Contents

Deducing the π^0's Structure via Data Analysis

A. Stetz, M. Ashfield (Summer, 2001, REU), (RHL Edits)

July 30, 2002

1 Introduction

1.1 Why this Experiment was Done

It is well known to those who study subatomic physics that interact via the strong force, such as protons and mesons, have sizes on the order of 1 fermi (10^{-13}cm). In contrast, elementary particles that interact via the electromagnetic and weak forces, such as electrons and neutrinos have, appear to have no size at at! They are what we call "point" particles.

The notion of "size" in physics deserves some comment. Although 1 fermi is approximately a million times smaller than a typical atom, and a billion times smaller than the wavelength of visible light, physicists feel comfortable saying that they have "measured the size" of a such an incredibly tiny object. Just as we think of seeing object smaller than our eyes can see by using a microscope, subatomic physicists think of seeing elementary particles by using beams of high energy particles as their "microscopes".

The basic physics is that in order to look at an object of size ℓ, we need to use light with a wavelength $\lambda \leq \ell$. To look at an elementary particle we use quantum-mechanical deBroglie relation between the momentum p of an elementary particle and its wavelength:

$$\lambda = \frac{h}{p}, \tag{1}$$

where h is Planck's constant. So we can, in some sense, see the shadow of neutrons and protons by observing by scattering beams of high energy particles from them.

Unfortunately, the quantum and relativistic nature of submicroscopic particles means that there might not be anything as simple as a single "size" to characterize them. Rather than being little spheres, strongly-interacting particles are more like clouds of particle and fields interacting with each other. Our goal is to measure something about the structure of that cloud.

Determining the structure of an elementary particle is not easy. If we chose to see them by "shining" electron on them, then the electrons must have some 10 GeV of energy to be able to see any detail in a 1 fm particle. This requires a world-class high-energy accelerator.

There is another problem in trying to study the structure of elementary particles. While the protons found inside atomic nuclei are stable, all other strongly-interacting particles decay too quickly to be used as the target in a scattering experiment. For example, charged pions have a half life of 24×10^{-9} seconds. While, 10^{-9} seconds may correspond to thousands of billions of years on a nuclear time scale (typically 10^{-23} seconds), it is short for we mortals and so we must be rather ingenious and indirect to see the structure within an elementary particle.

There is a clever approach to measuring elementary particle sizes that works in a few special cases. If a particle decays into another two particles, then the constraints of energy and momentum conservation are too great to let us learn anything about the dynamics. However, if a particle decays into three or more particles, then even though the sum of the energies and the sum of the momenta of the final particles are constrained, the distribution of energy and momentum among the particles can be used to tell us something about the structure of the decaying particle. While this is not the same as directly measuring the size of a decaying particle, in some abstract sense it is a measurement.

An example of a particle decaying into three final particles is afforded by the neutral pi meson or pion:

$$\pi^0 \to \gamma + e^+ + e^-, \tag{2}$$

which is usually written as just

$$\pi^0 \to \gamma e^+ e^-. \tag{3}$$

The π^0 is a close relative of the charged pions, π^+ and π^-, and all three should have approximately the same size and structure. However, while we have already indicated that charged pions decay slowly with a half life of 24×10^{-9} seconds, neutral pions decays quickly with a half life of 10^{-16} seconds. Usually the π^0 decays into two photons:

$$\pi^0 \to \gamma\gamma. \tag{4}$$

But in approximately 1% of the decays, one of the gammas is replaced by the electron-positron pair shown in (2), and we obtain what is called “Dalitz decay” (in honor of the theorist who first investigated it). One of the things that Dalitz showed was that since the decay occurs as a two step process:

$$\pi^0 \rightarrow \gamma \quad \gamma \tag{5}$$
$$\hookrightarrow \quad e^+e^-, \tag{6}$$

some subtle aspects of the decay are sensitive to the $< 1fm$ structure of the π^0.

1.2 Preview of this Lab

This lab is based on an elementary particle experiment conducted at the TRIUMF cyclotron in 1986 that studied the the Dalitz decay of the neutral pion[PL]. While this may at first sight seem like a simple experiment, it turned out to be very difficult to analyze the data with enough precision to deduce something about the structure of the π^0. Two graduate students, Peter Gumplinger and Farzin Farzanpay, worked on the analysis for several years before the final paper (presented in an Appendix) was published. Your lab is based on the data analyses from those two theses. However, you will not have to work as hard or long as the Ph.D. students since we have eliminated some of the details through which they struggled.

This lab is a typical elementary particle experiments in which the data consist of “events.” An event happens when a beam particle interacts with a target particle, and the particles produced by this interaction are detected by the experimental apparatus. The event data consist of records of all the numbers measured by the apparatus pertaining to that event.

Data analysis occurs in steps. First the raw data on the tape records are converted to more relevant physical variables. For example, by using our separate measurements of the magnetic fields, a set of particle coordinates is converted into the momentum of the particle. Next, these intermediate data are plotted, examined, and then plotted in a different way until we are satisfied that this is the best display of the physics. We then go back and eliminates or “cut” events that do not contain relevant information for the process of interest; this increases the strength of the signal relative to the noise. The data are then examined again until all we convinced that no more corrections can significantly affect the measured parameters.

The analysis of this experiment is challenging, in part, because there may be just a few interesting events and many uninteresting ones, and so we must be confident that these few events are real. Accordingly, it is important to understand just how the experimental equipment responds to all feasible conditions. This understanding is usually obtained by running a program that uses Monte Carlo simulations that create virtual particles and then trace out the paths of the particles through each piece of apparatus.

Of course a simulation of an experiment is based on an idealization of an actual experiment, and is no replacement for a real experiment. However, simulations let the experimentalists know how their equipment responds and what type of signal to expect. Finally, the validity of the measurements and the understanding of its significance is established by comparing actual data with simulated ones.

1.2.1 The Four Steps

This lab leads you to go through all some four steps needed to analyze realistic data:

Step 1. Physics: You explore the theory and visualize its kinematic consequences using the Maple worksheet `Kinematics_f.mws`. By plotting decay rates assuming various phase space boundaries, you gain the understanding of the kinematic variables needed to develop an experimental design.

Step 2. Calibration: You determine the calibration parameters for the pair of sodium iodide (NaI) detectors (TINA and MINA in Fig(1) by making use of a χ^2 fitting program to fit a theoretical distribution to data. The Fortran programs are in the direcory *Iteration.*

Step 3. Monte Carlo Simulation: You simulate results to calibrate your instruments and to compare to actual experimental measurements. The Fortran programs are in the direcory *Generator.*

Step 4. Analysis: You compare the simulated Monte Carlo data to the actual data to obtain your final results. The Fortran programs are in the direcory *Sorter.*

The original analysis of the data required many days of computer time during the 1980's, when computers were slow and expensive by modern standards. The current analysis takes only a few seconds on a 1.3 GHz machine and requires several large ($\sim$ 100 MB) files.

2 The Physics

The purpose of this experiment is to extract the form-factor slope parameter a from the decay of the π^0 meson. In a simple view, a is a measure of the "size" of the π^0, a quantities of fundamental interest to particle physics.

2.1 Theory

The world of strongly-interacting particles (hadrons) is a fast-paced one in which events take place on a time scale of 10^{-23} seconds. So while we may think of the π^0 as unstable, its lifetime is some 10 millions hadronic cycles, which is a large enough number to be considered hadronically stable.

The π^0 meson most often decays into two photons,

$$\pi^0 \rightarrow \gamma\gamma, \tag{7}$$

with a lifetime of approximately 10^{-16} sec. While much past effort has gone into measuring the properties of this decay, there is very little, aside from the value of the lifetime, to learn from it.

On rare occasion (approximately 1% of the time), the π^0 meson may decays into three particles:

$$\pi^0 \rightarrow e^- e^+ \gamma. \tag{8}$$

The Dalitz decay (8 can be thought of as a version of the 2γ decay (7) in which one of the photons converts internally to an electron-positron pair,

$$\begin{aligned} \pi^0 \rightarrow \gamma \quad & \gamma & (9) \\ & \hookrightarrow \quad e^+ e^-. & (10) \end{aligned}$$

Because there are three particles in the final state in (8), the momentum of each particle is not uniquely fixed by energy and momentum conservation. This extra bit of freedom opens a small window into the world of hadronic interactions through which we shall peak to learn more about the structure of the π^0. Yet as we shall see, it is hard to find the window and it is only opened a crack.

The π^0's that this experiment observes are created by taking a beam of π^- mesons and slowing them down in a liquid hydrogen (protons) until they come to rest. Once the π^- is at rest, the Coulomb force attracts it to a positively charged proton which then exchanges charge with it:

$$\pi^- p \rightarrow \pi^0 n. \tag{11}$$

The resulting π^0's have a velocity $\beta = v/c \approx 0.2$, and are emitted uniformly into the full 4π steradians solid angle.

Shortly after the π^0's are produced they decay into photons and electrons via (7) and (8). The emitted photons are difficult to detect, so one usually measures the magnitude and direction of the momenta of the positrons and electrons (both referred to as "electrons"). This is the decay of interest and is called an "event". An event contains six pieces of information: the three components of the electron's momentum and the three components of the positron's momentum.

2.2 Experimental Layout

The electron energies were measured with sodium iodide detectors consisting of large transparent crystals. Electrons passing through them produce electromagnetic showers consisting of electrons, positrons and photons. As the charged particles interact with the NaI they produce scintillation light with a total intensity approximately proportional to the total energy of the shower. This light is converted to a current pulse by means of an array of photomultiplier tubes, and the pulses are digitized by analog-to-digital converters, (ADC's for short). The output of these ADC's is stored as part of the event.

