A high performance partial wave analysis framework for hadron spectroscopy*

Benhou Xiang (向本后)[1,2†] Shuangshi Fang (房双世)[1,2,3‡] Beijiang Liu (刘北江)[1,2§]

[1]Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100049, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
[3]School of Physics, Henan Normal University, Xinxiang 453007, China

Abstract: Partial wave analysis (PWA) is a key method in hadron physics for extracting the properties of hadronic resonances. As experimental statistics increase, conventional implementations face substantial challenges in memory usage and computational speed. We present `CTPWA`, a high-performance PWA framework based on the covariant tensor formalism. This framework incorporates extensive precomputation and caching mechanisms, together with fully GPU-based likelihood computation and minimization. These optimizations accelerate fitting by two orders of magnitude relative to autograd-based GPU PWA programs, making high-statistics partial wave analyses feasible in high-precision experiments such as BESIII.

Keywords: partial wave analysis, hadron spectroscopy, high performance tools

DOI: 10.1088/1674-1137/aeb254 **CSTR:** 32044.14.ChinesePhysicsC.51

CPC Accepted

I. INTRODUCTION

Hadron spectroscopy provides a critical window into the non-perturbative regime of Quantum Chromodynamics (QCD) through the study of bound states of quarks and gluons. In the conventional quark model, ordinary hadrons are color-singlet states composed of a quark–antiquark pair (mesons) or three quarks (baryons). QCD, however, also allows more complex configurations beyond this picture, including glueballs, hybrids, tetraquarks, and pentaquarks [1]. Mesonic exotic candidates are traditionally classified into two categories [2]: states with special quantum numbers, either in J^{PC} or in flavor, that cannot be generated by an ordinary quark–antiquark pair; and crypto-exotic states, which share the quantum numbers of conventional mesons and therefore readily mix with them. Beyond the meson sector, analogous exotic signatures also appear in baryon spectra, notably in pentaquark candidates. Determining the genuine nature of these resonances across both meson and baryon systems, whether they are compact multiquark states, hadronic molecules, hybrids, or admixtures thereof, is therefore crucial for mapping the effective degrees of freedom of QCD in the low-energy domain and remains one of the foremost open questions in hadron spectroscopy.

Moreover, these resonances are generally too short-lived to be observed directly, and their properties must be inferred from the final-state particles in the processes in which they are produced. In multi-body decays, multiple intermediate states usually contribute to the same event sample and interfere with one another. Partial wave analysis (PWA) addresses this problem by modeling the decay amplitude and extracting the quantum numbers, masses, widths, and couplings of the contributing states through an unbinned maximum-likelihood fit.

The computational cost of PWA, however, becomes a serious limitation in the high-statistics era. The amplitude must be computed for every event, often for multiple helicity configurations and many interfering partial waves, while the event sample may easily reach 10^5–10^6 events, although the number of free fit parameters is usually moderate. Because the events are mutually independent, this computation is well suited to parallel processing. GPU-based frameworks such as GPUPWA [3] and TF-PWA [4] have therefore been developed to offload these calculations onto graphics processing units; these frame-

Received 27 August 2026; Accepted 9 October 2026; Accepted manuscript online 10 October 2026

* Supported in part by the National Key R&D Program of China (2025YFA1613900), the National Natural Science Foundation of China (NSFC) (12225509, 12235017), and the Strategic Priority Research Program of the Chinese Academy of Sciences (XDA0480600)

† E-mail: xiangbh@ihep.ac.cn
‡ E-mail: fangss@ihep.ac.cn
§ E-mail: liubj@ihep.ac.cn

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI. Article funded by SCOAP[3] and published under licence by Chinese Physical Society and the Institute of High Energy Physics of the Chinese Academy of Sciences and the Institute of Modern Physics of the Chinese Academy of Sciences and IOP Publishing Ltd

works have been widely adopted in the community and underpin many analysis workflows at experiments.

