter Two describes a method to obtain "pure" neutron sources for PSD calibration at low energies by using a time-of-flight selection technique with a ^{252}Cf neutron source, and Chapter Three outlines implementation of dynamic bias techniques by using neural network classifiers. Chapter Four describes two statistical models for the zero-crossing and the charge comparison methods, which are used for evaluation of PSD properties of scintillation materials and of PSD systems. Chapter Five cites conclusion from the present work.

1.1 Pulse Shape Discrimination (PSD) for Radiation Detection

Pulse shape discrimination is a special radiation detection technique that takes advantage of differences in signal shape produced by different types of radiation in some detectors[1-4].

When radiation impacts a detector, its energy is transferred to the detector through interactions between the radiation and the detector material, where the mechanisms depend on the type of radiation. For example, in the case of gamma-rays, the photoeletric effect, Compton scattering and pair production are the main interaction mechanisms. For neutrons and beta-rays, elastic scattering, inelastic scattering, and other nuclear reactions take place. Due to different interaction mechanisms in various detector

materials, the detector time responses associated with various types of radiation may be quite different. For example, the fast decay time component (< 6 ns) of scintillation induced by neutrons in a stilbene crystal is about 75 % of the total, while it is about 63 % of the total for photons. The decay time of scintillation caused by photons in a BC400 plastic scintillator is less than 5 ns, and it is about 250 ns in a NaI(Tl) crystal. Most detectors produce a prompt output signal proportional to the radiation energy deposited within a small time delay (<1 μs). Some detectors (such as thermoluminescent detectors (TLD)) store excitation energy in electron and hole-trap centers for many days, and the information is read out during processing. Usually, only energy spectra of radiation or event counts are measured. However, identification of radiation types is important in some applications, such as dosimetry in mixed neutron and gamma-ray fields. Since the relative biological effect of neutrons and gamma-rays is significantly different, they must be counted separately. One method for solving this problem is to use detectors, which are sensitive to neutrons and to gamma-rays. Another approach is to apply a pulse shape discrimination technique and use a single probe. Since detector time responses to various types of radiation are different, it is possible to acquire time information in addition to the total energy deposited of radiation in a detector to identify radiation type.

For example, the rise time of the output pulse from a proportional counter caused by a gamma-ray event is larger than that caused by a proton event due to a longer range of gamma-rays[5]. In scintillation detectors, there often exist more than two components of scintillation light with different decay time constants and with different amplitudes. In

some scintillation materials, such as anthracene, stilbene, and BC501A, this difference is sufficiently large to identify neutrons from gamma-rays over a wide energy range.

However, only a few scintillators can produce a large enough difference for practical use; therefore, the phoswich (Phosphor sandwich) detector, which consists of several different scintillation materials with different scintillation decay time constants, is often used. Radioactive particles of different types deposit most of their energy at different scintillation layers of a phoswich detector to produce light outputs with different decay times. For example, the ZnS(Ag)-plastic Phoswich detector consists of a ZnS(Ag) scintillation layer (~ 20 μm thick) and a plastic scintillation layer (a few centimeters thick). In a mixed α–γ radiation field, α particles are stopped in ZnS(Ag); therefore, they produce pulses with a longer decay time (~ μs), while γ-rays deposit most of their energy in the plastic layer to produce pulses with a shorter decay time (~ ns). These pulse can be identified by using PSD methods; thus, the alphas and gammas can be counted separately.

Another application of PSD is for suppression of electronic noise or background. Signals from electric noise generally have different time spectra than radiation events so they can be rejected by PSD. As a result, detectors with a PSD capability are useful for a very low radiation level application.

Pulse shape discrimination can also be used to improve energy resolution and to suppress the Compton background in high resolution gamma-ray detectors, such as the HPGe detector[6]. When an event occurs near the outside edge of a cylindrical HPGe detector, the signal rise time is longer than when an event occurs at the inner portion of the detector. The collection of charge at the outside layer of the detector is more often

incomplete due to a larger escape probability for electrons. Therefore, events near the outside edge broaden the full energy peak and cause a higher Compton background. With PSD, events with a long rise time can be rejected; therefore, the energy resolution of the full energy peak is improved and the continuous Compton background is suppressed.

1.2 Pulse Shape Discrimination for Scintillation Detectors

Scintillation was the first method used for radiation detection. Roentgen discovered X-rays in 1894 by observing the fluorescence of a screen with a scintillation material. When the radiation interacts with a scintillation material, called a scintillator or phosphor, its energy is transferred to excited states in that material. The excited states decay by emitting photons, whose number is in proportion to the energy deposited and is a function of band width. Scintillators for radiation detection are usually surrounded by reflecting surfaces to trap as much light as possible, and they are optically coupled to a photomultiplier tube (PMT) or a multiple channel plate (MCP) for generation of an electrical signal. The signal is amplified 10^5-10^8 times, and it remains proportional to the energy deposited. The energy spectrum of radiation can be measured by using a Multiple Channel Analyzer (MCA) after further amplification by a preamplifier and a main amplifier.

Other important information that can be obtained from an excited scintillator is the time information. The decay time of the excited states varies widely from seconds or even shorter to 10^{-9} second, depending on materials and radiation type. Processes with shorter decay times are defined as fluorescence, and those with longer decay time are

phosphorescent. Many decay components exist in an scintillator; however, only a few contribute to the signal. If ratios between these components are radiation dependent, they can be used to identify the types of incident radiation. For example, CsI(Na) has two components of exponential decay time, 0.44 μs and 4.2 μs, and the ratio of the fast to slow component is 83 when excited by alpha particles, and 14 when excited by beta-rays[3].

Inorganic scintillators are manufactured with small amounts of activator impurities to increase the fluorescence efficiency and to produce photons in the visible region which match the response function of the photomultiplier. When radiation interacts with an inorganic crystal, it transfers the lattice of the crystal into excited states[3], then the excited states de-excite by transfer of excitation energy to the activator impurities, finally the excited impurities de-excite by emitting visible light. Usually only one dominant decay component exists in the inorganic crystal; therefore, PSD performance of inorganic crystals with a few exceptions, such as CsI(Na), and CsI(Tl), is usually not as good as some organic scintillators. However, they can be applied to pulse shape discrimination by using the Phoswich approach. For example, a phoswich detector[9] consisting of thin (1-2 mm) NaI(Tl) crystal in front coupled to a thick (30 mm) CsI(Tl) can be used to count beta particles or low-energy photons in a high photon background. In this case, the high-energy photons lose most of their energy in the CsI(Tl), while the low-energy photons or beta particles deposit almost all their energy in NaI(Tl). The signal caused by high-energy events in CsI(Tl) has a decay time of 900 ns

which can be easily separated from the signal caused by the low energy event in NaI(Tl) with 250 ns decay time.

Unlike inorganic scintillators which depend on the existence of a crystalline lattice, the excitation process of an organic scintillator is through molecules in the organic scintillator[4]; hence, there exist solid, liquid and vapor states in organic scintillators. In organic scintillators, the incident radiation transfers individual molecules into discrete states from which they decay by photon emission. Compared with inorganic scintilators, organic materials have much faster response, but they generally yield less light output. Organic scintillators are available in a variety of forms. Anthracene has the highest efficiency of energy transfer of all organic scintillators, but it has no PSD capability. Stilbene is another common organic crystalline scintilator, which has high light output and excellent PSD performance; however, it is fragile and orientation dependent. Thus, it is not suitable for some applications, such as for environmental surveys and for detectors with large volume. Organic scintillators can be polymerized into plastic with very fast response times but they then have no PSD properties. Liquid scintillators are often used when larger volumes are required, or when soft beta- and alpha-rays are to be measured.

Liquid scintillators consist of three main components[4]: solvent (e. g., xylene, toluene), solute and wavelength shifter. The light output of a liquid scintillator consists of three or four main components with different time constants. The first component corresponds to energy transfer between solvent to solute, which requires about one nanosecond. The other two or three are related to the excited decay states of solute molecules and range from a few to a few hundred nanoseconds. The fast decay comes

from single molecule de-excitation, while the slow decay is from the multi-molecule de-excitation[3, 4]. The ratio of a fast to slow decay component is radiation dependent. The events caused by heavier particles, such as neutron, and alpha particles, have a smaller value of the fast component, while events from gamma-rays are larger. This property can be used to identify radiation type.

1.3. Requirement and Criteria for Pulse Shape Discrimination

Pulse shape discrimination radiation detection systems are more complicated than energy measurement systems, and one must address several issues. First (and most important) is that the detector material should have high sensitivity to the radiation types of interest. The transfer factor from the energy deposition to the amplitude of signals should be as large as possible so that the system can work well at low energy region when the signal statistics are poor. Second, in order to classify radiation types by using time information, the system should preserve time information during any signal processing including the signal shaping for noise reduction, pile up rejection, and base line restoration. Third, the noise level of this system should be as low as possible for less pulse shape fluctuation, which will cause failure of pulse classification, especially at low energies. Finally, an appropriate PSD method has to be adopted.

Three main criteria for quantitative assessment of PSD performance are as follows: figure of merit (FOM), ratio of peak-to-valley, and cross-talk. FOM is defined as quotient of separation between two peaks on the PSD spectrum divided by the summation of the two peak's full width at half maximum. The ratio of peak to valley is

defined as the ratio of the count of a broader peak to the count at the valley. Cross-talk is defined as the percentage of another radiation (say neutrons) that is mis-counted as this given type (gamma-rays)[27] when a bias is set to identify a given radiation (say gamma-rays) at a certain correct level, e.g. 95% (which means 95 % gamma-ray events are classified as gamma-rays correctly). Unlike the ratio of peak-to-valley, FOM is not affected by the ratio of radiation components in mixed fields, and it only depends on the PSD performance of a given system. Therefore, it is used in the present work for evaluation of scintillators and of methods.

1.4. Methods of PSD for Scintillation Detectors

Based on the difference of signal processing and data acquisition, pulse shape discrimination methods can be categorized into the following five main methods:

1. zero-crossing method[10-15],
2. charge comparison method[15-20],
3. Owen's method[20-24],
4. digitization method[24-30], and
5. single electron time sampling[38, 39].

1.4.1. Zero-Crossing

For a typical zero-crossing method, the decay time of the scintillation light output is first transferred to the rise time of the anode output of a photomultiplier tube by integrating the anode pulse using a circuit with a large RC time constant ($1/RC << \lambda$, see

details in Appendix A). Then the 0.1 and 0.9 (or other fraction) points of the pulse amplitude are identified by appropriate circuits as start and stop signals for a time-to-amplitude converter to obtain a rise time measurement. Each type of radiation with a different rise time can be identified based on the difference of the rise time. There are several ways to pick up timing signals, such as the zero-crossing and leading-edge techniques. In the leading-edge approach, a threshold is set up, and only the signal with an amplitude larger than this threshold produces a output signal as a start (or stop) signal. The leading-edge method is easy to implement; however, the time resolution over a large energy range obtained by using this method, is poor due to significant energy dependence of time walk of this method. To improve the time resolution of a rise time spectrum over a wider energy region, the zero crossing method (which transforms the unipolar signals to bipolar signals and picks up zero crossing points as start and stop signals) is raised. The zero crossing point is almost energy independent so that the time resolution will be greatly improved. There are many methods to obtain the zero-crossing point, such as differentiation, double delay line shaping and constant fraction discrimination. The constant fraction discrimination method is most commonly used today.

The following is a sample to illustrate how the zero-crossing points are formed. Figure 1.1 shows a functional representation of a constant fraction discrimination (CFD) trigger. The input single-polar pulse is shunted into two branches. Then one is delayed and the other is attenuated (CFD factor f) and reversed. Finally they are added together at the "sum" unit to produce the waveform as shown in Figure 1.2. Note that a linear rise

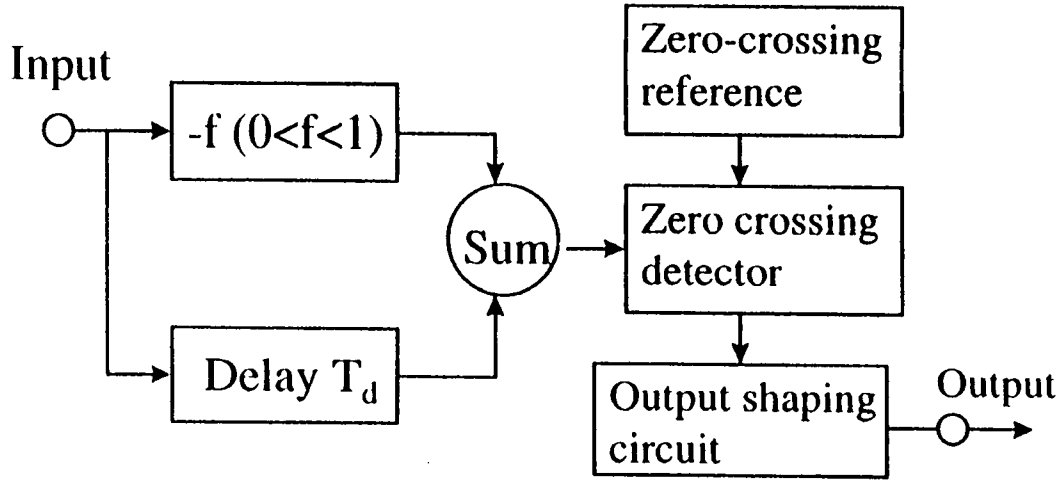


Figure 1.1 Functional Representation of a Constant Fraction Trigger.

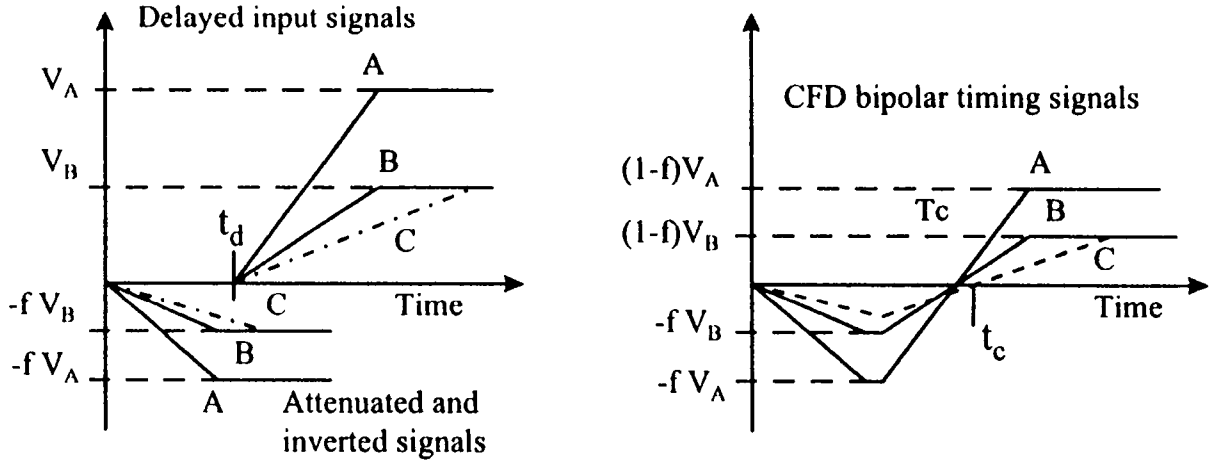


Figure 1.2 Signal Formation in a Constant Fraction Discriminator
(see text for identification of symbols)

time is assumed for illustration in this figure. In Figure 1.2, T_c is the zero-crossing point and is determined by

$$T_c = T_d + f\,T_r,$$

for the case of $T_d > (1-f)\,T_r$, where T_d and T_r are the delay time of the circuit and the rise time of a signal, and f is the attenuation factor (CFD value mentioned previously). If the start and stop signals of TAC are taken from 0.1 CFD (f = 0.1) and 0.9 CFD (f=0.9) units, respectively, the output of TAC will be proportional to the rise time of signals T_r, where

$$T_r = \frac{T_c(0.9) - T_c(0.1) + T_d(0.1) - T_d(0.9)}{0.9 - 0.1} = C1 * (T_C(0.9) - T_C(0.1)) + C2,$$

where C1 and C2 are constants.

It can be seen from Figure 1.2 that pulses A and B have the same rise time, but they have different amplitudes and produce zero-crossing points at the same location (T_c), while the zero-crossing point of pulse C has a different rise time and produces a different zero- crossing point (t_c). Therefore, the location of a zero-crossing point becomes a mark for the rise time of signals, and it is not affected by the signal amplitude. This makes it good for rise time measurements with little energy dependence.

The zero crossing method is suitable for a variety of scintillators with various ranges of decay time from a few nanoseconds to a few microseconds with only a minor adjustments (delay line and the constant fraction value). This characteristic makes it good for multiple-radiation identification. However, the pulse processing time needed is longer than the charge comparison method due to a large anode integration time constant.

1.4.2. Charge Comparison

In the charge comparison method (or charge integration), a faithful reproduction of the scintillation light pulse is produced at the anode of a photomultiplier tube by the use of a small anode time constant ($1/RC >> \lambda$, see details in Appendix A), then the fraction of the fast (or slow) component is measured and used to identify different radiation types.

Generally speaking, there are fast and slow decay components of the scintillation light in scintillation materials, and the ratio between them is dependent on the radiation type. The charge comparison method takes advantage of this feature and sets two different time integration windows, which correspond to either fast and total decay charge intervals or fast and slow intervals. Then it measures the ratio of the two integrated charges, and different radiation types are identified by this ratio difference. An example is illustrated in Figure 1.3, which shows the wave form of the signal processing in the TC-5020 analyzer (designed for neutron-photon pulse shape discrimination). The fast component is integrated in the first 25 nanoseconds, and the total charge is integrated for 120 nanoseconds. The ratio is calculated and used to identify neutrons from photons.

Since a small anode time constant is used, the pulse processing time for the charge comparison method is shorter than for the zero-crossing. Thus it is suitable for high-count rate measurements.

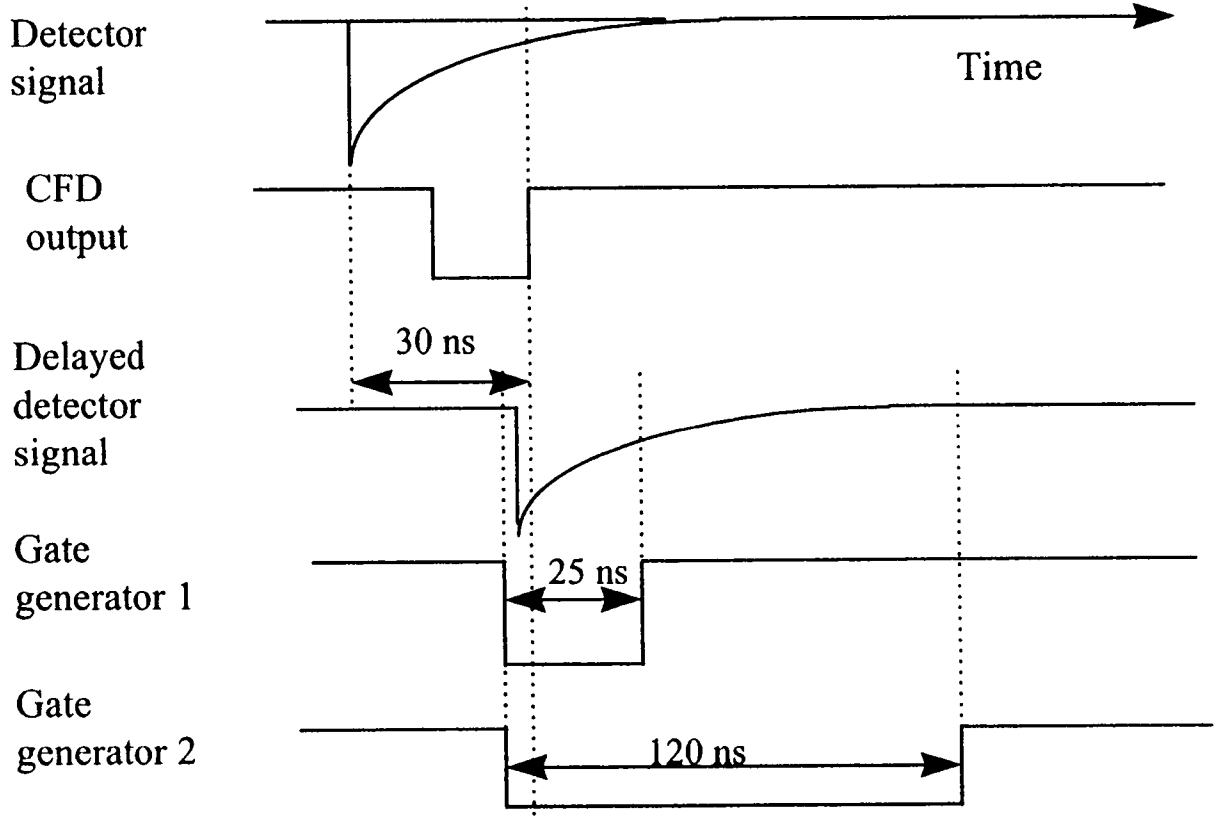


Figure 1.3 Wave Form of Signal Processing in a TC-5020 Analyzer

1.4.3. Owen's method

In Owen's method, the potential between the last dynode and the anode is carefully adjusted to a very low value (~ few volts). Thus, the space charge between these two electrodes is saturated for the scintillation caused by a radiation event with a large portion of fast components, such as gamma-rays, while not saturated for the scintillation caused by a radiation with a smaller portion of fast components, such as neutrons or protons. Due to the saturation effect, no matter how large the gamma-ray's energy, the amplitude of its anode pulse output will be unchanged and remain smaller than those caused by neutrons or protons. When a bias is set properly, gamma-rays can be separated from other radiation. The main advantage of this method is its simplicity; however, fine adjustment is required. The saturation approach makes this method count-rate sensitive, and it is not good at the high count rate condition.

1.4.4. Digitization

Some work has been done to compare the PSD performance of the methods mentioned previously in the low energy region [13, 20, 18, 27]. The results from different authors are contradictory and a common conclusion is that all analogue PSD methods have troubles at low energies. The intrinsic reason is the poor statistics of photoelectrons in the pulse produced by low energy events. Another reason is the noise influence from the photomulitiplier and electronics, which becomes significant when compared with the weak signals.

In the digitization method, the scintillation decay pulse shape is directly digitized with a digitizer or a sampling oscilloscope, then filtered and classified by using a computer software with an appropriate algorithm. With this method the PSD performance may be close to the theoretical low energy limit, but is still in the stage of laboratory research.

1.4.5. Single Electron Time Sampling

This is another type of the digitization method and is also called the delay-coincidence method[38 39]. Figure 1.4 illustrates how it works. The scintillation material studied is viewed by two photomultiplier tubes (PMTs), one of which is fully optically coupled to that scintillator and another is so weakly coupled that each event in the scintillator only produces less than one single photoelectron on the cathode of a photomultiplier tube. The signal from the strongly coupled PMT is used as the start of a time-to-amplitude converter, while the signal from the weakly coupled PMT, which functions as a sampler to draw a single electron from a large single electron population, is used as the stop signal. The time spectrum measured in such an arrangement is a single electron time distribution, which is directly related to the scintillation decay pulse shape.

In this method, two photomultiplier tubes are used and the measurement time is quite long; therefore, it is only used for the study of scintillation properties of materials.

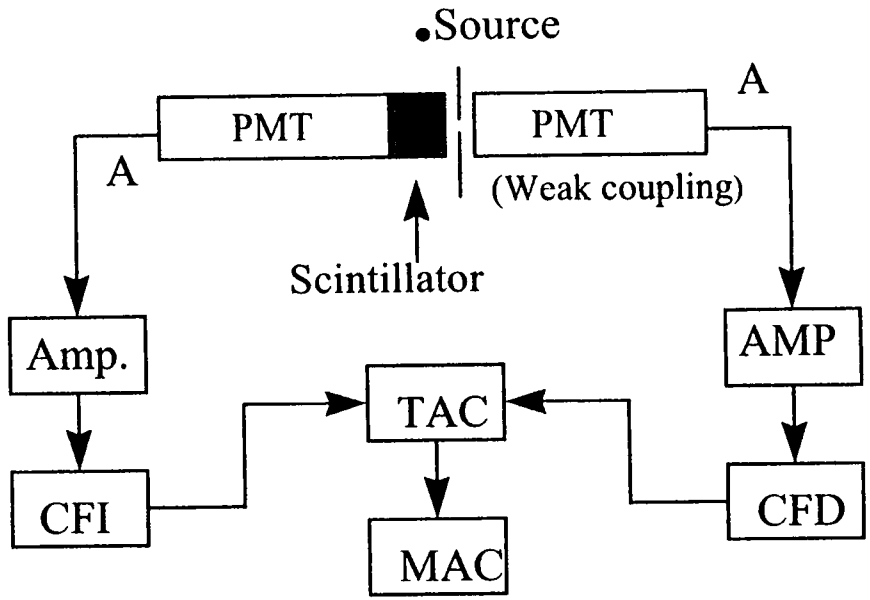


Figure 1.4 Schematic Diagram of Single Electron Time Sampling

CHAPTER TWO

TIME-OF-FLIGHT SELECTION METHOD FOR PSD CALIBRATION AT LOW NEUTRON ENERGIES

2.1 Necessity of the Method

The determination of cross-talk between neutron and photon channels at low neutron energies is a difficult task due to overlap of neutron and photon peaks in the PSD spectrum (see Appendix B, Figure B.14). Photon contributions to neutron bins can be measured at different thresholds with a pure gamma source, such as Cs^{137} and Co^{60}. However, this approach cannot be used for the calibration of the contribution of neutrons to the photon channel since there are no pure (readily available) neutron sources without associated gamma-rays. Some experiments are performed by using proton and electron beams to emulate neutrons and gamma-rays. However, there are two problems with these methods: one, it is expensive and requires accelerators and two, the interaction of mono-energetic protons with the scintillator is quite different from that of mono-energetic neutrons, which produce protons with different energies. Therefore, a PSD spectrum obtained from protons is different from one obtained from neutrons due to the PSD energy dependence, especially at low energy.

A new approach is to measure PSD spectrum with a time-of-flight selection technique which can provide a "pure" neutron source. One advantage of this method is

that no accelerator is required and is thus suitable for small institutes.

2.2 Experiment Arrangement

Figure 2.1 shows the experimental set up for the time-of-flight experiment. A ^{252}Cf neutron source is placed close to a BC-400 plastic scintillator (1 mm thick), which detects the prompt gamma-rays from the fission of ^{252}Cf and provides start signals. The calibrated neutron detector is placed at about 0.5 m from the source, which produces stop signals for the time-to-amplitude converter (TAC). The time-to-amplitude converter window is set either at the neutron peak of the time-of-flight spectrum for the "pure neutron" event measurement, or at the photon peak for the "pure photon" event measurement. The PSD signals from PSD analyzer (details in Appendix B; Figure B.1 for the zero-crossing method or Figure B.2 for the charge comparison) are recorded by a gated multichannel analyzer (MCA). All the photon events (or neutrons, depending on the window position) will be rejected by the gate circuit and only neutron events are recorded. The energy selection and the radiation type selection for the measurement are provided by SCA units inside Ortec552 (Figure B.1 in Appendix B) and the TAC (Figure 2.1), respectively.

2.3 Results and Discussions

Figure 2.2 is the time-of-flight (TOF) spectrum with and without a carbon sample between the two detectors. The carbon sample is used only for testing the experimental

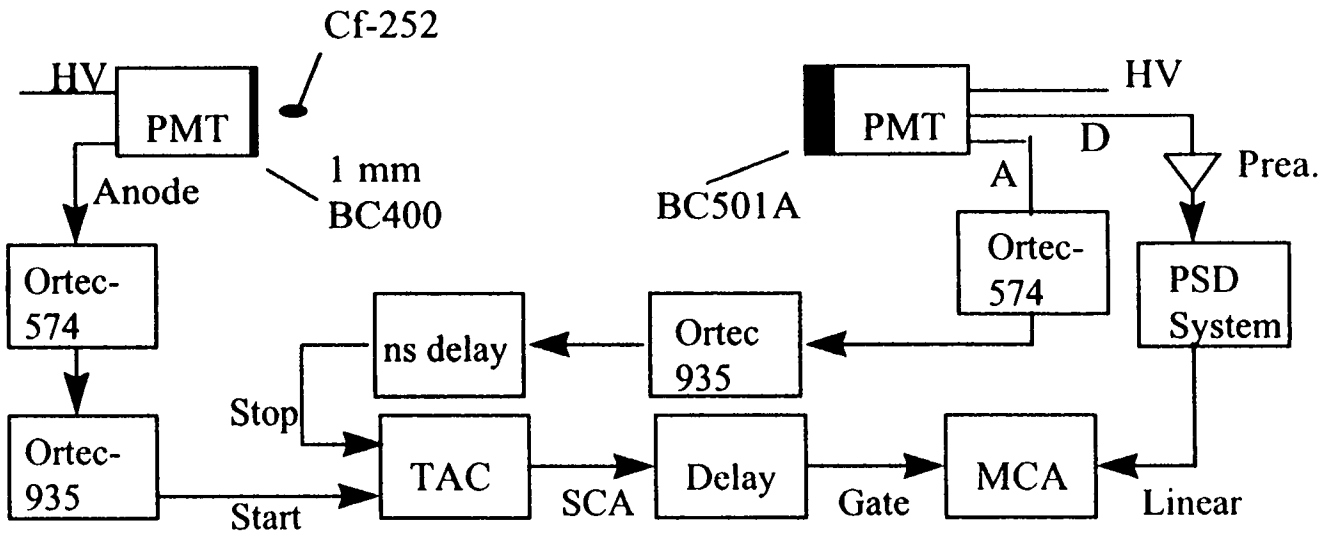


Figure 2.1 Experimental Arrangement for the Time-of-Flight Selection PSD Measurement.

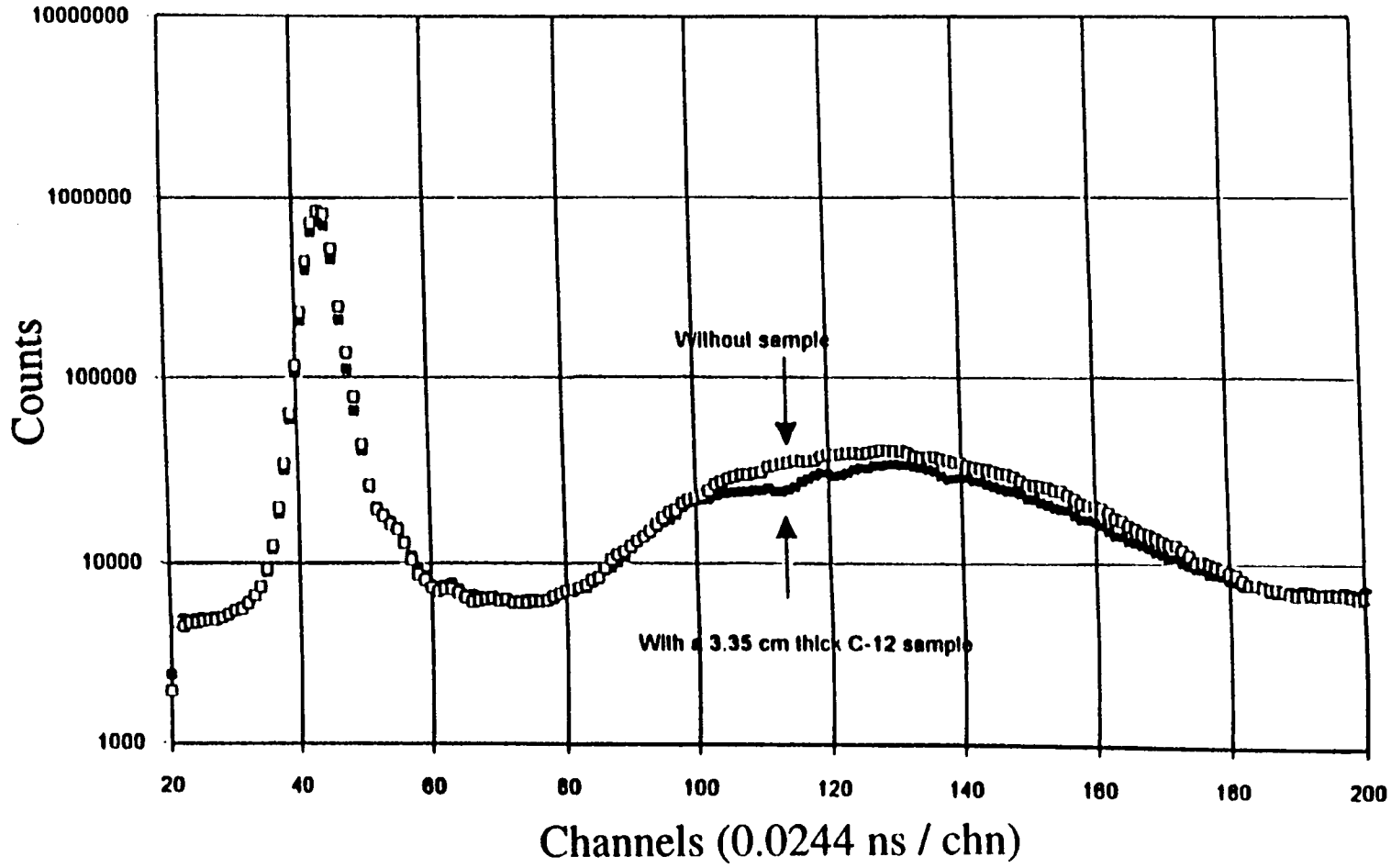


Figure 2.2 Time-of-Flight Spectra from a Cf-252 Neutron Source.

system. The rise time spectrum at 15 $\pm$ 10 keV equivalent electron energy (about 75 keV neutron energy, see Appendix C) without TOF selection is shown in Figure 2.3. It can be seen that the neutron peak is almost merged with the photon peak. Figure 2.4 shows the "pure" neutron rise time spectrum. The photon peak is reduced by a factor of 20 with TOF selection. Note a small hump in the position of the photon channel is caused by the random coincidence of the intense photon background, which can be eliminated by subtracting a background run (set window at the background part of TOF spectrum, such as at the location before the photon peak). A narrow window will help to reduce this hump; however, this will increase data acquisition time. Figure 2.5 shows the "pure photon" rise time spectrum, which is acquired with TOF selection of photons only. The neutron peak is under the fluctuation of the tail of the gamma-ray peak and cannot be seen. Figure 2.6 shows the "pure neutron" rise time spectrum after the photon contribution subtraction by using data from Figure 2.5. The uncertainty of the neutron area at this energy is less than 5 % (estimated by the formula: $\Delta N = \Delta N1 - \Delta N2$, for $N = N1 - N2$). At higher energy selection, this uncertainty is reduced to 1-3%.

The "pure" neutron and photon spectra can be used for the cross-talk calculations at low energies with fixed bias settings. They can also be used to construct low energy calibration data and training data for neural networks by the artificial synthesis method. The synthesis procedure is as follows:

1. calculate the total neutron counts under the "pure" neutron peak,
2. calculate the total photon counts under the "pure" photon peak,

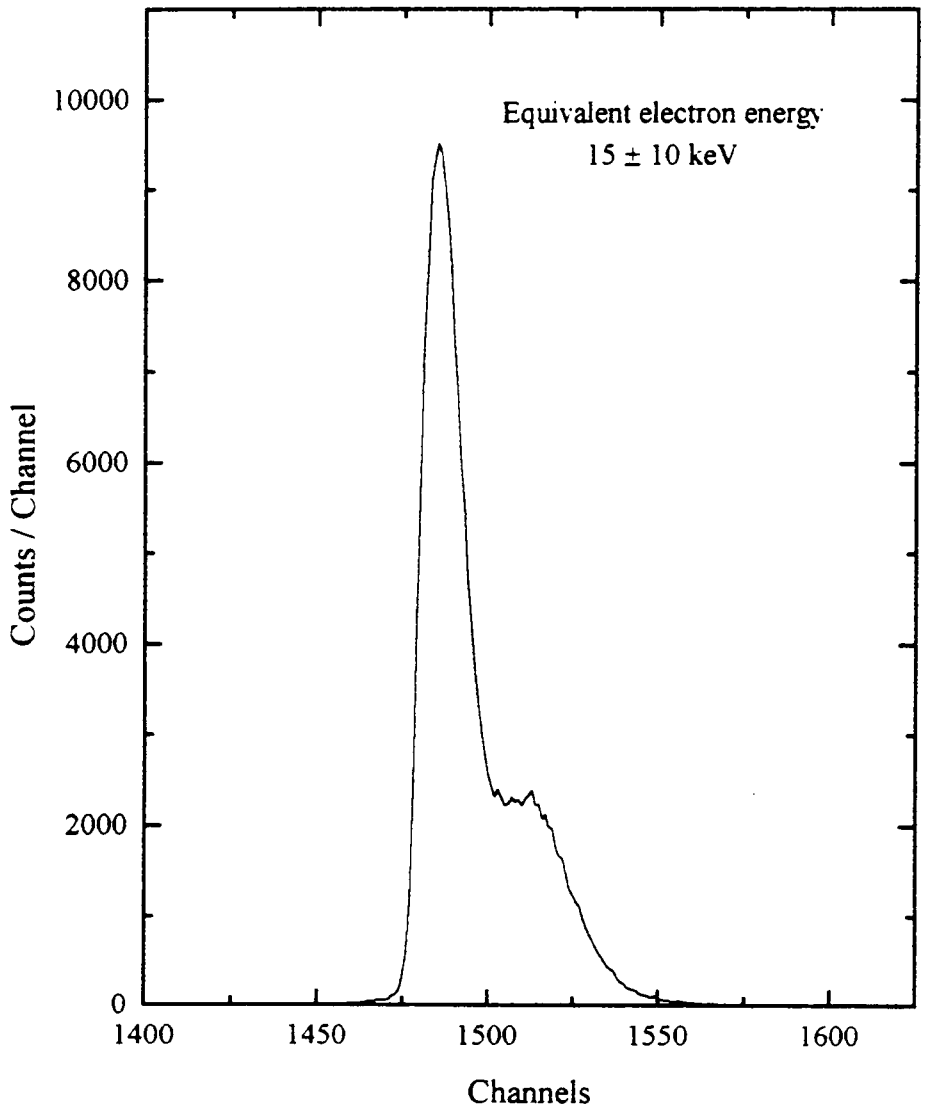


Figure 2.3 Rise Time Spectrum without Time-of-Flight Selection.

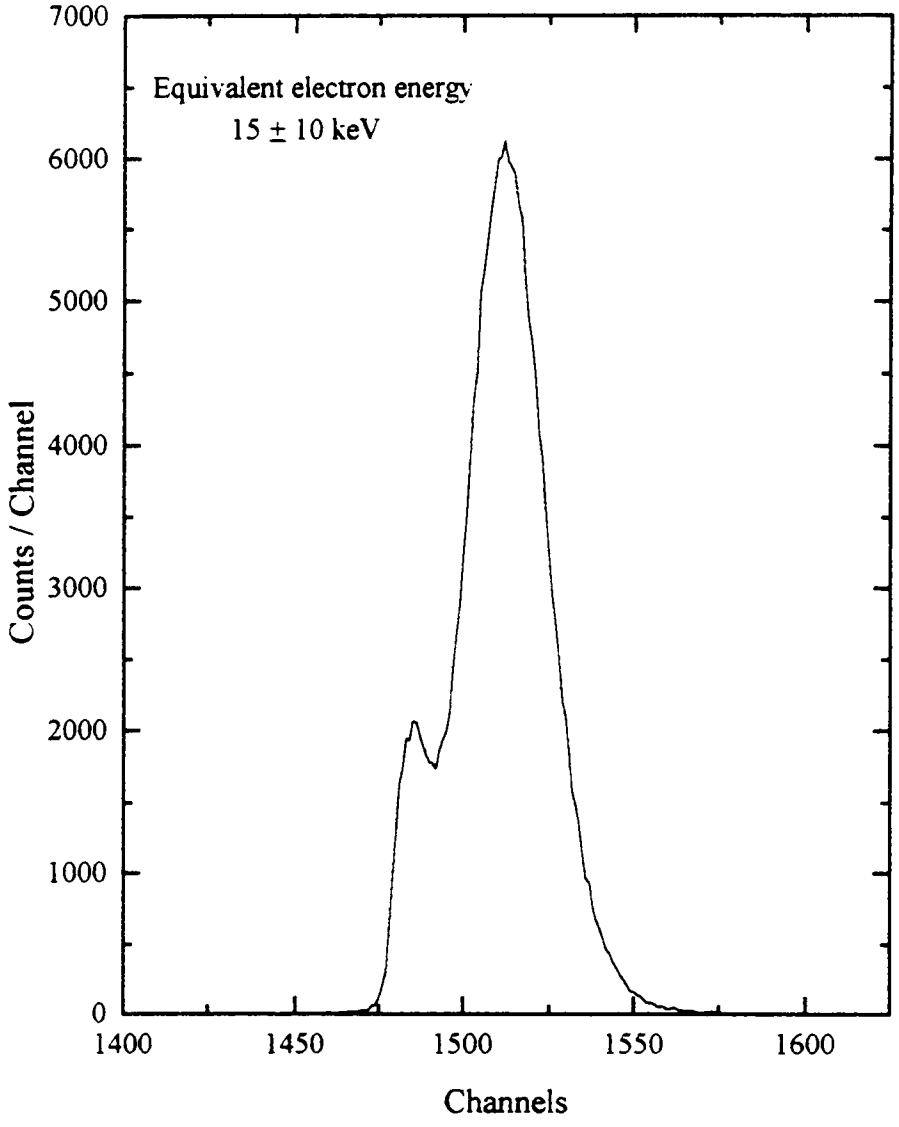


Figure 2.4 " Pure" Neutron Rise Time Spectrum with Time-of-Flight Selection on Neutrons Only.

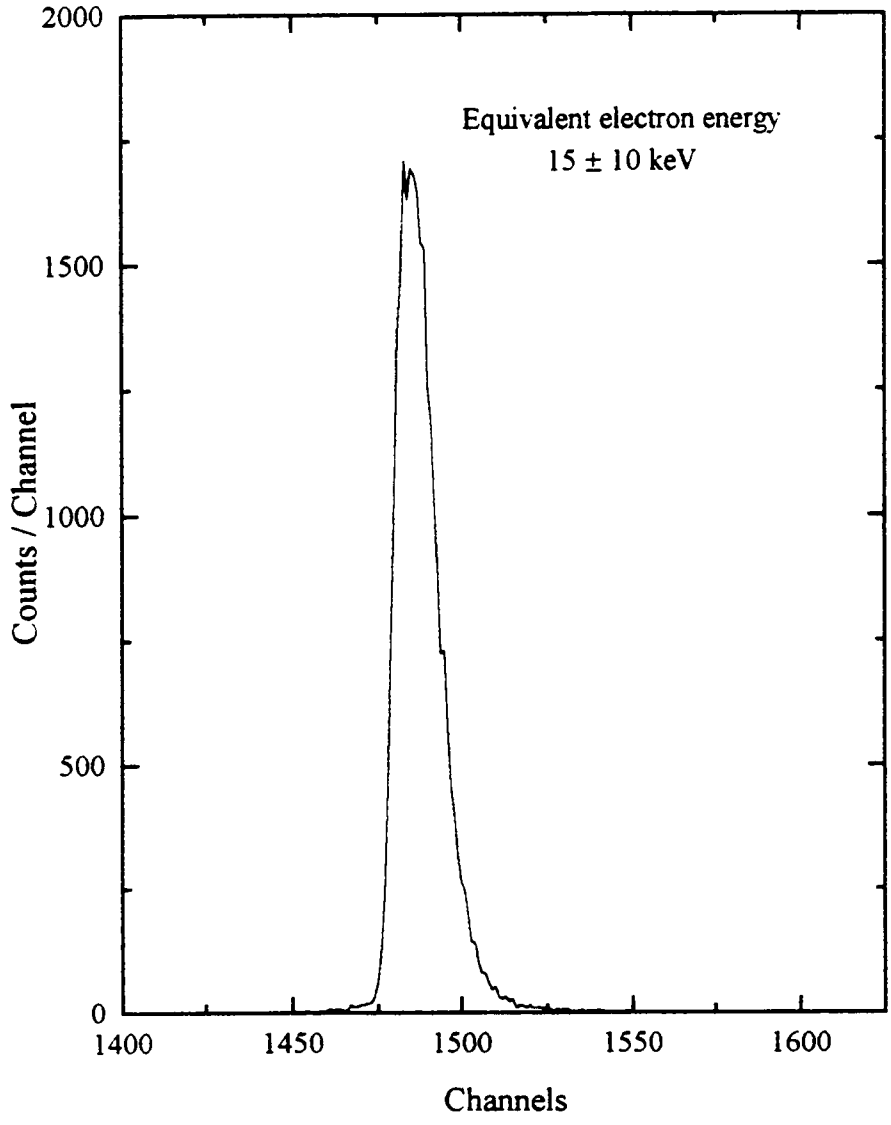


Figure 2.5 " Pure " Photon Rise Time Spectrum with Time-of-Flight Selection on Photons Only.

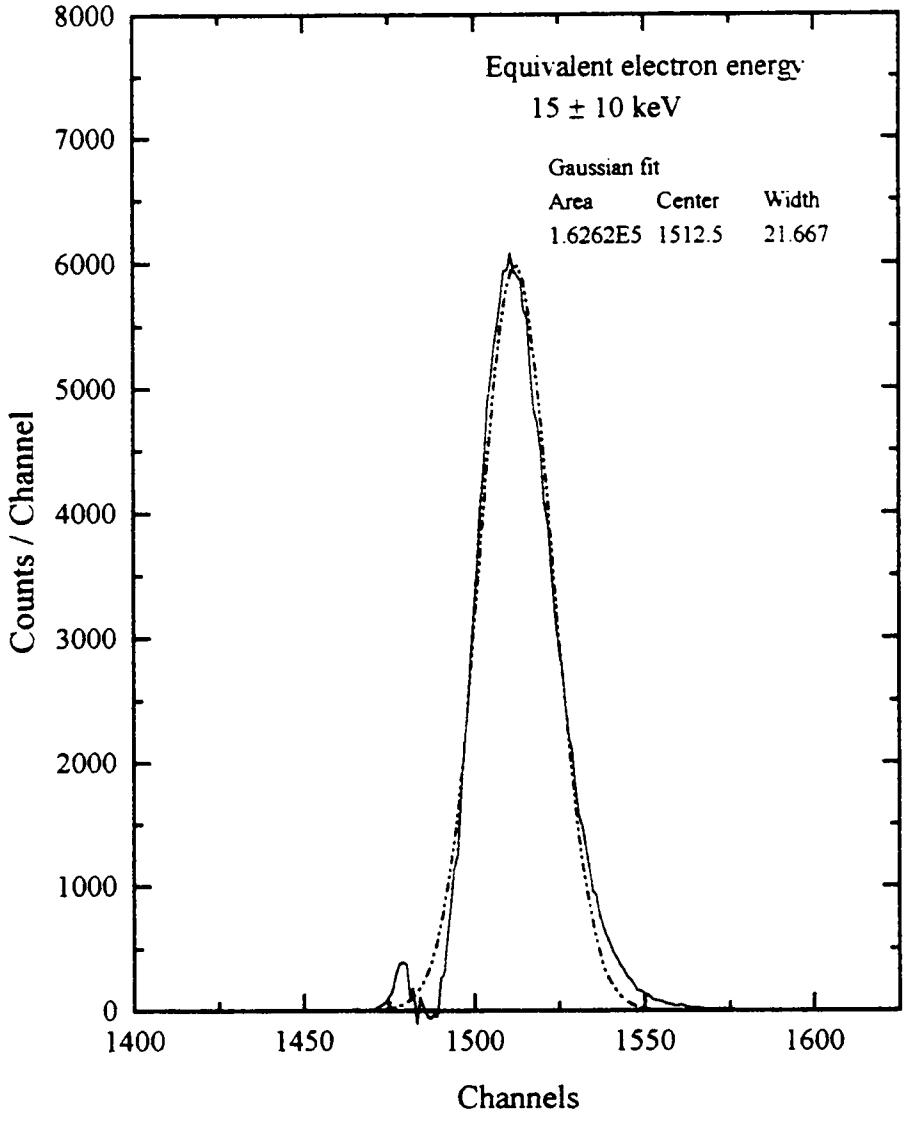


Figure 2.6 " Pure " Neutron Rise Time Spectrum after Subtraction of the Random Coincidence Photons.

