

# The ComPWA Project

Facilitating and automating amplitude analysis  
with modern Python tools  
and transparent, interactive documentation

**Remco de Boer**  
Ruhr University Bochum

**8 September 2021**  
PWA 12 / ATHOS 7  
held at the University of Bristol

# What is the ComPWA project?



[github.com/ComPWA](https://github.com/ComPWA)

- “Common Partial Wave Analysis”
- Open-source GitHub organization
- Originally a C++ framework ([ComPWA](#))
- Now maintains a collection of Python tools for amplitude analysis
- Developer group at Ruhr University Bochum, JGU Mainz, and GSI
- So far developed in the context of BESIII and PANDA analyses

# What does ComPWA aim for?

- **Academic continuity:**  
long-term, collaboration-independent  
PWA software development
- Provide an easy starting point  
for researchers new to the field of PWA
- Build up modern, interlinked, and  
**interactive PWA knowledge-bases**
- Maintain libraries that facilitate and automate  
common procedures in amplitude analysis

⇒ Narrow the gap between theory and code

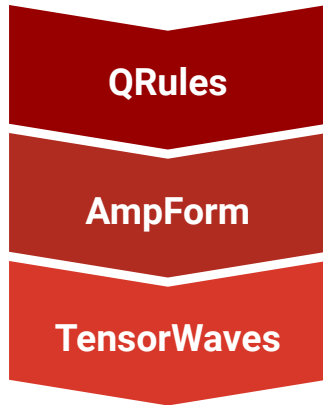
⇒ Bring usage and development closer together

The screenshot shows the API documentation for the `CoupledWidth` class. At the top, there is a search bar and navigation icons. The class signature is: `class CoupledWidth(s: Symbol, mass0: Symbol, gamma0: Symbol, m_a: Symbol, m_b: Symbol, angular_momentum: Symbol, meson_radius: Symbol, phsp_factor: Optional[PhaseSpaceFactorProtocol] = None, name: Optional[str] = None, evaluate: bool = False) [source]`. Below the signature, it lists the base class: `Bases: ampform.sympy.UnevaluatedExpression`. A description follows: "Mass-dependent width, coupled to the pole position of the resonance." It references PDG2020, §Resonances, p.6 and [11], equation (6), and states that the default value for `phsp_factor` is `PhaseSpaceFactor()`. A note explains that the `BlattWeisskopfSquared` of `AmpForm` is normalized in the sense that equal powers of  $z$  appear in the nominator and the denominator, while the definition in the PDG (as well as some other sources), always have 1 in the nominator of the Blatt-Weisskopf. In that case, one needs an additional factor  $(q/q_0)^{2L}$  in the definition for  $\Gamma(m)$ . It then states: "With that in mind, the 'mass-dependent' width in a `relativistic_breit_wigner_with_ff` becomes:" followed by the equation: 
$$\Gamma_0(s) = \frac{\Gamma_0 B_L^2(q^2(s)) \rho(s)}{B_L^2(q^2(m_0^2)) \rho(m_0^2)} \quad (3)$$
 where  $B_L^2$  is defined by (1),  $q$  is defined by (2), and  $\rho$  is (by default) defined by (4). At the bottom, there is a search bar containing the text `phsp_factor`.

Screenshot from the API of one of ComPWA's packages 3

# What do we provide?

Three main Python packages that together cover a full amplitude analysis:



**QRules**

Automated quantum number conservation rules

**AmpForm**

Formulate symbolic model templates

**TensorWaves**

Fit models to data and generate data samples  
with multiple computational back-ends

All are designed as **libraries**, so they can be used by other packages

# QRules

Automated quantum number conservation rules

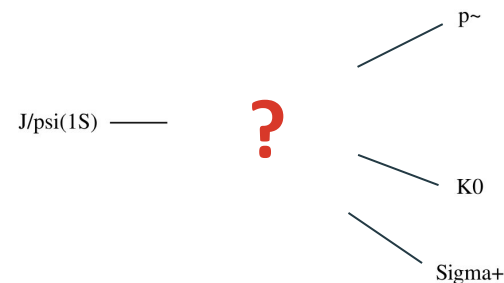
**Aim:** compute which particle reactions are allowed  
between a given initial and final state