2.3 Kinematics

It is convenient to describe the decay (8) in terms of two dimensionless variables that have magnitudes between 0 and 1. They are called x and y and are defined as:

$$x = \frac{(p_+ + p_-)^2}{m_0^2}, \tag{12}$$

$$y = \frac{E_+ - E_-}{p_T}. \tag{13}$$

Here $p_\pm$ are the electron's and positron's four-momenta,

$$p_+ = (E_{e^+}, \mathbf{p}_{e^+}), \tag{14}$$

$$p_- = (E_{e^-}, \mathbf{p}_{e^-}), \tag{15}$$

m_0 is the π^0 mass, and

$$p_T = |\mathbf{p}_+ + \mathbf{p}_-|, \tag{16}$$

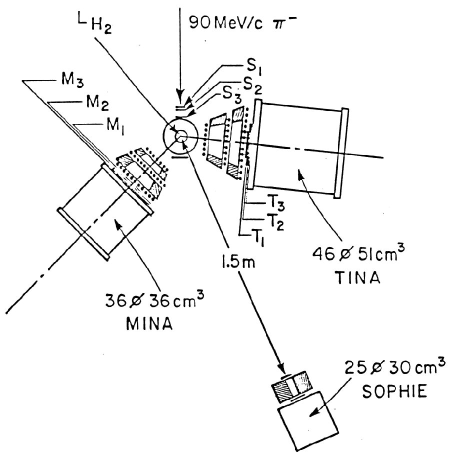

Figure 1: The experimental layout showing incident π^- beam, the NaI scintilators TINA and MINA, and the π^0 monitor SOPHIE. (Figure from Gumplinger's thesis.)

is the total momentum of the electron-positron pair. Note that we always use relativistic kinematics, which means that the energy is given by the Einstein energy-momentum relation

$$E = \sqrt{p^2c^2 + m^2c^4} \equiv \sqrt{p^2 + m^2}, \tag{17}$$

and that the squared modulus of the momentum four-vector is the mass-squared of the particle:

$$p^2 = (E, \mathbf{p}) \cdot (E, \mathbf{p}) = E^2 - \mathbf{p} \cdot \mathbf{p} = m^2. \tag{18}$$

We imagine the Dalitz decay of the π^0 to occur as shown in the Feynman diagram in Fig. 2. Here the wiggly lines represent photons, the straight lines are electrons, and the circle represent the initial π^0 at rest. Notice in Fig. 2 that one of the photons decays into an electron-positron pair. Since energy and momentum conservation do not permit photons with their zero mass to decay like that, the intermediate photon is not a real and is instead called a "virtual" photon. While virtual photons cannot be observed in the real world, the theory of Feynman diagrams makes it clear that it is permissible to have them occur as intermediate states.

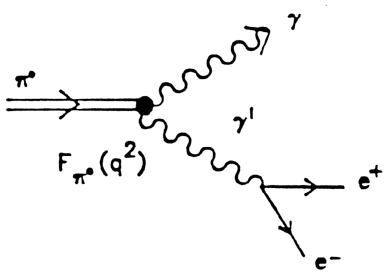

Figure 2: Dalitz decay of π^0 into a photon γ and an electron-positron pair. The electron-positron pair is seen to arise from an intermediate (virtual) photon γ'. The black dot represents the finite size of the π^0 and is labeled by the form factor $F_{\pi^0}(q^2)$. (Figure taken from Gumplinger's thesis.)

What makes virtual photons unreal is that they have a nonzero value for their mass. In fact, if we look at the virtual photon in the Dalitz decay diagram Fig. 2, we see that since it decays into the electron-positron pair, momentum conservation requires that the photon's four momentum be equal to the sum of the four momentum of the electron and of the positron:

$$p_\gamma = p_{e^+} + p_{e^-}. \tag{19}$$

So we can say that $p_\gamma^2 = (p_{e^+} + p_{e^-})^2$, is a nonzero quantity and is the squared mass of the virtual photon. Accordingly, the variable x defined in (12) equals the squared ratio of the mass of the virtual photon to the mass of the neutral pion. It is a Lorentz scalar, which means that it has the same value in the lab system and the rest frame of the π^0.

The variable y in (13) is seen to be the ratio of the difference in the electron's and positron's energy to their total momentum. It is a measure of how the electrons share their energy, with $y = 0$ corresponding to equal sharing, with $y \simeq 1$ corresponding to one electron taking away almost all the energy.

If you work out the detailed constraints of momentum and energy conservation, you will find that the allowed values of x and y are given by

$$r \le x \le 1, \qquad -\eta \le y \le \eta, \tag{20}$$

$$\text{where} \qquad r = 4m_e^2/m_0^2, \qquad \eta = \sqrt{1 - r/x}. \tag{21}$$

We will work with x and y since they show the physics the best, and since all other variables can be written in terms of them.

A final result of the experimental analysis is to obtain a single value for the decay rate Γ, that is, the number of decays per unit time. We can be more specific and consider $d\Gamma/(dxdy)$, that is, the number of π^0 decays per unit time for unit changes in x and y. This differential decay rate will be extracted from our data and compared to the theoretical prediction for it.

The theoretical prediction of the differential decay rate is called the **Kroll-Wada distribution**, and has the form

$$\frac{d\Gamma}{dx\,dy} = \frac{\alpha}{4\pi}\frac{(1-x)^3}{x}\left[1+y^2+\frac{r}{x}\right]F_{\pi^0}(q^2). \tag{22}$$

Here α is the fine structure constant,

$$\alpha = \frac{e^2}{\hbar c} \simeq \frac{1}{137}, \tag{23}$$

and $F_{\pi^0}(q^2)$ is a *form factor* or *structure factor* whose deviation from 1 is a measure of the neutral pion's finite size. We parameterize the form factor as

$$F = 1 + 2a, \tag{24}$$

where a is called the *slope parameter* and would equal 0 if the pion were a point particle. It is the single number we wish most to determine. In some sense and in some units, a is proportional to the "size" of the π^0. However, the effect of this correction is small and thus hard to measure. Calculations done before the experiment suggested a value $a \approx 0.03$. The present experiment was designed to have a sensitivity $\Delta a \approx \pm 0.01$.

The Kroll-Wada distribution is normalized so that if it is integrated over the allowed range of x and y, it gives a value that is just the ratio of the decay rates (the *branching ratio*),

$$\int dx\,dy \frac{d\Gamma}{dxdy} = \frac{\Gamma(\pi^0 \to e^+e^-\gamma)}{\Gamma(\pi^0 \to \gamma\gamma)} = 0.011854. \tag{25}$$

This value is in good agreement with experiment.

3 Your Experiment

The experiment was published in Physics Letters B 278 (1992) 413-418, as well as in Peter Gumplinger's 1986 thesis. Read the Letter now, before going any further. It should make good sense now (but better when you have completed this lab).

Doing a real experiment entails carrying through many steps, with each step requiring fussy attention to details. For example, electronics drift, detectors get out of alignment, particle beams become unstable, and, worst of all, events occur in our detectors of types we did not plan for. We call these unplanned events “background”, and we try to keep their number as small as possible.

Since analyzing the complete TRIUMF experiment would be too complicated and time consuming for a teaching lab such as this, we are presenting you with simulated data. These simulated data contain no background events and have been measured by perfect detectors (the data were, in fact, generated by the original Monte Carlo program).

Simulated or “theoretical” experiments such as this are increasingly popular throughout science. They are usually cheaper, faster, and more flexible than real experiments, and, most critrically, they appear to represent nature well. For example, you can try out modifications of your equipment to see which provides the best signal to noise ratio. While in some sense this takes some of the “experimenting” out of the experiment, believe us, if it does not work on the computer, it will not work in the lab!

In the Directory/Folder *Sorter* you are given a file `BigData.dat` with 99 Meg’s of raw “data” for you to ultimately analyze by comparing to a data set that you have generated by running a Monte Carlo simulation. As indicated in the Introduction, you will proceed in four steps:

Step 1: Physics and Kinematics: There is a Maple worksheet `Kinematics_fort.mws` that leads you through the steps. Alternatively, if you do not have Maple available, you can follow the same steps by doing the calculations in the supplied Fortran code [generated directly from the Maple commands].

Step 2: Simulate Data: Use the Fortran program in the director `Generator` to generate your own data set.

Step 3: Calibration: Independently of all this, determine the resolution function for the NaI detectors TINA and MINA using a χ-square minimization program in the `Iteration` directory.

Step 4: Analyze BigData Determine the size a by analyzing the data in the file `BigData.dat`. The program and data are in the directory `Sorter`.

3.1 Step 1: Kinematics and Decay Rates (Kinematics_f.mws)

You here follow the first step in designing an experiment, which is to investigate the constraints imposed by decay rates and kinematics what you can measure. Although not available when the experiment was being planned, we here take advantage of Maple's ability to do computations and visualize the results easily.

We suggest that you use Maple to work through the worksheet `Kinematics_f.mws`. We give this worksheet and some of its output below, but reading it is not as effective as working through the real thing. When you are done, you should have a better understanding of just what the experiment measures and how it relates to theory.

3.1.1 Kinematics_f.mws Worksheet Listing

3.2 Step 2: Detector Calibration (LineShape.mws)

The piece of equipment that detects the electrons resulting from the $\pi^0 \rightarrow e^+e^-\gamma$ decay is called the *detector.* In this section we lead you through some of the necessary background work needed to do an experiment, namely, the calibration of the detector. The piece of equipment that detects the electrons resulting from the $\pi^0 \rightarrow e^+e^-\gamma$ decay is called the "detector". This is somewhat of an aside from our main thrust of determining a value for the pion size a, but is, in fact, one of the things we had to do in the experiment. Yet since we are presenting you with somewhat idealized data that assume perfect detector efficiency, you will not make actual use of these calibrations.

As shown in Fig. 1, the electron energies were measured with sodium iodide detectors consisting of large transparent crystals. Electrons passing through them produce electromagnetic showers consisting of electrons, positrons and photons. As the charged particles interact with the NaI they produce scintillation light with a total intensity approximately proportional to the total energy of the shower. This light is converted to a current pulse by means of an array of photomultiplier tubes, and the pulses are digitized by analog-to-digital converters, (ADC's for short). The output of these ADC's is stored as part of the event.

This process is complicated and not fully predictable. This is partly because shower formation is stochastic and partly because the responses of the individual phototubes change with time. Accordingly, a monoenergetic beam of electrons produces a signal that is distributed over a range of energies, and this distribution changes slowly over time as the detector ages. Since the detector's output is a peak with finite width, the resolution of the detector, that is, its ability to distinguish electrons of different energy, is limited. A further corrceion is needed, in addition, because the maximum of this peak is not necessarily at the incident electron's energy.

In order to analyze the experiment, it was necessary to know the response of these detectors to a high degree of accuracy. So one of the major tasks was to monitor this performance and quantify it so that it would be useful for analysis. We quantify the process by using an empirical function $R(E)$ that gives the output response of the detector at energy E when an electron of energy E_0 enters the detector:

$$R(E) = A\ \exp\left[\frac{E - E_0}{a}\right]\left[1 - \operatorname{erf}\left(\frac{E - E_0}{b}\right)\right] \tag{26}$$

Here erf is the error function, A is a normalization constant, and a and b are parameters obtained by fitting this function to real data. Other researchers who have investigated this function suggest that a better description of the detector is obtained by including an energy dependence in the parameters of the form

$$a = a_0 E^c. \tag{27}$$

In order for you to get a feel for the detector's reponse and calibration, you should now examine the Maple worksheet `LineShape.mws`.

