BWFIT with ROOT

Zhang Xueyao
High Energy Physics Group, Shandong University

Abstract

The invariant mass spectrum analysis program *BWFIT* has been converted to C++ class and can be run in the *ROOT* environment. This article describes the mass fitting procedure in *BWFIT* and how to run it in *ROOT*.

1 Introduction

The purpose of the invariant mass spectrum analysis is to extract the masses, widths and the number of detected events for the resonances which appear in the invariant mass distribution. This can be achieved by performing a fit to the mass spectrum over a mass range of $[m_{min}, m_{max}]$ using the maximum likelihood or the least square technique. To perform this task, a program called *BWFIT* has been extensively used in the BES offline data analysis. *BWFIT* was written in FORTRAN and had been integrated into the interactive data analysis program PAW.

Recently, a new data analysis tool called *ROOT* has been developed at CERN. *ROOT* is an object-oriented framework for large scale scientific data analysis and is intended to replacing the existing analysis tools like PAW. The system, written in C++, contains an efficient hierarchical OO database, a C++ interpreter, advanced statistical analysis (including multi dimensional histogramming, fitting and minimization algorithms) and visualization tools. The user interacts with *ROOT* via a graphical interface, the command line or batch scripts. The command and scripting language is C++ and large scripts can be compiled and dynamically linked in.

To use these interesting features of *ROOT*, we have converted the FORTRAN program *BWFIT* to C++ class. By dynamically linking *BWFIT* to *ROOT*, the mass spectrum analysis can easily be done by using *ROOT* functions. In this article, the detailed description of the basic formula of the mass fitting, the structure of the *BWFIT* program and how to run the new program me within *ROOT* will be given.

2 Basic formula for mass fitting

An experimentally observable quantity often has an origin in several physical processes. The overall probability density can be thought of as a sum of terms where each term gives the contribution from one process. For the mass spectrum analysis, the invariant mass m may originates from some resonances which are the signals to be found and from some background processes. The resonance distribution can well be described by a Breit-Wigner function. The overall probability density for mass spectrum analysis can be parameterized as incoherent sum of N_{bw} Breit-Wigner probabilities plus a coherent sum of N_{amp} Breit-Wigner amplitudes with relative phases ϕ_k plus a possible background term f_b:

$$
\begin{aligned}
f(m) \quad \sim \quad & \sum_{i=1}^{N_{bw}} \alpha_i |BW_i|^2 + \\
& \beta|BW_1 + \sum_{k=2}^{N_{amp}} \sqrt{\delta_{k-1}} e^{-i\phi_{k-1}} BW_k|^2 + \\
& \gamma f_b
\end{aligned} \tag{1}
$$

where α_i, β, γ, δ_k and ϕ_k are unknown parameters to be determined by the fit. The Breit-Wigner amplitudes BW_k can be taken to be of the none-relativistic form

$$BW_k = \frac{\sqrt{\Gamma_k}}{m - m_{k0} + i\Gamma_k/2} \tag{2}$$

or of the relativistic form

$$BW_k = \frac{\sqrt{m_{k0}\Gamma_k}}{m^2 - m_{k0}^2 + i m_{k0}\Gamma_k/2} \tag{3}$$

where m_{k0} and Γ_k are the mass the width of the k-th resonance respectively. The background term can be represented as a sum of Legendre polynomials modulated by a phase space distribution f_{ps}:

$$f_b = f_{ps} \frac{1}{m_{max} - m_{min}} [1 + \sum_{i=1}^{N_p} b_i P^i(x)] \tag{4}$$

where b_i is the unknown polynomial coefficient, P^i is the i-th Legendre polynomial and x is a scaled variable defined as

$$x = -1 + 2[\frac{m - m_{min}}{m_{max} - m_{min}}] \tag{5}$$

To describe the experimentally observed distributions, the ideal probability density function $f(m)$ has to be modified by the resolution and efficiency of the detector:

$$f(m) = \int f(m')R(m, m')\eta(m')dm' \tag{6}$$

where $R(m, m')$ and $\eta(m')$ are the resolution and efficiency functions respectively. The resolution function $R(m, m')$ describes the distribution of the observable variable m for a given true value of m'. $R(m, m')$ is assumed to be Gaussian:

$$R(m, m') = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(m-m')^2}{2\sigma^2}} \tag{7}$$

where σ is the measurement uncertainty of the m. The $\eta(m')$ function has to be determined by Monte Carlo simulation.

