PyPWA: A TOOLKIT FOR PARAMETER OPTIMIZATION AND AMPLITUDE ANALYSIS

Mark Jones

Carlos Salgado (NSU/Jlab), Mark Jones (NSU/Jlab), William Phelps (CNU/Jlab), Peter Hurck (University of Glasgow)

Outline

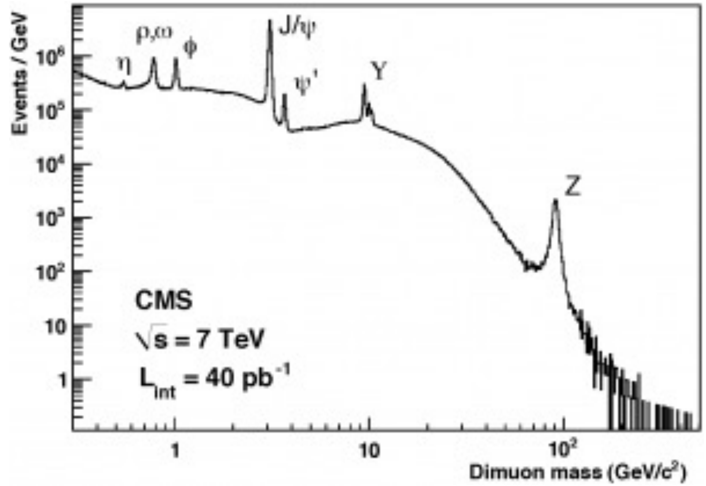
- Amplitude Analysis
- Software and Parallel Design
- Optimization
- Scaling on CPUs and GPUs
- PWA Examples
- Installing PyPWA
- Summary and Ongoing Work

Amplitude Analysis

- In order to identify hadrons we need to determine their quantum numbers: using

  Amplitude Analysis / Partial Wave Analysis (PWA)
- In PWA, the intensity is expanded in partial waves defined by the angular components (angular quantum numbers)
- PyPWA software is a flexible and modular toolkit used to define any type of amplitude and represent data by any set of variables
- PyPWA performs parameter optimization and generation of modeled or simulated data.

(Intensity)



For example:

Amplitude

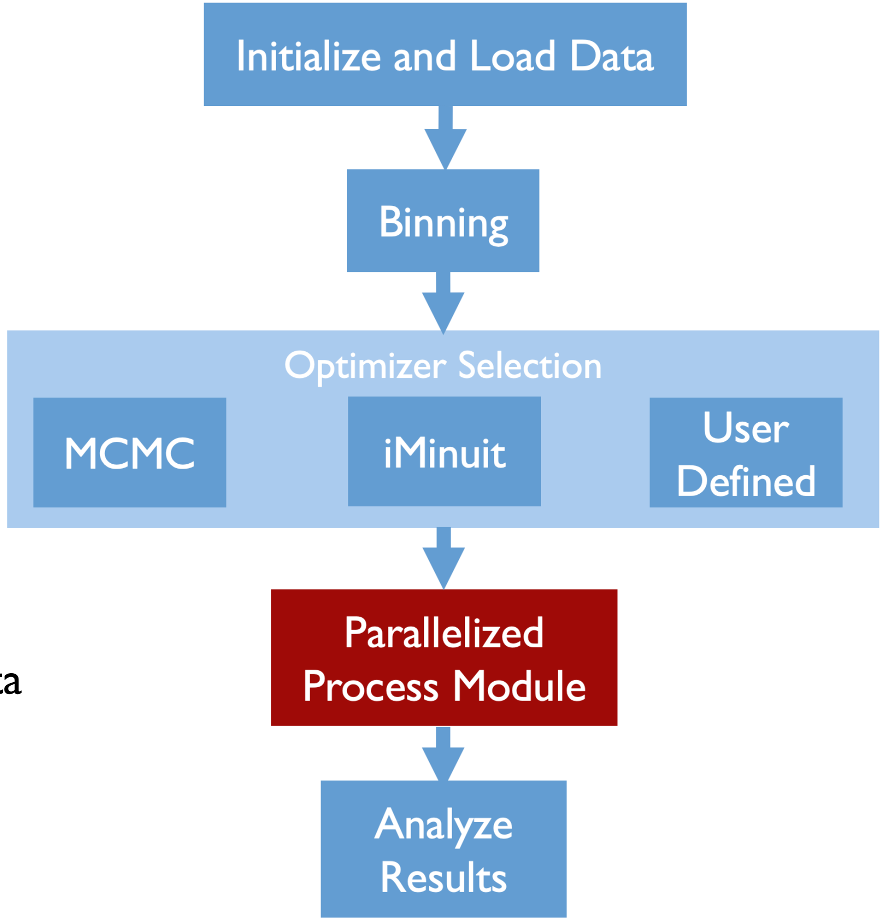
$$I(\Omega) = \sum_k \sum_{\epsilon_R} \sum_{l,|m|,l',|m'|} {}^{\epsilon_R}Y_l^{|m|}(\Omega)\, {}^{\epsilon_R}V^k_{l,|m|}\, {}^{\epsilon_R}V^{k*}_{l',|m'|}\, {}^{\epsilon_R}Y_{l'}^{|m'|*}(\Omega)$$