# QRules

## Automated quantum number conservation rules

**Aim:** compute which particle reactions are allowed between a given initial and final state

1. User specifies some boundary conditions  
(particle names, allowed interactions, isobar model, etc.)

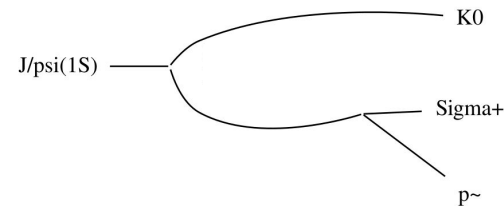
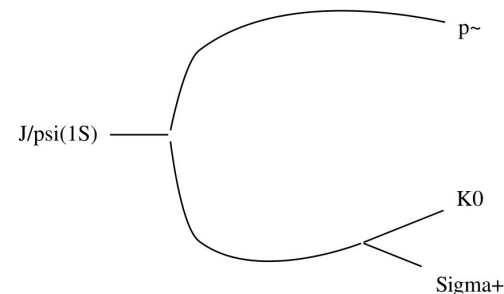
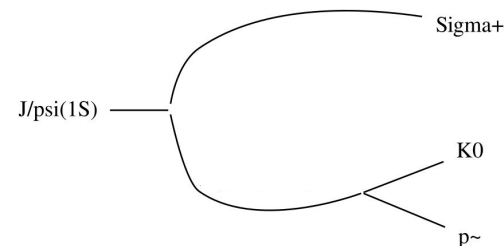


# QRules

## Automated quantum number conservation rules

**Aim:** compute which particle reactions are allowed between a given initial and final state

1. User specifies some boundary conditions  
(particle names, allowed interactions, isobar model, etc.)
2. QRules then:
  - gets corresponding particle properties from the PDG (or any custom definitions),
  - determines all possible decay topologies,



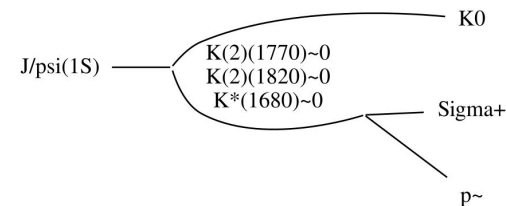
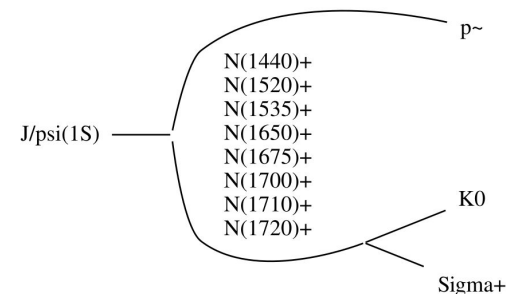
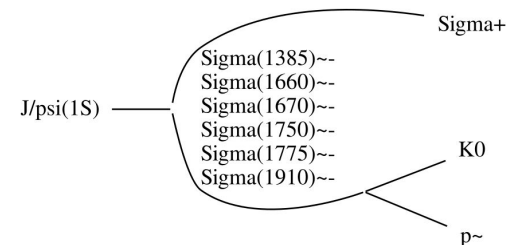
# QRules

## Automated quantum number conservation rules

**Aim:** compute which particle reactions are allowed between a given initial and final state

1. User specifies some boundary conditions (particle names, allowed interactions, isobar model, etc.)
2. QRules then:
  - gets corresponding particle properties from the PDG (or any custom definitions),
  - determines all possible decay topologies,
  - propagates quantum numbers through intermediate edges,
  - and selects all allowed transitions with its conservation laws