The normalized distribution for observable mass can then be written as

$$
\begin{aligned}
f(m) \quad = \quad & \sum_{k=1}^{N_{bw}} \alpha_k \frac{\int |BW_k|^2 R(m, m')\eta(m')dm'}{NBW_k} + \\
& \beta \frac{\int |BW_1 + \sum_{k=2}^{N_{amp}} \sqrt{\delta_{k-1}} e^{-i\phi_{k-1}} BW_k|^2 R(m, m')\eta(m')dm'}{NAMP} + \\
& (1 - \sum_{k=1}^{N_{bw}} \alpha_k - \beta) \frac{\int f_b(m')R(m, m')\eta(m')dm'}{NBACK}
\end{aligned} \tag{8}
$$

where NBW_k,$NAMP$ and $NBACK$ are the normalization factors which insure that the integral of $f(m)$ over the fit mass range is correctly normalized to one. The physical meanings of the parameters α_k,β and δ_k are defined as following:

α_k: The fraction of the total number of events attributed to the k-th incoherent Breit-Wigner resonance;

β: The fraction of the total number of events attributed to the coherent sum of the Breit-Wigner amplitudes;

δ_k: The fraction of events attributed to the k-th coherent Breit-Wigner relative to the first coherent Breit-Wigner.

Since the integrations in equ. 8 are very time consuming, following simplification are made during the fit:

1. In a specific invariant mass range, the efficiency $\eta(m)$ is only a slow varying function of m and is approximated as a constant when calculating the integrations in equ. 8;

2. The first term in equ. 8 is suitable for fitting the resonances with widths close to the limits of the detector resolutions, or for fitting states which do not interfere. In this case, the non-relativistic form of Breit-Wigner is chosen. Relativistic Breit-Wigner with an energy dependent width is only required for very wide resonances near threshold. The integration in the first term of equ. 8 can then be calculated analytically using the error function of a complex argument:
$$\int |BW_k(m')|^2 R(m,m')dm' = \frac{1}{\sqrt{2\pi}\sigma_k}\mathcal{R}e(W(z)) \tag{9}$$
where
$$\begin{aligned} W(z) &= e^{-z^2} erfc(-iz) \\ z &= \frac{m-m_{k0}}{\sqrt{2}\sigma_k} + i\frac{\Gamma_k}{2\sqrt{2}\sigma_k} \end{aligned} \tag{10}$$

3. The second term of equ. 8 is appropriate for the case where there are several overlapping states with large widths. The typical mass resolution of BES detector is in the range of 10-20 MeV which is negligible for a state with a width in the order of hundred MeV. Hence the resolution function in this term can also be approximated as a δ-function;

4. Since the background $f_b(m)$ changes slowly with m, the resolution function in the third term of equ. 8 is approximated as a δ-function;

Under these assumptions, the general probability density function to be fit is as following:
$$\begin{aligned} f(m|\theta) = & \sum_{k=1}^{N_{bw}} \alpha_k \eta(m) \frac{\int |BW_k|^2 R(m,m')dm'}{NBW_k} + \\ & \beta \frac{|BW_1 + \sum_{k=2}^{N_{amp}} \sqrt{\delta_{k-1}} e^{-i\phi_{k-1}} BW_k|^2 \eta(m)}{NAMP} + \\ & (1 - \sum_{k=1}^{N_{bw}} \alpha_k - \beta) \frac{f_b(m)\eta(m)}{NBACK} \end{aligned} \tag{11}$$

where θ represents the unknown parameters which includes N_p background polynomial coefficients b_k, N_{bw} sets of incoherent Breit-Wigner probability parameters $(\alpha_k, m_{k0}, \Gamma_k, \sigma_k)$, N_{amp} sets of coherent Breit-Wigner amplitudes parameters $(\delta_k, \phi_k, m_{k0}, \Gamma_k$ and the fraction β. Given the function $f(m|\theta)$, we can perform a fit to the experimental data by using either the maximum likelihood method or the least square method to get these unknown parameters. In the following, we assume that total of N events with invariant mass $m_i(i = 1...N)$ have been observed.