3. determine the ratio (R0) of total photons to total neutrons between above two spectra,
4. the "pure" photon spectrum is multiplied by a spesified number (R1/R0), where R1 is a given number, and
5. add the "pure" neutron spectrum and the modified "pure" photon spectrum together to obtain a new spectrum, whose ratio of photons to neutrons is known as R1.

An example of constructed spectrum is shown in Figure 2.7. These constructed spectra with known ratios of photons to neutrons can be used as standard calibration data to check the classification error of a PSD detector and of different methods.

The time-of-flight selection method could be also applied to study the scintillation properties of scintillation detectors. By combining the time-of-flight selection with the single electron time sampling method, the scintillation light caused by the neutron events could be separated from that of photon events, and vice versa. In this way, the scintillation decay time and components of scintillation materials corresponding to neutrons and photons could be measured separately.

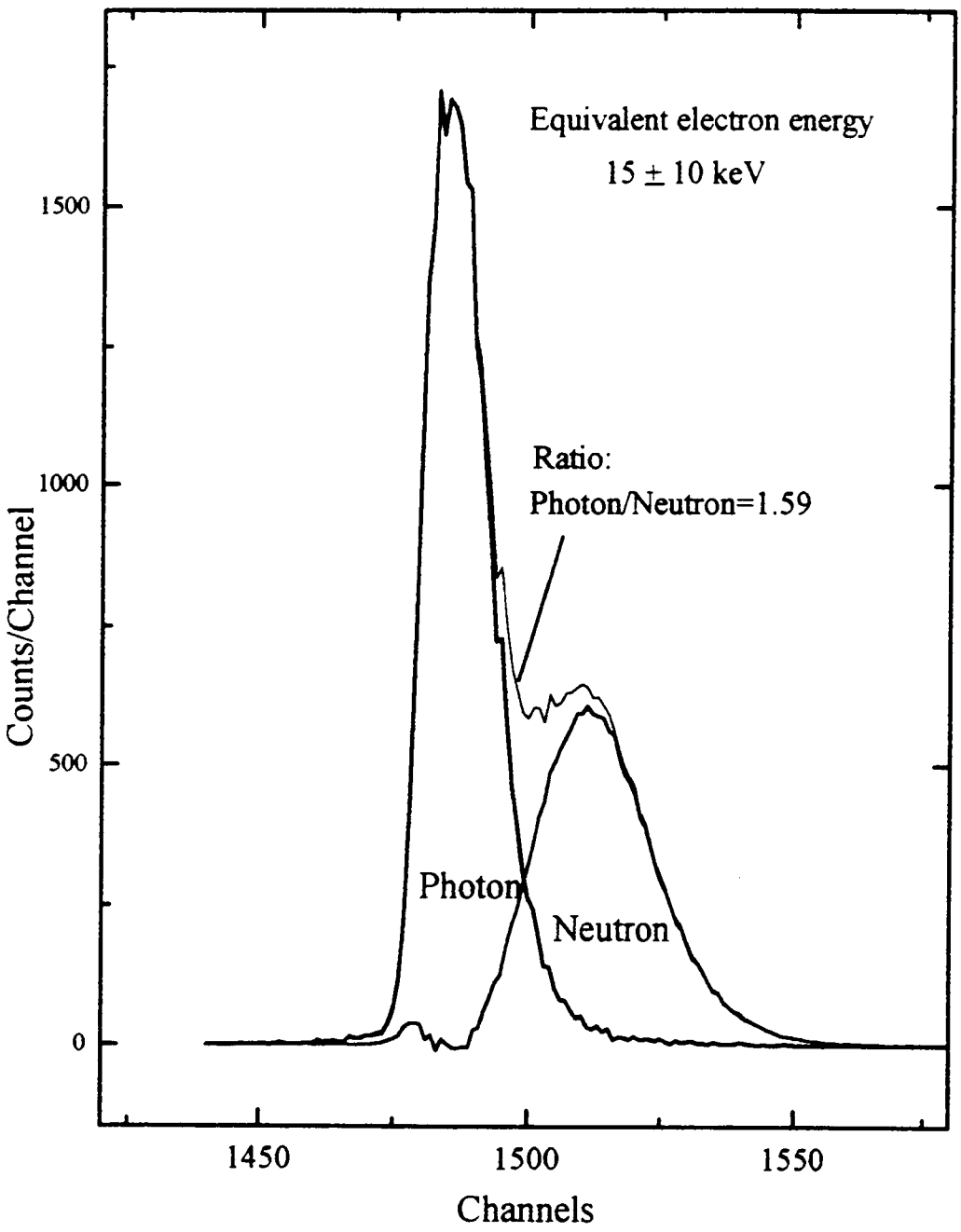


Figure 2.7 Constructed Rise Time Spectrum for PSD Calibration

CHAPTER THREE

IMPLEMENTATION OF DYNAMIC BIAS BY USING NEURAL NETWORK CLASSIFIERS

3.1 Necessity of Dynamic Bias for PSD

In the conventional single parameter neutron-photon pulse shape discrimination (PSD) technique, a fixed threshold on PSD (either the rise time or the fast component) spectrum is used to discriminate neutrons from photons. This approach neglects the energy dependence of the PSD bias and is inadequate over a large energy range. Figure 3.1 shows the rise time spectra measured using a 1.5"BC501A-XP2020 PMT detector for various equivalent electron energies with a 1.2 x 10^6/second Pu-Be neutron source at about 0.5 m distance. As can be seen from these spectra, the best bias settings for different energies depend on the energy, and the fixed bias will cause a large classification error over a large energy range. The classification error over a large energy range can be reduced by implementation of the dynamic bias technique which sets PSD bias at different locations depending on the event energy. A two parameter data acquisition system can be used to simultaneously record the energy and the decay time information and to implement dynamic bias (Figure 3.2). In this work, new methods based on a single data acquisition system are developed to implement dynamic bias

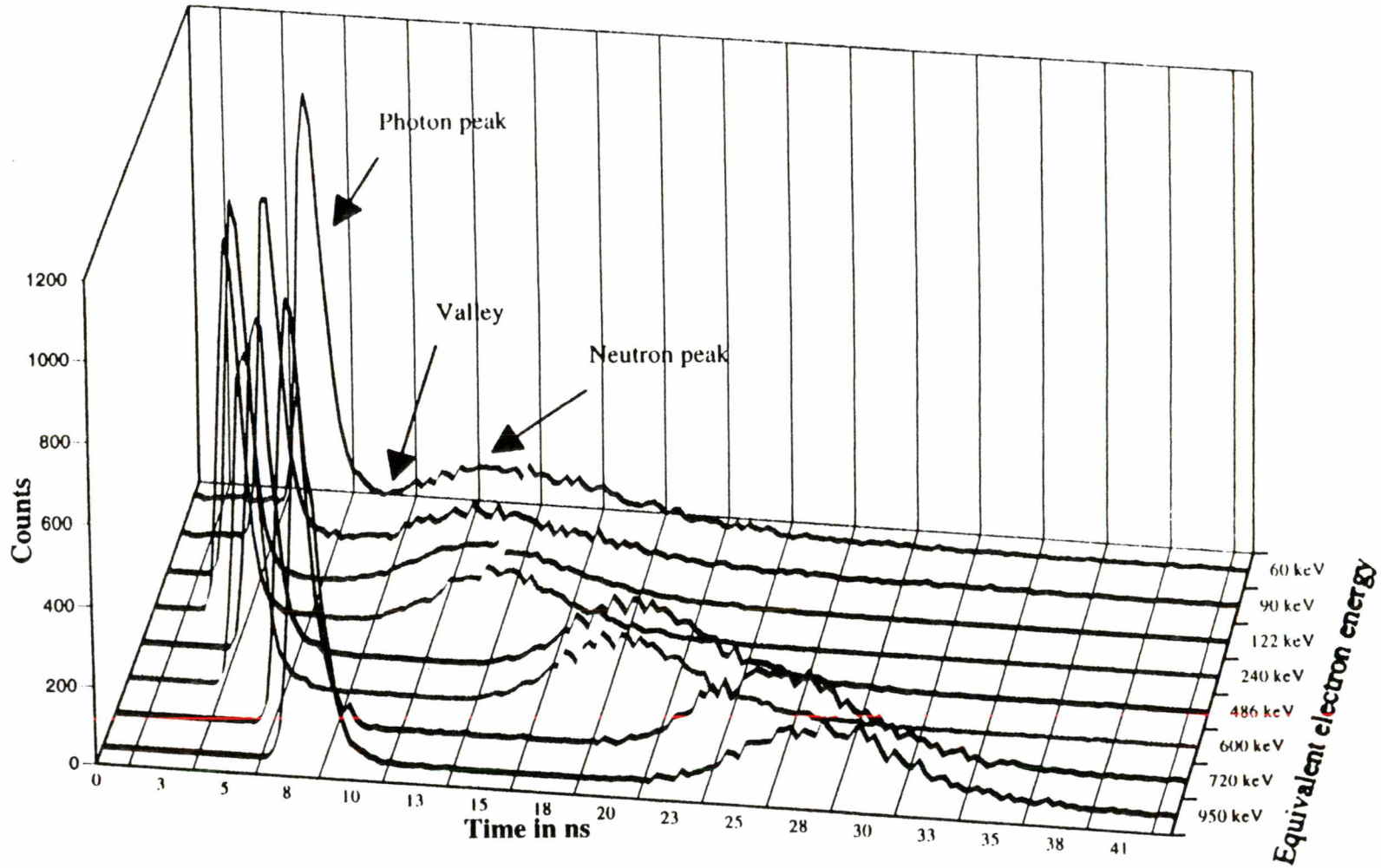
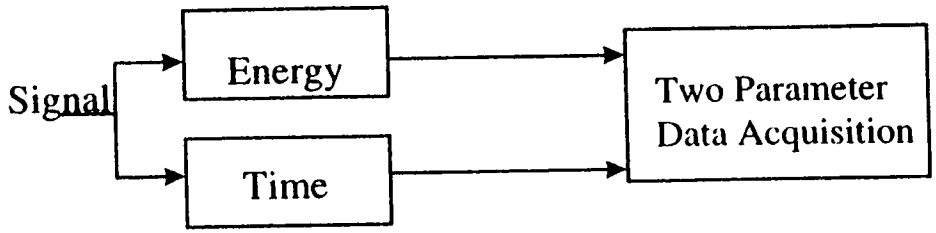
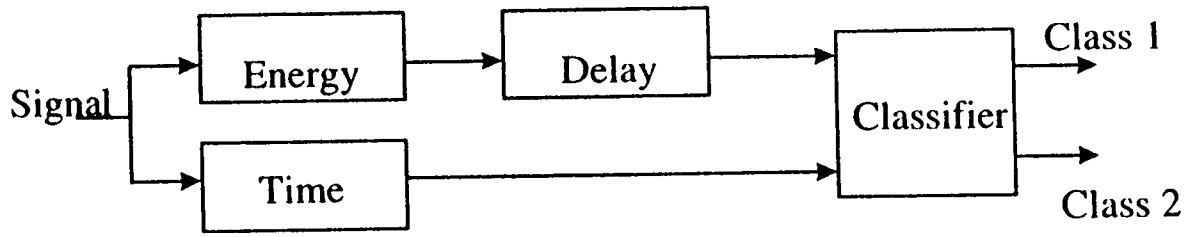


Figure 3.1 PSD Time Spectra for Various Bias Settings (1.5"BC501A/XP2020).



(a) With a two-parameter data acquisition system



(b) With a two-class classifier (no requirement for a two-parameter data acquiaition system)

Figure 3.2 Comparison of the Block Diagrams for Two Different Methods

by using neural networks and a Bayes classifier. Three two-class classifiers are established and evaluated for a range of 15 keV - 1 MeV equivalent electron energies (75 keV - 2.8 MeV neutron energy). The neural network training data below 150 keV equivalent electron energy are obtained from the data of time-of-flight selection measurements, and the others are from direct PSD measurements. After training with training data, the classifiers are tested with test data. All of them perform better than the static bias method. The perceptron classifier is the simplest one and can be easily incorporated into an IC chip, but the Bayes classifier has better accommodation to changes in neutron source condition.

3.2 Description of Methods

Figure 3.3 shows the principle of a two-class classifier[40]. The decision boundary separates two classes, which is defined by equation:

$$D = 0,$$

where D is the decision function in the pattern recognition theory[40]. In the two dimensional problem, the decision boundary is a straight or curved line.

A classifier calculates the decision function for each individual pattern {X} and determines its category. Three classifiers were developed in this work and are described in the following sections.

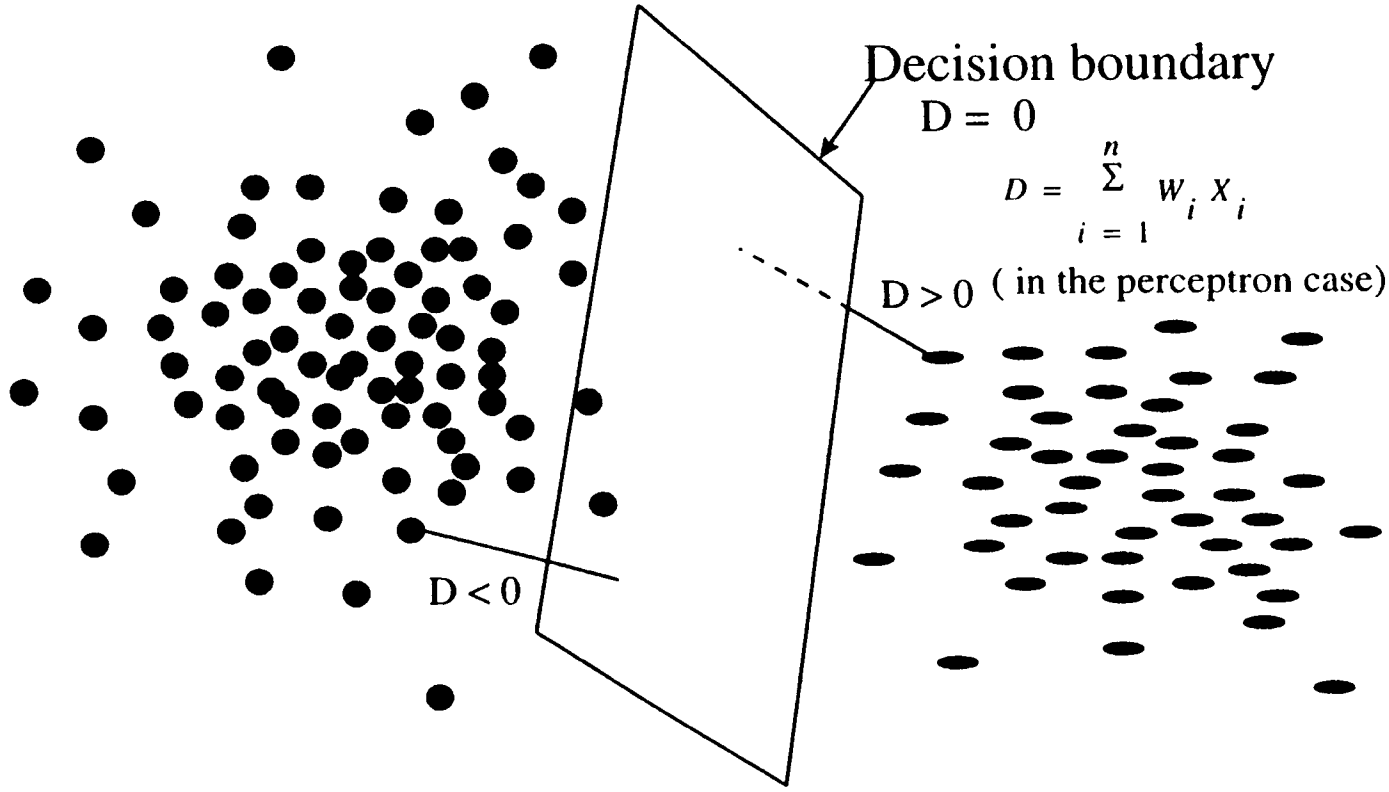


Figure 3.3 Principle of a Two-Class Classifier

3.2.1 Perceptron Classifier[40,42]

In the linear perceptron classifier, the decision function is given by

$$D = \sum_{i=1}^{3} W_i X_i ,$$

where W is the weight vector and {X} is the pattern vector consisting of the energy (x_1), the time (x_2) and an augment (x_3 = 1). The augment is necessary for neural works; otherwise, the decision plane will always pass through the zero point and fail to function.

The structure of this linear perceptron classifier is quite simple since there are only three nodes in the input layer and two nodes in the output layer (Figure 3.4). The linear perceptron is the simplest classifier in this case but it can only recognize a straight boundary line. Figure 3.4 shows a nonlinear perceptron with a x^2 term in the input layer, which is capable of identifying a slightly curved boundary line and is tested in the present work.

The training process is an iteration process to correct the weight vector of a perceptron. The "Reward-Punishment" training algorithm is used for this study. If a training datum X belongs to class 2 (the right side of the boundary plane in Figure 3.3), its decision function D must be larger than 0 for a correct classification. However, if at the kth iteration step of the training process, the decision function D is less than 0, that is

$$\sum_{i=1}^{n} W_i(k) X_i < 0,$$

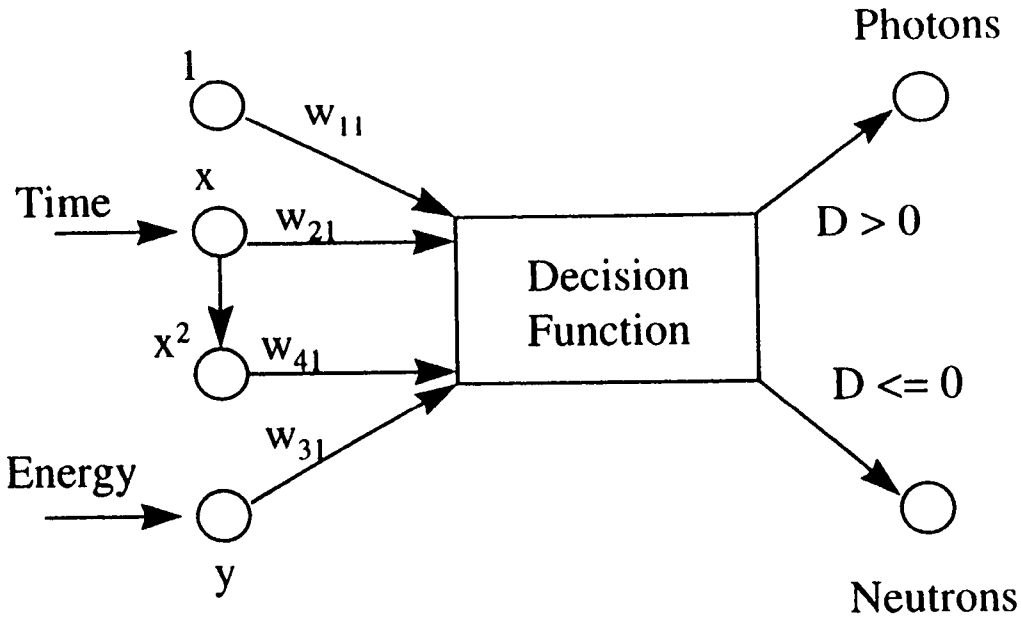


Figure 3.4 Two Dimensional Perceptron Network Classifier

an incorrect classification occurs. The weights are corrected by

$$W_i(k+1) = W_i(k) + C * X_i,$$

where C is a weight correction increment, n is the dimension of that pattern space (three for linear perceptron, four for nonlinear perceptron with a x^2 term).

If X belongs to class one (the left side of the boundary plane in Figure 3.3), its decision function D should be less than 0 for a correct classification. However, if at the kth iteration step of a training process, the decision function D is larger than 0, that is

$$\sum_{i=1}^{n} W_i(k) X_i > 0 ,$$

an incorrect classification occurs and the weight W(k) is replaced by

$$W_i(k+1) = W_i(k) - C * X_i .$$

The weight correcting algorithm changes the weight vector W only when the training pattern is incorrectly classified by the weight vector. As a reward, no weight correction occurs for the correct classification during the training process. Convergence occurs when a weight vector classifies all training patterns correctly, then the training is completed and the classifier is ready for use.

Three variations of the perceptron algorithm are formulated depending on how one selects the correction factor C. They are the fixed-increment, the absolute-correction, and the fractional-correction. In the fixed-increment case, C is a constant greater than zero. In the absolute-correction, C is a smallest integer but larger than

$$\frac{|W^{t}(k)X|}{X^{t}X}.$$

In the fractional-correction,

$$C = \lambda \frac{|W^{t}(k)X|}{X^{t}X},$$

where λ is a learn factor ($0 < \lambda < 2$).

The weight can be set to either zero, or randomized at the beginning of the training. In this work, the zero initialization was used since it had a faster convergence rate.

3.2.2 Back-Propagation Neural Network Classifier[41,42]

The structure of a backpropagation neural network classifier (BPNNC) used in the present work is shown in Figure 3.5. There are three layers: the input (3 neurods), the hidden (5 neurods), and the output layers (2 neurods).

Figure 3.6 shows the detailed structure of the hidden and output neurodes. The sigmoid function is used as a transfer function to map the linear input space to nonlinear space and the parameter α is taken to be one.

The decision function is

$$D = \Phi_{q2} - \Phi_{q1},$$

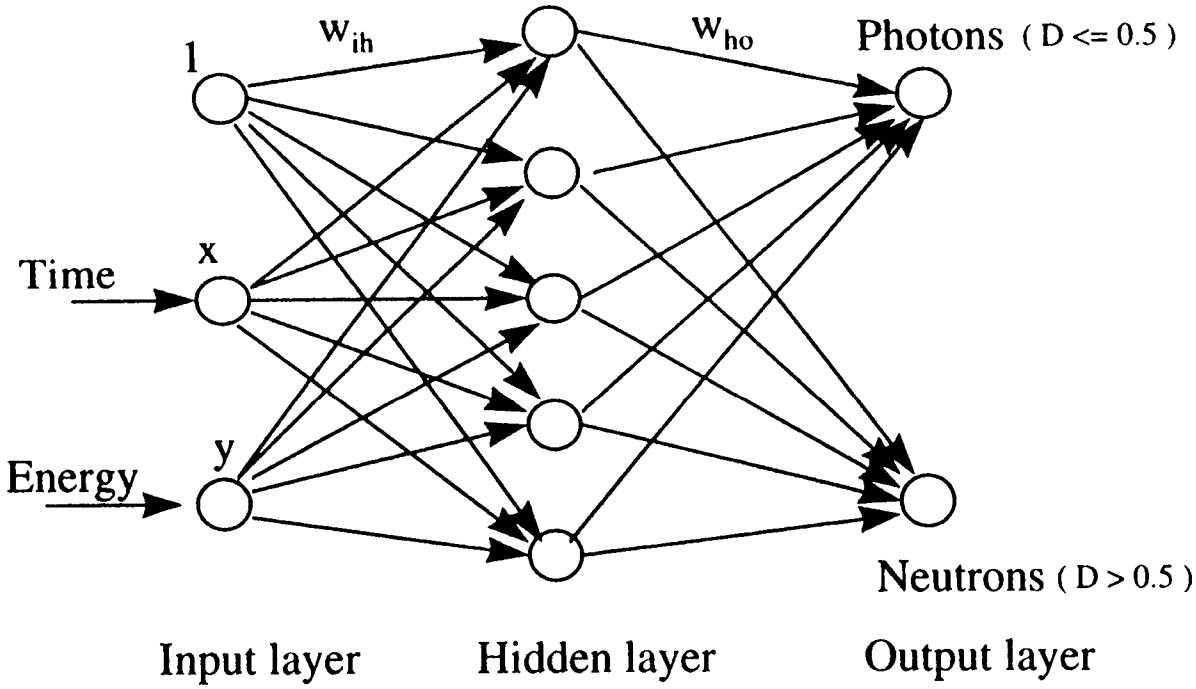


Figure 3.5 Two Dimensional Backpropagation Neural Network Classifier

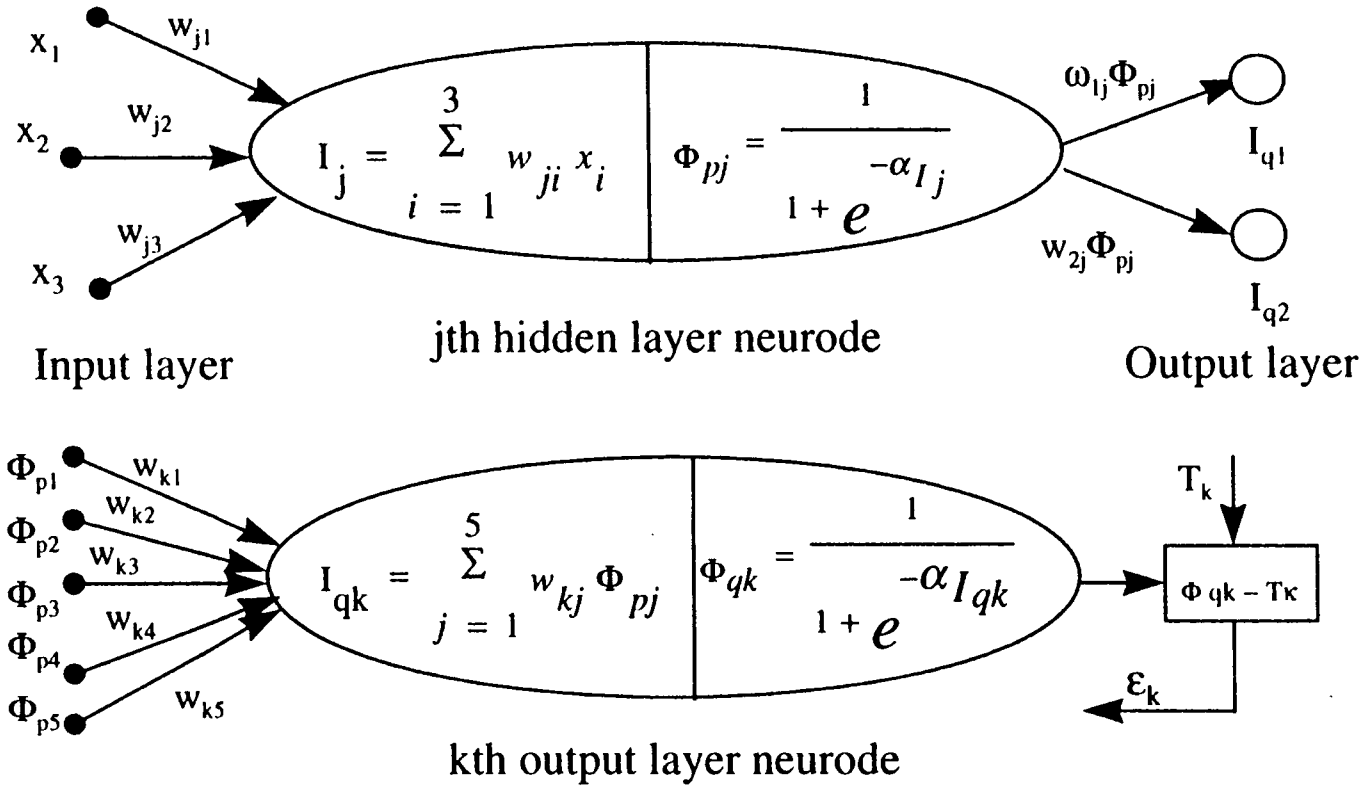


Figure 3. 6 Diagram of Neurodes in a BPNN Classifier

where Φ_{q2} and Φ_{q1} are the two sigmoid function values at the BPNNC output layer. If D is larger than 0, the pattern belongs to a neutron event; otherwise, to a photon event.

The BPNNC is trained with the supervised learning method and back-propagation algorithm. In the training, the output of the neural network classifier is compared with the corresponding target function to produce an error, then the "delta rule" (a least square minimization of error procedure) is used to reduce error down below a given tolerance. The principle is briefly described by

$$\varepsilon_\kappa = T_k - \Phi_{qk} ,$$

$$\Delta W_{kj} = \lambda \frac{\delta \varepsilon_k^2}{\delta W_{kj}} , \text{and}$$

$$\Delta W_{ji} = \lambda \frac{\delta \sum_{k=1}^{2} \varepsilon_k^2}{\delta W_{ji}} ,$$

where i, j and k are the index of the input, hidden, and output neurodes, T_k and Φ_{qk} are the target function and the output for the kth neurode in the output layer (Figure 3.6), ε_κ is the error function of kth output neurode, and λ is the learning rate (one in this work). The weight correction formula can be deduced further and expressed by

$$\Delta W_{kj} = \lambda * \delta_k * \Phi_{qj} ,$$

$$\Delta W_{ij} = \lambda * \delta_j * x_i ,$$

where

$$\delta_\kappa = \Phi_{qk}(1 - \Phi_{qk})(T_k - \Phi_{qk})$$

$$\delta_j = \Phi_{pj}(1 - \Phi_{pj}) \sum_{j=1}^{2} \delta_k W_{kj}$$

To accelerate the convergence and to avoid the local minimum, the momentum acceleration technique is implemented in the code. When the new total error value repeats an old value twice, a momentum term is added to the weight correction formula as follows

$$\Delta W_{kj}(K+1) = \lambda * \delta_\kappa * \Phi_{qj} + acc * \Delta W_{kj}(K)$$

for output layers, and

$$\Delta W_{ij}(K+1) = \lambda * \delta_j * x_i + acc * \Delta W_{ji}(K)$$

for hidden layers, where K is the iteration number, and acc is the acceleration factor(<2.5).

The training of BPNNC is a weight assignment process, and the weight vector is randomized at the beginning. Then, the output values of output layer neurodes for each input training data are calculated and compared with the corresponding target values to produce an total error. Based on minimization of the total error value, the weight vector is modified, and a new weight vector is produced. When the next training datum is tested, this new weight vector is used for the total error calculation, and is renewed again. This procedure is continued until the root mean square of total error is below a given tolerance. The root mean square of total error is given by

$$rms = \sqrt{\frac{\sum_{l=1}^{p}\sum_{k=1}^{2}\varepsilon_{lk}^{2}}{P * k_0}} ,$$

where p is the total number of the training data, k_0 is the number of output neurodes (two in this work)

3.2.3 Bayes Classifier[40,43,44]

The shape of the PSD spectra, such as the valley location, is affected by the ratio of photons to neutrons in the neutron source. Therefore, the discrimination bias should be slightly changed for different types of neutron sources in order to get the best performance. The present work shows that the neural network classifiers are quite robust to this change. For example, a factor of three increase of this ratio only causes 3 % error at 120 keV equivalent electron energy. However, when this ratio is changed by more than a factor of ten, such as in the case from the unmoderated to the moderated neutron source, the error reaches 15-20 %. Therefore, it is better to use neural network classifiers under conditions similar to their training situation or to train the networks using a various ratios of photons to neutrons.

The Bayes statistical classifier is also evaluated. In the Bayes classifier, the decision function is defined by

$$D = L_n - L_p ,$$

$$L_n = \int_{-\infty}^{x} P(x|W_n)P(W_n)dx \text{ , and}$$

$$L_p = \int_{x}^{\infty} P(x|W_p)P(W_p)dx \text{ ,}$$

where $P(w_n)$ and $P(w_p)$ are the priori probabilities of neutron and photon classes, and $P(x|w_n)$ and $P(x|w_p)$ are the likelihood functions for neutrons and photons. If D is less than 0, the event belongs to the photon; otherwise, it is a neutron.

The Gaussian with exponential tail is used as the likelihood function for the Bayes classifier as follows,

$$P(x|W_i) = \frac{e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}}{\sqrt{2\pi}\,\sigma_i}, \qquad \text{for } \frac{x-\mu_i}{\sigma_i} <= 1.5,$$

$$P(x|W_i) = 0.1295e^{-\frac{x-\mu_i}{\sigma_i}}, \quad \text{for } \frac{x-\mu_i}{\sigma_i} > 1.5,$$

where μ_i and σ_i^2 are the mean value and the variance of the class i, which are energy dependent and are obtained from calibration data.

There are three steps in a Bayes classifier, first the priori probability function is determined by a trained perceptron classifier, then the decision function is calculated by using Bayes formula, and finally classification is made. Since the priori probability is

taken into consideration, it should perform better than the BPNNC and the perceptron classifier.

3.3 Implementation, Results and Discussions

Computer codes in FORTRAN are written to implement the peceptron, Back-propagation and Bayes classifiers. The names of perceptron classifier codes, input and output data file examples are listed below (see details in Appendix D):

Percepn1.for	the code for the linear perceptron classifier.
dat1.dat	the training data example.
out.dat	the output data file after training.
weit.dat	the file stores the weight produced in training.
t8.dat	the test data example.
out1.dat	the output file for testing data.

The BPNNC source code and example files are listed as follows (see details in Appendix E.):

nnwm1.for	the source file for BPNNC with momentum acceleration.
dat4.dat	the training data file example.
nnt8.dat	the test data file example.
nnwm1.wwt	the weight file example.
nnwm1.out	the output data example.

The training time required in this procedure depends on the structure of networks and on the training data. For the simplest linear perceptron classifier, a 1/2048 (4 MCA channels in 8192 MCA gain) resolution can be obtained in a few minutes, while the BPNNC requires more than 10 hours. It is also dependent on the training data. It is very important to perform preprocess of the training data for a faster convergence. In the present work, each set of training data consists of two input values, the rise time and the corresponding energy, and two expected output values (target values). To have a better convergence rate, the energy and time signals are scaled within 0-10 for the perceptron training and 0-1 for the BPNNC training. The target function values in BPNNC are (0.4, 0.6) for the photon and (0.6, 0.4) for the neutron. A BPNNC with five hidden layer neurodes has good classification capabilities, and the required time for training is normal; thus, the structure is recommaned for the application

For BPNNC, training is stopped if a given tolerance is reached, and the tests show that the RMS tolerance should be less than 0.0001 to obtain a good neutron-photon classification. For the perceptron classifier, training is continued until all patterns are correctly classified.

The number of training data available is more than 10^6, and it takes quite long time (days for BPNNC) if they are directly used for the training. To speed up the training, an alternative approach is used. This approach is based on the fact that the most important training data lie close to the boundary line. If these data are correctly classified, others will also be correctly classified.. The preprocessing procedure is described as follows: The valleys of PSD calibration spectra (Figures 3.7 and 3.8) at

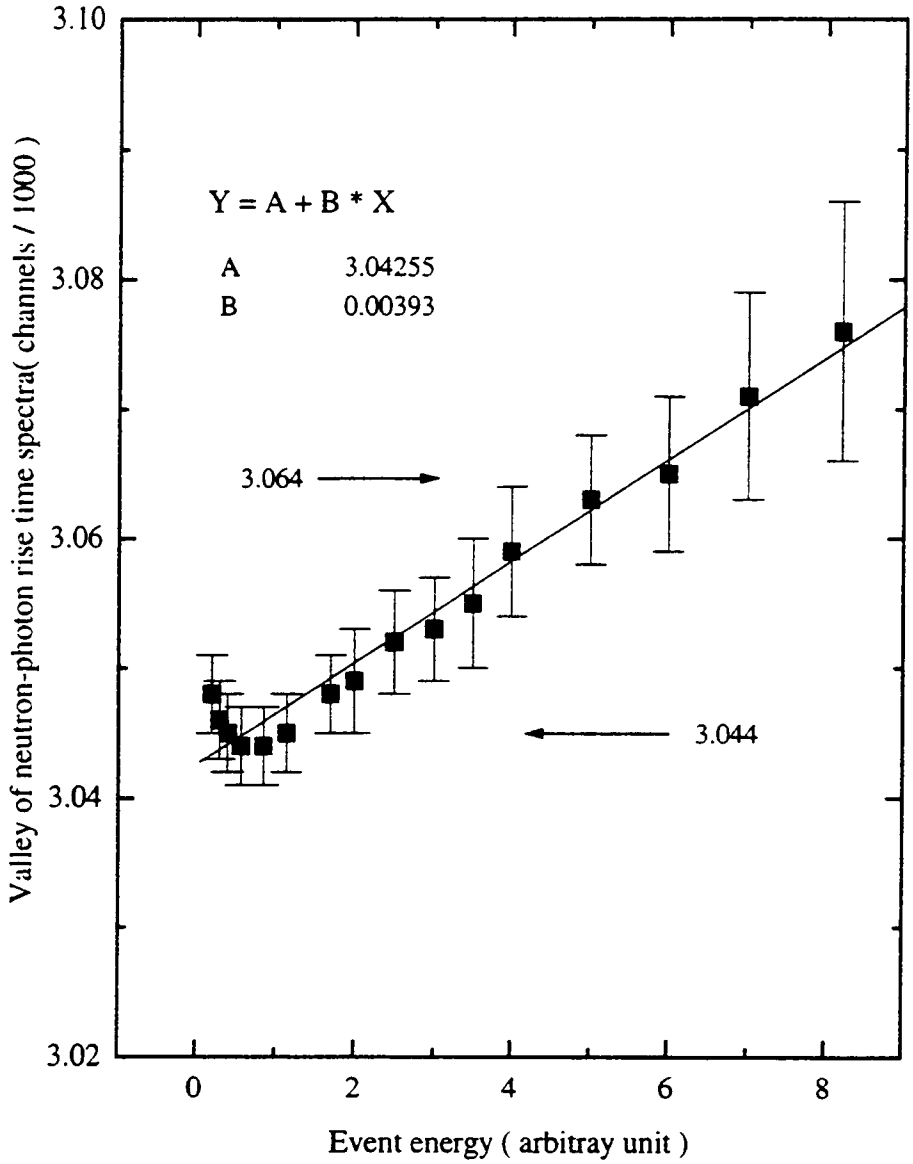


Figure 3.7 PSD Thresholds of Rise Time Spectra at Various Energies and the Linear Fitting Line.

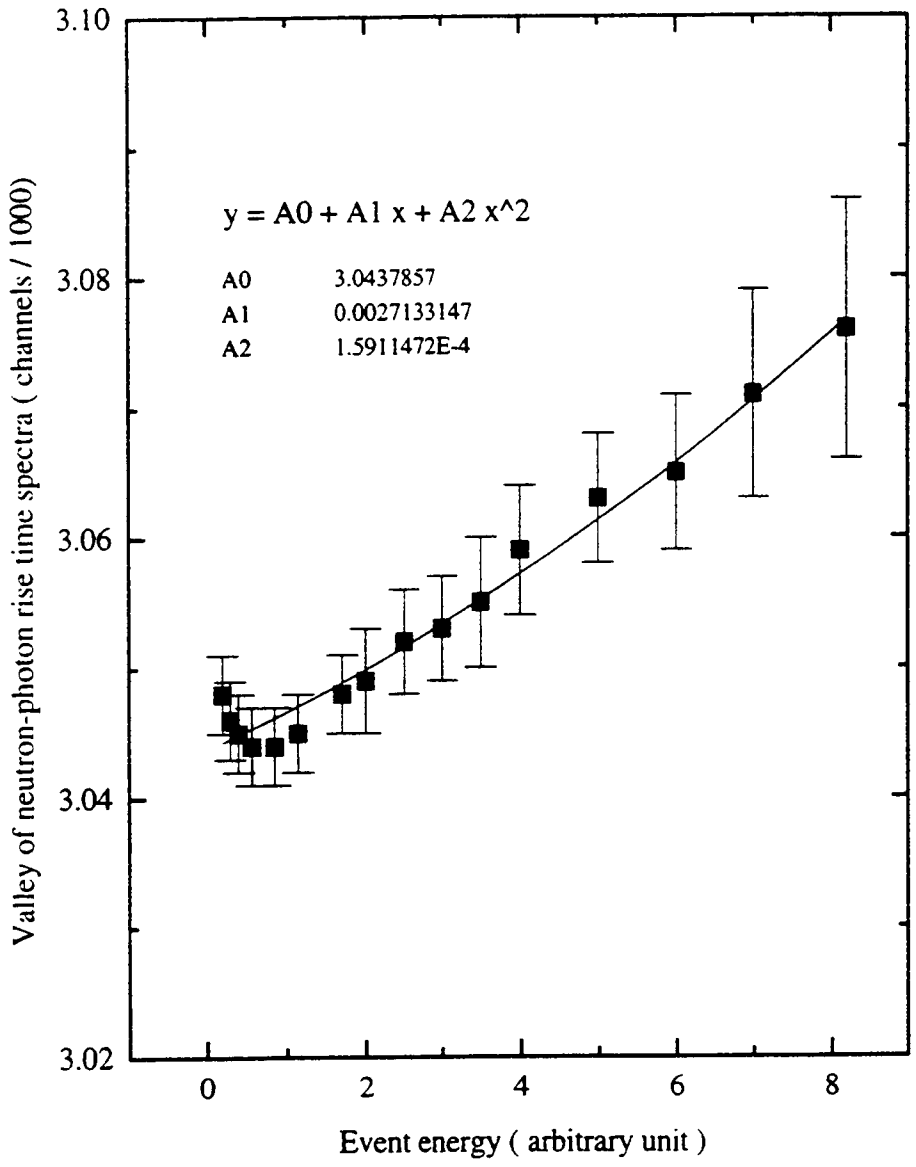


Figure 3.8 PSD Thresholds of Rise Time Spectra at Various Energies and the Nonelinear Fitting Curve.

various energies are fitted with a straight line (or add a x^2 term), then this line is used to construct a set of training data artificially. First, set the target functions (0.4, 0.6) for all data (photon events) on the left side of this line and set target functions (0.6, 0.4) for all data (neutron events) on the right side of this line. For example, at x=1.2 (related to energy), the fitting coefficient of the line is used to calculate the corresponding y (related to time) value on the boundary line (to get y = 3.044), then the point (1.2, 4.043) is set to be a photon pattern and the point (1.2, 4.045) set to be a neutron pattern. Since the gain of MCA is 10 V per 8192 channels, this arrangement ensures the classification resolution is

$$R = \frac{4.045 - 4.043}{8.192} = \frac{2}{8192}.$$

Using this preprocess, only 34 data points along the lines ($0.3 < x < 8.2$) are required for training a perceptron (or a BPNNC) to recognize the boundary line, and training time is greatly reduced and classification results are the same as that by using measured data directly.

Acquisition of the training data is an important procedure in training a neural network. The energy range of the present work is from 15 keV to 1 Mev equivalent electron energies. At higher energies, where the figure of merit of PSD is larger than one (see Appendix B, Figure B.13 for example), the training data are obtained directly from the rise time spectrum measurements for the selected energy bins (25 keV width). However, since the PSD performance deteriorates as energy decreases, it is very difficult

to separate photons from neutrons at low energies (<150 keV equivalent electron energy) without a large uncertainty (see Figure 2.3 for example). Therefore, the time-of-flight selection method is used to acquire the "pure" neutron and the "pure" photon rise time spectra below 150 keV equivalent electron energy. Then the random coincidence correction and the artificial syntheses are performed to produce training spectra with a known ratio of photons to neutrons (see Figure 2.7 for example). The width of selected energy bins at low energis is 10 keV.

The convergence rates of different weight correction methods in the perceptron training are listed in Table 3.1. It can be seen that the fractional-correction method has the fastest convergence rate of the three methods tested.

Figures 3.9 and 3.10 show the measured full width at half maximum(FWHM) of the neutron and photon peaks in the rise time spectra at different energies as well as the fitted curves. The experimental data are fitted separately in two energy regions with an exponential formula. The fitting parameters are used as the part of input data for the Bayes classifier, which determine the FWHM at different input energies (the paek poisition is determined by a perceptron in the code).

Figures 3.11 and 3.12 show the experimental results from different methods. The unmoderated Cf-252 neutron source is used, and the energy range covered is from 50 keV to 4.5 MeV neutron energy. The classification error is obtained by comparing the calculated ratio of photons to neutrons by the classifiers with calibration values, which are either pre-determined in the artificial synthesis process for the low energies (< 150

Table 3.1 Iteration Number of Different Weight Correction Methods in the Perceptron Classifier Training

Resolution	Fixed Correction	Absolute Correction	Fractional Correction
1/2048	6352	7139	2212
1/4096	19038	20269	6625
1/8192	39657	45773	13804

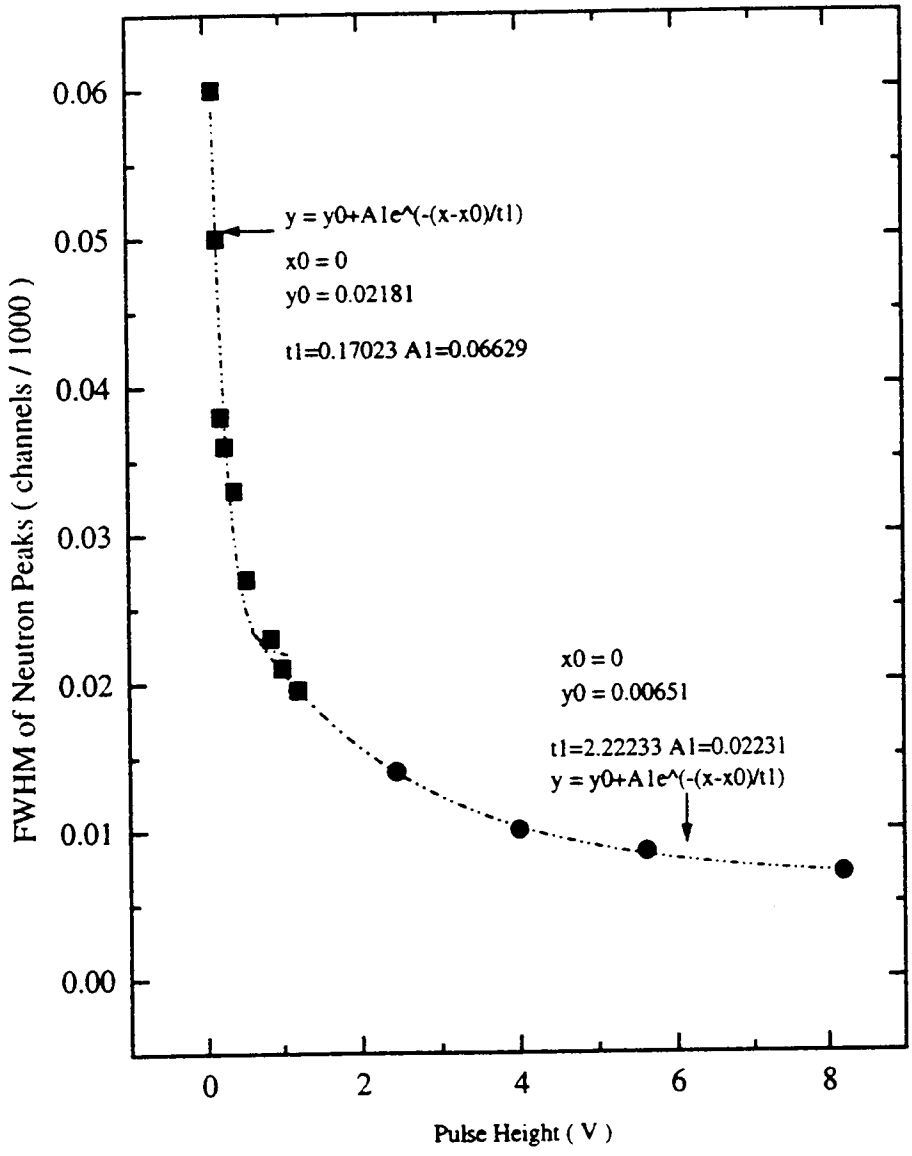


Figure 3.9 Measured FWHM of the Rise Time Spectra for Neutrons at Various Energy Pulse Amplitudes.

