Running $b_1\pi$ Amplitude Analysis with AmpTools

for a JLab PWA meeting

Igor Senderovich[1]

Department of Physics

ARIZONA STATE UNIVERSITY

Department of Physics

December 13, 2012

[1] Igor.Senderovich@asu.edu

Outline

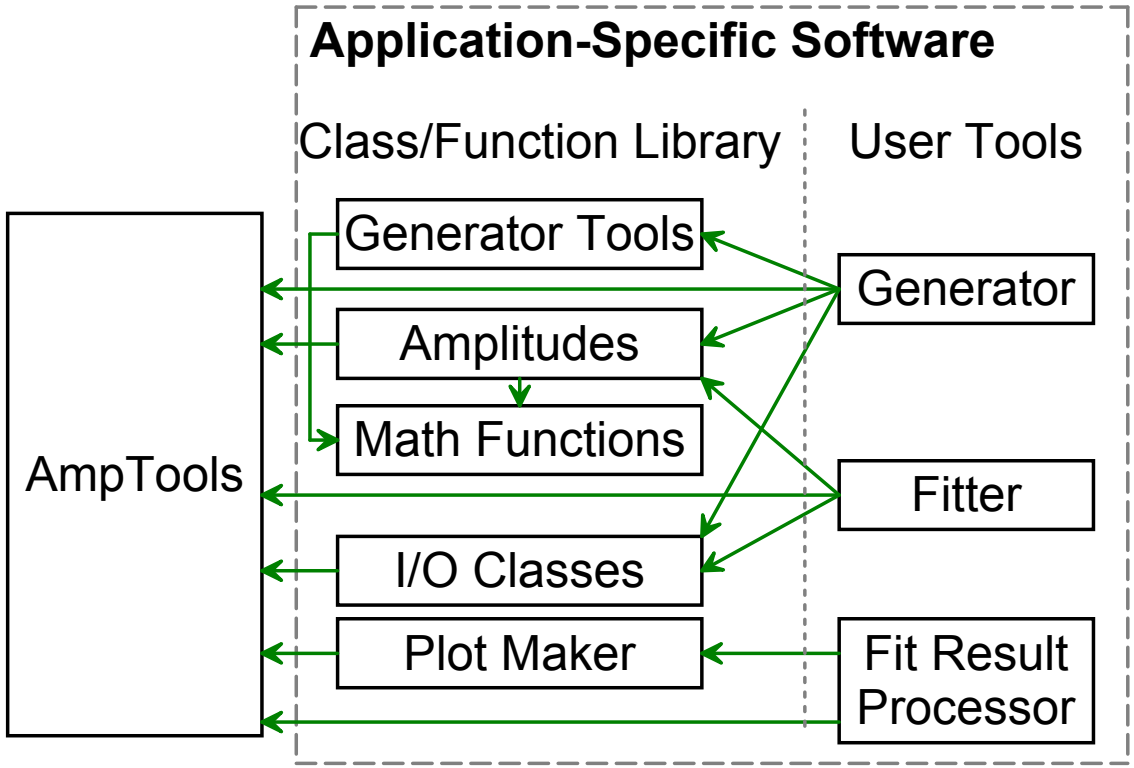
Disclaimer: work was done on AmpTools v0.3 - some details shown here may be outdated. But this is just the overview of implementing a PWA with AmpTools. For more exact usage, refer to AmpTools tutorials.

Building on AmpTools: an example

AmpTools:

- a flexible set of libraries to support event generating and fitting tools
- minimal by the design to free the physicist

GlueX example for building a framework on AmpTools:

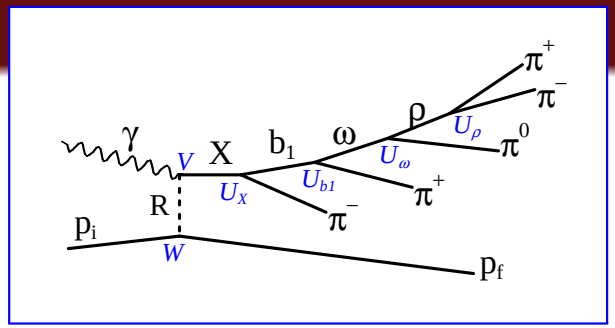


The middle column may evolve into a useful general library companion to AmpTools

Amplitude

Consider the transition:

