ComPWA Status
The Common Partial Wave Analysis Framework

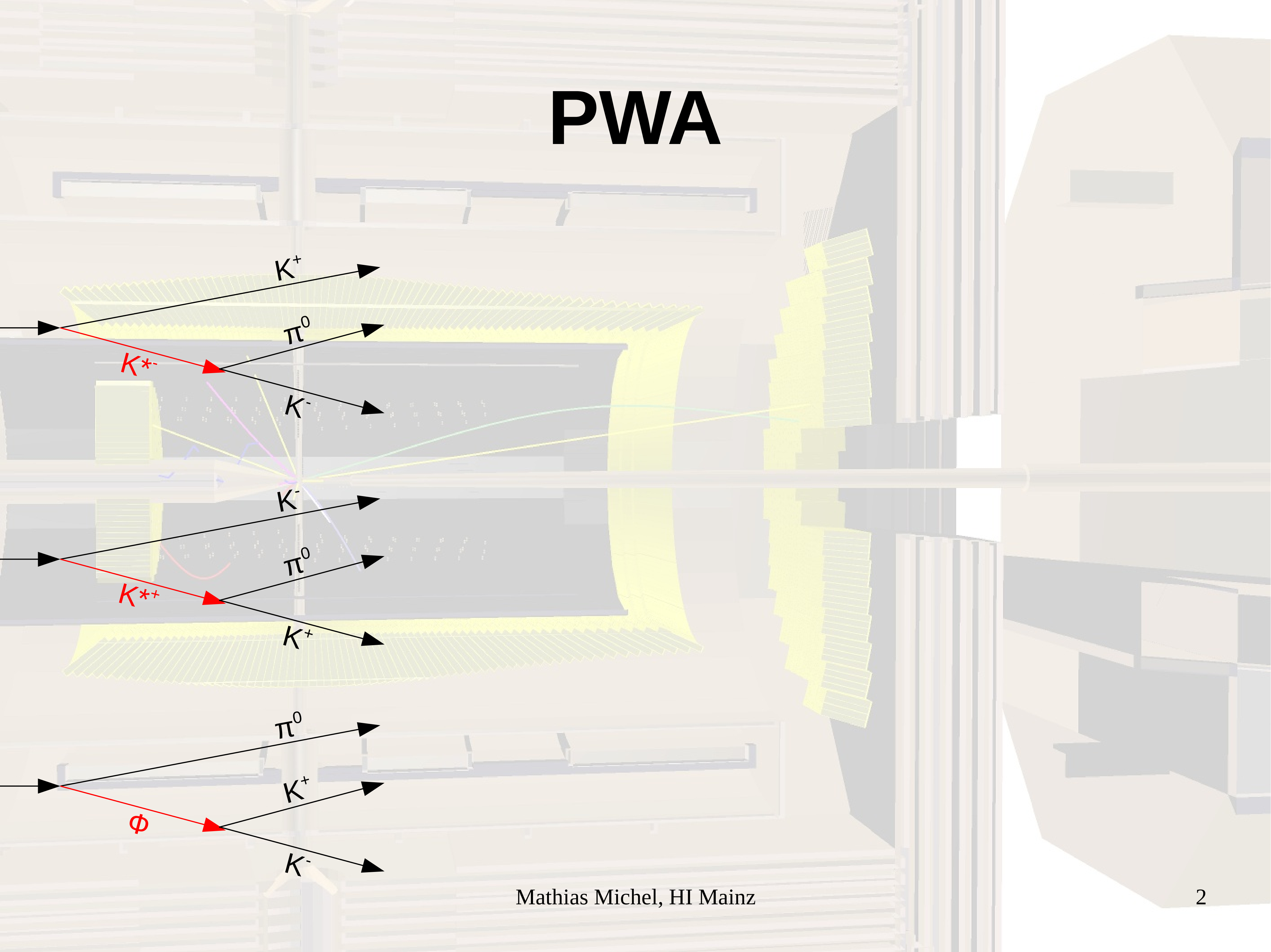
Mathias Michel
Helmholtz Institute Mainz

