ComPWA: A common amplitude analysis framework for PANDA

M Michel\textsuperscript{1}, F Feldbauer\textsuperscript{1,2}, K Götzen\textsuperscript{1,3}, P Jasinski\textsuperscript{1,2}, A Karavdina\textsuperscript{2}, K Peters\textsuperscript{1,3}, M Fritsch\textsuperscript{1,2}

\textsuperscript{1} Helmholtz-Institut Mainz, Germany
\textsuperscript{2} Johannes Gutenberg-Universität Mainz, Germany
\textsuperscript{3} GSI Helmholtzzentrum für Schwerionenforschung GmbH, Darmstadt, Germany

E-mail: michel@kph.uni-mainz.de

Abstract. A large part of the physics program of the PANDA experiment at FAIR deals with the search for new conventional and exotic hadronic states like e.g. hybrids and glueballs. For many analyses PANDA will need an amplitude analysis, e.g. a partial wave analysis (PWA), to identify possible candidates and for the classification of known states. Therefore, a new, agile and efficient amplitude analysis framework ComPWA is under development. It is modularized to provide easy extension with models and formalisms as well as fitting of multiple datasets, even from different experiments. Experience from existing PWA programs was used to fix the requirements of the framework and to prevent it from restrictions. It will provide the standard estimation and optimization routines like Minuit 2 and the Geneva library and be open to insert additional ones. The challenges involve parallelization, fitting with a high number of free parameters, managing complex meta-fits and quality assurance/comparability of fits. To test and develop the software, it will be used with data from running experiments like BaBar or BESIII. These proceedings show the status of the framework implementation as well as first test results.

1. Introduction

Future experiments like PANDA at FAIR, Darmstadt, will produce very precise data with high statistics in various final states [1]. One of the goals is to find new conventional and exotic hadronic states like e.g. hybrids and glueballs, and also to determine their quantum numbers, line shapes and fit fractions. Performing these tasks calls for an efficient, flexible and reliable PWA toolkit. Due to the expected high statistics, it must be able to provide efficient parallelization on different levels using computer farms, clusters etc. to handle big datasets. As the physics program of PANDA is broad, it must be flexible enough to allow easy implementation of new amplitude models as well as specialized estimator functions for new fitting methods. Finally, as most amplitude analyses aren’t straight forward, one needs a very reliable software and quality assurance.

These topics lead to the idea to start the development of a PWA toolkit for PANDA already years before the data taking starts. It provides enough time to design the software thoughtfully for all challenges which lie ahead, to learn from existing software tools and to test and revise the toolkit by using it for other experiments and cross check results obtained with existing tools. As cross-channel analyses will be performed by PANDA, part of the flexibility for cross-experiment
analyses must be provided anyway. Therefore a cross-experiment framework is just the next step. Fit results from different experiments obtained with the same software tool allows for comparison, which is not possible in the same way when using different software tools. To start the design, the PANDA group assigned to this project met with experts from other experiments which use PWA techniques to analyze their data. This was done to determine the requirements for the new framework, learn from restrictions of the existing tools and to sketch out what common ground can be found as basis for the cross-experiment software ComPWA.

2. Amplitude analysis
Amplitude analyses are performed to find models and model parameters for the dynamics of reactions which best describe a measured cross section. These models represent our knowledge about the production processes. Quantum mechanics tells us that all possible reactions with different resonances in the intermediate state cannot be distinguished by measurement due to the same final state and because they are interfering with each other. This makes the observation and separately fitting of these resonances very difficult. Therefore, the whole phase space has to be analyzed properly to be sure to interpret interference patterns correctly. One way to model the resonant behavior is called partial wave analysis [2], in which the scattering matrix element is modeled as a composition of spin dependent partial waves by inserting a complete set of states. Although this is just one possible way to perform an amplitude analysis, the acronym PWA is very settled in the community for many kind of analyses and will also be used in this note the same way.

3. Modular design
The requirement process led to the following modules, which describe the basic parts of a partial wave analysis (Figure 1). The four core modules which build up the fitting procedure are: the Data & MC module providing all experimental information, the Physics & Models module providing the amplitude, the Estimators module which estimates how well the amplitude describes the data and the Optimizer module to alter the parameters of the amplitude during the fit process. In addition there are the Documentation module, which performs logging and documentation of the fit procedure and the Controls module, which provides tools for managing the complete fit procedure. In the following each module is described in detail.

3.1. Data & MC module
This module contains all experimental data. It provides four-momenta of all particles per event for unbinned fits, binned data for binned fits, event independent information like beam energies and the possibility to add information, e.g. Monte Carlo generated events for future fits. In terms of efficiency, the large amount of data needs caching. Reading just a certain part of the
events must be possible to enable parallelization. For the interface to the other modules an internal format for the four-momenta and other values is used. In the setup phase, the module provides the information what final state is represented in the data. During runtime additional values derived from the initial data set are provided to the fit. It is taken care that this module is general enough to insert data reader for any kind of data format. Presently two kind of implementations are available, one using Ascii files and one using ROOT files [3] for in- and output.