$\langle \mathbf{p_f}, \epsilon_f; \mathbf{q}_\pi; \mathbf{q}_\rho \lambda_\rho; \mathbf{q}_\omega \lambda_\omega; \mathbf{q}_{b_1} \lambda_{b_1} | S | \epsilon_\gamma; \mathbf{p_i}, \epsilon_i \rangle$



$\langle \mathbf{q}_\pi; \mathbf{q}_\rho \lambda_\rho; \mathbf{q}_\omega \lambda_\omega; \mathbf{q}_{b1} \lambda_{b1} \| U \| J_X M_X \epsilon_X \rangle$	decay amplitude
$\langle J_X M_X \epsilon_X \| V \| \epsilon_\gamma; \lambda_R \epsilon_R; \Omega_0 \rangle$	production: γX vertex parametrized: $v^{X, M_X, \epsilon_X}_{\lambda_R, \epsilon_R}(s,t)$
$\langle \lambda_R \epsilon_R; \Omega_0; \mathbf{p_f}, \epsilon_f \| W \| \mathbf{p_i}, \epsilon_i \rangle$	production: $p_i p_f$ vertex drops out due to unpolarized target

The decay amplitude may be further broken down by decay stages:

$\langle \Omega_{b_1} \lambda_{b_1} 0 | U_X | J_X M_X \rangle \langle \Omega_\omega \lambda_\omega 0 | U_{b_1} | 1, M_{b_1} \rangle \langle \Omega_\rho \lambda_\rho 0 | U_\omega | 1, M_\omega \rangle \langle \Omega_\pi 00 | U_\rho | J_\rho, M_\rho \rangle$

$\langle \Omega \lambda_1 \lambda_2 | U | JM \rangle =$

$$\sum_{L,S} \langle \Omega \lambda_1 \lambda_2 | JM \lambda_1 \lambda_2 \rangle \langle JM \lambda_1 \lambda_2 | JMLS \rangle \langle JMLS | U | JM \rangle =$$

$$\sum_{L,S} \left[\sqrt{\frac{2J+1}{4\pi}} D^{J*}_{M\lambda}(\Omega) \right] \left[\sqrt{\frac{2L+1}{2J+1}} \begin{pmatrix} L & S \\ 0 & \lambda \end{pmatrix} \middle| \begin{matrix} J \\ \lambda \end{matrix} \right) \left(\begin{matrix} S_1 & S_2 \\ \lambda_1 & -\lambda_2 \end{matrix} \middle| \begin{matrix} S \\ \lambda \end{matrix} \right) \right] a^J_{LS}$$

Amplitude (continued)

Putting these components together...
Basic sums of amplitudes and their weights[2]: $T_{(f)(i)} =$

$$\overbrace{\sum_{\substack{X(J_X,I_X)\\L_X}}}^{\text{summed by the framework}} \overbrace{A^{J^{PC}}_{L_X}}^{\text{unknown coeff. set/fitted to via framework}} \overbrace{BW^{(X)}_{L_X}(m_X)}^{\text{omitted in mass-indep. fit}} \sum_{\substack{\lambda_{b_1},\lambda_\omega,\lambda_\rho\\M_X,\epsilon_X\\\lambda_R\epsilon_R}} \ldots \overbrace{v^{(X)}_{\epsilon_R}(s,t)}^{\text{reflectivity coupling}} \times$$

$$\sum_{L_{b_1}} \ldots BW^{(b_1)}_{L_{b_1}} \underbrace{u^{(b_1)}_{L_{b_1}}}_{\substack{\text{D/S ratio}\\\text{from PDG}}} \sum_{L_\omega} \ldots BW^{(\omega)}_{L_\omega} \underbrace{u^{(\omega)}_{L_\omega}}_{\substack{\text{one value:}\\\text{P-wave}\\\text{dominant}}} \sum_{L_\rho} \ldots BW^{(\rho)}_{L_\rho} \underbrace{u^{(\rho)}_{L_\rho}}_{\substack{\text{locked to } \omega\\\therefore \text{ P-wave}}}$$

■ - handled through AmpTools configuration files

[2] Wigner D-functions, Clebsch-Gordan coefficients and normalization terms omitted.