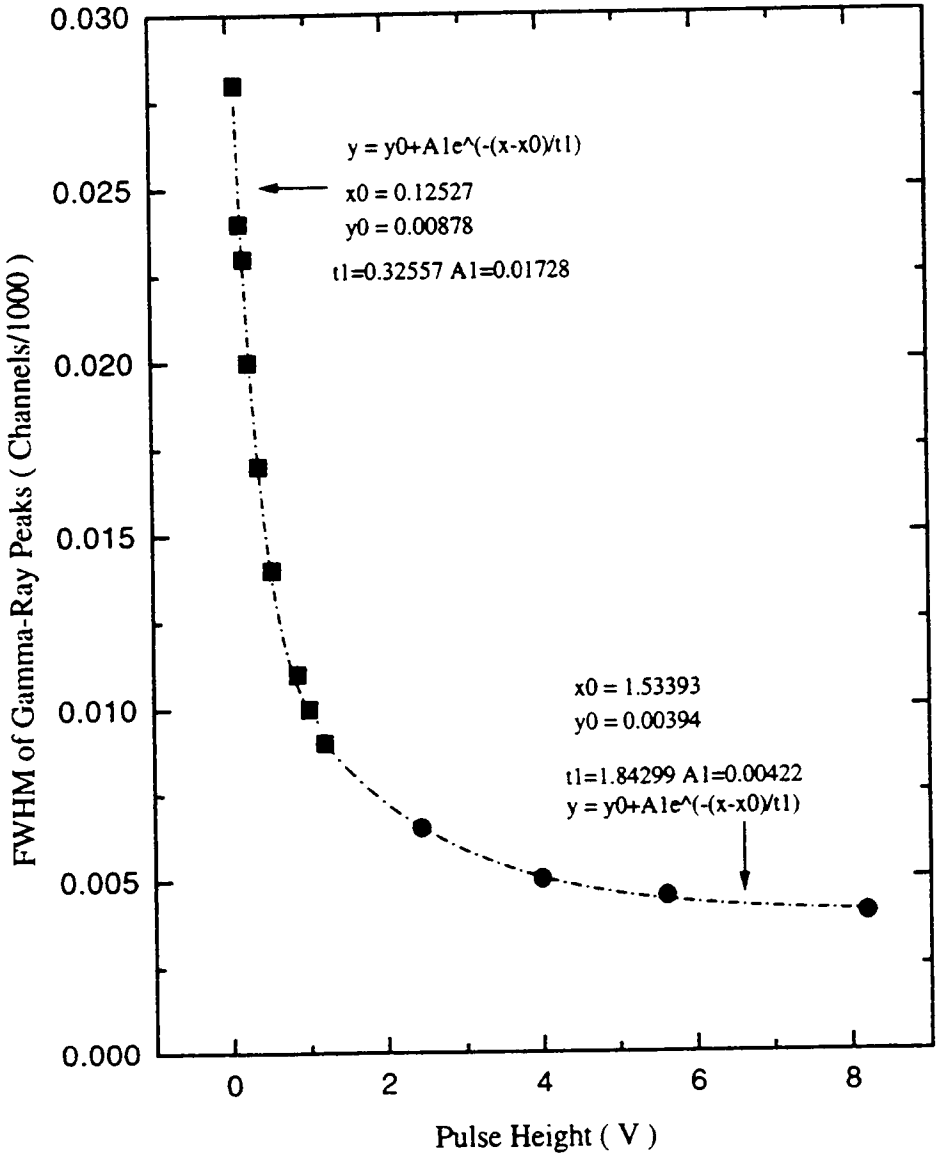


Figure 3.10 Measured FWHM of the Rise Time Spectra for Photons at Various Energy Pulse Amplitudes.

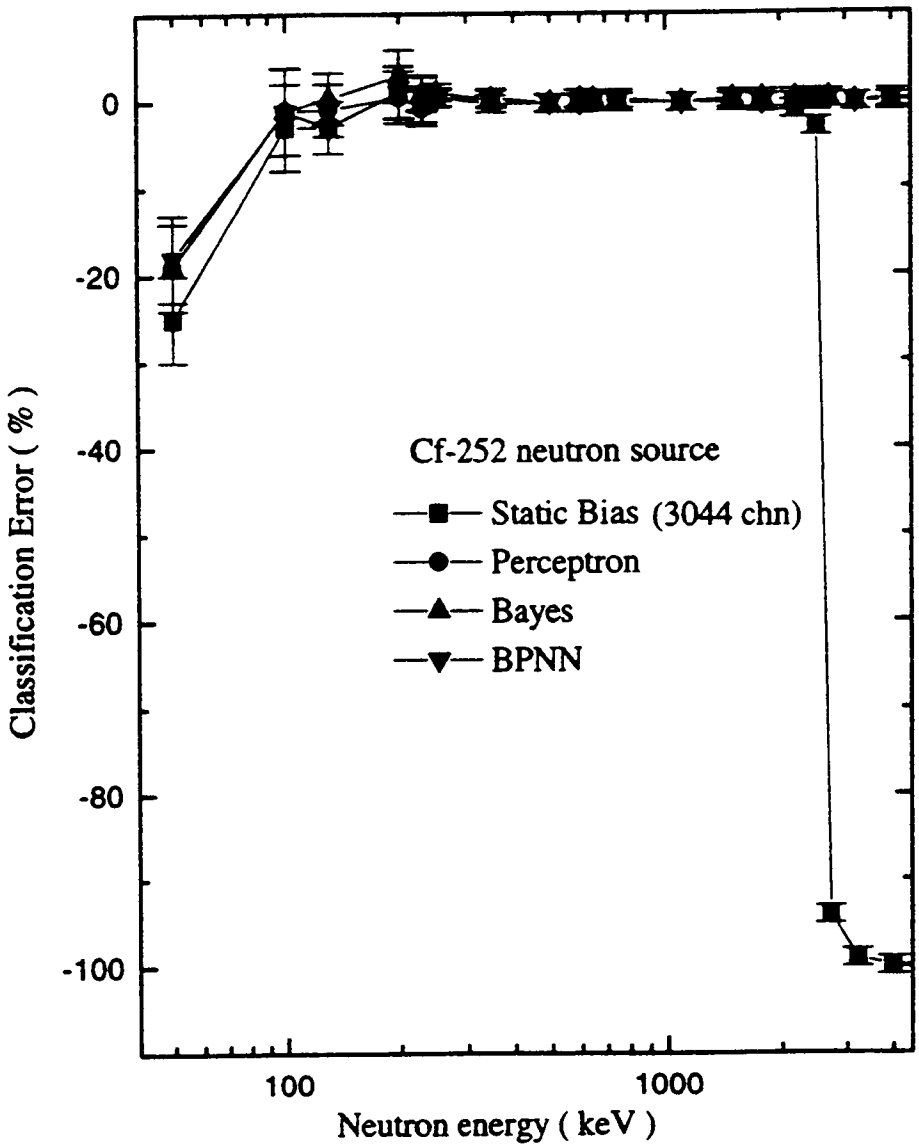


Figure 3.11 Classification Error vs. Neutron Energy for Various Methods (The static bias is set at 3.044).

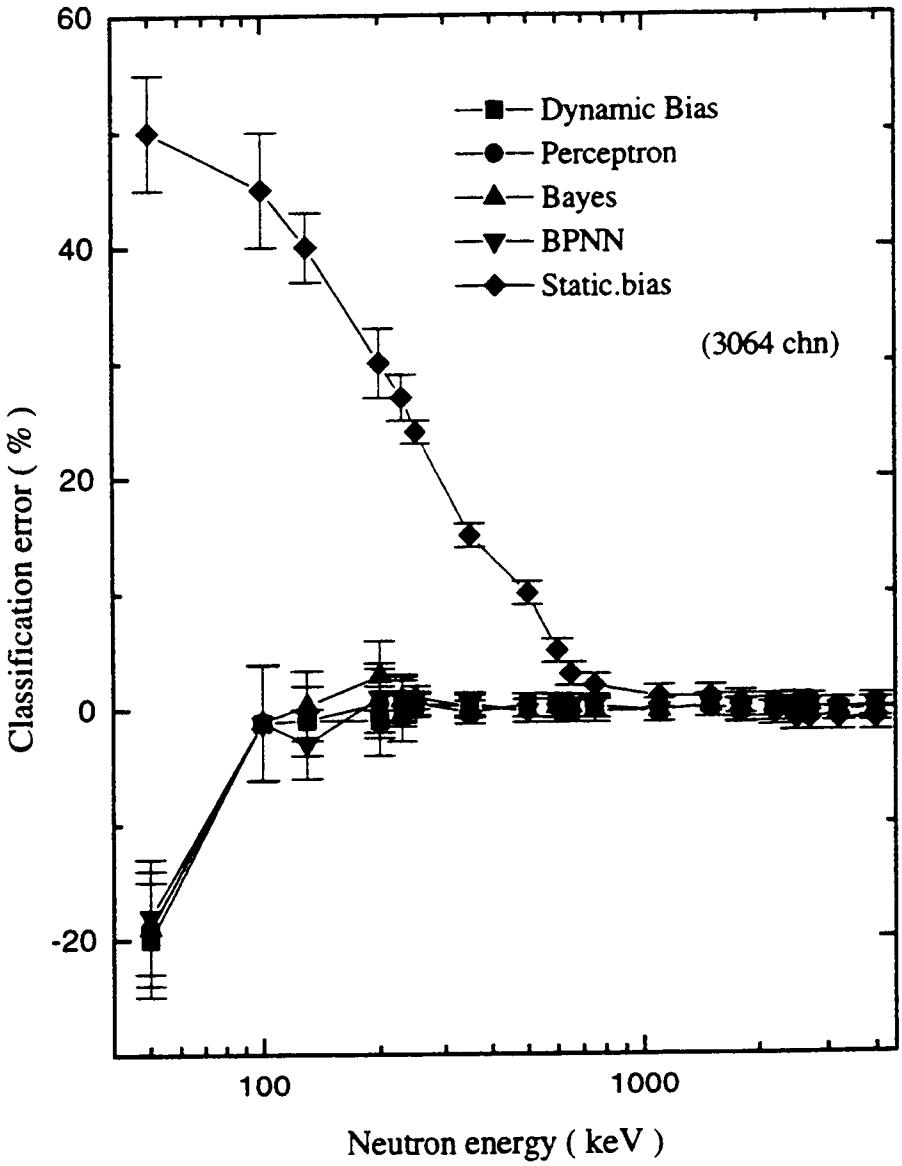


Figure 3.12 Classification Error vs. Neutron Energy for Various Methods (The static bias is set at 3.064).

keV equivalent electron energy) or are carefully measured at the high energy where the photon peak is well separated from the neutron peak (FOM > 1). It is apparent that all three classifiers, which implement the dynamic bias, have better performance than the static bias approach. The fixed bias method failed either at high energy when the PSD threshold is fixed at 3.044 in Figue 3.7, or at low energy when the PSD threshold is fixed at 3.064. The sharp drop in Figure 3.11 is due to the bias passing the narrow photon peak. Since the PSD bias is fixed for low energies (at 3.044 in Figure 3.7), at a certain high energy, all photons is mis-counted as neutrons. In Figure 3.12, the PSD bias is fixed for high energies (at 3.064 in Figure 3.7); therefore, as energy decreases, more neutrons mis-counted as the photons. However, the sharp drop doesn't occur in Figure 3.12 due to bias passing the broad neutron peak so the change is slow.

The classification quality of the linear perceptron is as good as that of the complicated BPNNC (Figures 3.11, 3.12, 3.14, and 3.15). The test using a nonlinear perceptron with a x^2 term shows no noticeable difference between the results of the linear and nonlinear perceptron; therefore, the straight boundary line is adequate and this makes the classifier much simpler (linear boundary line, see Figure 3.7 for example). As can be seen from these figures, the low energy limit is reduced to about 100 keV neutron energy with $\pm$ 5 % uncertainty by using dynamic bias classifiers.

As can be seen from Figure 3.13, 3.14 and 3.15, the Bayes classifier has better accommodation to the change in the ratio of photons to neutrons. When the ratio is increased ten times as that of the training condition, the classification error at 50 keV

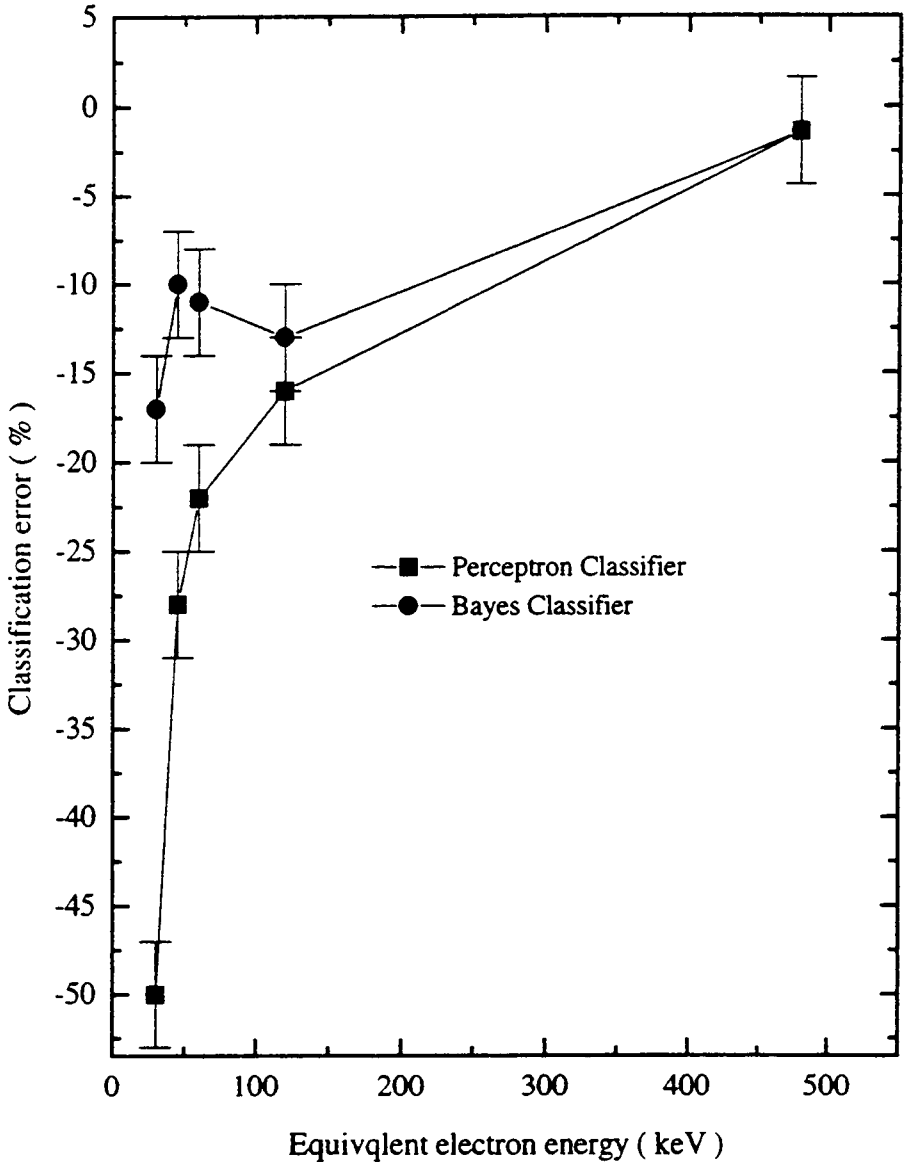


Figure 3.13 Classifacation Error for the Case with Ten Times Increase on the Ratio of Photons to Neutrons.

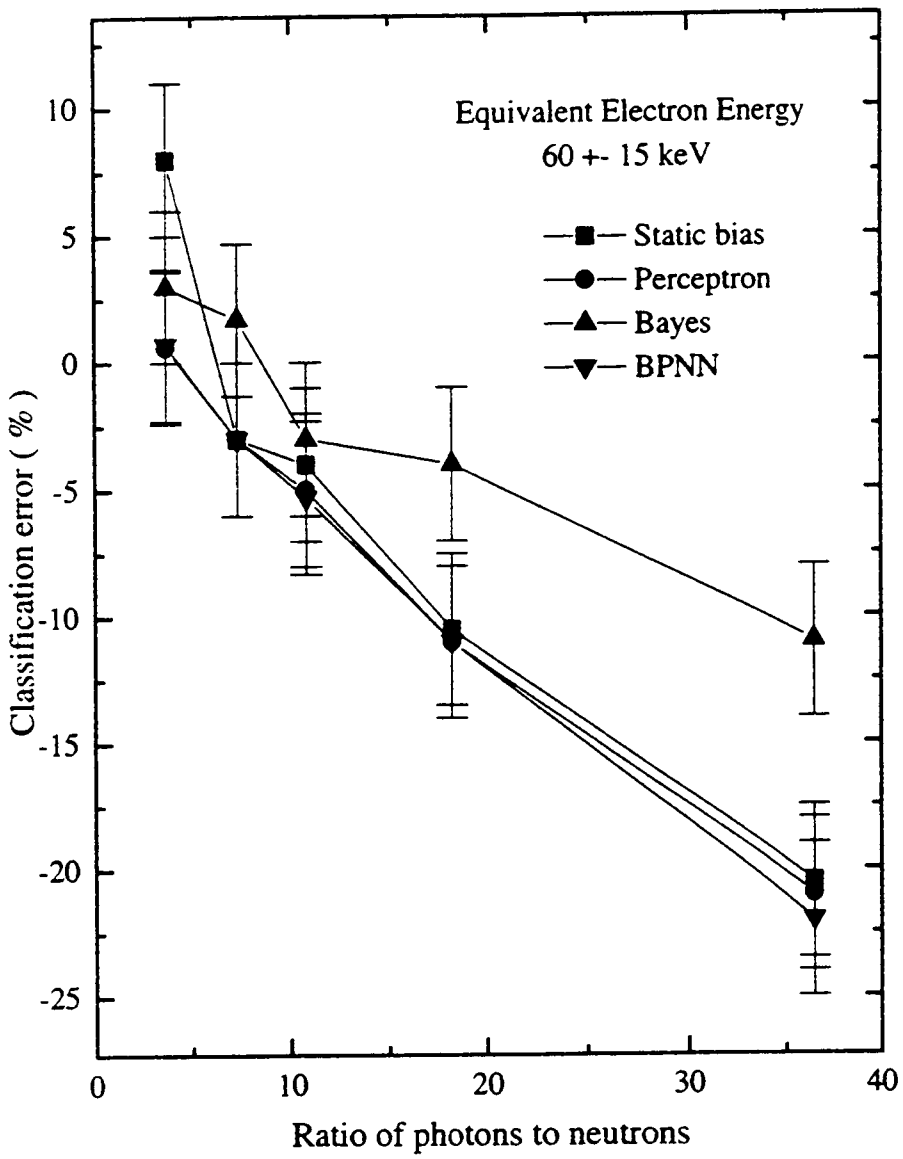


Figure 3.14 Accommodation Test Results of Classifiers at 60 keV.

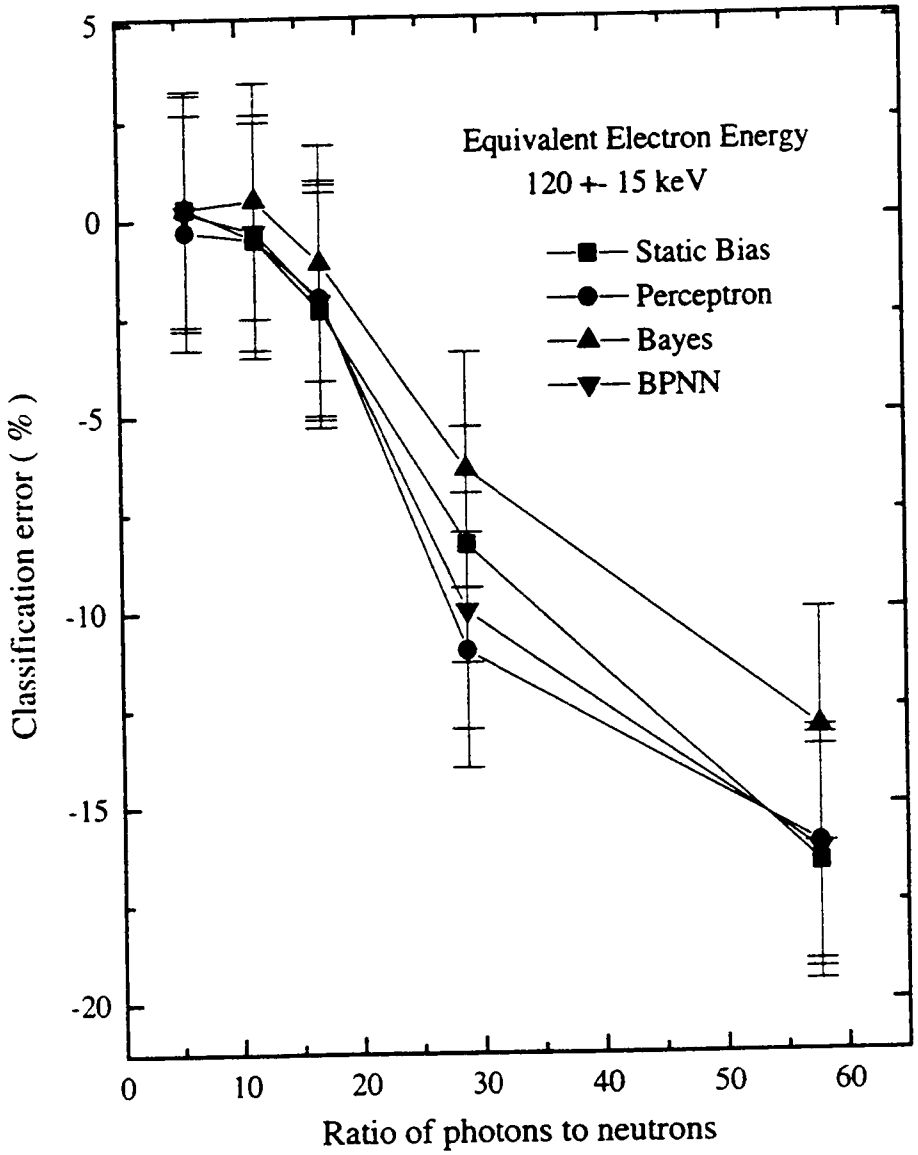


Figure 4.15 Accommodation Test Results of Classifiers at 120 keV.

equivalent electron energy is 10% for the Bayes, and 28 % for perceptron (Figure 3.13). At 120 keV equivalent electron energy (Figure 3.15), and with the ratio increase of a factor of ten, the classification error is 6% for the Bayes, and 11 % for the perceptron. These results show that the Bayes classifier has better accommodation to change in neutron source condition.

CHAPTER FOUR

STATISTICAL MODELS FOR EVALUATION OF SCINTILLATION MATERIALS AND PSD METHODS

4.1 Necessities of Statistical Models for Pulse Shape Discrimination

Parameter optimization for neutron-photon pulse shape discrimination at low energy by using "trial-and-error" methods is a time-consuming and difficult task. Experimental evaluation of the scintillation properties of materials and the experimental comparison of the PSD methods are affected by many factors, such as the light coupling, and equipment adjustment. Often conflicting conclusions are obtained, as mentioned in Appendix B.3.5. In order to eliminate human and equipment errors, statistical models for calculation of intrinsic PSD performance characteristics of scintillators and of PSD methods are necessary, especially at low energies where the number of photoelectrons is small and statistical fluctuation is a dominant factor. Another advantage of using models for evaluation of scintillation materials, comparison of methods and parameter optimization is time-savings and cost.

The present statistical models for the zero-crossing and the charge comparison methods are based on the Ranucci's work (1995)[19, 27]. The transit time spread and electronic noise, which are not included in the Ranucci's models, are included for this work. With this improvement, the present models provide better agreement with experiments than earlier studies by Ranucci. These models are tested with

characterization data of NE213, trans-stilbene and Borexino scintillators, and results show that the transit time spread and electronic noise are important for determining PSD performance.

4.2 Description of the Models

Statistical models that include scintillation decay time, transit time spread of a PMT and electronic noise of the equipment, are developed and programmed for zero-crossing and charge comparison methods. Assumptions for the models are as follows:

1. The light output of scintillators is proportional to energy deposition, and ambient conditions such as temperature, are negligible. Photoelectrons produced from an energy deposition event are assumed to be Gaussian distributed when it produces more than 30 photoelectrons are produced, and Poisson distributed when less than 30 are produced.

2. The time distribution of photoelectrons produced at the cathode of a photomultiplier tube is the same as the scintillation light time distribution of a scintillator. The time divergence caused by the reflection layer of a scintillator is neglected since its FWHM is less than 1 ns (for detectors less than 2" diameter). For a large volume scintillaor, this contribution should be included by using the equivalent transit time spread.

3. The transit time distribution is a Gaussian function. The average value is taken from the specification of the PMT manual. For example, the $FWHM_{transit}$ is about 2.4 ns for XP2020 PMT, while it is 8 ns for the XP2202B.

4. The electronic noise distribution is a Gaussian function and is measured by using either a single electron spectrum (determined by the FWHM of the single electron peak) or by using the spectrum of a precise pulse generator.

4.2 1 Model for the Zero-Crossing Method

In order to predict the FOM, the ratio of peak to valley, and the cross talk for the zero crossing method, the rise time distribution of the PMT output must be first calculated by the model. The rise time is defined as the time interval between 0.1 and 0.9 fraction of the maximum PMT voltage output (or some other value like 0.1 to 0.6). Therefore, to obtain the rise time distribution, the model first has to calculate the time distributions corresponding to the 0.1 and 0.9 fractions, and then combine them to obtain the rise time distribution.

The time distribution of organic scintillation produced by radiation is generally expressed by the form[27]

$$p\ (t) = \frac{1}{Q} \sum_{i=1}^{k} \frac{q_i}{T_i} e^{-\frac{t}{T_i}}, \qquad (1)$$

where p(t) is the normalized probability density function of a scintillation, Q is the total light output (proportional to total energy deposition), q_i and T_i are the light yield and the decay time constant of the ith scintillation component, respectively. Note that q_i and T_i are dependent on scintillator type and radiation.

In the case of NE213 and BC501A, the previous formula can be simplified to

$$p(t) = \frac{1}{Q}\left(\frac{q_1}{T_1}e^{-\frac{t}{T_1}} + \frac{q_2}{T_2}e^{-\frac{t}{T_2}}\right), \qquad (2)$$

where subscript 1 represents the fast component (T_1 is about 3.2 ns), and subscript 2 represents the slow component (T_2 is about 30 ns). The scintillation light is detected by the cathode of a photomultiplier tube. If one neglects the time spread caused by the reflector of a scintillation detector, the time distribution of the photoelectrons produced at the cathode will be the same as the scintillation light.

Each detected photoelectron is multiplied during its travel from the cathode to the anode and produces a single photoelectron pulse (SPE) at the anode. The integration of these SPE pulse results in a voltage pulse output whose rise time is related to the scintillation decay time and PMT response. Assume that n is the total number of emitted photoelectrons produced by a monoenergetic radiation, x is the fraction of the maximum of voltage pulses, and j is the number of photoelectrons corresponding to the fraction x. It is apparent that

$$x = \frac{j}{n};$$

thus, the calculation for the time distribution of fraction x can be replaced by the calculation of the distribution of the time at which j SPE pulses out of total of n reach the PMT anode.

Due to the statistical nature of electron multiplication by the dynodes of a PMT, the transit time has a distribution described by a Gaussian function $N(t,\sigma_m)$. Hence, the distribution of the SPE pulses is a convolution of two distribution functions, and can be expressed by

$$P(t) = \frac{1}{\sqrt{2\pi}\sigma_m} \int_0^{\infty} p(t') e^{-\frac{(t-t')^2}{2\sigma_m^2}} dt' . \qquad (3)$$

The distribution of a SPE pulse, which arrives at the anode at time t, is equal to

$$\frac{n!}{(n-1)!*1!} P(t),$$

while the distribution of the remaining (n-1) SPE pulses having not reached the anode at the time t is expressed by

$$(1\text{-}F(t))^{n-1},$$

where

$$F(t) = \int_0^t P(t')dt' . \qquad (4)$$

Therefore, the time distribution of the first SPE pulse at the anode is equal to

$$p_1(t) = nP(t)(1-F(t))^{n-1}. \qquad (5)$$

Equation (5) is a probability density function and

$$\int_0^{\infty} p_1(t')dt' = \int_0^{\infty} nP(t')(1-F(t'))^{n-1}dt' = \int_0^{\infty} n(1-f(t'))^{n-1}dF(t') = \int_{\infty}^{0} d(1-F(t'))^n = 1.$$

The time distribution of the jth micro-pulse arriving at the anode can be calculated as shown below.

The distribution of (j-1) SPE pulses arriving at the anode before the time t is equal to

$$p'_{j-1}(t) = \frac{n!}{(n-(j-1))!(j-1)!}(F(t))^{j-1} .$$

For the remaining (n-(j-1)) SPE pulses, based on Equation (5), the probability of the first SPE pulse arriving at the anode can be expressed by

$$(n-(j-1))P(t)(1-F(t))^{n-(j-1)-1} .$$

Therefore, the time distribution of the jth SPE pulse arriving at the anode is the intersection of the previous two functions and is equal to

$$p_j(t) = \frac{n!}{(n-j)!(j-1)!}(F(t))^{j-1} P(t)(1-F(t))^{(n-j)}. \qquad (6)$$

Furthermore, the statistical fluctuation of the total number of SPE pulses should be included in the calculation. Therefore, the distribution of the time at which the pulse reaches the fraction x of its maximum is given by

$$p_j(t) = \sum_{n=j}^{\infty} p(n) \frac{n!}{(n-j)!(j-1)!}(1-F(t))^{(n-j)}(F(t))^{(j-1)} P(t), \qquad (7)$$

j= x n,

$$p(n) = \frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{(n-\mu)^2}{2\sigma_n^2}}, \quad \text{for } \mu > 30, \qquad (8)$$

$$P(n) = \frac{\mu^n e^{-\mu}}{n!}, \qquad \text{for } \mu < 30, \qquad (9)$$

where p(n) is the distribution of the total number of SPE pulses, μ is the mean value of the total number n for a monoenergetic radiation, and is obtained in the energy measurement.

The full width at half maximum ($FWHM_{cal}$) and the peak separation between the neutrons and photons can be obtained from the rise time distribution. Considering the electronic noise contribution, the $FWHM_{cal}$ must be broadened to represent the real situation. The total FWHM is equal to

$$FWHM^2 = FWHM_{cal}^2 + FWHM_{noiset}^2, \qquad (10)$$

where $FWHM_{noiset}$ is the contribution from electronic noise. Note that $FWHM_{noiset}$ can be calculated by

$$FWHM_{noiset}=FWHM_{noise}/k, \qquad (11)$$

where $FWHM_{noise}$ is the full width at half maximum of the electronic noise spectrum and k is the average slop in the pulse rise time at a given fraction (CFD).

The average slope k can be calculated from Equations (4) and (3). First, let F(t)=x and find t_x. Then, calculate $P(t_x)$ by using Equation (3), where $P(t_x)$ is the slope rate of pulses at fraction x.

Finally, the figure of merit is calculated from $(FWHM)_{neutron}$, $(FWHM)_{\gamma}$, and peak separation in the rise time distribution.

4.2.2 Model for the Charge Comparison method

In the charge integration method, the ratio of the fast component (or the slow component) to the total light output is determined by integration of SPE pulses (corresponding to emitted photoelectrons) in two time windows. One window is for the fast component and the other for the total. The probability of one SPE pulse arriving at the anode in the fast time window T_f is

$$p_{fast} = \int_0^{T_f} P(t)dt,$$

where

$$P(t) = \frac{1}{\sqrt{2\pi}\,\sigma_m} \int_0^{\infty} p(t') e^{-\frac{(t-t')^2}{2\sigma_m^2}} dt'.$$

This is a Bernoulli trial problem, the distribution of j SPE pulses in T_f is a binomial distribution and is written as

$$p_{fast}(j) = \sum_{n=j}^{\infty} p(n) \frac{n!}{(n-j)!\,j!} (p_{fast})^j (1-p_{fast})^{(n-j)}, \qquad (12)$$

where p(n) is described by Equation (8) or (9).

Equation (12) describes the fast component distribution for both neutrons and photons. It can be used to calculate the FWHM and the peak separation for neutrons and photons, and thus used to determine FOM.

4.3 Implementation, Results and Discussions

Two computer codes are written in FORTRAN to implement the statistical models for the zero-crossing and charge comparison methods. The codes need an input data file which contains the time window set up in the charge comparison method (or the constant fraction discrimination factor in the zero-crossing method), the transit time spread of a photomultiplier tube, the total average photoelectron number (related to energy), the scintillation decay time constant and intensity. It is an interactive code so that one can set different parameters to compare the calculated PSD performance (FOM value) and to select the best parameters for specific scintillation materials. The output

data file is used for plotting PSD spectra. The name and functions of the source codes and input files are as follows (details are in Appendix F).

zc.for	the source file for the zero-crossing method.
Charg2.for	the code for the charge comparison method.
chargn.dat	the input data file for an early NE213 scintillator.
chargs.dat	the input data file for a trans-stilbene crystal.
chargb.dat	he input data file for a Borexino scintillator.

The codes are used to calculate the rise time distribution in the zero-crossing method and the slow component distribution in the charge comparison method under different conditions, for different scintillation materials, CFD values, time windows, and transit time spread of photomultiplier tubes. The slow component distribution is selected for computation, instead of the fast component distribution, because its photon and neutron peak locations are of the same order as that in the rise time distribution in the zero-crossing method, where the photon peak is on the left side of the neutron peak. This arrangement makes comparisons more intuitive.

The example for the calculated rise time distribution of photon and neutron induced scintillation pulses and for the distribution of delayed photoelectrons in the charge comparison are shown in Figures 4.1 and 4.2. Note that the total photoelectron number is proportional to the energy deposition of radiation inside the scintilltors. Since the photoelectron production rate for trans-stilbene is about 2.3/keV, a total 100

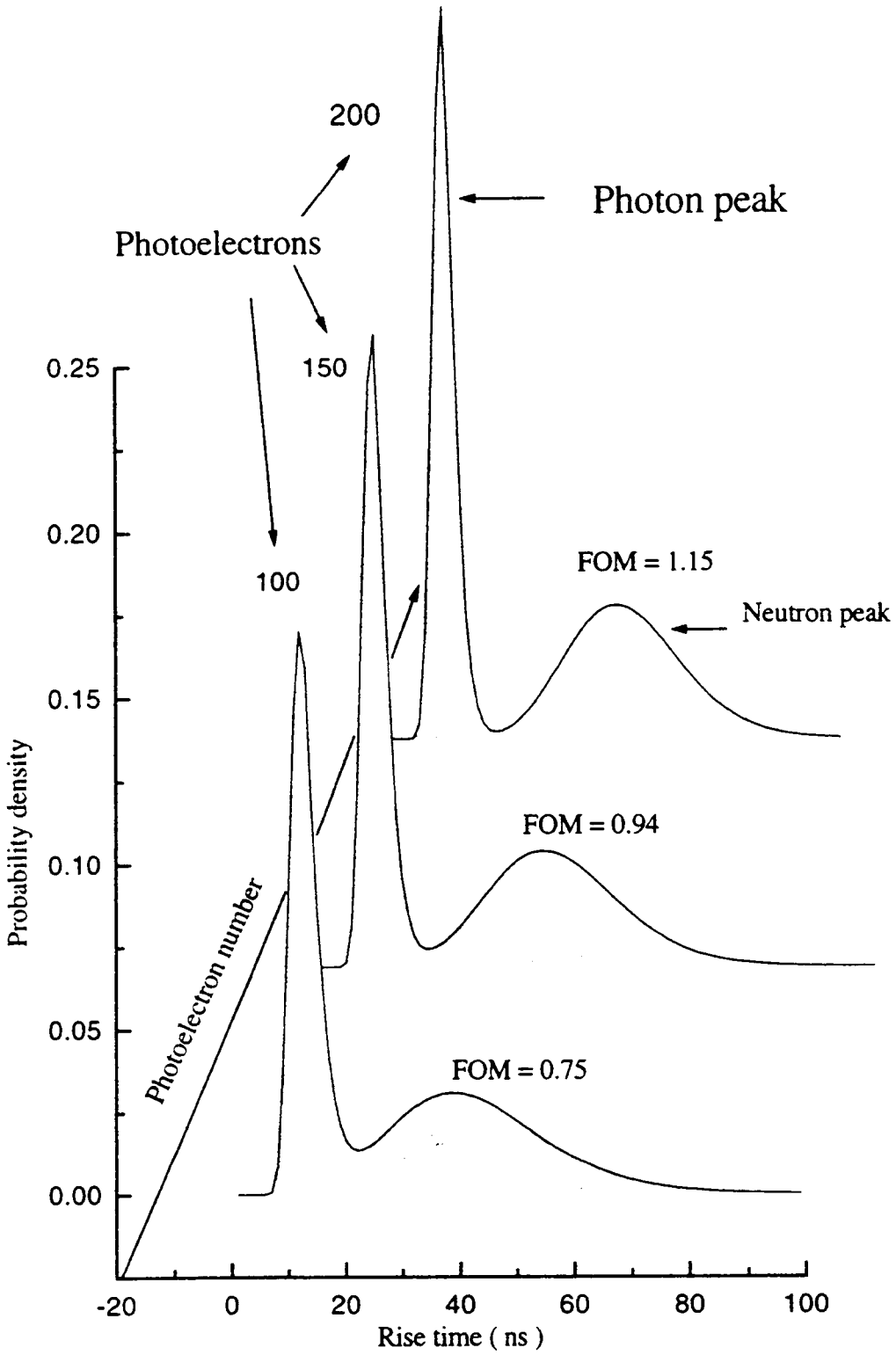


Figure 4.1 Calculated Rise Time Distributions of Photon and Neutron Induced Scintillation Pulses in a Trans-Stilbene Crystal.

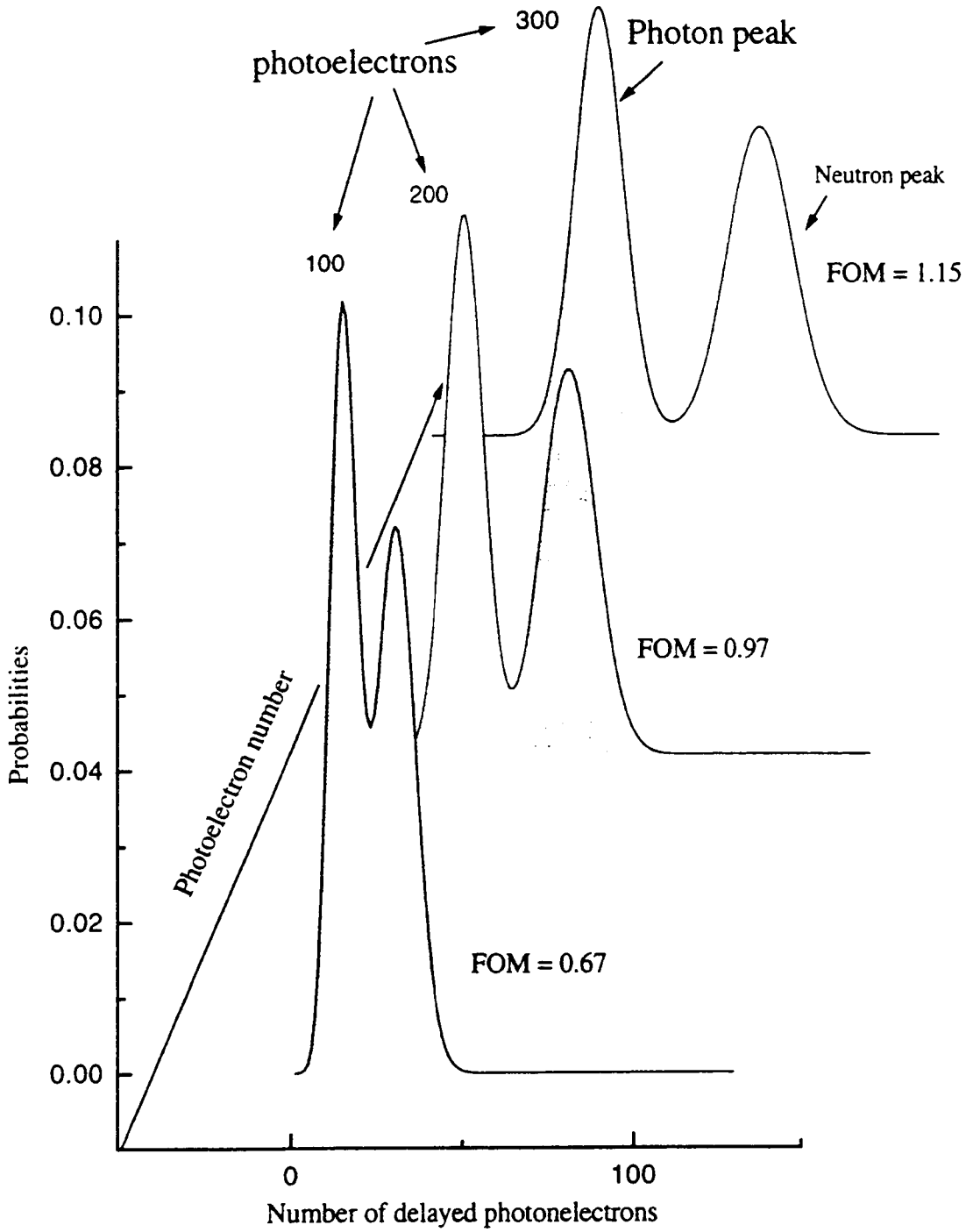


Figure 4.2 Calculated Distributions of Delayed Photoelectrons for Photon and Neutron Events in a Trans-Stilbene Crystal.

photoelectrons correspond to 43 keV deposited energy. Figures 4.3 and 4.4 show the calculated rise time spectrum of photon and neutron events and the calculated slow component distribution of photon and neutron events, respectively, at 44 keV energy deposited in the trans-stilbene crystal. The parameters used in calculation are 0.8 CFD for zero crossing and 20-120 ns slow window for the charge comparison. These figures also show that if the value of the figure of merit is larger than one, the neutron events will be discriminated from the photon with a small error (or small cross talk).

The ratios of photons to neutrons in all calculations are assumed to be one. However, this can be changed in the calculation to any value as necessary to predict PSD performance of a given detector at different photon-neutron source conditions.

The PSD performance characteristics, such the separation of two peaks, the full width at half maximum (FWHM), and the figure of merit can be identified from these distributions, and then compared. Figure 4.5 is an example of the results of these calculations. As can be seen from this figure, the full width at half maximum for both neutron and photon peaks increase and the separation between two peaks becomes smaller, as the energy deposited decreases; therefore, the figure of merit decreases as energy deposition decreases.

Figure 4.5 also shows the need for dynamic bias in the neutron-photon pulse shape discrimination since the best threshold to separate photons from neutrons is an energy dependent line. The dynamic bias is required from the intrinsic nature of pulse shape discrimination.

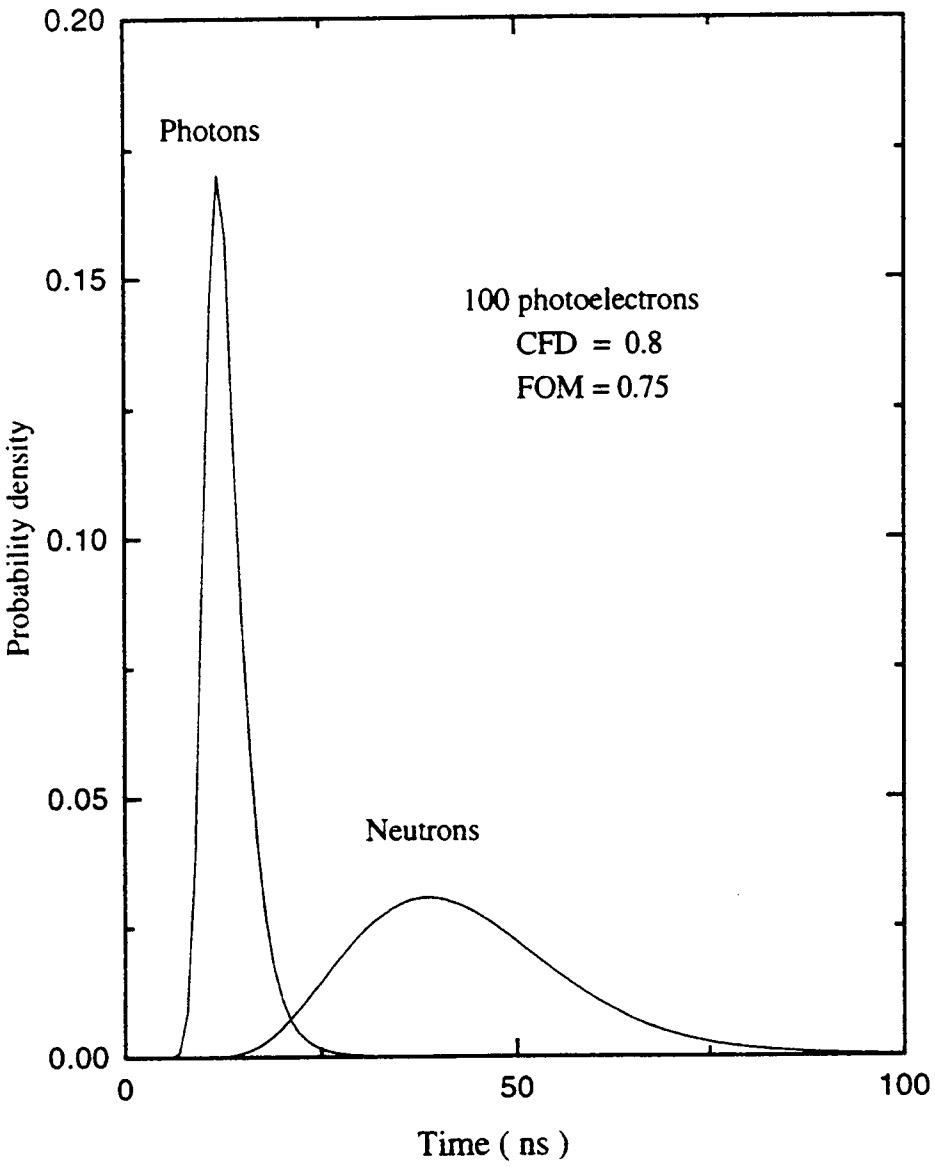


Figure 4.3 Separated Photon and Neutron Contributions for 100 Photoelectrons in Figure5.1.

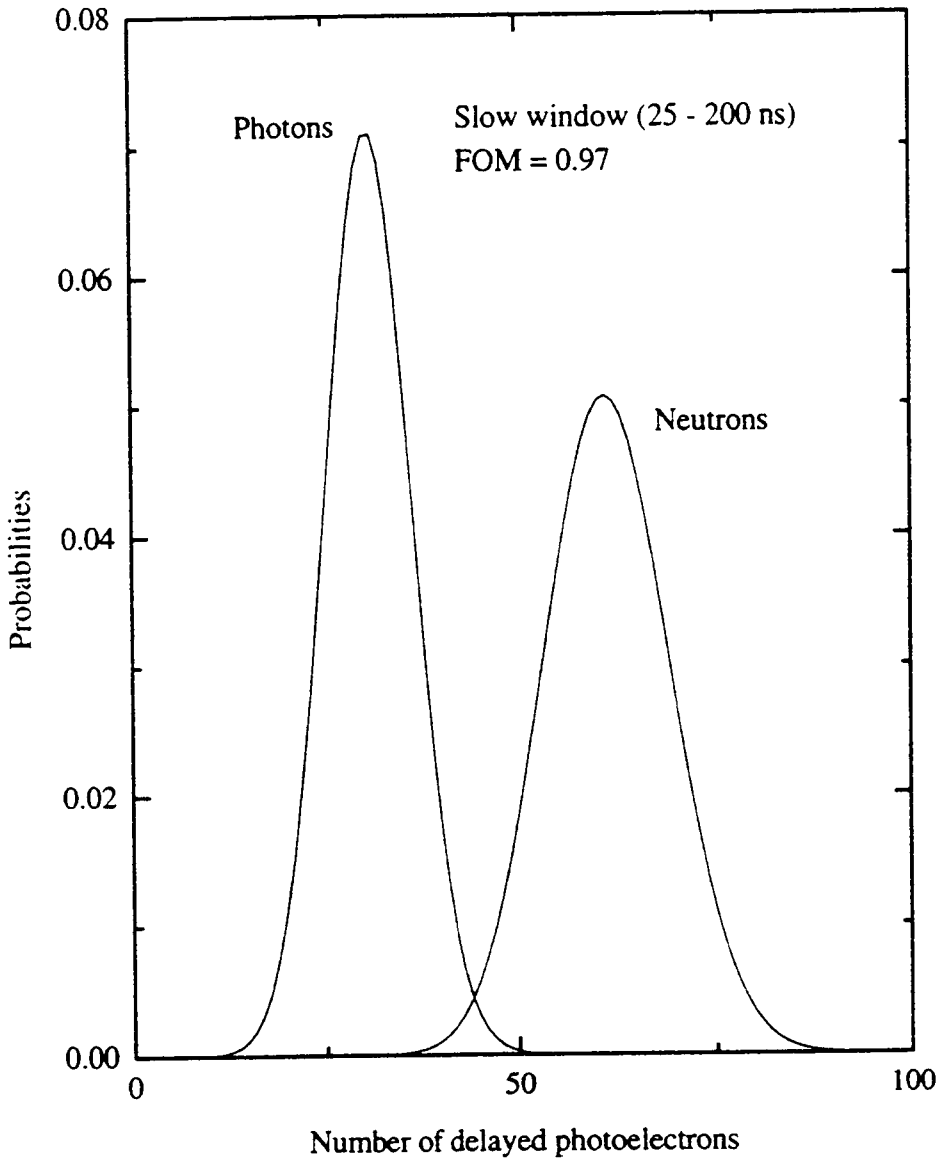


Figure 4.4 Separated Photon and Neutron Contributions for 200 Photoelectrons in Figure 5.2.