Maximum likelihood method

The likelihood function for a set of N observed events is given by

$$\mathcal{L} = \prod_{i=1}^{N} f(m_i|\theta) \tag{12}$$

This function represents the probability that the given data sample is distributed as given by $f(m|\theta)$ with the given values of parameters θ. According to the maximum likelihood principle, one should vary the values of θ to maximize the probability, for the observed distribution to agree with the parent distribution.

When the number of observed events N is very large, the numerical calculation of the likelihood function 12 may be very time consuming. In this case, one may reduce the amount of computation by binning the events in a histogram and write the likelihood function as the products of a smaller number of averaged probabilities. Let the N events be divided into M bins. The joint probability to have n_i events in the $i - th$ bin is given by the multinomial distribution law:

$$\mathcal{L} = N! \prod_{i=1}^{M} \frac{1}{n_i!} P_i^{n_i} \tag{13}$$

where P_i is the probability for a event to be in bin i. This probability can be fund by integrating the mass distribution function $f(m|\theta)$ over the bin width Δm:

$$P_i = \int_{\Delta m} f(m|\theta) dm \tag{14}$$

Fitting with likelihood function 12 is called un-binned fit while that with equ. 13 is called binned fit. Since $\mathcal{L}$ and the logarithm of $\mathcal{L}$ attain their maxima for the same value of θ and the sum is often easier to handle than the product, $\log \mathcal{L}$ is used in the fitting. The likelihood function for un-binned fit is then written as:

$$\log \mathcal{L} = \sum_{i=1}^{N} \log f(m_i|\theta) \tag{15}$$

and for binned fit

$$\log \mathcal{L} = \sum_{i=1}^{M} n_i \log P_i \tag{16}$$

where terms not dependent on the fit parameters θ has been dropped.

If we assume that the total number of observed events N is a Poisson variable with mean value ν, we can construct a extended likelihood function as

$$\mathcal{L} = \frac{\nu^N e^{-\nu}}{N!} \prod_{i=1}^{N} f(m_i|\theta) \tag{17}$$

With the extended the likelihood function, equ. 15 and 16 have to be modified as

$$\log \mathcal{L} = \sum_{i=1}^{N} \log f(m_i|\theta) + N \log \nu - \nu \tag{18}$$

$$\log \mathcal{L} = \sum_{i=1}^{M} n_i \log P_i + N \log \nu - \nu \tag{19}$$

The advantage of introducing the extended likelihood function is that the total number of observed events N adds an extra constraint in the determination of θ. But this is helpful only in the cases when the expected number of events ν can be calculated with considerable accuracy.

Least square method

The binned fit can also be done using the least square method. The χ^2 is defined as

$$\chi^2 = \sum_{i=1}^{M} \frac{(n_i - NP_i)^2}{n_i} \tag{20}$$

Goodness of fit

The goodness of the fit is tested using the Pearson χ^2 which is defined as

$$\chi^2 = \sum_{i=1}^{M} \frac{(n_i^f - n_i)^2}{n_i^f} \tag{21}$$

where $n_i^f = NP_i$ is the number of events predicted by the fit for the $i-th$ bin. This quantity is distributed like a chi-square variable for n_{dof} degree of freedom:

$$n_{dof} = M - n_{par} - 1 \tag{22}$$

where n_{par} is the number of fit parameters. The (-1) term comes from the constraint:

$$\sum_{i=1}^{M} n_i^f = \sum_{i=1}^{M} n_i = N \tag{23}$$

3 The TBWfit class

When the likelihood function or the χ^2 function is known, the *TMinuit* class in *ROOT* is used to perform the fitting. For this purpose, an interface class *TBWfit* is written. This class gets the data from user supplied data files and calls the member functions of *TMinuit* to find the best estimate of the parameter values and their errors.

The *TBWfit* class contains several user callable functions which can be used to perform the analysis. The description of these functions is given below.

void SetCardFile(char *FileName)

Set the MinCard file name to FileName. This function should be called before calling any other *TBWfit* member functions.

void SetFitHistogram(TH1 *id)

For binned fit, this function sets the histogram to be fitted. The input parameter *id* is a pointer to a one dimensional histogram objects. This histogram must exist in memory or in a opened *ROOT* file.

void SetFitHistogram(char *name, char *title, int nbin,float xmin,float xmax)

This function defines the histogram for un-binned fit. It must be called before the data points for the fit to be read.

void SetFitRange(float wmin,float wmax)

This function sets the fit range. The lower and upper limits of the range is *wmin* and *wmax* respectively.

void SetFitData(float *vec,int nev)

This function sets the data points for un-binned fit. The data points are stored in a floating point array *vec*. The number of data points is *nev*.