On behalf of the ComPWA group

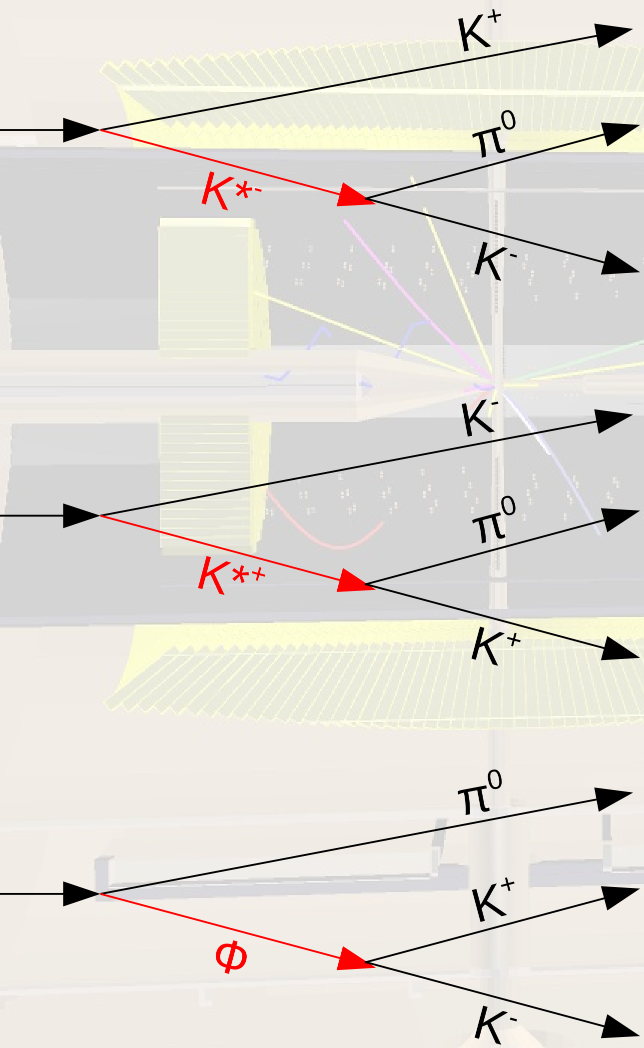
DPG Frühjahrstagung 2015, Heidelberg

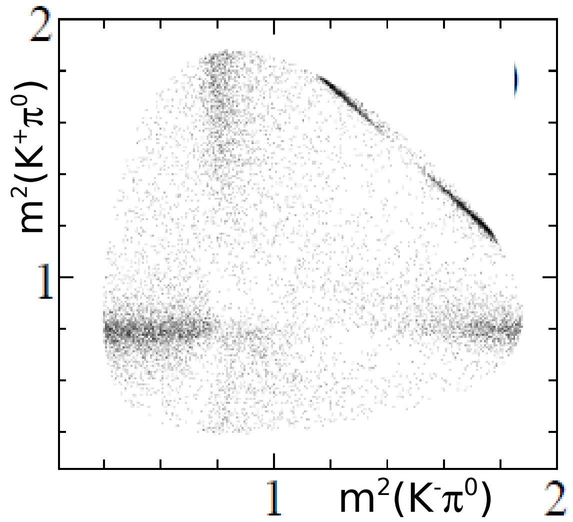
26 März 2015

PWA

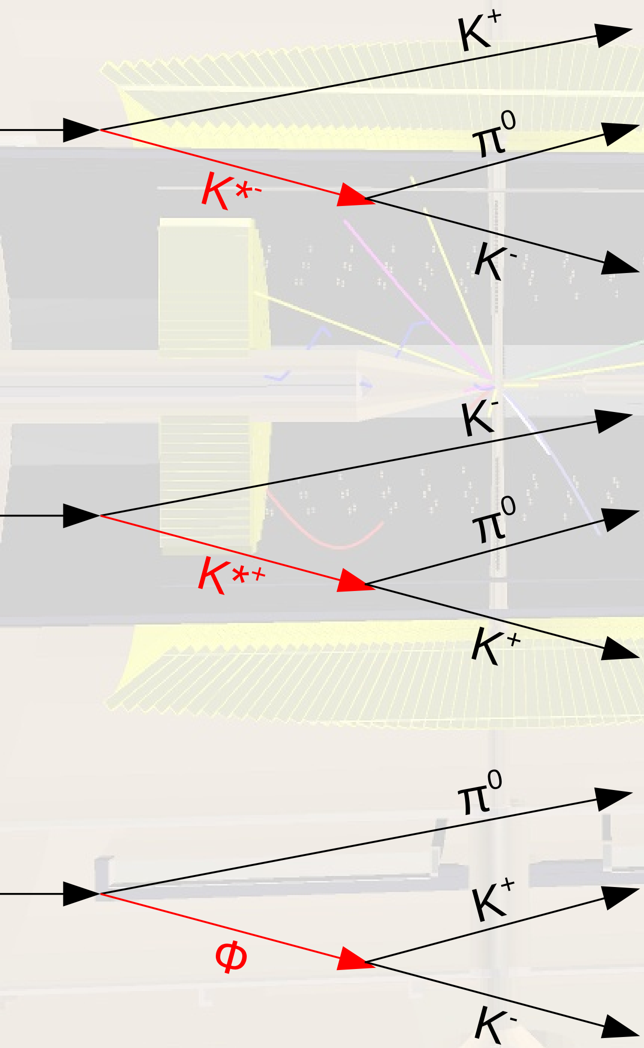


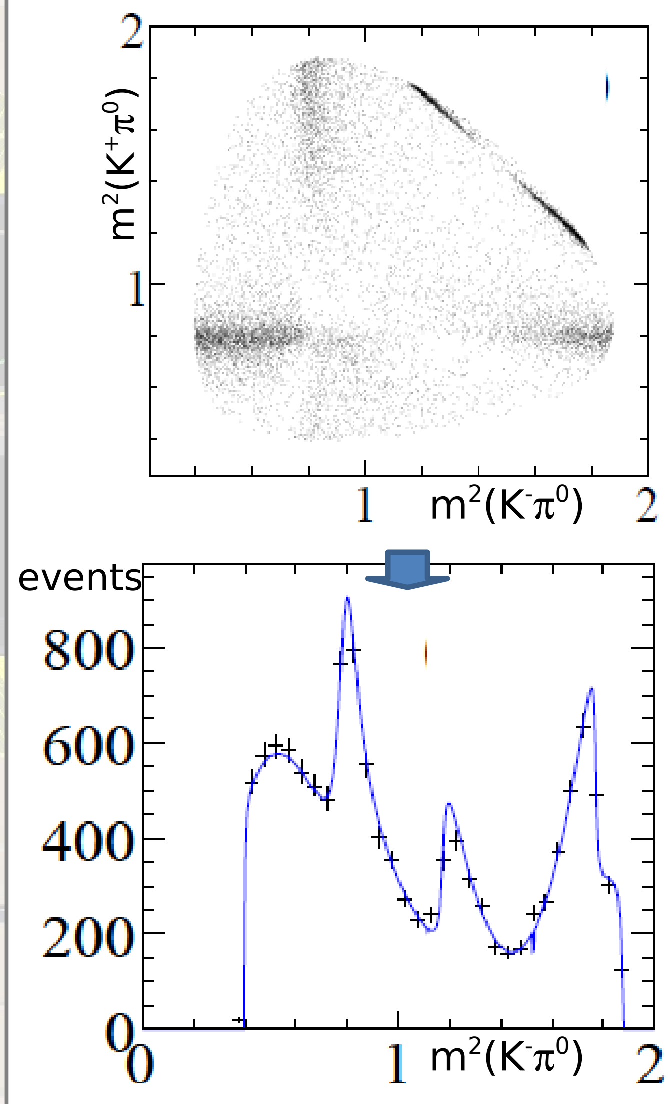
PWA



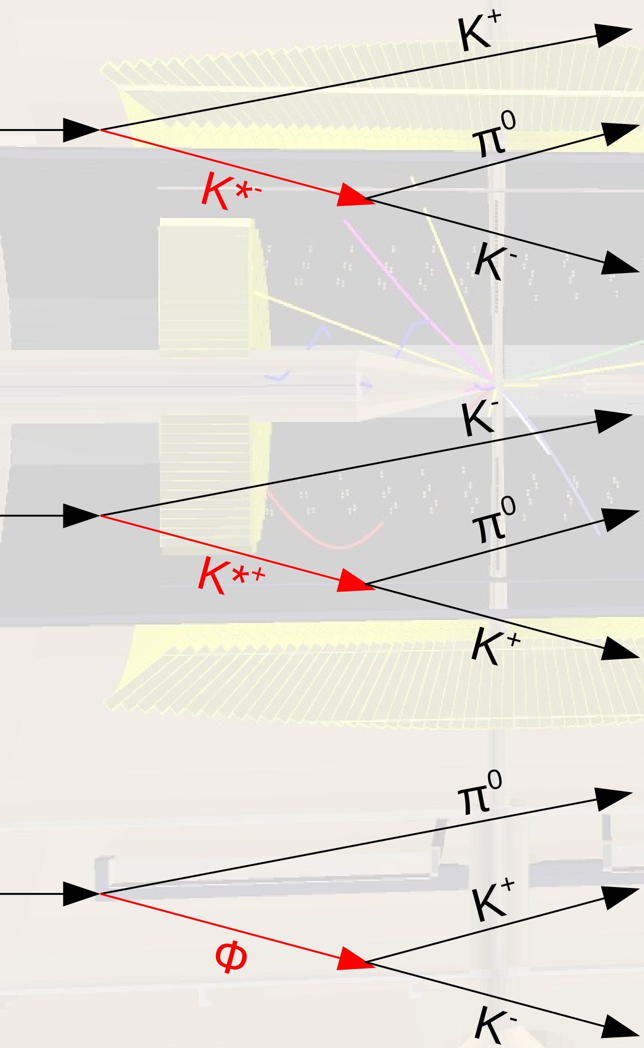


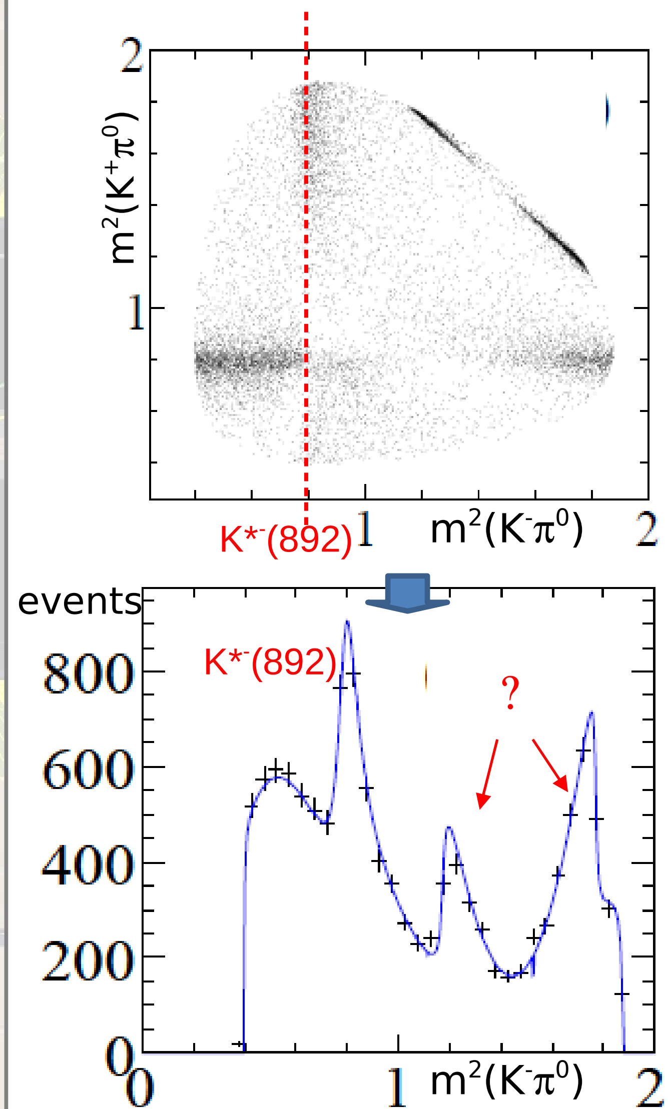
PWA


K+
π0
K*-
K-
K-
π0
K*+
K+
π0
K+
φ
K-

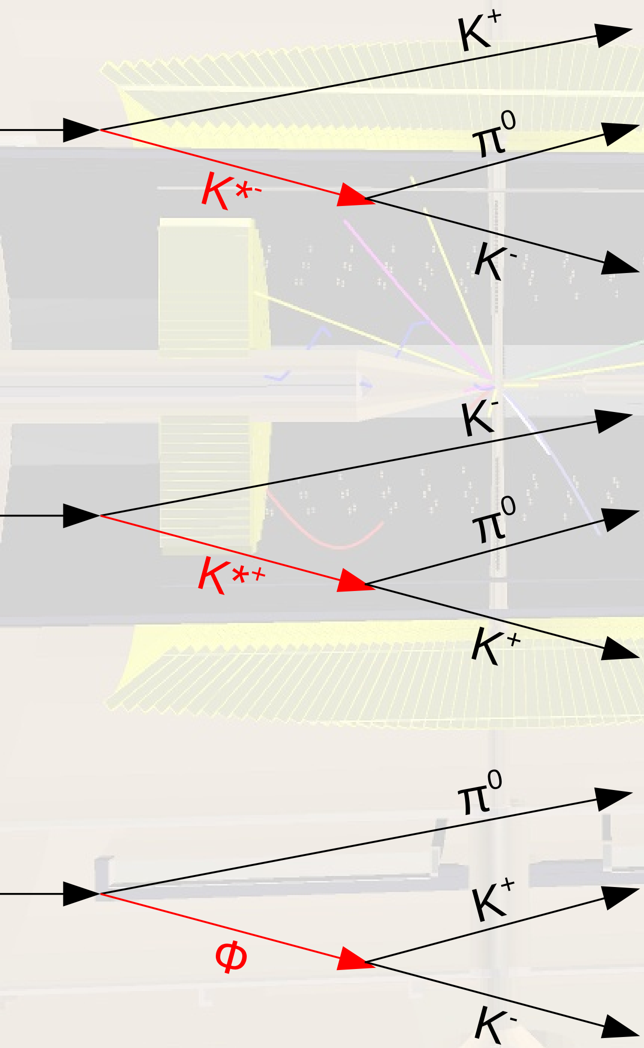

2
1
m2(K+π0)
1
m2(K-π0)
2
events
800
600
400
200
0