Generalized approach: **the constraints ‘span’ quantum number space**





# QRules

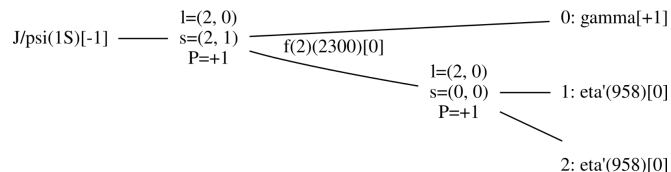
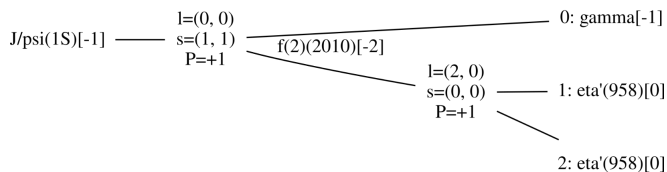
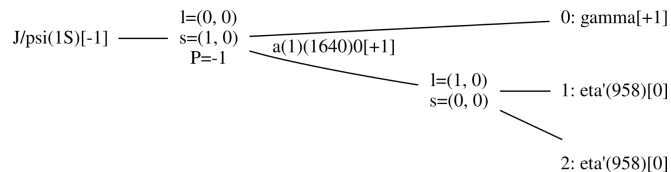
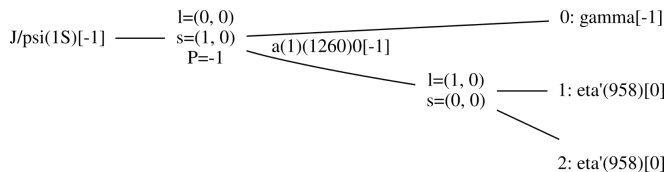
## Automated quantum number conservation rules

The returned objects contain **all information to build an amplitude model!**

```
reaction = grules.generate_transitions(
    initial_state="J/psi(1S)",
    final_state=["gamma", "eta'(958)", "eta'(958)"],
)
```

Propagating quantum numbers: 100%

36/36 [00:04<00:00, 7.55it/s]



# QRules

## Automated quantum number conservation rules

The library also provides several related features:

Check which conservation rules are violated:

```
grules.check_reaction_violations(  
    initial_state="pi0",  
    final_state=["gamma", "gamma", "gamma"],  
)
```

```
{frozenset({'c_parity_conservation'})}
```

Find particles by selecting quantum numbers:

```
selection = PDG.filter(lambda p: p.spin > 0 and p.charmness and p.mass > 2.82)  
selection.names
```

```
['Lambda(c)(2880)--', 'Lambda(c)(2880)+', 'Xi(c)(2815)0', 'Xi(c)(2815)~0']
```

Get particle properties:\*

```
PDG = grules.load_pdg()  
PDG.find("f(0)(980)")
```

```
Particle(  
    name='f(0)(980)',  
    pid=9010221,  
    latex='f_{0}(980)',  
    spin=0.0,  
    mass=0.99,  
    width=0.06,  
    isospin=Spin(0, 0),  
    parity=+1,  
    c_parity=+1,  
    g_parity=+1,  
)
```

\*PDG info computed from the  
scikit-hep [particle](#) package

# AmpForm

## Symbolic amplitude model formulation

- Implements **spin formalisms and dynamics**
- Can express QRules' state transitions as an amplitude model
- Amplitude models are formulated as algebraic expressions (SymPy CAS)
- User can further modify the expressions
- The models serve as a **mathematical template for fitter packages**

```
n = Symbol("n_R")
matrix = RelativisticKMatrix.formulate(
    n_channels=1,
    n_poles=n,
)
matrix[0, 0]
```

$$\rho(s) \sum_{R=1}^{n_R} \frac{\Gamma(s)\gamma_{R,0}^2 m_R}{-s+m_R^2}$$
$$-i\rho(s) \sum_{R=1}^{n_R} \frac{\Gamma(s)\gamma_{R,0}^2 m_R}{-s+m_R^2} + 1$$

```
matrix = NonRelativisticKMatrix.formulate(
    n_poles=1,
    n_channels=2,
).doit()
matrix[0, 0].simplify()
```