The data points can be read either from a disk file by using the conventional C I/O function or from a NTUPLE. The member function *Draw* of TNtuple class can be used to select the data points from a NTUPLE.

Example 1: Reading data from disk file. Suppose the invariant mass of each event is stored in file "data.dat" and bwfit is a *TBWfit* object.

```
        FILE *input;
        int nev = 1000;
        float vec[1000];
        input = fopen("data.dat","r");
        for(int i = 0; i < nev; i++)
            fscanf(input,"%f",&vec[i]);
  // Call TBWfit member function to set the data
        bwfit.SetFitData(vec,nev);
        .
        .
```

Example 2: Select data points from NTUPLE. Suppose the NTUPLE is in *ROOT* file "hsimple.root" and its name is "ntuple", the name of the invariant mass in the NTUPLE is "x".

```
// Open the file
     TFile *f = new TFile("hsimple.root");
// Get the pointer to the NTUPLE
     TNtuple *nt = (TNtuple *)f->Get("ntuple");
     nt->SetEstimate(nt->GetEntries());
// Call member function Draw to selection the data
// points. selection is a string to define the selection
// criteria. The option "goff" is used to turn off the
// graphics output of Draw function
     nt->Draw("x",selection,"goff");
// Call TBWfit member function to set the data
     bwfit.SetFitData(nt->GetV1(),nt->GetSlectedRows());
     .
     .
```

void SetAxisTitles(char *xtitle, char *ytitle)

Set the titles of the x and y axis to *xtitle* and *ytitle* respectively.

void Minimize(void)

Perform the minimization by executing the user supplied *MINUIT* command.

void Result(int iflag = 1)

Print the fitted results on screen or to a log file.

void Plot(int mask=0)

Plot the fitted results. The fitting function is superimposed on the origin histogram. The *ROOT* graphics tools can then be used to manipulate the plot.

void SetUBackFunc(TF1 *uback)

Sets the user defined background function. The input parameter *uback* is a pointer to a one dimensional function object defined as:

```
TF1 *uback = new TF1("uback",function,wmin,wmax,npar);
```

where *npar* is the number of parameters in function *function*. The prototype of function *function* is defined as

```
double function(double *x,double *par)
```

The definition of the function should be supplied by user in a *ROOT* macro.
Example: Exponential background function.

```
double uExponential(double *x, double *par)
{
   return TMath::Exp(par[0] + par[1] * x[0]);
}
```

void SetUFunc(int num,TF1 *ufun)

Sets the user supplied fitting function UFUN. *num* is the user function number. The definition of *ufun* is similar as the user supplied background function.

void SetEffFunc(TF1 *eff)

Sets the user efficiency function. The definition of *eff* is similar as the user supplied background function. Note that the input parameter *par* should not be used within the efficiency function. The efficiency function is assumed only dependent on the invariant mass.

4 Perform BWfit in ROOT

The *TBWfit* class has been pre-compiled into a shared library called "libBwfit.so". To use it in a *ROOT* session, you have to first load the shared library and create a *TBWfit* object using the following two command lines:

gSystem.Load("libBwfit.so");
TBWfit bwfit;

The first line load the shared library into *ROOT* and the second line creates a *TBWfit* object. You can put these two lines in your private *rootlogon.C* file.

To perform the BWfit using *TBWfit* class, user should provide two files. one is called MinCard file which provides the fitting parameter definition, the control parameters and the *TMinuit* command lines. The other is the data file which contains the experimental data to be fitted. This data file can be a ordinary data file or a *ROOT* file which contains histograms and NTUPLE, etc.