Input

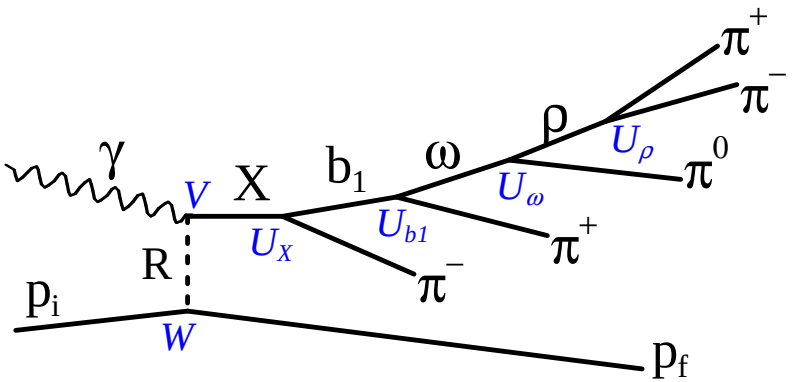
Generated signal put in with the following parameters:

- X resonance: two interfering waves:

wave (J^{PC})	L	S	m_0 (GeV)	Γ_0 (GeV)
1^{--}	0	1	1.89	0.16
2^{+-}	1	1	2.00	0.25

- $b_1(1^{+-})$: system allows $L_{b_1} = 0, 2$ with D/S amp. ratio: 0.28
- $\omega(1^{--})$: known dominant wave - $L_\omega = 1$
- "ρ": locked to $\omega \rightarrow L_\rho = 1$

Figure $b_1\pi$ photo-production and decay. ω is modeled as a sequence of two-body decays: pion and di-pion system (not physical ρ)



Expressing Beam Polarization

$$T_{\epsilon_\gamma} = \sqrt{\frac{-\epsilon_\gamma}{2}}\left(T_{-1}e^{-i\alpha} - \epsilon_\gamma T_{+1}e^{i\alpha}\right) \quad \text{where,} \quad T_{\pm 1} \propto (1 \ \pm 1)_{\text{lab}}$$

The average over the initial photon polarization states results in a cross section evaluated as follows:

$$\frac{d^8\sigma}{d\Omega_{b1}\, d\Omega_\omega\, d\Omega_\rho\, d\Omega_\pi} \propto$$

$$\frac{1+g}{2}\left|\frac{1}{\sqrt{2}}\left(T_{-1}e^{-i\alpha} + T_{+1}e^{i\alpha}\right)\right|^2 + \frac{1-g}{2}\left|\frac{i}{\sqrt{2}}\left(T_{-1}e^{-i\alpha} - T_{+1}e^{i\alpha}\right)\right|^2$$

where $g \in [0, 1]$ is the polarization fraction ($1 \rightarrow 100\%$ x-polarized, $0 \rightarrow$ unpolarized)

The coded amplitude function must therefore receive two arguments relevant to polarization

- ϵ_γ - component being calculated
- g - polarization fraction $\Leftarrow$ can be factored out to avoid amplitude recalculation when this is a free parameter

Thus, create:

```
calcAmplitude::polCoef : public Amplitude (g)
calcAmplitude::b1piAngAmp : public Amplitude (epsilon_gamma, g)
```

Summing Polarization Components

The η-th term in the incoherent sum - $I_\eta(\Omega) =$

Currently

$$\sum_{\alpha,\beta}^{n} u_\alpha^\eta u_\beta^{\eta *} A_\alpha^\eta(\Omega) A_\beta^{\eta *}(\Omega)$$

Take advantage of $\sum_\eta$:

