The PyPWA Project

A software framework used to perform Partial Wave and Amplitude Analysis (PWA) with the goal of extracting resonance information from multiparticle final states

Josh Pond
Norfolk State University

PyPWA by, S. Bramlett, B. DeMello, W. Phelps, J. Pond, C. W. Salgado

Outline

- Overview
- Uses
- Benefits
- Current Users
- General Shell
- Isobar
- Goals
- Download
- Summary

Overview

PyPWA is a two sided software framework and toolkit for Partial wave and Amplitude analysis.

GENERALSHELL

- Fitting and Simulation
- Can use any model
- Interface is through a user defined Python script taken from a template.
- We're investigating integrated batch farm interaction
- ISOBAR plotting can be used if the ISOBAR file structure is mimicked by the user.
- We're looking into ways to generalize plotting utilities.

ISOBAR

- Fitting and Simulation
- Exclusively uses the isobar amplitude model
- Easy install and mass binning
- Takes advantage of the GAMP event format and the GAMP amplitude generator utilizing keyfile physics descriptions
- Interface is with simple GUIs
- Interacts directly and exclusively with the Jlab batch farm
- Integrated plotting through Python

Uses

GENERALSHELL

- Parametric fitting of data using any physics model
- Simulating data from phase space Monte Carlo using Rejection Sampling

ISOBAR

- Meson Spectroscopy and Partial Wave Analysis using the Isobar model
- Simulating data from phase space Monte Carlo using Rejection Sampling
- Analysis of data using mass plotting tools

Benefits

General Shell

- Python is a high level language which eases the writing of intensities.
- Access to all Python libraries
- Integration with lower level languages is easy(F2PY, CYTHON)
- Optional use of Q factor signal probability
- GS is a convenient interface with Minuit.

ISOBAR

- Integrated Isobar model
- Ease and Speed of use for Jlab users
- Integration directly to the batch farm
- Optional use of Q factor signal probability

Current Users

PyPWA is currently being used in several JPAC and *g*12 analyses with great success and very positive early signs.

General Shell

The General shell side of PyPWA is focused on openness and generality.

The General Shell uses code inputs from the user, but can fit any model using the *Un-binned Extended Likelihood* method.

$$-ln\mathscr{L} = -\sum_{i=1}^{N} Q_i ln\left[I(\vec{x}_i, \vec{a})\right] + \frac{1}{N_g}\sum_{i=1}^{N_a} I(\vec{x}_i, \vec{a})$$

General Shell cont.

Beginning the Process

Binning and file structures and anything else to do with the user's specific fit, or simulation is up to them. The only thing they need to start is two files for fitting, or simulation, and an extra variable parser utility for fitting.

For both fitting and simulation there is one file that the user interacts with and one they can leave alone.

Simulation only takes the four momenta of each particle in the event, while fitting only takes text files of variables in a specific format: X1=0.25,X2=1.67,X3=90.5 ...

General Shell cont.

Main Point of Contact

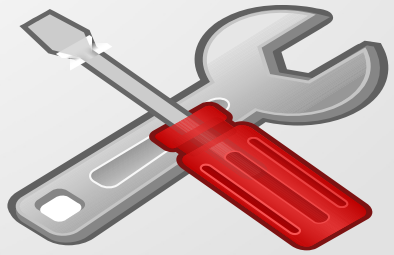
The main points of contact for the user within the General Shell are the Fn.py and FnSim.py files. They are in the download as FnTemplate.py and FnSimTemplate.py.

They include documentation and examples to help the user write their intensity function, but a basic knowledge of Python is required.

They are both a series of functions that each do a specific job for the calculations involved with fitting and simulation. This includes the intensity function, and the initial values and limits for fitting parameters. These files will have to be changed for every different fit, or simulation.

Results and Plotting

At this time the General Shell does not support any other utilities for analysis, or plotting.