Even with GPU offloading, PWA of large samples faces severe limitations. Device memory is one such limitation. Amplitude arrays or computational graphs are built and cached for all events, and the graph history, i.e. the intermediate values kept on the device for the backward pass, accumulates during the optimization. As a result, memory usage substantially exceeds that of the data themselves, and the device memory can be exhausted before the fit even starts. Speed is another limitation. The amplitude is recomputed at every iteration, so large amounts of data and intermediate results are continuously transferred between the CPU and the GPU, and the fit time is dominated by data-transfer and kernel-launch overheads. In addition, implementations built on PyTorch [5] or TensorFlow [6] re-trace the computational graph at every iteration, an overhead that is unnecessary for a likelihood that is exactly quadratic in the couplings.

In this work, we present `CTPWA`, a lightweight PWA framework based on the covariant tensor amplitude formalism [7–9]. To address the difficulties described above, the framework adopts extensive precomputation and caching mechanisms, as well as fully GPU-based likelihood computation and minimization. The key observation underlying the method is that a substantial part of the decay amplitude is determined solely by the event kinematics and the decay topology and is therefore independent of the fit parameters. These kinematic building blocks can be precomputed once and reused throughout the fit, and only the resonance propagators and coupling combinations need to be updated during minimization. We exploit this observation fully by casting the likelihood into a compact matrix form whose gradient and Hessian follow analytically, thereby avoiding the overhead of graph-based automatic differentiation. The likelihood evaluation, its derivatives, and the limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) minimization [10] run entirely on the GPU. As a result, the required device memory is fixed and known before the fit, no memory is allocated during the optimization, and traffic between the host and the device is reduced to the small parameter vectors (couplings and resonance parameters).

This paper is organized as follows. Section II presents the likelihood construction. Section III describes the precomputation and caching mechanisms, as well as the GPU-based likelihood computation and minimization. Section IV reports the computational performance and a realistic fit of $J/\psi \to K^+K^-\eta$ with one million events. Section V provides a summary.

II. LIKELIHOOD CONSTRUCTION

A partial wave analysis determines resonance parameters from the measured four-momenta of the final-state particles using an unbinned maximum-likelihood fit [3, 4]. Each event is characterized by the four-momenta $x \equiv \{p_1, \ldots, p_n\}$ of its final-state particles. The likelihood for the data sample is given by the product of the per-event probability densities

$$\mathcal{L}(x;\boldsymbol{\theta}) = \prod_{i=1}^{N_{\text{data}}} \mathcal{P}(x_i;\boldsymbol{\theta}), \tag{1}$$

where the fit parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_a, \boldsymbol{\theta}_r)$ comprise the coupling parameters $\boldsymbol{\theta}_a$ for the decay chains and the resonance parameters $\boldsymbol{\theta}_r$. The probability density for an event is the decay intensity normalized by its phase-space integral

$$\mathcal{P}(x;\boldsymbol{\theta}) = \frac{I(x;\boldsymbol{\theta})}{\sigma(\boldsymbol{\theta})}, \qquad \sigma(\boldsymbol{\theta}) = \int I(x;\boldsymbol{\theta})\,\mathrm{d}\Phi, \tag{2}$$

where I is the event intensity, i.e., the squared modulus of the decay amplitude summed over all physical polarizations, and $\mathrm{d}\Phi$ is the phase-space volume element. The normalization integral σ cannot be evaluated analytically and is therefore estimated using a phase-space Monte Carlo sample of N_{phsp} events

$$\sigma(\boldsymbol{\theta}) \approx \frac{1}{N_{\text{phsp}}} \sum_{j=1}^{N_{\text{phsp}}} I_j(\boldsymbol{\theta}). \tag{3}$$

Substituting Eq. (2) into Eq. (1) and taking the negative logarithm yields the negative log-likelihood to be minimized

$$-\ln \mathcal{L}(x;\boldsymbol{\theta}) = -\sum_{i=1}^{N_{\text{data}}} \ln I_i(\boldsymbol{\theta}) \;+\; N_{\text{data}} \ln \sigma(\boldsymbol{\theta}), \tag{4}$$

where the first term evaluates the intensity at the observed events, and the second term normalizes the model to the phase-space integral through the Monte Carlo estimate in Eq. (3). The minimum of Eq. (4) determines $\boldsymbol{\theta}$, namely the couplings and resonance parameters, from the measured four-momenta. The event intensity is defined as the squared total amplitude, summed over all physical polarizations

$$I(x;\boldsymbol{\theta}) = \sum_{\lambda} \left| \sum_{a} \boldsymbol{\theta}_a \mathcal{A}_{\lambda,a}(x;\boldsymbol{\theta}_r) \right|^2 . \tag{5}$$

