The **ComPWA** project

**Speeding up amplitude analysis with a Computer Algebra System**

**Three Python Libraries** for a full amplitude analysis

**QRules:** Automated quantum number conservation rules

- **Input:** boundary conditions
  - initial and final state particles
  - optional intermediate state restrictions

- **Output:** state transitions
  - Determines possible decay topologies
  - Gets corresponding particle properties from the PDG (or custom definitions)
  - Propagates quantum numbers through intermediate states
  - Selects all allowed transitions with its conservation laws

**AmpForm:** Symbolic amplitude models

- Library of spin formalisms and dynamics
- Formulate amplitude model for QRules output
- Implemented formalisms:
  - Helicity formalism for multibody decays
  - Transformation to canonical basis
  - Dynamics parameterisations, such as
    - analytic continuation
    - Spin alignments

**TensorWaves:** Fit data with multiple computational back-ends

- Mathematical expressions → Computational back-end
- Generation of amplitude-based Monte Carlo samples
- Fits with TensorFlow, NumPy, JAX...
- Different optimizers: Minuit2, SciPy...

**Core idea:**
1. Formulate amplitude model as formulas
   - Computer Algebra System
2. Usage of fast numerical back-ends

**Main benefits:**
- Physics separated from number crunching
- Switch back-end without changing analysis code
- Symbolic expressions result in self-documenting workflow
- Computations outsourced to fast, optimized back-ends with a large user-base
- Out-of-the-box GPU and multithreading support

**CAS simplifications result in performance boosts**

- **Example**
  - Amplitude model with 12 resonances and 59 parameters
  - Expression tree complexity: parametrized: 43,198 nodes substituted: 0.024 nodes
  - Backend: JAX
  - 1 CPU Intel i7-8750H 2.20GHz
  - 25% faster computation time

**Spin-off project:** polarimetry vector field of $\Lambda_c^+ \to pK^-\pi^+$

- Power by symbolic expressions
- Explanation of implemented physics
- Kind of an interactive book in the browser (see Executable Book Project)

**Self-documenting workflow**

- Thoroughly integrated with codebase
- Continuously tested links and code

**Powered by symbolic expressions**

- Example:
  - $M^{(\Phi)}(e^{i\theta}, e^{i\phi}) = I_0(s) \left( 1 + \sum_{\text{nonzer}} P_{\text{nonzer}}(e^{i\theta}, e^{i\phi}) \right)$

**CAS offers the flexibility to compute vector fields and easily test several parameterizations**