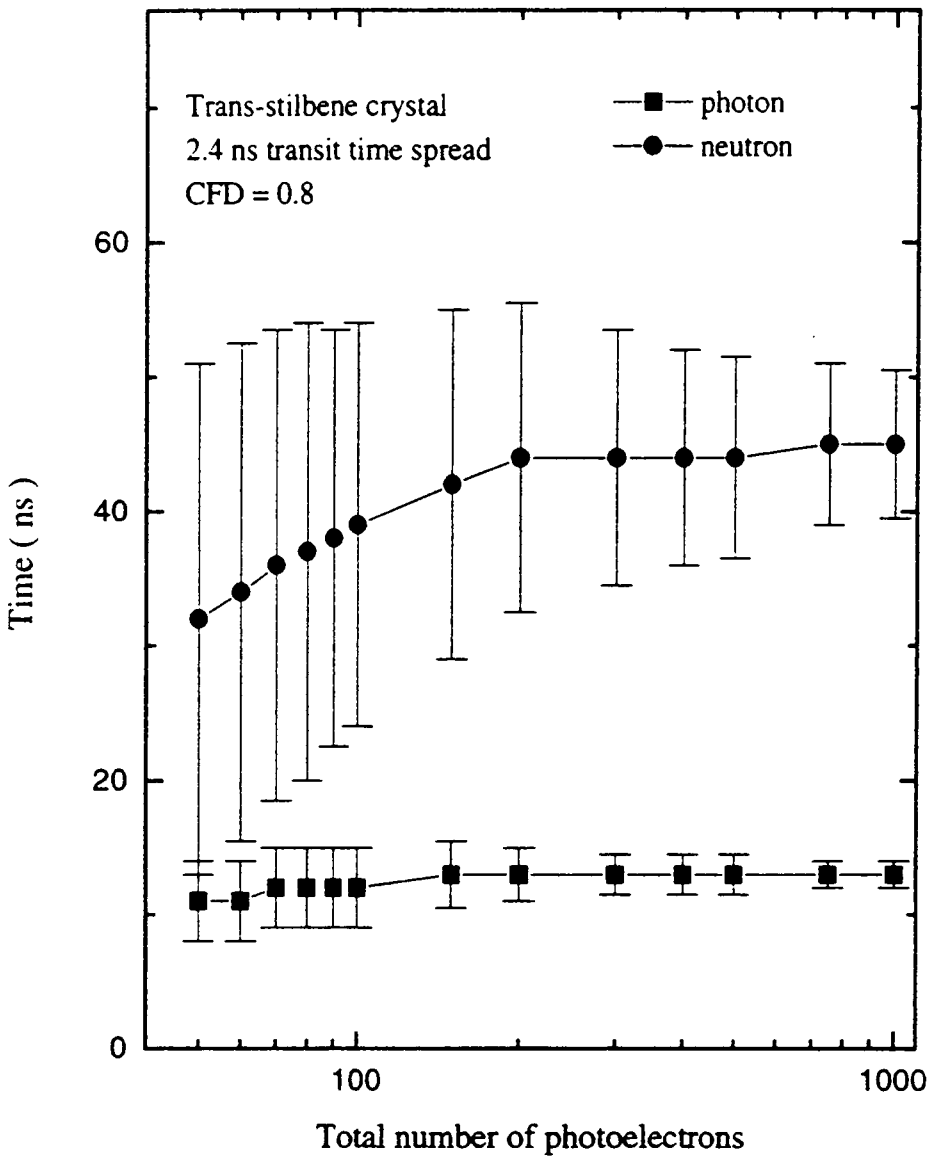


Figure 4.5 Comparison of the FWHMs of Photon and Neutron Peaks in the Rise Time Distributions.

4.3.1 Scintillation Properties of Different Scintillators

The most important data needed for the calculation are the scintillation decay time constants and intensities of scintillation components, which can be measured with the single electron time sampling technique. Three scintillation materials, early NE213[45], trans-stilbene[46] and Borexino scintillator[27], are used in these calculations. Table 4.1 lists their scintillation properties for photons and neutrons. The present calculations are based on these parameters and are directly relevant to these scintillators. It should be noticed that the procedure can be applied to other scintillation materials and has general validity.

Figures 4.6 and 4.7 show the integrated scintillation curves induced by photons and neutrons in these scintillators. These curves can be used to estimate the PSD properties of these materials. As shown in Figure 4.7, the Borexino sctintillator has larger separation between responses to photons and neutrons than the early NE213 so that better neutron-photon pulse shape discrimination is expected.

4.3.2 Parameter Optimization and the Low Energy Limitation

To obtain the best PSD performance for a given scintillator with a given method, optimization of measurement parameters is very important. The key parameter in the zero-crossing method is the CFD (constant fraction discrimination) value. For the charge comparison method it is the fast window setting (or the slow window setting). Both of them are optimized for the three scintillation detectors mentioned previously. The

Table 4.1 Properties of some Scintillators

Scitilator	Early NE213		Tans-Stilbene		Borexino	
Excitation	n	γ	n	γ	n	γ
τ_1 (ns)	3.94	3.36	5.5	4.5	2.03	1.75
τ_2 (ns)	47.1	24.6	46	14	13.10	8.12
τ_3 (ns)					56.19	71.25
τ_4 (ns)					399.9	
q_1	0.527	0.556	0.626	0.755	0.625	0.856
q_2	0.473	0.444	0.374	0.245	0.162	0.131
q_3					0.108	0.013
q_4					0.105	

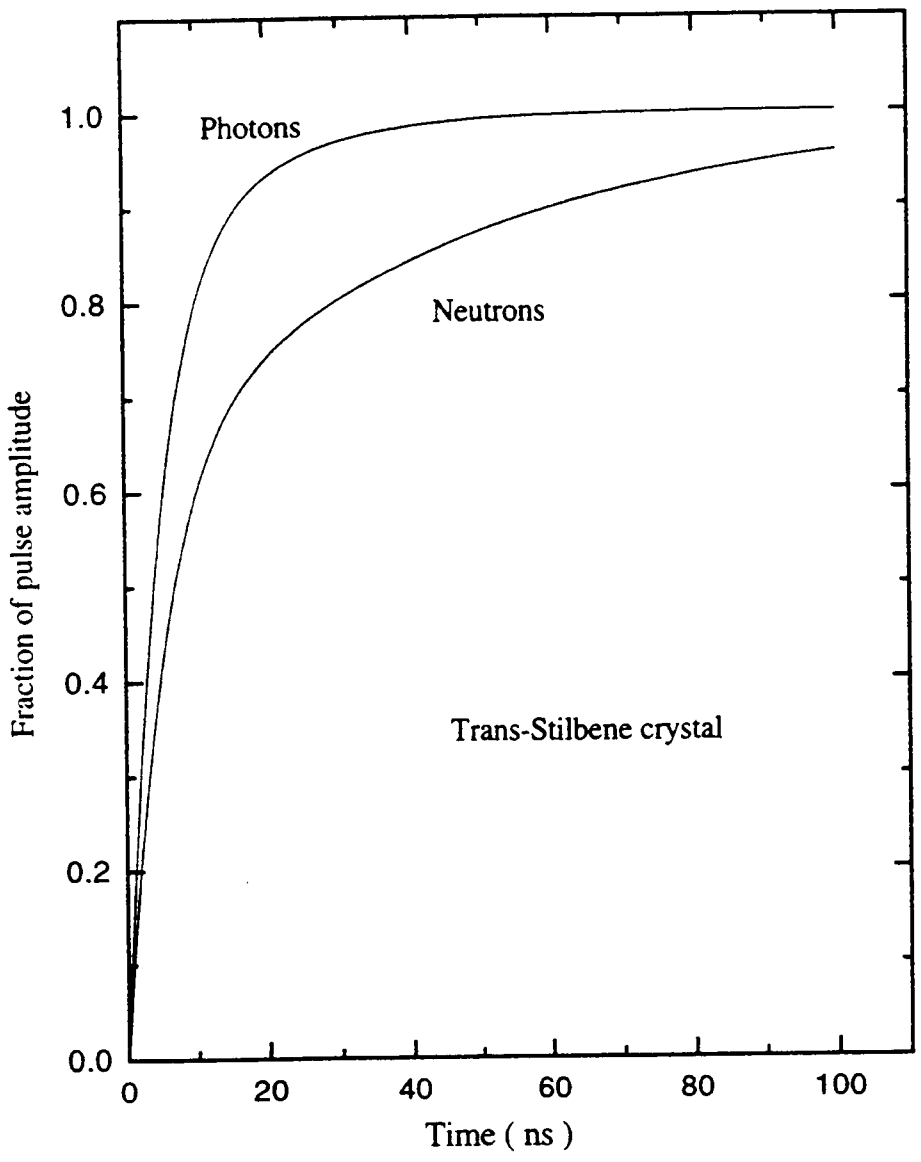


Figure 4.6 Integrated Scintillation Curves Induced by Photons and neutrons in a Trans-Stilbene Crystal.

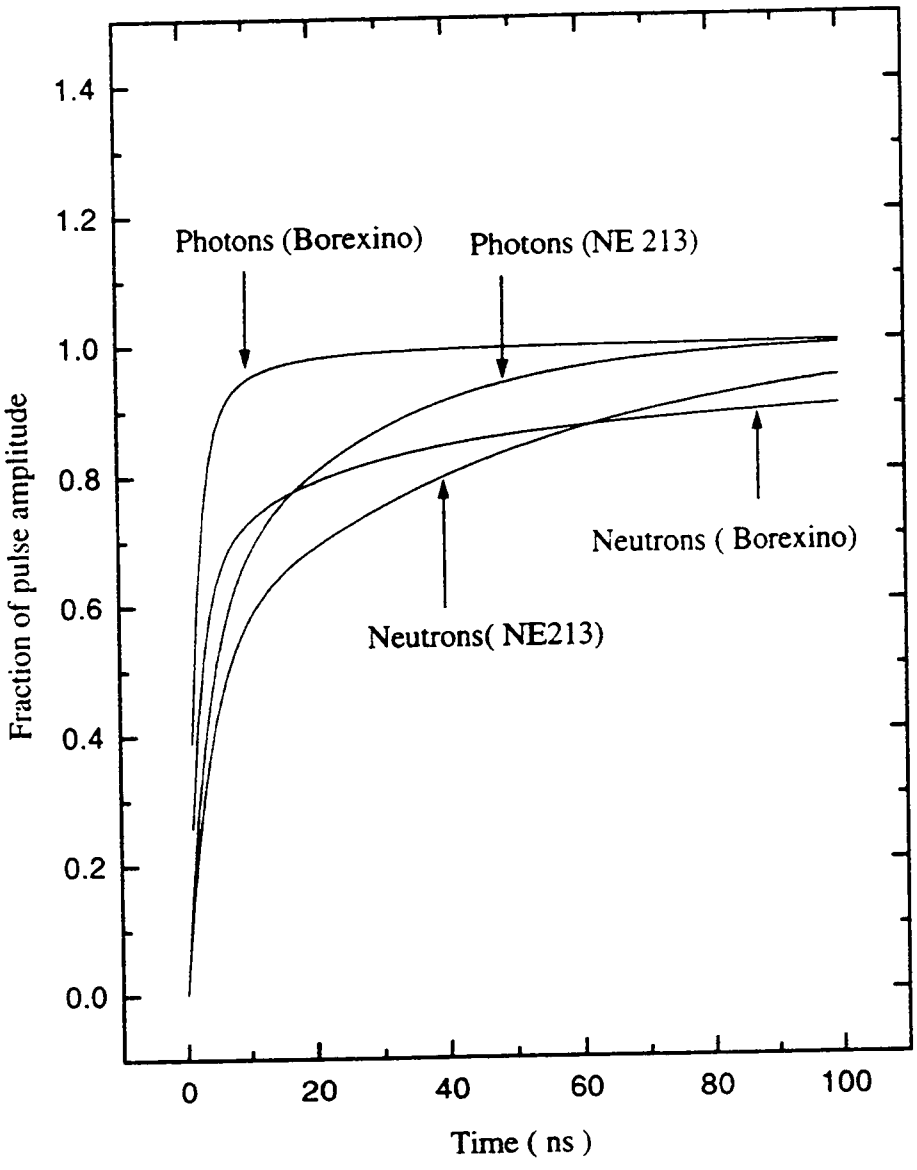


Figure 4.7 Integrated curves of scintillation induced by photons and neutrons in NE213 and Borexino scintillators.

calculations show that both the best CDF setting in the zero-crossing method and the best window setting in the charge comparison are scintillator dependent; however, the CFD setting in the zero-crossing method is less sensitive to the scintillator than the charge comparison method.

Figure 4.8 shows the effect of the CFD setting on the PSD performance of an early NE213 scintillator, where 0.6-0.9 CFD values are evaluated. As shown in this figure, 0.8 CFD has the best FOM value over the 100-1000 photoelectrons range. A similar conclusion is drawn from the test results for the trans-stilbene crystal (Figure 4.9). Figure 4.10 shows the results from the Borexino calculation, where 0.9 CFD setting provides the best PSD performance above 75 photoelectrons. The transit time spread of XP2020 PMT (2.4 ns) is used in these calculations.

It is concluded that 0.8 is the best CFD setting for NE213 and trans-stilbene for the zero-crossing method under some conditions. This result is consistent with experimental results presented in Appendix B.3.3 (the optimal CFD parameter in the zero-crossing method for the BC-501A-XP2020 PMT detector is 0.7, which is a selection on Ortec552).

When the transit time spread is 2.4, the best calculated CFD setting for the Borexino scintillator is 0.9. For the trans-stilbene crystal with a small tranit time spread PMT (< 1 ns, Figure 4.11), the PSD performance with 0.9 CFD is superior to others. However, it is apparent from Figure 4.11 that the PSD performance at 0.9 CFD setting is strongly affected by the transit time spread. As the transit time increases to more than 2

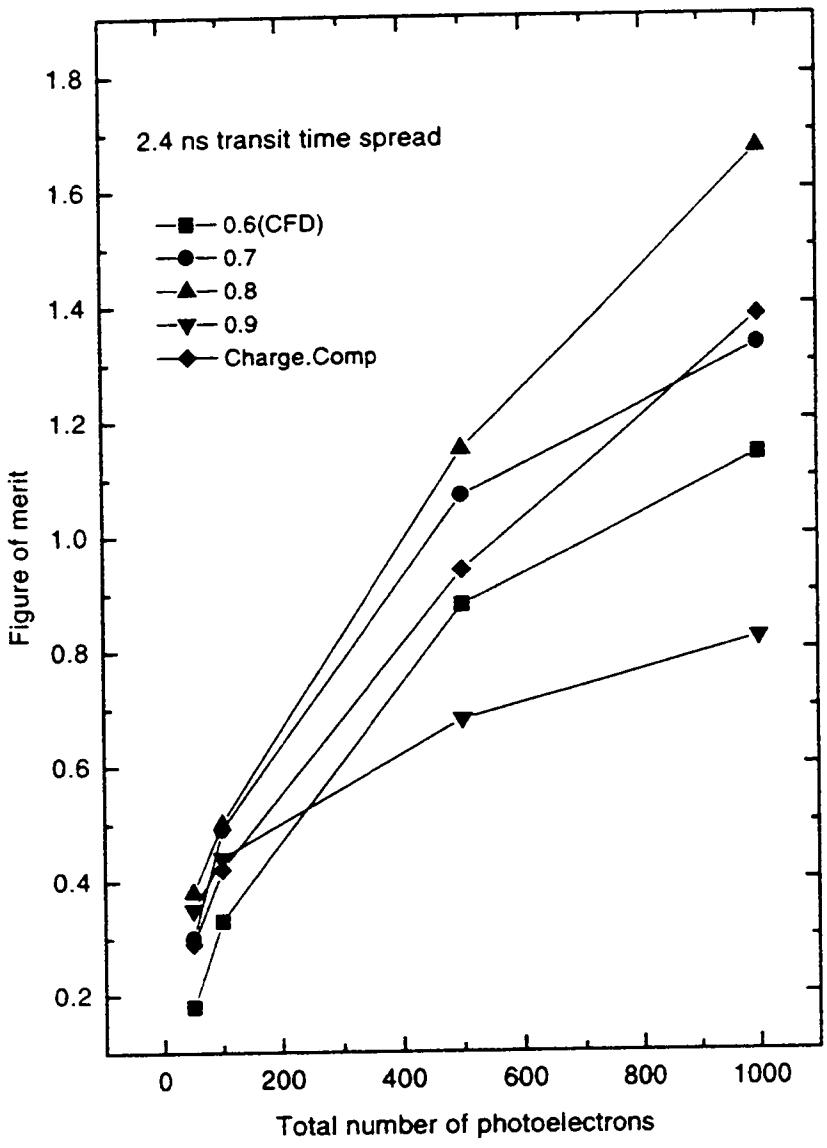


Figure 4.8 Effect of the CFD Setting on PSD Performance in the Zero-Crossing Method for NE213.

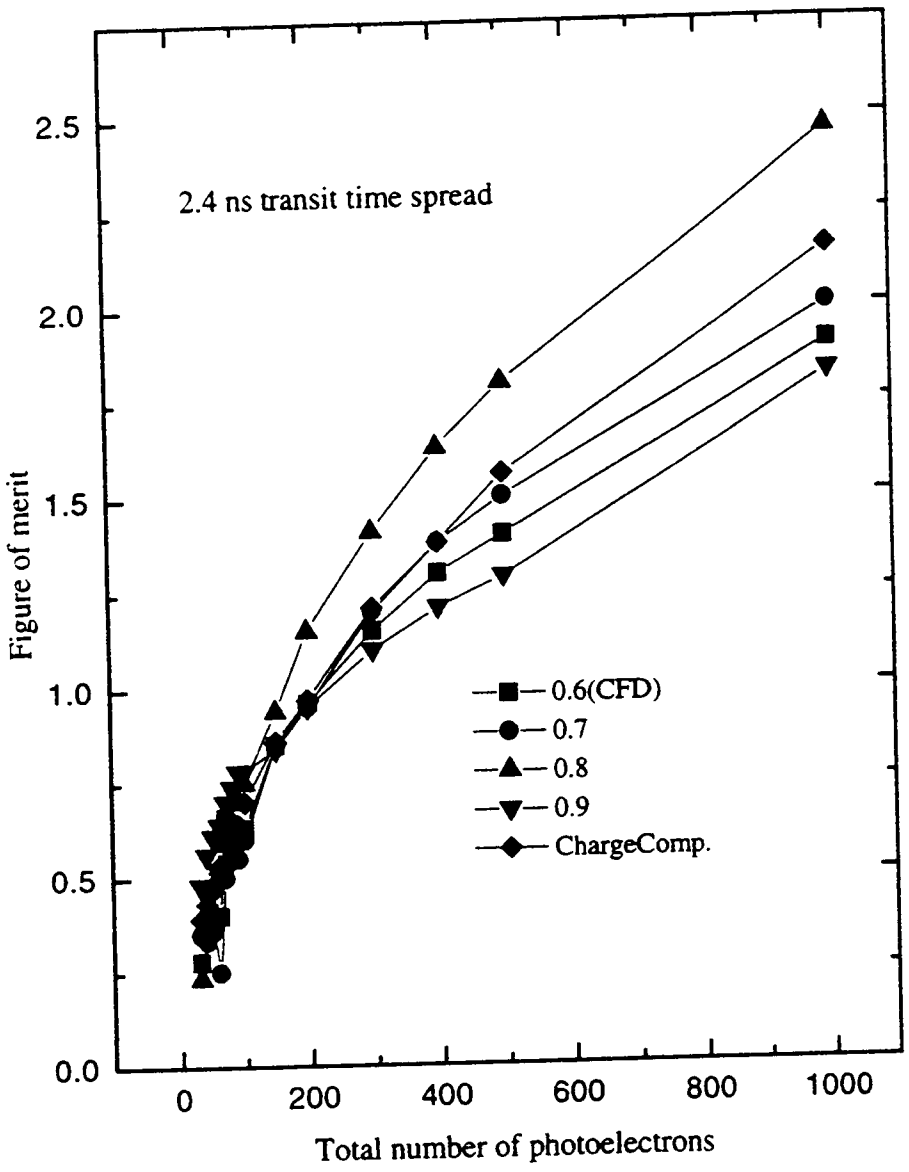


Figure 4.9 Effect of the CFD Setting on PSD Performance in the Zero-Crossing Method for the Transit-Stilbene Crystal.

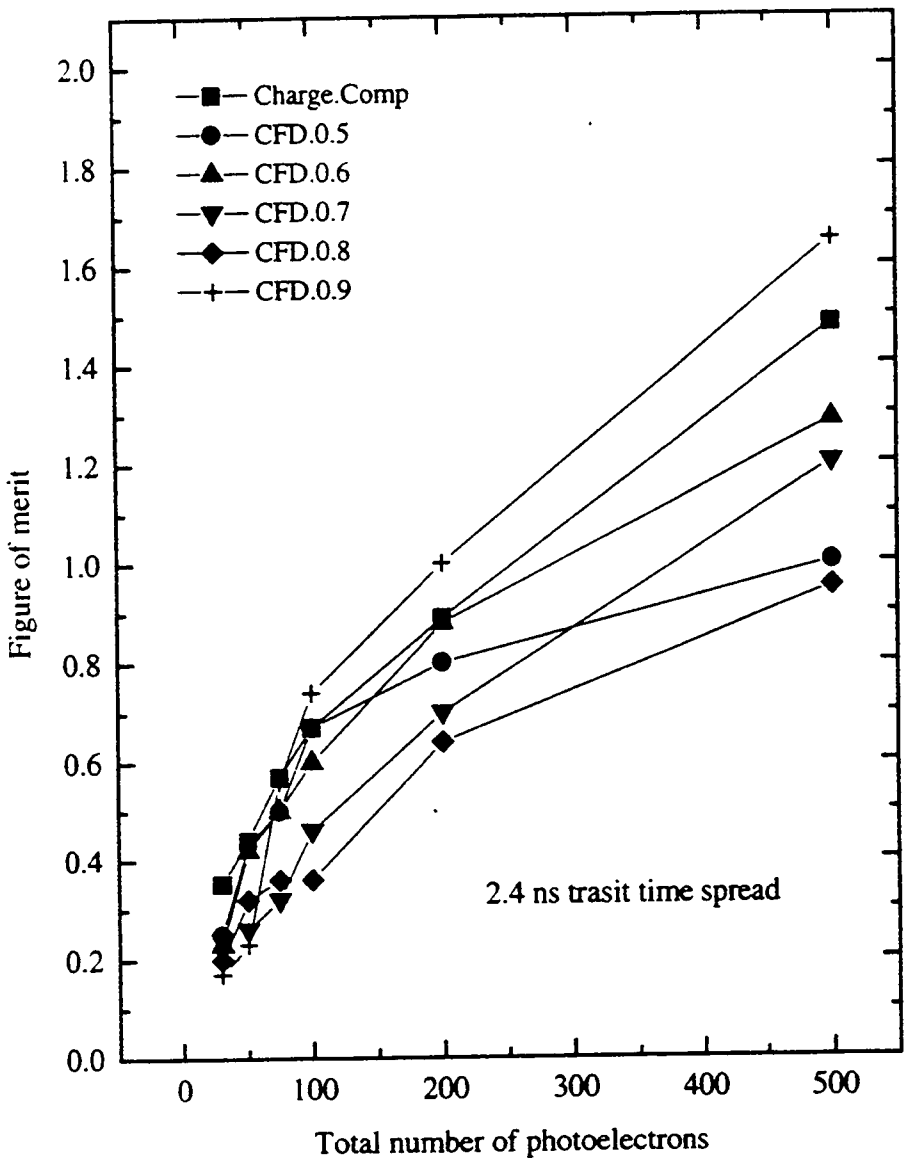


Figure 4.10 Effect of the CFD Setting on PSD Performance in the Zero-Crossing Method for the Borexino Scintilator.

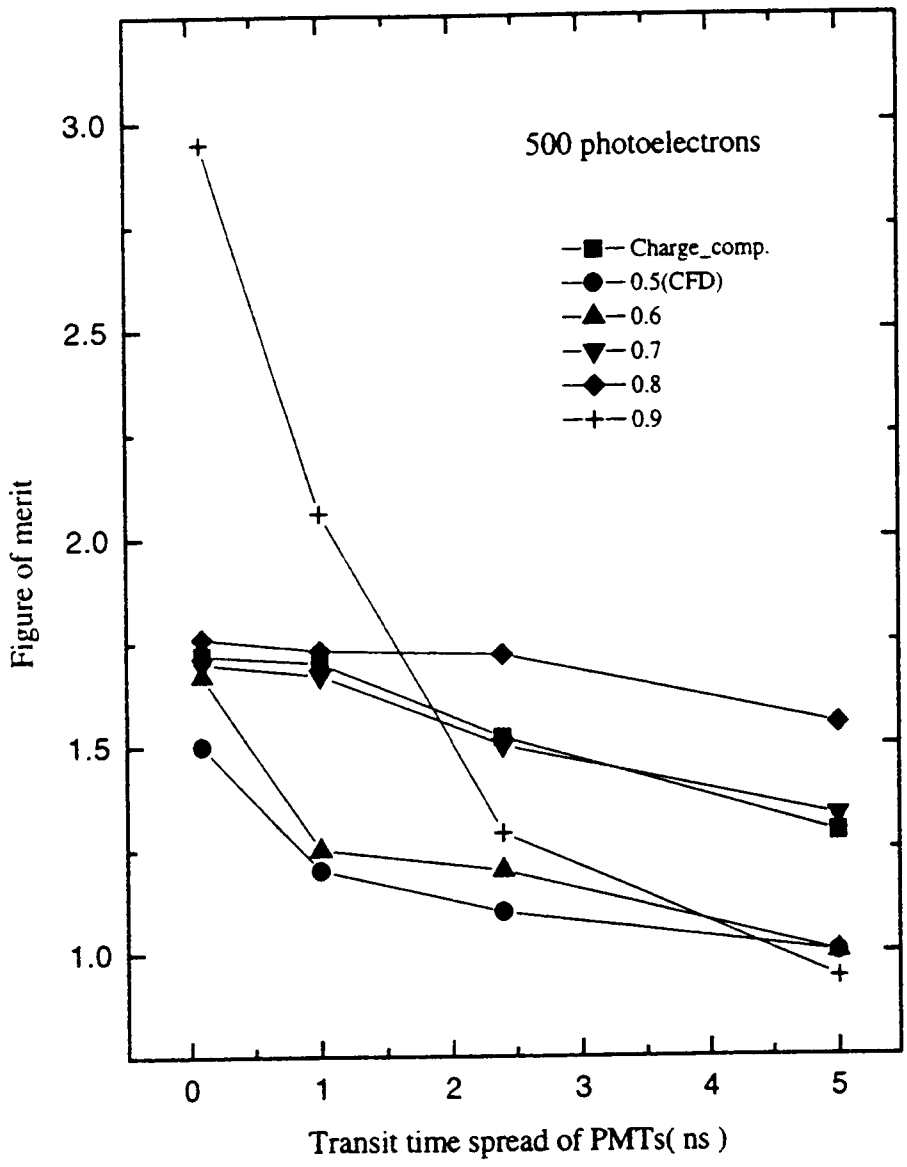


Figure 4.11 Effect of the Transit Time Spread of PMTs on PSD Performance for thr Trans-Stilbene Crystalr.

ns, which often happens in practice, especially with a large volume detectors, PSD performance degrades. This implies that the PSD performance with 0.9 CFD selection is not stable and is sensitive to the outside influences.

To optimize the integration time window in the charge comparison method, a series of FOM calculations at different window settings and at different energies for NE213, trans-stilbene and Borexino are performed. One of the results is plotted in Figure 4.12, which is for the trans-stilbene detector coupled to a XP2020 PMT at 100, 500 and 1000 photoelectrons. As can be seen from this figure, the best fast window setting is 0-22 ns for this scintillator, which is not affected by the event energy deposition. Figure 4.13 shows test results for the effect of the transit time spread of a PMT on the window setting. The transit time of the PMT varied from 0.1 to 5 ns in this calculation; however, the best fast window setting remains at 0-22 ns. It is concluded that the best fast time window setting for trans-stilbene is 0-22 ns, which is independent of the energy deposition and of PMT selection.

The same tests are performed for early NE213 and Borexino scintillators. It is found that the best fast window settings are 0-40 ns and 0-10 ns for the early NE213 and for the Borexino, respectively. It is concluded from the present work that the best time window setting only depends on scintillation material and is unique for each scintillation material.

The low energy limitation for a given detector with a 2.4 ns transit time spread is determined from the calculated results shown in Figures 4.8 - 4.10. Given a 0.5 FOM

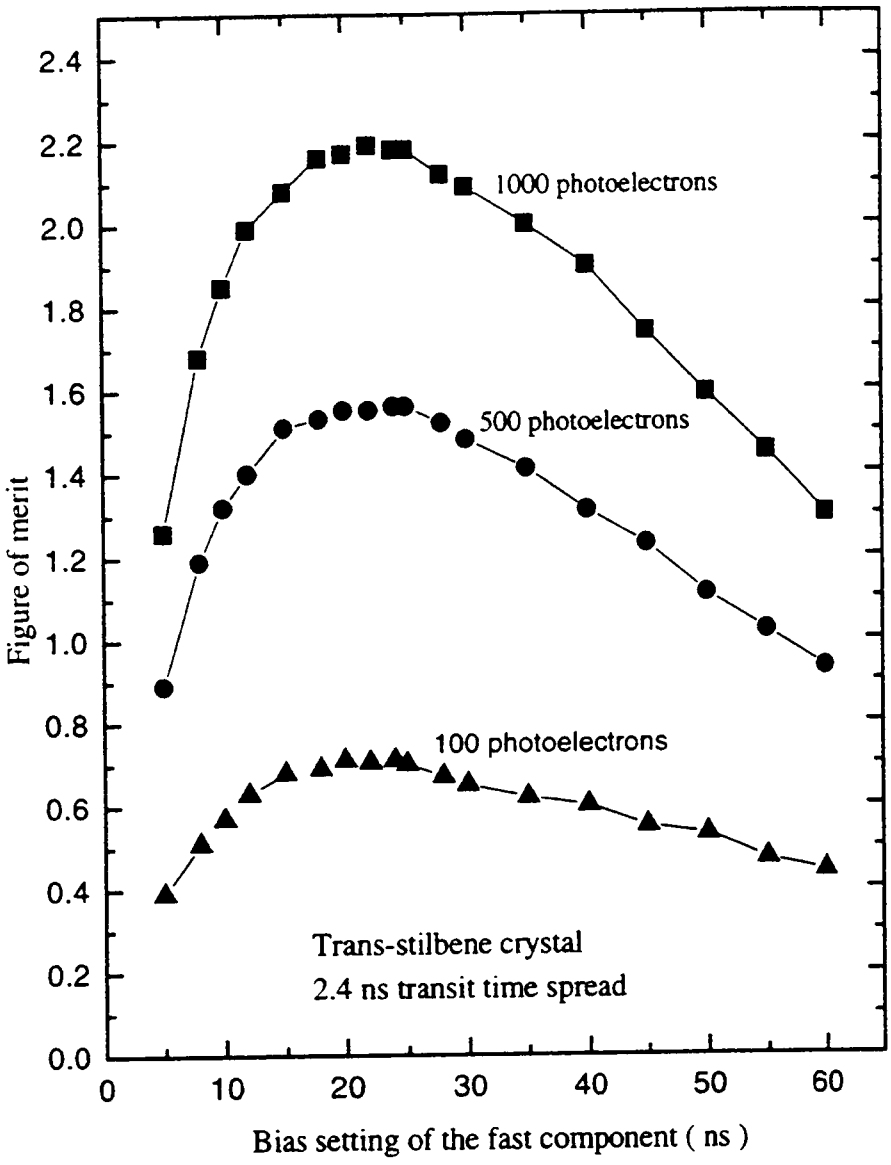


Figure 4.12 Effect of the Fast Window Location on PSD Performance.

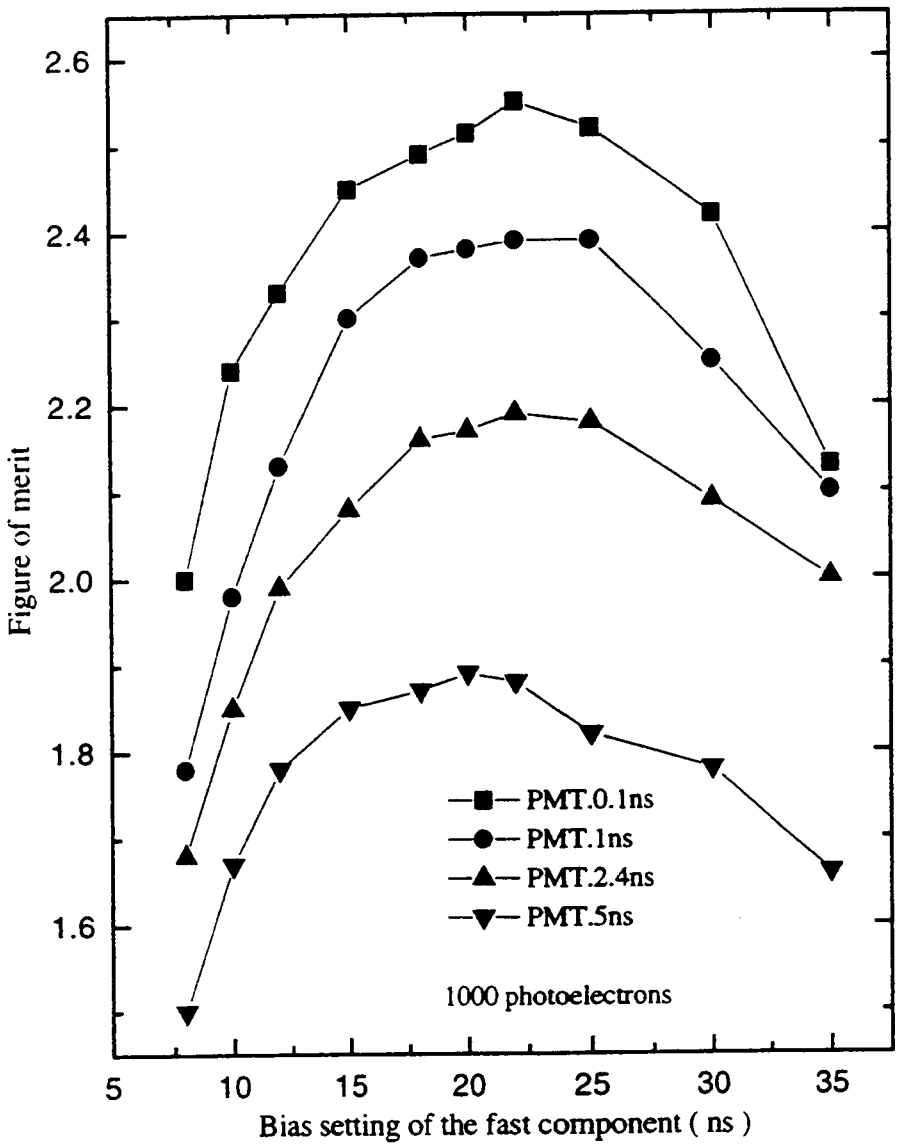


Figure 4.13 Effect of the Transit Time Spread of PMTs on the Fast Window Setting for the Trans-Stilbene Crystal.

(where the separation between neutron and photon peaks is equal to one half of the summation of FWHMs for both peaks) as the threshold for a low energy limit, the low limits for early NE213, trans-stilbene and Borexino scintillators are 100, 40 and 60 photoelectrons, respectively (which correspond to 59, 17 and 26 keV equivalent electron energy (or 260, 82 and 120 keV neutron energies), respectively). With a PMT that has a 0.1 ns transit time spread, the low energy limits are 70, 30 and 25 photoelectrons (corresponding to 180, 65 and 60 keV neutron energies) for NE213, stilbene and Borexino scintillators, respectively.

4.3.3 Effect of the PMT Transit Time Spread

In the electron multiplication process of the photomultiplier tube, each photoelectron produced by the scintillation light on the cathode travels from the cathode to the anode and is multiplied at each dynode to form a SPE pulse at the anode. Due to the statistical nature of this multiplication, the arrival times of the SPE pulses have a time distribution, which is called the transit time spread, and this affects the time distribution of the output signal. As mentioned in Section 4.2.1, the SPE pulse distribution is the convolution of the scintillation decay distribution and the transit time spread; therefore, the peak is broadened, and it has a negative effect on PSD performance. If the transit time spread is large enough, the scintillation decay time difference between neutrons and photons fades. Therefore, one of the optimization criteria is to reduce this effect. The present work performed a series of tests regarding this effect for different

scintillators and two PSD methods. The values used in the calculation are 0.1, 1, 2.4 and 5 ns. All results show that the PSD performance degrades with the increase in the transit time of the PMTs.

In the charge comparison method, this effect is significant and depends on the scintillator type. For example, the FOM value of the trans-stilbene detector for 1 ns transit time spread is 2.35, while it is only 1.85 when a 5 ns transit time spread is included. It is apparent from comparisons of the results shown in Figures 4.14 and 4.15 that this effect is less significant for early NE213 than for Borexino due to a smaller fast component of early NE213.

This effect in the zero-crossing method also depends on the CFD settings. Figure 4.14 shows that the effect of the transit time spread on the PSD performance of NE213 is dramatic for 0.9 and 0.8 CFD settings, while it is less significant for the 0.7 and 0.6 CFD values. The same phenomenon is observed in the tests for trans-stilbene; however, the dramatic effect only occurs at the 0.9 CFD setting (Figure 4.11). Figure 4.15 shows the results from the calculations for the Borexino scitillator for 100 photoelectrons. The effect of the transit time spread on the PSD performance is significant for all conditions. This means that the PSD performance of the Borexino scintillator is most sensitive to the transit time spread among the three scintillators tested due to its having the largest fast component (Table 4.1).

It is concluded that the transit time spread effect depends on scintillation properties as well as the CFD settings. The larger the fast component in a scintillator, the

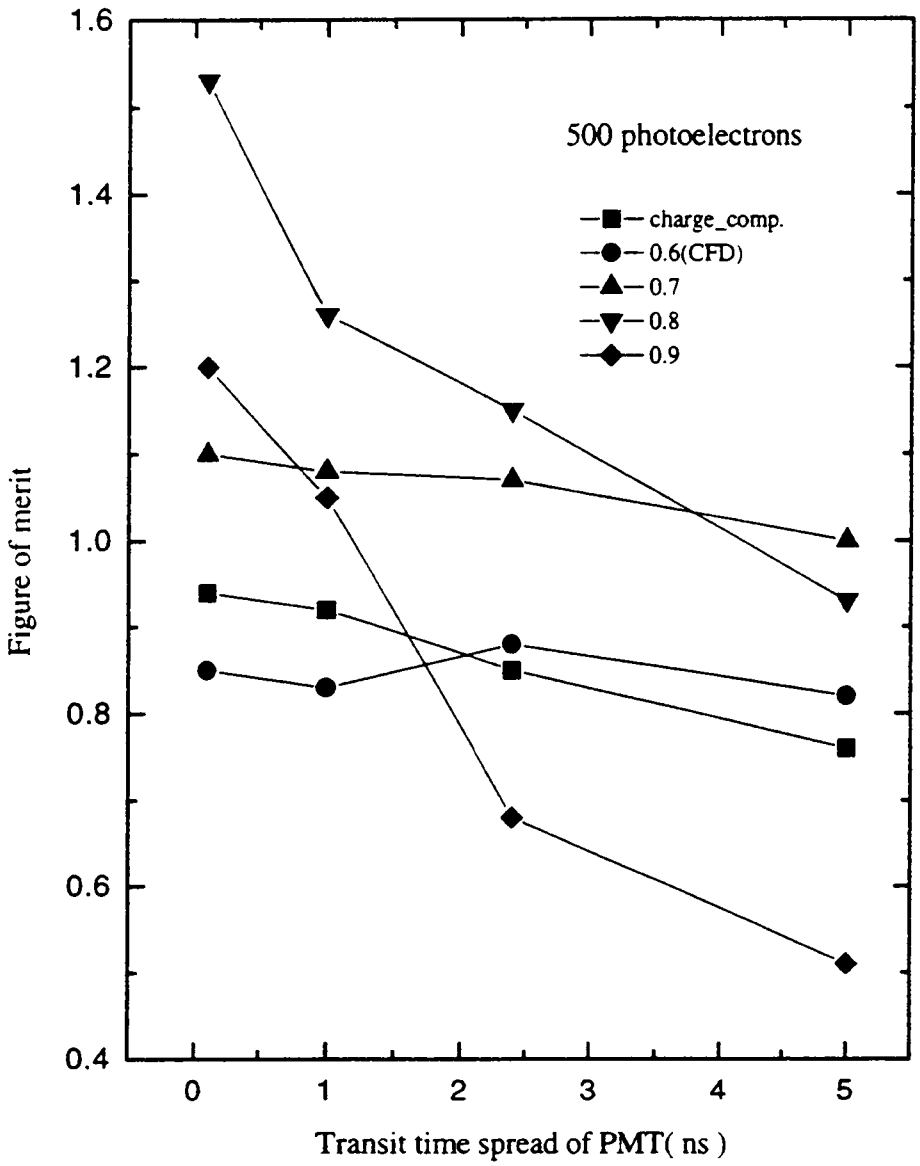


Figure 4.14 Effect of the Transit Time Spread of PMTs on PSD Performance for NE213.

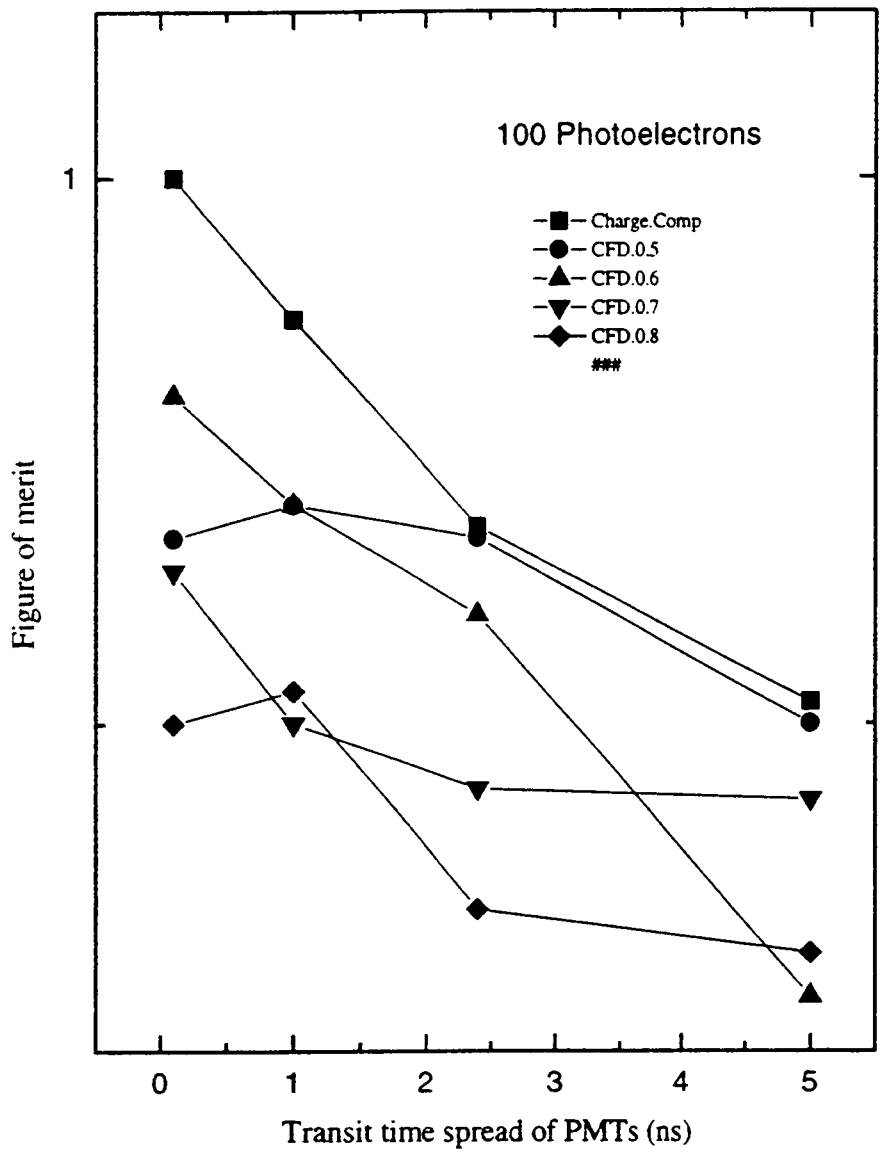


Figure 4.15 Effect of the Transit Time Spread of PMTs on PSD Performance for the Borexino Scintillator.

larger this effect. This effect is less at the 0.5-0.7 CFD settings than at the 0.8 and 0.9 CFD settings

4.3.4 Electronic Noise Contribution

The electron noise contribution to the PSD performance in the zero-crossing method is estimated by using Equation (11) in Section 4.2.2. The absolute electronics noise level is obtained from a single electron spectrum measurement for a XP2020 PMT (Figure B.5 in Appendix B.2), and its FWHM is estimated to be 0.25 photoelectron. Figures 4.16 and 4.17 show the calculated full width at half maximum ($FWHM_{noiset}$) at 0.1-0.9 CFD values for the early NE213, trans-stilbene and Borexino scintillators. The total photoelectron number is assumed to be ten and the system output is normalized to one so that the relative electronic noise becomes 0.025 (= 0.25/10), which is used as $FWHM_{noise}$ in Equation (11). The relative electronic noise level is inversely proportional to the total number of photoelectrons. For example, if the total number of photoelectrons is 100, the relative noise level will be 0.0025(= 0.25/100), and the calculated $FWHM_{noiset}$ value is 10 times smaller than the values plotted in Figure 4.16 and 4.17.