In the covariant orbital-spin (L–S) tensor formalism [7–9, 11], the two-body vertices in a decay chain are tensors constructed from the four-momenta and polarization tensors of the particles, without requiring chains of Lorentz boosts and rotations to connect the helicity

CPC Accepted

frames of successive decays. For a chain a characterized by quantum numbers $\{L_i, S_i\}$ at each vertex, the amplitude for an event with polarization configuration $\boldsymbol{\lambda} \equiv \{\sigma_1, \sigma_2, \ldots\}$ factorizes into the chain coupling, the propagator $F(m;\boldsymbol{\theta}_r)$ of the resonance r at the invariant mass m of its daughter system, and a kinematic factor that carries the spin structure of the chain

$$\mathcal{A}_{\boldsymbol{\lambda},a}(x;\boldsymbol{\theta}_r) = F(m;\boldsymbol{\theta}_r) \sum_{\sigma_X} [\mathcal{A}_{L_1,S_1}]^{\sigma_2\sigma_X}_{\sigma_1} [\mathcal{A}_{L_2,S_2}]^{\sigma_3\sigma_4}_{\sigma_X} \cdots \quad (6)$$

The kinematic factor, where $[\mathcal{A}_{L,S}]$ denotes the vertex tensors of the chain, depends only on the event four-momenta x and on the quantum numbers fixed by the decay topology, and never on the fit parameters $\boldsymbol{\theta}$. These parameters enter the chain amplitude only through $\boldsymbol{\theta}_a$ and $F(m;\boldsymbol{\theta}_r)$. Therefore, all kinematic factors can be evaluated once before the fit and reused at every iteration. The computational cost of a fit is dominated by evaluating $I(x;\boldsymbol{\theta})$ for every event at every iteration. The next section describes how `CTPWA` organizes this evaluation.

III. COMPUTATIONAL OPTIMIZATION

`CTPWA` implements several optimizations to achieve high-performance partial-wave analysis. To overcome the memory and CPU–GPU communication challenges of conventional frameworks, this section describes two key strategies. First, all parameter-independent kinematic quantities are precomputed and cached once, so device memory usage is fixed before the fit starts, and kinematic factors are not recomputed. Second, the entire likelihood, its analytic derivatives, and the minimization loop are implemented entirely on the GPU, eliminating host–device transfers, kernel-launch and graph-tracing overheads, and per-iteration allocations. As a result, each iteration evaluates only the small parameter-dependent part of the model, and device memory usage remains constant throughout the fit.

A. Precomputation and caching mechanisms

The storage design of `CTPWA` follows directly from the factorization in Eq. (6): all quantities entering the fit are divided into two classes with different lifetimes.

The first class comprises the parameter-independent vertex tensors $\mathcal{A}_{L,S}$, which depend only on the event kinematics and the decay topology. They are evaluated once at initialization from the event four-momenta, stored in contiguous arrays in device memory, and kept resident for the entire fit. Their memory usage is fixed and known before the fit starts. They are used in every iteration and derivative evaluation, and no quantity in this class is ever recomputed or reallocated.

The second class comprises the fit parameters themselves. They are stored as a single small vector of N_θ numbers and updated once per iteration; this vector is the only quantity that changes during minimization.

The two classes interact minimally. When the resonance parameters are fixed, the propagators are constants, and therefore the channel amplitudes are fixed as well: the entire amplitude model for each event reduces to a small matrix computed once, and each iteration requires only updating the coupling vector and multiplying it by the stored matrices. When the resonance parameters float, the vertex tensors are still precomputed once; each iteration recomputes only the scalar propagators $F(m;\boldsymbol{\theta}_r)$, one complex number per event per resonance, and multiplies them by the precomputed tensors, with the couplings applied last: coupling $\times$ precomputed $\mathcal{A}_{L,S} \times F$.

This strict separation is the core of the memory optimization. The dominant memory content is written once at initialization and subsequently only read, and the per-iteration work accesses only the small parameter vector and the current propagators. No allocation occurs during the fit, no kinematic factor is recomputed, and no event data are transferred between the host and the device.