The general format of the MinCard file is as following:

```
  1        'LEG-1'     start step low high
  2        'LEG-2'       "     "    "   "
  3        'LEG-3'       "     "    "   "
  .
  .
np         'LEG-np'      "     "    "   "
np+1       'ALPHA1'      "     "    "   "
  +2       'SIGMA1'      "     "    "   "
  +3       'GAMMA1'      "     "    "   "
  +4       'MEAN1'       "     "    "   "
  +5       'ALPHA2'      "     "    "   "
```

```
              +6           'SIGMA2'       "    "    "    "
              +7           'GAMMA2'       "    "    "    "
              +8           'MEAN2'        "    "    "    "
               .
               .
      np+4*nbw-3           'ALPHA-nbw'    "    "    "    "
      np+4*nbw-2           'SIGMA-nbw'    "    "    "    "
      np+4*nbw-1           'GAMMA-nbw'    "    "    "    "
      np+4*nbw             'MEAN-nbw'     "    "    "    "
              +1           'BETA'         "    "    "    "
              +2           'WIDTH-1'      "    "    "    "
              +3           'MEAN-1'       "    "    "    "
              +4           'WIDTH-2'      "    "    "    "
              +5           'MEAN-2'       "    "    "    "
              +6           'DELTA-1'      "    "    "    "
              +7           'PHASE-1'      "    "    "    "
              +8           'WIDTH-3'      "    "    "    "
              +9           'MEAN-3'       "    "    "    "
             +10           'DELTA-2'      "    "    "    "
             +11           'PHASE-2'      "    "    "    "
               .
               .
np+4*nbw+4*namp-1          'PHASE-namp-1' "    "    "    "
np+4*nbw+4*namp            'Uwait-1'      "    "    "    "
              +1           'Uparl-1'      "    "    "    "
               .
               .
              +npar1       'Uparl-npar1'  "    "    "    "
              +npar1+1     'Uwait-2'      "    "    "    "
                    +2     'Upar2-1'      "    "    "    "
                     .
                     .
                    +npar2 'Upar2-npar2'  "    "    "    "
               .
               .


FITYPE [fitype]
MFIT   [mfit]
MPOLY  [npoly]
MBW    [nbw]
MAMP   [namp]
MPHASP [xtot mass1 mass2 mass3]
MEFFLG [mefflg]
UFUN   [mufun  npar1 npar2 npar3 ...]
```

```
SET ERR 0.5
MINIMIZE
END RETURN
```

The MinCard file contains three blocks(separated by a blank line): the definitions of the parameters to be fitted, the *TBWfit* run time parameters and the command lines which control the *TMinuit* execution.

TBWfit runtime parameters

The *TBWfit* runtime parameters are used to select the fit method and fit function form. In the MinCard file, each of these parameters are defined by a key word followed by one or more numerical parameters.

The meaning of the runtime parameters are described bellow:

1. **FITYPE** [*fitype*]
   Parameter *fitype* is used to select the fit method:
   fitype = 0: use maximum likelihood method;
   fitype = 1: use extended maximum likelihood method.
   fitype = 2: use least square method;

2. **MFIT** [*mfit*]
   Parameter *mfit* specifies the fit type. *mfit*=1 means we are doing an un-binned fit. *mfit*$\geq$ 2 doing binned fit. When doing a binned fit, you have the choice of either performing an integration of the fit function over each bin for the likelihood calculation, or merely evaluating the function at the center of the bin(*mfit*=5). For this integration, the user now has three options:

   - Use Simpsons rule (*mfit*=2). This technique is relatively fast, but may not give the best estimates for the fitted parameters if the fit function varies rapidly over a bin.
   - Use an adaptive quadrature algorithm(*mfit*=3). This technique is slower, but will result in a better estimate for the fit parameters. In fact, this technique will give almost the same results as an un-binned fit.
   - Use Trapozoiodal rule(*mfit*=4)

3. **MPOLY** [*npoly*]
   Parameter *npoly* is the highest order of Legendre polynomial in the background function (the order of constant Legendre polynomial is 0). There are a total of *npoly* Legendre polynomials in the background function. The number of fitted parameters appearing in the parameter definition block is *np = npoly - 1* because the first older Legendre polynomial is a constant and is determined by the overall normalization factor. To invoke a user supplied background function, set *npoly*=-(number of user parameters+1).

4. **MBW** [*nbw*]
Sets the number of incoherent Breit-Wigners in the fit function to *nbw*.

5. **MAMP** [*namp*]
Sets the number of coherent Breit-Wigners in the fit function to *namp*.

