The New BNL Partial Wave Analysis Programs

John P. Cummings   Dennis P. Weygand

February 18, 1998

1 Introduction

Experiment E852 at Brookhaven National Laboratory is a meson spectroscopy experiment which took data at the Multi-Particle Spectrometer facility of the Alternating Gradient Syncrotron. Upgrades to the spectrometer's data acquisition and trigger electronics allowed over 900 million data events, of numerous topologies, to be recorded to tape in 1995 running alone. One of the primary goals of E852 is identification of states beyond the quark model, *i.e.*, states with gluonic degrees of freedom. Identification of such states involves the measurement of a systems spin–parity.

Such a measurement is usually done using Partial Wave Analysis. Programs to perform such analyses exist, in fact, one was written at BNL and used in previous experiments by some of our group. This program, however, was optimized for a particular final state, and modification to allow analysis of the broad range of final states in E852 would have been difficult. We therefore decided to write a new program, with an eye towards generality that would allow analysis of a large class of reactions.

2 Partial Wave Analysis Formalism

2.1 General Description of Pwa

Partial Wave Analysis, or PWA, is a technique used in hadron spectroscopy to extract information about the spin-parity and decay properties of resonances produced in hadronic interactions. Typically, these resonances are produced at accelerator experiments via a variety of production mechanisms. The resonances produced by these methods also appear in the decay products of other well known resonances, such as the J/ψ, and can be studied there. Although the tools we describe here could be used in an investigation that uses any of these production mechanisms, perhaps with slight modification, so far they have been used only in peripheral production experiments, and so we will use this type of production as an illustrative example in this paper.

The general idea is to parameterize the intensity distribution in terms of variables that have physical meaning when interpreted as properties of intermediate states in a particular reaction. In principle, any complete set of functions which span the appropriate space can be used. Although many parameterizations are possible, for instance a (almost) purely mathematical description in terms of the moments, we choose an expansion in terms of intermediate resonances and their decays. This has at least two advantages. Firstly, it allows us to take advantage of physics such as conservation laws to limit the number of terms we must include in our expansion to get a good description of the intensity distribution. Secondly, it allows a more direct interpretation of our results. A moment analysis, for instance, requires a complicated mapping from moments to physical states to understand the results in all but the simplest of cases.

The formalism we have used in implementing our system is based on the papers of Chung [Chu95] and Chung and Trueman [CT75]. The intensity distribution is written as a sum amplitudes, squared appropriately to account for interference

$$I(\tau) = \sum_{\alpha} \left\{ \left| \sum_{\beta} {}^{\alpha}\psi_{\beta}(\tau) \right|^2 \right\}. \tag{1}$$

The variable τ represents the set of variables necessary to define a configuration of the final state being investigated. It typically includes the angles of the decay products in various reference frames, masses of decay products, etc. The subscripts α and β are the parameters that describe the partial wave decomposition we are using, α specifying properties of the different intermediate states that do not interfere, such as the spin states of the incoming or outgoing particles in the detector. The subscript β, on the other hand, represents the properties whose differing values do interfere, for instance the spin states of broad resonances produced as intermediate states in a sequential decay.

In a peripheral production experiment, the amplitudes in the expansion can be drawn as shown in Figure 1. Guided by this picture, the amplitude $\psi_{\alpha\beta}(\tau)$ is factored into two parts: V, the amplitude to produce the state X, and A, the amplitude for the state X to decay into the final state observed.

These amplitudes are written in the reflectivity basis [CT75], which takes into account parity conservation in the production process by writing the amplitudes in terms of eigenstates of reflection through the production plane. The reflectivity of the amplitude, denoted ϵ, is defined to coincide with the naturality of the exchanged Regge trajectory. Because of parity conservation, waves of differing ϵ do not interfere. Also, amplitudes with different relative spin configurations for the incoming and outgoing baryon will not interfere. The spin configuration of the amplitudes is labeled by k, sometimes referred to as the rank of the fit, and is typically 2 for a spin $\frac{1}{2}$ baryon.