Code and Example

```
def intFn(kVars,params):

    tDist = params['A1']*numpy.exp(params['A2']*(kVars['tM']))

    wConst = (3.0/(4.0*math.pi))
    W = wConst*(0.5*(1-params['A3'])+0.5*(3*params['A3']-1)*math.cos(kVars['theta'])**2
        -math.sqrt(2.0)*params['A4']*math.sin(2*kVars['theta'])*math.cos(kVars['phi'])
        -params['A5']*(math.sin(kVars['theta']))**2*math.cos(2*kVars['phi']))
    F=AMP.amp(kVars['s'],kVars['t'],kVars['u'],params['A6'])
    Fsquare=F*numpy.conjugate(F)

    return tDist*W*kVars['P']*Fsquare
```

This is an example of the sort of function you can fit with PyPWA General. This is the intFn() function inside Fn.py and it's arguments are the two keyed dictionaries, kVars and params. Kvars is the variables parsed from the text file, while params is the parameters fitted by Minuit.

ISOBAR

The ISOBAR side is a continuation of PWA2000. With the use of the isobar model as described in *C. W. Salgado and D. P. Weygand, Physics Report 537 (2014) we are able* to make several assumptions within the mathematics which increases the simplicity and the speed of calculation.

This is accomplished by using GAMP and the increased ability to calculate pieces of the intensity and parts of the log likelihood function, like the normalization integral(Ψ^x), in advance and saving them to the disk before fitting.

$$-ln\mathscr{L} = -\sum_{i=1}^{N} ln\left[\sum_{\alpha,\alpha'} A_\alpha(\vec{x}_i) V_\alpha \rho_{\epsilon,\epsilon'} V^*_{\alpha'} A^*_{\alpha'}(\vec{x}_i)\right]$$

$$+\eta_x \sum_{\alpha,\alpha'} V_\alpha V^*_{\alpha'} \Psi^x_{\alpha,\alpha'}.$$

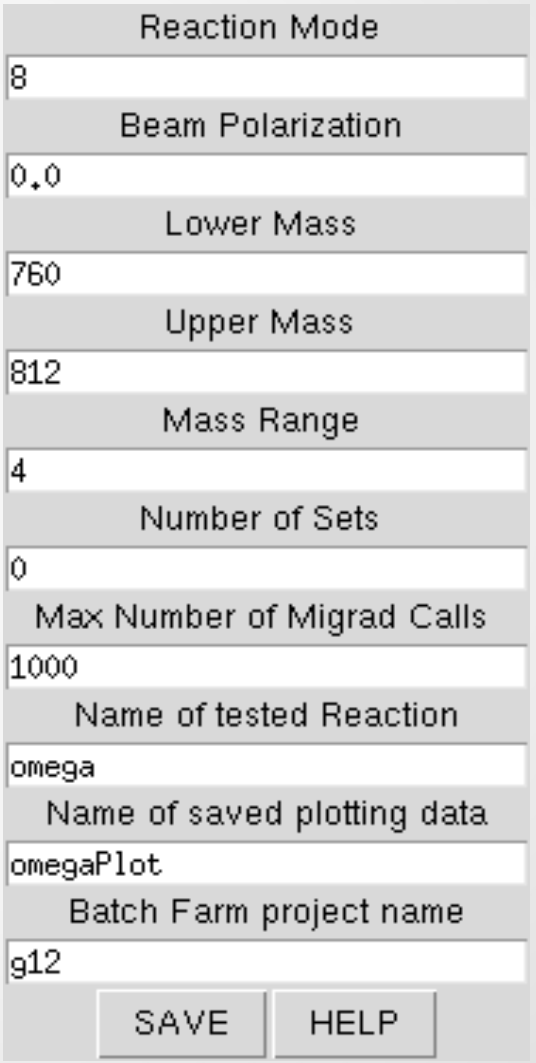
ISOBAR cont.

Beginning the Process

The Isobar framework is focused on ease of use and speed. So from the install process until plotting almost everything is automated.

Install is handled by a single program which opens the control GUI, creates the needed directory structure, moves files to their correct location, and does the mass binning, which can take awhile if the user has many events.

The control GUI at right is the first point of contact the user has with PyPWA and the information filled into it will be used throughout the fitting and simulating process.



ISOBAR cont.