B. Fully GPU-based likelihood computation and minimization

The events in a PWA sample are mutually independent. According to Eq. (4), the negative log-likelihood is a sum of per-event contributions, and the intensity of each event can be evaluated independently, making the likelihood computation embarrassingly parallel. GPUs are a natural platform for this task, but event-by-event amplitude evaluation underutilizes their computational capacity. The arithmetic workload per event is too small to saturate the device, and the computation is dominated by memory latency and branching over decay chains and spins. In contrast, matrix operations are the native workload of GPUs and are performed by highly optimized linear algebra kernels. The design principle of `CTPWA` is therefore to express the likelihood and all its derivatives as batched dense linear algebra over the event dimension, such that each event is an independent row and every computation maps to optimized matrix kernels without per-event branching.

The channel amplitudes for each polarization configuration form a complex vector $\mathcal{A}_{\boldsymbol{\lambda}}$ of length N_{ch}, and the event intensity becomes a quadratic form in the coupling parameters $\boldsymbol{\theta}_a$,

$$I(x;\boldsymbol{\theta}) = \sum_{\boldsymbol{\lambda}} \left| \boldsymbol{\theta}_a \mathcal{A}_{\boldsymbol{\lambda}} \right|^2 = \boldsymbol{\theta}_a^\dagger \mathbf{B} \boldsymbol{\theta}_a, \qquad \mathbf{B} \equiv \sum_{\boldsymbol{\lambda}} \mathcal{A}_{\boldsymbol{\lambda}}^* \mathcal{A}_{\boldsymbol{\lambda}}^\mathsf{T}, \quad (7)$$

where $\mathbf{B}$ is an $N_{\mathrm{ch}} \times N_{\mathrm{ch}}$ Hermitian positive-semidefinite matrix for each event. All kinematic and polarization information for an event is incorporated into $\mathbf{B}$. Thus, in

CPC Accepted

each iteration, the intensity update for the entire data set reduces to a batched matrix–vector product, and the negative log-likelihood in Eq. (4) is obtained from the elementwise logarithms of the resulting intensities. For phase-space normalization, the per-event matrices are averaged once in advance

$$\bar{\mathbf{B}} = \frac{1}{N_{\mathrm{phsp}}} \sum_{j=1}^{N_{\mathrm{phsp}}} \mathbf{B}_j, \tag{8}$$

so that the phase-space contribution to the likelihood and its derivatives require only a single $O(N_{\mathrm{ch}}^2)$ operation per iteration, independent of N_{phsp}.

All derivatives are evaluated from the same stored matrices in a batched pass over events. For derivatives with respect to the coupling parameters $\boldsymbol{\theta}_a$, the data contribution to the gradient of Eq. (4) is

$$\frac{\partial \mathcal{L}_{\mathrm{data}}}{\partial \boldsymbol{\theta}_a} = -\frac{2\mathbf{B}\boldsymbol{\theta}_a}{I}, \tag{9}$$

The division is performed elementwise across events. The phase-space contribution is obtained analogously through $\bar{\mathbf{B}}$.

When the resonance parameters $\boldsymbol{\theta}_r$ are allowed to vary, the derivative of the negative log-likelihood with respect to a specific parameter θ_r of resonance r takes the form

$$\frac{\partial \mathcal{L}_{\mathrm{data}}}{\partial \theta_r} = -2 \sum_{\lambda} \frac{1}{I_\lambda} \mathrm{Re}\left[\mathcal{P}_{r,\lambda} \bar{S}_\lambda \frac{\partial F(m;\boldsymbol{\theta}_r)}{\partial \theta_r} \right], \tag{10}$$

where $\mathcal{P}_{r,\lambda} \equiv \sum_{a \in r} \theta_a [\mathcal{A}_{L,S}]_{a,\lambda}$, with $[\mathcal{A}_{L,S}]_{a,\lambda}$ denoting the precomputed kinematic component of chain a, and $S_\lambda = \sum_a \theta_a \mathcal{A}_{\lambda,a}$ denoting the total amplitude for the polarization configuration $\boldsymbol{\lambda}$. The phase-space normalization contributes an analogous term through the averaged matrices $\bar{\mathbf{B}}$; its closed-form expression is obtained by replacing $\sum_\lambda$ with the phase-space average and differentiating Eq. (3) accordingly. In practice, the inner product $\mathcal{P}_{r,\lambda}$ is independent of the specific parameter θ_r and is computed once per event per resonance. Only the derivative of the scalar propagator, $\partial F/\partial \theta_r$, is evaluated anew using lightweight automatic forward-mode differentiation. This structure allows the full gradient with respect to all resonance parameters to be accumulated in a single batched pass over events, without any per-parameter backward traversal.