Figure 4.16 shows the electronic noise induced photon peak spread at different CFD settings. As can be seen from this figure, the smaller the CFD value, the smaller the extra spread caused by the noise. This is due to the steeper slopes of the integrated scintillation curve at the smaller CFD values (Figure 4.6). The largest noise contribution is at the 0.9 CFD setting due to the slowest pulse slope at this point (Figure 4.6). For

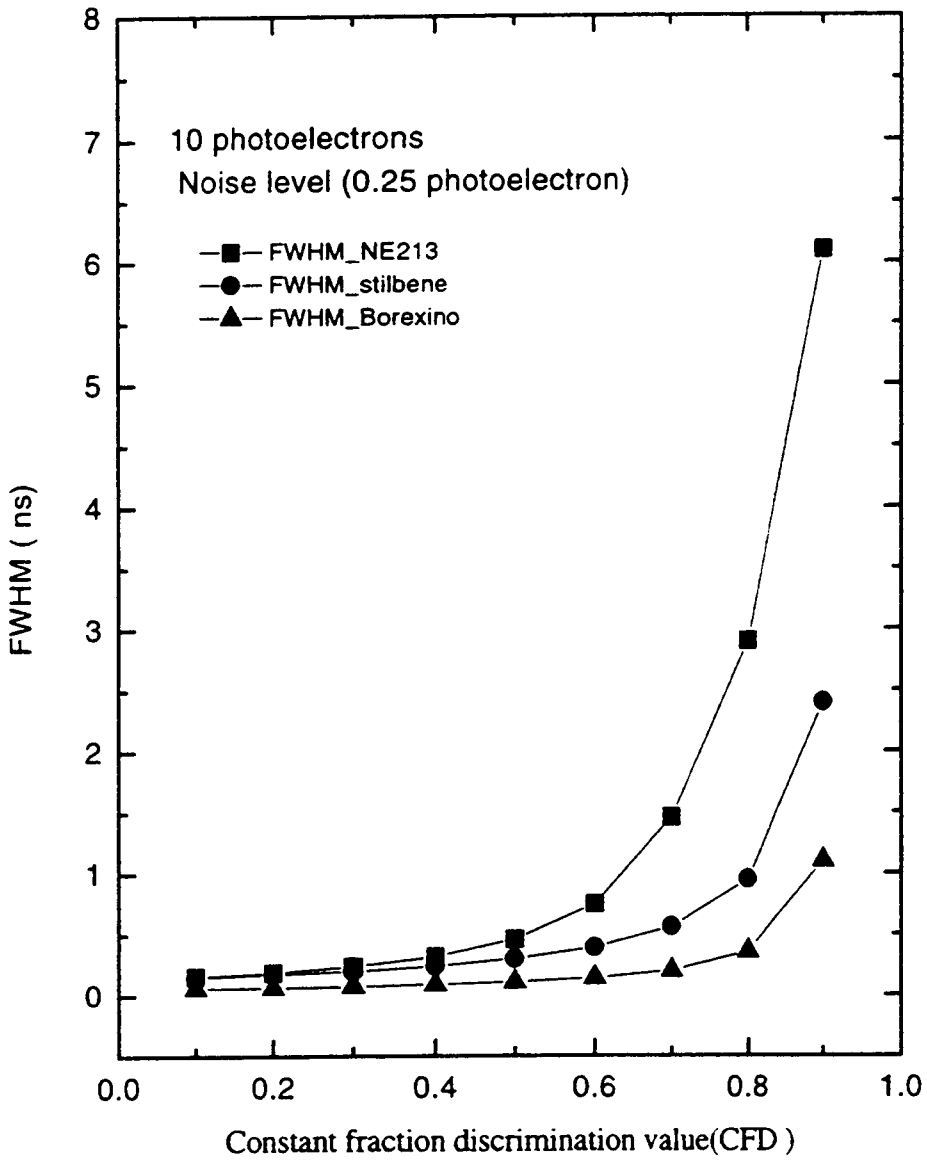


Figure 4.16 Electronics Noise Induced Photon Peak Spread in the Zero-Crossing Method.

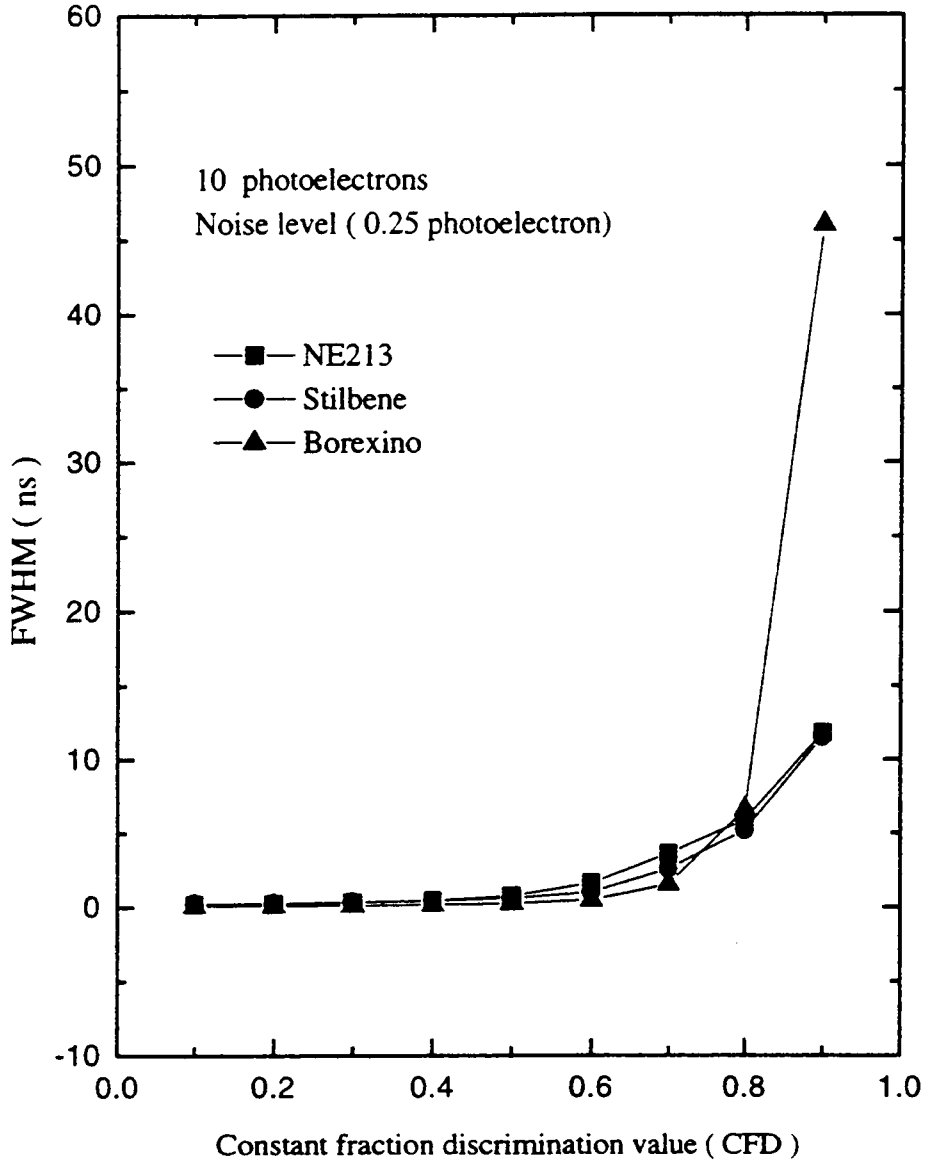


Figure 4.17 Electronics Noise Induced Neutron Peak Spread in the Zero-Crossing Method.

trans-stilbene, the $FWHM_{noiset}$ of the photon peak is 2.5 ns at the 0.9 CFD setting and 0.2 ns at the 0.6 CFD, while $FWHM_{noiset}$ of the neutron peak is 11 ns for the 0.9 CFD setting and 0.7 ns for the 0.6 CFD. This comparison shows that the noise effect on the neutron peak is stronger than that on the photon peak due to the larger slow component in the pulse produced by a neutron event. To reduce the noise contribution, the CFD value should be smaller; however, to measure the rise time, the CFD value should be larger. A compromise must be made; therefore, the values of 0.6-0.8 are chosen.

Calculations show that the noise effect in the charge comparison method is less significant than that in the zero-crossing as long as the time window is set on the flat part of the curve in Figure 4.12 and 4.13, where PSD performance is not sensitive to the window jitters. The electronic noise effect also depends on the scintillator. As shown in Figure 4.16 and 4.17, the Borexino is much more sensitive to the noise, especially at 0.9 CFD setting. This is due to the existence of its slow decay time component (399.9 ns). It is concluded that low noise electronics is vital to the zero-crossing method, especially at very low neutron energies (< 50 keV equivalent electron energy).

4.3.5 Comparison of the Zero-Crossing and the Charge Comparison Methods

As mentioned previously, which method is better for neutron-photon pulse shape discrimination remains in question (Appendix B.3.5); therefore, calculations are performed for comparison purpose, and the results are presented in Figures 4.8-4.11 and 4.14-4.15. Table 4.2 summarizes the calculated PSD performance for 200 photoelectrons

Table 4.2 Calculated Best PSD Figure of Merit for 500 Photoelectron Energy Deposition Under Various Conditions.

	0.1 ns Transit Time Spread		2.4 ns Transit Time Spread	
Scintillator	Zero-crossing	Charge comparison	Zero-crossing	Charge comparison
Early NE213	1.55	0.94	1.15	0.93
Trans-stilbene	1.30	1.74	1.74	1.55
Borexino	1.25	2.30	1.64	1.48

energy deposition under various conditions. As can be seen from this table and figures, the zero-crossing method at the optimal CFD setting will provide the best PSD performance for early NE213 and trans-stilbene cases, while the charge comparison method has upper medium PSD performance for a given system. For the Borexino scintillator, the situation is more complicated. Figure 4.18 shows results with a small transit time spread (0.1 ns). The PSD performance in the charge comparison is superior to all results from the zero-crossing method, and this is consistent with Ranucci's work[27] where a zero transit time spread is assumed. However, when the transit time spread increases to 2.4 ns, a 0.9 CFD setting in the zero-crossing method provides better results (Figure 4.10).

It is concluded that the answer to the question of which method is better strongly depends on the scintillation material. However, it still is not a simple question to answer in practice since it also depends on many factors, such as the volume of scintillators, which causes extra transit time spread, photomultiplier tubes (the transit time spread and rise time), measurement parameters (CFD or the time window), energy range of interest, and electronic noise level. It is even affected by the operator's skill for adjustment. For example, the zero-crossing point adjustment of two constant fractional discriminators at low energies is quite difficult due to the influence of noise. Therefore, for a given scintillation detector, one method may offer superior PSD performance to other under one condition, while in another, other methods may be better. This is why the conflicting conclusions about this issue are drawn from the different researchers in previous work. The Borexino case is a typical example. At 0.1 ns transit time spread, the charge

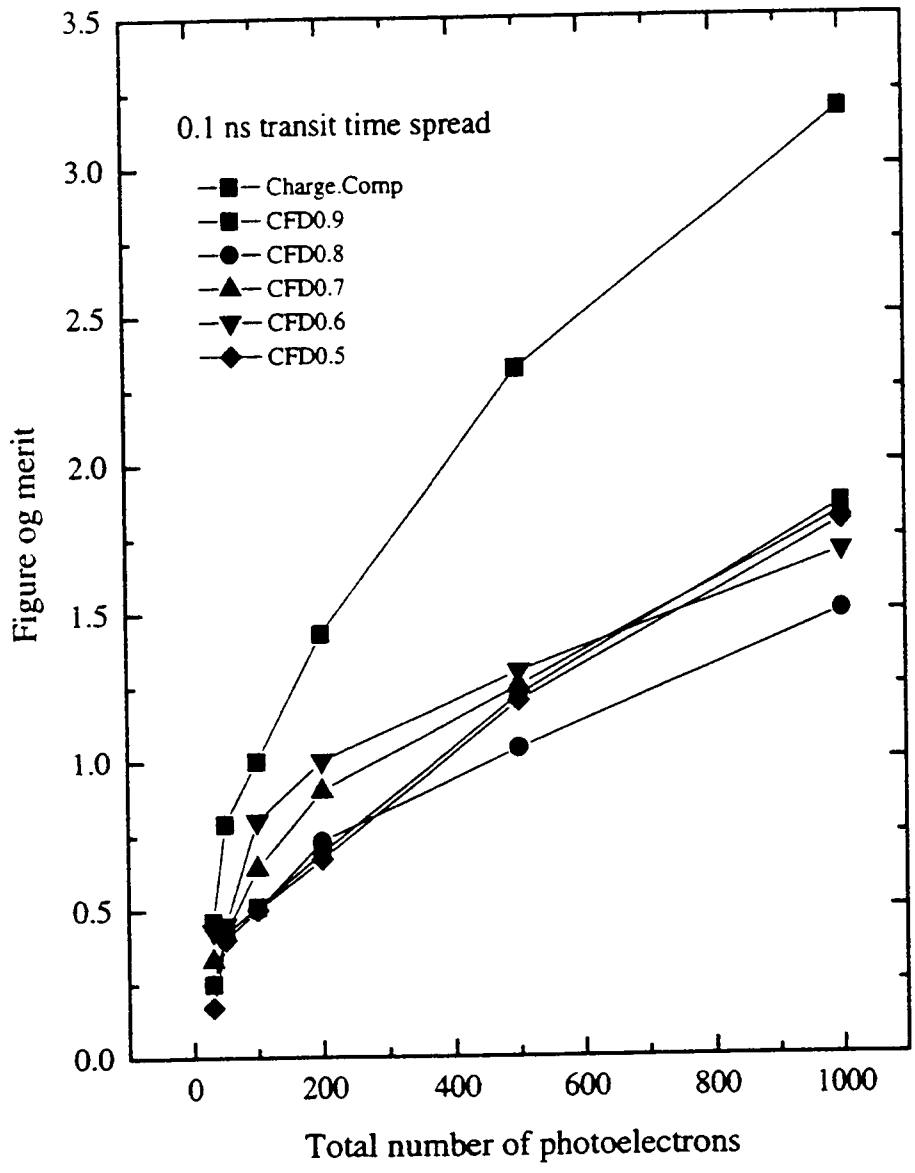


Figure 4.18 PSD Performance for Various Conditions for the Borexino scintillator.

comparison method is better than the zero-crossing, while at 2.4 ns transit time spread the zero-crossing with 0.9 CFD setting is better.

Other examples are shown in Figure 4.8 and 4.9, where under the condition of small electronic noise, and of optimal operation parameters, such as 0.8 CFD for the zero-crossing and 22 ns for the charge comparison, the zero-crossing method offers better PSD performance than the charge comparison method over a large energy range. However, the difference is not significant. Since the zero-crossing method is more sensitive to noise than the charge comparison, the performance will be close at low energies or with larger electronic noise. At other CFD settings, such as 0.7 CFD, both methods offer similar results. Compared to improper settings of the CFD value, such as 0.6 and 0.5, the charge comparison works better. Since optimization of the zero-crossing points at low energies is difficult in practice, and the optimization of the charge comparison window is relatively easy (only depending on the scintillator type) closer PSD performances will be obtained from two methods, or the charge comparison may even provide better results.

CHAPTER FIVE

CONCLUSIONS

The goal of this work is to improve the state-of-art of neutron-photon pulse shape discrimination so that it can be used for neutron-photon dosimetry. Current commercially available neutron-photon PSD equipment is not suitable for dosimetry since its low energy limitation is well above 100 keV neutron energy. In this work, PSD parameter optimization is performed not only with experiments but also analytically using model calculation. New contributions to the state-of-the-art include time-of-flight selection for the PSD calibration at low energies, dynamic bias neural network classifiers, and statistical models for prediction of the intrinsic low energy limitation of a PSD system.

To ensure that a neutron-photon PSD system is operated under optimal conditions, experimental parameter optimization is necessary. In the present work, different scintillators, photomultiplier tubes, tube bases, electronics and methods are tested and compared. Results show that the photoelectron production rate is an important property for a PSD system. A system with a larger value of photoelectron production rate will have a lower PSD energy limit. The best PSD detector tested, a 1.5" BC501A-XP2020 PMT, has 1.7/keV photoelectron production rate and its low energy limitation is about 120 keV neutron energy. To obtain higher photoelectron production rates, the diameter of the scintillator should be smaller than the diameter of the photomultiplier tube and

direct coupling between them (no light-guide) is preferable. The best CFD value of the BC501A scitntillator in the zero-crossing method is experimentally determined to be 0.7.

Since no pure neutron source without associated gamma-rays exists, the low energy cross-talk calibration is quite complicated and usually requires accelerator facilities to provide electron and proton beams. A new and simple approach, using Cf-252 neutron source and the time-of-flight selection method, is adopted in the present work to acquire "pure" neutron and "pure" gamma-ray PSD spectra. The calibration data and the neural network training data for the low energy region are constructed from these "pure" spectra at a given ratio. The time-of-flight selection method also can be applied to the single electron time sampling in the study of PSD properties of scintillators.

Dynamic bias is an efficient technique to improve PSD performance over a large energy range, especially when it covers low energies. In the present work, neural networks are used to implement dynamic bias. Three neural network-based classifiers, perceptron, back-propagation neural network and Bayes, are developed and tested. Compared with the conventional static bias method, all of them obtain much smaller classification errors over a large energy range. The perceptron classifier is the simplest one and its performance is satisfactory. The Bayes classifier has better performance for a wide range neutron source conditions. However, to reduce the classification error, the classifiers should be used at conditions close to training conditions. By using dynamic bias classifiers, the PSD low energy limitation is reduced to about 100 keV, which is a critical energy point since the neutron dose quality factor above this energy increases

rapidly. With such an improvement, the design and manufacture of neutron-photon dosimeters are feasible.

Two statistical models, which simulate the photoelectron pulse processing in the zero-crossing and the charge comparison PSD methods, are developed for evaluation of the PSD properties of scintillators and PSD methods. Transit time spread and electronic noise contributions are included in the models so that they represent real detection systems and can be used to predict the PSD performance of a given scintillation detector and associated electronics. Calculations show that these two factors significantly affect the PSD performance at low energies and should be included in the model.

The PSD performances of three scintillators (early NE213, trans-stilbene, and Borexino) are calculated with two models and their intrinsic low energy limitations with a small transit time spread (0.1 ns) are estimated to be 180, 65 and 60 keV neutron energies, respectively. Pulse shape discrimination parameter optimization is performed by using model calculation. The results show that the fast window setting in the charge comparison method depends on the scintillator property only and is not affected by the transit time spread and energy deposited. The calculated best settings for early NE213, trans-stilbene and Borexino scintillators are 0-40, 0-22, 0-10 ns, respectively. The calculated best CFD values in the zero-crossing method for early NE213 and trans-stilbene scintillators is 0.8, which is consistent with the present experimental results (0.7). The calculated best CFD value for the Borxino is 0.9.

Comparisons of the zero-crossing and the charge comparison methods are performed with model calculations. Results show that the answer to which method is better depends on scintillator type. However, it is also affected by the other factors, such as the noise level and the transit time spread of the detection system. Therefore, there is no simple conclusion to be drawn. The present calculations show that under optimal conditions the zero-crossing method is better for NE213 and trans-stilbene crystal, and the charge comparison is better for Borexino.

The present work shows that PSD low energy limitation cannot be reduced down to 50 keV neutron energy with scintillation materials considered in this work by using either the zero-crossing or charge comparison methods. The digitization method may have better performance than the conventional methods at low energies; however, no publication is available to date. It is questionable that significant improvement in PSD performance can be obtained by using the digitization method since the intrinsic low energy limit for commercial available scintillators is about 60 keV neutron energy, and with dynamic bias classifier a 100 keV neutron energy limit is obtained. The key issue to reduce PSD low energy limitation is to find better scintillation materials than those currently available. Further efforts should concentrate on research for new scintilation materials.

LIST OF REFERENCES

LIST OF REFERENCES

1. L. Varga, Pulse Shape Discrimination, Nucl. Instr. and Meth. 14 (1961) 24.
2. M.L. Roush, M.A.Wilson, et al., Pulse Shape Discrimination, Nucl. Instr. and Meth. 31 (1964) 112.
3. R.A. Winyard, J.E. Lutkin, Pulse Shape Discrimination Inorganic and Organic Scintillators. I, Nucl. Instr. and Meth. 95 (1971) 141.
4. F.D. Brook, Development of organic Scintillations (A Review of Techniques and applications), Nucl. Instr. and Meth. 162(1979) 477.
5. E. Mathieson & T.J. Harris, Pulse Shape Discrimination in Proportional Counters, Nucl. Instr. and Meth. 88 (1970) 181.
6. R.C. Hochel, D.W.Hayes, Background Reduction and Noise Discrimination in the Proportional Counting of Tritium Using Pulse-Shape Analysis, Nucl. Instr. and Meth. 130 (1975) 183.
7. M. Moszynski and B. Bengtson, Application of a Pulse Shape Selection Method to a True Coaxial Ge(Li) Detector for Measurement of Nanoseconds Half-Lives, Nucl. Instr. and Meth. 80 (1970) 233.
8. G. Pausch, M. Moszynski, et al., Application of the Pulse Shape Technique to Proton-Alpha Discrimination in Si-Detector Array, Nucl. Instr. and Meth. A365 (1995) 176.
9. E.G.Shapiro and A.L. Anderson, Dual Energy Analysis Using Phoswich Scintillation Detectors for Low Level, In-Vivo Counting, IEEE Trans.

Nucl.Sci.,Ns-21, No.1,(1974)201.

10. F.T. Kuchnir & F.J. Lynch, Time Dependence of Scintillations and the Effect on Pulse Shape Discrimination, IEEE Trns. Nucl.Sci., NS-15, No.3, 107(1968).

11. G. Jones, P.H.R. Orth, Time Resolution and Pulse Shapes in Zero Crossover Timing, Nucl. Instr. and Meth. 59(1968) 309.

12. P. Betz, G. Braun, et al., A Fast Neutron-Gamma Discriminator, Nucl. Instr. and Meth. 119 (1974) 109.

13. J. Bialkowski, J. Szczepankowski, Apulse Shape Discrimination Circuit for n and γ Multidetector Measurements, Nucl. Instr. and Meth. 152(1978) 589.

14. J. Bialkowski & J. Szczepankowski, A Pulse Shape Discrimination Circuit for Neutron and Gamma Multidetector Measurement, Nucl. Instr. and Meth. 152 (1978) 589.

15. L.J. Perkins, M.C. Scott, The Application of Pulse Shape Discrimination in NE213 to Neutron Spectrometry, Nucl. Instr. and Meth. 166(1979) 451.

16. J. Bialkowski, M. Moszynski, et al., A Pulse Shape Discriminator with high Precision of Neutron and Gamma Ray Selection at High Counting Rate, Nucl. Instr. and Meth. A275 (1989) 322.

17. D. Bazin, A.C. Mueller, et al., A NE213 Liquid Scintillator Neutron Detector Designed for Lifetime Measurements of Very Neutron-Rich Nuclei, Nucl. Instr. and Meth. A281 (1989) 117.

18. D. Wolski, M. Moszynski, et al., Comparison of n-γ Discrimination by Zero-Crossing and Digital Charge Comparison Methods, Nucl. Instr. and Meth.

A360 (1995)584.

19. G. Ranucci, Time Statistics of the Photoelectron Emission Process in Scintillation Counters, Nucl. Instr. and Meth. A335 (1993) 121.

20. B. Sabbah, A. Suhami, et al, An Accurate Pulse Shape Discriminator for a Wide Range of Energies, Nucl. Instr. and Meth. 58(1968) 102.

21. A. Bertin, A. Vitale, et al., A System of Large Liquid Scintillation Counters Used with a Simplified Neutron-Gamma Discrimination Technique, Nucl. Instr. and Meth. 68 (1969) 24.

22. R. W.St. Onge, J.A. Lockwood, A Simple High Resolution Pulse Shape Discriminator, Nucl. Instr. and Meth.69(1969) 25.

23. L.J. Heistek, L.Van Der Zwan, Pulse Shape Discrimination with a Comparator Circuit, Nucl. Instr. and Meth. 80(1970) 213.

24. J.M. Adams, G. White, A versatile Pulse Shape Discriminator for Charged Particle Separation and Its Application to Fast Neutron Time-of-Flight Spectroscopy, Nucl. Instr. and Meth. 156 (1978) 459.

25. M. Omori, Pulse Shape Discriminator Application to Low Level Counting Experiments, Nucl. Instr. and Meth. A335 (1993) 553.

26. M. Moszynski, G.J. Costa, et al., Study of n-γ Discrimination with NE213 and BC501A Liquid Scintillators of Different Size, Nucl. Instr. and Meth. A350 (1994)226.

27. G. Ranucci, An Analytical Approach to the Pulse Shape Discrimination Properties of Scintillators, Nucl. Instr. and Meth. A354 (1995) 389.

28. R.B. Owen, Pulse Shape Discrimination - a Survey of Current Techniques, IRE Ns-9, No.2, (1962) 285.

29. T. Hiramoto, N. Nohara, Neutron-Gamma Discriminating Detector for Use with Time-of-Flight Fast Neutron Spectrometer. Nucl. Instr. and Meth. 58(1968) 167.

30. J.D. Dewendra, R.B. Galloway, A Simple Fast Neutron Detector with Pulse Shape Discrimination against Gamma Rays, Nucl. Instr. and Meth. 125 (1975) 503.

31. S. Green, R. Koohi-Fraegh, et al., A Gated Single Parameter Data Acquisition System for Fast Neutron Spectroscopy with NE213, Nucl. Instr. and Meth. A317(1992) 399.

32. J.A. Miehe and G. Ambard, Statistics and Timing of the First Photoelectron with a high Gain Dynode Photomultiplier, IEEE Trans. Nucl.Sci.,NS-17, No.3, 115(1976).

33. M. Moszynski, Light Pulse Shape Study from Crystal Organic Scintillors, Nucl. Instr. and Meth. 153 (1978) 439.

34. P. Harinar, H. Chen, et al., Scintillation Decays in a Trans-Stillbene Crystal, Nucl. Instr. and Meth. A336 (1993) 176.

35. W.H. Miller, et al., Utilization of a GaGe CompuScope 1012 Computer-Based Oscilloscope for PSD and Radiation Spectroscopy, Nucl. Instr. and Meth. A353(1994) 254.

36. R.A. Cecil, et al., Improved Predictions of Neutron Detection Efficiency for Hydrocarbon Scintillators from 1 MeV to about 300 MeV, Nucl. Instr. and Meth. 161(1979)439.

37. Instruction Manual TC 5020 Oxford Instruments Inc. Nuclear Measurements

Group.

38. L. M. Bollinger, G. E. Thomas, Measurement of the Time Dependence of Scintillation Intensity by a Delayed-Coincidence Method, Rev. Sci. Instr. 32 (1961) 1044.

39. G. Ranucci, P. Ullucci, et.al., Scintillation Decay Time and Pulse Shape Discrimination of Binary Organic Liquid Scintillators for the Borexino Detector, Nucl. Instr. and Meth. A 350 (1994) 338.

40. J. T. Tou, R. C. Conzalez, Pattern Recognition Principles, 1974 Addison Wesley Publishing Company, Inc.

41. R. K. Miller, Neural Networks, 1990 Fairmont Press, distributed by Prentice-Hall.

42. L. H. Lsoukalas, and R. E. Uhrig, Fuzzy and Neural Appoaches in Engineering, 1997 John Wiley & Sons.

43. J. Giarratano and G. Riley, Expert Systems, Principles and Programming, 1994, PWS Publishing Company, Boston.

44. R. E. Walpole and R.H. Myers, Probability and Statistics for Engineers and Scientists, 1993 Macmillan Publishing Company.

45. P. G. Stolin, et. al., The Scintillation Decay of Some Commercial Organic Scintillators, Nucl. Instr. and Meth. 37 (1965) 45.

46. P. Harihar and H. Chen, Scintillation Decays in a Trans-Stilbebe Crystal, Nucl. Instr. and Meth. A336 (1993) 176.

APPENDIXES

APPENDIX A.

Signal Processing in Pulse Shape Discrimination

In energy related measurements, the output from a PMT passes through a series of manipulations, which include integration and differentiation to shape the pulse suitable for MCA recording, noise reduction, pile-up rejection and so on. No matter which electronics is used, the original energy information about events must be retained. There is a similar requirement in time related measurements. The PSD system usually identifies the radiation type by determining the decay time of the scintillation light. The scintillation light pulses must be converted into electrical signals that retain the time information about events. There are two approaches to accomplish this objective: the current mode with a smaller (<5 ns) RC time constant of the PMT anode, and the voltage mode with a larger (>5 μs) RC time constant. Figure A.1 shows schematically the pulse shapes in the two different methods. It can be seen that the decay time information is contained in the tail of signals for the current mode and in the leading edge for the voltage mode.

The shape of the pulse produced at the anode of a PMT following an event depends on the time constant of the anode circuit. If the time constant is chosen to be much larger than the decay time constant of scintillation, each pulse is integrated by the anode circuit and produces a voltage pulse whose amplitude is equal to charge over capacitance (Q/C). This situation is usually chosen for high pulse height resolution and at

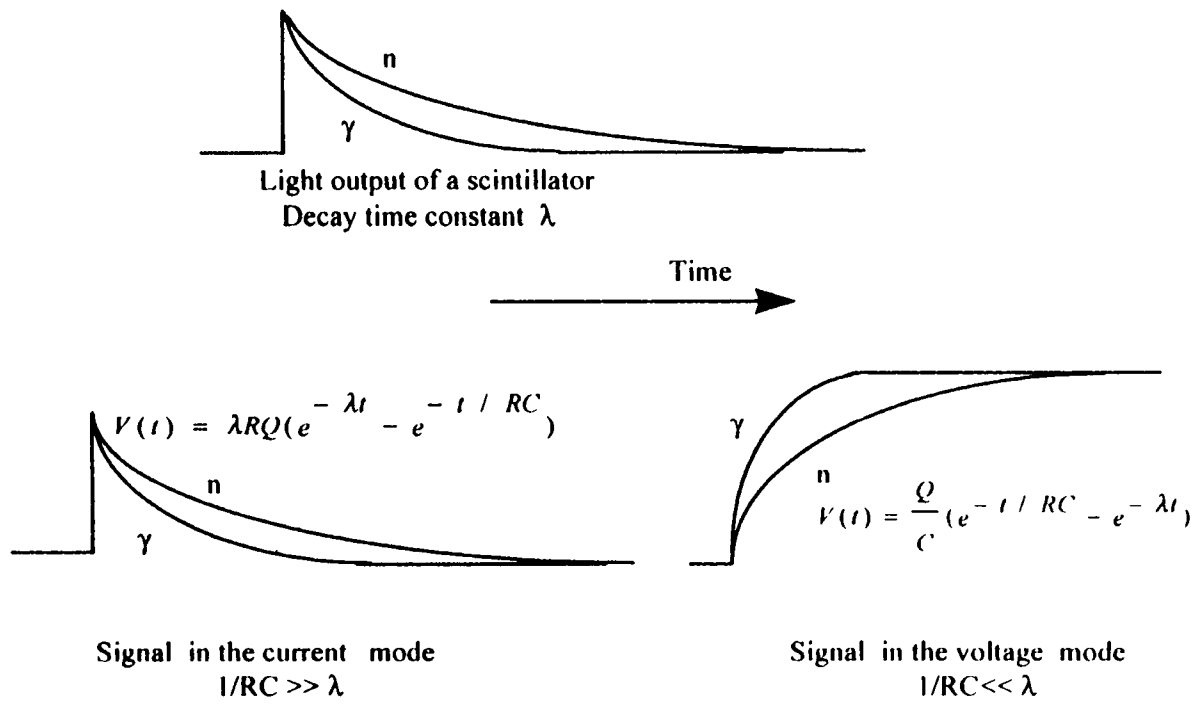


Figure A.1 Pulse Shapes of Two Different Signal Processing

low count rates. If the time constant is chosen to be much smaller than the scintillation decay time, a much faster pulse is produced, which is often used in fast timing applications or for high count rate.

An anode equivalent circuit is shown in Figure A.2 (a), where C represents the capacitance of the anode itself plus capacitance of the connecting cable and of input capacitance of the circuit connected to the anode, and R is a physical resistor between the anode and the ground or, if none is provided, the input impedance of the connected circuit. If the transit time spread of a PMT is small compared with the decay time of scintillation, the electron current i(t) that flows into the anode is

$$i\ (t) = i_0 e^{-\lambda t},$$

where λ is the scintillation decay time constant. The initial current i_0 is expressed in terms of the total charge, Q, over the entire pulse and is given by

$$Q = \int_0^\infty i\ (t)dt = \frac{i_0}{\lambda}.$$

Therefore,

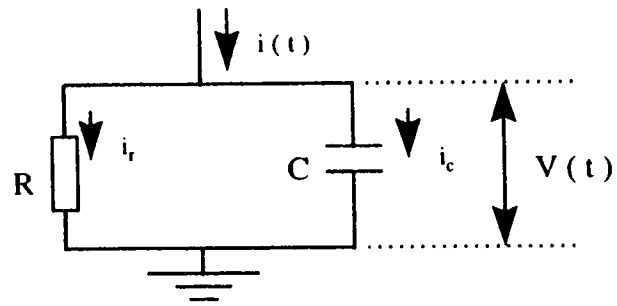
$$i_0 = \lambda Q$$

and

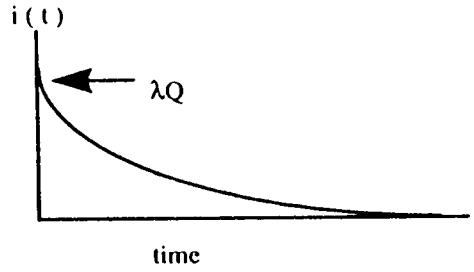
$$i\ (t) = \lambda Q e^{-\lambda t}.$$

To derive the voltage pulse V(t) at the anode, we use equations

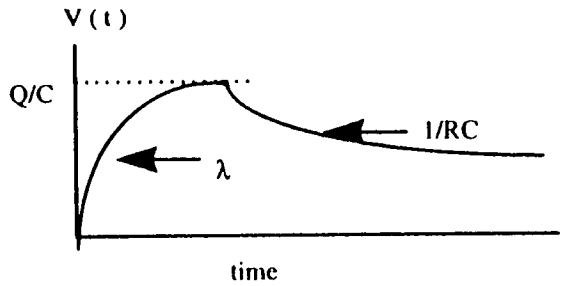
$$i\ (t) = i_R + i_C$$



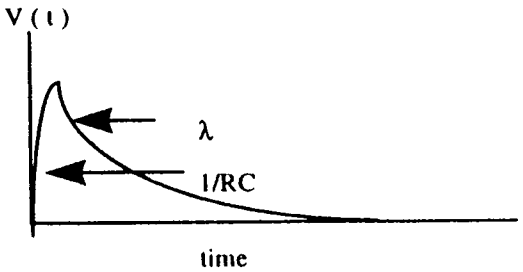
(a) Equivalent circuit of an anode



(b) Light output of scintillator



(c) Voltage mode , 1/RC << λ



(d) Current mode , 1/RC >> λ

Figure A.2 Equivalent RC Circuit of a PMT Anode and Its Input and Output Signals

$$i(t) = C\frac{dV(t)}{dt} + \frac{V(t)}{R}$$

and obtain

$$\frac{dV(t)}{dt} + \frac{1}{RC}V(t) = \frac{\lambda Q}{C}e^{-\lambda t}$$

The solution to this first-order inhomogeneous differential equation with initial condition v(0) = 0 is

$$V(t) = \frac{1}{\lambda - \lambda_{RC}}\frac{\lambda Q}{C}(e^{-\lambda_{RC}t} - e^{-\lambda t}) \qquad (1)$$

where λ_{RC} is the anode time constant.

In the voltage mode, the anode time constant is made much larger than the decay scintillation time, $\lambda_{RC} < \lambda$ and equation (1) can be approximated by

$$V(t) = \frac{Q}{C}(e^{-\lambda_{RC}t} - e^{-\lambda t}) \qquad (2)$$

A plot of this pulse shape is shown in Figure A.2(c). If $\lambda_{RC} << \lambda$, the first exponential in Equation (2) decay slowly and the short time behavior is approximately

$$V(t) = \frac{Q}{C}(1 - e^{-\lambda t}) ,$$

for $t << 1/\lambda_{RC}$. After a sufficiently long time the second exponential almost decays to zero, and the long time behavior is determined by the first exponential and is given by

$$V(t) = \frac{Q}{C}e^{-\lambda_{RC}t} , \qquad (3)$$

for $t >> 1/\lambda$. The pulse shape in the voltage mode is shown in Figure A.2(c). It can be seen that the decay time information is contained in the leading edge of the signal. The

rise time is determined by the decay time constant of the scintillation events and can be measured with the zero-crossing technique.

In the current mode, the anode time constant is set at a small value compared with the scintillator decay time ($\lambda_{RC} >> \lambda$), Now equation (1) becomes

$$V(t) = \frac{\lambda}{\lambda_{RC}} \frac{Q}{C} (e^{-\lambda t} - e^{-\lambda_{RC} t}) . \qquad (4)$$

The pulse shape is plotted in Figure A.2(b). The behavior at small values of t is now approximately,

$$V(t) = \frac{\lambda}{\lambda_{RC}} \frac{Q}{C} (1 - e^{-\lambda_{RC} t}) , \qquad (5)$$

for $t << 1/\lambda$, whereas for a large t,

$$V(t) = \frac{\lambda}{\lambda_{RC}} \frac{Q}{C} e^{-\lambda t} , \qquad (6)$$

for $t >> 1/\lambda_{RC}$.

As can be seen from the Figure A.2(d), the leading edge of the pulse is determined by the anode time constant and the tail of the pulse has the time behavior $\exp(-\lambda t)$, which is identical to that of the scintillation light. In the current mode, an integrated charge comparison electronics (such as TC-5020) is adopted to pick up the decay time information contained in the tail of a pulse and to identify the type of events.

Appendix B

Evaluation of Commercial Available PSD Equipment

B.1 Current Status of Neutron-Photon Pulse Shape Discrimination

Neutron-photon PSD has been used in neutron and gamma-ray measurements for many years. Organic sintillation materials, such as stillbene, NE213 and BC501A, are offen selected for this application because of their neutron-photon PSD properties which are characterized by a signoficant difference in ratio of the fast component to the total light output response to different types of radiation. There are two trends in the development of the neutron-photon PSD technique in recent years. One is to study the PSD property of large volume liquid scintillation detectors which are designed for neutrino measurements or high energy physics experiments. Another is to improve neutron-proton discrimination at low energies. The main effects of large volume detectors on PSD performance are neutron multiple-scattering and the effectiveness of light collection, while the main problems at low energies are statistical limitations and electronic noise. Neutron-photon PSD performance begins to deteriorate at a few hundred keV neutron energy, and it fails to discriminate events from neutrons and gamma-rays below 50 keV equivalent electron energy, which is essential for neutron dosimetry. The motivation of parameter optimization for PSD is to evaluate PSD parameters at low energies using commercially available PSD equipment.

B.2 Description of the Experimental Arrangement

The performance of neutron-photon pulse shape discrimination is affected by many parameters as well as methods. It is important to optimize these operational parameters and to select appropriate methods for particular experimental need. The zero-crossing and charge comparison methods are studied in this work. The most important factors affecting PSD performance are as follows:

1) organic scintillator type and size,

2) light coupling between a scintillator and a PMT,

3) photomultiplier tubes(PMT) and tube bases, and

4) circuits, signal processing, such as CFD values and integration time.

A example of an experimental block diagram for the zero-crossing method is shown in Figure B.1. A thick lead plate (3 cm) is used to reduce the gamma-ray count rate. The 2 mm thick Li glass scintillator is added for detection of thermal neutrons. The intensity of the Pu-Be neutron source used is about 1.2×10^6 neutrons / second.

Figure B.2 is a block diagram for the charge comparison method using a TC5020 pulse shape analyzer. The detector is operated in the pulse current mode (Appendix A), and the anode is floated above the ground and directly connected to an outside charge integrator with a low input impedance (51 Ω). There are TTL (logical signal output) output terminals to separate neutron and gamma-ray counts as well as analog outputs of events in a TC5020 analyzer; therefore, the total energy spectrum and number of neutron

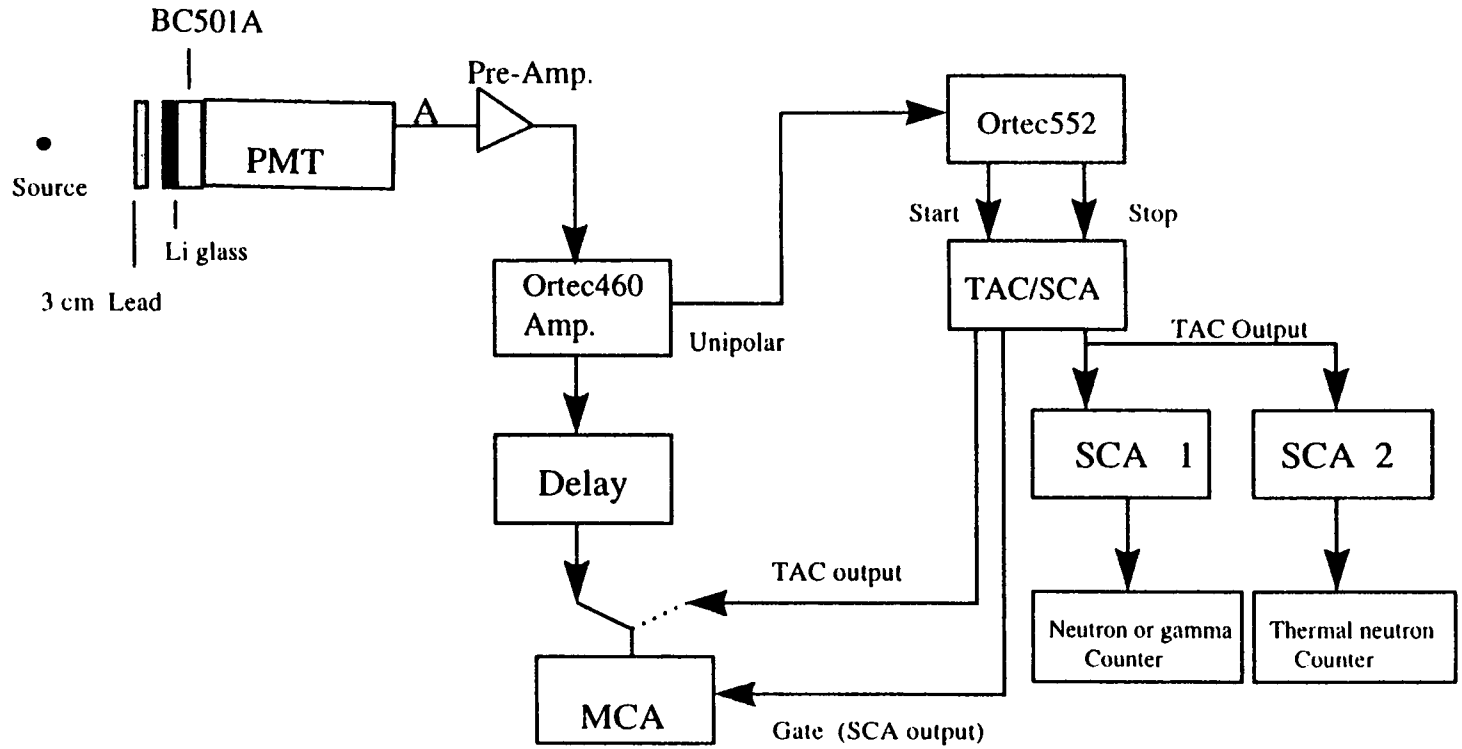


Figure B.1 Block Diagram for Thermal Neutron-Neutron-Photon PSD Experiments

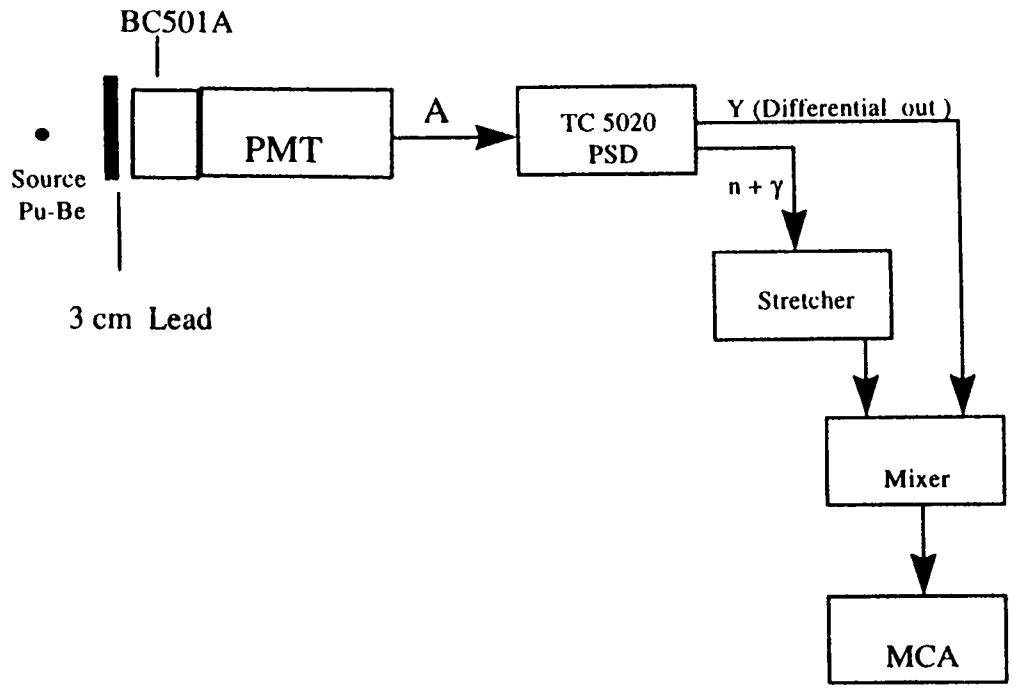


Figure B.2. Block Diagram for Neutron-Photon PSD Experiments with TC 5020

and gamma-ray events can be obtained directly from this unit. Figure B.3 illustrates an oscilloscope display of the charge comparison from a TC-5020 analyzer. Note that the output of the differential amplifier of TC-5020 is positive for photons and negative for neutrons[37], which cannot be accepted by a MCA. To measure the PSD spectrum with a TC 5020 pulse shape analyzer, post-pulse processing (stretching and addition) is performed. The electronic block diagram and the wave-form of the signal for the post-pulse processing are shown in Figures B.2 and B.4. The output (Y) of the differential amplifier in TC-5020 (which is proportional to the voltage difference between the outputs of two integrators and usually for oscilloscope monitoring) is added to the stretched $n+\gamma$ output and then measured by a MCA.

Since the photoelectron production rate (PPR) characterizes the PSD performance of a detector at low energies, where the number of photoelectrons is quite small and statistical fluctuation dominates the PSD behavior, it is an important property to be measured. Four steps are needed to measure the photoelectron production rate. First, measure the single electron energy spectrum and mark the position of the single electron peak (SEP) caused by the single electron emission from the cathode of a PMT. Second, measure the ^{241}Am energy spectrum (or use 32.5 keV peak in the ^{137}Cs energy spectrum) with the same detector and mark the full energy peak position of ^{241}Am. Third, carefully calibrate the gain correction factor between above two steps with a precision pulser, and calculate the peak position of ^{241}Am with a gain correction factor. Fourth, determine the photoelectron production rate by using the following expression:

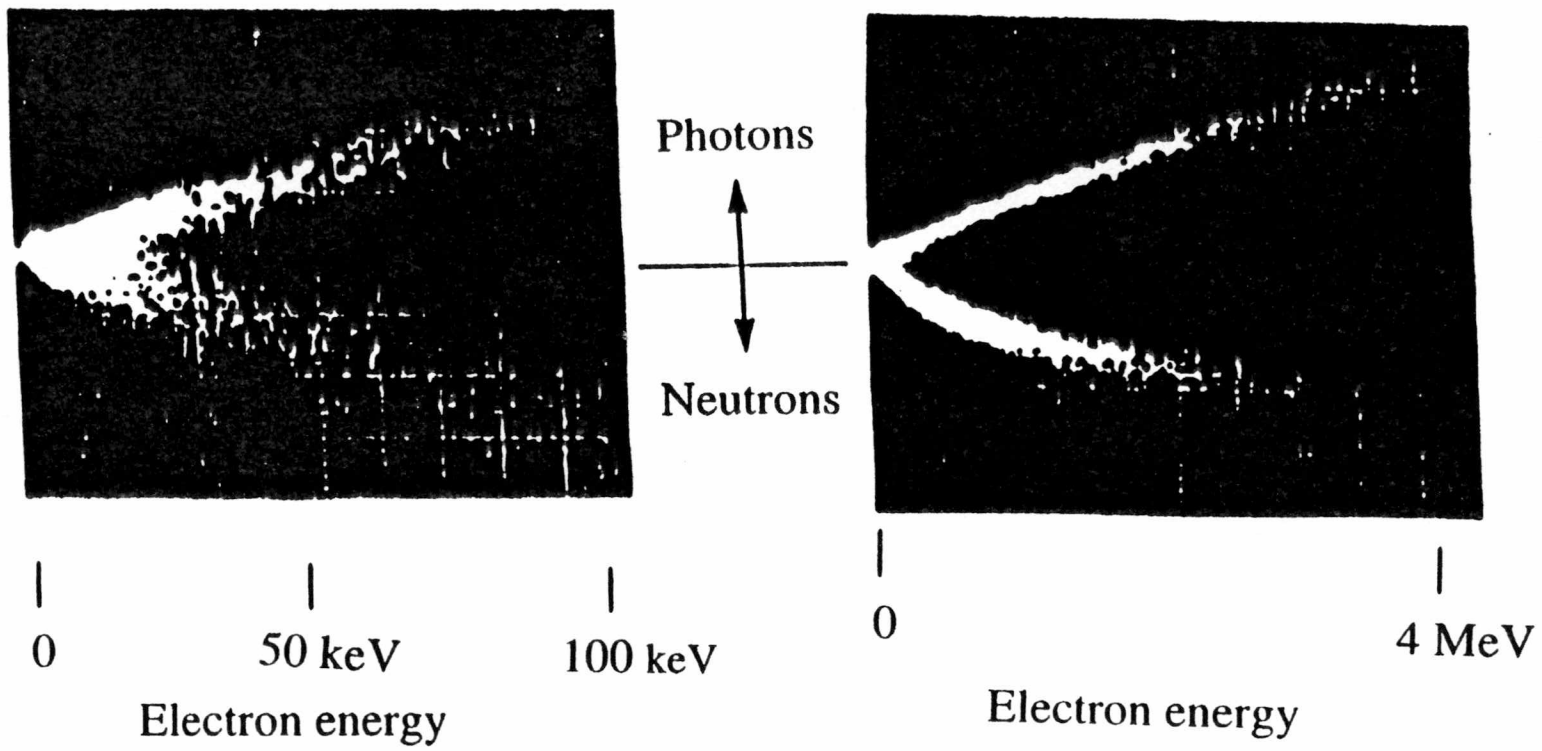


Figure B.3 Oscilloscope Display of Charge Comparison.