Main Point of Contact

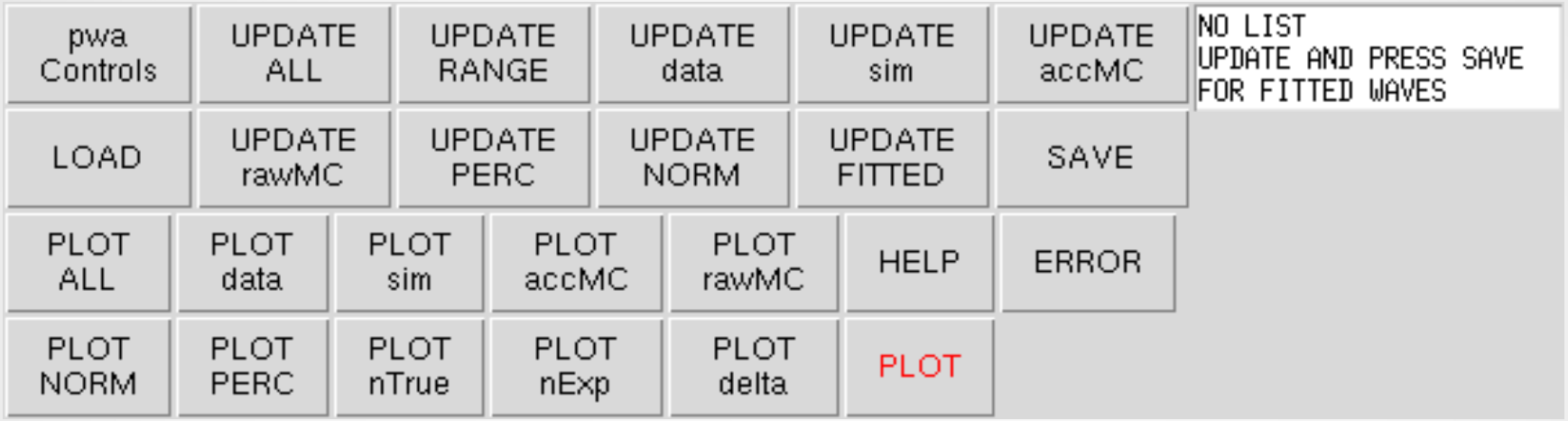
The Isobar framework's main point of contact for the user is the PWA_GUI at right. The left column is what appears when the program is run and the right is what appears after the FITTING button is pressed.

Each button on the right represents a different step in the fitting process and runs a different program. Each of these buttons will run the program which creates and submits many jsub files directly to Auger.

This GUI also has access to the control, the plotter, and the Waves utility.

ISOBAR cont.