The per-event contribution to the real Hessian naturally separates into three blocks according to parameter type: coupling–coupling (aa), coupling–resonance (ar), and resonance–resonance (rr). The coupling–coupling block has the compact analytic form

$$\mathbf{H}_{aa} = -\frac{2\mathbf{B}}{I} + \frac{4}{I^2} (\mathbf{B}\boldsymbol{\theta}_a)(\mathbf{B}\boldsymbol{\theta}_a)^{\mathsf{T}}, \tag{11}$$

which is a scaled copy of $\mathbf{B}$ that has already been formed during the intensity evaluation. The coupling–resonance block is given by

$$\begin{aligned} \mathbf{H}_{ar} = &-\frac{2}{I} \mathrm{Re}\left(\bar{S} \cdot \nabla_{\boldsymbol{\theta}_r} \mathbf{d}S_a^{\mathsf{T}}\right) \\ &+ \frac{4}{I^2} [\mathrm{Re}(\mathbf{B}\boldsymbol{\theta}_a)] \left[\mathrm{Re}\left(\bar{S} \cdot \nabla_{\boldsymbol{\theta}_r} \mathbf{d}S_a^{\mathsf{T}}\right)\right]^{\mathsf{T}}, \end{aligned} \tag{12}$$

where $\mathbf{d}S_a = \partial S / \partial \boldsymbol{\theta}_a$. This block involves only first-order derivatives of the amplitude. For the resonance–resonance block

$$\begin{aligned} \mathbf{H}_{rr} = &-\frac{2}{I} \mathrm{Re}\left(\bar{S} \cdot \nabla_{\boldsymbol{\theta}_r} \nabla_{\boldsymbol{\theta}_r}^{\mathsf{T}} S\right) \\ &+ \frac{4}{I^2} \left[\mathrm{Re}\left(\bar{S} \cdot \nabla_{\boldsymbol{\theta}_r} S\right)\right] \left[\mathrm{Re}\left(\bar{S} \cdot \nabla_{\boldsymbol{\theta}_r} S\right)\right]^{\mathsf{T}}. \end{aligned} \tag{13}$$

The second-derivative term $\nabla_{\boldsymbol{\theta}_r} \nabla_{\boldsymbol{\theta}_r}^{\mathsf{T}} S$ is block-diagonal across resonances because cross terms between different resonances vanish. Thus, only the diagonal sub-blocks corresponding to individual resonances receive second-derivative contributions.

The full Hessian is accumulated in a single batched pass over events. This analytic construction avoids the $O(N_{\mathrm{params}})$ backward passes required by automatic differentiation and makes Hessian-based uncertainty estimation practical even for high-statistics data sets.

In conventional pipelines, minimization is performed on the CPU, while likelihood evaluation is offloaded to the GPU. At each iteration, the events or recomputed amplitudes are transferred to the device, and the resulting likelihood value is copied back to the host. Consequently, data and intermediate results repeatedly cross the host–device boundary, and transfer time often dominates the iteration cost. In contrast, `CTPWA` performs the entire optimization on the GPU. The negative log-likelihood, its analytic gradient, and its Hessian are exposed directly to the L-BFGS optimizer [10] in PyTorch [5], which runs on the device. All precomputed tensors and resident data quantities remain in GPU memory throughout the fit. No computational graph is built during optimization, because the computationally intensive derivatives are analytic and only the scalar propagator functions use lightweight forward-mode automatic differentiation. The analytic gradient has been cross-checked against PyTorch autograd on the same likelihood and agrees to machine precision.