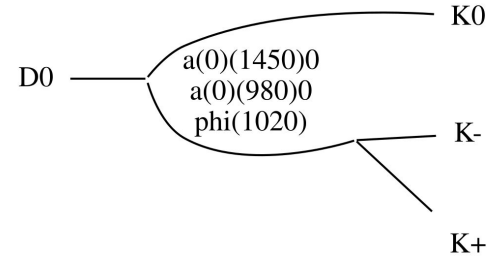
$$\frac{\Gamma_{1,0}\gamma_{1,0}^2 m_1}{s + i\Gamma_{1,0}\gamma_{1,0}^2 m_1 + i\Gamma_{1,1}\gamma_{1,1}^2 m_1 - m_1^2}$$

# AmpForm

## Symbolic amplitude model formulation

### Example

Building an amplitude model for  $D \rightarrow K^0 K^- K^+$   
with three resonances



```
builder = ampform.get_builder(reaction)
for p in reaction.get_intermediate_particles():
    builder.set_dynamics(p.name, create_relativistic_breit_wigner_with_ff)
model = builder.formulate()
```

$$\left| A_{D_0^0 \rightarrow K_0^0 \phi(1020)_0; \phi(1020)_0 \rightarrow K_0^+ K_0^-} + A_{D_0^0 \rightarrow K_0^0 a_0(1450)_0; a_0(1450)_0 \rightarrow K_0^+ K_0^-} + A_{D_0^0 \rightarrow K_0^0 a_0(980)_0; a_0(980)_0 \rightarrow K_0^+ K_0^-} \right|^2$$

- User selects Breit-Wigner to parametrize each resonance
- AmpForm takes care of spin (helicity formalism)
- Resulting amplitude model expressed symbolically

*LaTeX generated  
by the code!*

# AmpForm

## Symbolic amplitude model formulation

Expression for the amplitude model can be further inspected:

```
some_amplitude = model.components[
    R"A_{D^0_{-0}} \to K^+_{-0} a_{-0}(980)^{0}_{-0}; a_{-0}(980)^{0}_{-0} \to K^+_{+0} K^-_{-0}"
]
```

$$\begin{aligned}
 & \frac{C_{D^0 \rightarrow K_0^0 a_0(980)_0^0; a_0(980)^0 \rightarrow K_0^+ K_0^-} \Gamma_{a_0(980)^0} m_{a_0(980)^0} \sqrt{B_0^2 \left( (d_{a_0(980)^0})^2 q_{122}^2 (m_{12}^2) \right)} D_{0,0}^0(-\phi_{1+2}, \theta_{1+2}, 0) D_{0,0}^0(-\phi_{1,1+2}, \theta_{1,1+2}, 0)}{-m_{12}^2 + (m_{a_0(980)^0})^2 - im_{a_0(980)^0} \Gamma(m_{12}^2)} \\
 &= \frac{C_{D^0 \rightarrow K_0^0 a_0(980)_0^0; a_0(980)^0 \rightarrow K_0^+ K_0^-} \Gamma_{a_0(980)^0} m_{a_0(980)^0}}{i \Gamma_{a_0(980)^0} (m_{a_0(980)^0})^2 \sqrt{\frac{(m_{12}^2 - (m_1 - m_2)^2)(m_{12}^2 - (m_1 + m_2)^2)}{m_{12}^2}} - m_{12}^2 + (m_{a_0(980)^0})^2} \\
 & \quad - \frac{m_{12} \sqrt{\frac{\left( (m_{a_0(980)^0})^2 - (m_1 - m_2)^2 \right) \left( (m_{a_0(980)^0})^2 - (m_1 + m_2)^2 \right)}{(m_{a_0(980)^0})^2}}}{0.074} \\
 &= \frac{0.074}{-m_{12}^2 + 0.96 - \frac{0.599 \sqrt{m_{12}^2 - 0.975}}{m_{12}}}
 \end{aligned}$$

