Long live Amplitude Models

(COMAP-V mini-workshop)

Misha Mikhasenko

Rurh University Bochum

Quantum process phenomenology

$$\boldsymbol{I} = \sum_{\textbf{spin stats}} |\boldsymbol{A}|^2$$

- *I:* Observed spectrum ~ probability distribution (observation)
- *A:* transition amplitude (model)

Amplitude Analysis and Partial waves

$$A = \sum_{\text{partial waves}}^{\text{many}} T(\text{mass})\, \psi\,(\text{angles})$$

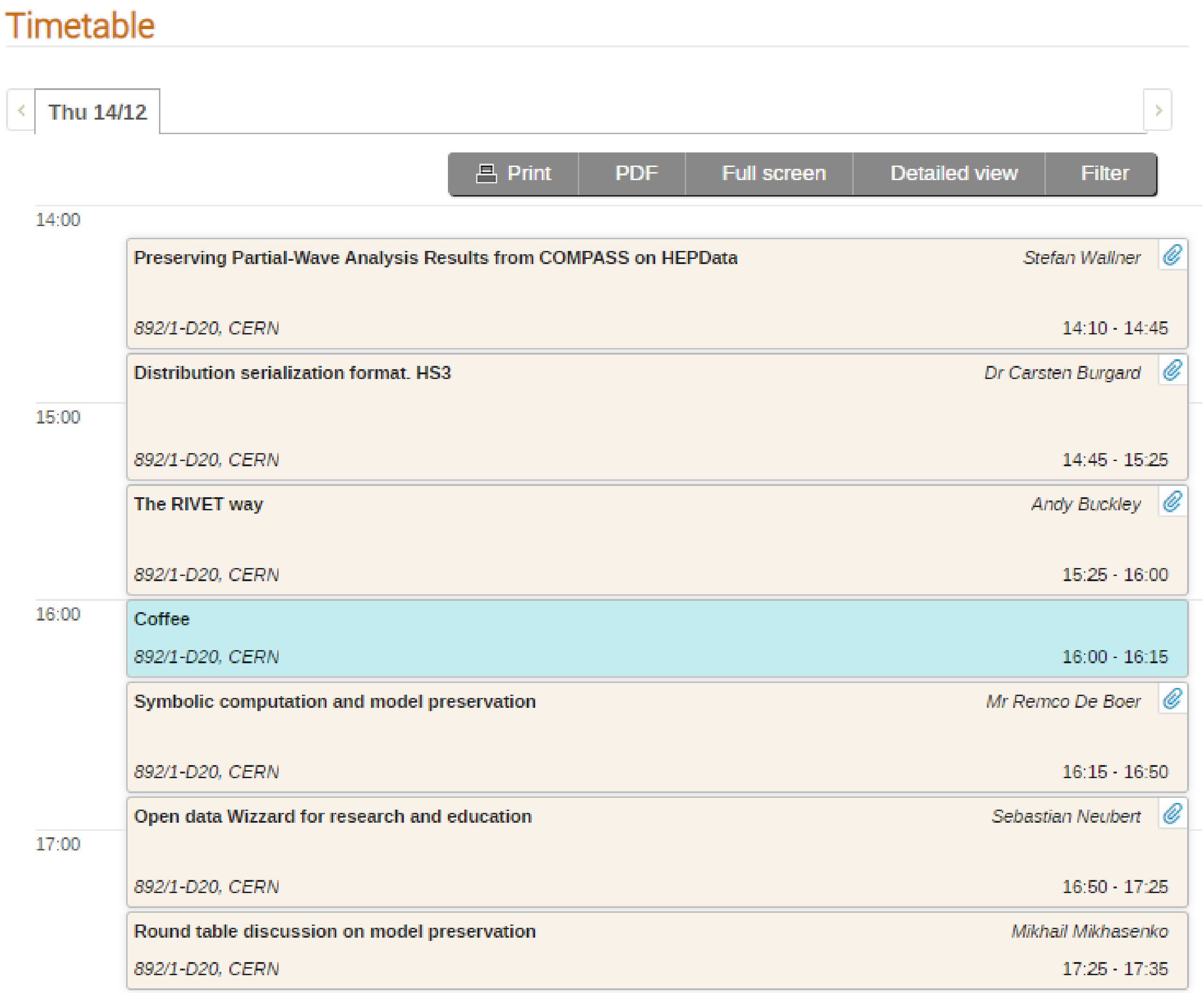
LHCb, ATLAS, CMS, BELLE, BES:

decays kinematics (1->3 mostly, 1->4 sometimes , + polarization)