- specify beam polarization component and fraction for each η value
- constrain u_α to be independent of η

Should become

$$\left[\sum_{\gamma,\delta} \rho_{\gamma\delta} \sum_{\alpha,\beta}^{n} u_\alpha u_\beta^* A_\alpha^\gamma(\Omega) A_\beta^{\delta *}(\Omega) \right]_\eta$$

- additional incoherent sum (γ, δ)
- new set of parameters (ρ elements)
- may be the natural implementation of moments expansion

Disadvantage: can only represent diagonal terms of beam spin-density matrix
- sufficient for purely-linear, partially-polarized beams

Advantage: can use arbitrary beam spin-density matrix
- good for general (elliptical) polarization

Overview of a Generator

Basic steps in a generator:

1. throw final state particles isotropically
2. compute the weight of each event based on squared amplitude (or just phase space probability)
3. reject events based on probability given by the weight
4. repeat until desired sample size is achieved

Efficiency may be very poor when parent probability distribution deviates from phase space significantly (especially so for narrow resonances
Improvement: importance sampling – throw events with a bias toward known resonances with the bias incorporated in the event weight

With AmpTools, such program is a fairly simple wrapper, instantiating AmpTools configuration parser, amplitude managing classes, data writers etc.

Generator Configuration File: Basic Setup

Begin configuration file with reaction-related definitions:

```
# masses and widths of overlapping resonances
define X1  1.89 0.16
define X2  2.00 0.25

# particle index: 0 1  2    3     4   5   6
reaction b1pi gamma p Pi-  Pi+   Pi0 Pi- Pi+
#                    { X  [ b1  (  omega    )]}
define IsoZ_b1pi +1 -1
sum b1pi xpol
sum b1pi ypol

# beam polarization:
define polFrac 0.4
define xpolInd 0
define ypolInd 1

#b1piAngAmp arguments: polBeam (0[1] = x[y] pol.)
#J_X, Parity_X, L_X, Isospin_X, epsilon_R (exch. natur.) Iz_b1 Iz_pi
# 1^{--} (L=0, S=1)
define X1_LX 0
amplitude b1pi::xpol::X1_b1pi_S BreitWigner X1  X1_LX  2 3456
amplitude b1pi::xpol::X1_b1pi_S polCoef    xpolInd polFrac
amplitude b1pi::xpol::X1_b1pi_S b1piAngAmp xpolInd  1 -1 X1_LX 0 1 IsoZ_b1pi
# permutations for the other isospin combination: b1- pi+
permute   b1pi::xpol::X1_b1pi_S 0 1  3 2 4 5 6
...
```

Generator Configuration File: Amplitude Definition 1

We continue with the amplitude definition

```
#b1piAngAmp arguments: polBeam (0[1] = x[y] pol.)
#J_X, Parity_X, L_X, Isospin_X, epsilon_R (exch. natur.) Iz_b1 Iz_pi

# 1^{--} (L=0, S=1)
define X1_LX 0

# x component ------
amplitude b1pi::xpol::X1_b1pi_S BreitWigner X1  X1_LX  2 3456
amplitude b1pi::xpol::X1_b1pi_S polCoef    xpolInd polFrac
amplitude b1pi::xpol::X1_b1pi_S b1piAngAmp xpolInd  1 -1 X1_LX 0 1 IsoZ_b1pi
# permutations for the other isospin combination: b1- pi+
permute   b1pi::xpol::X1_b1pi_S 0 1  3 2 4 5 6
permute   b1pi::xpol::X1_b1pi_S 0 1  6 2 4 5 3
permute   b1pi::xpol::X1_b1pi_S 0 1  3 5 4 2 6
permute   b1pi::xpol::X1_b1pi_S 0 1  6 5 4 2 3

# y component ------
amplitude b1pi::ypol::X1_b1pi_S BreitWigner X1  X1_LX  2 3456
amplitude b1pi::ypol::X1_b1pi_S polCoef    ypolInd polFrac
amplitude b1pi::ypol::X1_b1pi_S b1piAngAmp ypolInd  1 -1 X1_LX 0 1 IsoZ_b1pi
# permutations for the other isospin combination: b1- pi+
permute   b1pi::ypol::X1_b1pi_S 0 1  3 2 4 5 6
permute   b1pi::ypol::X1_b1pi_S 0 1  6 2 4 5 3
permute   b1pi::ypol::X1_b1pi_S 0 1  3 5 4 2 6
permute   b1pi::ypol::X1_b1pi_S 0 1  6 5 4 2 3
```

