Using AmpTools to fit GlueX Data

Example Fit Goal: Extract a_2 from $\eta\pi^0$

- Use data sample from previous exercise
- The kinematics of $\eta\pi^0$ system (angular distributions) are given by the Z_ℓ^m functions discussed in the morning session
- Dynamical assumptions about how the amplitude depends on $M(\eta\pi^0)$
  - D-waves: a complex-valued Breit-Wigner with parameters consistent with the $a_2(1320)$
    - certain m-projections/reflectivities excluded based on tensor meson dominance (TMD) model
  - S-waves: fixed complex number in four coarse bins of $M(\eta\pi^0)$ (a piecewise complex function)
- Goal: extract the total efficiency corrected yield of $a_2(1320)$ (the coherent sum of all D-waves)

show plot

Required Code: AmpTools

- AmpTools: library that is distributed independently of GlueX code base
  - available through GitHub: github.com/mashephe/AmpTools
  - installed as external package in GlueX environment
  - (include version.xml lines here)
- The AmpTools library (not an executable) that provides a general interface for doing fits
  - fully functional example of how to use AmpTools is also provided in Tutorials/Dalitz
    - copyDalitz.py is provided to generate your own project based on the Dalitz tutorial (not needed for a typical GlueX user)
  - GlueX has a set of libraries and executables that rely on the core AmpTools package
    - halld_sim/src/libraries/AMPTOOLS_...
    - halld_sim/src/programs/AmplitudeAnalysis
  - Documentation concerning the theory of operation is available in the Git repository
  - AmpTools "knows" nothing about GlueX data format, physics, etc.
- Issue tracking system on GitHub is used -- report AmpTools problems there if they pertain to the core package and not the GlueX implementation

Required Code: Amplitudes

- Provide AmpTools with a method (code via a library) to convert the four-vectors of an event to a complex number
  - inherits from the Amplitude class in AmpTools which defines the interface for the object -- a template is provided to handle some necessary functions
    - see Tutorials/Dalitz/DalitzLib/DalitzAmp/BreitWigner.h
  - accepts arguments as an arbitrarily long list of strings (which will be specified in the config file)
- A collection of GlueX related amplitudes is here:
  - halld_sim/src/libraries/AMPTOOLS_AMPS
- Several optional features:
  - embed floating fit parameters (e.g., BreitWigner mass) into the calculation of the amplitude
  - perform a data reduction step to reduce four-vectors to "user variables," e.g., angles or Lorentz invariants, that are used to compute the amplitude
  - GPU acceleration of amplitude calculation -- requires additional code
  - *use of features increases complexity but can optimize performance -- not a one-size-fits-all solution: ask for advice if you considering additional development to make fits run faster*
- The executable you write knows about the existence of the amplitudes through the static registration methods of the AmpToolsInterface
  - AmpToolsInterface::registerAmplitude(BreitWigner());
  - register before creating an instance of AmpToolsInterface to do your fit, generate MC, ...

Required Code: DataReader

- Provide AmpTools with a class that is able to turn a file on disk into a set of four-vectors
  - similar to Amplitude class: inherits from DataReader, uses a template for some key functions, and accepts arguments as a list of strings
  - see example in Tutorials/Dalitz
- Not usually analysis specific, but more specific to the file format
  - GlueX collection of data readers is here: halld_sim/src/libraries/AMPTOOLS_DATAIO
- Common GlueX formats for AmpTools input:
  - standard ROOT tree from ??? : ROOTDataReader
  - FSRoot format tree: FSRootDataReader
- Complex functionality can be added:
  - perform filtering or cuts during read into AmpTools
  - bootstrap: random sample with oversampling to evaluate uncertainties
- Not all components of a fit need to use the same data reader
- Like amplitudes, readers need to be registered prior to use:
  - AmpToolsInterface::registerDataReader(DalitzDataReader());

Configuration File: General Remarks

- See sample: Tutorials/Dalitz/run/dalitz3.cfg
- all lines begin with a keyword that informs the parser how to process the rest of the line
  - no continuation character: put it all on one line
  - ordering of the lines is not important
- useful keywords for organizing files:
  - include <file>
  - define <word> (defn1) (defn2) (defn3) ...
    - <word> must be isolated (spaces on each side) to be replaced
  - loop <word> <value1> (value2) (value3) ...
    - any line containing <word> will be repeated replacing <word> with <value1>, (value2), ...
    - multiple loops can be in a single line but they must be of the same length N -- then the line is repeated N times stepping through all loops in sync simultaneously
- some special syntax:
  - # as the first character denotes a comment
  - :: is treated like a space
  - [parname] -- use square brackets when the name of a parameter (instead of a numerical value) should be passed as an argument to an amplitude

