IL NUOVO CIMENTO **47 C** (2024) 203
DOI 10.1393/ncc/i2024-24203-0

Colloquia: HADRON2023

Amplitude analysis tools at BESIII

Yi Jiang

University of Chinese Academy of Sciences - Beijing, People's Republic of China

received 21 December 2023

Summary. — Many amplitude analysis tools have been used at BESIII. The new amplitude analysis tool TFPWA is developed to cover new requirements, and, for a general framework, to do the amplitude analysis automatically, efficiently and conveniently. Many new technologies are used especially automatic differentiation. TFPWA has been already used in BESIII and shows good performances.

1. – Introduction

In the field of particle physics, there are numerous particles that we study and of which we analyze the interactions. One useful technique for studying these interactions is amplitude analysis or, as it is often named, partial wave analysis (PWA). We construct an amplitude model to describe the interactions, which represents the probability density of the 4-momentum for particles. And then we can use it to fit the data. Based on the model components and the parameters, the interesting information of the interactions can be extracted. For example, we can determine the necessary new states and their properties. And we can use them to understand the variation over the phase space, such as charge-parity violation.

Table I shows the amplitude analysis tools used in BESIII. Many of the previous tools are hand-coded and not so easy for other processes. Deep understanding of the formula is required. We propose a general amplitude analysis tools, TFPWA. It aims for automatic and powerful framework of PWA.

The code is open source at `https://github.com/jiangyi15/tf-pwa`.

2. – Framework

The main problem of PWA is to build a model which can describe the real data. The common parts for models are the particles included and the decays that can happen. Usually, we have one initial particle and some final particles. One particle can decay into two particles. This condition is often called isobar model. The initial particle and final particles are specified. Particles between initial particle and final particles can be added to connect them. The main properties of those particles are their spins, parities, masses and widths. It can be represented as a simple dict. The decay can be represented as a

Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0)

TABLE I. – *Amplitude analysis tools in BESIII.*

Tools	Comment	Example
closed source/hand coded	–	$D^+ \to K^0_S\pi^+\pi^0\pi^0$ [1],$e^+e^- \to \omega\pi^+\pi^-$ [2]
GPUPWA [3]	tensor formalism	$J/\psi \to \gamma\eta\eta$ [4],$J/\psi \to \gamma\eta\eta'$ [5]
FDC-PWA [6]	tensor formalism	$\psi' \to p\bar{p}\eta$ [7],$e^+e^- \to pK^-\bar{\Lambda}$ [8]
TFPWA (this work)	helicity formalism	$\Lambda_c^+ \to \Lambda\pi^+\pi^0$ [9]
from other experiments	–	$\chi_{c1} \to \eta\pi^+\pi^-$ [10], $D^0 \to K_SK^+K^-$ [11]

tuple of the input particle and output particles with some additional information in a dict. They are the simple configuration of the model. In TFPWA, we used YAML [12] format to store that configuration in a file. That makes the configuration file simple. The YAML file is easy to read, write and modify for different processes.

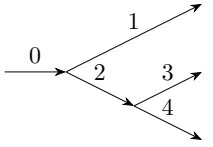
The model is built from this configuration as a tree structure in fig. 1.

TFPWA uses the helicity formula (fig. 1 decay) to calculate the amplitude $A^{0\to1+2}_{\lambda_0,\lambda_1,\lambda_2}$ of one decay where particle 0 decays to particle 1 and particle 2. λ_i is the helicity of particle i. $H^{0\to1+2}_{\lambda_1,\lambda_2}$ is helicity coupling that includes our fit parameters for the model. $D^J_{m,m'}(\alpha,\beta,\gamma)$ is the Wigner-D function. ϕ,θ is the helicity angle that is calculated from 4-momentum (p_i^μ).

The full process has different decay chains including different resonances or different structure. Each decay chain has some decays connecting initial particle and final particles. The amplitude of such decay chain is the combination of all decays included and the shape of inner particles (fig. 1 decay chain). $R_1(m_1)$ is the amplitude of particle 1 that is dependent on the invariant mass. More decays can be added in the same way.

All the decay chains are stored in a decay group. The total amplitude is the summation (fig. 1 decay group). The probability density function (PDF) of the process is proportional to the absolute square of the amplitude.