*Dynamics*

*Spin formalism*

# AmpForm

## Symbolic amplitude model formulation

Expression for the amplitude model can be further inspected:

```
some_amplitude = model.components[
    R"A_{D^0_{\to} K^0_{a_0(980)^0} \to K^+_{a_0(980)^0} a_{\to} K^+_{a_0(980)^0} a_{\to} K^+_{a_0(980)^0} K^+_{a_0(980)^0}"
]
```

$$\frac{C_{D^0 \rightarrow K_0^0 a_0(980)^0; a_0(980)^0 \rightarrow K_0^+ K_0^-} \Gamma_{a_0(980)^0} m_{a_0(980)^0} \sqrt{B_0^2 \left( (d_{a_0(980)^0})^2 q_{122}^2 (m_{12}^2) \right)} D_{0,0}^0(-\phi_{1+2}, \theta_{1+2}, 0) D_{0,0}^0(-\phi_{1,1+2}, \theta_{1,1+2}, 0)}{-m_{12}^2 + (m_{a_0(980)^0})^2 - im_{a_0(980)^0} \Gamma(m_{12}^2)}$$

$$= \frac{C_{D^0 \rightarrow K_0^0 a_0(980)^0; a_0(980)^0 \rightarrow K_0^+ K_0^-} \Gamma_{a_0(980)^0} m_{a_0(980)^0}}{i \Gamma_{a_0(980)^0} (m_{a_0(980)^0})^2 \sqrt{\frac{(m_{12}^2 - (m_1 - m_2)^2)(m_{12}^2 - (m_1 + m_2)^2)}{m_{12}^2}}} - m_{12}^2 + (m_{a_0(980)^0})^2$$

$$= \frac{m_{12} \sqrt{\frac{\left( (m_{a_0(980)^0})^2 - (m_1 - m_2)^2 \right) \left( (m_{a_0(980)^0})^2 - (m_1 + m_2)^2 \right)}{(m_{a_0(980)^0})^2}}}{0.074}$$

$$= \frac{0.074}{-m_{12}^2 + 0.96 - \frac{0.599 \sqrt{m_{12}^2 - 0.975}}{m_{12}}}$$

← CAS performs algebraic simplifications

# AmpForm

## Symbolic amplitude model formulation

Expression for the amplitude model can be further inspected:

```
some_amplitude = model.components[
    R"A_{D^0_{\to} K^0_{a_0}(980)^0; a_0(980)^0 \to K^+_0 K^-_0} \Gamma_{a_0(980)^0} m_{a_0(980)^0} \sqrt{B_0^2 \left( (d_{a_0(980)^0})^2 q_{122}^2 (m_{12}^2) \right)} D_{0,0}^0(-\phi_{1+2}, \theta_{1+2}, 0) D_{0,0}^0(-\phi_{1,1+2}, \theta_{1,1+2}, 0)
    ]
```

$$\begin{aligned}
 & \frac{C_{D^0 \to K^0_{a_0}(980)^0; a_0(980)^0 \to K^+_0 K^-_0} \Gamma_{a_0(980)^0} m_{a_0(980)^0} \sqrt{B_0^2 \left( (d_{a_0(980)^0})^2 q_{122}^2 (m_{12}^2) \right)} D_{0,0}^0(-\phi_{1+2}, \theta_{1+2}, 0) D_{0,0}^0(-\phi_{1,1+2}, \theta_{1,1+2}, 0)}{-m_{12}^2 + (m_{a_0(980)^0})^2 - im_{a_0(980)^0} \Gamma(m_{12}^2)} \\
 &= \frac{C_{D^0 \to K^0_{a_0}(980)^0; a_0(980)^0 \to K^+_0 K^-_0} \Gamma_{a_0(980)^0} m_{a_0(980)^0}}{i \Gamma_{a_0(980)^0} (m_{a_0(980)^0})^2 \sqrt{\frac{(m_{12}^2 - (m_1 - m_2)^2)(m_{12}^2 - (m_1 + m_2)^2)}{m_{12}^2}} - m_{12}^2 + (m_{a_0(980)^0})^2} \\
 & \quad - \frac{m_{12} \sqrt{\frac{\left( (m_{a_0(980)^0})^2 - (m_1 - m_2)^2 \right) \left( (m_{a_0(980)^0})^2 - (m_1 + m_2)^2 \right)}{(m_{a_0(980)^0})^2}}}{0.074} \\
 &= \frac{0.074}{-m_{12}^2 + 0.96 - \frac{0.599 \sqrt{m_{12}^2 - 0.975}}{m_{12}}} \quad \leftarrow \text{AmpForm provides suggested parameter values}
 \end{aligned}$$