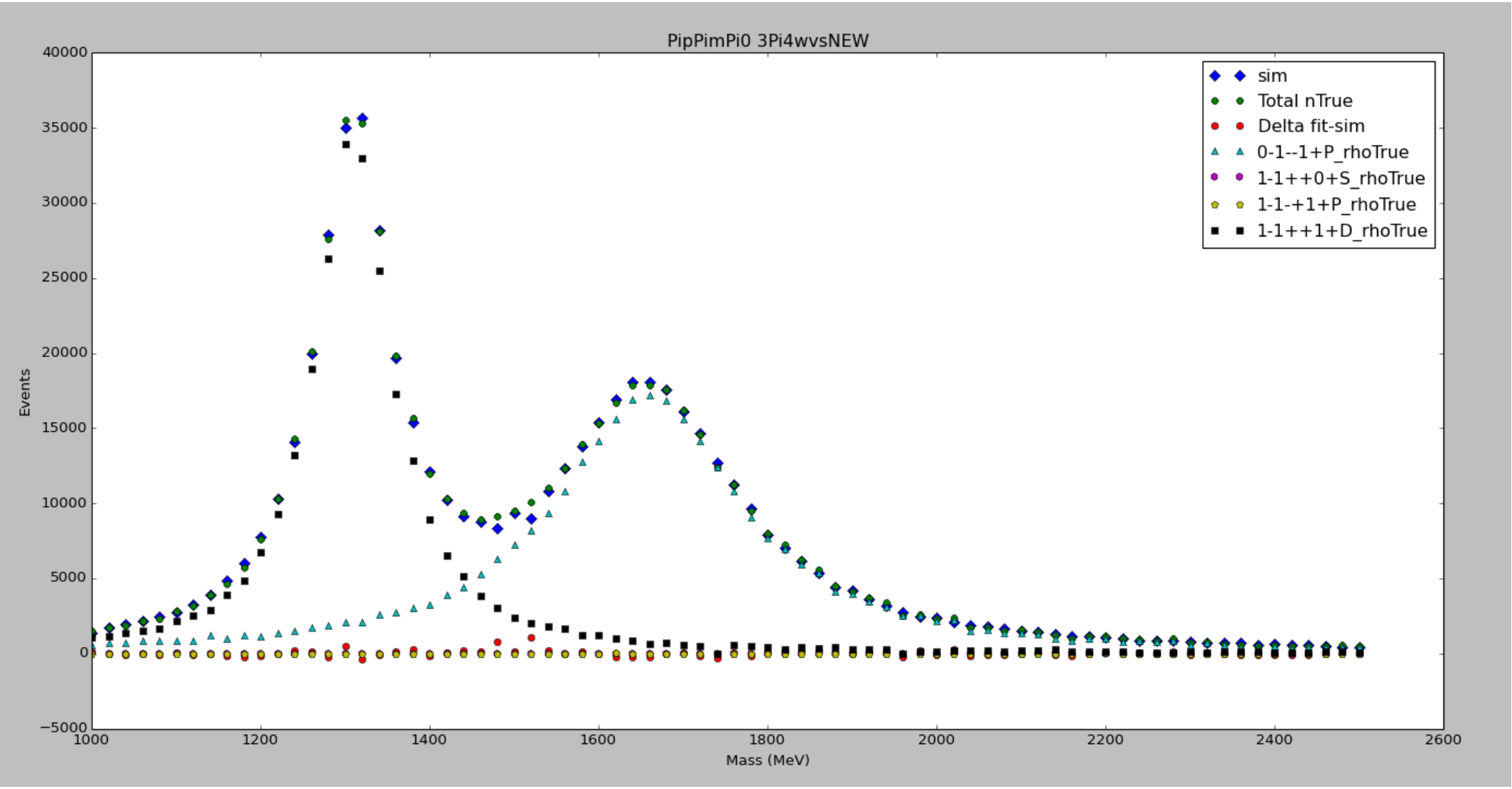
Plotting



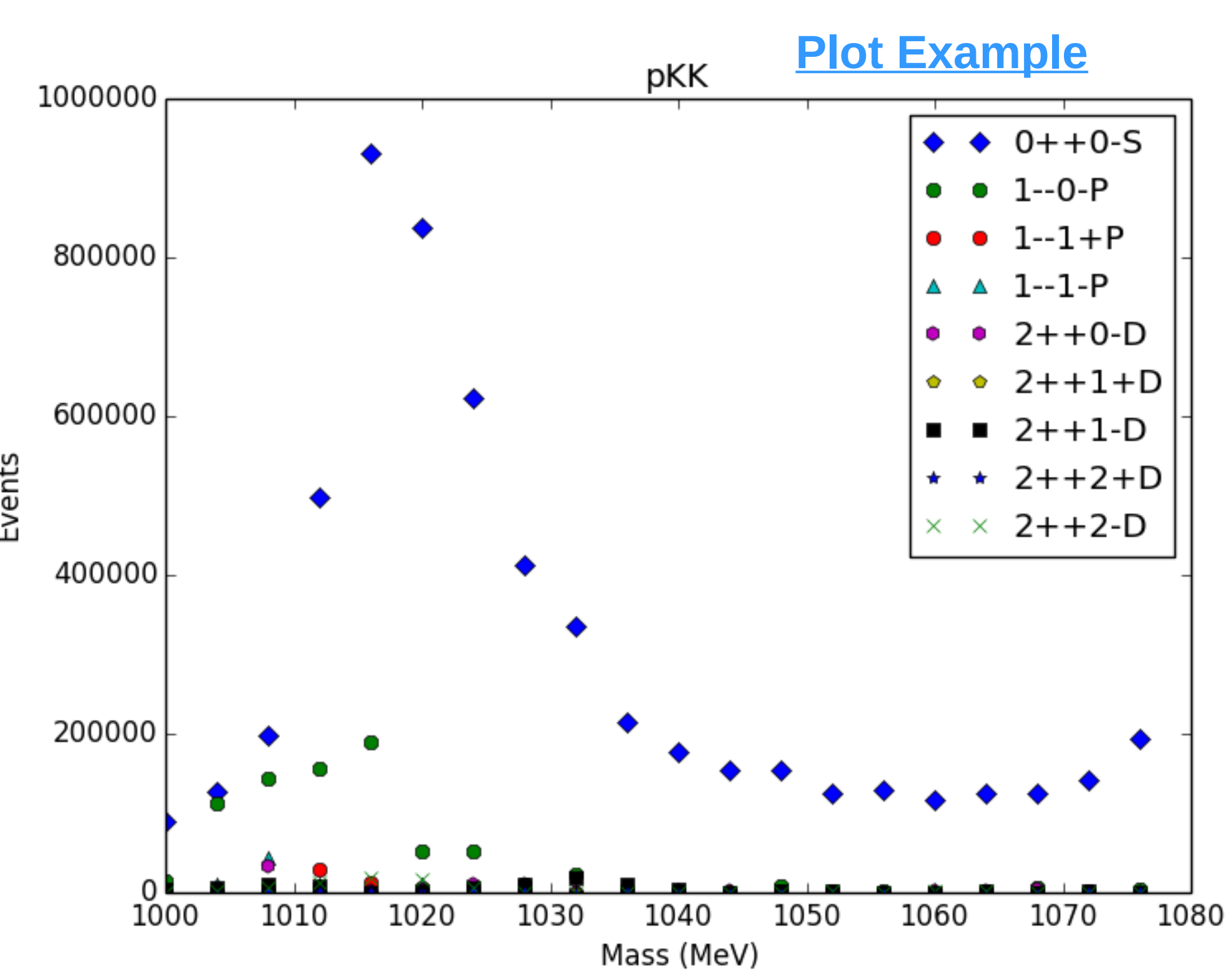
Plotting in PyPWA Isobar is handled by the above GUI which uses the MatPlotLib Python library for all plotting.

This program also consolidates all data for plotting into single file named in the control. This file can be loaded in the future and multiple files can be saved and loaded at different times.