In the reflectivity basis our sum over amplitudes splits into four non-interfering sets of fully interfering amplitudes, or $\alpha = \{\epsilon, k\}$. While both the production and decay amplitudes depend on epsilon, the decay amplitude does not depend on k, since the decay amplitude for a particular state X cannot depend on what the proton spin did during the production process. Similarly, the production amplitude does not depend on τ, the configuration of the final state the X

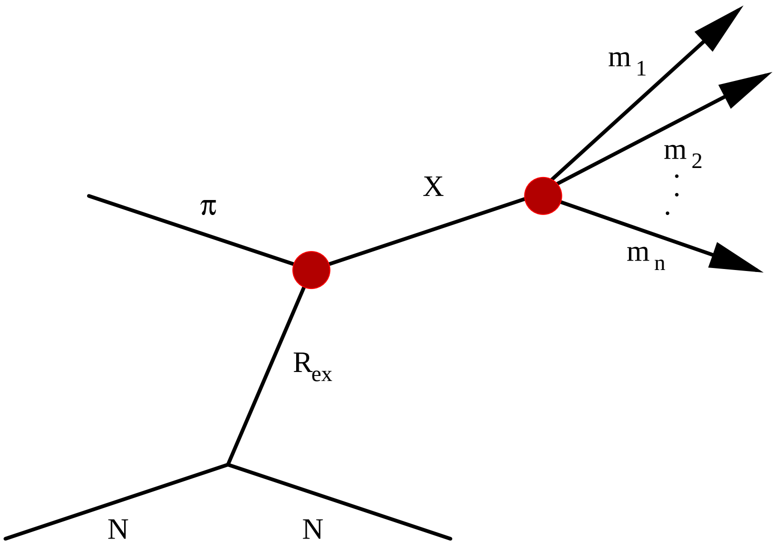

Figure 1: Diagram representing the amplitude $\psi_{\alpha\beta}(\tau)$. The intermediate state X is produced by the exchange of a Reggion R_{ex} between the π beam and the nucleon target.

decays into.

The intensity distribution now becomes

$$I(\tau) = \sum_{\epsilon,k} \left\{ \left| \sum_{\beta} {}^{\epsilon}V_{k\beta} {}^{\epsilon}A_{\beta}(\tau) \right|^{2} \right\}. \tag{2}$$

The decay amplitudes ${}^{\epsilon}A_{k\beta}(\tau)$ can be calculated for each event. By varying the unknown production amplitudes ${}^{\epsilon}V_{k\beta}$ the predicted intensity distribution above is matched as closely as possible to the observed intensity. This is done through an extended maximum likelihood fit. During the fitting process, the finite acceptance of the detector is taken into account on a wave by wave basis, *i.e.*, the processes corresponding to different amplitudes in the fit (and their interferences) will have different distributions, and therefore different acceptances, which are corrected for separately.

The function which is maximized in the likelihood fit is

$$\ln \mathcal{L} = \sum_{i}^{n} \ln \left[\sum_{k,\epsilon,\beta,\beta'} {}^{\epsilon}V_{k\beta} {}^{\epsilon}V^{*}_{k\beta'} {}^{\epsilon}A_{\beta}(\tau_i) {}^{\epsilon}A^{*}_{\beta'}(\tau_i) \right] - n \left[\sum_{k,\epsilon,\beta,\beta'} {}^{\epsilon}V_{k\beta} {}^{\epsilon}V^{*}_{k\beta'} {}^{\epsilon}\Psi^{x}_{\beta\beta'} \right] \tag{3}$$

The first sum is over data events, where the term being summed over is simply the intensity $I(\tau_i)$ for each event. The second term contains the accepted normalization integrals ${}^{\epsilon}\Psi^{x}_{\beta\beta'}$ where the superscript x denotes accepted. This integral is evaluated numerically:

$${}^{\epsilon}\Psi^{x}_{\beta\beta'} = \frac{1}{M_x} \sum_{i}^{M_x} {}^{\epsilon}A_{\beta}(\tau_i) {}^{\epsilon}A^{*}_{\beta'}(\tau_i). \tag{4}$$

The sum is over an accepted Monte-Carlo data set of M_x events. A similar integral is calculated for the raw Monte-Carlo data set, to be used in the calculation of observables. For instance, the number of acceptance corrected events the fit predicts is

$$N = \frac{n}{\eta_x} \sum_{\epsilon k\beta\beta'} {}^{\epsilon}V_{k\beta} {}^{\epsilon}V^{*}_{k\beta'} {}^{\epsilon}\Psi_{\beta\beta'}. \tag{5}$$

${}^{\epsilon}\Psi_{\beta\beta'}$ here is the raw normalization integral. By varying the range of the values of $\{\epsilon k\beta\beta'\}$ included in the sum, the number of events due to different combinations of amplitudes can be determined.