3.2.1 LineShape.mws Worksheet Listing

3.2.2 Analysis of Line Shape

We have just seen that real detectors do not output one number for an electron's energy but rather yield a spread of energies. They thereby give a 1D line a width, and shift the position of this line by an amount that depends on the energy.

There is an addition complication that arises from the analog-to-digital converters (ADC's) used to convert the output of our instruments to the digital form we record on magnetic tapes. The ADC's do not measure energy directly, but instead output an integer N that in theory is proportional to the light output of the crystal:

$$E = \lambda N \tag{28}$$

So before we can decode the magnetic tape, we must determine the constant of proportionality λ for each detector and then verify that they are independent of energy.

Fortunately, nature provides us a number of reactions that we can use to calibrate our detector. The dominant decay of the neutral pion

$$\pi^0 \to \gamma\gamma, \tag{29}$$

produces photons with energy $m_\pi/2$ in the π^0 rest frame. In the lab we see these photons with a uniform distribution from 55 to 86 MeV. In addition, the reaction

$$\pi^- p \to n\gamma \tag{30}$$

produces an 131 MeV gamma ray. Fortunately, NaI detectors responds to photons much like they do to electrons, and so we have two benchmarks with which to calibrate our detectors.

3.2.3 Fitting Lines (Iteration.f)

The program `iteration.f` fits lines to data, that is, it searches for values of the parameters in the resolution function that provide a best fit to the data. There are also two data files `t330.dat` and `m330.dat` containing a list of channel numbers and the corresponding number of events (`t` and `m` stand for the NaI detectors TINA and MINA shown in Fig. 1.)

The first step you should now take is to examine `t330.dat` and `m330.dat` by plotting them both on the same graph.

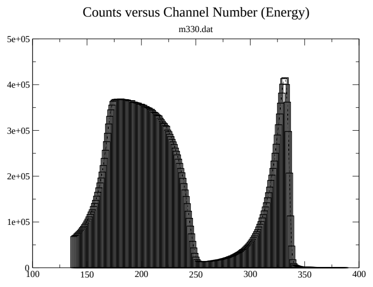

Figure 3: Plot of data in file `m330.dat` in Iteration directory/folder.

You should obtain a spectrum of counts versus channel number (gamma ray energy) that looks like Fig. 3. The broad peak from 150 to 250 is from the $\pi^0 \rightarrow 2\gamma$ decay, while the narrower peak at 330 is due to the $\pi^- p \rightarrow n\gamma$ reaction.

Note that if the experimental equipment had infinite resolution, then the narrow peak would be a single line. Accordingly, the width and shape of this line gives provides us with a direct measure of our experimental resoltion. Note also these cases the curves look smooth because they represent such a vast number of events that the statistical variation is small in comparison to the signal. The statistical variations will show up after we impose cuts on the kinematics.

The next step for you is to fit the Kroll-Wada distribution (22) to the data in the files `t330.dat` and `m330.dat`.

When you run the executeable in `Iteration`, you will be asked a number of questions for you to answer at your terminal. The program then fits equation (22) to the data that lies between the minimum and maximum channel numbers that you have enterred. You are given the option of fitting one parameter at a time until they are all close enough to a minumum for the automated fitting routine to take over and refine the fit. The final values for the parameters are saved in file `CHIPLOT.dat`. Simulation output data corresponding to these parameters is stored in`DataFit.dat`.

3.3 Step 3: Monte Carlo Simulation (Generator.f)

Finally, the pion size parameter a will now be determined by comparing a complete Monte Carlo simulation of the experiment with the experimental data. In other words, we will compare portions of the data set with output of the Monte Carlo simulation until we find a region of data that is sensitive to a and which agrees with the simulation for some specific value of a. That specific value is the number we deduce from the data.

The first step in the simulation is to generate a table (x_i, y_i) of values for the kinematic variables x and y defined by (12) and (13). The x and y values are to be chosen randomly and independently with a probability given by K-W distribution (22). There are two basic ways to generate random numbers weighted by a specified distribution function $f(x)$, that is, with the probability of a number falling the the range $x \to x + dx$ given by

$$\mathcal{P}(x) = f(x)\,dx. \tag{31}$$

In both cases, there will be more random x's near where $f(x)$ is large, but the x's will still be uncorrelated with each other.

The **rejection method** starts by generating two sets of uniformly distributed random numbers, x and z, with the separate ranges

$$x_{min} \leq x \leq x_{max}, \quad 0 \leq z \leq f_{max}. \tag{32}$$

Here f_{max} is the maximum value of $f(x)$ in the interval $x_{min} \leq x \leq x_{max}$. We next convert the uniform x distribution into one weighted by $f(x)$ by rejecting some x's. If $f(x) > z$, you reject x and pick another pair of random numbers. If $f(x) < z$, you accept x and add its value to a table of(x_i values. The x's accumulated in this way will be random with weight $f(x)$.

This technique is straightforward, guaranteed to work, but expensive. Its efficiency, that is, the ratio of trials to successes, is given by

$$\frac{\int_{x_{min}}^{x_{max}} f(x)dx}{f_{max}\,(x_{min} - x_{max})}. \tag{33}$$

Yet if $f(x)$ is the Kroll-Wada distribution, we have such a sharply peaked $f(x)$ that the efficiency is only about 10^{-3}. This means that we will reject 1000 events for each 1 that we accept.

The **direct method** is 100% efficient, though it's also less elegant and requires more programming. We start by examining the expression for the cumulative probability function $g(x)$ for the K-W distribution:

$$g(x) = \int_{x_{min}}^{x} f(x')\,dx' = \int_{x_{min}}^{x} \frac{d\Gamma}{dx\,dy}\,dx' \tag{34}$$

$$= \int_{x_{min}}^{x} \frac{\alpha}{4\pi}\frac{(1-x)^3}{x}\left[1 + y^2 + \frac{r}{x}\right](1+2a)\,dx. \tag{35}$$

We next imagine that $g(x)$ were a uniform, random distribution between 0 and 1, and set it to a new variable z:

$$z = g(x) = \quad \text{uniform}. \tag{36}$$

The key observation now is that if we could somehow solve this equation for x

$$x = g^{-1}(z), \tag{37}$$

then the x's will be a random distribution with weight $f(x)$ if the z's are uniform. In other words, we chose a uniform random distribution for z, and then insert those z's into (37) to get an appropriately weighted x distribution.

The problem with the direct method is that it is often difficult or impossible to find an analytic expression for the inverse. We get around that evaluating the integral numerically and then doing a table lookup to determine the inverse. Explicitly, we generate a table of 1000 uniformly spaced random numbers x_i, and then evaluate the integral (34) to obtain the corresponding values of $g(x_i)$. We then generate a uniform random g_j and interpolate to find the the corresponding g^{-1}.

The table of x_i's and g_i's is in the file `Xtable.dat`. The needed distribution in y is not sharply peaked and so is generated by the rejection method (32) in `DIVONDZ.f`.

3.4 Step 4: Analysis (Sorter.f, Generator.f)

In order to generate your own Monte Carlo events, now **run the program in `Generator`**. You will be asked three questions when you start the program:

1. The number of events you want to simulate.
2. The name of the file you want to store the events in.
3. Whether to use the structure-independent ($F = 1$) or structure-dependent distribution. (The reason for this will be explained shortly.)

The output from `Generator` includes the number of π^0's responsible for these events. Be sure to keep track of this number. The data given to you to analyze correspond to a 4044 million π^0's. Also be careful to give unique filenames for the data, lest you overwrite a previous file. You are assisted in this by the file `BOOKKEEPER` that keeps a record of the files you have created.

The "data" for you to analyze is in the 90 MB file `BigData.dat` in the folder/directory `Sorter`. When analyzed, its 5×10^5 events yields approximately 5000 good events.

The obvious way to analyze these data is to fit equation (22) to data with a as an adjustable parameter. This approach was used in earlier experiments but yielded poor results for two reasons. (1) the experimental distribution depends on the apparatus in a more complicated way than just equation (22). (2) The fit really involves a number of parameters, but these parameters are correlated in a complicated way, yet fitting a number of parameters to the data produces a value of a with poor statistical accuracy.

Our less obvious approach avoids the statistical problems with fitting. First we note that we can determine the actual number of π^0's produced in the target by counting (very carefully) the number of γ's from the common two photon decay,

$$\pi^0 \rightarrow \gamma\gamma. \tag{38}$$

The **Scheme to Measure** a was already explored in the Maple worksheet `Kinematics_f.mws`. We know that the experimental data, which are mainly at small x values, are described rather well by the KW distribution with the structure function $F = 1 + 2ax = 1$, that is, with $a = 0$. However, we saw that by looking at larger values of x, we can extract a statistically meaningful value for a.

We start by examining the theoretical expression for the measured decay rate Γ_M with the structure function written out explicitly:

$$\Gamma_M = \int_{x_{min}}^{x_{max}} (1 + 2ax) \frac{d\Gamma_{SI}(x)}{dx} dx, \tag{39}$$

where $d\Gamma_{SI}(x)/dx$ is the KW distribution function with no structure function include ("SI" for structure independent). We now write this integral as the two separate integrals

$$\begin{aligned} \int_{x_{min}}^{x_{max}} (1 + 2ax) \frac{d\Gamma_{SI}(x)}{dx} dx &= \int \frac{d\Gamma_{SI}(x)}{dx} dx + 2a \int x \frac{d\Gamma_{SI}(x)}{dx} dx && (40) \\ &= \Gamma_{SI} + \Gamma_{SD}. && (41) \end{aligned}$$

So we have written our measured rate Γ_M as a structure independent rate Γ_{SI} and a structure dependent rate Γ_{SD} where

$$\Gamma_{SI} = \int_{x_{min}}^{x_{max}} \frac{d\Gamma_{SI}(x)}{dx} dx \tag{42}$$

$$\Gamma_{SD} = \int_{x_{min}}^{x_{max}} x \frac{d\Gamma_{SI}(x)}{dx} dx. \tag{43}$$

We can solve this expression for a to obtain

$$a = \frac{\Gamma_M - \Gamma_{SI}}{2\Gamma_{SD}}. \tag{44}$$

The scheme is then to measure Γ_M and simulate Γ_{SI} and Γ_{SD} with the Monte Carlo program `Generator`.