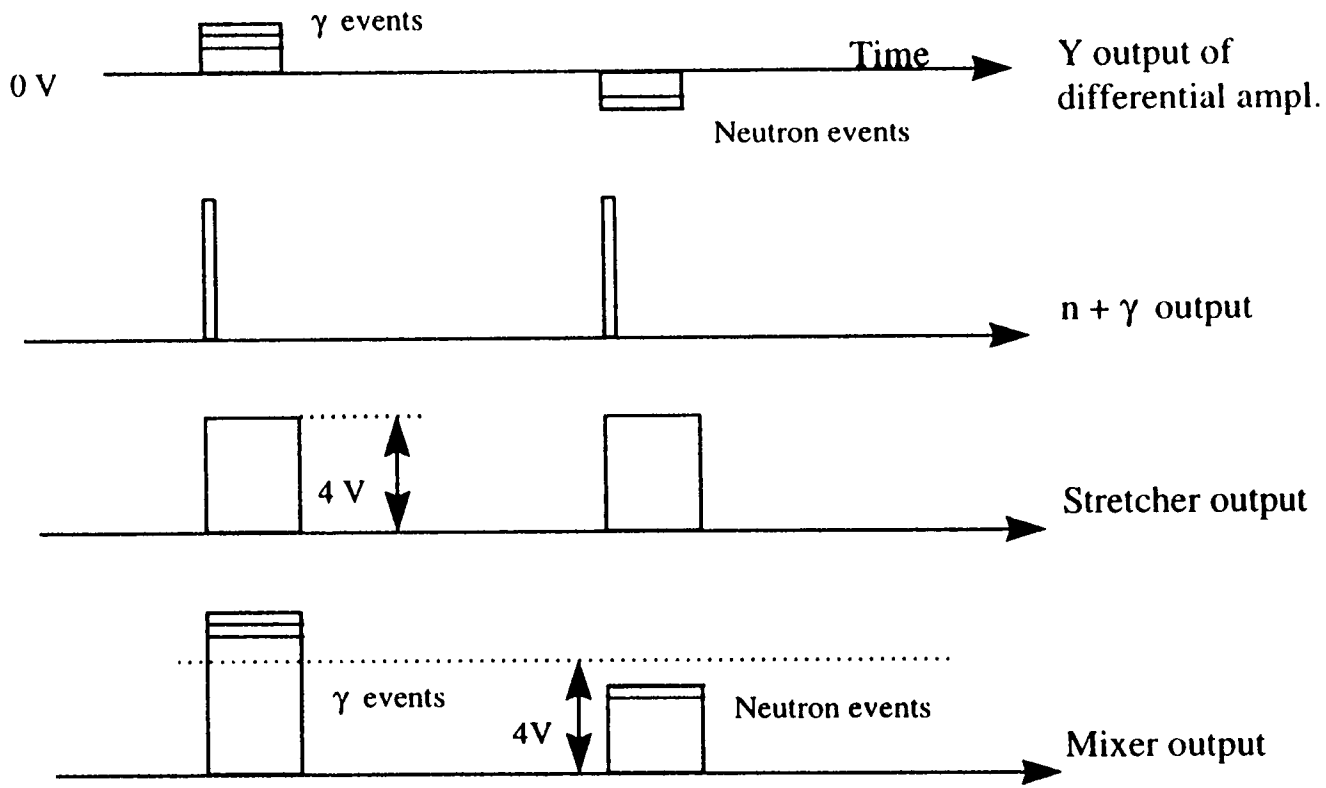


Figure B.4 Wave Form of Signals Processing for the Output of TC-5020

PPR= corrected peak position/ (SEP 59.5 keV).

Note that 59.5 keV should be replaced by 32.5 keV (X-rays from ^{137}Ba), if 32.5 keV peak in ^{137}Cs energy spectrum is used for calibration.

Figure B.5 and B.6 illustrate samples for the measured ^{241}Am and single photoelectron spectra which were acquired with a 1.5" BC501A -XP2020 PMT detector.

B.3 Results and Discussions

The PSD parameter optimization for commercial available equipment is performed experimentally to improve PSD performance at low energies (<500 keV neutron energy).

B.3.1 Scintillators and Light Coupling

The liquid scintillator, BC501A, is selected in the present work due to its higher light output and good PSD properties[26] (and due to the fact that it was available!). Different sizes (10, 15 mm, 1.5", and 2") BC501A liquid scintillators as well as a 1" BC501 scintillator were used for testing.

Figure B.7 is a rise time spectrum from a Li-glass-BC501 detector based on a 120 keV equivalent electron energy threshold. It can been seen from this figure that the thermal neutron events are widely separated from the fast neutron and gamma-rays and that the latter two are difficult to identify with BC501 at 120 keV equivalent electron energy (see Appendix C for equivalent electron energy). If BC501 is replaced with a BC501A scintillator, much better performance is obtained. As can be seen from Table

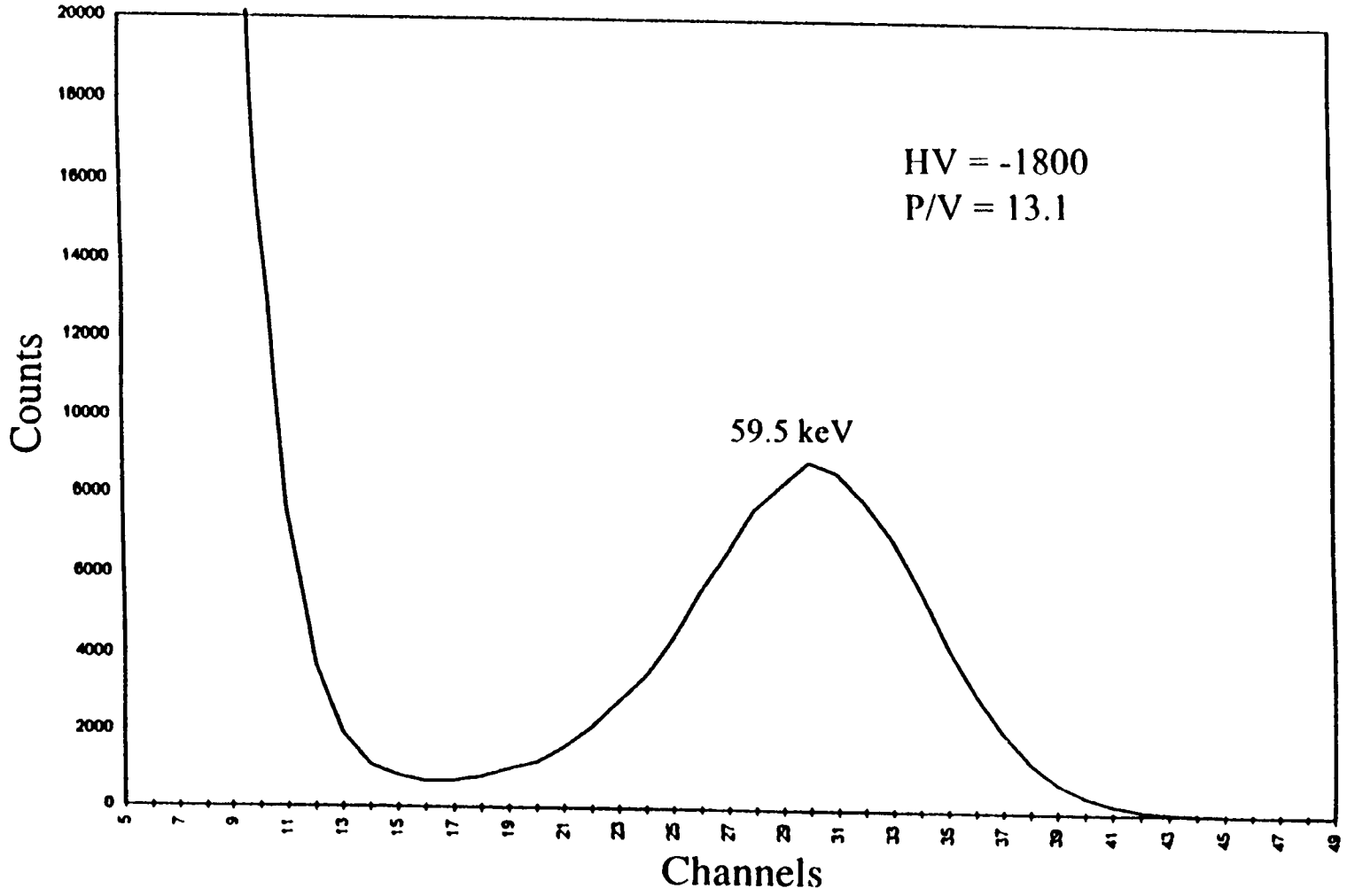


Figure B.5 Am-241 Energy Spectrum Measured with a 1.5" BC501A-XP2020 PMT Detector.

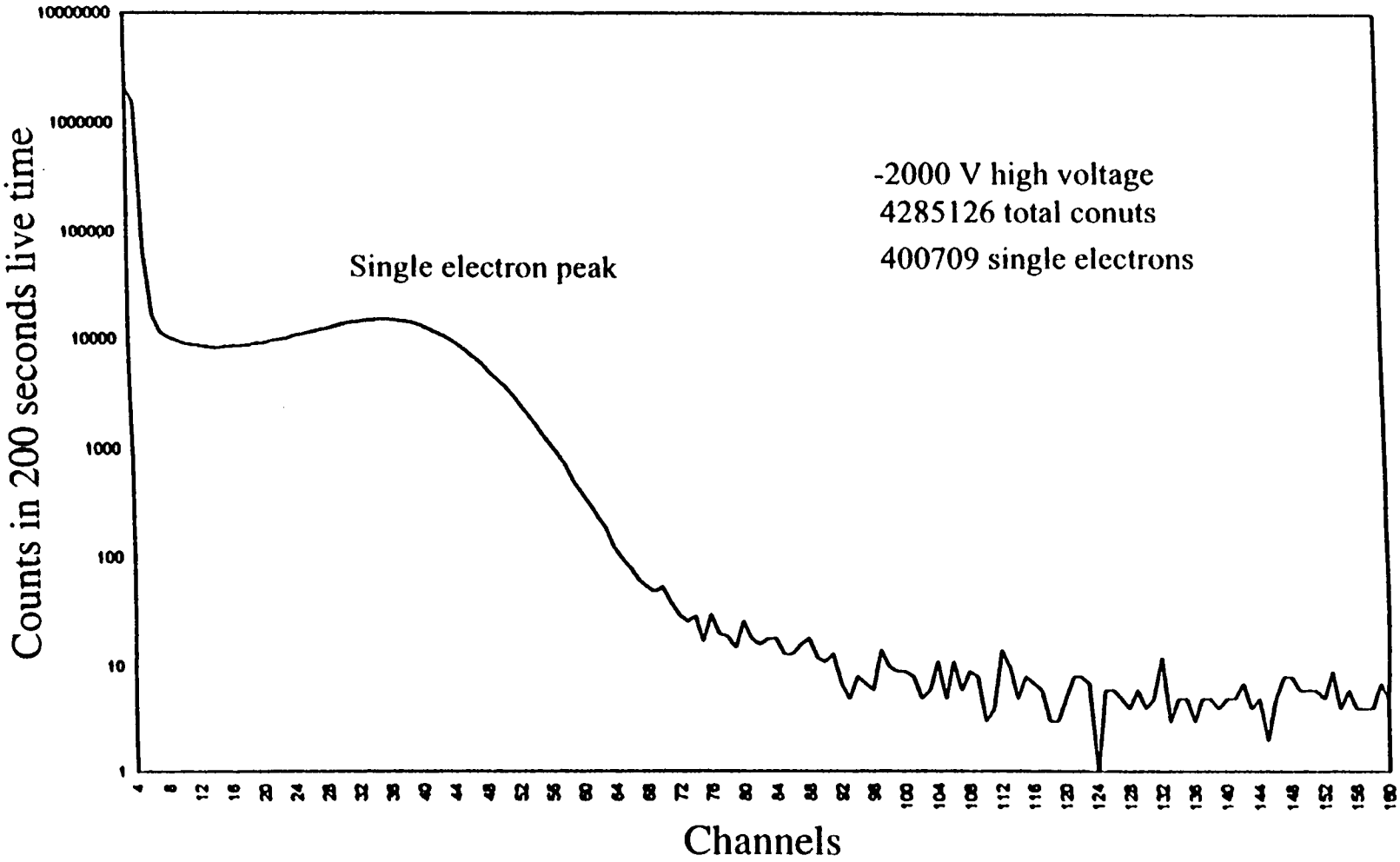


Figure B.6 Single Electron Spectrum of a XP2020 Photomultiplier Tube.

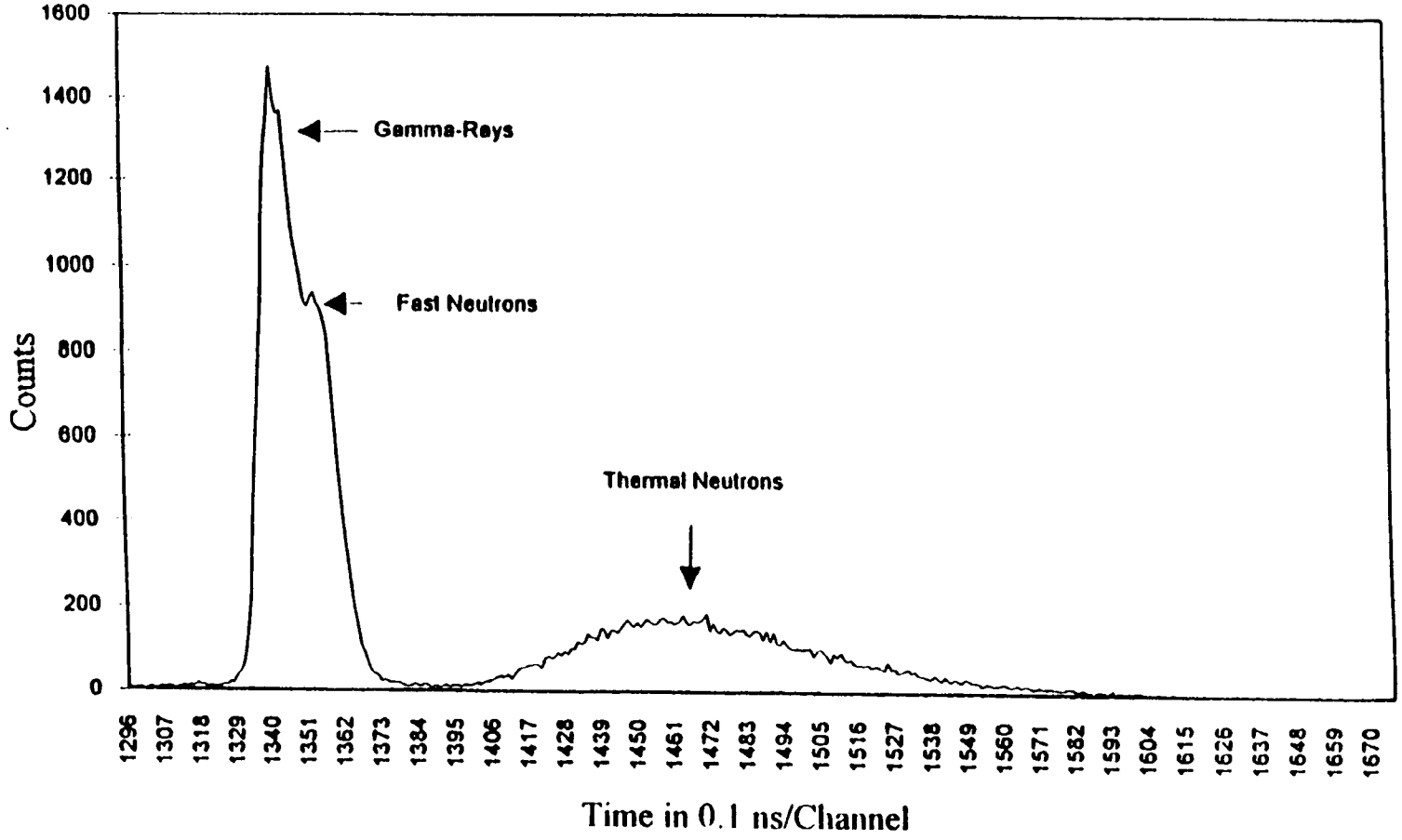


Figure B.7 PSD Spectrum of Pu-Be Source Neutrons from a Li Glass-BC501 Detector.

B.1, the photoelectron production rate for BC501-Hamamatsu R1924 PMT is only 0.6/keV, while PSD performance (1.5 FOM in the 50-500 keV equivalent electron energy range) is better than the former (0.67 FOM in the 200- 500 keV). This test illustrates that the BC501A scintillator has better PSD properties than the BC501.

Size effect tests include two components. One, the photoelectron production rate is measured for four different sizes of BC501A scintillators coupled to the same XP-2020 PMT. Two, a 1.5 cm BC501A detector is coupled to the different areas of a XP-2020 PMT photocathode, and the Am-241 energy spectra are measured to determine the light collection efficiencies. The photoelectron production rate for a detector with a 1.5"BC501A (or 1.0, 1.5 cm BC501A) coupled to a XP2020 PMT is 1.7 $\pm$ 0.2/ keV, and 1.1/keV for a detector with a two inch diameter BC501A-XP2020 PMT. Tests also show that the light collection efficiency for the detector with 1.5 cm BC501A coupled to the edge area of a XP2020PMT is 60-70% lower than that for the detector with central area coupling.

A conclusion can be drawn that the diameter of the scintillator must be smaller than that of the PMT to ensure the maximum photoelectron production rate.

B.3.2 Photomultiplier Tubes and Tube Bases

Optimal PSD performance at low energy requires photomultipliers with high gain, low noise, fast rise time and small transit time spread. However, some compromise is often necessary due to cost. The rise time of the photomultiplier tube (PMT) should be

Table B.1 PSD Performance between 50-500 keV Equivalent Electron Energies of Different detectors.

Detector Type	Photoelectron Production Rate	Rise time/ Transit time Spread in PMT (ns)	Ratio of Neutron Peak to Valley	Figure of Merit
1.5"BC501A -RCA8850	1.0	2.8/-	2.1	0.87
1.5"BC501A -Burle8850	1.1	2.8/-	3.5	1.3
1.5"BC501A -XP2202B	1.6	4.0/8	21	1.5
1.5"BC501A -XP2020	1.7	1.5/2.4	36	1.5
2"BC501A -XP2020	0.8	1.5/2.4	5.6	1.52
15cm BC501A -XP2020	1.7	1.5/2.4	25	1.45
1"BC501 -XP2020	0.8	1.5/2.4	8.8	1.05
2"BC501A -Burle8850	0.6	2.8/-	3.1	1.1
2"BC501A -RCA8850	0.5	2.8/-	0.9	0.5
1"BC501 Hama.1924	0.6	2.0/-	--- (*1.52)	--- (*0.67)
15cmBC501A -XP5600P	0.5	0.65/-	--- (**1.32)	--- (**0.6)
10cmBC501A -XP5600P	0.5	0.65/-	---	---

* Measured in the 200-500 keV energy window.

** Measured in the 100-500 keV energy window.

\- No data available.

faster than the decay time of the scintillator so that the pulse shape information is not distorted. A PMT with a rise time of less than 1 ns should be excellent since the main fast decay time of BC501A is 3.2 ns. However, the PMTs currently available, and with a reasonable price, have rise times in the range of 2 ns-4ns. Since the slow component of the scintillation light has a decay time constant more than 30 ns (30-270 ns), these PMTs are adequate for many applications. The photomultiplier tube with a small transit time spread is also necessary preferable to obtain good time resolution. Note that the rise time and transit time spread must be less than the peak separation between the neutrons and photons in the PSD time spectrum.

An experiment is designed to evaluate PSD performance for several PMTs. Photoelectron production rates and PSD performances for detectors with different PMTs mounted to a same 1.5" BC501A scintillator are studied, and results are listed in Table B.2. As can be seen from the table, the best result is from a XP2020 PMT, which has 1.5 ns rise time, 2.4 ns transit time spread and 1.7 /keV photo-electron production rate when coupled to a 1.5"BC501A scintillator. The next best result is from a XP2202B PMT, which has 4 ns rise time, 8 ns transit time spread and 1.6/ keV photoelectron production rate. These results also show that the photo-electron production rate is the most important factor that influences the PSD behavior at low energies, where statistical effects dominate.

The voltage applied to a photomultiplier tube is another important factor in PSD performance. In order to reduce the PMT noise and to avoid the saturation over a large

Table B.2 PSD Performance in 0.27-1.8 MeV Neutron Energies of 1.5" BC501A Detectors with Different PMTs.

PMT Type	Rise Time (ns)	Transit Time Spread (ns)	PMT gain X 10^6	* P.P. Rate / keV	Single-Electron Noise /second	P/V Ratio	FOM
XP2020	1.5	2.4	30	1.7	4000	36	1.5
XP2202 B	4.0	8.0	1	1.6	400	21	1.5
Burle-8850	2.8	-	16	1.1	200	3.5	1.3
RCA-8850 (old)	3.8	-	16	1.1	7000	2.1	0.87
** XP2262 B	2.0	3.0	30	-	-	-	-

* P.P. Rate: Photoelectron production rate.

** XP2262B may be a good candidate for PSD application since its has a high blue light sensitivity (11.2 mA/Lm), which means higher photoelectron production rate, and excellent time property.

dynamic range, lower value is preferable; however, high gain and the good time properties are essential. Results demonstrate that -1800 to -2000 V is the best working voltage range for a XP2020 photomultiplier tube. Within this voltage range, the single electron noise level is about 4000-4500 counts/second and increases quite slowly with the increasing working voltage. At larger than 2000 V (absolute value) the noise rapidly increases.

The effect of polarity of the PMT high voltage on the noise is tested on a XP2202B tube. The single electron noise is determined to be about 1800 $\pm$ 100 counts/second at $\pm$ 1300 V working voltage, and no significant difference is observed between their results.

If one uses the anode and the last several dynode output signals as start and stop signals, respectively, in the zero-crossing method, better results are obtained than by using the dynode (or the anode) only for both the start and the stop signals. For the tests performed with the 1.5"XPBC501A-XP2020 PMT detector, the ratio of the neutron peak-to-valley is 13.0 $\pm$ 0.2 for the former, and 8.3 $\pm$ 0.2 for the latter under the same conditions.

Table B.1 is an experimental summary for PSD performance of different detectors for equivalent electron energies in the range of 50-500 keV. It shows that the 1.5"BC501A-XP2020 PMT detector is the best combination for components available for this research. Note that the photoelectron production rate of this detector reaches 1.7/keV. Figures B.8 and B.9 provide examples of rise time spectra which were measured

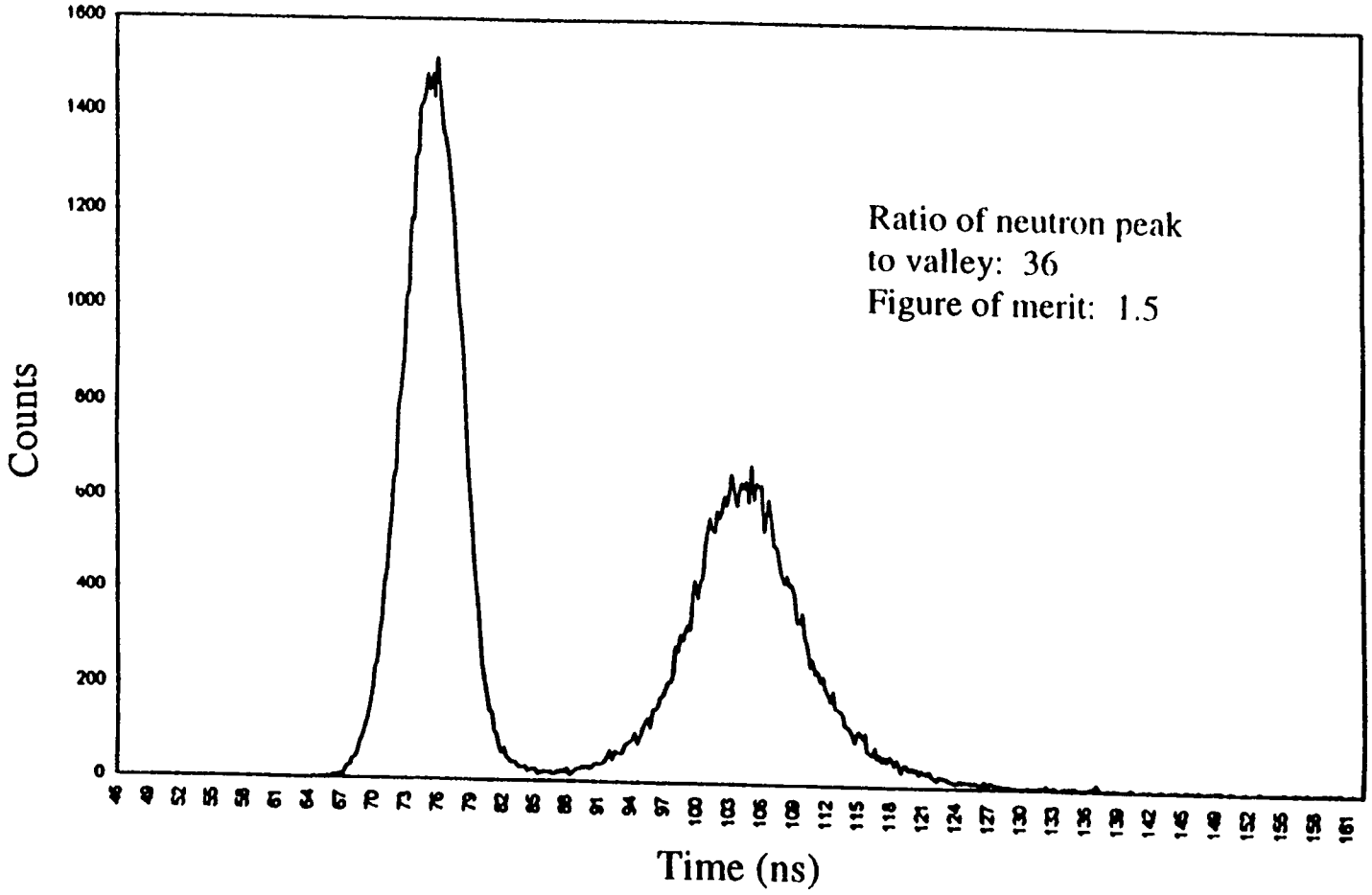


Figure B.8 PSD Spectrum from a 1.5" BC501A-XP2020 PMT Detector
(Pu-Be Source, 0.25-2.5 MeV Neutron Energies).

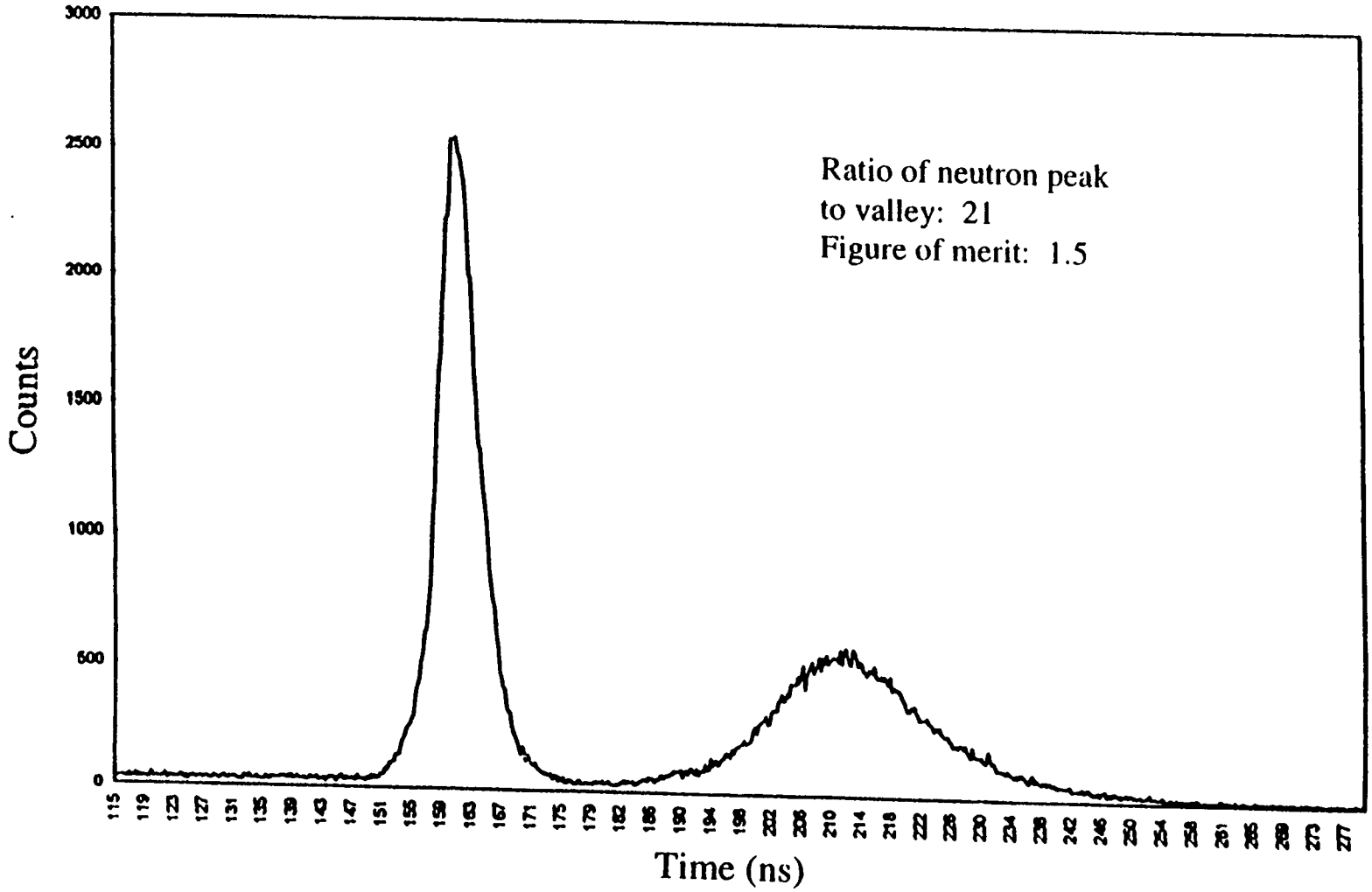


Figure B.9 PSD Spectrum from a 1.5" BC501A-XP2202B PMT Detector (Pu-Be Source, 0.25-2.5 MeV Neutron Energies).

above 250 keV neutron energy with the 1.5"BC501A-XP2020 PMT and the 1.5"Bc501A-XP2202B PMT detectors, respectively. The ratios of neutron peak-to-valley are 36 $\pm$ 3 and 21 $\pm$ 2, respectively. Both figures of merits are the same (1.5 $\pm$ 0.1). The 0.6 CFD value, 40 ns integration time constant and the "anode-for-start, dynode-for-stop" mode are used in the experiments.

Stabilization of the dynode potentials in the PMT base is important and large chain currents result in better PSD performance than small chain currents. Figure B.10 illustrates two neutron-photon rise time spectra above 1.5 MeV neutron energy measured with a same detector (1"BC501-Hammatsu1294), but different tube bases with different chain currents. It is apparent that the 1 mA chain current tube base provides much better separation of neutrons from gamma-rays than the 0.1 mA chain current base. With a larger chain current (1 mA instead of 0.1 mA), the potentials of the dynodes and the gain of the PMT are increased significantly. In order to cover a large energy range and high count rate, a tapered base design, Zenar diodes or the active stabilizer are necessary to suppress the space charge effect between last few dynodes and the anode, instead of the equal voltage divider used here,

B.3.3 Signal Processing

An experiment is designed to test the effect of the constant fraction discrimination factor (CFD) on PSD performance. The results are listed in Table B.3. These data are

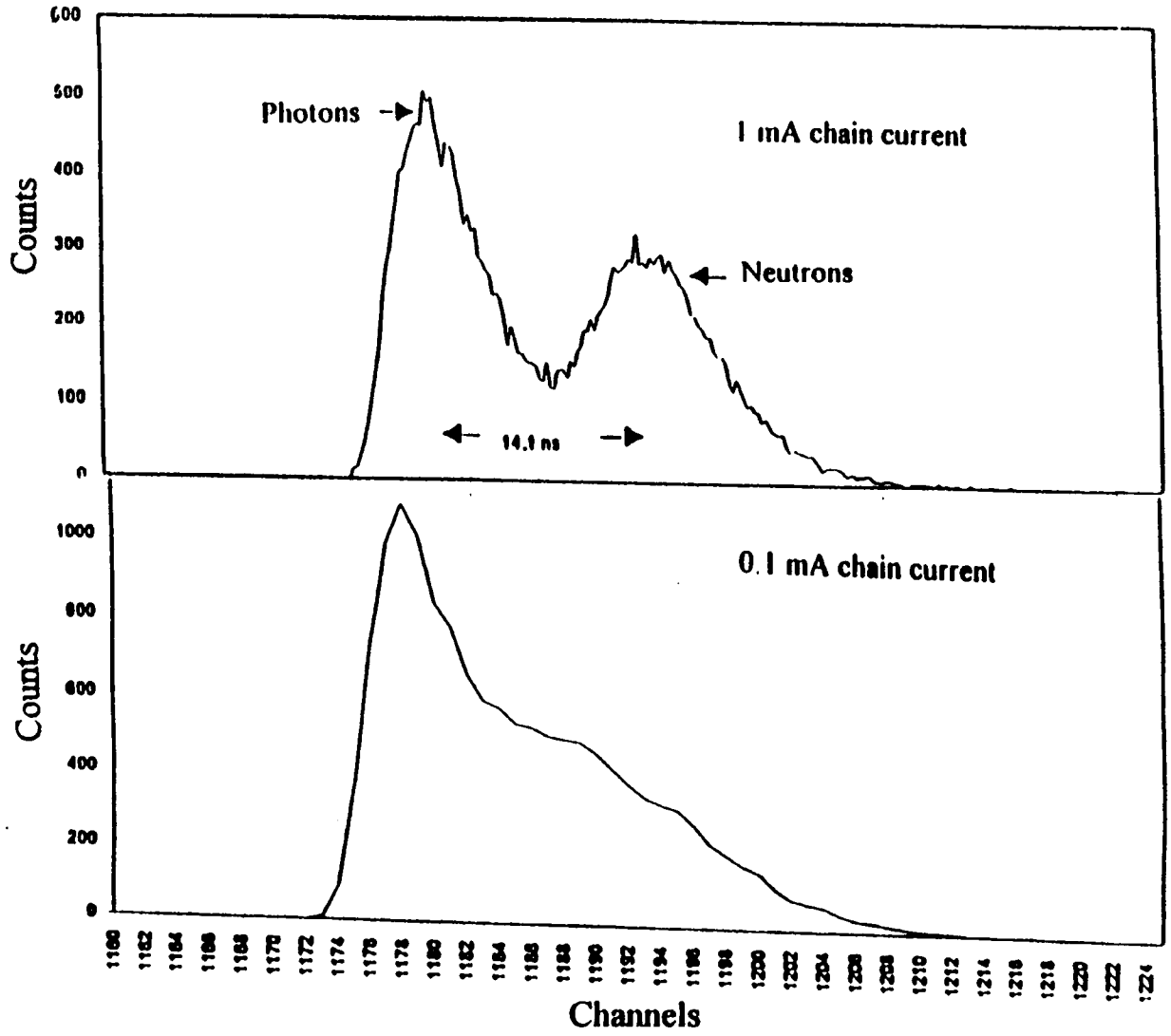


Figure B.10 Neutron-Photon PSD Spectra from Different Tube Bases.

Table B.3 CFD Setting Effect on the PSD Performance.

Fraction of CFD trigger	Ratio of Neutron peak to Valley	Figure of Merit
0.9	11	1.36
0.8	21	1.63
0.7	30	1.83
0.6	32	1.71
0.5	31	1.70
0.4	27	1.66
0.3	20	1.50
0.2	12	1.30
0.1	9	0.90

Note : Measurement uncertainty is 5% .

acquired under the exact same condition but different CFD values (selection on Ortec552). Results demomstrates that 0.7 is the best CFD selections. The test also shows that best

integration time of amplifier Ortec460 is 40 ns (mimimum setting available).

Unfortunately, the integration time window inside TC-5020 is fixed; hence, the window location effect in the charge comparison method is not tested.

Figure B.11 illustrates the anode cable length effect on the PSD for the charge comparison method. As the cable length increases, PSD performance decreases. This is due to uneven signal attenuation in the propagation along the output cable, which deteriorates the time information, especially in the small time constant of current mode. However, tests show that 5 meters of cable length is acceptable at a bias of 500 keV neutron energy.

B.3.4 Energy Dependence of the PSD Performance

The energy dependence of PSD performance of a 1"BC501-Hamamatsu R1924 PMT is measured using the zero-crossing method. The results are illustrated in Figure B.12. It is apparent that the ratio of neutron peak to valley and the figure of merit decrease as the detection energy threshold decreases. It is concluded that the 1" BC501-Hamamatsu R1924 PMT is not suitable for low energy (< 500 keV neutron energy) detection because of poor performance below this threshold. This is caused by its low photoelectron production rate(0.6/keV; Table B.1) and its large transit time spread (none-focusing PMT).

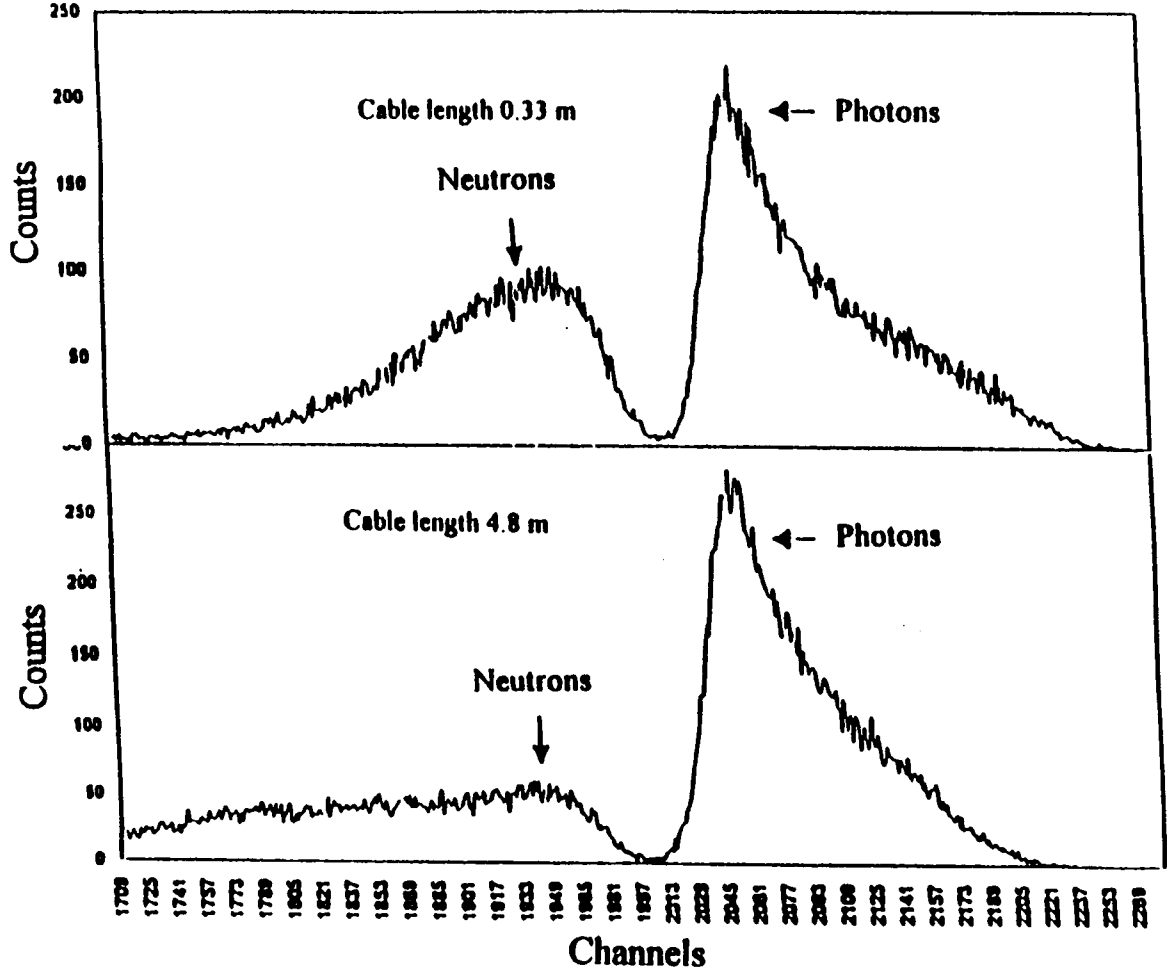


Figure B.11 PSD Spectra of Pu-Be above 500 keV Neutron Energy from TC5020 with Different Cable Connection.

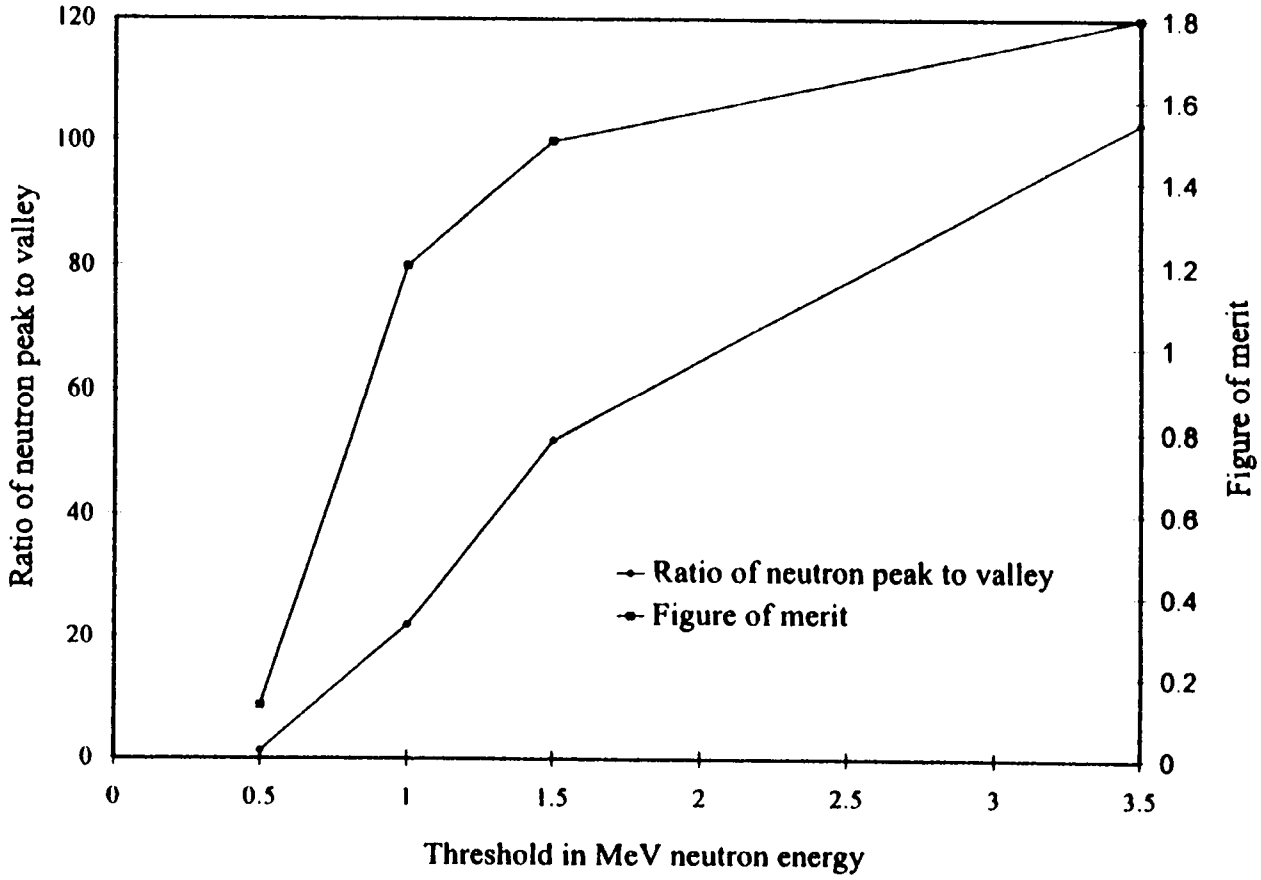


Figure B.12 Energy Dependent PSD Performance of a 1" BC501-Hama R1924 PMT Detector

Figure B.13. shows the energy dependence of the PSD performance of the 1.5"BC501A-XP2020 PMT detector. The photoelectron production rate of this detector is 1.7/keV and is much higher than that of 1"BC501-Hamamtsu R1924. Its PSD performance is much better than the later. At 500 keV neutron energy, the figure of merit and the ratio of neutron peak to valley are 1.6 and 40, respectively, compared to 0.1 and 1.2 for the 1"BC501-Hamamatsu R1924 PMT detector.

Figure B.14 shows rise time spectra for different neutron energy windows (ΔE = 25 keV) measured with the 1.5"BC501A-XP2020 detector. The PSD performance below 350 keV neutron energy deteriorates rapidly as the energy decreases.

B.3.5 Comparison of the Zero-Crossing and the Charge Comparison Methods

The zero-crossing and the charge comparison methods are two commonly used methods for pulse shape discrimination and there is disagreement in several publications regarding which method is better at low energies. For example, B. Sabbah, et al.(1968)[20] compared two methods and concluded that the charge comparison is better than the zero-crossing. G. Ranucci (1995)[27] performed a evaluation of the PSD properties of scintillators with statistical analysis and pointed out that charge comparison offers superior PSD performance and that the best discrimination fraction of the amplitude for zero-crossing is optimal about 0.9. However, recently D. Wolski, M. Moszynski (1995)[18] claimed that the zero-crossing method is superior to the charge comparison at low energies.