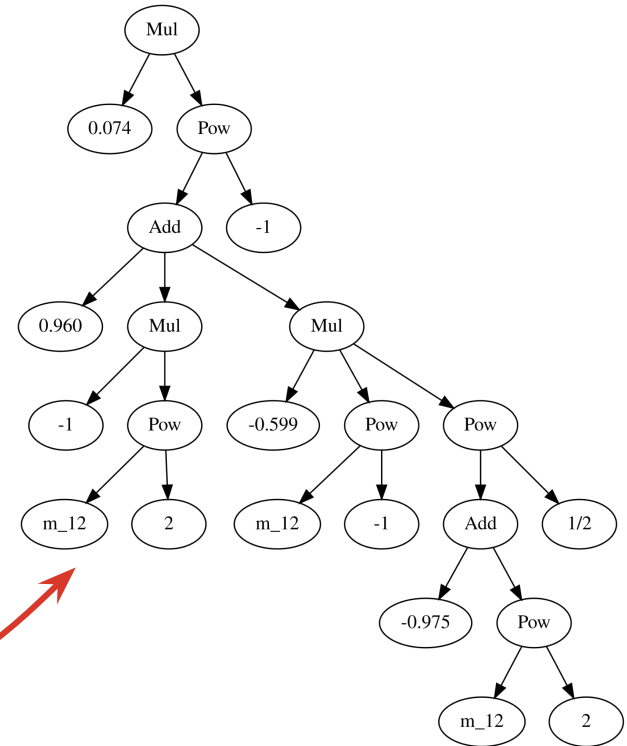
# AmpForm

## Symbolic amplitude model formulation

Original motivation: these expressions are actually **trees** that represent fundamental mathematical operations!

⇒ Can serve as template for faster computational software

$$\begin{aligned}
 & \frac{C_{D^0 \rightarrow K_0^0 a_0(980)^0} a_0(980)^0 \rightarrow K_0^+ K_0^- \Gamma_{a_0(980)^0} m_{a_0(980)^0} \sqrt{2} \left( (m_{a_0(980)^0})^2 - \frac{1}{2} (m_1^2 + m_2^2) \right)^{1/2}}{-m_{12}^2 + (m_{a_0(980)^0})^2 - i m_{a_0(980)^0} \Gamma(m_{12}^2)} \\
 &= \frac{C_{D^0 \rightarrow K_0^0 a_0(980)^0} a_0(980)^0 \rightarrow K_0^+ K_0^- \Gamma_{a_0(980)^0} m_{a_0(980)^0}}{i \Gamma_{a_0(980)^0} (m_{a_0(980)^0})^2 \sqrt{\frac{(m_{12}^2 - (m_1 - m_2)^2)(m_{12}^2 - (m_1 + m_2)^2)}{m_{12}^2}} - m_{12}^2 + (m_{a_0(980)^0})^2} \\
 &= \frac{0.074}{-m_{12}^2 + 0.96 - \frac{0.599 \sqrt{m_{12}^2 - 0.975}}{m_{12}}}
 \end{aligned}$$