2.2 Decay Amplitudes

Calculation of decay amplitudes for the resonance X is done recursively using the isobar model, regarding the n-body final state as the result of a series of sequential decays (usually two body) through intermediate states known as the isobars. The amplitude for X to decay into the final state is then simply the

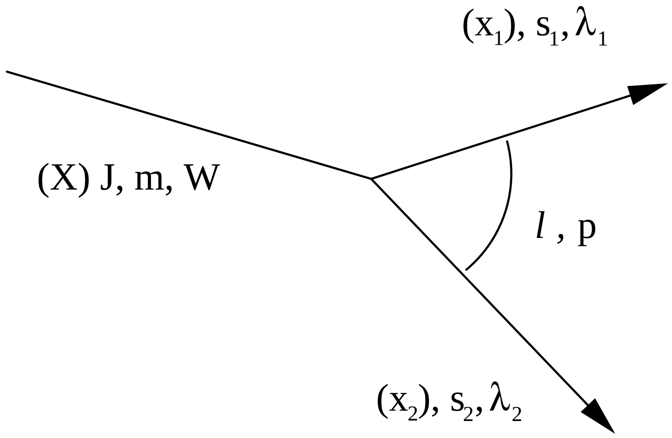

Figure 2: Decay of the resonance X into x_1 and x_2.

amplitude for X to decay into its immediate children times the amplitude for each of its children to decay.

The initial decay of the X into its children is evaluated in the Gottfried-Jackson frame. This frame is a rest frame of the resonance X with the z axis in the direction of the beam and the y axis perpendicular to the production plane. The quantities used in defining the amplitude are shown schematically in Figure 2 The state X of mass W has spin J with z projection m. It decays into children x_i with spin σ_i and helicity λ_i. These children have a breakup momentum $\mathbf{p}$ and relative orbital angular momentum l. The decay amplitude can then be written down [Chu71]

$$A_X = \tilde{l} \sum_{\lambda} D^{J*}_{m\lambda}(\Omega)(l0s\lambda|J\lambda)(s_1\lambda_1 s_2 - \lambda_2|s\lambda)F_l(p)a_{ls}A_{x_1}A_{x_2} \tag{6}$$

where $\lambda = \lambda_1 - \lambda_2$ and $\vec{s} = \vec{s}_1 + \vec{s}_2$, *i.e.*, $\vec{s}$ is the total spin of the two children and λ is the component of $\vec{s}$ in the direction defined by x_i's momentum. The A_{x_i} are the decay amplitudes of each child.

The $\tilde{\ell} \equiv (2\ell + 1)^{1/2}$ factor, along with the two Clebsch-Gordan coefficients, come from the fact that we are using helicity states and must relate the helicity coupling constant to the ℓs coupling constant $a_{\ell s}$ we want to find through the partial wave expansion. [Chu71, section 4.3]

Rotational properties of the helicity states lead to the D function $D^{J*}_{m\lambda}(\Omega)$ where $\Omega = (\theta, \phi, 0)$ are the Euler angles of x_1 in the Gottfried–Jackson frame. The choice of the third angle $\gamma = 0$ defines a phase convention different from that of Jacob and Wick [JW59], who choose $\gamma = -\phi$.