1. Generate a large number of events using the structure-independent distribution
$$\frac{d\Gamma_{SI}}{dx\, dy} = \frac{\alpha}{4\pi}\frac{(1-x)^3}{x}\left[1 + y^2 + \frac{r}{x}\right]. \tag{45}$$

2. Analyze these events in exactly the same way and with the same cuts as the measured data. The program that does this will tell you how many π^0's were required.
$$\Gamma_{SI} \equiv \frac{\text{SI-type MC events remaining after all cuts}}{\text{The number of } \pi^0\text{'s that produced them}} \tag{46}$$

3. Next generate events using the structure-dependent distribution,
$$\frac{d\Gamma_{SD}}{dx\, dy} = \frac{\alpha}{4\pi}\frac{(1-x)^3}{x}\left[1 + y^2 + \frac{r}{x}\right] x \tag{47}$$

Use the program `Sorter` to analyze your events to obtain

$$\Gamma_{SD} \equiv \frac{\text{SD-type MC events remaining after all cuts}}{\text{The number of } \pi^0\text{'s that produced them}} \tag{48}$$

When you run this program you will have a chance to impose different cuts on the data and look at the results as histograms. There are two cuts that cannot be changed (1) all events are required to have a hit in the positron detector, and (2) all events are required to have $x > 0.1$.

You should play with your data, as do experimentalists, to see what it takes tomake the statistical error as small as possible. The strategies for doing this were discussed in the Maple worksheet, `Kinematics_f.mws`, and include:

1. Increasing the x_{min} cut.

2. Changing the position of electron detector by changing the opening angle parameter.

3. Setting a minimum energy cut.

The program automatically generates various files for you to plot and study:

`EMspec`	electron energy spectra
`EPspec`	positron energy spectra
`Tspec`	opening angle spectrum
`Xspec`	the x spectrum

Part of the responsibility of an experimentalist is to give an error estimate of any measured quantity. In this analysis with simulated data there are no systematic errors, but there are statistical ones. To obtain an estimate of the error in a we use (22) and (24) to solve for a:

$$a = \frac{d\Gamma/dx/dy}{\frac{2x\alpha}{4\pi}\frac{(1-x)^3}{x}\left[1+y^2+\frac{r}{x}\right]} - 1. \tag{49}$$

If we view this equation in terms of data fitting, then it has the form

$$a = a(\Gamma, x, y), \tag{50}$$

where x, y, and Γ are the quantities we have measured. The uncertainty Δa in a can then be approximated by the chain rule,

$$\Delta a \simeq \frac{\partial F}{\partial \Gamma}\Delta\Gamma + \frac{\partial F}{\partial x}\Delta x + \frac{\partial F}{\partial y}\Delta y \tag{51}$$

where Δx, for example, is the uncertainty in the measured x.

Use this equation to estimate, as best you can, the uncertainty in a.

A Appendix A: List of Subroutines

Each folder (directory) `Generator`, `Iteration`, and `Sorter` has a complete Fortran program and the data files for it to read. You can compile the source programs in any one directory independent of the other directories.

.1 Generator Directory

Generator	Generates Dalitz events in the lab frame
Setup	Sets up Bookkeeper file
Inikine	Initializes kinematic parameters (e.g. masses)
Dfun(x,y)	joint probability function for x and y varuiables
Thdz	Generates kinematics for Dalitz decay
Create	Generates particle with specified momenta
Divondz	Reads x table
Locate	locate point as part of rejection method
Polint	

.1 Iteration Directory

Iteration	Extracts NaI response parameters photon spectrum.
Chi2	Calculate χ^2 by iteration
Amoeba	
Amotry	
BLUR	Simulate energy resolution by smearing
Erf	Error function
Gammln	Ln of gamma function
Gammp	Gamma' function
Gcf	
Gser	
T4hist	Histogram construction
Xyread	

.1 Sorter Directory

Sorter	Analyze data files written by Sorter (Dalitz.f)