6. **MPHASP** [*xtot mass1 mass2 mass3*]
Defines the phase space parameters of the three body decay of particle X.

$$X \rightarrow 1 + 2 + 3 \tag{24}$$

xtot,mass1,mass2 and *mass3* are the masses of the particle X and the three daughter particles respectively. If this card is present in the MinCard file, the phase space distribution of the 2-3 system modulated by the background polynomial is taken as the shape of the background function.

7. **MEFFLG** [*mefflg*]
Parameter *mefflg* controls the efficiency correction in the fit. If *mefflg*= 1, efficiency corrections is done in the fit, yielding efficiency corrected resonance parameters. In this case, the user must supply a function which returns an efficiency in the region of [0,1].

8. **UFUN** [*mufun npar1 npar2 npar3 ...*]
User-supplied function can be fitted. To enable the call to the user function, the UFUN control card must be exist in the MinCard file. *mufun* is the number of user-supplied functions to be fitted. Total of 20 user functions can be selected. *npar1* is the number of fit parameters in the first user function, *npar2* the second user function ... etc. The number npari must include the weight parameter Uwait-i.

Fitting Parameter Definition

The parameter definitions follow the *MINUIT* convention. For each parameter one line gives the parameter number, the parameter name, the starting value of the parameter(start),the starting step size(step) , or expected uncertainty of this parameter, the lower(low) and upper(high) bound of the parameter. The parameter name must be given between single quotes. The last three items are optional and can be left blank. If the step size is zero or blank, the parameter is a constant.

The parameter definition block contains in general five parts:

1. Parameters in background function: the first *np* lines;

2. Parameters for incoherent Breit-Wigners: lines from *np*+1 to *np*+*4***nbw*, where *nbw* is the number of Breit-Wigners in the fit. Each Breit-Wigner has four parameters.

3. Parameters for coherent Breit-Wigners: lines from $np + 4{*}nbw + 1$ to $np + 4{*}nbw + 4{*}namp\text{-}1$,where *namp* is the number of coherent Breit-Wigners.

4. The last part is the parameters for user functions.

For extended maximum likelihood fit, the last parameter in the definition must be mean ν of the Possion distribution.

Special case of the fit parameter definition

1. If only a constant background is to be fitted, set *npoly*=1, and no background parameter cards appear in the list (the size of the constant background term is determined by the overall normalization and the other normalization parameters).

2. If no background is to be included, set *npoly*=0 and leave off the ALPHA-1 Breit-Wigner parameter (parameter 1 is SIGMA-1,etc.).

3. If *npoly*=0, *nbw*=0, and no user function,leave off the BETA parameter in the Breit-Wigner amplitude block (parameter 1 is WIDTH-1). In this case the fit function consists only of a coherent sum of Breit-Wigner amplitudes whose overall normalization is fixed by the number of events in the fit.

4. If *npoly+nbw+namp*=0, then Uwait-1 must be left off.

TMinuit command line

These cards define the command lines used by *TMinuit* to perform the minimization. See *MINUIT* manual for valid commands.

If no command lines defined in the MinCard file, the default action will be taken.

5 Examples

Suppose we want to fit the $k^{*0} \to k\pi$ mass distribution with one incoherent Breit-Wigner convoluted with a gaussian plus a polynomial background using maximum likelihood method.

The MinCard file(assume its name is test.mincard) for un-binned fit is as the following.

```
1 'poly_1'   0.       1e-3
2 'poly_2'   0.       1e-3
3 'ratio'    0.2      1e-3
4 'sigma'    0.01     1e-3
5 'width'    0.
6 'mass'     0.892    1e-3

FITYPE 0
MFIT   1
MPOLY  3
MBW    1

SET ERROR 0.5
SET PRINT 0
MINIMIZE
END RETURN
```

Following macro can be used to perform the fit when data are read from a Ntuple:

```
{
   TFile *hfile = new TFile("hsimple.root");
   nx->Draw("x","1","goff");
   bwfit.SetCardFile("test");
   bwfit.SetFitHistogram("test","test",100,0.0,3.0);
   bwfit.SetFitData(nx->GetV1(),nx->GetSelectedRows());
   bwfit.SetAxisTitles("","");
   bwfit.Minimize();
   bwfit.Plot();
}
```