# TensorWaves

Fit and generate data with multiple computational back-ends

- General fitter package
- Express amplitude templates in a computational back-end
- Generate (deterministic) amplitude-based Monte Carlo samples
- Perform unbinned fits with different back-ends  
(TensorFlow, NumPy, JAX, ...)
- Also integrates different optimizers (Minuit2, SciPy, ...)



```
intensity = LambdifiedFunction(template, backend="jax") # numpy, tensorflow
estimator = UnbinnedNLL(intensity, data_sample, phsp_sample)
optimizer = Minuit2() # Scipy
fit_result = optimizer.optimize(estimator, initial_parameters)
```

■ 187/? [00:33<00:00, 13.15it/s, estimator=-7.59e+4]

# TensorWaves

Fit and generate data with multiple computational back-ends

## Why is this nice?

- Heavy computations outsourced to specialized packages from the machine learning and data science community
- Get support for GPUs, multithreading, etc. for free
- Very small code-base — easy to maintain
- More time for physics!



```
intensity = LambdifiedFunction(template, backend="jax") # numpy, tensorflow
estimator = UnbinnedNLL(intensity, data_sample, phsp_sample)
optimizer = Minuit2() # Scipy
fit_result = optimizer.optimize(estimator, initial_parameters)
```

■ 187/? [00:33<00:00, 13.15it/s, estimator=-7.59e+4]

# TensorWaves

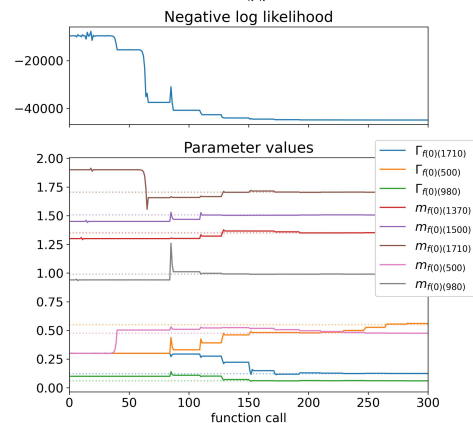
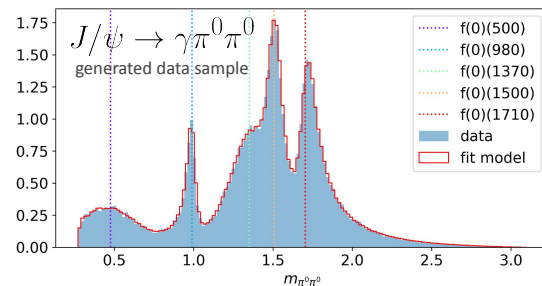
Fit and generate data with multiple computational back-ends

## Why is this nice?

- Heavy computations outsourced to specialized packages from the machine learning and data science community
- Get support for GPUs, multithreading, etc. for free
- Very small code-base — easy to maintain
- More time for physics!

Some JAX+Minuit2 performance numbers on a single machine:

	data	phsp	params.	expr. complexity	duration
mini-demo on the right	$10^5$	$10^6$	8	2,187 nodes	~1 minute
benchmark fit $J/\psi \rightarrow K^0 \Sigma^+ \bar{p}$	$1.2 \times 10^5$	$2.3 \times 10^5$	84	176,023 nodes	3 hours



# PWA Software Pages

## Interactive knowledge-base for PWA theory and software

All packages come with well-maintained websites:

- Extensive explanations of implemented physics
- Run demos directly from the browser
- Easily navigate to library interface
- Kind of an **interactive book** (see [Executable Book Project](#))
- Continuously tested: links and code examples won't break

Narrow the gap between code and theory!

Search the docs ...