3.2. Physics & Models module
This module calculates the amplitude at a given phase space point or bin. Therefore it needs a complete parameter list, which is used by the model. In the setup phase, it declares for what states the model is suited and provides a list of parameters with starting values. During runtime the module calculates the complete amplitude for the actual parameter list at every required phase space point. There also can be different models in the Physics & Models module, e.g. the isobar model, and different formalism, e.g. helicity, K-Matrix etc. The amplitude can optionally be provided in form of a function tree, which is described in section 4. Presently, in ComPWA only one amplitude model is available so far using the function tree: the description of a resonance using the product of a relativistic Breit-Wigner function with a Wigner-D function.

3.3. Estimators module
The Estimators module contains estimation functions which calculate how good the amplitude with the actual set of parameters describes the data. For this it needs for every event or bin the information from the Data & MC module, either only the phase space information or in addition the number of entries per bin, as well as the amplitude from the Physics & Models module at this point with the given parameter list. Special estimation functions might be needed in case of parallelization on data distribution level, when the likelihood estimation function has to gather parts of its calculation from different nodes which use only part of the data. Examples already implemented in ComPWA are the unbinned negative log likelihood and the $\chi^2$ estimation functions.

3.4. Optimizer module
The Optimizer module varies parameters of the parameter list to find the optimal set of parameter values, where the estimator function has its optimal value. ComPWA makes use of the optimization libraries Minuit2 [4] and Geneva (Grid ENabled Evolutionary Algorithms) [5]. Minuit2 offers well tested optimization routines which are used widely in high energy physics. Geneva offers new techniques for optimization especially for environments with a lot of free parameters using genetic and Monte-Carlo algorithms. Also it provides parallelization on optimization level out of the box in a very efficient way, because it is natural for genetic algorithms. Both external libraries are used in ComPWA by providing wrapper around their native interfaces. It is also possible to write own implementations of optimization algorithms or wrapper to other external optimization libraries.

3.5. Documentation module
The documentation and fit logging will be provided in an one way approach. All necessary information will be send to a static documentation module, which handles the storage. The module is called Documentation module, as it is meant to provide not only logging and results, but also persisted snapshots of the program and more. This enables not only the possibility to reproduce results but also to restart fits at a certain step of the optimization procedure e.g. in case of interrupted fits. Information will be extracted by using histogramming and other
tools which are stand alone, although they might use part of ComPWA. There is no special implementation for this module and its interface yet, instead ROOT files [3] and log files are used for documentation.

3.6. Controls module
The Controls module will provide an interface to control a complete fit or a set of fits by using a high-level scripting language. This is necessary to perform meta fits, e.g. re-performing the same fit multiple times with different starting values to determine systematic errors. The choice of the scripting language will be kept open for individual choices. At this stage the set up and runtime of a fit in ComPWA is managed using executable binaries.

4. Function Tree
The function tree is used to speed up the calculation of amplitudes and estimation functions of any kind. Standard optimization algorithms, e.g. gradient descent strategies, usually change one parameter at a time. This means only changing part of the amplitude, and most of its terms stay constant during recalculation of the estimation function. Using the function tree structure provided by ComPWA parts of the amplitude will be cached and only parts which are flagged for recalculation are evaluated.

Figure 2 shows an example for a calculation performed with the function tree to explain the basic functionality. The calculation

\[ A = a \times (b + c) \]

is split into two parts

\[ A = a \times (bc) \quad \text{and} \quad bc = b + c \]

In terms of the function tree, it consists of a top node \( A \) which contains the result of the calculation. Also it consists of three leaves representing the tree parameters \( a, b \) and \( c \). In addition, it has a node \( bc \), which calculates \( b + c \) and stores the result. This storage of intermediate results is the key feature of the function tree. If only parameter \( a \) changes in our example, then \( bc \) is not recalculated but the stored result will be used to calculate \( A \). This method can save a lot of computing time in case of difficult decay trees with a lot of resonances, especially when using a gradient descent optimization routine.