CPC Accepted

IV. PERFORMANCE AND DEMONSTRATION

A. Computational performance

To quantify the performance of the proposed method, we benchmark it using synthetic PWA samples with $N_{\rm ch} = 10$ decay channels, $N_{\rm pol} = 3$ polarization configurations, and $N_{\rm phsp} = 10 \times N_{\rm data}$ phase-space events, in double precision. Three approaches are compared: the full amplitude calculation on the CPU within PyTorch [5], the same PyTorch-based computation on the GPU, and the optimized formulation, in which the kinematic amplitudes are precomputed once and the gradient and Hessian are obtained analytically. This comparison separates the impact of hardware acceleration from the algorithmic improvements. All timings are averaged over repeated evaluations after warm-up.

Figure 1 compares the per-iteration time of the conventional pipeline with that of the precomputed method for a typical 4-body decay topology. The per-iteration cost of the conventional approach increases linearly with the data size, from about 4 ms at 10^2 events to about 1.2 s at 10^6 events. By contrast, the precomputed pipeline remains essentially constant at about 0.5–3.5 ms, because only batched matrix–vector operations remain. Including the one-time precomputation, a complete fit of 10^3 iterations is accelerated by a factor that increases from about 8 at $N_{\rm data} = 10^2$ to more than two orders of magnitude at $N_{\rm data} = 10^6$. The memory usage shows a similarly pronounced advantage. Beyond $N_{\rm data} = 10^6$, the conventional pipeline exhausts the device memory, whereas the precomputed representation completes without difficulty.

Because the matrix formulation in Sec. III evaluates the likelihood and its gradient using batched matrix operations, the optimized formulas compute the NLL and gradient for 10^6 events in approximately 4 ms, compared with 45 ms for GPU autograd and 2.5 s for CPU autograd, as shown in Fig. 2. The speedup increases with data size: at 10^5 events, it is about 9-fold relative to GPU autograd and about 500-fold relative to CPU autograd, increasing to about 11-fold and about 660-fold, respectively, at 10^6 events. This favourable scaling reflects the weak dependence of the batched matrix–vector operations on $N_{\rm data}$, in contrast to the per-event overhead of graph-based differentiation.

The greatest performance gain occurs in the Hessian evaluation, as shown in Fig. 3. For 10^6 events, the analytic Hessian of Sec. 3 requires approximately 13 ms for the full 20×20 matrix, compared with 1.4 s for GPU autograd and 81 s for CPU autograd, corresponding to speedup factors of approximately 101 and 6×10^3, respectively. The single-pass accumulation replaces the $O(N_{\rm params})$ backward passes required by automatic differentiation, making Hessian-based uncertainty estimation practical at high statistics.

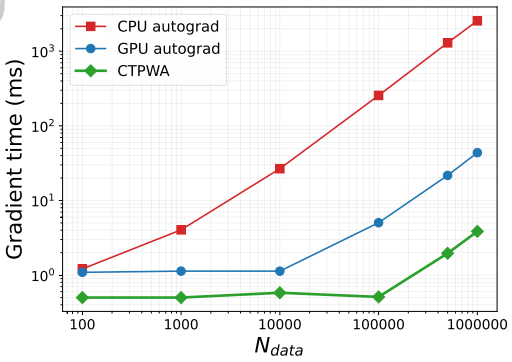


Fig. 2. (color online) NLL+gradient evaluation time as a function of $N_{\rm data}$. The red, blue, and green curves correspond to CPU-based PyTorch autograd, GPU-based PyTorch autograd, and the analytic formulation implemented in `CTPWA`, respectively. The analytic formulation represented by the green curve is consistently faster by one to two orders of magnitude.

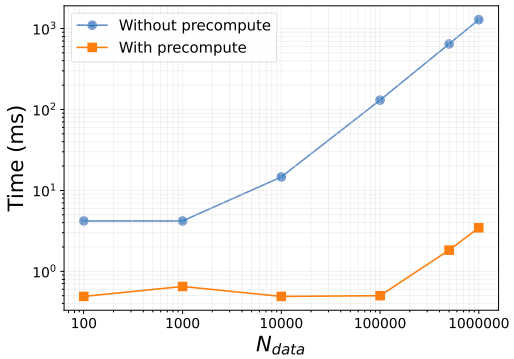


Fig. 1. (color online) Per-iteration time as a function of $N_{\rm data}$, comparing recomputing amplitudes at each iteration with the precomputed method. The runtime of the precomputed method is nearly independent of the data size, whereas that of the conventional pipeline increases linearly and eventually exhausts the GPU memory.