$F_\ell(p)$ is an angular momentum barrier factor added to give the amplitude the correct behavior near threshold. We used the Blatt-Weisskopf centrifugal-

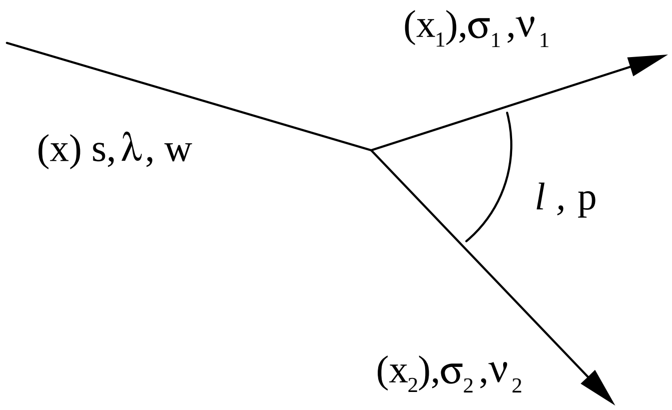

Figure 3: Decay of a child x into x_1 and x_2.

barrier functions as given by von Hippel and Quigg. [vHQ72]

Finally, the $a_{\ell s}$ is the ℓs coupling constant which contains the dynamics of the decay. This factor is absorbed into the production amplitude for this wave, which is then determined through the fit. The coupling constant $a_{\ell s}$ is in general a function of the mass W of the state X, and this is usually handled in the fits by performing them in bins of W which are narrow enough to assume the $a_{\ell s}$ constant over the width of the bin.

The decay amplitudes of the children A_{x_i}, and recursively their children, *etc.*, are calculated in the helicity frame of the child that is decaying. For instance, consider one of the x_i above, but now we call it simply x since it is the parent particle, as in Figure 3. Here the particle decaying, x has mass w, spin s, and helicity λ. Its decay products x_i have spins σ_i and helicities ν_i, with relative orbital angular momentum ℓ and breakup momentum p.

The form of the amplitude for this piece of the decay is very similar to what was written above

$$A_x = \tilde{l} \sum_{\lambda} D^{s*}_{\lambda\nu}(\Omega)(l0\sigma\nu|s\nu)(\sigma_1\nu_1\sigma_2 - \nu_2|s\nu)F_l(p)\Delta_x(w)A_{x_1}A_{x_2} \tag{7}$$

where again $\nu = \nu_1 - \nu_2$ and $\vec{\sigma} = \vec{\sigma}_1 + \vec{\sigma}_2$.

The pieces of this decay that look similar to the initial decay in the Gottfried–Jackson frame do so because they come from the same place. The same Clebsch-Gordan coefficients and tilde factor show up from the relationship of the helicity to ℓs states, the D–function from the rotational properties of the states, and the angular momentum barrier factor suppresses high-ℓ decays near threshold where the breakup momentum is small.

The only difference, in fact, is the appearance of $\Delta_x(w)$ in place of the coupling constant $a_{\ell s}$ from the initial decay. In the decays of the isobars it is usually assumed that the dynamics of the decay, which depend on the isobar mass w, are known and put in explicitly. Often this is done as a simple relativistic Breit-Wigner, although it is sometimes necessary to use a more complex parameterization, such as coupled-channel Breit-Wigners [Fla76] or a K–matrix parameterization [CK95].

Total decay amplitudes are made up as a product of these intermediate decays. In order to parameterize the spin-density matrix, which is determined by the fit, in the simplest way, it turns out to be advantageous to transform the states into the reflectivity basis. The reflectivity basis is defined by eigenstates of reflection in the production plane, the details of the transformation can be found in Chung and Trueman. [CT75]

3 Program Description

The BNL partial wave analysis programs were designed using object-oriented methods to produce a general system for doing partial wave analysis. The resulting objects were implemented using C++. Four programs comprise the basis of the system, although numerous support programs and scripts have been developed.

3.1 gamp

The program `gamp` (*g*enerate *amp*litudes) is the decay amplitude calculator. The command line useage is available by running the program with no arguments:

```
usage: gamp -wkeyfile -aampfile [-v] -n# -d[#] -Rreaction file1 file2 etc
-wwave.key: wave definition file.     default: wave.key
-aamplitude.amps: file to write amplitudes. default: amps.dat
-v verbose mode: print the particle data table
-n#: maximum number of events. default: 999999
-d[#]: debug printout
-Rreaction: reaction =
pi0pi0
etaeta
etapi0
etapi-
etapi+pi-pi-
etapi+pi-
Pim2Pi0
etaprimepi-
pi_pi-
ppi+pi-
npi+
OmegaEta
OmegaPi0
OmegaPi-
```

```
file1 file2 etc:      list of DST files
```

It requires as input a *keyfile* which contains the list of amplitudes to calculate, and a *datafile* containing events for which the amplitudes will be found. The output of the program is an *ampfile* containing for each event a list of amplitudes, the mass of the X and the momentum transfer t. The reaction contained in the *datafile* is specified with the `-R`*reaction* flag. To extend to a new reaction, some small modification to the code must be done to teach `gamp` how to read the new events and the *keyfile* entries it will need. This is briefly outlined in section 4, Internals.