Plot Example

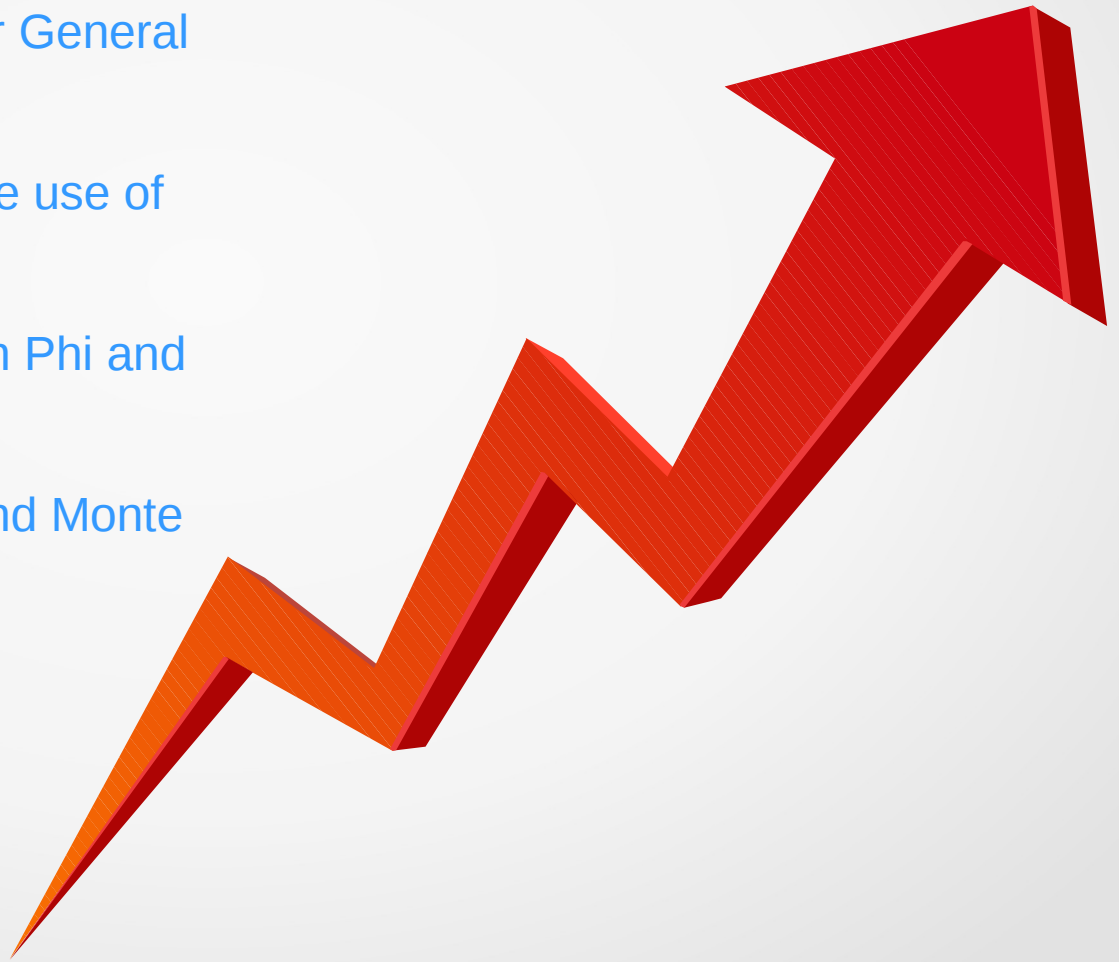


Plot Example



Goals

- Adding more add-ons and utilities to make General Shell even easier to use
- Farm and plotting integration for General Shell
- Increased parallelization with the use of threading
- Hardware acceleration with Xion Phi and GPU's
- Integrating more optimization and Monte Carlo methods



Privacy and Security Notice

Jefferson Lab
EXPLORING THE NATURE OF MATTER

Download PyPWA

LINKS
PYPWA Home
General Description
Installation
Tutorials
Documentation - Sphinx
Software Download
Wiki
References
Contact Us

A Partial-Wave/Amplitude Analysis Software Framework

The PyPWA Project
Thomas Jefferson National Accelerator Facility
Newport News, VA

Home

The PyPWA Project aims to develop a software framework that can be used to perform parametric model fitting to data. In particular, Partial Wave and Amplitude Analysis (PWA) of multiparticle final states. PyPWA is designed for photoproduction experiments using linearly polarized photon beams. The software makes use of the resources at the JLab Scientific Computer Center (Linux farm). PyPWA extract model parameters from data by performing extended likehood fits. Two versions of the software are develop: one where general amplitudes (or any parametric model) can be used in the fit and simulation of data, and a second where the framework starts with a specific realization of the Isobar model, including extensions to Deck-type and baryon vertices corrections. Tutorials (Step-by-step instructions) leading to a full fit of data and the use of simulation software are included. Most of the code is in Python, but hybrid code (in Cyhon or Fortran) has been used when appropiate. Scripting to make use of vectorization and parallel coprocessors (Xeon-Phi and/or GPUs) are expected in the near future. The goal of this software framework is to create a user friedly enviroment for the spectroscopic analysis of linear polarized photoproduction experiments. The PyPWA Project software expects to be in a continue flow (of improvements!), therefore, please check on the more recent software download version.

Summary

- PyPWA, both General and Isobar, is the easiest way for Jlab users to get into partial wave and amplitude analysis.
- Python is a high level language which eases the writing of intensities.
- Access to all Python libraries
- Integration directly to the batch farm
- Integration with lower level languages is easy
- Includes optional use of Q factor signal probability
- Download PyPWA at pypwa.jlab.org.