0
1
m2(K-π0)
2

PWA

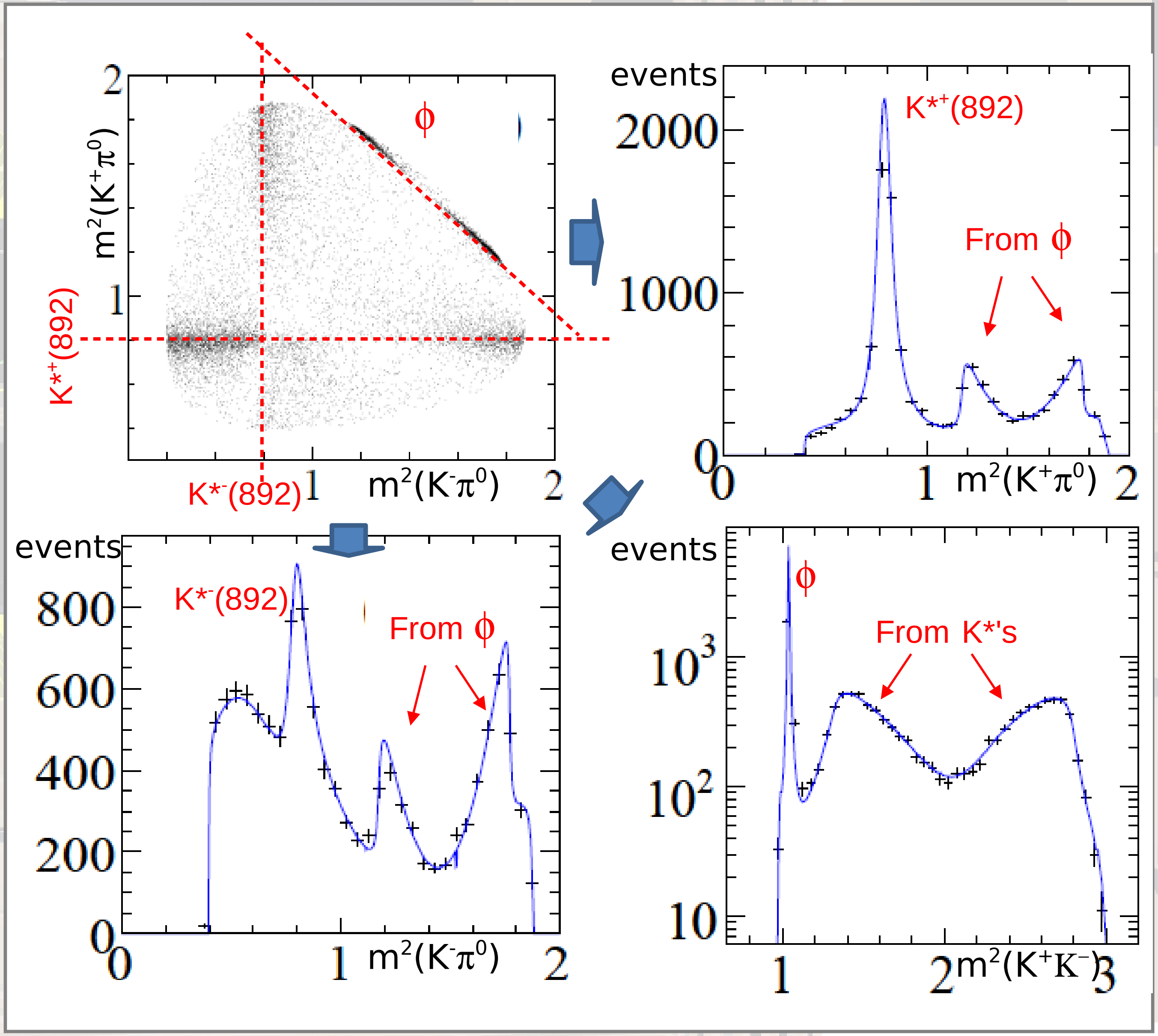

K+
π0
K*-
K-
K-
π0
K*+
K+
π0
K+
Φ
K-


$m^2(K^+\pi^0)$
2
1
K*-(892)
1
$m^2(K^-\pi^0)$
2
events
800
600
400
200
0
K*-(892)
?
0
1
$m^2(K^-\pi^0)$
2

PWA



K+
π0
K*-
K-
K-
π0
K*+
K+
π0
K+
Φ
K-



m2(K+π0)
K*+(892)
K*-(892)
m2(K-π0)
ϕ
events
K*+(892)
From ϕ
m2(K+π0)
events
K*-(892)
From ϕ
m2(K-π0)
events
ϕ
From K*'s
m2(K+K-)

PWA Challenges

Example: $\overline{\mathrm{P}}$ANDA

broad physics program
→ various models needed