The amplitude specification in the *keyfile* includes all parameters that define the wave, *i.e.*, β. The syntax is

```
{ J P C G M eps L [``S'' s] iso/bach l [``s'' s] ... } [ref] *
```

for example

```
{ 1 -1 1 -1 0 -1 1 rho0/pi   1 } *
{ 1 -1 1 -1 1 -1 1 rho0/pi   1 } *
{ 1  1 1 -1 1 -1 0 rho0/pi   1 } *
{ 2 -1 1 -1 1 -1 0 f2/pi     2 } *
```

J, P, C, G, and M are the quantum numbers of the X for which we are calculating the decay amplitude. L is the relative orbital angular momentum of its children, and the optional s, introduced by the key-letter "`S`", is the total spin of its children. Typically the bachelor is spinless, in which case this may be omitted. Next is the isobar/bachelor combination, followed by l the relative orbital angular momentum of the isobars children and the optional s, the total spin of the isobars children. The isobar/bachelor l [`s` s] combination is repeated for each subsequent decay in the chain. The optional `[ref]` statement can be either `ref_non_flip` or `ref_flip`. each group of interfering waves has an overall phase ambiguity which is removed in the parametrization of Chung and Trueman [CT75] by requiring one production amplitude in each interfering set to be real and positive. This `[ref]` statement is ignored by `gamp` and is used only later by `fit`.

Typically `gamp` is run from a script `gen_amps` which takes a *keyfile* containing several waves, and feeds the waves to `gamp` one at a time and generates a filename that encodes information about the wave. For example, some files created by `gen_amps`:

```
0-+1-0+0:f0_980,pi,0:.data.amps
0-+1-0+1:rho0,pi,1:.data.amps
1++1-0+0:rho0,pi,1:.data.amps
1++1-0+1:AMP_M,pi,0:.data.amps
1++1-0+1:CB_S2,pi,0:.data.amps
1++1-0+1:f2,pi,2:.data.amps
1++1-0+2:rho0,pi,1:.data.amps
1++1-1+0:rho0,pi,1:.data.amps
```

show `gamp` was run on the *datafile* `data.itape`. The characters before the first "`:`" are the quantum numbers of the state X, *i.e.* $JPCIGM\epsilon$. Following this, up to the second "`:`" the decay of the X is given in the format isobar/bachelor,l where l is the orbital angular momentum of the isobar decay. Had this reaction had additional isobar decays, they would be specified in an analogous way between the second and third "`:`"'s, *etc.*

The *datafile*, at present, must be in an E852 itape format containing an event summary record (ESR) group. For purposes of analyzing other data, for example LASS data described below, custom conversion programs were written to put the data into the required format.

3.2 int

The program `int` calculates the normalization integrals used in the maximum likelihood fit. It also has command line usage available:

```
usage: int -[wmialuLUn]
-wfile file = keyfile
-mfile file = datafile
-i[a|r] a = accepted (.ani); r = raw (.rni)
-a add amplitudes to integral
-l# # = lower edge of mass region
-u# # = upper edge of mass region
-s# # = size of one mass bin
-L# # = lower edge of t bin
-U# # = upper edge of t bin
-c use compressed amplitude files
-n# Print NI out every N events in bin (verbose mode only)
-v verbose output
```

It needs as input a *keyfile* and amplitudes for all waves in the *keyfile* for the Monte-Carlo *datafile*. This is a simple program that just reads amplitudes from disk and performs the sum in equation 4, writing as output a matrix of complex numbers to *nifiles*. The indices of the matrix correspond to the waves in the *keyfile*. This links a set of normalization integrals to a particular *keyfile* which must be used in the fit. To say this another way, if the set of waves used in the fit is changed, then the normalization integrals must be recalculated.