(Intensity)

Fitted

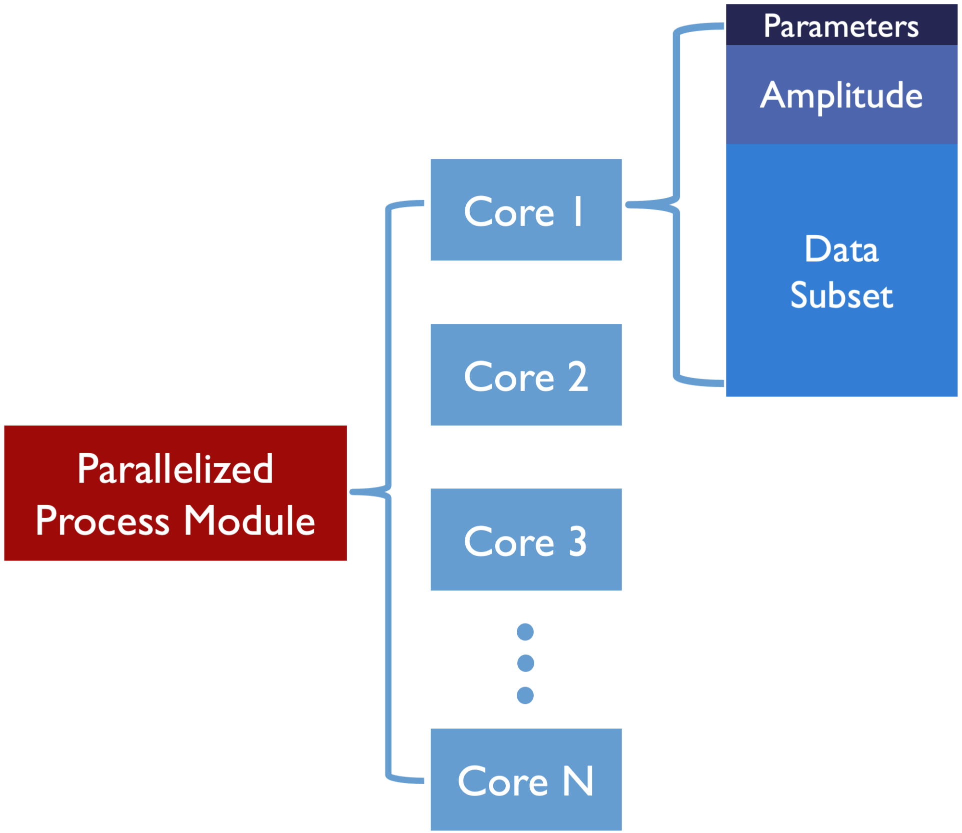
Software Design

- PyPWA: flexible set of tools for fitting multi-dimensional models and generating simulations
  - Object-oriented design for data structures and components with runtime state or data plugins
  - Functional design for the remaining package
- Two main components: data processing and data analysis
  - Data processing: libraries for parsing, masking, binning and operating on data
  - Data analysis: tools for developing likelihoods, amplitudes, fitting, and visualization