A Appendix B: Fortran Program Listings

.1 Generator

```
 !  Generator.f  (use f90 compiler)
!   Generates Dalitz events in the lab frame
!
      IMPLICIT REAL(M)
      CHARACTER(12) :: FILENAME
      DIMENSION PE(4), PP(4), PG(4)
      COMMON/FILENAME/FILENAME
      COMMON / KIN / PE, PP, PG, X, Y
      COMMON / RANDOM / ISEED
      COMMON / USERMASSES / MP, MN, MPO, MPM, ME
    CALL Setup(NEVENTS)
    OPEN(UNIT=9, FILE=FILENAME, STATUS='REPLACE')
    ISEED=1234567
      CALL Inikine
      DO I = 1, NEVENTS
      CALL Thdz
!  Recalculate x from the transformed data.
        PDOTP=PE(4)*PP(4)
        DO J=1,3
            PDOTP=PDOTP-PE(J)*PP(J)
        END DO
      REX=2.*(ME*ME+PDOTP)/MP0**2
      DIFF=X-REX
!  Finally, write everything to a file.
      WRITE(9,100) I,X,Y
100   FORMAT(I10,4X,2(E10.4,4X))
      WRITE(9,110) PP(1),PP(2),PP(3),PP(4)
      WRITE(9,110) PE(1),PE(2),PE(3),PE(4)
      WRITE(9,110) PG(1),PG(2),PG(3),PG(4)
110   FORMAT(4(E10.4,4X))
      END DO
      CLOSE(9)
      STOP
      END
C**********************************************************************
      SUBROUTINE Setup(NEVENTS)
      CHARACTER*25 FILENAME
      CHARACTER*100 COMMENT
      CHARACTER*8 DATE
      CHARACTER*10 TIME
      CHARACTER*5 ZONE
      INTEGER VALUES(8)
      REAL NO
      COMMON/FILENAME/FILENAME
      COMMON /SPECTRUM/ITYPE
      WRITE(*,*) 'TYPE 1 FOR SI, 2 FOR SD SPECTRA.'
      READ(*,*) ITYPE
      WRITE(*,*) 'GENERATE HOW MANY EVENTS? NEVENTS = '
      READ(*,*) NEVENTS
      WRITE(*,*) 'WHAT SHALL WE NAME YOUR FILE?'
      READ(*,*) FILENAME
      WRITE(*,*) 'BRIEF COMMENT?'
      READ(*,30) COMMENT
      WRITE(*,30) COMMENT
30    FORMAT(100A)
      XMIN=0.10
      BRSI=1.16882E-3
      BRSD=2.54067E-4
      SANGLE=2*5.176E-2
      WRITE(*,*) "XMIN = ", XMIN
      WRITE(*,*) "INTEGRATED SI BRANCHING RATIO = ", BRSI
      WRITE(*,*) "INTEGRATED SD BRANCHING RATIO = ", BRSD
      WRITE(*,*) "26.3 DEGREE OPENING ANGLE CORRECTION = ", SANGLE
      IF(ITYPE.EQ.1) THEN
      NO=REAL(NEVENTS)/(BRSI*SANGLE)
      GO TO 10
      ELSE IF(ITYPE.EQ.2) THEN
      NO=REAL(NEVENTS)/(BRSD*SANGLE)
      GO TO 10
      ELSE
      WRITE(*,*) 'NO CHOICE.  I QUIT!'
      STOP
      END IF
10    WRITE(*,*) "NUMBER OF ORIGINAL PI-ZEROS = "
      WRITE(*,20) NO
20    FORMAT(E10.4)
!   Set up the Bookkeeper file
      CALL DATE_AND_TIME(DATE,TIME,ZONE,VALUES)
      OPEN(UNIT=10, FILE='BOOKKEEPER',STATUS='OLD', POSITION='APPEND')
      WRITE(10,*) 'YEAR = ',VALUES(1)
      WRITE(10,*) 'MONTH = ',VALUES(2)
      WRITE(10,*) 'DAY = ',VALUES(3)
      WRITE(10,*) 'HOUR = ',VALUES(5)
      WRITE(10,*) 'MINUTES =',VALUES(6)
      WRITE(10,*) 'FILENAME = ',FILENAME
      WRITE(10,30) COMMENT
      WRITE(10,*) 'EVENTS PROCESSED = ',NEVENTS
      WRITE(10,*) 'DATA TYPE = ',ITYPE
      WRITE(10,*) 'XMIN = ', XMIN
      WRITE(10,*) 'INTEGRATED SI BRANCHING RATIO = ', BRSI
      WRITE(10,*) 'INTEGRATED SD BRANCHING RATIO = ', BRSD
      WRITE(10,*) '26.3 DEGREE OPENING ANGLE CORRECTION = ', SANGLE
      WRITE(10,*) 'NUMBER OF ORIGINAL PI-ZEROS = '
      WRITE(10,20) N0
      CLOSE(10)
      RETURN
      END
C*********************************************************************
C SUBROUTINE Inikine  initialize the parameters needed in the kinematics:
C particle masses, physical constants...
       SUBROUTINE INIKINE
        IMPLICIT REAL(M)
       COMMON / USERMASSES / MP, MN, MP0, MPM, ME
       COMMON / KINCONST / PI, RR, MAXEN, ETOTAL, B, GAMMA
       COMMON / NCONSTRAIN / CTHEMIN, CTHEMAX

       DATA  PI / 3.1415926D0 /
       DATA  MP / 938.27231D0/          ! proton mass, MeV/c*c
       DATA  MN / 939.56563D0 /                ! neutron mass
       DATA  MP0 / 134.9739D0 /         ! pi-0 mass
       DATA  MPM / 139.5675D0 /                ! pi-minus mass
       DATA  ME / 0.51099906D0 /               ! electron mass
       DATA BIND / 0.4E-3 /             ! binding E of pi- p system
       DATA CTHEMIN / -1.0 /            ! cos of max angle (180)
C                                       the neutron can go and still hit
C                                       the counters
       DATA CTHEMAX / 1.0 /             ! cos of min angle (164)...

       RR = 4*ME*ME / (MP0*MP0)         ! magic ratio R
       ETOTAL = MPM + MP - BIND         ! total energy of pi-minus
C                                       and proton, in their rest frame
       S = ETOTAL*ETOTAL
       PPI0 = 0.5*SQRT( (S - (MN+MP0)**2) * (S - (MP0-MN)**2) / S )
       EPI0 = SQRT(PPI0**2 + MP0**2)
       B = PPI0 / EPI0                  ! velocity of pi0
       GAMMA = 1.D0 / SQRT(1. - B*B)
       MAXEN = MPM + MP - MN - BIND     ! maximum energy available to the
C                                       2 electrons in the 3-body process
C                                           pi- p --> e+ e- n
       RETURN
       END
C***********************************************************************
       FUNCTION Dfun(X,Y)
C joint probability fn. for X and Y
       COMMON / USERMASSES / MP, MN, MP0, MPM, ME
       COMMON / CUTS / XMIN, XMAX
       COMMON / KINCONST / PI, RR
       DATA CONST / 5.80857D2 /
       DATA ASLOPE / 0.00 /
       DATA XMIN/0./
       DATA XMAX/1./
       IF (X .LT. RR) GOTO 100
       IF (X .LT. XMIN .OR. X .GT. XMAX) GOTO 100
       ETA = SQRT(1. - RR/X)
       IF (ABS(Y) .GT. ETA) GOTO 100

       FORM0 = 1.D0 + 2.*ASLOPE*X
       DFUN = CONST * (1. - X)**3 * (1. + RR/X + Y*Y) * (1./X)
     .                                  * FORM0
       RETURN
100    DFUN = 0.0D0
       RETURN
       END
C    ******************************************************************
C generate kinematics for Dalitz decay
       SUBROUTINE Thdz
       IMPLICIT REAL*4 (A-H, O-Z)
       REAL*4 MP, MN, MP0, MPM, ME, MAXEN
       REAL*4 PP0(3), PE0(3), PG0(3), BETA(3)
       REAL*4 A(2), X, Y
       REAL*4 PE(4), PP(4), PG(4)
       COMMON / NCONSTRAIN / CTHEMIN, CTHEMAX
       COMMON / RANDOM / II
       COMMON / USERMASSES / MP, MN, MP0, MPM, ME
       COMMON / KINCONST / PI, RR, MAXEN, ETOTAL, B, GAMMA
       COMMON / KIN / PE, PP, PG, X, Y
C use DIVON to get a random value of X, Y. X is normalized to its max.
C value, in this case, MPI0*MPI0.  NOTE: this is all in the pi0 frame!!!!
1       CALL Divondz(X,Y)
C total energy and momentum of e+ e- pair
        EPAIR = MP0/2. * (1. + X)
        PPAIR = MP0/2. * (1. - X)
C energy and momentum of positron and electron
        EP0 = (EPAIR + PPAIR*Y) / 2.
        EE0 = EPAIR - EP0
        PPM0 = SQRT(EP0*EP0 - ME*ME)
        PEM0 = SQRT(EE0*EE0 - ME*ME)
C opening angle between e+ e-
        COSA = (2.*EP0*EE0 + 2.*ME*ME - MP0*MP0*X) / (2.*PPM0*PEM0)
        IF (ABS(COSA) .GT. 1.) GOTO 1
        SINA = SQRT(1. - COSA*COSA)
C put e+ along z axis and spin e- around it by some arbitrary angle
        DEL = 2.*PI*RAN(II)
          COSDEL = COS(DEL)
        SINDEL = SIN(DEL)
        PEX = PEM0*COSDEL*SINA
        PEY = PEM0*SINDEL*SINA
        PEZ = PEM0*COSA
C throw positron into 26.3  degree half-angle cone.
        COSMIN = 0.896486
        COSMAX = 1.
        CALL CREATE(CTPOS, STPOS, CPPOS, SPPOS, COSMIN, COSMAX)
        PP0(1) = PPM0*CPPOS*STPOS
        PP0(2) = PPM0*SPPOS*STPOS
        PP0(3) = PPM0*CTPOS
C now connect the electron to this by rotating by theta and phi of e+
        PE0(1) = PEX*CTPOS*CPPOS - PEY*SPPOS + PEZ*STPOS*CPPOS
        PE0(2) = PEX*CTPOS*SPPOS + PEY*CPPOS + PEZ*STPOS*SPPOS
        PE0(3) = -PEX*STPOS + PEZ*CTPOS
C momentum of photon by momentum conservation
        PG0(1) = -PE0(1) - PP0(1)
        PG0(2) = -PE0(2) - PP0(2)
        PG0(3) = -PE0(3) - PP0(3)
        EG0 = MP0 - EP0 - EE0
C now we create the neutron, firing it into the neutron counters.
C NOTE that this is now in the lab frame. NOTE THAT FOR DALITZ EXPT
C THERE ARE NO NEUTRON COUNTERS SO CTHEMIN AND CTHEMAX ARE 1.0
        CALL Create(CTNEUT, STNEUT, CPNEUT, SPNEUT, CTHEMIN, CTHEMAX)
        UNEUTX = CPNEUT*STNEUT
        UNEUTY = SPNEUT*STNEUT  ! unit v. in neutron dir.
        UNEUTZ = CTNEUT
        BETA(1) = B*UNEUTX
        BETA(2) = B*UNEUTY
        BETA(3) = B*UNEUTZ
C Lorentz boost e+, e- and photon into lab frame
        BDOTPP = BETA(1)*PP0(1) + BETA(2)*PP0(2) + BETA(3)*PP0(3)
        BDOTPE = BETA(1)*PE0(1) + BETA(2)*PE0(2) + BETA(3)*PE0(3)
        BDOTPG = BETA(1)*PG0(1) + BETA(2)*PG0(2) + BETA(3)*PG0(3)
        DO I = 1, 3
          PP(I) = PP0(I) - BETA(I)*GAMMA*(EP0 - GAMMA/(GAMMA+1.)*BDOTPP)
          PE(I) = PE0(I) - BETA(I)*GAMMA*(EE0 - GAMMA/(GAMMA+1.)*BDOTPE)
          PG(I) = PG0(I) - BETA(I)*GAMMA*(EG0 - GAMMA/(GAMMA+1.)*BDOTPG)
        END DO
C energies in lab frame
        PP(4) = GAMMA*(EP0 - BDOTPP)
        PE(4) = GAMMA*(EE0 - BDOTPE)
        PG(4) = GAMMA*(EG0 - BDOTPG)
        RETURN
        END
C*********************************************************************
C little subroutine to generate a particle with momentum
C within a certain ring of solid angle about the z axis.
C
       SUBROUTINE Create(COSTHETA, SINTHETA, COSPHI, SINPHI,
     .                         COSMIN, COSMAX)
C
       IMPLICIT REAL*4 (A-H, O-Z)
       COMMON / KINCONST / PI
       COMMON / RANDOM / II
C
```

```
C phi is random between 0 and 2pi
C
       PHI = 2.*PI*RAN(II)
       COSPHI = COS(PHI)
       SINPHI = SIN(PHI)
C
C theta is random between min and max
C
       COSTHETA = (COSMAX - COSMIN)*RAN(II) + COSMIN
       SINTHETA = SQRT(1. - COSTHETA*COSTHETA)
C
       RETURN
       END
C *****************************************************
        SUBROUTINE Divondz(xval,yval)
        COMMON/SPECTRUM/ITYPE
      COMMON / KINCONST / PI, RR, MAXEN, ETOTAL, B, GAMMA
      COMMON / NCONSTRAIN / CTHEMIN, CTHEMAX
        DIMENSION X(1001), Z(1001), ZA(1001)
        LOGICAL INIT
        DATA INIT/.FALSE./
        DATA ISEED/123456789/
!       The open statement must be changed when reading the secret x table.
        IF(INIT) GOTO 30
        OPEN(UNIT=7, FILE='Xtable', STATUS='OLD', ACTION= "READ")
!       OPEN(UNIT=7, FILE='Secret', STATUS='OLD', ACTION= "READ")
        N=0
10      READ(7,*,END=20) xval,zval,zaval
!       The Xtable is organized with three columns. xval is the
!       usual x, zval is the integrated SI spectrum, and zaval is
!       the integrated SD spectrum.
        N=N+1
        IF(N.GT.1001) GO TO 20
        X(N)=xval
        Z(N)=zval
        IF(ITYPE.eq.2) Z(N)=zaval
        GOTO 10
20      CLOSE(7)
        NDAT=N
        INIT=.TRUE.
        IF(ITYPE.EQ.1) THEN
        WRITE(*,*)'Structure-independent spectrum will be generated.'
                ELSE IF (ITYPE.EQ.2) THEN
        WRITE(*,*)'Structure-dependent spectrum will be generated.'
                ELSE
        WRITE(*,*) 'NO CHOICE, I QUIT.'
        STOP
        END IF
30      CONTINUE
        zval=RAN(ISEED)
        n=1001
        CALL Locate(Z,n,zval,j)
        IF (j.EQ.0 .OR. j.EQ.N) GOTO 50
        m=4 ! Interpolation order
        k=MIN(MAX(j-(m-1)/2,1), n+1-m)  ! array offsets
        CALL Polint(Z(k),X(k),m,zval,xval,dx)
!       write(*,*) zval,xval,dx
!  The y distribution is generated with the classic
!  rejection technique.
        eta=sqrt(1.-RR/xval)
        fmax=DFUN(xval,eta)
        fmin=DFUN(xval,0.)
40      yval=eta-RAN(ISEED)*2.*eta
        zval=fmax*RAN(ISEED)
        if(zval .GT. DFUN(xval,yval)) GOTO 40
        RETURN
50      WRITE(*,*) 'ARRAY OUT OF BOUNDS j= ',j
        RETURN
        END
C****************************************************************
      SUBROUTINE Locate(xx,n,x,j)
      INTEGER j,n
      REAL x,xx(n)
      INTEGER jl,jm,ju
      jl=0
      ju=n+1
```

```
10    if(ju-jl.gt.1)then
        jm=(ju+jl)/2
        if((xx(n).ge.xx(1)).eqv.(x.ge.xx(jm)))then
          jl=jm
        else
          ju=jm
        endif
      goto 10
      endif
      if(x.eq.xx(1))then
        j=1
      else if(x.eq.xx(n))then
        j=n-1
      else
        j=jl
      endif
      return
      END
C*****************************************************************
      SUBROUTINE Polint(xa,ya,n,x,y,dy)
      INTEGER n,NMAX
      REAL dy,x,y,xa(n),ya(n)
      PARAMETER (NMAX=10)
      INTEGER i,m,ns
      REAL den,dif,dift,ho,hp,w,c(NMAX),d(NMAX)
      ns=1
      dif=abs(x-xa(1))
      do 11 i=1,n
        dift=abs(x-xa(i))
        if (dift.lt.dif) then
          ns=i
          dif=dift
        endif
        c(i)=ya(i)
        d(i)=ya(i)
11    continue
      y=ya(ns)
      ns=ns-1
      do 13 m=1,n-1
        do 12 i=1,n-m
          ho=xa(i)-x
          hp=xa(i+m)-x
          w=c(i+1)-d(i)
          den=ho-hp
          if(den.eq.0.)Write(*,*) x
          den=w/den
          d(i)=hp*den
          c(i)=ho*den
12      continue
        if (2*ns.lt.n-m)then
          dy=c(ns+1)
        else
          dy=d(ns)
          ns=ns-1
        endif
        y=y+dy
13    continue
      return
      END
\endd{verbatim}
\end{tiny}


\appendix
\subsection{Iteration}

\begin{tiny}
\begin{verbatim}
        PROGRAM Iteration
    EXTERNAL T4hist, Chi2
C       "Iteration" is a program to extract the NaI response parameters from
C       a photon "singles" spectrum.  The prominent features are
C       the uniform distribution of gamma energies from the decay,
C       pi-zero -> gamma + gamma, which ranges from 57 to 84 MeV,
C       and the 131.5 MeV peak from pi-minus + p -> pi-zero + n.
C       Note that the channel number, X(I), is close to three
```

```
C       times the actual energy measured by the NaI detector.
C       Deviations from this are adjusted with the scaling parameters,
C       P(2), P(3), and P(9), i.e. X=P*3*E.
        PARAMETER (NP=15, MP=16, FTOL=0.1)
        DIMENSION PVEC(16,15),CHIVEC(16), POLD(15)
      DIMENSION P(15)
      COMMON/INDATA/ NDAT,X(250),DX(250),Y(250),DY(250)
      COMMON/PAR/ PB, PA
        EPS=0.5
        NDIM=NP
C
      DATA P(1)/5.3E5/
      DATA P(2)/1./
      DATA P(3)/1./
      DATA P(4)/4.5/
      DATA P(5)/14.0/
      DATA P(6)/0.03/
      DATA P(7)/0.02/
      DATA P(8)/7.0E6/
      DATA P(9)/0.85/
      DATA P(10)/2.5/
      DATA P(11)/13./
      DATA P(12)/2.5E5/
      DATA P(13)/0.0/
      DATA P(14)/38./
      DATA P(15)/0.0/
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                              C
C       P(1) == Normalization of the pi0-box                   C
C       P(2) == Left edge of the pi0-box                       C
C       P(3) == Right edge of the pi0-box                      C
C       P(4) == Width parameter of higher energy tail          C
C       P(5) == Width parameeter of lower energy tail          C
C       P(6) == Energy dependence of P(4)                      C
C       P(7) == Energy dependence of P(5)                      C
C                                                              C
C       P(8) == Normalization of 129-peak                      C
C       P(9) == 129-Peak position                              C
C       P(10)== Width parameter of higher energy tail          C
C       P(11)== Width parameter of lower energy tail           C
C                                                              C
C       P(12)== Normalization of the exp. background           C
C       P(13)== Pedestal position                              C
C       P(14)== Exponential decay parameter of noise           C
C                                                              C
C       P(15)== Constant background                            C
C                                                              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CALL Xyread
C