The normalization integrals are done in mass and momentum transfer bins that correspond to the bins that will be used in the fit. The values of the ranges for these values, as well as the *keyfile* that was used is encoded into the names of the *nifiles*. For example, the file names

```
Kset13_2:1.32-1.36:0.00-0.15:.ani
Kset13_2:1.32-1.36:0.00-0.15:.rni
Kset13_2:1.32-1.36:0.10-1.00:.ani
Kset13_2:1.32-1.36:0.10-1.00:.rni
Kset13_2:1.32-1.36:0.15-0.30:.ani
```

```
Kset13_2:1.32-1.36:0.15-0.30:.rni
```

indicate that the *keyfile* `Kset13_2.key` was used with the mass bin set to the range from 1.32–1.36 GeV/c^2 and three different $-t$ ranges. The `.ani` or `.rni` extensions indicate the file is a raw or accepted integral, respectively.

3.3 fit

`fit`, surprisingly, is the program which performs the maximum likelihood fit in a single bin, which is specified on the command line:

```
Usage: fit -wkeyfile -adatafile -mamcfile -rrmcfile -ooutfile
-llowmass -uuppermass -Llowt -Uuppert [-d] [-f] [-h]
-wkeyfile: set wave key file to keyfile
-adatafile: data file used to generate amplitudes
-mamcfile: accepted MC file used to generate xNI
-rrmcfile: raw MC file used to generate NI
-ooutfile: file for output
-llowmass set low edge of mass bin to lowmass
-uuppermass set upper edge of mass bin to uppermass
-Llowt set low edge of t bin to lowt
-Uuppert set upper edge of t bin to uppert
-d: debug output
-f: fit spin flip amplitudes (default non-flip only)
-h: print usage
```

The required inputs are a *keyfile*, a *datafile*, and a set of *nifiles*. These are specified on the command line by the `-w` and `-a` flags for the *keyfile* and *datafile* respectively. The program will generate the proper *nifile* names from the values of *keyfile*, and the variables indicating the mass and $-t$ bin: *lowmass*, *uppermass*, *lowt*, and *uppert*. Events falling in the bin specified are selected when *datafile* is read.

`fit` uses the CERNLIB routine MINUIT to perform the likelihood fit, and by default will enter into an interactive minuit session. Again, the program is typically run from a script with standard input connected to a file of MINUIT commands. A typical file might look something like

```
set input 1 minuit.input
set err 0.5
set str 2
migrad 999999 0.05
improve
simplex 999999 0.01
migrad 999999 0.05
call fcn 3
exit
```

The file `minuit.input` is a list of the fit parameters in MINUIT input format. This is generated from the *keyfile* using a program `wv2min`. The script that runs successive `fit` commands for a range of bins typically takes care of the initial generation of MINUIT parameters before running fit.

For each bin in which the script runs `fit`, an output file is generated containing the fit results. These *logfiles* contain all the information available about the fit results: fitted production amplitudes, normalization integrals, the *keyfile*, and the error matrix from the fit. Any observable may be calculated from this.

3.4 xvf

`xvf` [1] is a windowing post-processor program which uses the *logfiles* produced by `fit` to calculate observables. The appearance of the program is shown in Figure 4. The fit being looked at is a fit of the $\eta\pi$ final state data recorded in the 1994 run of Brookhaven experiment 852. Plotted is the number of events, acceptance corrected, that are associated with the partial waves selected in the left pane. In this example, the 2^{++} amplitudes are selected. Both are amplitudes are for D-wave decays into $\rho^0\pi^-$, one having $m = 0$ and the other having $m = 1$.

`xvf` calculates this predicted number of events using

$$N = \frac{n}{\eta_x} \sum_{\alpha\alpha'} V_\alpha V_\alpha^* \Psi_{\alpha\alpha'} \tag{8}$$

The normalization integral $\Psi_{\alpha\alpha'}$ is calculated using `int`, and $\frac{n}{\eta_x}$ is the number of observed events in a bin divided by the acceptance. The V_α are the production amplitudes found by the fit. In the sum over waves, α takes on all values selected, such as all the 2^{++} waves above.