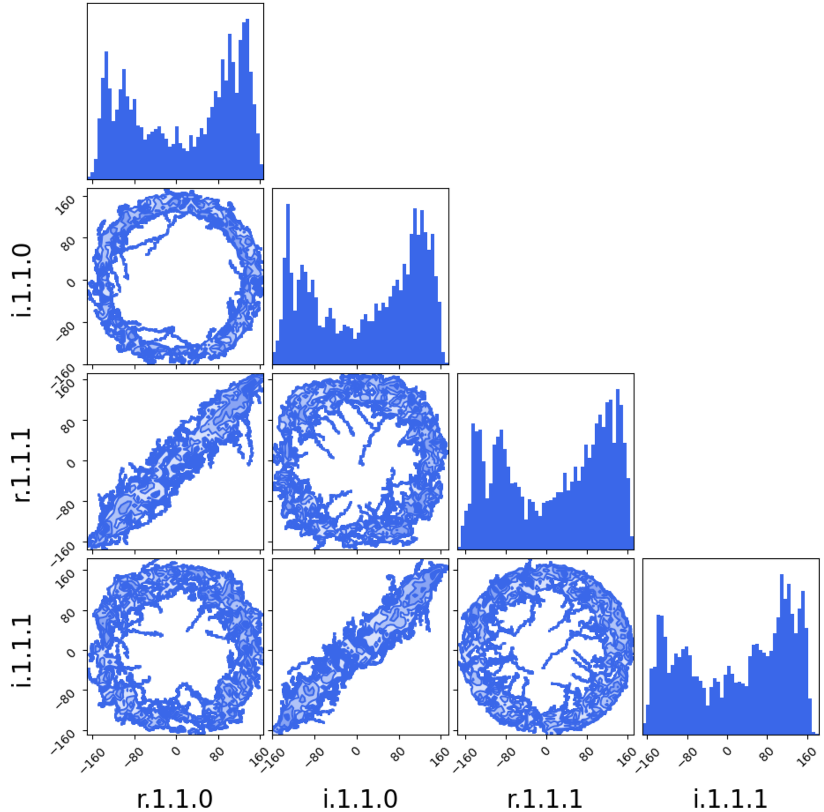
Parallel Design

- Bypasses Global Interpreter Lock (GIL) limitation using multiprocessing module
  - Implements multiprocessing by inheriting from Process class from the Multiprocessing module
  - Scales kernel and dataset across available hardware threads
  - Communication object enables exchange of information between parent and child processes
- Multi-GPU support through multithreading for compatibility with CUDA
- High scalability across hardware resources, built directly into PyPWA's Likelihoods