$\bar{p}p$ initial state at $\sqrt{s}$ 2 – 5 GeV
→ high initial spin (≈ up to 6-7)
→ many possible waves
→ many parameters

large number of events
→ parallelization needed

detector effects
→ distorted phasespace
→ accaptence

coupled channels
→ different efficiencies

quality assurance

ComPWA

Why a common _framework_?

support for various models
re-usage of implementations
easily comparison of methods

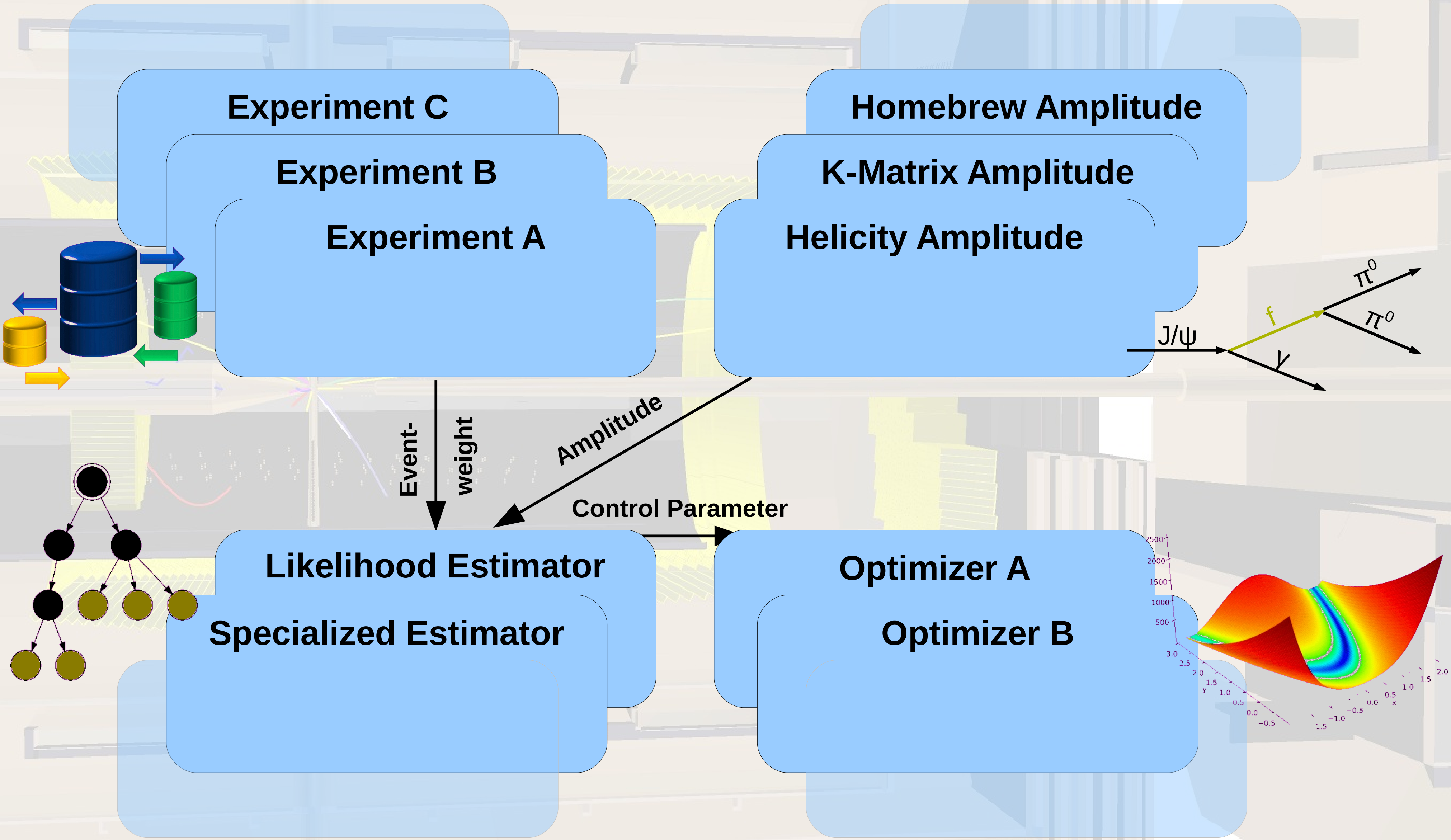
Why a _common_ framework?