`xvf` can also display other observables using similar formulae, such as the phase difference between two production amplitudes or the magnitude (`abs`) of the production amplitudes.

3.5 Support Programs

In addition the four major programs in the suite, there are also various support programs that are used in an analysis. `wv2min`, for example, reads a *keyfile* and produces a list of fit parameters in a format that is understood by MINUIT. This program is generally run from a script such as `do_fit`. `do_fit` is written in Perl and takes care of running `wv2min`, running `fit` in a range of bins specified, possibly using the results of a previous bin to initialize the next fit. Similar scripts exist to run `int`, and, as mentioned above, `gamp`.

`ViewAmps` is a very simple program which is used extensively for diagnostics. It simply reads an *ampfile* and writes the mass, t and amplitudes to standard output.

[1] `xvf` was written by Alexander Ostrovidov based on a much cruder program written by one of this papers authors.

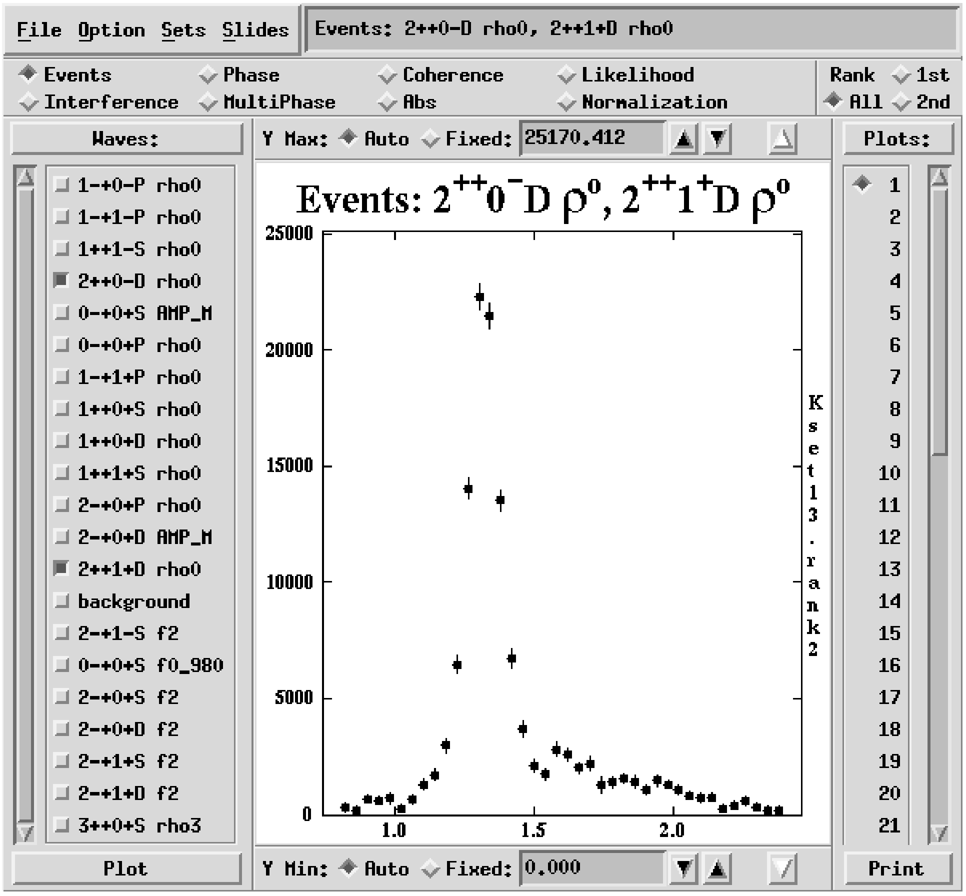

Figure 4: `xvf` window showing number of events due to 2^{++} amplitudes.

4 Internals

In this section we give a brief description of the internal workings of the programs and the design philosophy that was used. We begin with a description of the general strategy adopted to calculate the decay amplitudes in a very flexible way, allowing extension to many final states. An outline of the objects developed to accomplish this is then given.

4.1 General Strategy

The isobar model leads naturally to the idea of a recursive algorithm to calculate the total decay amplitude. Each level of the recursion calculates the contribution to the total due to individual two body decays in the sequence.