      CHIMIN=1.E10
C
400   WRITE(*,*)'Which parameter(1-15) do you want to vary? (Zero to exi
     Ct.)'
      READ(*,*)IGO
    CHIMIN = 10.E10
C
      IF(IGO.EQ.0)GO TO 500
        WRITE(*,*) 'P(',IGO,')=',P(IGO)
          WRITE(*,*) 'Enter the parameter range (P) and Delta-P'
          WRITE(*,*)' PARmin, PARmax, Delta-P'
          READ(*,*) PMIN, PMAX, PD
C
1       P(IGO)=PMIN
        PHOLD=P(IGO)
        WRITE(*,*) 'LOOP NUMBER = ',IGO
        WRITE(*,*)
        WRITE(*,*)  'LOOP,  PARAMETER, CHISQ'
        DO 11 I=1,100
        CHITRY=CHI2(P)
        IF (CHITRY.LT.CHIMIN) THEN
        PHOLD=P(IGO)
        CHIMIN=CHITRY
        END IF
        WRITE(*,*) IGO,P(IGO),CHITRY
        P(IGO)=P(IGO)+PD
        IF(P(IGO).GT.PMAX) GO TO 12
11      CONTINUE
12      P(IGO)=PHOLD
C
        WRITE(*,*) 'Output summary, P(',IGO,')=',P(IGO)
        WRITE(*,*)
        WRITE(*,*) 'Look for output on file DataFit.dat'
        OPEN(UNIT=8, FILE="DataFit.dat", STATUS="REPLACE")
        DO 501, I=1,NDAT
501     WRITE(8,*) X(I), T4HIST(P,X(I))
        CLOSE(8)
        WRITE(*,*)
        WRITE(*,*) 'Any further changes?  Yes=1, No=2'
        READ(*,*) ILAST
        IF(ILAST.EQ.1) GO TO 400
500     WRITE(*,*) 'Begin multi-dimensional minimization.'
    WRITE(*,*)
        DO 100 I=1,MP
        DO 100 J=1,NP
100     PVEC(I,J)=P(J)
        DO 101 J=1,NP
101     PVEC(J+1,J)=PVEC(J+1,J)*(1+EPS)

        DO 102 I=1,MP
        DO 102 J=1,NP
        POLD(J)=P(J)
        P(J)=PVEC(I,J)
102     CHIVEC(I)=CHI2(P)
C   WRITE(*,*) "p =", PVEC(2, 1)
        CALL Amoeba(PVEC,CHIVEC,MP,NP,NDIM,FTOL,CHI2,ITER)
        OPEN(UNIT=13, FILE= "CHIPLOT.dat", STATUS="REPLACE")
        DO 103 J=1,NP
103     P(J)=PVEC(1,J)
        WRITE(*,*) 'Number of iterations: ',ITER
        WRITE(*,*)
        WRITE(*,*) 'Final chi square: ',CHIVEC(1)
        WRITE(*,*)
        WRITE(*,*) 'Output summary'
        WRITE(*,*)
        WRITE(*,*) 'Par Number, P-old, P-new'
        DO 105, J=1,15
105     write(*,149) J, POLD(J), PVEC(1,J)
        WRITE(*,*)
        WRITE(*,*) "Look for histograms on file DataFit.dat"
        WRITE(*,*) "The fitted paarameter values are in CHIPLOT.dat"
        WRITE(*,*)
    WRITE(13,*) 'Number of iterations: ',ITER
        WRITE(13,*)
        WRITE(13,*) 'Final chi square: ',CHIVEC(1)
        WRITE(13,*)
        WRITE(13,*) 'Output summary'
        WRITE(13,*)
        WRITE(13,*) 'Par Number, P-old, P-new'
        DO 106, J=1,15
106     write(13,149) J, POLD(J), PVEC(1,J)
        WRITE(13,*)
    CLOSE(13)
148 FORMAT(1X, A3, F6.3)
149 FORMAT(I2, E12.4, E12.4)
150     FORMAT(1X, A7, F6.3)
        OPEN(UNIT=8, FILE="DataFit.dat", STATUS="REPLACE")
        DO 502, I=1,NDAT
502     WRITE(8,*) X(I), T4HIST(P,X(I))
        CLOSE(8)
C
      STOP
      END
C*************************************************************
        FUNCTION Chi2(P)
        COMMON/INDATA/ NDAT,X(250),DX(250),Y(250),DY(250), ITER
        DIMENSION P(15)
        CHI2=0.0
        DO 100 J=1,NDAT
        CHI2=CHI2+(T4HIST(P,X(J))-Y(J))**2/(DY(J)*DY(J))
100     CONTINUE
```

```
        RETURN
        END
C***********************************************************
      SUBROUTINE Amoeba(p,y,mp,np,ndim,ftol,funk,iter)
      INTEGER iter,mp,ndim,np,NMAX,ITMAX
      REAL ftol,p(mp,np),y(mp),funk,TINY
      PARAMETER (NMAX=20,ITMAX=5000,TINY=1.e-10)
      EXTERNAL Funk
C     USES Amotry, Funk
      INTEGER i,ihi,ilo,inhi,j,m,n
      REAL rtol,sum,swap,ysave,ytry,psum(NMAX),amotry
      iter=0
C      WRITE(*,*) "p =", p
C       STOP
C       WRITE(*,*) "y =", y
C       STOP
C       WRITE(*,*) "mp =", mp
C       STOP
C       WRITE(*,*) "np =", np
C       STOP
C       WRITE(*,*) "ndim =", ndim
C       STOP
C       WRITE(*,*) "ftol =", ftol
C       STOP
C       WRITE(*,*) "funk =", funk(p)
C   WRITE(*,*) "Chi2 =", CHI2(p)
C       STOP
C       WRITE(*,*) "iter =", iter
C   STOP
1     do 12 n=1,ndim
        sum=0.
        do 11 m=1,ndim+1
          sum=sum+p(m,n)
11      continue
        psum(n)=sum
12    continue
2     ilo=1
      if (y(1).gt.y(2)) then
        ihi=1
        inhi=2
      else
        ihi=2
        inhi=1
      endif
      do 13 i=1,ndim+1
        if(y(i).le.y(ilo)) ilo=i
        if(y(i).gt.y(ihi)) then
          inhi=ihi
          ihi=i
        else if(y(i).gt.y(inhi)) then
          if(i.ne.ihi) inhi=i
        endif
13    continue
      rtol=2.*abs(y(ihi)-y(ilo))/(abs(y(ihi))+abs(y(ilo))+TINY)
      if (rtol.lt.ftol) then
        swap=y(1)
        y(1)=y(ilo)
        y(ilo)=swap
        do 14 n=1,ndim
          swap=p(1,n)
          p(1,n)=p(ilo,n)
          p(ilo,n)=swap
14      continue
    return
      endif
      if (iter.ge.ITMAX) pause 'ITMAX exceeded in amoeba'
      iter=iter+2
      ytry=amotry(p,y,psum,mp,np,ndim,funk,ihi,-1.0)
      if (ytry.le.y(ilo)) then
        ytry=amotry(p,y,psum,mp,np,ndim,funk,ihi,2.0)
      else if (ytry.ge.y(inhi)) then
        ysave=y(ihi)
        ytry=amotry(p,y,psum,mp,np,ndim,funk,ihi,0.5)
        if (ytry.ge.ysave) then
          do 16 i=1,ndim+1
            if(i.ne.ilo)then
```

```
              do 15 j=1,ndim
                psum(j)=0.5*(p(i,j)+p(ilo,j))
                p(i,j)=psum(j)
15            continue
              y(i)=funk(psum)
            endif
16        continue
          iter=iter+ndim
          goto 1
        endif
      else
        iter=iter-1
      endif
      goto 2
      END
C***************************************************************
      FUNCTION Amotry(p,y,psum,mp,np,ndim,funk,ihi,fac)
      INTEGER ihi,mp,ndim,np,NMAX
      REAL amotry,fac,p(mp,np),psum(np),y(mp),funk
      PARAMETER (NMAX=20)
      EXTERNAL funk
CU    USES funk
      INTEGER j
      REAL fac1,fac2,ytry,ptry(NMAX)
      fac1=(1.-fac)/ndim
      fac2=fac1-fac
      do 11 j=1,ndim
        ptry(j)=psum(j)*fac1-p(ihi,j)*fac2
11    continue
      ytry=funk(ptry)
      if (ytry.lt.y(ihi)) then
        y(ihi)=ytry
        do 12 j=1,ndim
          psum(j)=psum(j)-p(ihi,j)+ptry(j)
          p(ihi,j)=ptry(j)
12      continue
      endif
      amotry=ytry
      return
      END
C***************************************************************
       SUBROUTINE Blur(P,M)
C
       IMPLICIT REAL (A-H,O-Z)
       DIMENSION P(*)
       DIMENSION X(500),XB(151),E(500),Y(500),YB(151),YC(500)
        COMMON/VALUE/YC
        COMMON/PAR/ CP, DP
        DATA INIT/0/,ELO/54.887/,EHI/82.97/
C
CCCCCCCCuncomment to debugCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                               C
C       P(1) == Normalization of the pi0-box                    C
C       P(2) == Left edge of the pi0-box                        C
C       P(3) == Right edge of the pi0-box                       C
C       P(4) == Width parameter of higher energy tail           C
C       P(5) == Width parameeter of lower energy tail           C
C       P(6) == Energy dependence of P(4)                       C
C       P(7) == Energy dependence of P(5)                       C
C                                                               C
C       P(8) == Normalization of 129-peak                       C
C       P(9) == 129-Peak position                               C
C       P(10)== Width parameter of higher energy tail           C
C       P(11)== Width parameter of lower energy tail            C
C                                                               C
C       P(12)== Normalization of the exp. background            C
C       P(13)== Pedestal position                               C
C       P(14)== Exponential decay parameter of noise            C
C                                                               C
C       P(15)== Constant background                             C
C                                                               C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       OPEN(UNIT=10,FILE='YCTEST',STATUS='REPLACE')
C       P(1)=5.35E5
C       P(2)=1.
C       P(3)=1.
```

```
C       P(4)=4.5
C       P(5)=14.0
C       P(6)=0.03
C       P(7)=0.02
C       P(8)=9.7E6
C       P(9)=1.
C       P(10)=2.5
C       P(11)=9.2
C       P(12)=2.5E5
C       P(13)=0.0
C       P(14)=38.
C       P(15)=0.0
C       M=0

        IF(INIT.EQ.0)THEN
          INIT=1
          DO 10 J=1,151
10        XB(J)=(J-112.0)
          DO 20 J=1,500
20        X(J)=J
        END IF
C
        IF(M.GT.0)GOTO 1
C
        IXLO=INT(3.*57.0*P(2))
        IXHI=INT(3.*83.6*P(3))
        NBOX=IXHI-IXLO
C
        DO 30 I=1,500
        Y(I)=0.0
30      E(I)=0.0
C
        DO 40 I=1,500
        IF(I.GE.IXLO.AND.I.LE.IXHI)Y(I)=1.0
        IF(Y(I).GT.0.0)E(I)=ELO+(EHI-ELO)/NBOX*(I-IXLO)
40      CONTINUE
C ********************
1       CONTINUE
C ********************

        DO 50 I=1,500
50      YC(I)=0.0
C
        DO 101 K=1,500
        IF(Y(K).EQ.0.0)GOTO 101
        CP=P(4)*E(K)**P(6)
        DP=P(5)*E(K)**P(7)
        RNORM=2.*DP*EXP(CP*CP/4./DP/DP)
        DO 100 J=1,151
        L=J+K-112
        IF(L.LT.1.OR.L.GT.500)GOTO 100
        XTEST=XB(J)/CP
C       WRITE(*,*) J, CP, XB(J)/CP, XTEST

        YB(J)=EXP(XB(J)/DP)*(1.-ERF(XTEST))/RNORM
        YC(L)=YC(L)+YB(J)*Y(K)

100     CONTINUE
101     CONTINUE
C
C       DO 102 L=1,500
C102    WRITE(10,*) YC(L)
C
C       CLOSE(10)
        RETURN
        END
C*******************************************************************
      FUNCTION Erf(x)
      REAL erf,x
CU    USES gammp
      REAL gammp
        if(x.lt.0.)then
        erf=-gammp(.5,x**2)
      else
        erf=gammp(.5,x**2)
      endif
      return
      END
C*****************************************************************
      FUNCTION Gammln(xx)
      REAL gammln,xx
      INTEGER j
      DOUBLE PRECISION ser,stp,tmp,x,y,cof(6)
      SAVE cof,stp
      DATA cof,stp/76.18009172947146d0,-86.50532032941677d0,
     *24.01409824083091d0,-1.231739572450155d0,.1208650973866179d-2,
     *-.5395239384953d-5,2.5066282746310005d0/
      x=xx
      y=x
      tmp=x+5.5d0
      tmp=(x+0.5d0)*log(tmp)-tmp
      ser=1.000000000190015d0
      do 11 j=1,6
        y=y+1.d0
        ser=ser+cof(j)/y
11    continue
      gammln=tmp+log(stp*ser/x)
      RETURN
      END
C*****************************************************************
      FUNCTION Gammp(a,x)
      REAL a,gammp,x
CU    USES gcf,gser
      REAL gammcf,gamser,gln
      if(x.lt.0..or.a.le.0.)pause 'bad arguments in gammp'
      if(x.lt.a+1.)then
        call gser(gamser,a,x,gln)
        gammp=gamser
      else
        call gcf(gammcf,a,x,gln)
        gammp=1.-gammcf
      endif
      return
      END
C*****************************************************************
         SUBROUTINE Gcf(gammcf,a,x,gln)
      INTEGER ITMAX
      REAL a,gammcf,gln,x,EPS,FPMIN
      PARAMETER (ITMAX=100,EPS=3.e-7,FPMIN=1.e-30)
CU    USES gammln
      INTEGER i
      REAL an,b,c,d,del,h,gammln
      gln=gammln(a)
      b=x+1.-a
      c=1./FPMIN
      d=1./b
      h=d
      do 11 i=1,ITMAX
        an=-i*(i-a)
        b=b+2.
        d=an*d+b
        if(abs(d).lt.FPMIN)d=FPMIN
        c=b+an/c
        if(abs(c).lt.FPMIN)c=FPMIN
        d=1./d
        del=d*c
        h=h*del
        if(abs(del-1.).lt.EPS)goto 1
11    continue
      pause 'a too large, ITMAX too small in gcf'
1     gammcf=exp(-x+a*log(x)-gln)*h
       RETURN
       END
C*****************************************************************
      SUBROUTINE Gser(gamser,a,x,gln)
      INTEGER ITMAX
      REAL a,gamser,gln,x,EPS
      PARAMETER (ITMAX=100,EPS=3.e-7)
CU    USES gammln
      INTEGER n
      REAL ap,del,sum,gammln
      gln=gammln(a)
      if(x.le.0.)then
        if(x.lt.0.)pause 'x < 0 in gser'
        gamser=0.
        return
      endif
      ap=a
      sum=1./a
      del=sum
      do 11 n=1,ITMAX
        ap=ap+1.
        del=del*x/ap
        sum=sum+del
        if(abs(del).lt.abs(sum)*EPS)goto 1
11    continue
      pause 'a too large, ITMAX too small in gser'
1     gamser=sum*exp(-x+a*log(x)-gln)
      return
      END
C ******************************************************************
        FUNCTION T4hist(P,XIN)
C
        IMPLICIT REAL (A-H,O-Z)
        COMMON/INDATA/ NDAT,X(250),DX(250),Y(250),DY(250)
        DIMENSION P(15)
        COMMON/VALUE/YC(750)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                               C
C       P(1) == Normalization of the pi0-box                    C
C       P(2) == Left edge of the pi0-box                        C
C       P(3) == Right edge of the pi0-box                       C
C       P(4) == Width parameter of higher energy tail           C
C       P(5) == Width parameeter of lower energy tail           C
C       P(6) == Energy dependence of P(4)                       C
C       P(7) == Energy dependence of P(5)                       C
C                                                               C
C       P(8) == Normalization of 129-peak                       C
C       P(9) == 129-Peak position                               C
C       P(10)== Width parameter of higher energy tail           C
C       P(11)== Width parameter of lower energy tail            C
C                                                               C
C       P(12)== Normalization of the exp. background            C
C       P(13)== Pedestal position                               C
C       P(14)== Exponential decay parameter of noise            C
C                                                               C
C       P(15)== Constant background                             C
C                                                               C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

        DATA P1OLD/0.0/,P2OLD/0.0/,P3OLD/0.0/,P4OLD/0.0/
        DATA P5OLD/0.0/,P6OLD/0.0/,P7OLD/0.0/
C
        M=0
C
        IF(P(2).EQ.P2OLD.AND.P(3).EQ.P3OLD)M=1
        IF(M.EQ.1.AND.P(4).EQ.P4OLD.AND.P(5).EQ.P5OLD)M=2
        IF(M.EQ.2.AND.P(6).EQ.P6OLD.AND.P(7).EQ.P7OLD)GOTO 1
C
        P1OLD=P(1)
        P2OLD=P(2)
        P3OLD=P(3)
        P4OLD=P(4)
        P5OLD=P(5)
        P6OLD=P(6)
        P7OLD=P(7)
C
        CALL Blur(P,M)
        BOX=P(1)*YC(INT(3.*XIN))
        GOTO 10
C
1       CONTINUE
        P1OLD=P(1)
        BOX=P(1)*YC(INT(XIN))
10      CONTINUE
C       XIN=333.
        PEAK=EXP((XIN-3.*131.5*P(9))/P(11))*
     C(1.-ERF((XIN-3.*131.5*P(9))/P(10)))
```

```
        PEAK=PEAK/(2.*P(11)*EXP(P(10)*P(10)/4./P(11)/P(11)))
        PEAK=P(8)*PEAK
C
20      BACK=P(12)*EXP(-(XIN-P(13))/P(14))
C
30      BACK=BACK+P(15)
C
        T4HIST=BOX+PEAK+BACK
C   WRITE(*,*) "T4hist =", T4HIST
C   STOP
C
        RETURN
        END
C ****************************************************************
        SUBROUTINE Xyread
        IMPLICIT REAL (A-H,O-Z)
        INTEGER :: CL
        COMMON/INDATA/ NDAT,X(250),DX(250),Y(250),DY(250)
        CHARACTER*25 FILENAME
        WRITE(*,100)
100     FORMAT(/' enter filename of data: ',$)
        READ(*,101) FILENAME
101     FORMAT(A)
        WRITE(*,*) FILENAME
        WRITE(*,102)
102     FORMAT(' Enter first and last channel number ')
        READ(*,*)XMIN,XMAX
        WRITE(*,*)
        WRITE(*,*) XMIN, XMAX
        OPEN(UNIT=7,FILE=FILENAME,STATUS='OLD',ACTION= "READ")
        N=0
150     READ(7,*,END=200)XVAL,YVAL
        IF(XVAL.LT.XMIN.OR.XVAL.GT.XMAX)GOTO 150
        N=N+1
        X(N)=XVAL
        Y(N)=YVAL
        DX(N)=.5
        DY(N)=SQRT(ABS(Y(N)))
        GOTO 150
200     NDAT=N
        CLOSE(7)
201     WRITE(*,300)N,X(1),X(N)
300     FORMAT(' Number of points read in : ',I4,/,
     &         ' Range of Nchannel : ',G10.4,' to ',G10.4)
        RETURN
        END
```