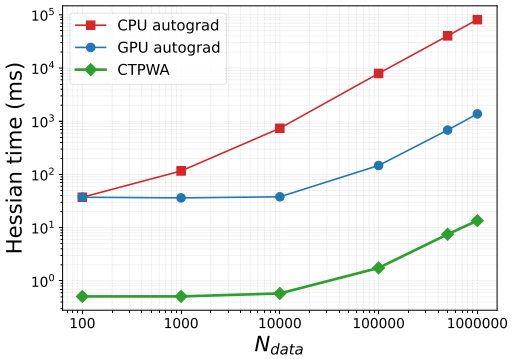


Fig. 3. (color online) Hessian evaluation time as a function of $N_{\rm data}$. The red, blue, and green curves correspond to CPU-based PyTorch autograd, GPU-based PyTorch autograd, and the analytic formulation of `CTPWA`, respectively. The analytic method is faster by two to three orders of magnitude across the full range.

CPC Accepted

B. Fit demonstration with $J/\psi \to K^+K^-\eta$

As a demonstration of the framework in a realistic setting, we perform a fit to the decay $J/\psi \to K^+K^-\eta$ with a sample of one million toy Monte Carlo events. This three-body decay probes intermediate resonances in both the K^+K^- and $K\eta$ subsystems, making it a representative test case for modern PWA studies. The amplitude model includes 16 intermediate resonances selected from the PDG [1] with 33 complex couplings, and the phase-space integral is evaluated with 5×10^6 uniformly generated Monte Carlo events. The kinematic amplitudes are precomputed once in about 94 s and stored in GPU memory. We consider two scenarios: (i) resonance masses and widths fixed to their input values, and (ii) ten resonance parameters allowed to float together with the couplings.

Table 1 reports the measured wall-clock time for both scenarios. In the case with fixed resonance parameters, the per-iteration cost is dominated by the batched matrix–vector products in Eqs. (7) and (9), and the full fit is completed in about 7 s after about 500 iterations. When resonance parameters are floated, the phase-space pre-averaging in Eq. (8) no longer applies, and the gradient must be recomputed for each event, increasing the per-iteration cost to about 100 ms. The fit still converges in about 100 s after about 1000 iterations. The Hessian evaluation time increases from about 0.35 s to about 1.6 s due to the additional resonance-parameter blocks.

Table 1. Measured wall-clock time and device memory usage for the $J/\psi \to K^+K^-\eta$ fit with $N_{\rm data} = 10^6$ and $N_{\rm phsp} = 5\times10^6$. The two columns correspond to the cases in which the resonance parameters are held fixed and allowed to float, respectively.

Stage	Fixed resonance params	Float resonance params
Amplitude initialization	~94 s	~94 s
NLL + gradient per iteration	~15 ms	~100 ms
Full Hessian	~0.35 s	~1.6 s
Device memory	~20 GB	~32 GB
Complete single fit	~7.0 s	~100 s

Figure 4 shows the invariant-mass projections of the best-fit model superimposed on the signal Monte Carlo sample. The model accurately reproduces the resonant structures across the full kinematic range. The residuals are consistent with statistical fluctuations, and the fitted parameters agree with the generated input values. A complete high-statistics amplitude fit can thus be completed within minutes, a performance level that readily meets the typical requirements of BESIII partial-wave analyses.

V. SUMMARY

We have developed `CTPWA`, a GPU-oriented partial wave analysis framework for high-statistics hadron spectroscopy. It addresses the main limitations of conventional unbinned maximum-likelihood fits, in which repeated event-by-event amplitude evaluation incurs substantial computational time and memory requirements, especially for samples of 10^5–10^6 events and models with many interfering amplitudes.