COMPASS, GlueX, Crystall Barrel:

production process (2->4 mostly, + polarization)

Program to today



Timetable

Thu 14/12

14:00

Preserving Partial-Wave Analysis Results from COMPASS on HEPData — *Stefan Wallner*
892/1-D20, CERN — 14:10 - 14:45

Distribution serialization format. HS3 — *Dr Carsten Burgard*
892/1-D20, CERN — 14:45 - 15:25

15:00

The RIVET way — *Andy Buckley*
892/1-D20, CERN — 15:25 - 16:00

16:00

Coffee
892/1-D20, CERN — 16:00 - 16:15

Symbolic computation and model preservation — *Mr Remco De Boer*
892/1-D20, CERN — 16:15 - 16:50

17:00

Open data Wizzard for research and education — *Sebastian Neubert*
892/1-D20, CERN — 16:50 - 17:25

Round table discussion on model preservation — *Mikhail Mikhasenko*
892/1-D20, CERN — 17:25 - 17:35

COMPASS PW:
one of the most complex setup

Preservation of "nested"
statistical models

Preserving / explaining
the analysis workflow

Symbolic model serialization

Open data and
analysis workflow in LHCb.

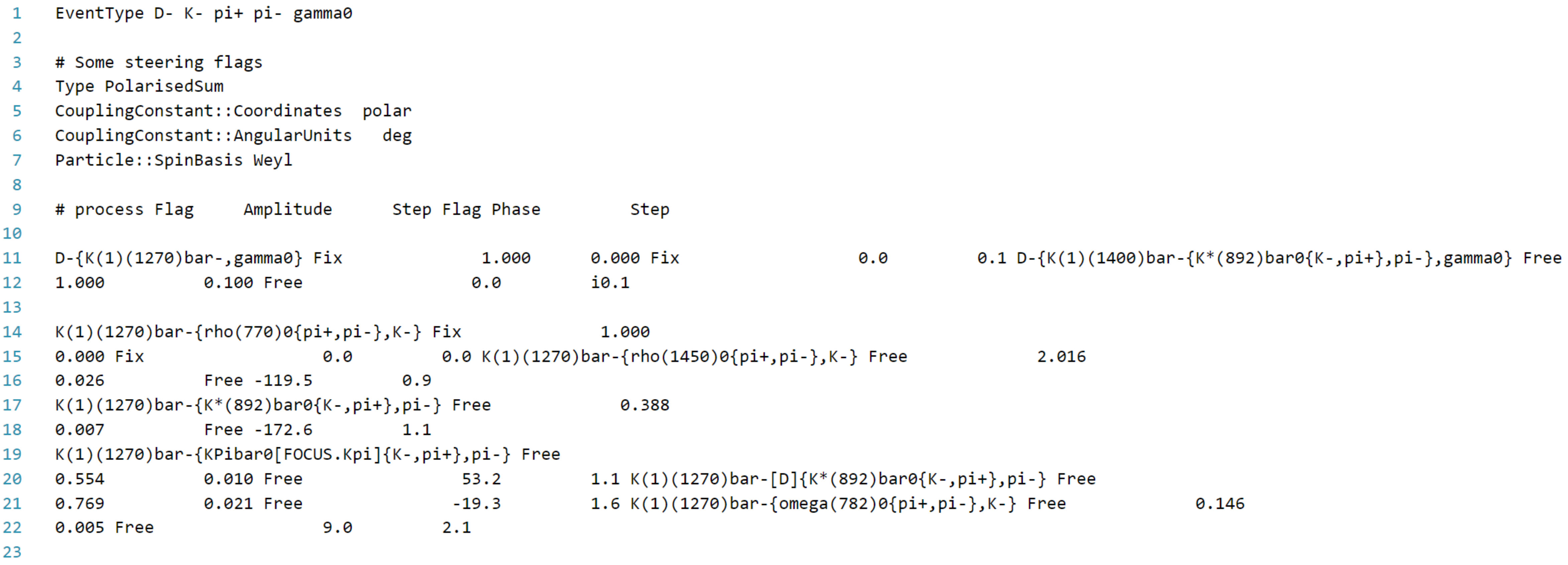
Round table discussion.