Reactions, sums, and amplitudes

- Within a reaction, the intensity must be defined as a sum of coherent sums of amplitudes, where each amplitude can be a product of factc

$$\mathcal{I}(\mathbf{x}) = \sum_{\sigma} \left| \sum_{\alpha} s_{\sigma,\alpha} V_{\sigma,\alpha} A_{\sigma,\alpha}(\mathbf{x}) \right|^2$$

$$A_{\sigma,\alpha}(\mathbf{x}) = \prod_{\gamma=1}^{n_{\sigma,\alpha}} a_{\sigma,\alpha,\gamma}(\mathbf{x})$$

- Amplitudes can be scaled by a real number $s_{\sigma,\alpha}$ and have a complex production coefficient $V_{\sigma,\alpha}$
- This matches the general form for $\eta\pi$ production:

$$I(\Omega,\Phi) = 2\kappa \sum_k \Bigg\{ (1-P_\gamma) \left| \sum_{\ell,m} [\ell]^{(-)}_{m;k} \mathrm{Re}[Z^m_\ell(\Omega,\Phi)] \right|^2 + (1-P_\gamma) \left| \sum_{\ell,m} [\ell]^{(+)}_{m;k} \mathrm{Im}[Z^m_\ell(\Omega,\Phi)] \right|^2 + (1+P_\gamma) \left| \sum_{\ell,m} [\ell]^{(+)}_{m;k} \mathrm{Re}[Z^m_\ell(\Omega,\Phi)] \right|^2 + (1+P_\gamma) \left| \sum_{\ell,m} [\ell]^{(-)}_{m;k} \mathrm{Im}[Z^m_\ell(\Omega,\Phi)] \right|^2 \Bigg\}.$$

- We absorb $\sqrt{1 \pm P_\gamma}$ into the definition of Z_l^m and write $[\ell]^{(\pm)}_m$ as either a BreitWigner (for $\ell = 2$) or piecewise-defined function (for $\ell = 0$)
- Note that one $[\ell]^{(\pm)}_m$ appears in two sums: the production coefficients for each must be constrained to be the same and both terms must be included when computing anything physical from the result
- Multiple reactions are like doing multiple fits simultaneously: contributions to ln(L) add, parameters can be constrained across reactions

Example: Configuring Inputs

```
######################################
# GLOBAL VARIABLES
######################################

fit etapi0_SD_TMD_onePol

define polVal_00 0.3519
define polAngle_00 0.0


define atwo 1.312 0.113

parameter pcwsBin_1ImPos 0.0 fixed
parameter pcwsBin_1ImNeg 0.0 fixed

include starting_params.cfg


######################################
# SETUP INPUT, REACTIONS, SUMS
######################################
```

data: signal region events usually with unity weight, may contain backgrounds

bkgnd: (often) weighted sample that is statistically consistent with the background contribution to the signal region

accmc: accepted signal MC, consistent with the signal portion of the data sample

genmc: generated MC, used for denominator in efficiency calculations; in GlueX this should be tagged, generated MC

```
reaction EtaPi0_00 Beam Proton Eta Pi0

data EtaPi0_00 FSRootDataReader fsroot/tree_pi0eta__B4_M17_M7_DATA_sp17_pol0_SIGNAL_SKIM_A2.root ntFSGlueX_101_1 3

bkgnd EtaPi0_00 FSRootDataReader fsroot/tree_pi0eta__B4_M17_M7_DATA_sp17_pol0_SIDEBANDS_SKIM_A2.root ntFSGlueX_101_1 3
fsroot/tree_pi0eta__B4_M17_M7_DATA_sp17_pol0_SIDEBANDS_SKIM_A2.root.weight ntFSGlueX_101_1_weight weight

accmc EtaPi0_00 FSRootDataReader fsroot/tree_pi0eta__B4_M17_M7_MC_sp17_pol0_SIGNAL_SKIM_A2.root ntFSGlueX_101_1 3
fsroot/tree_pi0eta__B4_M17_M7_MC_sp17_pol0_SIGNAL_SKIM_A2.root.weight ntFSGlueX_101_1_weight weight

genmc EtaPi0_00 FSRootDataReader fsroot/tree_pi0eta__B4_M17_M7_MCGEN_sp17_pol0_GENERAL_SKIM_A2.root ntFSGlueX_101_1 3 MC
```