Installation

Usage

Formulate amplitude model

Inspect model interactively

Helicity versus canonical

Custom dynamics

Analytic continuation

Bibliography

API

Changelog

Help developing

RELATED PROJECTS

QRules

TensorWaves

PWA Pages

COMPWA ORGANIZATION

Website

GitHub Repositories

About

Contents

Physics

- Partial wave expansion
- Transition operator
- Ensuring unitarity
- Lorentz-invariance
- Production processes**
- Pole parametrization
- Implementation
- Interactive visualization

*Launch interactive examples*

^ Pole parametrization

After all these matrix definitions, the final challenge is to find the correct parametrization for the elements of  $\mathbf{K}$  and  $\mathbf{P}$  that accurately describes the resonances we observe.<sup>[3]</sup> There are several choices, but a common one is the following summation over the poles  $R$ :<sup>[4]</sup>

$$K_{ij} = \sum_R \frac{g_{R,i} g_{R,j}}{m_R^2 - s} + c_{ij} \quad (14)$$
$$\hat{K}_{ij} = \sum_R \frac{g_{R,i}(s) g_{R,j}(s)}{(m_R^2 - s) \sqrt{\rho_i \rho_j}} + \hat{c}_{ij}$$

with  $c_{ij}$ ,  $\hat{c}_{ij}$  some optional background characterization and  $g_{R,i}$  the **residue functions**. The residue functions are often further expressed as:

$$g_{R,i} = \gamma_{R,i} \sqrt{m_R \Gamma_{R,i}^0} \quad (15)$$
$$g_{R,i}(s) = \gamma_{R,i} \sqrt{m_R \Gamma_{R,i}(s)}$$

with  $\gamma_{R,i}$  some *real* constants and  $\Gamma_{R,i}^0$  the **partial width** of each pole. In the Lorentz-invariant form, the fixed width  $\Gamma^0$  is replaced by an “energy dependent” **CoupledWidth**  $\Gamma(s)$ .<sup>[5]</sup> The **width** for each pole can be computed as  $\Gamma_R^0 = \sum_i \Gamma_{R,i}^0$ .

The production vector  $P$  is commonly parameterized

[4] Eqs. (75-78)

# PWA Software Pages

## Interactive knowledge-base for PWA theory and software

All packages come with well-maintained websites:

- Extensive explanations of implemented physics
- Run demos directly from the browser
- Easily navigate to library interface
- Kind of an **interactive book** (see [Executable Book Project](#))
- Continuously tested: links and code examples won't break

Narrow the gap between code and theory!

The screenshot displays a JupyterLab environment with a Python 3 (ipykernel) session. The notebook content includes:

- Code defining `relativistic_breit_wigner` and `beta1`, `beta2` symbols.
- Mathematical formulas for the Breit-Wigner function and its squared magnitude.
- Two plots: `Im A (P-vector)` and `Im A (Breit-Wigner)`, showing resonance peaks.
- A complex plane plot showing the imaginary part of the amplitude `Im(A)` and the real part `Re(A)`.
- Interactive sliders for parameters: `c1`, `phi1`, `m1`, `Gamma1`, `gamma1`, and `z-cutoff`.
- A legend for the `s-plane plot` with options for `imag` (selected), `real`, and `abs`.

The interface also shows a file browser, a terminal, and a status bar at the bottom indicating the current file is `k-matrix.ipynb`.

# PWA Software Pages

Interactive knowledge-base for PWA theory and software

⇒ Spin-off project: PWA Pages ([pwa.rtf.d.io](https://pwa.rtf.d.io))

- Intended as a guide through the main ingredients of Partial Wave Analysis
- Readers can easily navigate to literature or existing PWA software
- Currently skeletal, but infrastructure is there and easy contribute to
- No need to know HTML, CSS, etc.

**Happy to include or reference your PWA project!**

# PWA Software Pages

Interactive knowledge-base for PWA theory and software

⇒ Spin-off project: PWA Pages ([pwa.rtf.d.io](https://pwa.rtf.d.io))

- Intended as a guide through the main ingredients of Partial Wave Analysis
- Readers can easily navigate to literature or existing PWA software
- Currently skeletal, but infrastructure is there and easy contribute to
- No need to know HTML, CSS, etc.

**Happy to include or reference your PWA project!**

*Thank you  
for your attention!*