A few comments

AmpGen example

https://github.com/GooFit/AmpGen

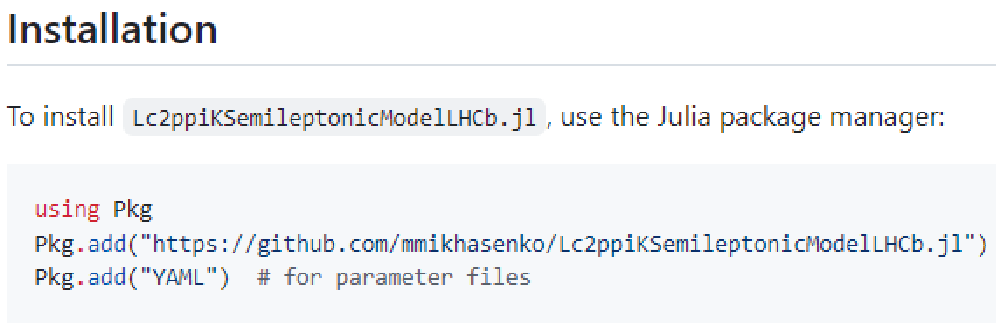
Model building / Fitting framework.
Models are shared between experiments using AMPGEN DSL



```
EventType D- K- pi+ pi- gamma0

# Some steering flags
Type PolarisedSum
CouplingConstant::Coordinates  polar
CouplingConstant::AngularUnits   deg
Particle::SpinBasis Weyl

# process Flag     Amplitude      Step Flag Phase         Step

D-{K(1)(1270)bar-,gamma0} Fix              1.000      0.000 Fix                  0.0         0.1 D-{K(1)(1400)bar-{K*(892)bar0{K-,pi+},pi-},gamma0} Free
1.000          0.100 Free                 0.0         i0.1

K(1)(1270)bar-{rho(770)0{pi+,pi-},K-} Fix              1.000
0.000 Fix                  0.0         0.0 K(1)(1270)bar-{rho(1450)0{pi+,pi-},K-} Free             2.016
0.026          Free -119.5         0.9
K(1)(1270)bar-{K*(892)bar0{K-,pi+},pi-} Free             0.388
0.007          Free -172.6         1.1
K(1)(1270)bar-{KPibar0[FOCUS.Kpi]{K-,pi+},pi-} Free
0.554          0.010 Free                53.2         1.1 K(1)(1270)bar-[D]{K*(892)bar0{K-,pi+},pi-} Free
0.769          0.021 Free               -19.3         1.6 K(1)(1270)bar-{omega(782)0{pi+,pi-},K-} Free             0.146
0.005 Free                 9.0         2.1