Generator Configuration File: Amplitude Definition 2

Amplitude definition continued: second (exotic) resonance

```
#b1piAngAmp arguments: polBeam (0[1] = x[y] pol.)
#J_X, Parity_X, L_X, Isospin_X, epsilon_R (exch. natur.) Iz_b1 Iz_pi

# 2^{+-} (L=1, S=1)
define X2_LX 1

# x component ------
amplitude b1pi::xpol::X2_b1pi_P BreitWigner X2  X2_LX  2 3456
amplitude b1pi::xpol::X2_b1pi_P polCoef    xpolInd polFrac
amplitude b1pi::xpol::X2_b1pi_P b1piAngAmp xpolInd  2 +1 X2_LX 0 1 IsoZ_b1pi
# permutations for the other isospin combination: b1- pi+
permute   b1pi::xpol::X1_b1pi_P 0 1  3 2 4 5 6
permute   b1pi::xpol::X1_b1pi_P 0 1  6 2 4 5 3
permute   b1pi::xpol::X1_b1pi_P 0 1  3 5 4 2 6
permute   b1pi::xpol::X1_b1pi_P 0 1  6 5 4 2 3

# y component ------
amplitude b1pi::ypol::X2_b1pi_P BreitWigner X2  X2_LX  2 3456
amplitude b1pi::ypol::X2_b1pi_P polCoef    ypolInd polFrac
amplitude b1pi::ypol::X2_b1pi_P b1piAngAmp ypolInd  2 +1 X2_LX 0 1 IsoZ_b1pi
# permutations for the other isospin combination: b1- pi+
permute   b1pi::ypol::X1_b1pi_P 0 1  3 2 4 5 6
permute   b1pi::ypol::X1_b1pi_P 0 1  6 2 4 5 3
permute   b1pi::ypol::X1_b1pi_P 0 1  3 5 4 2 6
permute   b1pi::ypol::X1_b1pi_P 0 1  6 5 4 2 3
```

(Technical) Overview of Amplitude Fitting

Inputs to a fit:

1. real or simulated data
2. phase space-distributed Monte Carlo:
   - original, full sample
   - detector-accepted (reconstructed) version

Fit outputs:

1. fit parameters, error matrix etc.
2. normalization integrals over full and accepted MC

One is often interested in mass-independent fitting
$\Rightarrow$ need some scaffolding for splitting data into bins and iterating with fitter and plot maker

Fitter Configuration File: Basic Setup

Begin configuration file with reaction-related definitions:

```
fit b1pi
# files with event samples  of MC (phase space), accepted MC and data
genmcfile b1pi b1pi_gen_0.root
accmcfile b1pi b1pi_acc-hdg_stdcut_0.root
datafile  b1pi b1pi_data-hdg_stdcut_0.root
# normalization integral output file:
normintfile b1pi bin_0.ni

# particle index: 0 1  2    3     4   5   6
reaction b1pi gamma p Pi-  Pi+   Pi0 Pi- Pi+
#                    { X  [ b1  (  omega    )]}
define IsoZ_b1pi +1 -1
sum b1pi xpol
sum b1pi ypol
sum b1pi BG

# beam polarization:
define polFrac 0.4
define xpolInd 0
define ypolInd 1
```

```
# Or, to allow polFrac to float:
parameter polFrac 0.35 bounded 0.0 1.0
# and replace everywhere:
#   [polFrac] -> polFrac
```

(to be continued...)