support for various experiments
tools on the market are specialised
PANDA has time for preparation

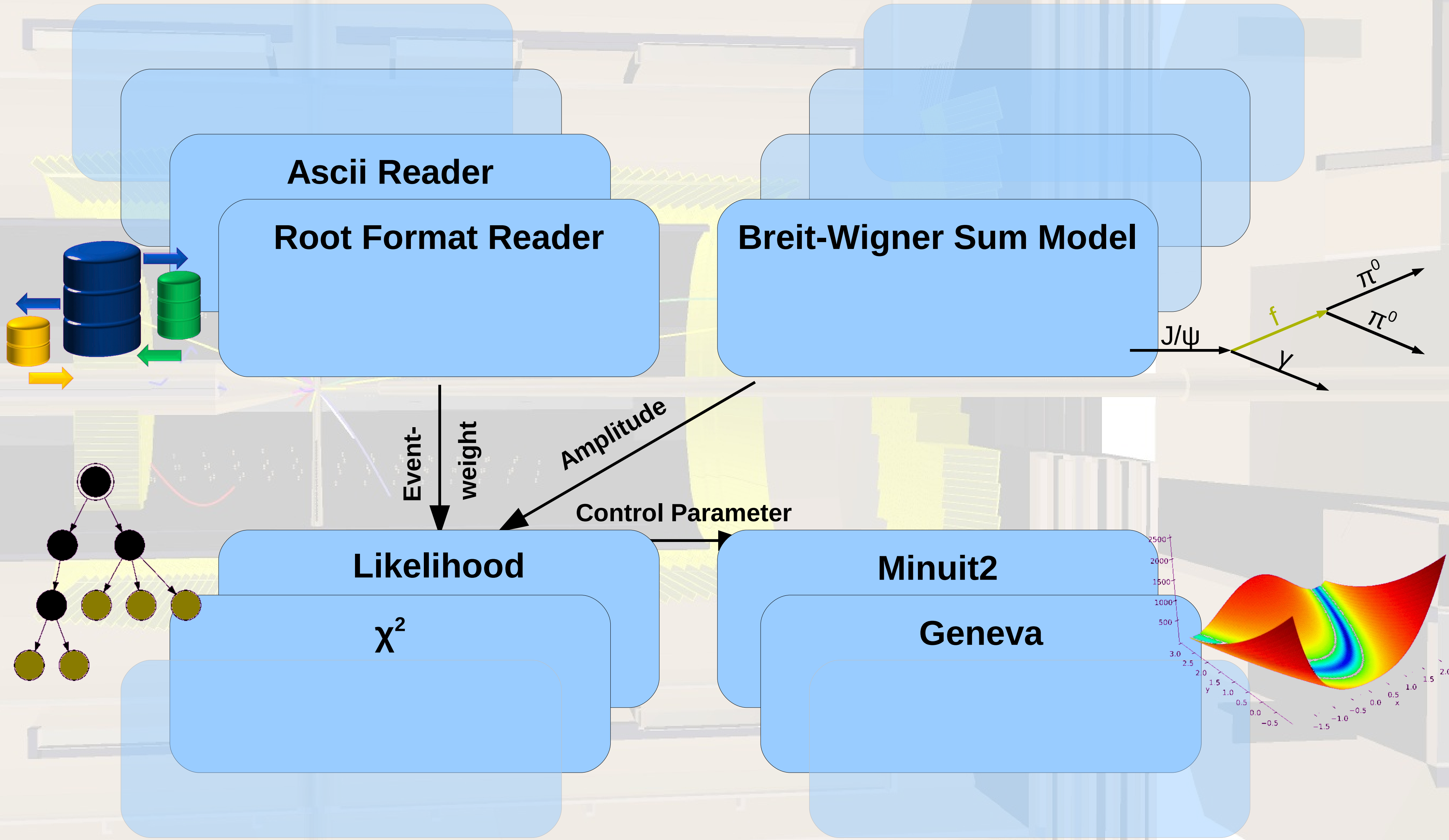
→ start now and we could have a well tested and reliable software ready