```

Amplitude models as Julia package

https://github.com/mmikhasenko/Lc2ppiKSemileptonicModelLHCb.jl

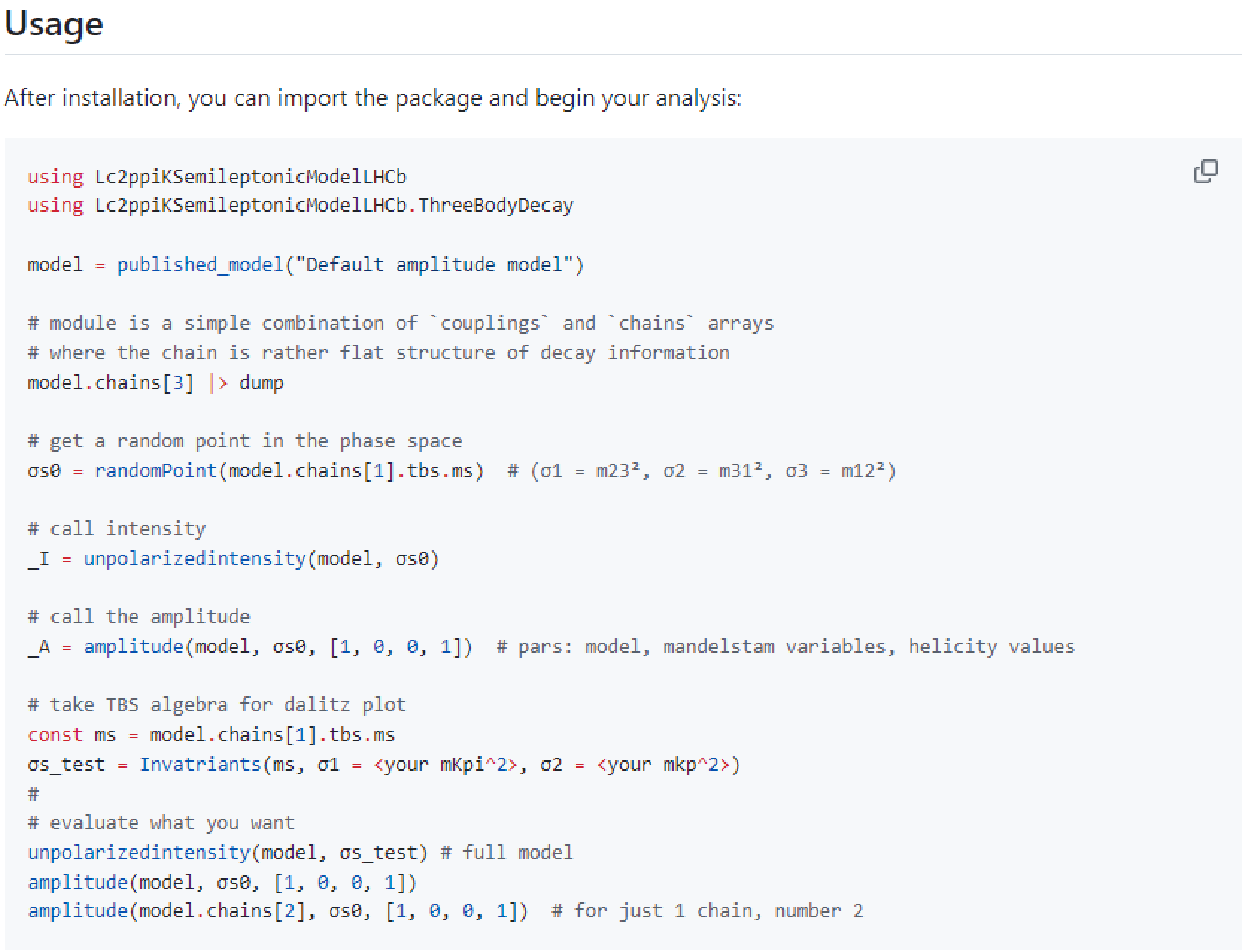


Installation

To install `Lc2ppiKSemileptonicModelLHCb.jl`, use the Julia package manager:

```
using Pkg
Pkg.add("https://github.com/mmikhasenko/Lc2ppiKSemileptonicModelLHCb.jl")
Pkg.add("YAML")  # for parameter files
```

Thanks to excellent package manager in Julia.
Straightforward to pull model,
do whatever you want



Usage

After installation, you can import the package and begin your analysis:

```
using Lc2ppiKSemileptonicModelLHCb
using Lc2ppiKSemileptonicModelLHCb.ThreeBodyDecay

model = published_model("Default amplitude model")

# module is a simple combination of `couplings` and `chains` arrays
# where the chain is rather flat structure of decay information
model.chains[3] |> dump

# get a random point in the phase space
σs0 = randomPoint(model.chains[1].tbs.ms)  # (σ1 = m23², σ2 = m31², σ3 = m12²)

# call intensity
_I = unpolarizedintensity(model, σs0)

# call the amplitude
_A = amplitude(model, σs0, [1, 0, 0, 1])  # pars: model, mandelstam variables, helicity values

# take TBS algebra for dalitz plot
const ms = model.chains[1].tbs.ms
σs_test = Invatriants(ms, σ1 = <your mKpi^2>, σ2 = <your mkp^2>)
#
# evaluate what you want
unpolarizedintensity(model, σs_test) # full model
amplitude(model, σs0, [1, 0, 0, 1])
amplitude(model.chains[2], σs0, [1, 0, 0, 1])  # for just 1 chain, number 2
```