Example: Setting Up Amplitudes

```
sum EtaPi0_00 ReZ_1-P
sum EtaPi0_00 ImZ_1+P
sum EtaPi0_00 ReZ_1+P
sum EtaPi0_00 ImZ_1-P
```

$$I(\Omega,\Phi) = 2\kappa \sum_k \left\{ (1-P_\gamma) \left| \sum_{\ell,m} [\ell]^{(-)}_{m;k} \mathrm{Re}[Z^m_\ell(\Omega,\Phi)] \right|^2 + (1-P_\gamma) \left| \sum_{\ell,m} [\ell]^{(+)}_{m;k} \mathrm{Im}[Z^m_\ell(\Omega,\Phi)] \right|^2 + (1+P_\gamma) \left| \sum_{\ell,m} [\ell]^{(+)}_{m;k} \mathrm{Re}[Z^m_\ell(\Omega,\Phi)] \right|^2 + (1+P_\gamma) \left| \sum_{\ell,m} [\ell]^{(-)}_{m;k} \mathrm{Im}[Z^m_\ell(\Omega,\Phi)] \right|^2 \right\}.$$

```
#####################################
# DEFINE AMPLITUDES
#####################################

# S-wave amplitudes
amplitude EtaPi0_00::ReZ_1-P::S0- Zlm 0 0 +1 -1 polAngle_00 polVal_00
amplitude EtaPi0_00::ImZ_1+P::S0- Zlm 0 0 -1 +1 polAngle_00 polVal_00
amplitude EtaPi0_00::ReZ_1-P::S0- Piecewise 1.04 1.56 4 23 Neg ReIm [pcwsBin_0ReNeg] [pcwsBin_0ImNeg] [pcwsBin_1ReNeg]
[pcwsBin_1ImNeg] [pcwsBin_2ReNeg] [pcwsBin_2ImNeg] [pcwsBin_3ReNeg] [pcwsBin_3ImNeg]
amplitude EtaPi0_00::ImZ_1+P::S0- Piecewise 1.04 1.56 4 23 Neg ReIm [pcwsBin_0ReNeg] [pcwsBin_0ImNeg] [pcwsBin_1ReNeg]
[pcwsBin_1ImNeg] [pcwsBin_2ReNeg] [pcwsBin_2ImNeg] [pcwsBin_3ReNeg] [pcwsBin_3ImNeg]

amplitude EtaPi0_00::ImZ_1-P::S0+ Zlm 0 0 -1 -1 polAngle_00 polVal_00
amplitude EtaPi0_00::ReZ_1+P::S0+ Zlm 0 0 +1 +1 polAngle_00 polVal_00
amplitude EtaPi0_00::ImZ_1-P::S0+ Piecewise 1.04 1.56 4 23 Pos ReIm [pcwsBin_0RePos] [pcwsBin_0ImPos] [pcwsBin_1RePos]
[pcwsBin_1ImPos] [pcwsBin_2RePos] [pcwsBin_2ImPos] [pcwsBin_3RePos] [pcwsBin_3ImPos]
amplitude EtaPi0_00::ReZ_1+P::S0+ Piecewise 1.04 1.56 4 23 Pos ReIm [pcwsBin_0RePos] [pcwsBin_0ImPos] [pcwsBin_1RePos]
[pcwsBin_1ImPos] [pcwsBin_2RePos] [pcwsBin_2ImPos] [pcwsBin_3RePos] [pcwsBin_3ImPos]

# D-wave amplitudes
amplitude EtaPi0_00::ImZ_1-P::a2_D0+ Zlm 2 0 -1 -1 polAngle_00 polVal_00
amplitude EtaPi0_00::ReZ_1+P::a2_D0+ Zlm 2 0 +1 +1 polAngle_00 polVal_00
amplitude EtaPi0_00::ImZ_1-P::a2_D0+ BreitWigner atwo 2 2 3
amplitude EtaPi0_00::ReZ_1+P::a2_D0+ BreitWigner atwo 2 2 3
```

amplitude factors with the same reaction::sum::amplitude are multiplied together

Example: Constraints and Initialization

Running the Fit

Viewing the Output

Extending to Multiple Polarization States

MPI Acceleration

GPU Acceleration