.1 Sorter

```
!***************************************************************************
!
!  PROGRAM: Sorter (for f90 compiler)
!
!  PURPOSE:  Analyze data files written by Dalitz
!
!***************************************************************************
        Program Sorter
        implicit real(M)
        character*25 Filename
        logical flag
  ! Variables
        dimension PP(4), PE(4), PG(4), DETEC(3)
        integer XSPEC(100),EPSPEC(100),EESPEC(100),TSPEC(180)
        data N0/0/, N1/0/, N2/0/, N3/0/, N4/0/,N5/0/
        data ME/0.511/, PI/3.14159/, MP0/134.9739/
        data THETAMAX/0.2618/  ! 15 DEGREES OPENING ANGLE
        TMAX=TAN(THETAMAX)
        CMAX=COS(THETAMAX)
        data XSPEC/100*0./, EPSPEC/100*0./
        data EESPEC/100*0./, TSPEC/180*0./
  ! Body of Sorter
        write(*,*) 'Enter filename of data'
        read(*,1) Filename
1       format(A)
```

```
        write(*,*) 'Enter minimum x cut'
        read(*,*) XMIN
        write(*,*) 'Enter minimum energy cut'
        read(*,*) EMIN
        write(*,*) 'Enter the detector opening angle in degrees'
        read(*,*) TOPEN
        COPEN=COS(TOPEN*PI/180.)
        SOPEN=SIN(TOPEN*PI/180.)
        DETEC(1)=SOPEN
        DETEC(2)=0.
        DETEC(3)=COPEN
        open(unit=7, File=Filename, Status='old', ACTION= "READ")
        open(unit=8, File='Xspec', Status='replace')
        open(unit=9, File='EPspec', Status='replace')
        open(unit=10, File='EMspec', Status='replace')
        open(unit=11, File='Tspec', Status='replace')
5       read(7,10,IOSTAT=istat) I,X,Y
10      format(I10,4X,2(E10.4,4X))
    if(istat<0) go to 100
    if(istat>0) go to 5
        read(7,20) PP(1),PP(2),PP(3),PP(4)
        read(7,20) PE(1),PE(2),PE(3),PE(4)
        read(7,20) PG(1),PG(2),PG(3),PG(4)
20      format (4(E10.4,4X))
        N0=N0+1
        flag=.true.
!  Analyze these events.
!  Does PP hit the detector?
        PPperp2=PP(1)*PP(1)+PP(2)*PP(2)
        PPperp=sqrt(PPperp2)
        TanPP=PPperp/PP(3)
        if(TanPP .gt. TMAX) go to 5
        N1=N1+1
!  Does PE hit the detector?
        PP2=0.
        PE2=0.
        do I=1,3
                PE2=PE2+PE(I)*PE(I)
                PP2=PP2+PP(I)*PP(I)
        end do
        PEmag=sqrt(PE2)
        PPmag=sqrt(PP2)
!  Construct the dot product of DETEC and a unit vector
!    in the PE direction.
        PdotD=0.
        do I=1,3
                PdotD=PdotD+PE(I)*DETEC(I)/PEmag
        end do
        if( PdotD .gt. CMAX) then
                N2=N2+1
        else
                flag=.false.
        end if
        DETangle=180.*acos(PdotD)/PI
!  All events that hit the positron detector are histogramed.
!  Calculate the opening angle
        PdotE=0.
        do I=1,3
                PdotE=PdotE+PP(I)*PE(I)
        end do
        PdotE=PdotE/(PPmag*PEmag)
        PEangle=180.*acos(PdotE)/PI
!  Now impose the energy cuts.
        if(PP(4).gt.EMIN .and. PE(4).gt.EMIN) then
                N3=N3+1
        else
                flag=.false.
        end if
!  Finally the x cut.
        Xval=2.*ME*ME
        do I=1,3
                Xval=Xval-2.*PP(I)*PE(I)
        end do
        Xval=Xval+2.*PP(4)*PE(4)
        Xval=Xval/(MP0*MP0)
        if(Xval .gt. XMIN) then
```

```
                N4=N4+1
        else
                flag=.false.
        end if

        if(flag.eqv..true.) N5=N5+1
!  Histogram results.
        I=Xval*100.+1.
        if(I.lt.0 .or. I.gt.100) go to 30
        Xspec(I)=Xspec(I)+1
        I=PP(4)+1.
        if(I.lt.0 .or. I.gt.100) go to 30
        EPspec(I)=EPspec(I)+1
        I=PE(4)+1.
        if(I.lt.0 .or. I.gt.100) go to 30
        EEspec(I)=EEspec(I)+1
        I=PEangle+1.
        if(I.lt.0 .or. I.gt.180) go to 30
        TSPEC(I)=TSPEC(I)+1
        go to 5
!  Write up the summary and histograms.
30      write(*,*) 'Out of bounds error I = ',I
        go to 5
100     continue
        write(*,*) 'Filename = ',Filename
        write(*,*) 'Events read from file = ',N0
        write(*,*) 'Events hitting positron detector = ',N1
        write(*,*) 'Events hitting electron detector = ',N2
        write(*,*) 'Events passing energy cuts = ',N3
        write(*,*) 'Events passing xmin cut = ',N4
        write(*,*) 'Events passing all cuts = ',N5
        do I=1,100
                write(8,*)  XSPEC(I)
                write(9,*)  EPSPEC(I)
                write(10,*) EESPEC(I)
        end do
        do I=1,180
                write(11,*) TSPEC(I)
        end do
        close(7)
        close(8)
         close(9)
        close(10)
        close(11)
        end
```