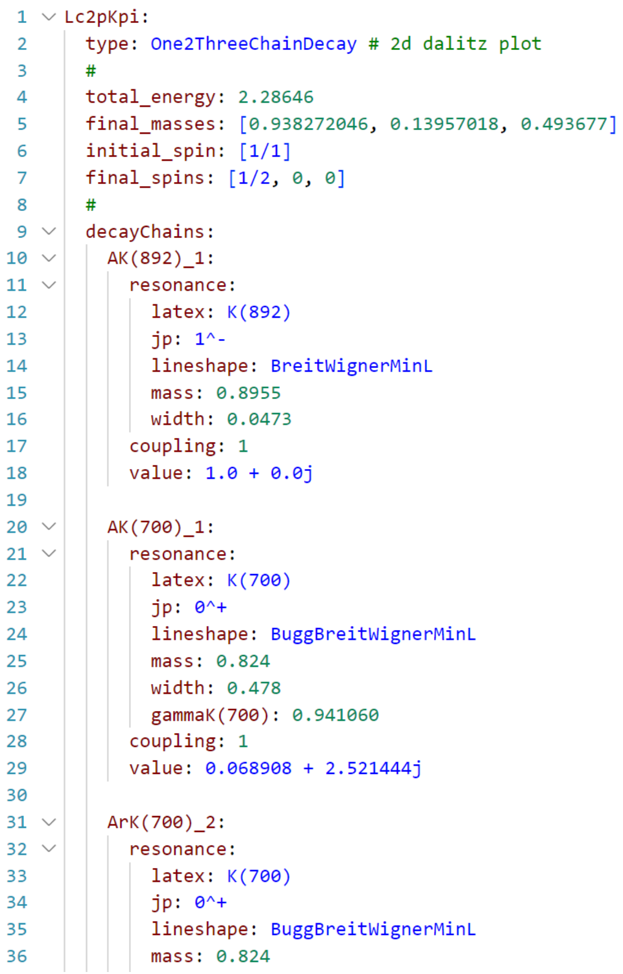
Fostering Benchmarks

Example: https://github.com/iris-hep/adl-benchmarks-index

Common format on model would allow

- comparing frameworks in term of correctness,
- in term of performance,
- gauge optimization gain on multithreading, GPU
- gauge a gain of autodiff

How three-body decay model might look like?



```
Lc2pKpi:
  type: One2ThreeChainDecay # 2d dalitz plot
  #
  total_energy: 2.28646
  final_masses: [0.938272046, 0.13957018, 0.493677]
  initial_spin: [1/1]
  final_spins: [1/2, 0, 0]
  #
  decayChains:
    AK(892)_1:
      resonance:
        latex: K(892)
        jp: 1^-
        lineshape: BreitWignerMinL
        mass: 0.8955
        width: 0.0473
      coupling: 1
      value: 1.0 + 0.0j

    AK(700)_1:
      resonance:
        latex: K(700)
        jp: 0^+
        lineshape: BuggBreitWignerMinL
        mass: 0.824
        width: 0.478
        gammaK(700): 0.941060
      coupling: 1
      value: 0.068908 + 2.521444j

    ArK(700)_2:
      resonance:
        latex: K(700)
        jp: 0^+
        lineshape: BuggBreitWignerMinL
        mass: 0.824
```

Building complex amplitude.
Dalitz plot density ~ sum |A|^2

resonance: should be an amplitude

How do we proceed?

1. Collect sample of published models
2. Explore HS3++like format what would deal with complex amplitudes
3. Implement interfaces: Julia, RooFit, ..?
4. ???