The main point of the calculation is combining different parts of amplitudes together. TFPWA provides an implementation of that combination (fig. 1 decay chain) as a tree traversal. TensorFlow [13] with GPU support is used for the calculation. It can be used



initial particle: 1 or polarization $\rho = 1 + \vec{p}\cdot\vec{\sigma}$.

resonances: $R(m) = \frac{1}{m_0^2 - m^2 - im_0\Gamma(m)}$.

final particles: alignment $D^{J*}_{\lambda',\lambda}(\alpha,\beta,\gamma)$.

decay: $A^{0\to1+2}_{\lambda_0,\lambda_1,\lambda_2} = H^{0\to1+2}_{\lambda_1,\lambda_2} D^{J*}_{\lambda_0,\lambda_1-\lambda_2}(\phi,\theta,0)$.

decay chain: $A^{0\to1+2,1\to3+4}_{\lambda_0,\lambda_2,\lambda_3,\lambda_4} = \sum_{[\lambda_1],\lambda'_j} A^{0\to1+2}_{\lambda_0,\lambda_1,\lambda'_2}\left[R_1(m_1)A^{1\to3+4}_{\lambda_1,\lambda'_3,\lambda'_4}\right] \prod_j D^{J*}_{\lambda_{j'},\lambda_j}(\alpha_j,\beta_j,\gamma_j)$.

decay group: $A^{0\to2+3+4}_{\lambda_0,\lambda_2,\lambda_3,\lambda_4} = A^{0\to1+2,1\to3+4}_{\lambda_0,\lambda_2,\lambda_3,\lambda_4} + A^{0\to5+3,5\to2+4}_{\lambda_0,\lambda_2,\lambda_3,\lambda_4} + \cdots$.

probability density function: $f_{\rm sig}(p_i^\mu) = \sum_{\lambda_0,\lambda_2,\lambda_3,\lambda_4} |A^{0\to2+3+4}_{\lambda_0,\lambda_2,\lambda_3,\lambda_4}|^2$.

Fig. 1. – Amplitude model in TFPWA.

in any body decays. TFPWA provides a plugin system for new models. All models will be stored in a global dict. A new model can be added simply through a series of register function. An interface for customizing models is the key point for general framework.

The default model for $H^{0\to 1+2}_{\lambda_1,\lambda_2}$ uses the LS coupling formula

$$H^{0\to 1+2}_{\lambda_1,\lambda_2} = \sum_{l,s,\delta=\lambda_1-\lambda_2} g_{l,s}\sqrt{\frac{2l+1}{2j_0+1}} C^{j_0\delta}_{l0;s\delta} C^{s\delta}_{j_1\lambda_1;j_2-\lambda_2} q^l B_l(q,q_0,d). \tag{1}$$

$C^{j_0 m_0}_{j_1 m_1;j_2 m_2}$ is the Clebsch-Gordan coefficients. $q(q_0)$ is the breakout momentum in the 2-body rest frame calculated though 4-monumenn (or calculated though resonance mass). $B_l(q,q_0,d)$ is the a Blatt-Weisskopf barrier factor [14-16]. $d = 3.0\ \mathrm{GeV}^{-1}$ is set as default. The factors $g_{l,s}$ are complex number fit parameters.

The default model for $R(m)$ is a simple Breit-Wigner (fig. 1 resonances) with running width

$$\Gamma(m) = \Gamma_0 \frac{m_0}{m}\left(\frac{q}{q_0}\right)^{2l+1} B_l^2(q,q_0,d). \tag{2}$$

The minimal l is used to keep the sample shape for resonances. Mass (m_0) and width (Γ_0) can be the fit parameters which are determined by the fit.

The input data for each part is different. Structured data are used as input. TFPWA provides an algorithm to calculate the angle $(\theta,\phi,\alpha,\beta,\gamma)$ automatically through 4-momentum. The alignment angle (α,β,γ in fig. 1) calculation is a full automatic implementation of [17]. An additional D matrix with the alignment angle is added to align the helicity of each final particle separately.