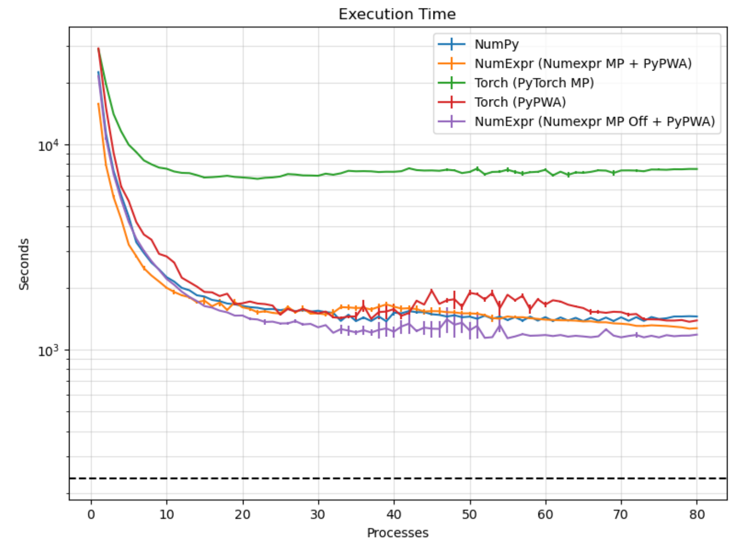
Optimization

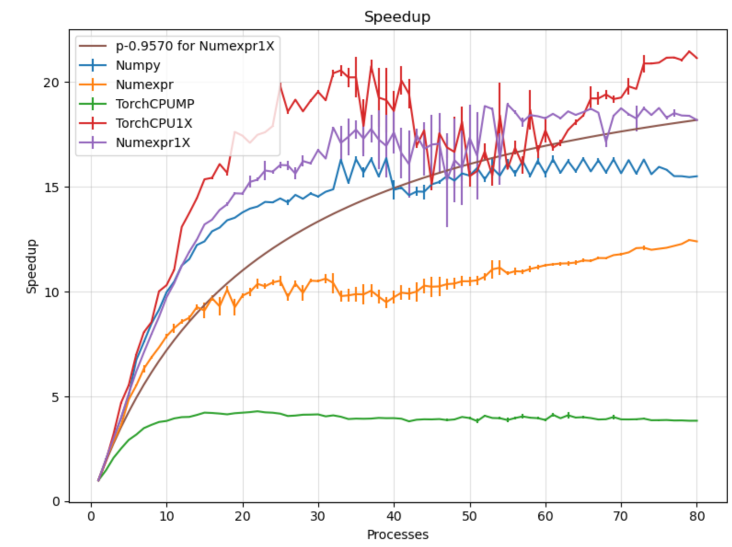
- Minimize loss functions for model parameters
- Built-in support for optimizers:
  - iMinuit:
    - Python implementation of MINUIT2
    - MIGRAD, HESSE, and MINOS algorithms
  - emcee (MCMC):
    - Parameter estimation via MCMC
    - Ensemble sampling with multiple chains
  - Likelihood objects can be called as if they were a standard function, allowing for support for most Python optimizers.
- Analyze results for correlations and best parameters



Scaling on CPUs

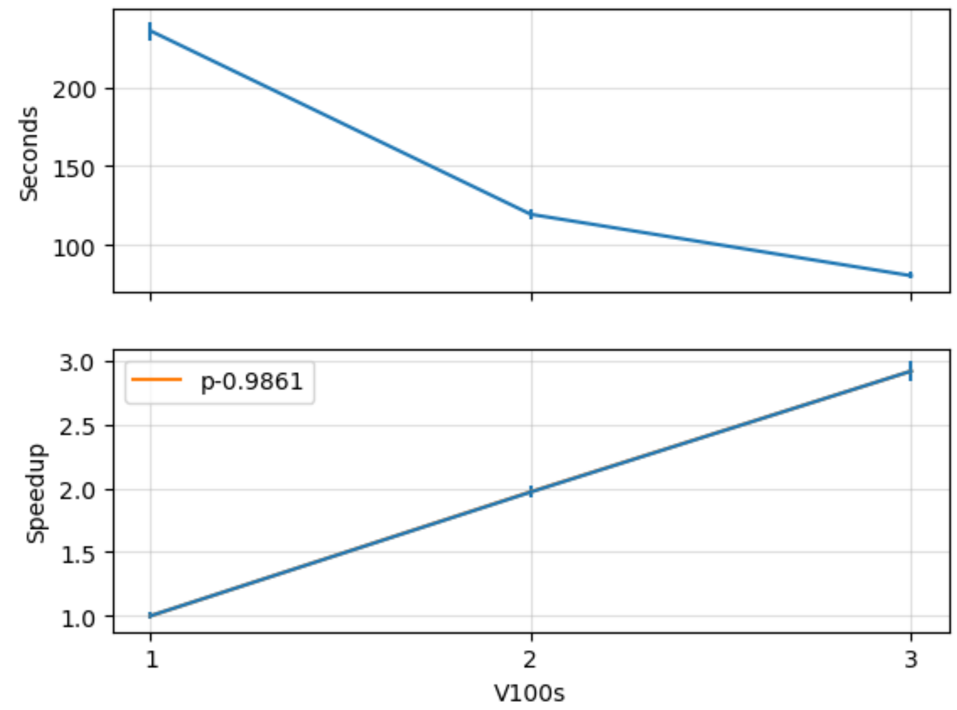
- PyPWA demonstrates excellent scalability on multi-core CPUs
  - NumExpr library locked to a single thread with PyPWA processing module provides the best performance
    - Numexpr with default threading still outperforms pure Numpy and PyToch amplitudes.
  - PyPWA processing module outperforms PyTorch OpenMP implementation.