The method exploits the factorization of the covariant tensor amplitude formalism. The kinematic tensors depend only on the event four-momenta and decay topology and are therefore independent of the fit parameters. They are precomputed once, stored in device memory, and reused throughout the fit, while only the couplings and, when allowed to float, the resonance propagators are updated during minimization. The likelihood is rewritten in matrix form, enabling the negative log-likelihood, gradient, and Hessian to be evaluated using batched dense linear algebra on the GPU. The dominant derivatives are obtained analytically, avoiding the overhead of graph-based automatic differentiation, and the full minimiza-

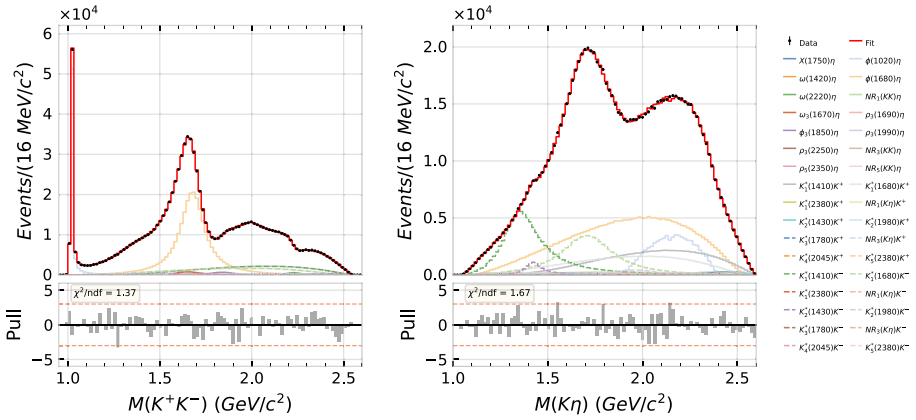


Fig. 4. (color online) Fit results for $J/\psi \to K^+K^-\eta$. The data points represent the signal Monte Carlo sample, and the solid histogram shows the projection of the best-fit model. The left and right panels show $M(K^+K^-)$ and $M(K\eta)$, respectively.

CPC Accepted

tion loop is executed on the device.

The framework demonstrates substantial gains in both speed and memory efficiency. For synthetic samples with 10^6 events, the optimized implementation evaluates the negative log-likelihood and gradient in a few milliseconds, approximately one order of magnitude faster than GPU autograd and more than three orders of magnitude faster than CPU autograd. The analytic Hessian is approximately two orders of magnitude faster than GPU autograd and more than four orders of magnitude faster than CPU autograd. The precomputed representation also avoids the memory exhaustion encountered in conventional iterative amplitude recomputation. To further validate the framework in a realistic application, we fit $J/\psi \to K^+K^-\eta$ using Monte Carlo samples of 10^6 signal events and 5×10^6 phase-space events. A fit with fixed resonance parameters converges within seconds, while a fit with floating resonance parameters completes in about 10^2 seconds. The fitted projections reproduce the generated distributions well, and the extracted parameters are consistent with the input values.

These results demonstrate that `CTPWA` provides an efficient solution to the computational challenges of large-scale PWA at modern experiments such as BESIII and LHCb. In addition, this framework is expected to improve substantially the efficiency of resonance searches, precision measurements, and systematic model comparisons and to provide a practical basis for AI-assisted studies in hadron spectroscopy in the near future.

ACKNOWLEDGMENTS

We thank Prof. Haojie Jing and Prof. Jiajun Wu for useful discussions.

PROGRAM AVAILABILITY

The `CTPWA` framework described in this paper is publicly available at https://github.com/BHXiang/ctpwa.

References

[1] S. Navas *et al*. (Particle Data Group), Phys. Rev. D **110**, 030001 (2024)
[2] F. Gross, E. Klempt, S. J. Brodsky *et al*., Eur. Phys. J. C **83**, 1125 (2023)
[3] B. Liu (BESIII), J. Phys. Conf. Ser. **523**, 012008 (2014)
[4] Y. Jiang, Nuovo Cim. C **47**, 203 (2024)
[5] A. Paszke, S. Gross, F. Massa *et al*., arXiv: 1912.01703
[6] M. Abadi, P. Barham, J. Chen *et al*., in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (USENIX Association, Berkeley, CA, 2016) p. 265
[7] B. S. Zou and D. V. Bugg, Eur. Phys. J. A **16**, 537 (2003)
[8] H. J. Jing, D. Ben, S. M. Wu *et al*., JHEP **06**, 039 (2023)
[9] H. J. Jing, S. M. Wu, and J. J. Wu, Phys. Rev. D **110**, 016014 (2024)
[10] D. C. Liu and J. Nocedal, Math. Program. **45**, 503 (1989)
[11] X. Y. Li, X. K. Dong, and H. J. Jing, Nucl. Phys. A **1040**, 122761 (2023)

CPC Accepted