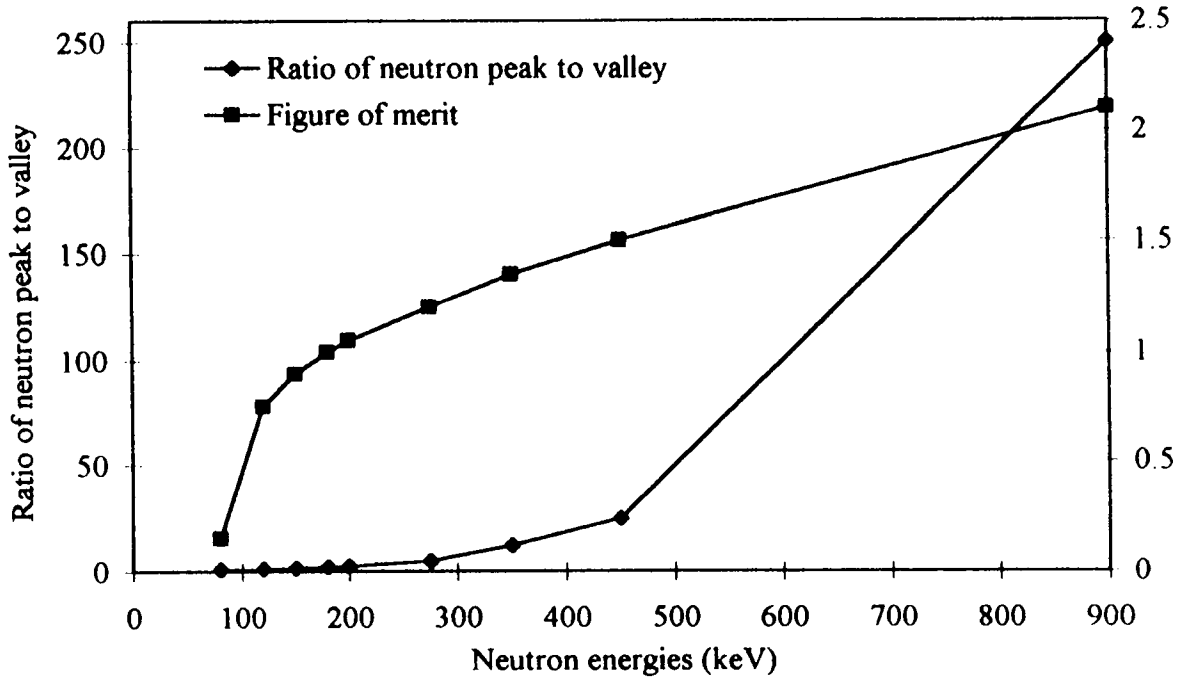


Figure B.13 PSD Performances of a 1.5" BC501A-XP2020 PMT Detector

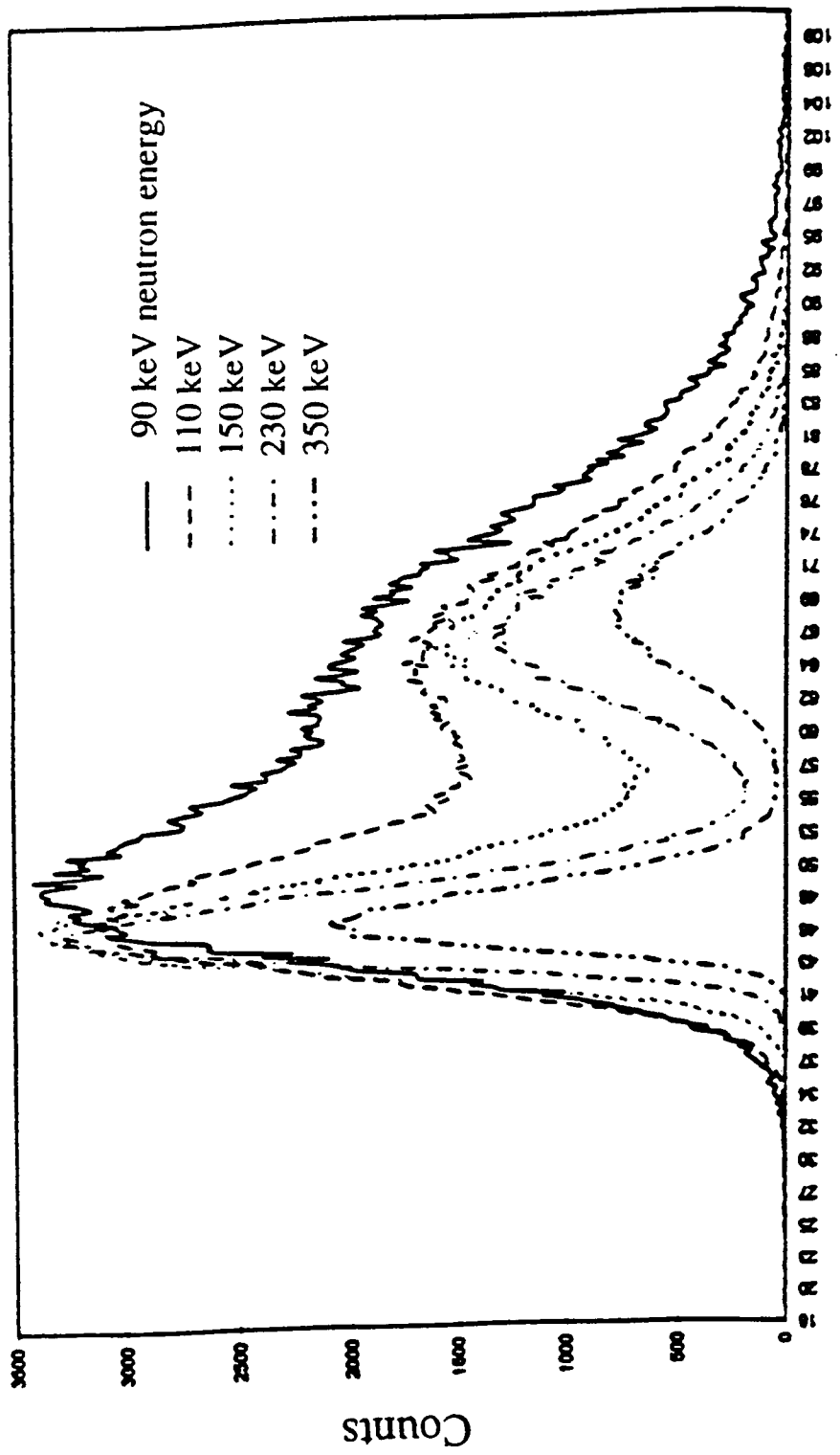


Figure B.14 PSD Spectra from a 1.5" BC501A-XP2020 PMT Detector at Different Energies (ΔE=25 keV).

The present work evalulates zero-crossing and charge comparison methods for a 2"BC501A-RCA8850 detector and results above 1.5 MeV neutron energy are shown in Figure B.15. The PSD spectrum on the lower portion of the figure is measured with the TC-5020, and is the spectrum of the fast scintillation component (the photon peak on the right side). The PSD performance (ratio of neutron peak-to-valley: 150:1) of the charge comparison is better than that of the zero-crossing (55:1). Unfortunately, this experiment at low neutron energies was not completed since this equipment was borrowed from the Oxford company and was later sold. Therefore, no conclusion for low energies can be drawn from these experiments.

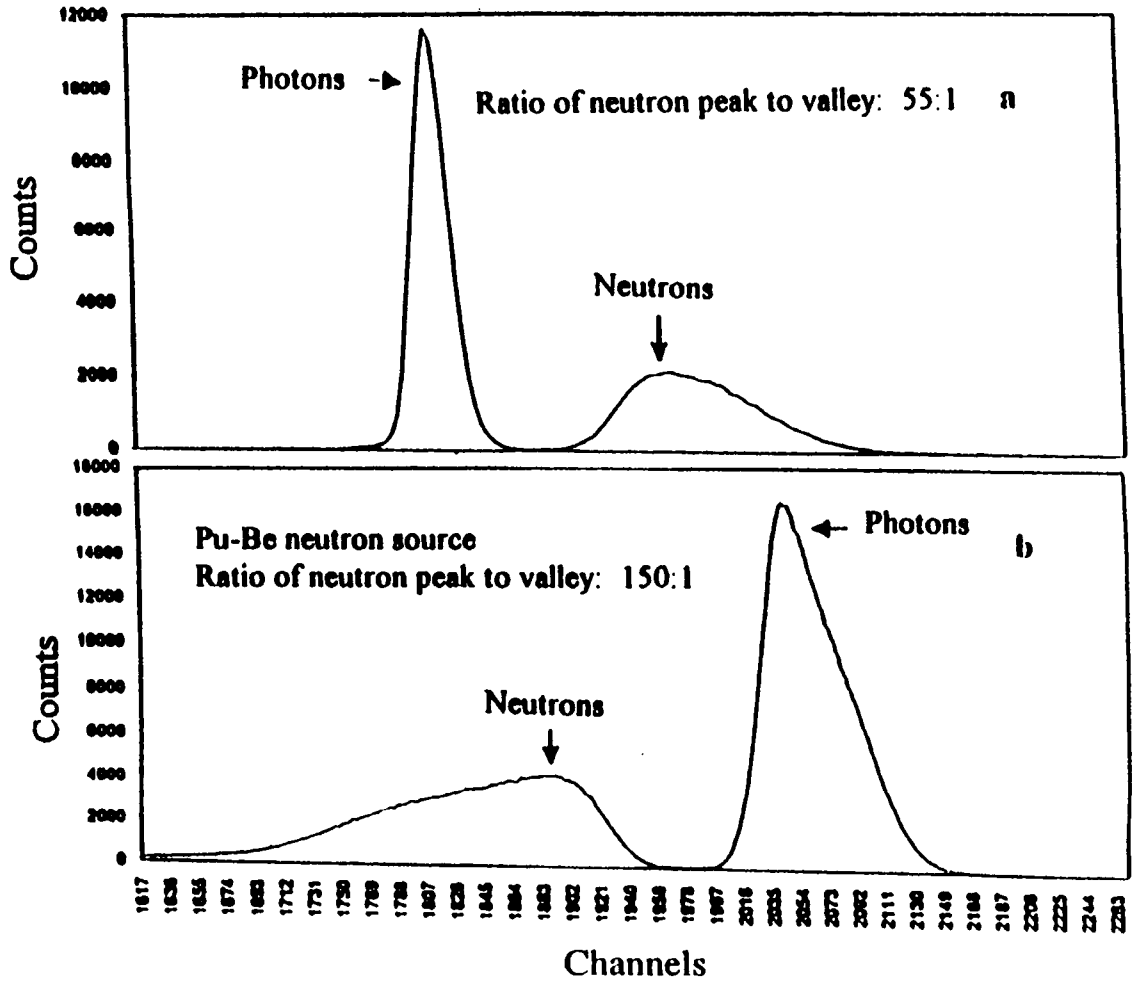


Figure B.15 Neutron-Photon PSD Spectra (Pu-Be Source above 1.5 MeV Neutron Energy) (a) with the Zero-Crossing Method, (b) with the Charge Comparison Method.

Appendix C

Response of Liquid Scintillators to Photons and Neutrons

The responses of liquid scintillators to the photons and neutrons are quite different due to their different mechanisms. Even with the same energy deposition, the light output from a neutron event is much smaller than that from a photon event. Therefore, to compare the neutron-photon PSD performance at different energies, the term "equivalent electron energy" is frequently used. Althrough the energy deporsited by a photon or a neutron is of interest, it is also important to know the equivalent electron energy.

Compton scattering is the most important interaction between the liquid scintillator and photons below 5 MeV photon energy. The maximum scintillation light output of a liquid scintillator exposed to photon radiation is proportional to the maximum energy of the Compton scattering electrons, which can be calculated by the formula,

$$E_{e,Max} = \frac{2\ E_\gamma^2}{511\ +2\ E_\gamma},$$

where E_γ is the photon energy in keV. $E_{e,\,Max}$ is also called equivalent electron energy of a photon. Figure C.1 shows the calculated $E_{e,\,Max}$ at different photon energies.

The responses of the liquid scintillator to electrons and neutrons are quite different. The light product rate of electrons is much higher than that of neutrons at the same energy deposit condition. The neutron energies (below 1 MeV) corresponding to

equivalent electron energies are shown in Figure C.2, which is from J.M. Adams et al.'s work[24]. As can be seen, from the fitted curve, the relationship between these two energies can be expressed by

$$E_n = 4.1852\ E_e + 13.643\ .$$

The relationship between the proton energy in 0.1-30 MeV and the equivalent electron energy can be expressed by[36]

$$E_e = 0.83\ E_p - 2.82\ (1 - e^{-0.25\ E_p^{0.93}})\ .$$

Since the maximum recoil energy of a hydrogen nucleus caused by a neutron elastic scattering is equal to the neutron energy, this formula is valid for neutron energies above 0.1 MeV. The calculated results are shown in Figure C.3.

By using the equivalent electron energy conversion and previous figures, the corresponding neutron energy to a given photon energy can be easily estimated, and vice versa.

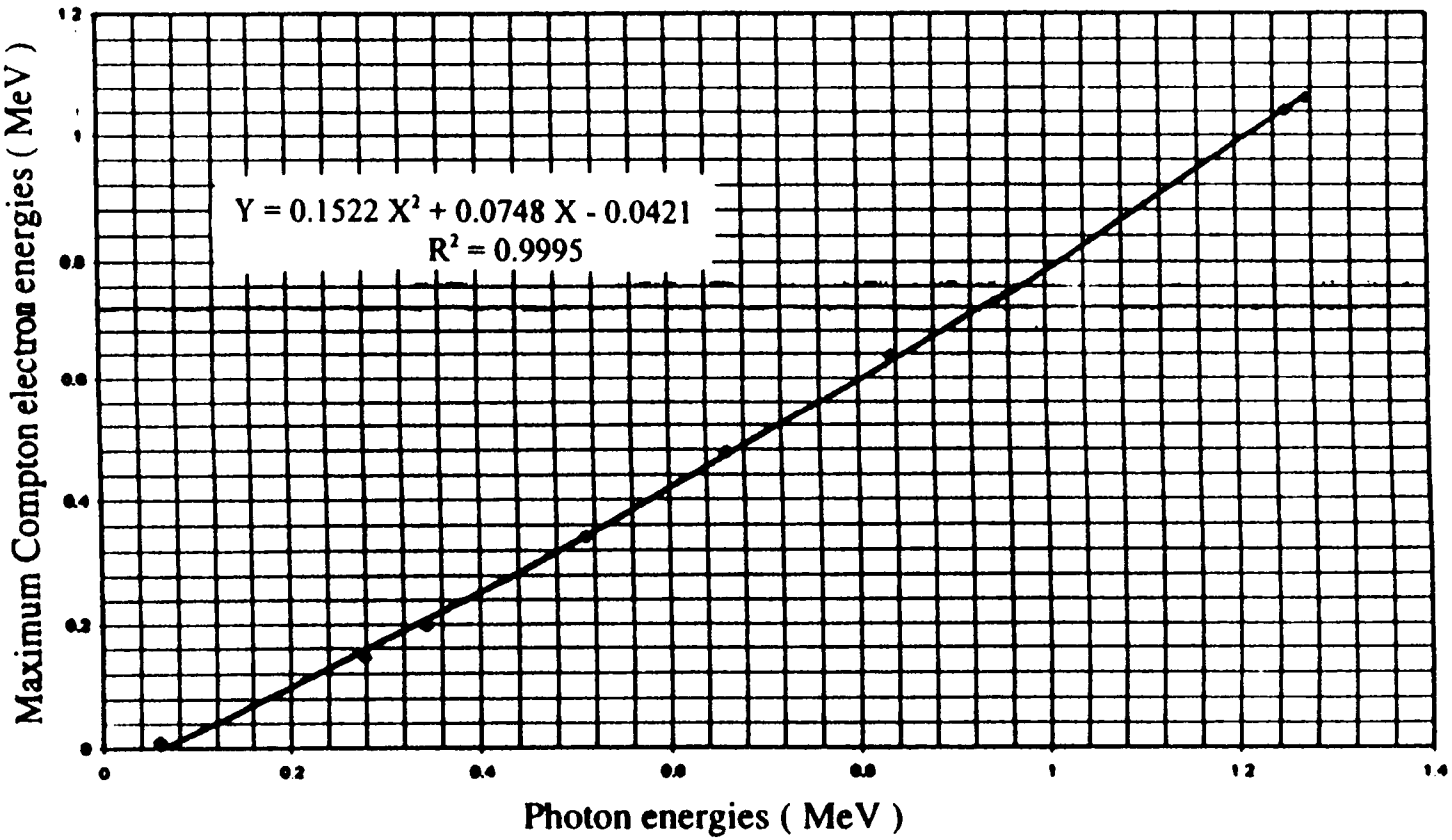


Figure C.1 Maximum Compton Electron Energies vs. Photon Energies

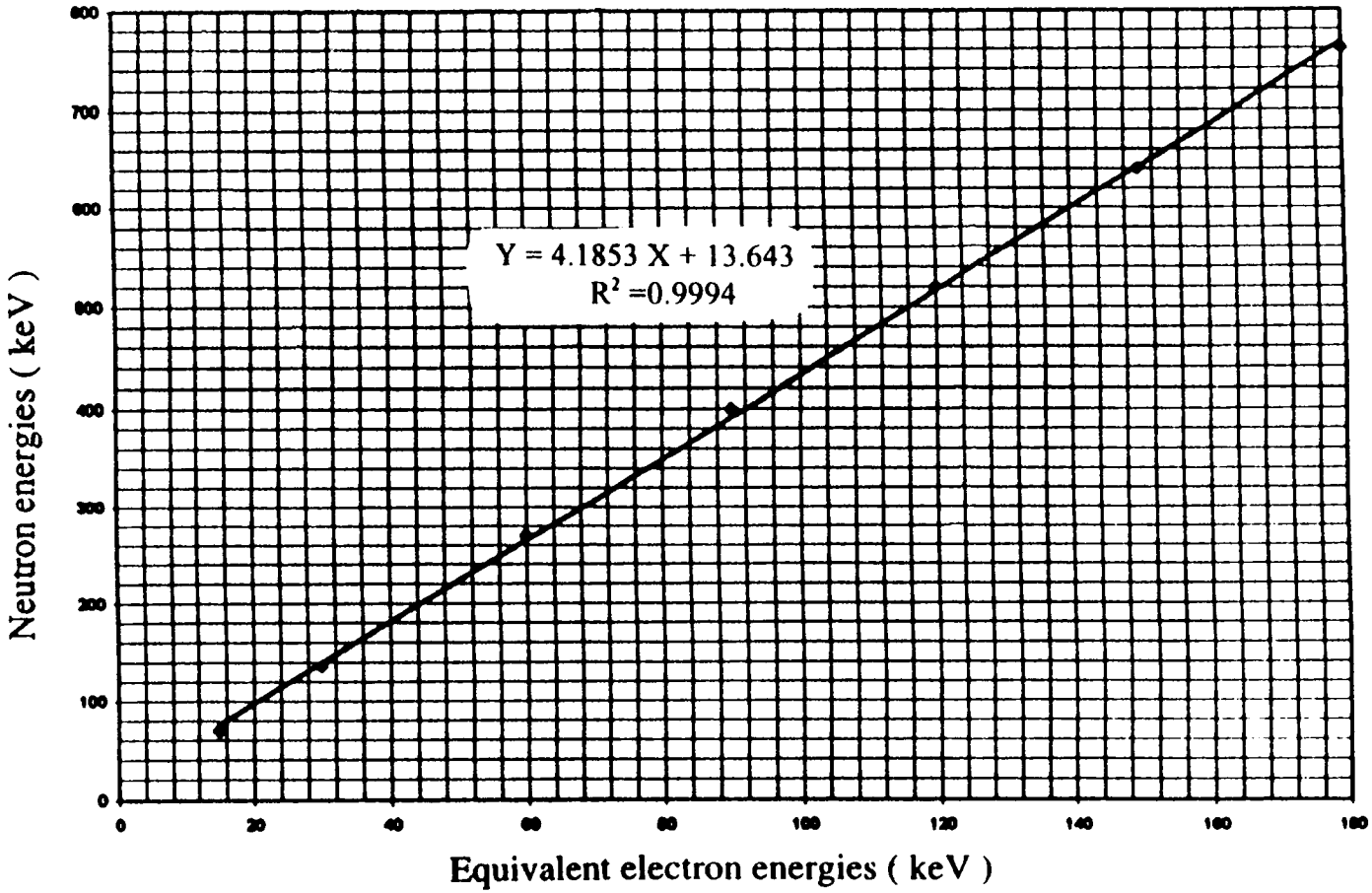


Figure C.2 Neutron Energies vs. Equivalent Electron Energies (for NE213).

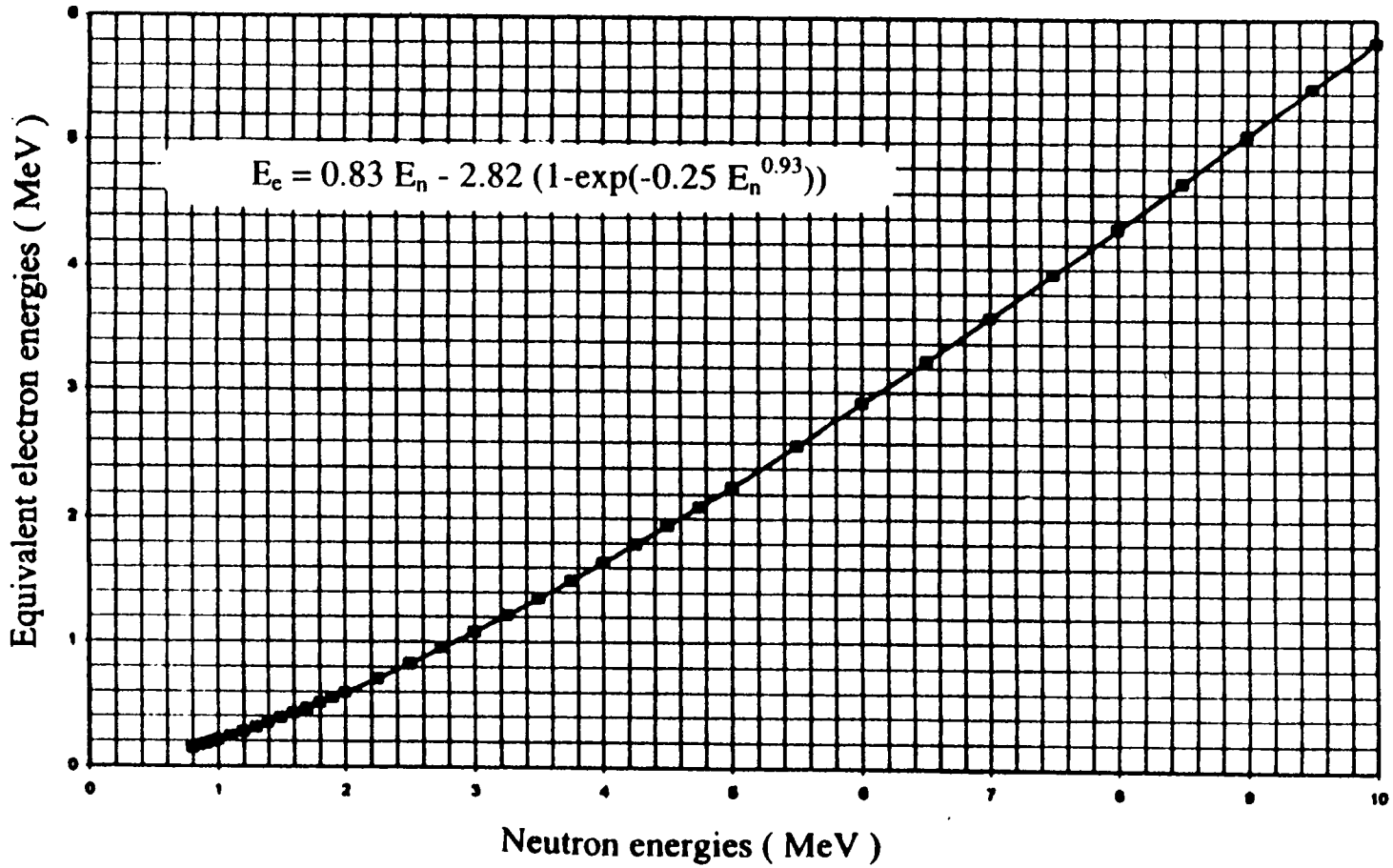


Figure C.3 Scintillation Light Output of NE213.

Appendix D

Program Files for the Perceptron Classifier

```
C*******************************************************************************
C      percepn1.for
c       Linear Perceptron Classifier Source file
c           Perceptron classifier + Bayes classifier program
c           Fractional-correction method
c
c
c            By  Zhong Cao
c
c                March, 1997
c
c     For 2 class classification w1/w2 such as neutron-photon
c     Note the training data have different format from testing data
c         (see example dat1.dat and t8.dat)
c     After training, weit.dat file is automatically produced.  You only need prepare data
c        file tested.
c     If you just use perceptron classifier only, weit.dat can be used directly without a
c        modification.
c     If  you want use both the perceptron and the Bayes, you must attach the
c        calibrated FWHM parameters for neutrons and photons to it.
c         (see example weit.dat)
c
c     Usage:
c        fl percepn1.for                <- for compile
c        percepn1 dat1.dat              <-for training   it will produce out.dat and weit.dat
c                                              files.   dat1.dat is any training file.
c        percepn1 t8.dat                <-for testing.   t8.dat is any testing data file.  It will
c                                              generate out.dat files.
c
c
c*******************************************************************************
      dimension x(3,400),w(4,4),d(3,400),y(3,3,100),nj(3),iter(3)
      dimension icountg(5),icountn(5),ratio(5)
      dimension verg(5),vern(5)
        !x() for testing data
          !x(1,k) is channel # of data k
          !x(2,k) is energy of data k
          !x(3,k) is counts of data k
          !icountg(4): gamma-counts in four energy regions
          !   <1.0; 1.0-2.0; 2.0-3.0;3.0-4.7; >4.7 ( v )
          !icountn(5): for neutrons
          !ratio(5) =icountg(5)/icountn(5)
        !d() for decision function
```

```
      !y() for training data
      character*1 ans,char(80)
      open(1,file=' ')
      open(2,file='weit.dat')
      open(3,file='out.dat')
      Write(0,*)"training ? y/n.----if no do classification  "
      read(0,2) ans
      write(0,*)"input ID------"
      read(0,3) char
      write(3,*)"*****************************************"
      write(3,4) char
      write(3,*)"*****************************************"
      write(3,*)
      write(3,*)"    Fractional Correction Algorithm       "
      write(3,*)
2     format(1a)
3     format(80A)
4     format(5x,80a)
      if(ans.eq.'y') go to 40
      write(3,*) "----------------Tested data-----------------"
      write(3,*)
      read(1,*) k0,k1,j1   ! k0 is data number; k1 is dimension (2 or 3)
c                    ! j1 is number of decision function(1 or 3)
      do 5 j=1,j1
        read(2,*) (w(i,j),i=1,k1+1)     ! input weights already tested
5     continue
      read(1,*)
      write(3,9)
9     format(8X,'data(i)',7x,'x(i)',13x,'y(i)',10x,'z(i)')
      do 6 k=1,k0
        read(1,*) (x(i,k),i=1,k1)
c       write(0,*) (x(i,k),i=1,k1)
c       write(3,*) k,(x(i,k),i=1,k1)
  6   continue
c     pause 1
      write(3,*)
      write(3,*)"------------Classification Results-------------"
      write(3,11)
11    format(6x,'data(i)',3x,'decision(1)',10x,'d(2)',10x,'d(3)')
      do 10 k=1,k0          ! data #
        do 15 j=1,j1        ! j is output node #
          d(j,k)=w(1,j)
          do 17 i=1,k1      ! dimension      !i=1---1(augment)
            d(j,k)=d(j,k)+ x(i,k)*w(i+1,j)   !i=2---x, i=3--y,i=4--z
```

```
 17       continue
 15     continue
c      write(0,*) " data index: ",k,(d(j,k),j=1,j1)
c      write(3,*) k,(d(j,k),j=1,j1)
      if(j1.eq.1) go to 30
      if(d(1,k).gt.0) go to 21
      if(d(2,k).gt.0) go to 22
      if(d(3,k).gt.0) go to 23
      write(0,*) " ------------belong to the indetermined classes"
      write(3,*) "------------- belong to the indetermined classes"
      go to 24
21     write(0,*) "---------- Class 1"
      write(3,*) "---------- Class 1"
      go to 24
22     write(0,*) "-----------Class 2"
      write(3,*) "-----------Class 2"
      go to 24
23     write(0,*) "---------- Class 3"
      write(3,*) "-----------Class 3"
24     continue
      go to 35
30     continue

      if(d(1,k).gt.0) go to 32
c      write(0,*) "----------------------- Class neutron"
c      write(3,*) "----------------------- Class neutron"
      if (x(2,k) .lt.1) then
           icountn(1)=icountn(1)+x(3,k)
           vern(1)=vern(1)+x(1,k)*x(3,k)
        else if(x(2,k).lt.2) then
           icountn(2)=icountn(2)+x(3,k)
           vern(2)=vern(2)+x(1,k)*x(3,k)
        else if(x(2,k).lt.3) then
           icountn(3)=icountn(3)+x(3,k)
           vern(3)=vern(3)+x(1,k)*x(3,k)
        else if(x(2,k).lt. 4.7) then
           icountn(4)=icountn(4)+x(3,k)
           vern(4)=vern(4)+x(1,k)*x(3,k)
        else
           icountn(5)=icountn(5)+x(3,k)
           vern(5)=vern(5)+x(1,k)*x(3,k)
        end if
      go to 35
 32     continue
```

```
c     write(0,*) "---------------------- Class photon"
c      write(3,*) "---------------------- Class photon"
      if (x(2,k) .lt.1) then
           icountg(1)=icountg(1)+x(3,k)
           verg(1)=verg(1)+x(1,k)*x(3,k)
        else if(x(2,k).lt.2) then
           icountg(2)=icountg(2)+x(3,k)
           verg(2)=verg(2)+x(1,k)*x(3,k)
        else if(x(2,k).lt.3) then
           icountg(3)=icountg(3)+x(3,k)
           verg(3)=verg(3)+x(1,k)*x(3,k)
        else if(x(2,k).lt. 4.7) then
           icountg(4)=icountg(4)+x(3,k)
           verg(4)=verg(4)+x(1,k)*x(3,k)
        else
           icountg(5)=icountg(5)+x(3,k)
           verg(5)=verg(5)+x(1,k)*x(3,k)
        end if
35     continue
c      write(3,*) k,(d(j,k),j=1,j1)
10     continue
      if(j1.ne.1) go to 100   ! not for two classes case; go to stop
      do 36 j=1,5       ! j is energy region number
      igcount=icountg(j)+igcount
      incount=icountn(j)+incount
      if(icountn(j).eq.0) icountn(j)=1
      ratio(j)=float(icountg(j))/float(icountn(j))
      vern(j)=vern(j)/icountn(j) !average peak position of neutrons in j
      if(icountg(j).eq.0) icountg(j)=1
      verg(j)=verg(j)/icountg(j) !average peak position of gamma in j
      write(3,*) "-------------------------------------------"
c      write(0,*) "Neutron number in region:",j,icountn(j)
c      write(3,*) "Neutron number in region:",j,icountn(j)
c      write(0,*) "Photon number in region: ",j,icountg(j)
c      write(3,*) "Photon number in region: ",j,icountg(j)
c      write(0,*) "Ratio of gammas to neutrons in region:",j,ratio(j)
c      write(3,*) "Ratio of gammas to neutrons in region:",j,ratio(j)
c     write(0,*) "Average peak position of neutrons in region",j,vern(j)
c     write(3,*) "Average peak position of neutrons in region",j,vern(j)
c     write(0,*) "Average peak position of photons in region",j,verg(j)
c     write(3,*) "Average peak position of photons in region",j,verg(j)

36     continue
      if(incount.eq.0) incount=1
```

```
      ratiot=float(igcount)/float(incount)!total ratio of gammas to neutrons
      write(0,*) "Total Photon number: ",igcount
      write(3,*) "Perceptron classification results******"
      write(3,*) "Total Photon number: ",igcount
      write(0,*) "Total neutron number: ",incount
      write(3,*) "Total neutron number: ",incount
      write(0,*) "Total Ratio of gammas to neutrons:",ratiot
      write(3,*) "Total Ratio of gammas to neutrons:",ratiot
      write(3,*) "-------------------------------------------"
c-------------------------------------------------------------
      write(0,*) "use Bayes classifier ? y/n"
      read(0,2) ans
      if(ans.ne.'y') go to 100      ! go to stop
      read(2,*)  yg1,ag1,tg1,xg1,yg2,ag2,tg2,xg2
                 !FWHM parameters for gammas
      read(2,*)  yn1,an1,tn1,xn1,yn2,an2,tn2,xn2
                 !FWHM parameters for neutrons
      write(0,*)"gamma", yg1,ag1,tg1,xg1,yg2,ag2,tg2,xg2
      write(0,*) "neutron", yn1,an1,tn1,xn1,yn2,an2,tn2,xn2
c      pause 1
      igcount=0
      incount=0
      do 205 k=1,k0
      if (x(2,k) .gt. 1.0) then          ! for energy > 1.0 ( V )
        fwhmg=yg2+ag2*exp(-(x(2,k)-xg2)/tg2)  ! fwhm for gammas
        fwhmn=yn2+an2*exp(-(x(2,k)-xn2)/tn2)  ! fwhm for neutrons
       else    ! energy <= 1.0
        fwhmg=yg1+ag1*exp(-(x(2,k)-xg1)/tg1)  ! fwhm for gammas
        fwhmn=yn1+an1*exp(-(x(2,k)-xn1)/tn1)  ! fwhm for neutrons
       end if
c        write(0,*) "gamma fwhm",fwhmg
c         write(0,*)"neutron fwhm",fwhmn
c       write(3,*) "gamma fwhm",fwhmg
c         write(3,*)"neutron fwhm",fwhmn
        sigmag=fwhmg/2.35
        sigman=fwhmn/2.35
c         pause 2
      if (x(2,k) .lt.1) then
            xg=(x(1,k)-verg(1))/sigmag   ! normarization to FWHM
            xn=(x(1,k)-vern(1))/sigman
            aa=ratio(1)
         else if(x(2,k).lt.2) then
            xg=(x(1,k)-verg(2))/sigmag   ! normarization to FWHM
            xn=(x(1,k)-vern(2))/sigman
```

```
          aa=ratio(2)
        else if(x(2,k).lt.3) then
          xg=(x(1,k)-verg(3))/sigmag   ! normarization to FWHM
          xn=(x(1,k)-vern(3))/sigman
          aa=ratio(3)
        else if(x(2,k).lt. 4.7) then
          xg=(x(1,k)-verg(4))/sigmag   ! normarization to FWHM
          xn=(x(1,k)-vern(4))/sigman
          aa=ratio(4)
        else
          xg=(x(1,k)-verg(5))/sigmag   ! normarization to FWHM
          xn=(x(1,k)-vern(5))/sigman
          aa=ratio(5)
        end if
c        write(0,*) "xg, xn:",xg,xn
        if(xg.ge.10.0) then
          incount=incount+x(3,k)    ! neutron events
          go to 204
        else if (xg.lt.-10.0) then
          igcount=igcount+x(3,k)    ! gamma events
          go to 204
        else
          call simpson(probg,xg,10.0,1,0.001)
        end if
c        pause 3
        if(xn .le.-10.0) then
          igcount=igcount+x(3,k)    ! gamma events
          go to 204
        else if (xn .ge. 10.0) then
          incount=incount+x(3,k)
          go to 204
        else
          call simpson(probn,-10.0,xn,0,0.001)   ! 0.01 is accuracy
        end if
c        pause 4
        probg=aa*probg          ! aa is ratio of gamma to neutron
c        write(0,*)"probg, probn: ",probg,probn

        if(probg.gt.probn) then
          igcount=igcount+x(3,k)
        else
          incount=incount+x(3,k)
        end if
204      continue
```

```
c          write(0,*)"gamma counts",igcount
c          write(0,*)"neutron counts",incount
c          pause 5

205        continue
      ratiot=float(igcount)/float(incount)! total ratio of gammas to neutrons
      write(3,*) "-------------------------------------------"
      write(0,*) "Bayse's classifier results:"
      write(3,*) "Bayse's classifier results:"
      write(0,*) "Total Photon number: ",igcount
      write(3,*) "Total Photon number: ",igcount
      write(0,*) "Total neutron number: ",incount
      write(3,*) "Total neutron number: ",incount
      write(0,*) "Total Ratio of gammas to neutrons:",ratiot
      write(3,*) "Total Ratio of gammas to neutrons:",ratiot
      write(3,*) "-------------------------------------------"
c-------------------------------------------------------------
         go to 100     ! stop
CC*****************************************************************************
c  For trainning
c
40     read(1,*) k1,j2   ! k1---dimension number (2 or 3)
      read(1,*) c,cl   ! weight correction constant
      read(1,*)
      write(3,*) "   Weight correction constant=",c
      do 45 j=1,j2      ! j2---number of types (2 or 3)
       read(1,*) nj(j)    ! nj(j)---# of data in type j
       do 46 k=1,nj(j)
        read(1,*) (y(i,j,k),i=1,k1)
        write(0,*) j,(y(i,j,k),i=1,k1)   ! check
46      continue
       pause 0
45      continue
c  ____________________________________________________________
c   Randomixe the weights W(i,j)
      write(3,*) "***************************************"
      write(3,*) "     original randomized weight "
      write(3,*) "***************************************"
      do 50 j=1,4
      do 55 i=1,4
c       call random(w(i,j))
      w(i,j)=0.0
55     continue
      write(0,*) "output node=",j
```

```
      write(0,*) "    weight:",(w(i,j),i=1,4)
      write(3,*) "output node=",j
      write(3,*) "    weight:",(w(i,j),i=1,4)
50     continue                    ! output weight
      pause 0
      if(j2.eq.2) go to 56
      j22=j2
      go to 58
56     j22=j2-1          ! classify 2 types----one decision node
58     continue
c*************************************************************
c  Start correction
c
      do 60 j=1,j22     ! output nodes # j
59       icount=0
       do 62 jj=1,j2       ! type # of data jj
        do 64 k=1,nj(jj)     ! # of data in type jj
        df=w(1,j)
         do 66 i=1,k1      !k1 ----dimension of data( 2 or 3)
         df=df+y(i,jj,k)*w(i+1,j)     ! calculate decision function
 66       continue
         if(jj.eq.j) go to 73        ! same type !
         if(df.lt.0.0) go to 64       ! no correction go to next data
        dnm=1               ! dnm=||y||**2
         do 75 i=1,k1
         dnm=dnm+y(i,jj,k)**2
 75       continue
          sum=df
         c=cl*sum/dnm
         if(c.eq.0) c=cl
c         ic=int(c)+1
         w(1,j)=w(1,j)-c
           do 72 i=1,k1
           ii=i+1
           w(ii,j)=w(ii,j)-c*y(i,jj,k)
 72       continue
         Icount=Icount+1
         go to 64          ! test next data.
 73       if(df.gt.0.0) go to 64       ! no correction go to next data
        dnm=1               ! dnm=||y||**2
         do 76 i=1,k1
         dnm=dnm+y(i,jj,k)**2
 76       continue
          sum=abs(df)
```

```
        c=cl*sum/dnm
        if(c.eq.0) c=cl
c        ic=int(c)+1
        w(1,j)=w(1,j)+c
         do 74 i=1,k1
         ii=i+1
         w(ii,j)=w(ii,j)+c*y(i,jj,k)
 74       continue
        Icount=Icount+1    ! cumulate correction #
 64      continue          ! test next data.
 62      continue          ! finish check one type of data
        iter(j)=iter(j)+1  ! finish check one output node
c        write(0,*) j,iter(j)
c        pause 2
        if(iter(j).eq.2000000) go to 101
        if(icount.ne.0) go to 59    ! continue at the same node
cc
c--------correction # of  zero  means convergence!-----------
cc
        write(3,*)
        write(3,*) "-------------------------------------------"
        write(0,61) j,iter(j)
        write(3,61) j,iter(j)
        write(3,*) "-------------------------------------------"
        write(3,*)
61       format(1x,'weight correction number in the test node #',
   1i2/' is  0   after iteration of ',i8)
        go to 60
101       write(3,*) "!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!"
        write(3,*) j," diverge after iteration of",iter(j)
        write(3,*) "!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!"
        write(3,*)
67       write(0,*) "!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!"
        write(0,*) j," diverge after iteration of",iter(j)
        write(0,*) "!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!"
        write(0,*)
c      start next node
c       pause 2
60      continue
c
c*****************************************************************
c  output weight
c
c
```

```
      write(3,*) "****************************************"
      write(3,*) "      The weight after training"
      write(3,*) "****************************************"
      do 80 j=1,3         !decision nodes
      write(0,*) "output node:",j
      write(0,*) "     weight:",(w(i,j),i=1,k1+1)
      write(3,*) "output node:",j
      write(3,*) "     weight:",(w(i,j),i=1,k1+1)
      write(2,*) (w(i,j),i=1,k1+1)
80     continue
100    continue
      close(1)
      close(2)
      close(3)
      end
c*************************************************************************
**
c
      subroutine simpson(sinteg,a,b,idd,esn)  ! simpson adaptive integration
      dimension v(8),tol(15),ai(15),hi(15),fai(15),fci(15),fbi(15)
      dimension si(15),l(15)
      sinteg=0.0
      i=1
      tol(i) = 10*esn      ! esn is tolerence
      ai(i) = a
      hi(i)=(b-a)/2
      if(idd .eq.1) then
       if(a .le. 1.5) then
         fai(i)=fun(a)
         fci(i)=fun(a+hi(i))
         fbi(i)=fun(b)
       else
         fai(i)=fun1(a)
         fci(i)=fun1(a+hi(i))
         fbi(i)=fun1(b)
       end if
      else
       fai(i)=fun(a)
       fci(i)=fun(a+hi(i))
       fbi(i)=fun(b)
      end if

      si(i)=hi(i)*(fai(i)+4*fci(i)+fbi(i))/3
      l(i)=1
```

```
      ll=1
10    if(i.le.0) go to 30
      if (idd.eq.1) then
       if( a .lt. 1.5) then
         fd=fun(ai(i)+hi(i)/2)
         fe=fun(ai(i)+1.5*hi(i))
       else
         fd=fun1(ai(i)+hi(i)/2)
         fe=fun1(ai(i)+1.5*hi(i))
       end if
      else
       fd=fun(ai(i)+hi(i)/2)
       fe=fun(ai(i)+1.5*hi(i))
      end if

      s1=hi(i)*(fai(i)+4*fd+fci(i))/6
      s2=hi(i)*(fci(i)+4*fe+fbi(i))/6
      v(1)=ai(i)
      v(2)=fai(i)
      v(3)=fci(i)
      v(4)=fbi(i)
      v(5)=hi(i)
      v(6)=tol(i)
      v(7)=si(i)
      v(8)=l(i)
      i=i-1
      error=abs(s1+s2-v(7))
      if(error.lt.v(6)) go to 20
      if(v(8).ge.15) go to 1010    !error message
      i=i+1                ! add one level
      ai(i)=v(1)+v(5)
      fai(i)=v(3)
      fci(i)=fe
      fbi(i)=v(4)
      hi(i)=v(5)/2
      tol(i)=v(6)/2
      si(i)=s2
      l(i)=v(8)+1
      i=i+1          !data for left half subinterval
      ai(i)=v(1)
      fai(i)=v(2)
      fci(i)=fd
      fbi(i)=v(3)
      hi(i)=hi(i-1)
```

```
      tol(i)=tol(i-1)
      si(i)=s1
      l(i)=l(i-1)
      ll=ll+1
      go to 10
20     sinteg=sinteg+s1+s2
      go to 10
30     continue
      n=4*ll       !n is number of subintervals used
c      write(0,*) 'Integral sum is',sinteg
c      write(0,*) 'Number of subintervals used is ', n
      go to 1015
1010   write(0,*) 'Maximum spliting level(15) exceeded'
1015   continue
      return
      end
c
      function fun(z)
      x=exp(-0.5*z**2)
      fun=x/2.50663        ! 2.50663=sqrt(2*3.1416)
      end
      function fun1(z)
      fun1=0.12952*exp(-(z-1.5))  !0.12952=exp(-(1.5^2)/2)/sqrt(2*3.1416)
c      fun1=0.1942*exp(-(z-1.2))
      end
```

dat1.dat(for training data file example)

```
2 2     !!!for training of percepn1.for  # of dimension and # of classes
2 2          correction constant and landa ( learning factor )
x         y     z          <-----X is time information; y is energy information; Z is class.
17                            <------number of class 1
3.0445    1.0   1
3.045     1.2   1
3.046     1.5   1
3.047     1.7   1
3.049     2.0   1
3.051     2.5   1
3.053     3.0   1
3.055     3.5   1
3.057     4.0   1
3.059     4.5   1
3.060     4.7   1
3.062     5.0   1
3.066     6.0   1
3.068     6.5   1
3.070     7.0   1
3.072     7.5   1
3.075     8.0   1
17                        <----Number of class 2
3.0455    1.0    2
3.046     1.2    2
3.047     1.5    2
3.048     1.7    2
3.050     2.0    2
3.052     2.5    2
3.054     3.0    2
3.056     3.5    2
3.058     4.0    2
3.060     4.5    2
3.061     4.7    2
3.063     5.0    2
3.067     6.0    2
3.069     6.5    2
3.071     7.0    2
3.073     7.5    2
3.076     8.0    2
```

out.dat (one of output files of training)

```
****************************************
  training with dat1.dat
****************************************

   Fractional Correction Algorithm

  Weight correction constant=      2.000000
**************************************
   original randomized weight
**************************************
output node=        1
   weight:  0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
output node=        2
   weight:  0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
output node=        3
   weight:  0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
output node=        4
   weight:  0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00

-------------------------------------------
weight correction number in the test node # 1
 is  0   after iteration of   16942
-------------------------------------------

**************************************
    The weight after training
**************************************
output node:        1
   weight:    17.593400      -5.786346   2.498146E-02
output node:        2
   weight:  0.000000E+00   0.000000E+00   0.000000E+00
output node:        3
   weight:  0.000000E+00   0.000000E+00   0.000000E+00
```

weit.dat

It stores the weight for future use, and is automatically produced by training. For a perceptron classifier, only the first line is required. The second and third lines, which store FWHM information, are required by the Bayes classifier.