→ comparison of results from different experiments possible

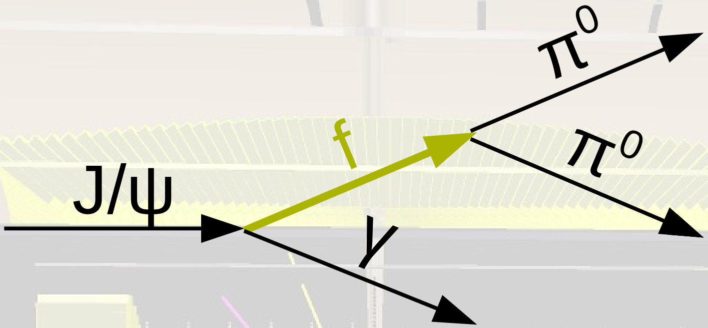
ComPWA Framework

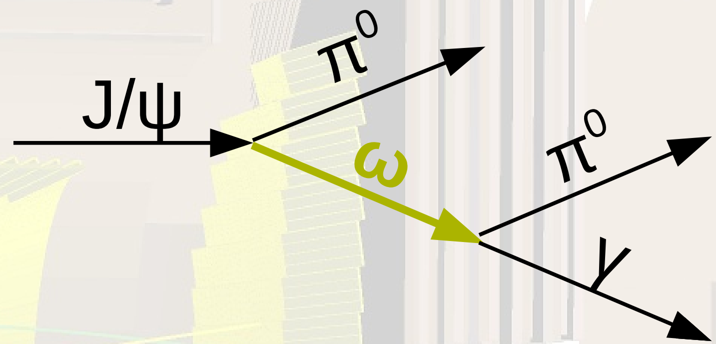


ComPWA Framework



$J/\psi \rightarrow \gamma\pi^0\pi^0$ Model

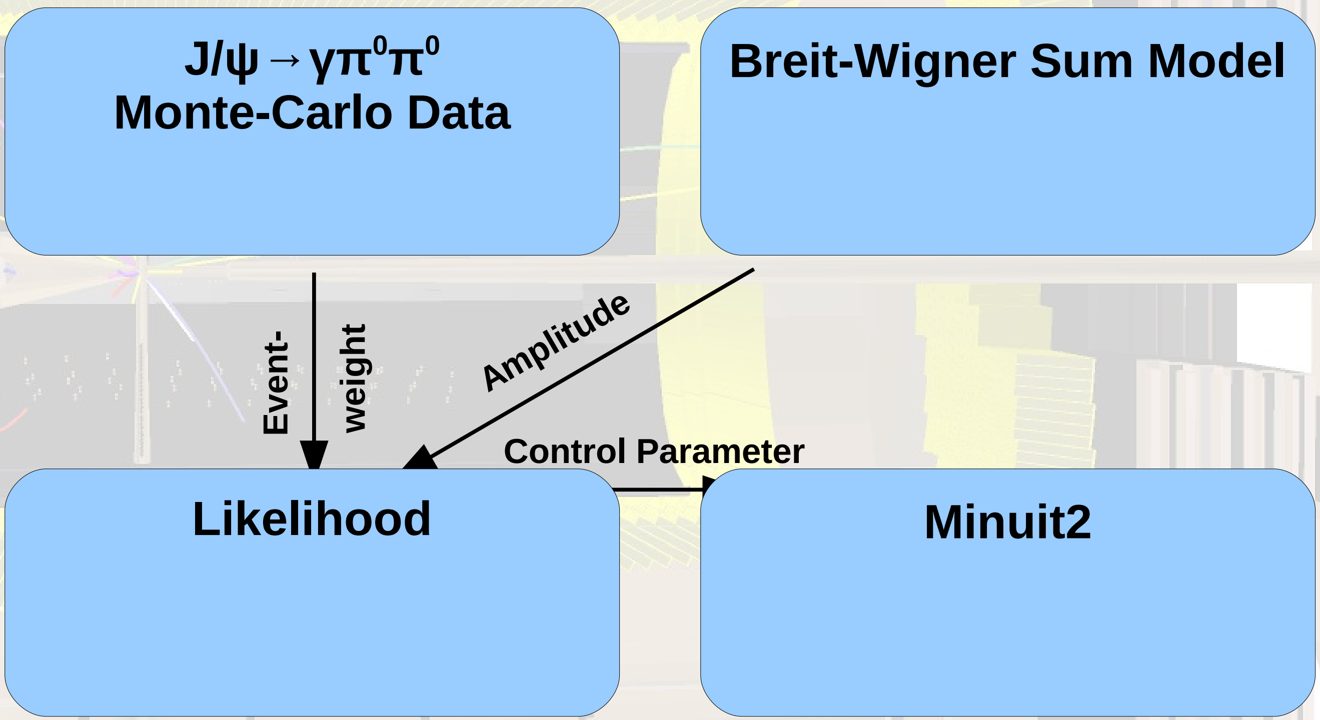




$$I = \left| \sum_n T_n r_n e^{i\varphi_n} D_n \right|^2$$

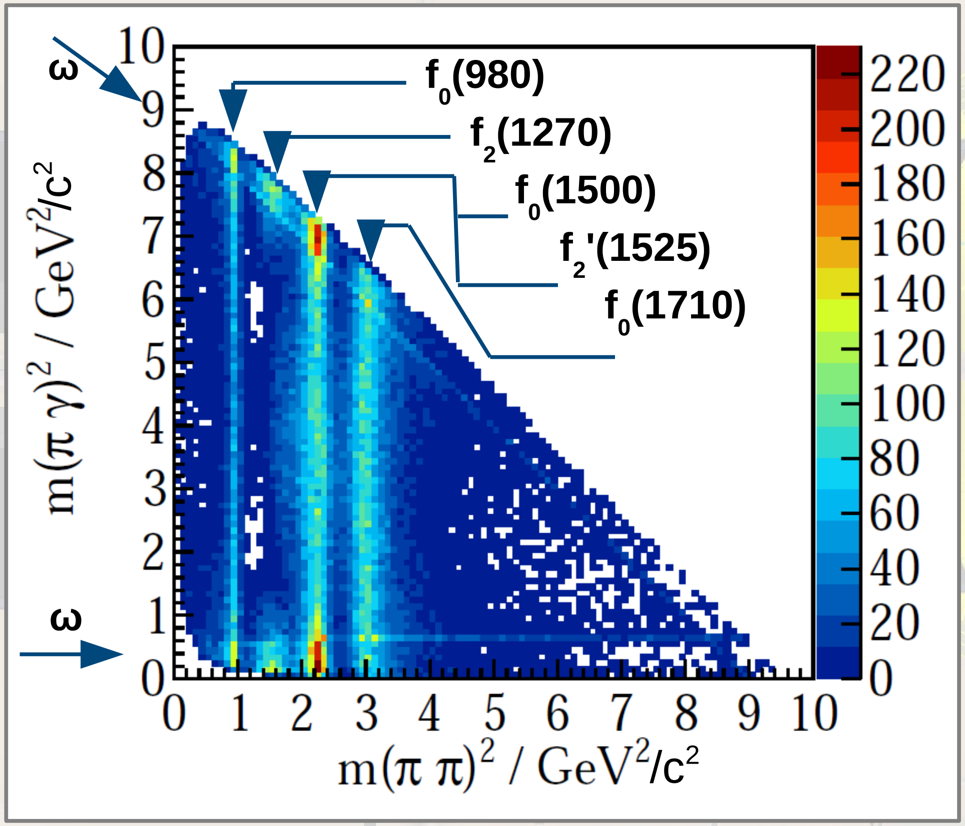
T = Breit-Wigner Function
D = D-Wigner Function
r = Strength of Resonance
φ = Phase of Resonance