4.2 Description of Objects

4.2.1 Matrices, *etc.*

Essential to any particle physics analysis is the notion of vectors (both three and four dimensional) and matrices of an arbitrary dimension. We choose to write a `threeVector` class first, and a `fourVector` *contains* a `threeVector` (We felt this was a more natural implementation than using inheritance). All the immediately obvious operators and functions were defined for the vectors: addition, subtraction, scalar multiplication, vector multiplication, rotations and boosts (using the `Matrix` class), transformations from a cartesian to a polar representation, *etc.*

A `Matrix` class provided for arbitrary size (and shape) matrices of real numbers, and once again, most of the typical algebraic operators are supplied. In addition, a `CMatrix` class proved to be very useful for partial wave analysis, where it is often convenient to write the (complex) amplitudes using a matrix notation. It maybe is interesting to note that while this would be a textbook example of a use for templates, we wrote these classes before the introduction of templates into the C++ language. The `CMatrix` class was written (almost) completely by the UNIX commands:

```
sed 's/double/complex/' Matrix.hxx > CMatrix.hxx
sed 's/double/complex/' Matrix.cxx > CMatrix.cxx
```

4.2.2 Waves and Amplitudes

Partial waves are specified in the code using the `WaveType` class. `WaveType` has public data members for the quantum numbers of the system being studied and its decay products, which are stored as arrays of `ParticleData`:

```
class WaveType {
public:
int J;              /* Total spin */
int P;              /* Intrinsic parity */
```

```
int C;              /* C parity */
int G;              /* G parity */
int m;              /* z-component of J */
int L;              /* orbital angular momentum of X decay */
int S;              /* total spin of isobar1 and bachelor1 */
int epsilon;        /* reflectivity */
int l[6];           /* l of decay products */
int s[6];           /* total spin of decay products */
ParticleData Isobar[6];
ParticleData Bachelor[6];

                                      ⋮

};
```

4.2.3 The PWA object

The `PWA` object does the lions share of the work in amplitude calculation. It is formed from a `PWA_DST` object by constructors that must be added by the user for each new final state. This constructor takes care of setting up the tree-like decay chain of four-vectors for each event, based on the `WaveType` being calculated. Once the `PWA` object is constructed, the `DecayAmplitude()` member function returns the decay amplitude for that wave. The calculation is done by descending the decay chain, recursively calling the member function `Dfactor()` for each isobar decay.

4.2.4 The fitsummary object

The decay amplitudes returned by `PWA::DecayAmplitude()` are stored on disk and used in the maximum likelihood fit of the production amplitudes to the total intensity. The fit is done using the CERNLIB routine `MINUIT` and the results in each bin of mass and $-t$ are stored as `fitsummary` objects. This object contains all the information necessary to calculate observables from the fit results, as well as logging information about the date of fit, data files used, *etc.*.

References

[Chu71] Suh Urk Chung. Spin formalisms. CERN report 71-8, CERN, March 1971. Lectures given in the Aacademic Training Program of CERN 1969–1970.

[Chu95] Suh Urk Chung. Formulas for partial-wave analysis. Research Note BNL-QGS-93-05, Brookhaven National Laboratory, March 1995. Version II.

[CK95] Suh Urk Chung and Eberhard Klempt. A primer on K-matrix formalism. BNL Preprint Uni Mainz IP 92-03, Brookhaven National Laboratory, March 1995. Version VII.

[CT75] S. U. Chung and T. L. Trueman. Positivity conditions on the spin density matrix: a simple parametrization. *Physical Review D*, 11(3):633–646, February 1975.

[Fla76] S. M. Flatte. Coupled–channel analysis of the $\pi\eta$ and $K\bar{K}$ systems near $K\bar{K}$ threshold. *Physics Letters*, 63B:224, 1976.

[JW59] M. Jacob and G. C. Wick. On the general theory of collisions for particles with spin. *Annals of Physics*, 7:404–428, 1959.

[vHQ72] Frank von Hippel and C. Quigg. Centrifugal-barrier effects in resonance partial decay widths, shapes, and production amplitudes. *Physical Review D*, 5(3):624–638, February 1972.