Generator Configuration File: Amplitude Definition 1

We continue with the amplitude definition

```
#b1piAngAmp arguments: polBeam (0[1] = x[y] pol.)
#J_X, Parity_X, L_X, Isospin_X, epsilon_R (exch. natur.) Iz_b1 Iz_pi

# 1^{--} (L=0, S=1)
define X1_LX 0
# x component ------
amplitude b1pi::xpol::X1_b1pi_S polCoef    xpolInd polFrac
amplitude b1pi::xpol::X1_b1pi_S b1piAngAmp xpolInd  1 -1 X1_LX 0 1 IsoZ_b1pi
# permutations for the other isospin combination: b1- pi+
permute   b1pi::xpol::X1_b1pi_S 0 1  3 2 4 5 6
permute   b1pi::xpol::X1_b1pi_S 0 1  6 2 4 5 3
permute   b1pi::xpol::X1_b1pi_S 0 1  3 5 4 2 6
permute   b1pi::xpol::X1_b1pi_S 0 1  6 5 4 2 3

# y component ------
amplitude b1pi::ypol::X1_b1pi_S polCoef    ypolInd polFrac
amplitude b1pi::ypol::X1_b1pi_S b1piAngAmp ypolInd  1 -1 X1_LX 0 1 IsoZ_b1pi
# permutations for the other isospin combination: b1- pi+
permute   b1pi::ypol::X1_b1pi_S 0 1  3 2 4 5 6
permute   b1pi::ypol::X1_b1pi_S 0 1  6 2 4 5 3
permute   b1pi::ypol::X1_b1pi_S 0 1  3 5 4 2 6
permute   b1pi::ypol::X1_b1pi_S 0 1  6 5 4 2 3

constrain  b1pi::xpol::X1_b1pi_S b1pi::ypol::X1_b1pi_S
```

Many other such waves definitions follow...

Fitter Configuration File: Initialize Parameters

Skipping to end of configuration file: adding an isotropic background wave and setting initial guess of amplitude coefficients

```
# Isotropic Background
amplitude b1pi::BG::BG Uniform

# Parameter initialization
initialize b1pi::xpol::X1_b1pi_S cartesian 2.29734e+06 0 real
initialize b1pi::xpol::X2_b1pi_P cartesian 1.74782e+06 -189193
initialize b1pi::xpol::X6_b1pi   cartesian -5076.55 458.246
initialize b1pi::xpol::X3_b1pi   cartesian -2790.48 6966.86
initialize b1pi::xpol::X5_b1pi   cartesian 1072.81 -1771.6
initialize b1pi::xpol::X4_b1pi   cartesian -1455.29 -596.001
initialize b1pi::xpol::X8_b1pi   cartesian 7703.04 -3554.39
initialize b1pi::BG::BG cartesian .005 0 real
```

May be convenient to insert instead:

```
include param_init.cfg
```

to append initial guess *e.g.* from previous mass bin's fit

Permutations and Isospin

Permutations are taken into account by AmpTools internally:

- identically-named particles in `reaction` permuted automatically
- further permutation (from known indistinguishable diagrams) imposed by `permute` command

This is handled in the ampltidue code like so:
In class constructor:

```
// config's IsoZ_b1pi -> Iz1, Iz2
mIz[2]=Iz2;
mIz[3]=Iz1;
mIz[4]= 0
mIz[5]=-1;
mIz[6]=+1;
```

When calculating amplitude:

```
const vector< int >& perm = getCurrentPermutation();
int Iz_b1 = mIz[perm[2]];
int Iz_pi = mIz[perm[3]];
clebschGordan(1, 1, Iz_b1, Iz_pi, mI_X, Iz_b1 + Iz_pi);
```

Amplitude Fit Results: 40 nb 2^{+-} & Pythia (GlueX)

Example of results for a PWA study:

- intensities for each wave (if it existed in isolation) trivial to extract through `PlotGenerator`
- plotting true values in MC study takes some work
  1. load original configuration file
  2. instantiate `NormIntInterface`, `AmplitudeManager` etc.
  3. process the MC phase space files with generator configuration which outputs integrals
  4. instantiate `PlotGenerator` on the new "truth" files to get the bands of expected values