The direct calculation process in fig. 1 is robust, but not efficient in all the cases. TFPWA provides additional factor system to calculate the amplitude as

$$A_i = \sum_i c_i f_i(m) T_i(\theta,\phi). \tag{3}$$

The factors c_i are some combinations of $g_{l,s}$. $f_i(m)$ are the mass-dependent parts. $T_i(\theta,\phi)$ are the angle parts which are always independent of the fit parameters.

3. – Automatic differentiation

The fit process is to minimize the negative log-likelihood eq. (4). The signal distribution is the amplitude square. And the background distribution is a fixed distribution

$$-\ln L = -\sum_{x\in\mathrm{data}} \ln\left(\frac{f_{\mathrm{sig}}(x)}{I_{\mathrm{sig}}} + \frac{f_{\mathrm{bg}}(x)}{I_{\mathrm{bg}}}\right). \tag{4}$$

$I_{\mathrm{sig/bg}} = \int f_{\mathrm{sig/bg}}(x)\mathrm{d}x$ is the normalised factor. Due to the complex formula of amplitude model and phase space (PHSP) distribution, such integration is difficult to solve analytically. Large size of Monte Carlo (MC) sample is required to do the numerical integration as $I_{\mathrm{sig/bg}} = 1/N\sum_{x\in MC} f_{\mathrm{sig/bg}}(x)$.

To minimize the $-\ln L$, `minimize` function provided by `scipy.optimize` [18] is used in TFPWA. It provides a better control of the minimization in Python. This function

mainly uses the quasi-Newton method, which required gradients to do the minimization. Automatic differentiation (AD) [19] is a widely used technology to calculate the gradients efficiently through recording the operations of variables and combining the Jacobi matrix of those operations. TensorFlow provides the implementation of AD with complex number support.

However it is not so straightforward to use AD in PWA. Recording the operations requires large memory. When the number of events becomes large, especially for the MC sample, it causes the problem "out of memory". The total samples have to be divided into small batches. TFPWA provides a way to calculate the gradients in small batches and combine them into the total $-\ln L$. Special treatment is required for I_{sig} that is not direct for batches. I_{sig} itself can be rewritten into the sum of different batch sizes as eq. (5), and extend the AD with batch formula too,

$$I_{\text{sig}} = \sum_{x_i \in \text{batch1}} f_{\text{sig}}(x_i) + \sum_{x_i \in \text{batch2}} f_{\text{sig}}(x_i) \cdots , \tag{5a}$$

$$\frac{\partial}{\partial x} I_{\text{sig}} = \frac{\partial}{\partial x} \sum_{x_i \in \text{batch1}} f_{\text{sig}}(x_i) + \frac{\partial}{\partial x} \sum_{x_i \in \text{batch2}} f_{\text{sig}}(x_i) \cdots . \tag{5b}$$

And in the gradients of total $-\ln L$, I_{sig} can be extracted from the gradients as two parts,

$$-\frac{\partial \ln L(x)}{\partial x} = -\frac{\partial \ln L(x; I_{\text{sig}})}{\partial x} - \frac{\partial \ln L(x; I_{\text{sig}})}{\partial I_{\text{sig}}} \frac{\partial I_{\text{sig}}}{\partial x}. \tag{6}$$

This formula is used to build the total gradients out of the AD system of TensorFlow. It extends the power of AD in any scale of data sets in PWA.

Besides minimizing the $-\ln L$, AD can be also used for the uncertainties estimation. The Hessian matrix can be calculated simply thought AD. And AD can also be used in the error propagation as $\sigma[f(x)] = \sqrt{\frac{\partial f}{\partial x_i} V_{ij} \frac{\partial f}{\partial x_j}}$. V_{ij} is the error matrix of parameters. This formula propagates the uncertainties from fit parameters to $f(x)$. For example, the fit fraction of one decay chain can be calculated as $FF_i = \frac{\int |A_i|^2 \mathrm{d}\Phi}{\int |\sum_i A_i|^2 \mathrm{d}\Phi}$. The gradients can be easy to access through AD. Then the uncertainties can be evaluated smoothly.

4. – Performance

TFPWA has already been used in some amplitude analysis at BESIII ($\Lambda_c^+ \to \Lambda\pi^+\pi^0$ [9]) and LHCb ($B^+ \to D^+ D_s^- \pi^+$ [20,21]) experiments. Figure 2 shows the main time components for the normal fit in ($\Lambda_c^+ \to \Lambda\pi^+\pi^0$ [9]). This simultaneous fit includes 7 separate data samples and a total of 8.5×10^5 MC samples([1]). The main component is the time for minimizing $-\ln L$ (fit). Besides, time is required for loading data and visualising the fit results (uncertainties, plots and fit fractions).