Scaling on GPUs

- Near perfect scaling on GPUs
- Utilizes PyTorch's Tensors for math operations
  - Amplitudes remain in Python, no C or C++ required.
- Leverages multithreading to remain compatible with CUDA

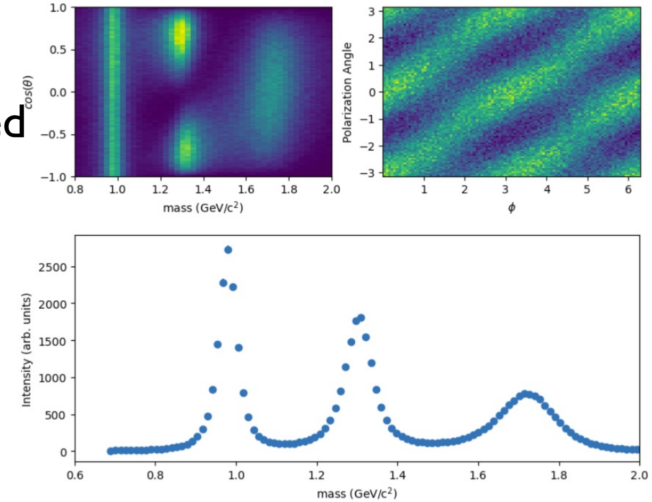


PVVA Example

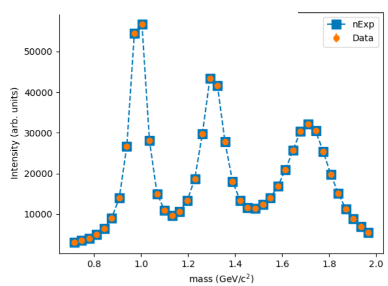
- Eta pi photoproduction
  - Extract resonances and associated quantum numbers with Mass-independent partial wave analysis, using iMinuit for extended log-likelihood fit
- Results:
  - Good agreement between fitted values and simulated data
  - Successfully extracted input resonances and waves
- Figures:
  - Generated mass distribution and angular distributions
  - Fitted intensities vs mass (total and for different waves)

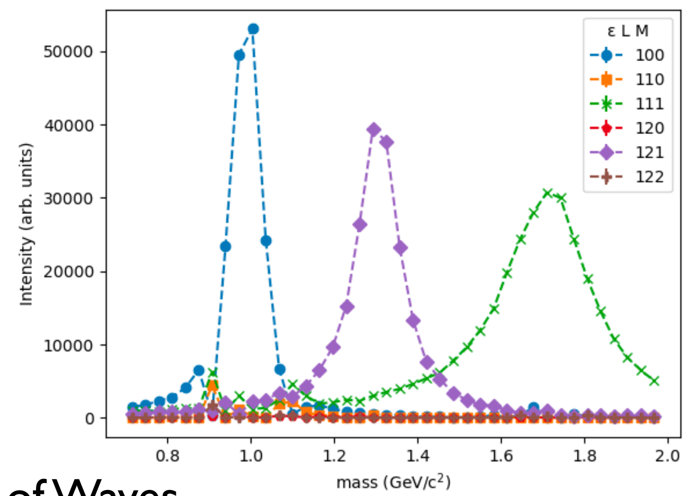
$$I(\theta, \phi, \mathscr{P}, \Phi) = I^{(0)}(\theta, \phi) - \mathscr{P} I^{(1)}(\theta, \phi) \cos 2\Phi \quad - \mathscr{P} I^{(2)}(\theta, \phi) \sin 2\Phi$$

Generated



Fit





Mix of Waves

Installing PyPWA

- Available on MacOS (x86/Arm64) and Linux (x86)
- PIP:

```
> pip install git+https://github.com/JeffersonLab/PyPWA.git
```

- Anaconda:

```
> conda install -c markjonestx pypwa
```

Summary and Ongoing Work

- PyPWA offers a flexible toolkit for amplitude analysis in multi-particle final states within the Python ecosystem
  - Users can utilize various independent components to solve a range of optimization problems
  - Supports parallel processing and GPU acceleration with PyTorch for improved performance
  - User-friendly installation on Linux and MacOS with Anaconda, and extensive support from the Python community
- Future developments can focus on further optimization, expanding capabilities, and we are incorporating new AI technologies to enhance performance and user experience

PyPWA Links

- Github: https://github.com/JeffersonLab/PyPWA
- ReadTheDocs: https://pypwa.readthedocs.io/en/main/
- Web page: https://pypwa.jlab.org