A Appendix C: How Simulated Data were Generated

Recall, the data are from the successive processes

$$\begin{aligned} \pi^- p \rightarrow \quad & \pi^0 \quad n & (52)\\ \hookrightarrow \quad & \gamma e^- e^+. & (53) \end{aligned}$$

There are three simplications in the simulated data:

1. We assume that a known number of π^0's (and nothing else) are produced precisely in the center of the proton target in which the π^-'s stop.

2. The decay electrons in (53) are detected in two NaI detectors. These are large crystals that give off a flash of light when electrons pass into them. We measure the amount of light given off with photomultiplier tubes, and use that to deduce the energy of the electrons. We assume that the two NaI detectors are identical, and that they both subtend a cone with its apex at the production point with a half-angle of 15^0. We also assume that particles inside these cones are detected with 100% efficiency, while those outside of the cone are not detected at all.

3. To avoid you having to remove the effects of the finite angular and energy resolution of the detectors, we assume that the detectors determine the directions and energies of the particles with perfect accuracy. In contrast, the calibration project in § 6 makes use of real data.

If you are not familiar with Monte Carlo techniques, now is the time to review them. Chapters 6 and 7 in *Computational Physics* [CP] is a good place to look for a discussion of these techniques.

The variables x and y, in (12)-(13), representing the sum and differences of the electron pairs energy, are chosen randomly but with a weighting given by the Kroll-Wada distribution (22). In theory one should generate the distribution over the complete, kinematically-allowed range $r \leq x \leq 1$. However, in order not to waste time, we leave out the small-x events since our apparatus could not detect most of them anyway. Next, the direction of the emitted e^+ is chosen randomly within the full 4π range of solid angle. However, to save time and not create events that cannot be observed, its direction is restricted to a cone centered around one of the detectors.

The energy of the two decay electrons and the angle between them θ (their "opening angle") is determined from the values of x and y.

The azimuthal angle of the electron (with respect to the positron direction) in the π^0 rest frame is determined randomly. [How is this different from opening angle??] All momenta are transformed from the π^0 rest frame into the lab system. Since we have used the photon's momentum vector to orient the two electrons, and since this photon is emitted symetrically over the entire sphere, there is no prefer direction for the photon. Accordingly, the transformation velocity β is given a random chosen direction.

The resulting "event" is written to a file with the Fortran code:

```
!  Finally, write everything to a file.
      WRITE(9,100) I,X,Y
100   FORMAT(I10,4X,2(E10.4,4X))
      WRITE(9,110) PP(1),PP(2),PP(3),PP(4)
      WRITE(9,110) PE(1),PE(2),PE(3),PE(4)
      WRITE(9,110) PG(1),PG(2),PG(3),PG(4)
110   FORMAT(4(E10.4,4X))
```

This section of code is executed for each event, and leads to a a huge output file! Here, `I` is the event number, `X` and `Y` are the x and y variables (12)-(13), `PP`, `PE`, and `PG` are the positron's, electron's, and gamma's momentum vectors respectively, with the fourth component of each momentum vector the energy of the particle.

B Reprint of Original Paper

References

[PL] *Measurement of the slope of the π^0 electromagnetic form factor*, F. Farzanpay, P. Gumplinger, A. Stetz, j.-M. Poutissou, I. Blevis, M. Hassinoff, C.J. Virtue, C.E. Waltham, B.C. Robertson, T. Mulera, A. Shor, J. Lowe, and S.H. Chew, Physocs Letters **B 278** (1992) 413-418.

[CP] *Computational Physics, Problem Solving with Computers*, R.H. Landau and M.J. Paez, John wiley, N.Y., 1997.