Different options are tested. "No option" is the default one. "Mixed likelihood" mixes the 7 separate samples together. "Mass dependent" caches the T_i parts in eq. (3). It will increase the memory costs. "Factor only" caches the $f_i T_i$ in eq. (3), and rewrites

([1]) The code is available at `https://github.com/jiangyi15/tf-pwa-example`.

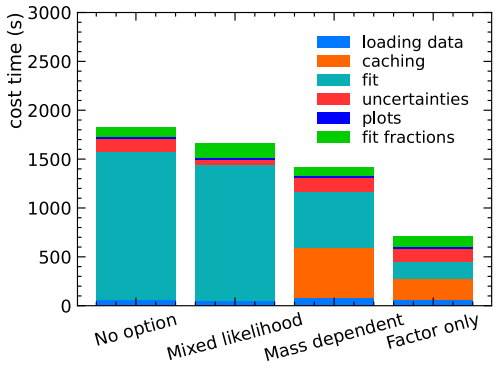


Fig. 2. – Fit time components in $\Lambda_c^+ \to \Lambda\pi^+\pi^0$ [9] for different options.

the integration into $I_{sig} = \sum_{i,j} c_i c_j^* B_i B_j^*$. It is the best one for fit but limited to the condition that f_i are not dependent on fit parameters. A full fit costs about 200 to 300 iterations. Caching will cost much more time than a simple evaluation.

5. – Summary

In conclusion, we propose a general partial wave framework, TFPWA. TFPWA uses YAML format to configure different decay processes. The tree structure allows amplitude calculation automatically. An interface for customizing models is available. It uses powerful AD to do PWA efficiently, and extend it to large data size samples. To balance the performances, different options are provided. It has already been used in real analysis and produced meaningful results. More useful functions will be implemented in the future.

REFERENCES

[1] Ablikim M. *et al.*, arXiv:2305.15879 [hep-ex].
[2] Ablikim M. *et al.*, *JHEP*, **8** (2023) 159.
[3] Berger N., Liu B. and Wang J., *J. Phys.: Conf. Ser.*, **219** (2010) 042031.
[4] Ablikim M. *et al.*, *Phys. Rev. D*, **87** (2013) 092009.
[5] Ablikim M. *et al.*, *Phys. Rev. D*, **106** (2022) 072012.
[6] JianXiong W. *et al.*, *FDC-PWA*, `http://www1.ihep.ac.cn/wjx/pwa/index.html` (2000).
[7] Ablikim M. *et al.*, *Phys. Rev. D*, **88** (2013) 032010.
[8] Ablikim M. *et al.*, arXiv:2303.01989 [hep-ex].
[9] Ablikim M. *et al.*, *JHEP*, **12** (2022) 033.
[10] Ablikim M. *et al.*, *Phys. Rev. D*, **95** (2017) 032002.
[11] Ablikim M. *et al.*, arXiv:2006.02800 [hep-ex].
[12] Ben-Kiki O. *et al.*, *YAML: Yaml ain't markup language*, `https://yaml.org/` (2004).
[13] Abadi M. *et al.*, *TensorFlow*, software available from `tensorflow.org` (2015).
[14] von Hippel F. and Quig C., *Phys. Rev. D*, **5** (1972) 624.
[15] Chung S. U., *Phys. Rev. D*, **48** (1993) 1225.
[16] Chung S. U., *Phys. Rev. D*, **57** (1998) 431.
[17] Wang M. *et al.*, *Chin. Phys. C*, **45** (2021) 063103.
[18] Virtanen P. *et al.*, *Nat. Methods*, **17** (2020) 261.
[19] Baydin A. G. *et al.*, *J. Mach. Learn. Res.*, **18** (2018) 1.
[20] Aaij R. *et al.*, *Phys. Rev. D*, **108** (2023) 012017.
[21] Aaij R. *et al.*, *Phys. Rev. Lett.*, **131** (2023) 041902.