As this tree needs to be calculated for a set of events or bins, everything will be hold in vectors, whose size is the number of events or bins. An interface is provided to easily create a function tree. For the calculation of the value of a node a strategy pattern [6] is used. In our example (Figure 2) the implementations of the strategies are only the basic operations (sum and product of two values). Depending on the amplitude model the user can define his own strategies, e.g. a Breit-Wigner function to describe the dynamics of a resonance. The observer pattern [6] was used for the leaves of the tree. Instead of storing their own value, the leaves could link to an external parameter, e.g. of the parameter list used by the optimizer. The leaves are acting as observers and the parameter, acting as the observed object, informs its leaf and other observers if it has changed. With these features the function tree is easy to use and provides efficient calculations for all kind of models.
<table>
<thead>
<tr>
<th>Resonance</th>
<th>PDG mass $[\text{GeV}/c^2]$</th>
<th>mass initial $[\text{GeV}/c^2]$</th>
<th>mass fitted $[\text{GeV}/c^2]$</th>
<th>$\Delta/\text{PDG}$ [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_0(980)$</td>
<td>0.9900</td>
<td>0.940500</td>
<td>0.990035</td>
<td>0.00354</td>
</tr>
<tr>
<td>$f_2(1270)$</td>
<td>1.274</td>
<td>1.21030</td>
<td>1.27245</td>
<td>0.1217</td>
</tr>
<tr>
<td>$f_0(1500)$</td>
<td>1.505</td>
<td>1.42975</td>
<td>1.50483</td>
<td>0.0113</td>
</tr>
<tr>
<td>$f_0'(1525)$</td>
<td>1.525</td>
<td>1.44875</td>
<td>1.52454</td>
<td>0.0302</td>
</tr>
<tr>
<td>$f_0(1710)$</td>
<td>1.720</td>
<td>1.63400</td>
<td>1.72052</td>
<td>0.0302</td>
</tr>
</tbody>
</table>

5. Test results
A first test with Monte Carlo generated data of the decay channel

$$J/\psi \rightarrow \pi^0\pi^0\gamma$$

was performed using ComPWA with the function tree. The following intermediate resonance were used for the generation:

$$f_0(980), f_2(1270), f_0(1500), f_2'(1525) & f_0(1700) \rightarrow \pi^0\pi^0 \omega \rightarrow \pi^0\gamma$$

The amplitude of the complete model was constructed by using the complex sum of all resonant terms which takes into account all interference among the resonances. The complete amplitude is described as

$$A = \sum_n T_n c_n D_n$$

where $T$ is a relativistic Breit-Wigner function, $c$ a complex parameter for strength and phase and $D$ the spin-dependent Wigner D-function (using qft++ [7]). This amplitude was used to generated the toy data by the Hit&Miss method and afterwards to fit this amplitude to the data, in order to compare the parameters from the generation of the toy data with the fit results. The distribution of 100000 events of toy data within the phase space is shown in Figure 3 by using the resonance parameters from the Particle Data Group [8]. All $c_n$ where set to strength 1.0 and phase 0.0. To this toy data a fit was performed using an unbinned negative log likelihood estimator function and for the optimization the Minuit2 [4] gradient descent strategy. The result is shown in Figure 4. Again, 100000 events were generated in the Hit&Miss procedure, in this case by using the model parameters from the fit result. In addition, the ratio of the toy data sets (with the parameters from PDG and from fit results) is shown in Figure 5, were bins with a high statistic uncertainty (> 5%) were excluded. Table 1 shows the parameters which were optimized in the fitting procedure. Only the masses of the $f$ resonances were altered during the fit, all other parameters were kept fixed.

For completeness, Figure 6 shows the same amount of events generated without any resonance model. It proves that no underlying effects like background, detector inefficiencies or reconstruction ambiguities are simulated. As expected the fit performed nicely and is mostly limited by statistics and binning effects, as one can see e.g. at the edges of the phase space.
Figure 3. Distribution of toy data (100000 events) within the three-particle phase space using resonance parameters from PDG for the amplitude calculation.

Figure 4. Distribution of toy data (100000 events) within the three-particle phase space using fitted resonance parameters for the amplitude calculation.

Figure 5. Ratio between toy data distributions of figure 3 and figure 4.

Figure 6. Distribution of toy data (100000 events) within the three-particle phase space without model.
6. Conclusion
A first implementation of the common amplitude analyses framework ComPWA is ready and the interfaces are now being tested. ComPWA consists of a Root and an Ascii Data & MC module, a Physics & Models module which contains the sum of relativistic Breit-Wigner functions for amplitude calculation, an unbinned negative log likelihood and a binned $\chi^2$ estimation function in the Estimators module and a wrapper to external optimization libraries in the Optimizer module which supports Minuit2 and Geneva. The Documentation module and the Controls module are still missing and done by hand up to now. In order to optimize the computing resources a function tree is implemented and tested. As a proof of the functionality, ComPWA is able to generate toy data and perform a fit to them. Results are presented using a simplified model in an idealistic case which shows that the fitting procedure is working correctly. The next steps, beside adding the mentioned missing modules, are to add additional physics cases and physics models in order to test the interfaces of ComPWA and to cross-check the results.

References