17.593400 -5.786346 2.498146E-02 0.0 ←weight vector
.00878 .01728 .32557 .12527 .00394 .00422 1.84299 1.534 ← FWHM for gamma
.02181 .06629 .17023 0.0 .00651 .02231 2.22233 0.0 ←FWHM for neutron

y0(1) a(1) t(1) x0(1) y0(2) A(2) t(2) x0(2) ← Data input format for FWHM
---------- ----------
low energy region(1) high energy region(2)

y=y0(j)+A(j)*exp(-(x-x0(j))/t(j))

x related to energy; y related to FWHM at x.

t8.dat (example for test data file)

```
38 3 1                          ←# of data, dimension and # of decision(1 for 2 dimension
problem)
Chn  energy(v)  Counts          ←  !!!! This is for testing, not for training.
2.971  0.44      7                    Chn = channels/1000.
2.972  0.44     10
2.973  0.44      3
2.974  0.44     11
2.975 0.44       6
2.976  0.44      7
2.977  0.44     10
2.978  0.44     12
2.979  0.44      7
2.980  0.44      4
2.981  0.44      6
2.982  0.44     13
2.983  0.44     11
2.984  0.44     16
2.985  0.44     23
2.986  0.44     33
2.987  0.44     25
2.988  0.44     29
2.989  0.44     48
2.990  0.44     42
2.991  0.44     68
2.992  0.44     74
2.993  0.44     79
2.994  0.44    124
2.995  0.44    154
2.996  0.44    190
2.997  0.44    274
2.998  0.44    398
2.999  0.44    685
3.000  0.44   1111
3.001  0.44   1721
3.002  0.44   2726
3.003  0.44   4033
3.004  0.44   5567
3.005  0.44   7455
3.006  0.44   9309
3.007  0.44  11449
3.008  0.44  13252
```

out1.dat

The following "out1.dat" is a output data file for the testing data t8.dat.

```
*****************************************
   60 keV(Ee) t8.dat
*****************************************

    Fractional Correction Algorithm

----------Tested data----------

     data(i)      x(i)          y(i)        z(i)

-----------Classification Results----------
    data(i)  decision(1)        d(2)        d(3)
-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
Perceptron classification results******
Total Photon number:     270113
Total neutron number:     73829
Total Ratio of photons to neutrons:      3.658630
-------------------------------------------
-------------------------------------------
Bayse's classifier results:
Total Photon number:     271464
Total neutron number:     72478
Total Ratio of photons to neutrons:      3.745468
-------------------------------------------
```

Appendix E

Program Files for the Back-Propagation Neural Network Classifier

nnwm1.for(source file)

```
c*********************************************************************
c     nnwm1.for (source file for the back-propagation neural network classifier)
c
c                by Zhong Cao
c
c                March, 1997
c
c     Neuron Network program with Back Propagation Algorithm with momentum
c       acceleration
c      During training, the code automatically produces weight file (nnwm1.wwt ) for
c       future classification.  It also create file " nnwm1.out " to store the training
c       message.
c
c      Need input data file for testing, computer automatically load weight file for user,
c       the results will show both on the screen and in file "nnwm1.out".
c
c             Usage:
c                       fl  nnwm1.for          <- for compile
c                       nnwm1 dat4.dat         <- for training, dat4.dat is any training file.
c                       nnwm1  nnt8.dat        <- for testing, nnt8,dat is any testing file.
c
c
c*********************************************************************
c
c
      dimension x(3,1200),px2(1200),px3(1200),T(2,1200),dtq(1200)
      dimension wp(4,50),wpo(4,50),wq(50,2),wqo(50,2),cc(1200)
      character*1 ans
      open(1,file=' ')
      open(2,file='nnwm1.out')
      open(3,file='nnwm1.wwt')        ! store weight data
      write(0,*) "training( y/n )----n means testing "
      read(0,5) ans
5     format(1a)
      if(ans.ne.'y') go to 60      ! for testing
c---------------------------------
c   start training
c---------------------------------
      read(1,*) kx1,kx2,kx3,a,rl      ! PE # of 1st, 2nd, and 3rd layer
      read(1,*) nx,rms0         ! a---alpha in sigmoid transfer function
      kk=kx1+1                ! kx1+1 PE # in the first layer
```

```
      do 8 j=1,nx              ! nx  # of data and target
      read(1,*) (x(i,j),i=1,kk)        ! tolerance rms0
      read(1,*) (t(i,j),i=1,kx3)         ! rl--learning rate
      write(0,*) (x(i,j),i=1,kk)        ! tolerance rms0
      write(0,*) (t(i,j),i=1,kx3)         ! rl--learning rate
      cc(j)=x(kk,j)
8     continue
c     pause 1
      write(0,*) " Input the acceleration factor"
      read(0,*) acc
c
c-----------------------------------------------------------------
c     start training
      iii=1
      do 6 i=1,nx
      x(kk,i)=1
6     continue
      write(0,*)"Using old weights? y/n"
      read(0,5) ans1
      if(ans1.eq.'y') go to 13
      do 10 i=1,kk          ! randomize the initial weight withi+-0.1
        do 15 j=1,kx2
        call random(wp(i,j))
        wp(i,j)=0.1*(wp(i,j)-0.5)
 15   continue
 10   continue
      do 20 i=1,kx2
        do 16 j=1,kx3
        call random(wq(i,j))
        wq(i,j)=0.1*(wq(i,j)-0.5)
 16   continue
 20   continue
      go to 9
13    continue                    ! Using old weight!
      read(3,*)
      do 11 i=1,kk
11    read(3,*) (wp(i,j),j=1,kx2)
      do 12 i=1,kx2
      read(3,*) (wq(i,j),j=1,kx3)
12    continue
9     continue
c ----------------------------------------------------------------
22    rms=0
    call gettim(ihr1,imin1,isec1,i100th1)
```

```
      Icount=Icount+1
      do 50 ii=1,nx  ! calculate sigmoid transfer function at all PEs
      do 26 j=1,kx2                ! 2nd layer
       px2(j)=0
       do 25 i=1,kk
 25    px2(j)=px2(j)+x(i,ii)*wp(i,j)
       px2(j)=1+exp(-a*px2(j))
       px2(j)=1/px2(j)
 26   continue
      do 30 j=1,kx3                ! 3rd layer
       px3(j)=0
       do 28 i=1,kx2
 28    px3(j)=px3(j)+px2(i)*wq(i,j)
      px3(j)=1+exp(-a*px3(j))
      px3(j)=1/px3(j)
  30  continue
c-----------------------------------------------------------------
c calculate error at all output layer (3 rd) and rms
c
      do 35 i=1,kx3
      err=t(i,ii)-px3(i)
      rms=(err**2)*cc(ii) +rms          !x(3,ii) is counts of data ii
      dtq(i)=2*a*err*px3(i)*(1-px3(i))     ! calculate delta(pq)
35     continue
      cnumb=cnumb+cc(ii)
c-----------------------------------------------------------------
c   Check if need mometum acceleration
c-----------------------------------------------------------------
c       if(rms.lt.rmss-0.001) go to 43   ! no acceleration!
c       irm=irm+1
      if(imm.eq.0) go to 43    ! no acceleration
      do 46 i=1,kk           ! momentum acceleration
      do 47 j=1,kx2
      wp(i,j)=wp(i,j)+acc*wpo(i,j) ! Acceleration for hidden layers
  47   continue
  46   continue
      do 48 j=1,kx3
       do 49 i=1,kx2              ! Acceleration for output layers
 49    wq(i,j)=wq(i,j)+acc*wqo(i,j)        ! rl is learning rate
 48    continue
      imm=0
 43    continue
c-----------------------------------------
c    Check if meet the tolerance rms0
```

```
c
      do 45 j=1,kx2
       dtp=0
       do 41 k=1,kx3                    ! calculate delta(hp)
       dd=a*px2(j)*(1-px2(j))*dtq(k)*wq(j,k)
       dtp=dtp+dd
 41    continue
      do 42 i=1,kk
      wpo(i,j)=rl*x(i,ii)*dtp
      wp(i,j)=wp(i,j)+wpo(i,j)   ! Correct weights of hidden layers
 42    continue
 45    continue
      do 40 j=1,kx3
       do 38 i=1,kx2              ! Correct weights of output layers
       wqo(i,j)=rl*px2(i)*dtq(j)
 38    wq(i,j)=wq(i,j)+wqo(i,j)     ! rl is learning rate
 40    continue
 50    continue
      ij=icount-1000*iii
      rms=sqrt(rms/cnumb/kx3)
      if(ij.ne.0) go to 77
c      rms1=sqrt(rms/nx/kx3)
      write(0,*) iii," X 1000",rms
      write(0,*) (px3(j),j=1,kx3)
      iii=iii+1
 77    if(iii.eq.880000) go to 59
      rms1=0.9999*rmss
      rms2=1.0001*rmss
      if(rms.gt.rms1 .and. rms.lt. rms2) imm=2
           !acceleration index
      rmss=rms
      if(rms.gt.rms0) go to 22
 59    write(0,52) rms,Icount
 52    format(1x,'reach the tolerance:',e10.3,'  iteration # ',i10)
     call gettim(ihr2,imin2,isec2,i100th2)
     ih=ihr2-ihr1
     imi=imin2-imin1
     ise=isec2-isec1
     i100=i100th2-i100th1
c     write(2,139)
c139    format(2x,'*****Computer running time')
     write(0,*) "running time",ih,imi,ise,i100
     write(2,130) ih,imi,ise,i100
 130    format(2x,'running time:', 4(I4,2x),'(100th second)',/)
```

```
      write(0,*) "Try a smaller tolerance? y/n"
      read(0,5) ans
      if(ans.ne.'y') go to 51
      write(0,*) "input new tolerance rms0 "
      read(0,*) rms0
      go to 22
51    Write(2,*)"*****************************************************"
      write(2,*)"   Back Propagation with Momentum Acceleration "
      write(2,*)"*****************************************************"
      write(2,*)"--------Training Parameters----------"
      write(2,86) kx1,kx2,kx3,acc,a,rl,nx
86    format(1x,'  # of 1st PE=',i3,'     # of 2nd PE=',i3/
     1'   # of 3rd PE=',i3,'   Acceleration factor=',f6.3
     2/'Alpha :',f6.2,'  learning constant:',f6.2,'   # of data=',i4)
      write(2,*)"-------------------------------------------------"
      write(2,52) rms,Icount
      write(2,*)"-----------------------------------------------"
      write(2,*) " second layer weights output "
      write(2,*)"-----------------------------------------------"
      do 53 i=1,kk
53      write(2,*) (wp(i,j),j=1,kx2)
      write(2,*)"-----------------------------------------------"
      write(2,*) " third layer weights output"
      write(2,*)"-----------------------------------------------"
      do 54 i=1,kx2
54      write(2,*) (wq(i,j),j=1,kx3)
      write(3,*) kk,kx2,kx3,a        ! store the weight on file 3
      do 57 i=1,kk
57      write(3,*) (wp(i,j),j=1,kx2)
      do 58 i=1,kx2
58      write(3,*) (wq(i,j),j=1,kx3)
      go to 100
c
c*****************************************************************
c    testing part  !!!
c
60     continue
      read(1,*) kx1,kx2,kx3,a      ! PE # of 1st, 2nd, and 3rd layer
      read(1,*) nx        ! a---alpha in sigmoid transfer function
      read(3,*) kk,kx2,kx3,a
      do 61 j=1,nx              ! nx  # of data
      read(1,*) (x(i,j),i=1,kk)
      cc(j)=x(kk,j)
c       read(1,*)          ! skip target function value
```

```
      x(kk,j)=1
61     continue
      do 63 i=1,kk
63     read(3,*) (wp(i,j),j=1,kx2)
      do 64 i=1,kx2
      read(3,*) (wq(i,j),j=1,kx3)
64     continue
c      do 65 i=1,nx
c65       x(kk,i)=1
c--------------------------------------
cc  start calculation
c
      do 70 ii=1,nx
      do 76 j=1,kx2              ! 2nd layer
      px2(j)=0
      do 75 i=1,kk
75     px2(j)=px2(j)+x(i,ii)*wp(i,j)
      px2(j)=1+exp(-a*px2(j))
      px2(j)=1/px2(j)
c      write(0,*) px2(j)
76     continue
      do 80 j=1,kx3              ! 3rd layer
       px3(j)=0
      do 78 i=1,kx2
78     px3(j)=px3(j)+px2(i)*wq(i,j)
c      write(0,*) px3(j)
      px3(j)=1+exp(-a*px3(j))
c      write(0,*) px3(j)
      px3(j)=1/px3(j)
      t(j,ii)=px3(j)
80     continue
70     continue
      write(2,96) kx1,kx2,kx3,a,rl,nx
96    format(1x,'  # of 1st PE=',i3,'     # of 2nd PE=',i3/
     1'   # of 3rd PE=',i3
     1/'Alpha :',f6.2,'  learning constant:',f6.2,'   # of data=',i4)
       write(2,*)"------------------------------------------------"
       cnumg=0.0
       cnumn=0.0
       do 85 j=1,nx
c      write(0,*) j," N.N. input  "
c      write(2,*) j," N.N. input  "
c      write(0,*) (x(i,j),i=1,kx1),cc(j)
c      write(2,*) (x(i,j),i=1,kx1),cc(j)     !x(3,j) is counts of j data
```

```
c       write(2,*) t(1,j),t(2,j)
        IF (t(1,j).le.t(2,j)) Then
              cnumg=cnumg+cc(j)
        ELSE
              cnumn=cnumn+cc(j)
        END IF
85      continue
        if(cnumn .eq. 0.0) cnumn=1.0
        ratio=cnumg/cnumn
        write(0,*) " Total gamma number:",cnumg
        write(2,*) " Total gamma number:",cnumg
        write(0,*) " Total neutron number:",cnumn
        write(2,*) " Total neutron number:",cnumn
        write(0,*) " Ratio of gamma-rays to neutrons:",ratio
        write(2,*) " Ratio of gamma-rays to neutrons:",ratio
100     continue
        close(1)
        close(2)
        close(3)
        end
```

dat4.dat(training data example)

```
2 5 2 1 1 ← # of input, hidden and output neurodes; alpha of sigmoid, and learning rate.
34  0.001      ← # of data, tolerance;
.144   .10  1
0.4 0.6                    ←!!! For training only.
.145   .12  1
0.4 0.6
.146   .15  1
0.4 0.6
.147   .17  1              ←  Time; energy; counts
0.4 0.6                    ←  Target value for class 1 (0.4, 0.6)
.149   .20  1
0.4 0.6
.151   .25  1
0.4 0.6
.153   .30  1
0.4 0.6
.155   .35  1
0.4 0.6
.157   .40  1
0.4 0.6
.159   .45  1
0.4 0.6
.160   .47  1
0.4 0.6
.162   .50  1
0.4 0.6
.166   .60  1
0.4 0.6
.168   .65  1
0.4 0.6
.170   .70  1
0.4 0.6
.172   .75  1
0.4 0.6
.175   .80  1
0.4 0.6
.146   .10   1        <-  Time; energy; counts
0.6 0.4                   <-  Target value for class 2 (0.6, 0.4)
.147   .12    1
0.6 0.4
.148   .15    1
0.6 0.4
```

.149 .17 1
0.6 0.4
.151 .20 1
0.6 0.4
.153 .25 1
0.6 0.4
.155 .30 1
0.6 0.4
.157 .35 1
0.6 0.4
.159 .40 1
0.6 0.4
.161 .45 1
0.6 0.4
.162 .47 1
0.6 0.4
.164 .50 1
0.6 0.4
.168 .60 1
0.6 0.4
.170 .65 1
0.6 0.4
.172 .70 1
0.6 0.4
.174 .75 1
0.6 0.4
.177 .80 1
0.6 0.4

nnt8.dat (example for the test data)

2 5 2 1.0 ← # of input, hidden, output neurodes; alpha of sigmoid function
27 ← # of data for testing

7.099986E-02	4.400000E-02	7.000000	←-time, energy, counts
7.199979E-02	4.400000E-02	10.000000	
7.299995E-02	4.400000E-02	3.000000	
7.399988E-02	4.400000E-02	11.000000	
7.499981E-02	4.400000E-02	6.000000	
7.599998E-02	4.400000E-02	7.000000	
7.699990E-02	4.400000E-02	10.000000	
7.799983E-02	4.400000E-02	12.000000	
7.900000E-02	4.400000E-02	7.000000	
7.999992E-02	4.400000E-02	4.000000	
8.099985E-02	4.400000E-02	6.000000	
8.200002E-02	4.400000E-02	13.000000	
8.299994E-02	4.400000E-02	11.000000	
8.399987E-02	4.400000E-02	16.000000	
8.499980E-02	4.400000E-02	23.000000	
8.599997E-02	4.400000E-02	33.000000	
8.699989E-02	4.400000E-02	25.000000	
8.799982E-02	4.400000E-02	29.000000	
8.899999E-02	4.400000E-02	48.000000	
8.999991E-02	4.400000E-02	42.0000	
9.099984E-02	4.400000E-02	68.000000	
9.200001E-02	4.400000E-02	74.000000	
1.199999E-01	4.400000E-02	8496.000000	
1.789999E-01	4.400000E-02	2134.000000	
1.799998E-01	4.400000E-02	2034.000000	
1.869998E-01	4.400000E-02	1394.000	
1.880000E-01	4.400000E-02	1269.000000	

nnwm1.wwt
(weight file examples for a 3-5-2 neural network, automatically generated in the training).

```
3  5  2  1.0 ← neurode # of input, hidden and output layers; alpha value in the sigmoid.
    -3.814443  -4.389702E-01    -41.698470     -8.466096      ←---Weight vector
 -3.393160E-01
    -1.101436     1.481905     1.586169   4.738604E-01
    -1.337924
    -1.592355    -2.646218     4.836476  -5.884265E-01
    -2.985474
     1.619361    -1.616149
     1.154873    -1.154046
    -2.308885     2.308927
     2.795463    -2.796900
  6.475934E-01  -6.519461E-01
```

nnwm1.out (example of the classification output file)

```
# of 1st PE=  2     # of 2nd PE=  5
   # of 3rd PE=  2
Alpha :  1.00   learning constant:   .00    # of  data =  26
 ---------------------------------------------------
  Total gamma number:    8961.000000
  Total neutron number:    5562.000000
  Ratio of gamma-rays to neutrons:       1.611111
```

Appendix F

Program Files for the Statistical Models of the Zero-Crossing and the Charge Comparison Methods

zc.for (source code for the statistical model of the zero-crossing method)

```
c**********************************************************************
c   zc.for
c
c     by Zhong Cao
c
c        April, 1997
c
c        zc.for  is a code for the statistical model of the zero-crossing method.
c            A code to calculate the PSD time distribution of scintillation events.
c
c        Input data file contains:
c                Total photoelectron number (n0),
c                Up to four decay components,
c                CFD value and time range,
c                Trasit time spread.
c
c        Usage:
c                fl zc.for                ← for compile
c                zc  input.dat    ←for running code,  input.dat is any input data file
c                                          , e.g. chargn.dat, chargs.dat and chargb.dat.
c
c**********************************************************************

        real*8 a,fgj,fnj,fgj_1,fnj_1
        dimension zumg(2500),zumn(2500)
        common sigpmt,id,tg1,fg1,tg2,fg2,tn1,fn1,tn2,fn2,n0
        common tg3,fg3,tg4,fg4,tn3,fn3,tn4,fn4
        common sigg,rng0,signn,rnn0
        open(1,file='chargs.dat')   ! for input parameters
        open(2,file=' ')    ! for output data

        read(1,*) sigpmt,maxtime,cfd
                              !FWHM of PMT transit time spread ( in ns )
                               !constant fraction discrimination factor
        read(1,*) indexg    !number of scintillation components for photons
c       write(0,*) indexg
        if (indexg .eq. 1) then
                read(1,*) Tg1,fg1
        else if (indexg .eq. 2) then
                read(1,*) Tg1,fg1,tg2,fg2
```

```
      else if (indexg .eq. 3) then
            read(1,*) Tg1,fg1,tg2,fg2,tg3,fg3
      else
            read(1,*) Tg1,fg1,tg2,fg2,tg3,fg3,tg4,fg4
      end if
      read(1,*) indexn    !number of scintillation components for neutron
c     write(0,*) indexn,tg1,tn1
      if (indexn .eq. 1) then
            read(1,*) Tn1,fn1
      else if (indexn .eq. 2) then
            read(1,*) tn1,fn1,tn2,fn2
      else if (indexn .eq. 3) then
            read(1,*) Tn1,fn1,tn2,fn2,tn3,fn3
      else
            read(1,*) Tn1,fn1,tn2,fn2,tn3,fn3,tn4,fn4
      end if
      sigpmt=sigpmt/2.35
      write(0,*) "input new cfd"
      read(0,*) cfd

10    write(0,*) "Enter average number of photoelectrons, -1 stop"
      read(0,*) n0
      if(n0 .le. 0) go to 100         ! stop
      write(0,*) "  Total# and cfd factor",n0,cfd
      write(2,*) "  Total# and cfd factor",n0,cfd
c     write(0,*) " t #, 3( Photons(t), Neutrons(t))"
      write(2,*) " t   Photons   Neutrons )" ,n0,maxtime,cfd
      smaxg=0.0
      smaxn=0.0

      do 50 it=1,maxtime

      sumg=0.0
      sumn=0.0
      sumg1=0.0
      sumn1=0.0
      t=float(it)

      id=1          !for photons calculate p(t)
      pg1=fun(t)
      id=0          !for neutrons
      pn1=fun(t)
c     write(0,*) "t, pg(t), pn(t) ",t,pg1,pn1
      write(0,*) t
```

```
      id=1                ! for phtons   Fg(t)
      call simpson(fgfast,0.0,t)
      id=0                ! for netrons  Fn(t)
      call simpson(fnfast,0.0,t)

ccc   write(0,*)"F(t)g, F(t)n",fgfast,fnfast

      ffgfast=1-fgfast    ! for photns 1-Fg(t)
      ffnfast=1-fnfast    ! for neutrons 1-Fn(t)
      sigman=sqrt(n0)     ! total number fluctuation
      nd=int(sigman)
      nd1=n0-3*nd           !take +-3*sigma width ( 0.997)
      nd2=n0+3*nd


      if(n0 .gt. 100 ) go to 55    ! for n0 > 100 case, using Gaussian formula


      do 30  n=nd1,nd2      !consider fluctuation of total number
      fgj=1.0
      fnj=1.0
      fgj_1=1.0
      fnj_1=1.0
      j=int(cfd*n)    ! cfd= .9, or .8, .7, .6, .5, .4, .3,
      rjj=float(n)
      id=3
      pn=fun(rjj)  ! calculate p(n)


      call factorl(n,j,a)          !n must .le. 1000 to prevent upper overflow

      a=pn*a*j

            do 15 i=1,j-1  ! calculate( Fg(t))**(j-1)) for cfd (j) case
            fgj=fgj*fgfast
            if(fgj .le. 1e-299) go to 16
 15       continue
 16       do 17 i=1,j-1   ! calculate( Fn(t))**(j-1))
            fnj=fnj*fnfast
            if(fnj .le. 1e-299) go to 18
 17       continue
 18       continue

c     write(0,*) "fgj, fnj", fgj,fnj
```

```
              do 32 k=1,n-j     ! calculate (1-F(t))**(n-j) for photons
              fgj_1=fgj_1*ffgfast
              if(fgj_1 .le.1e-299) go to 34
 32   continue
 34   continue
              do 35 k=1,n-j     ! ... for neutrons
              fnj_1=fnj_1*ffnfast
              if(fnj_1 .le.1e-299) go to 36
 35   continue
 36   continue
ccc     write(0,*) "(1-F(t))**(n-j)",fnj_1,fgj_1
       sumg=sumg+a*fgj_1*fgj
       sumn=sumn+a*fnj_1*fnj


ccc     write(0,*) "sumg, sumn",sumg,sumn
 30   continue
       sumg=sumg*pg1
       sumn=sumn*pn1
       go to 77       ! go to next time value

 55   continue
       do 65  n=nd1,nd2
       j=int(cfd*n)
       rjj=float(n)
       id=3
       pn=fun(rjj)       ! calculate p(n)
       rng0=n*fgfast     ! calculate average of n
       rnn0=n*fnfast       ! for neutrons
       sigg=sqrt(rng0*ffgfast)  ! calculate variance for photons
       signn=sqrt(rnn0*ffnfast)    ! for neutrons
       rjj=float(j)
       rr1=abs(rjj - rng0)
       rr2=abs(rjj - rnn0)
       rg3=5*sigg
       rn3=5*signn
       if(rr1 .ge. rg3 .and. rr2 .ge.rn3) then
              go to 65
       else if(rr1 .lt. rg3 .and. rr2 .ge. rn3) then
              id=4
              sumg=sumg+pn*fun(rjj)*rjj
              sumn= sumn
       else if(rr1 .ge. rg3 .and. rr2 .lt. rn3) then
```

```
                id=5
                sumn=sumn+pn*fun(rjj)*rjj
                sumg= sumg
        else
                id=4
                sumg=sumg+ pn*fun(rjj)*rjj
                id=5
                sumn=sumn + pn * fun(rjj)*rjj
        end if
65      continue
        sumg=sumg*pg1/fgfast
        sumn=sumn*pn1/fnfast

77      continue
write(2,*) it,sumg,sumn   ! output results
        zumg(it)=sumg
        zumn(it)=sumn
        if (sumg .gt. smaxg) then
                smaxg=sumg
                itg=it
        else if ( sumn .gt. smaxn) then
                smaxn=sumn
                itn=it
        else
        end if
50    continue

        h1=zumg(itg)/2.0
        h2=zumn(itn)/2.0
        rt1=itn-itg         !peak separation
c       rt2=itn1-itg1
        do 80 it=1,maxtime-1
        if(zumg(it) .lt. h1 .and. zumg(it+1) .gt. h1) t11=it
        if(zumg(it) .gt. h1 .and. zumg(it+1) .lt. h1) t12=it
80      continue
        fwhmg=t12-t11
        do 81 it=1,maxtime-1
        if(zumn(it) .lt. h2 .and. zumn(it+1) .gt. h2) t11=it
        if(zumn(it) .gt. h2 .and. zumn(it+1) .lt. h2) t12=it
81      continue
        fwhmn=t12-t11
        fom=rt1/(fwhmg+fwhmn)
        write(0,*) "cfd, and total number n0",cfd,n0
        write(0,*) "Maximum at peaks",zumg(itg),zumn(itn)
```

```
        write(0,*) "peak(g),peak(n),fwhm(g),fwhm(n),figure of merit"
        write(0,*) itg,itn,fwhmg,fwhmn,fom
        go to 10    ! back to beginging

100     close(1)
        close(2)
        stop
        end

c-------------------------------------------------------------
c  Subroutines and functions
c-------------------------------------------------------------

        subroutine simpson(sinteg,a,b)   ! composite Simpson integration
        h0=b-a

        if (h0 .gt. 50.0 ) then
                nn=200          ! nn must be even
        else if (h0 .gt. 100.0) then
                nn=1000
        else
                nn=100
        end if

        h=h0/nn
        f1=fun(a)+fun(b)
        f2=0.0
        f3=0.0
        do 10 i=2,nn-2,2
                f2=f2+fun(a+i*h)
10      continue
        do 20 i=1,nn-1,2
20      f3=f3+fun(a+i*h)
        sinteg=h*(f1+f2*2+f3*4)/3
        return
        end

c----------------- second simpson adaptive integration
      function fun(z)
      dimension v(8),tol(32),ai(32),hi(32),fai(32),fci(32)
      dimension si(32),l(32),fbi(32)
      common sigpmt,id

      sinteg=0.0
```

```
      i=1
      if(id .lt. 3) then
           sig=3*sigpmt
           a=z-sig
           if(a .lt. 0.0) a=0.0
           b=z+sig
      else
           a=z
           b=z+1
      end if

      tol(i) = 10*0.001      ! esn=0.01 is tolerance
      ai(i) = a
      hi(i)=(b-a)/2

         if(id .eq. 1) then
           fai(i)=fun1(a-z,z)
           fci(i)=fun1(a+hi(i)-z,z)
           fbi(i)=fun1(b-z,z)
         else if (id .eq. 0) then
           fai(i)=fun2(a-z,z)
           fci(i)=fun2(a+hi(i)-z,z)
           fbi(i)=fun2(b-z,z)
         else if (id .eq. 3) then
           fai(i)=fun3(a)
           fci(i)=fun3(a+hi(i))
           fbi(i)=fun3(b)
         else if (id .eq. 4) then
           fai(i)=fun4(a)
           fci(i)=fun4(a+hi(i))
           fbi(i)=fun4(b)
         else
           fai(i)=fun5(a)
           fci(i)=fun5(a+hi(i))
           fbi(i)=fun5(b)
         end if

      si(i)=hi(i)*(fai(i)+4*fci(i)+fbi(i))/3
      l(i)=1
      ll=1
10    if(i.le.0) go to 30

         if(id .eq. 1) then
           fd=fun1(ai(i)+hi(i)/2-z,z)
```

```
 fe=fun1(ai(i)+1.5*hi(i)-z,z)
else if (id .eq. 0) then
 fd=fun2(ai(i)+hi(i)/2-z,z)
 fe=fun2(ai(i)+1.5*hi(i)-z,z)
else if(id .eq. 3) then
 fd=fun3(ai(i)+hi(i)/2)
 fe=fun3(ai(i)+1.5*hi(i))
else if(id .eq. 4) then
 fd=fun4(ai(i)+hi(i)/2)
 fe=fun4(ai(i)+1.5*hi(i))
else
 fd=fun5(ai(i)+hi(i)/2)
 fe=fun5(ai(i)+1.5*hi(i))
end if

s1=hi(i)*(fai(i)+4*fd+fci(i))/6
s2=hi(i)*(fci(i)+4*fe+fbi(i))/6
v(1)=ai(i)
v(2)=fai(i)
v(3)=fci(i)
v(4)=fbi(i)
v(5)=hi(i)
v(6)=tol(i)
v(7)=si(i)
v(8)=l(i)
i=i-1
error=abs(s1+s2-v(7))
if(error.lt.v(6)) go to 20
if(v(8).ge.50) go to 1010    !error message
i=i+1                ! add one level
ai(i)=v(1)+v(5)
fai(i)=v(3)
fci(i)=fe
fbi(i)=v(4)
hi(i)=v(5)/2
tol(i)=v(6)/2
si(i)=s2
l(i)=v(8)+1
i=i+1          !data for left half subinterval
ai(i)=v(1)
fai(i)=v(2)
fci(i)=fd
fbi(i)=v(3)
hi(i)=hi(i-1)
```

```
      tol(i)=tol(i-1)
      si(i)=s1
      l(i)=l(i-1)
      ll=ll+1
      go to 10
20    sinteg=sinteg+s1+s2
      go to 10
30    continue
      n=4*ll      !n is number of subintervals used
c     write(0,*) 'Integral sum is ',sinteg
c     write(0,*) 'Number of subintervals used is *****', n
      fun=sinteg
      go to 1015
1010  write(0,*) 'Maximum level(32) exceeded in fun()**********'
      pause 0
1015  continue
      return
      end

      function fun1(z1,z0)          !for photons
      common sigpmt,id,tg1,fg1,tg2,fg2,tn1,fn1,tn2,fn2,n0
      common tg3,fg3,tg4,fg4,tn3,fn3,tn4,fn4
      x=exp(-0.5*z1**2/sigpmt**2)
      fun1=x/sqrt(2*3.1416)/sigpmt
      fast=fg1*exp(-z0/tg1)
      if(fg2.ne.0.0) fast=fast + fg2*exp(-z0/tg2)
      if(fg3.ne.0.0) slow=fg3*exp(-z0/tg3)
      if(fg4.ne.0.0) slow=slow +fg4*exp(-z0/tg4)
      fun1=fun1*(fast+slow)
      end

      function fun2(z2,z0)              !for neutrons
      common sigpmt,id,tg1,fg1,tg2,fg2,tn1,fn1,tn2,fn2,n0
      common tg3,fg3,tg4,fg4,tn3,fn3,tn4,fn4
      x=exp(-0.5*z2**2/sigpmt**2)
      fun2=x/sqrt(2*3.1416)/sigpmt
      fast=fn1*exp(-z0/tn1)
      if(fn2.ne.0.0) fast=fast + fn2*exp(-z0/tn2)
      if(fn3.ne.0.0) slow=fn3*exp(-z0/tn3)
      if(fn4.ne.0.0) slow=slow +fn4*exp(-z0/tn4)
      fun2=fun2*(fast+slow)
      end

      function fun3(z)              !for P(n) calculation
```

```
      common sigpmt,id,tg1,fg1,tg2,fg2,tn1,fn1,tn2,fn2,n0
      x=exp(-(z-n0)**2/(2*n0))
      fun3=x/sqrt(2*3.1416*n0)
      end

      function fun4(z)              !for neutrons
      common sigpmt,id,tg1,fg1,tg2,fg2,tn1,fn1,tn2,fn2,n0
      common tg3,fg3,tg4,fg4,tn3,fn3,tn4,fn4
      common sigg,rng0,signn,rnn0
      x=exp(-(z-rng0)**2/(2*sigg**2))
      fun4=x/sqrt(2*3.1416)/sigg
      end

      function fun5(z)              !for neutrons
      common sigpmt,id,tg1,fg1,tg2,fg2,tn1,fn1,tn2,fn2,n0
      common tg3,fg3,tg4,fg4,tn3,fn3,tn4,fn4
      common sigg,rng0,signn,rnn0
      x=exp(-(z-rnn0)**2/(2*signn**2))
      fun5=x/sqrt(2*3.1416)/signn
      end

      subroutine factorl(n2,j2,a)
      real*8 a
      a=float(n2)
      j1=j2
      j3=j2
      n1=n2
5     a=a*float(n1-1)
      b=float(n2-j1)
      if(b .le. 1.0) go to  6
      a=a/b
6     if(j3.le. 1) go to  10
      a=a/float(j3)
10    j3=j3-1
      j1=j1+1
      n1=n1-1
      if(n1.le.1) go to 20   !stop
      go to 5
20    return
      end
```

charg2.for (source code for the statistical model of the charge comparison method)

```
c**********************************************************************
c
c     charg2.for  (source code for the statistical model of charge comparison method)
c          For calculation the delayed component distribution of PSD scintillators using
c           the  charge comparison method.
c
c            By Zhong Cao     May 25, 1997
c
c       Input data file contains:
c               Total photoelectron number (n0),
c               Up to four decay components,
c               CFD value and time range,
c               Transit time spread.
c
c       Usage:
c               fl charg2.for            ← for compile
c               charg2  input.dat        ←for running code,  input.dat is any input data file
c                                              e.g. chargn.dat, chargs.dat and chargb.dat
c
c**********************************************************************
        dimension zumg(2000),zumn(2000)
        real*8 a,sumg,sumn,pgj,pnj,pgj_1,pnj_1
        common sigpmt,id,tg1,fg1,tg2,fg2,tn1,fn1,tn2,fn2,n0
        common tg3,fg3,tg4,fg4,tn3,fn3,tn4,fn4
        common sigg,rng0,signn,rnn0

        open(1,file='chargb.dat')   ! for input parameters
        open(2,file=' ')    ! for output data

        read(1,*) sigpmt,wfast
                          !FWHM of PMT transit time spread ( in ns )
                           !constant fraction discrimination factor
        read(1,*) indexg    !number of scintillation components for photons
        if (indexg .eq. 1) then
                read(1,*) Tg1,fg1
        else if (indexg .eq. 2) then
                read(1,*) Tg1,fg1,tg2,fg2
        else if (indexg .eq. 3) then
                read(1,*) Tg1,fg1,tg2,fg2,tg3,fg3
```

```
      else
            read(1,*) Tg1,fg1,tg2,fg2,tg3,fg3,tg4,fg4
      end if
      read(1,*) indexn    !number of scintillation components
                          ! for neutron
      if (indexn .eq. 1) then
      read(1,*) Tn1,fn1
      else if (indexn .eq. 2) then
            read(1,*) Tn1,fn1,tn2,fn2
      else if (indexn .eq. 3) then
            read(1,*) Tn1,fn1,tn2,fn2,tn3,fn3
      else
            read(1,*) Tn1,fn1,tn2,fn2,tn3,fn3,tn4,fn4
      end if
      sigpmt=sigpmt/2.35

10    write(0,*) "Enter average number of photon-electrons, -1 -1 stop"
      write(0,*) " ****** and width of window"
      read(0,*) n0,wfast
      if(n0 .le. 0) go to 100
      write(0,*) "j# Photon Neutron total# and window",n0,wfast
      write(2,*) "j# Photon Neutron total#",n0
      id=1
      call simpson(pgfast,0.0,wfast)  ! for photons
      id=0
      call simpson(pnfast,0.0,wfast)  ! for netrons
      pgfast=1-pgfast      ! slow component
      pnfast=1-pnfast      ! slow component
      ppgfast=1-pgfast
      ppnfast=1-pnfast
      write(0,*) "p_gfast, P_nfast: ",pgfast,pnfast
      sigman=sqrt(n0)
      nd=int(sigman)
      nd1=n0-3*nd             !take +-3*sigma width ( 0.997)
      nd2=n0+3*nd
c     write(0,*) nd1,nd2
      smaxg=0.0
      smaxn=0.0
c     pause 0
      if(n0 .gt. 150) go to 55   ! use approximation formula
c-----------------------------------

      do 50 j=1,nd2
      pgj=1.0
```

```
      pnj=1.0
      sumg=0.0
      sumn=0.0

            do 15 i=1,j
            if(pgj .le. 1e-299) go to 16
            pgj=pgj*pgfast
 15       continue
 16       do 17 i=1,j
            if(pnj .le. 1e-299) go to 18
            pnj=pnj*pnfast
 17       continue
 18   continue
c     write(0,*) "pgj,pnj", pgj,pnj
      if(j .gt. nd1) then
            jj=j
      else
            jj=nd1
      end if
c     n=n0
      do 30  n=jj,nd2
      rjj=float(n)
      id=3
      pn=fun(rjj)
c     write(0,*) pn
c     pause 1

      call factorl(n,j,a) !n must .le. 1000 to prevent upper overflow

c     write(0,*)n,j,a
c     pause 2
      a=pn*a
      pgj_1=1.0
      pnj_1=1.0
            do 20 k=1,n-j
            if(pgj_1 .le. 1e-299) go to 22
            pgj_1=pgj_1*ppgfast
 20   continue
 22   continue
            do 24 k=1,n-j
            if(pnj_1 .le. 1e-299) go to 26
            pnj_1=pnj_1*ppnfast
 24   continue
 26   continue
```

```
c       write(0,*) "1-pfast",pgj_1,pnj_1
c          pause 3

          sumg=sumg+a*pgj_1
          sumn=sumn+a*pnj_1
 30     continue

          sumg=sumg*pgj
          sumn=sumn*pnj
          write(0,*) " j,Pg,Pn",j,sumg,sumn
          write(2,*) j,sumg,sumn                    ! output results

          zumg(j)=sumg
          zumn(j)=sumn
          if (sumg .gt. smaxg) then
                  smaxg=sumg
                  itg=j
          else if ( sumn .gt. smaxn) then
                  smaxn=sumn
                  itn=j
          else

          end if

c       pause 4

 50     continue
          go to 99
c-----------------------------------------------
c for large total number n0 ( > 200 )
c-----------------------------------------------
 55     continue
          do 60 j=1,nd2
          sumg=0.0
          sumn=0.0
          if(j .gt. nd1) then
                  jj=j
          else
                  jj=nd1
          end if

          do 65  n=jj,nd2
          rjj=float(n)
          id=3
```

```
        pn=fun(rjj)

        rng0=n*pgfast
        rnn0=n*pnfast
        sigg=sqrt(rng0*ppgfast)
        signn=sqrt(rnn0*ppnfast)
        rjj=float(j)
        rr1=abs(rjj - rng0)
        rr2=abs(rjj - rnn0)
        rg3=5*sigg
        rn3=5*signn
c       write(0,*) "rr1,rr2,rg3,rn3",rr1,rr2,rg3,rn3
c       write(0,*) pn
        if(rr1 .ge. rg3 .and. rr2 .ge.rn3) then
                go to 65
        else if(rr1 .lt. rg3 .and. rr2 .ge. rn3) then
                id=4
                sumg=sumg+pn*fun(rjj)
                sumn= sumn
        else if(rr1 .ge. rg3 .and. rr2 .lt. rn3) then
                id=5
                sumn=sumn+pn*fun(rjj)
                sumg= sumg
        else
                id=4
                sumg=sumg+pn*fun(rjj)
                id=5
                sumn=sumn+pn*fun(rjj)
        end if
c       write(0,*)"j,n,sumg,   sumn",j,n,sumg,sumn
c       pause 0
65      continue

c       write(0,*) " j #,  Pg(j), Pn(j)",j,sumg,sumn
        write(2,*) j,sumg,sumn                  ! output results

        zumg(j)=sumg
        zumn(j)=sumn
        if (sumg .gt. smaxg) then
                smaxg=sumg
                itg=j
        else if ( sumn .gt. smaxn) then
                smaxn=sumn
                itn=j
```

```
        else

        end if

60      continue

99      continue
c-------------------------------------------------
c calculate FOM and FWHM.....
c
        write(0,*) "maximum at ",itg,itn
        h1=zumg(itg)/2.0
        h2=zumn(itn)/2.0
        rt1=itn-itg         !peak separation

        do 80 it=1,nd2-1
        if(zumg(it) .lt. h1 .and. zumg(it+1) .gt. h1) t11=it
        if(zumg(it) .gt. h1 .and. zumg(it+1) .lt. h1) t12=it
80      continue
        fwhmg=t12-t11
        do 81 it=1,nd2-1
        if(zumn(it) .lt. h2 .and. zumn(it+1) .gt. h2) t11=it
        if(zumn(it) .gt. h2 .and. zumn(it+1) .lt. h2) t12=it
81      continue
        fwhmn=t12-t11
        fom=rt1/(fwhmg+fwhmn)


        write(0,*) " windows and total number n0",wfast,n0
        write(0,*) "peak(g),peak(n),fwhm(g),fwhm(n),figure of merit"
        write(0,*) itg,itn,fwhmg,fwhmn,fom
        write(0,*) "Maximum at peaks",zumg(itg),zumn(itn)
        go to 10    ! back to beging



         go to 10        ! back to begin

100     close(1)
        close(2)
        stop
        end
c--------------------------------------------------------------------
c  Subroutines and functions
```

```
c-----------------------------------------------------------------------
      subroutine simpson(sinteg,a,b)   ! composite simpson integration
      h0=b-a

      if (h0 .gt. 50.0 ) then
            nn=200          ! nn must be even
      else if (h0 .gt. 100.0) then
            nn=1000
      else
            nn=28
      end if

      h=h0/nn
      f1=fun(a)+fun(b)
      f2=0.0
      f3=0.0
      do 10 i=2,nn-2,2
            f2=f2+fun(a+i*h)
10    continue
      do 20 i=1,nn-1,2
20    f3=f3+fun(a+i*h)
      sinteg=h*(f1+f2*2+f3*4)/3
      return
      end

c----------------- second simpson adaptive integration
c
    function fun(z)
    dimension v(8),tol(100),ai(100),hi(100),fai(100),fci(100)
     dimension si(100),l(100),fbi(100)
     common sigpmt,id

     sinteg=0.0
     i=1
     if(id .lt. 3) then
          sig=3*sigpmt
          a=z-sig
          if(a .lt.0.0) a=0.0
          b=z+sig
     else
          a=z
          b=z+1
     end if
```

```
      tol(i) = 10*0.01      ! esn=0.01 is tolerence
      ai(i) = a
      hi(i)=(b-a)/2

         if(id .eq. 1) then
           fai(i)=fun1(a-z,z)
           fci(i)=fun1(a+hi(i)-z,z)
           fbi(i)=fun1(b-z,z)
         else if (id .eq. 0) then
           fai(i)=fun2(a-z,z)
           fci(i)=fun2(a+hi(i)-z,z)
           fbi(i)=fun2(b-z,z)
         else if (id .eq. 3) then
           fai(i)=fun3(a)
           fci(i)=fun3(a+hi(i))
           fbi(i)=fun3(b)
         else if (id .eq. 4) then
           fai(i)=fun4(a)
           fci(i)=fun4(a+hi(i))
           fbi(i)=fun4(b)
         else
           fai(i)=fun5(a)
           fci(i)=fun5(a+hi(i))
           fbi(i)=fun5(b)
         end if

      si(i)=hi(i)*(fai(i)+4*fci(i)+fbi(i))/3
      l(i)=1
      ll=1
10     if(i.le.0) go to 30

         if(id .eq. 1) then
           fd=fun1(ai(i)+hi(i)/2-z,z)
           fe=fun1(ai(i)+1.5*hi(i)-z,z)
         else if (id .eq. 0) then
           fd=fun2(ai(i)+hi(i)/2-z,z)
           fe=fun2(ai(i)+1.5*hi(i)-z,z)
         else if(id .eq. 3) then
           fd=fun3(ai(i)+hi(i)/2)
           fe=fun3(ai(i)+1.5*hi(i))
         else if(id .eq. 4) then
           fd=fun4(ai(i)+hi(i)/2)
           fe=fun4(ai(i)+1.5*hi(i))
         else
```

```
        fd=fun5(ai(i)+hi(i)/2)
        fe=fun5(ai(i)+1.5*hi(i))
      end if

      s1=hi(i)*(fai(i)+4*fd+fci(i))/6
      s2=hi(i)*(fci(i)+4*fe+fbi(i))/6
      v(1)=ai(i)
      v(2)=fai(i)
      v(3)=fci(i)
      v(4)=fbi(i)
      v(5)=hi(i)
      v(6)=tol(i)
      v(7)=si(i)
      v(8)=l(i)
      i=i-1
      error=abs(s1+s2-v(7))
      if(error.lt.v(6)) go to 20
      if(v(8).ge.50) go to 1010    !error message
      i=i+1                ! add one level
      ai(i)=v(1)+v(5)
      fai(i)=v(3)
      fci(i)=fe
      fbi(i)=v(4)
      hi(i)=v(5)/2
      tol(i)=v(6)/2
      si(i)=s2
      l(i)=v(8)+1
      i=i+1          !data for left half subinterval
      ai(i)=v(1)
      fai(i)=v(2)
      fci(i)=fd
      fbi(i)=v(3)
      hi(i)=hi(i-1)
      tol(i)=tol(i-1)
      si(i)=s1
      l(i)=l(i-1)
      ll=ll+1
      go to 10
20    sinteg=sinteg+s1+s2
      go to 10
30    continue
      n=4*ll      !n is number of subintervals used
c     write(0,*) 'Integral sum is',sinteg
c     write(0,*) 'Number of subintervals used is ', n
```

```
      fun=sinteg
      go to 1015
1010  write(0,*) 'Maximum spliting level(100) exceeded'
      pause 0
1015  continue
      return
      end

      function fun1(z1,z0)          !for photons
      common sigpmt,id,tg1,fg1,tg2,fg2,tn1,fn1,tn2,fn2,n0
      common tg3,fg3,tg4,fg4,tn3,fn3,tn4,fn4
      x=exp(-0.5*z1**2/sigpmt**2)
      fun1=x/sqrt(2*3.1416)/sigpmt
      fast=fg1*exp(-z0/tg1)
      if(fg2.ne.0.0) fast=fast + fg2*exp(-z0/tg2)
      if(fg3.ne.0.0) slow=fg3*exp(-z0/tg3)
      if(fg4.ne.0.0) slow=slow +fg4*exp(-z0/tg4)
      fun1=fun1*(fast+slow)
      end


      function fun2(z2,z0)              !for neutrons
      common sigpmt,id,tg1,fg1,tg2,fg2,tn1,fn1,tn2,fn2,n0
      common tg3,fg3,tg4,fg4,tn3,fn3,tn4,fn4
      x=exp(-0.5*z2**2/sigpmt**2)
      fun2=x/sqrt(2*3.1416)/sigpmt
      fast=fn1*exp(-z0/tn1)
      if(fn2.ne.0.0) fast=fast + fn2*exp(-z0/tn2)
      if(fn3.ne.0.0) slow=fn3*exp(-z0/tn3)
      if(fn4.ne.0.0) slow=slow +fn4*exp(-z0/tn4)
      fun2=fun2*(fast+slow)
      end

      function fun3(z)              !for neutrons
      common sigpmt,id,tg1,fg1,tg2,fg2,tn1,fn1,tn2,fn2,n0
      x=exp(-(z-n0)**2/(2*n0))
      fun3=x/sqrt(2*3.1416*n0)
      end

      function fun4(z)              !for neutrons
      common sigpmt,id,tg1,fg1,tg2,fg2,tn1,fn1,tn2,fn2,n0
      common tg3,fg3,tg4,fg4,tn3,fn3,tn4,fn4
      common sigg,rng0,signn,rnn0
      x=exp(-(z-rng0)**2/(2*sigg**2))
```

```
      fun4=x/sqrt(2*3.1416)/sigg
      end

      function fun5(z)              !for neutrons
      common sigpmt,id,tg1,fg1,tg2,fg2,tn1,fn1,tn2,fn2,n0
      common tg3,fg3,tg4,fg4,tn3,fn3,tn4,fn4
      common sigg,rng0,signn,rnn0
      x=exp(-(z-rnn0)**2/(2*signn**2))
      fun5=x/sqrt(2*3.1416)/signn
      end

      subroutine factorl(n0,j2,a)
      real*8 a
      a=float(n0)
      j1=j2
      j3=j2
      n1=n0
5     a=a*float(n1-1)
      b=float(n0-j1)
      if(b .le. 1.0) go to  6
      a=a/b
6     if(j3.le. 1) go to  10
      a=a/float(j3)
10    j3=j3-1
      j1=j1+1
      n1=n1-1
      if(n1.le.1) go to 20   !stop
      go to 5
20    return
      end
```

chargn.dat (input data file for a old NE213 scintillator)

0.1 200 0.9 ← Transit time spread of PMT(ns), time range and CFD value
2 ← number of scintillation components for photons
3.36 0.165524 24.6 0.018042 ← tg(1) fg(1) tg(2) fg(2) for photons
2 ← number of scintillation components for neutrons
3.94 0.133824 47.1 0.010037 ← tn(1) fn(1) tn(2) fn(2) for neutrons

$$pg(t) = \sum_{i=1}^{ig0} fg(i) * \exp(-\frac{t}{tg(i)}),$$

ig0: number of scintillation components for photons

$$pn(t) = \sum_{i=1}^{in0} fn(i) * \exp(-\frac{t}{tn(i)})$$

in0: number of scintillation components for neutrons

chargs.dat (input data file for a trans-stilbene crystal)

2.4 200 0.9
2
4.5 0.1678 14 0.0175 Tg1 fg1 tg2 fg2
2
5.5 0.11382 46 0.00813 Tn1 fn1 tn2 fn2 .....

chargb.dat (input data file for the Borexino Scintillator)

2.4 70 0.9
3 ← three scintillation components for photons
1.75 0.489 8.12 0.0161 71.25 0.00018
4 ← four scintillation components for photons
2.03 0.308 13.10 0.0124 56.19 0.0019 399.6 0.0026

VITA

Zhong Cao was born in Changshu, Jiangsu, China. He graduated in Nuclear Physics from Peking University in June 1965, after a six year program leading to a degree equivalent to a M.S. degree in the United States. Between August 1965 and June 1981, he worked as a research assistant and a research associate in the Nuclear Physics division, at the China Institute of Atomic Energy. He went to the Department of Physics, University of Kentucky in June 1981 and worked as a research associate in nuclear physics until November 1983. He was an Associate Senior Research Fellow in the China Institute of Atomic Energy after November 1983 and became the leader of a Free Electron Laser research group in January 1989. In December 1992, he came to the United States to be reunited with his family. He entered the Department of Nuclear Engineering of the University of Tennessee, Knoxville, in January 1994 to study and expects to be awarded the Doctor of Philosophy degree with a major in Nuclear Engineering in August of 1997.