$J/\psi \to \gamma\pi^0\pi^0$ Dalitz Fit

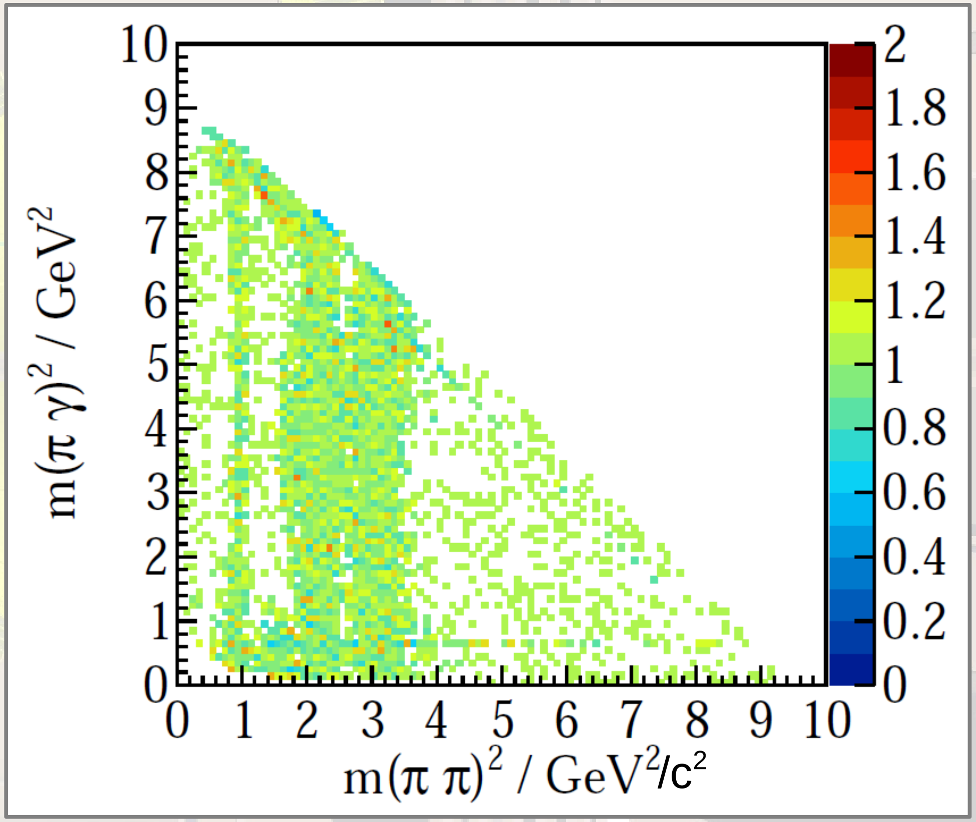


$J/\psi \rightarrow \gamma\pi^0\pi^0$ Dalitz Fit

Hit & miss Monte-Carlo with **fitted parameter**:

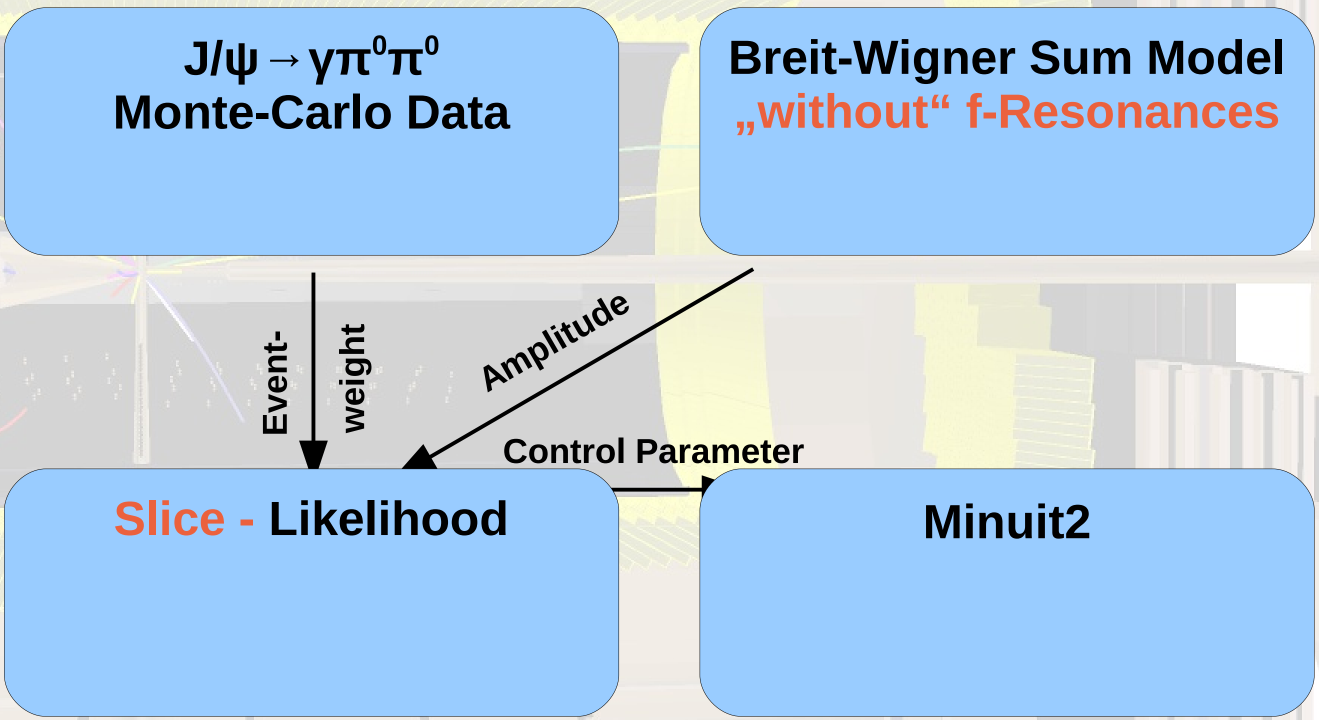


Ratio to ideal parameter MC data:

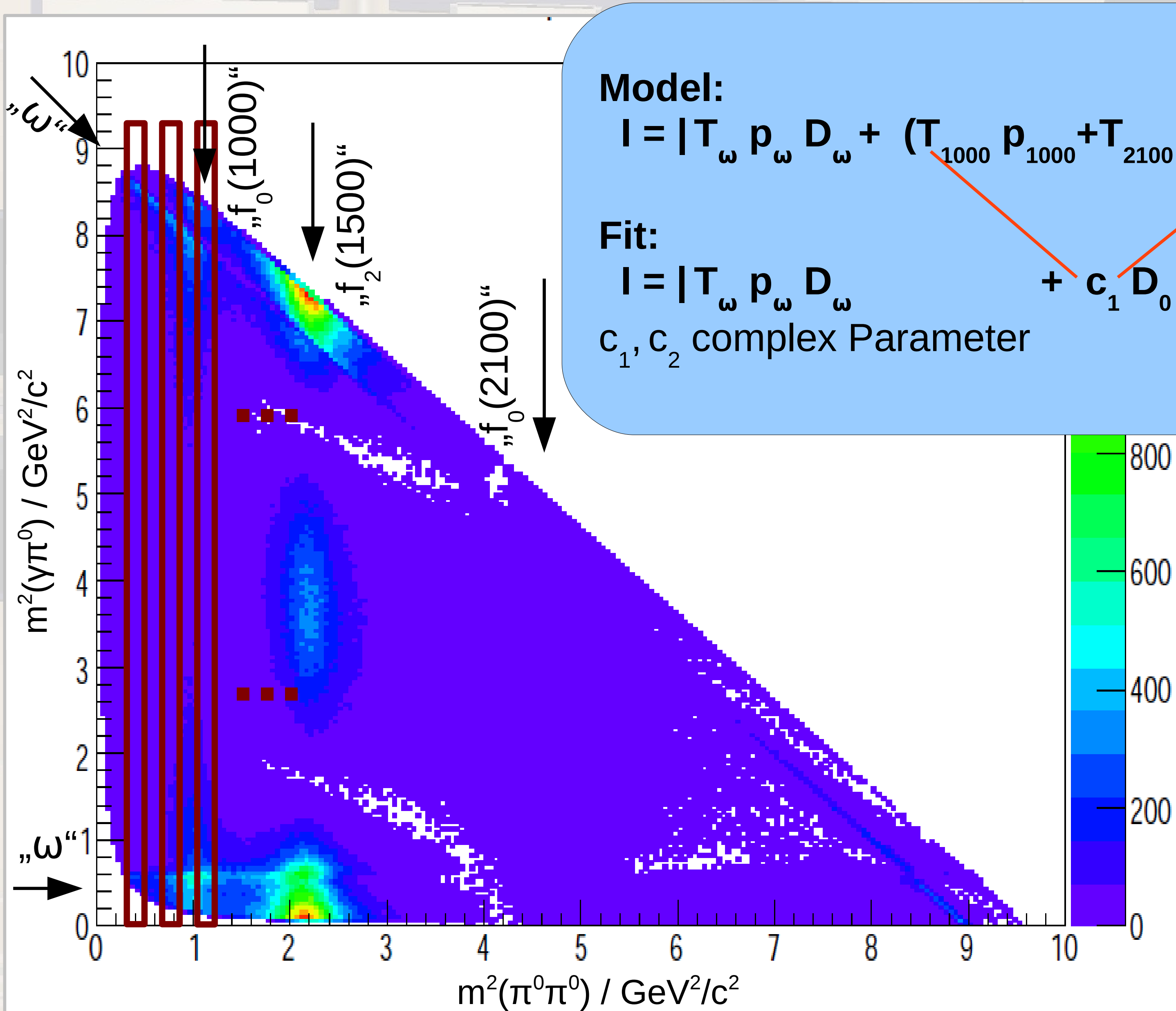


All masses <1% difference to ideal values

$J/\psi \rightarrow \gamma\pi^0\pi^0$ Wave Extraction



$J/\psi \rightarrow \gamma\pi^0\pi^0$ Wave Extraction



Model:

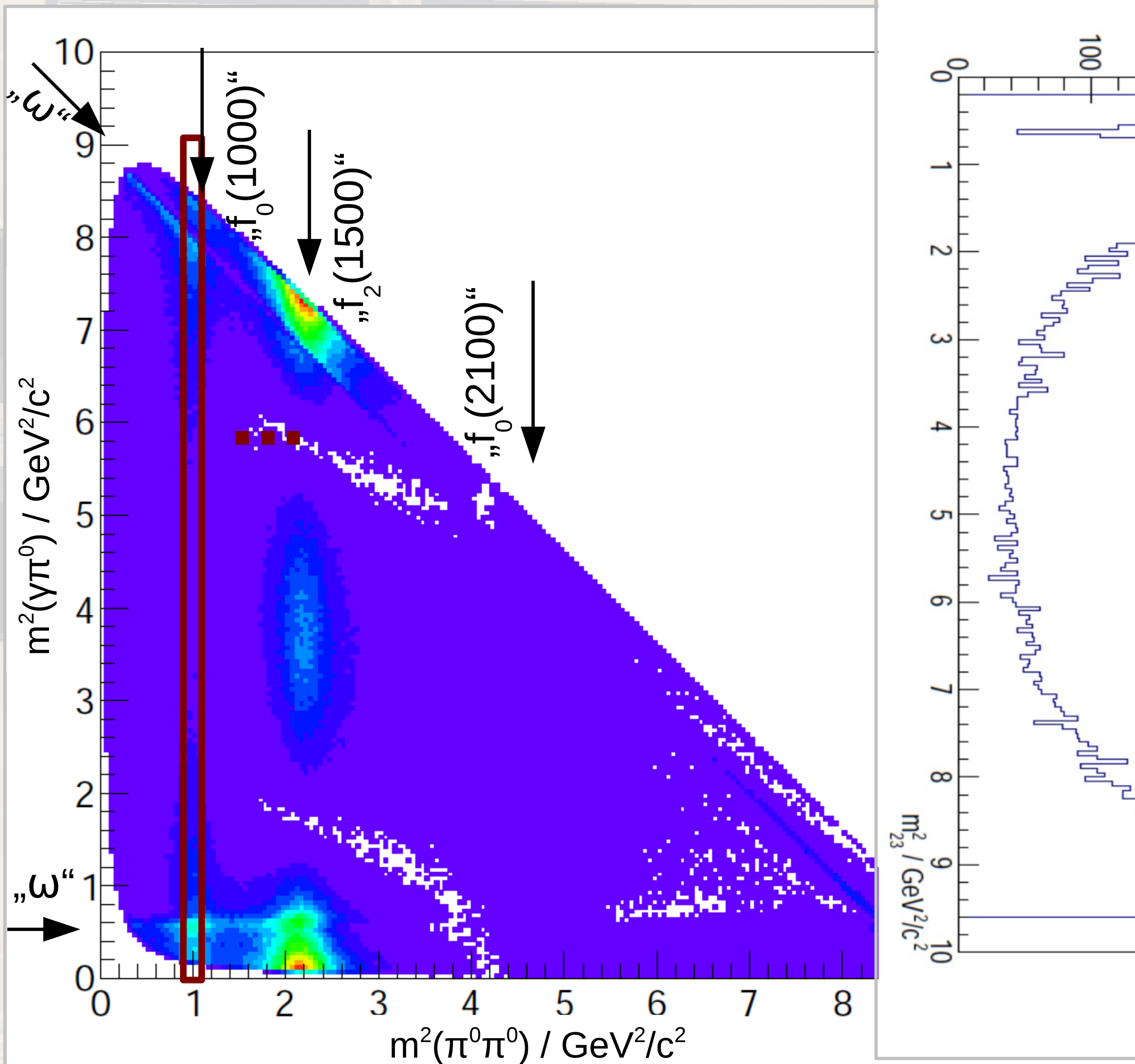
$$I = |T_\omega\, p_\omega\, D_\omega + (T_{1000}\, p_{1000} + T_{2100}\, p_{2100})\, D_0 + T_{1500}\, p_{1500}\, D_2|^2$$

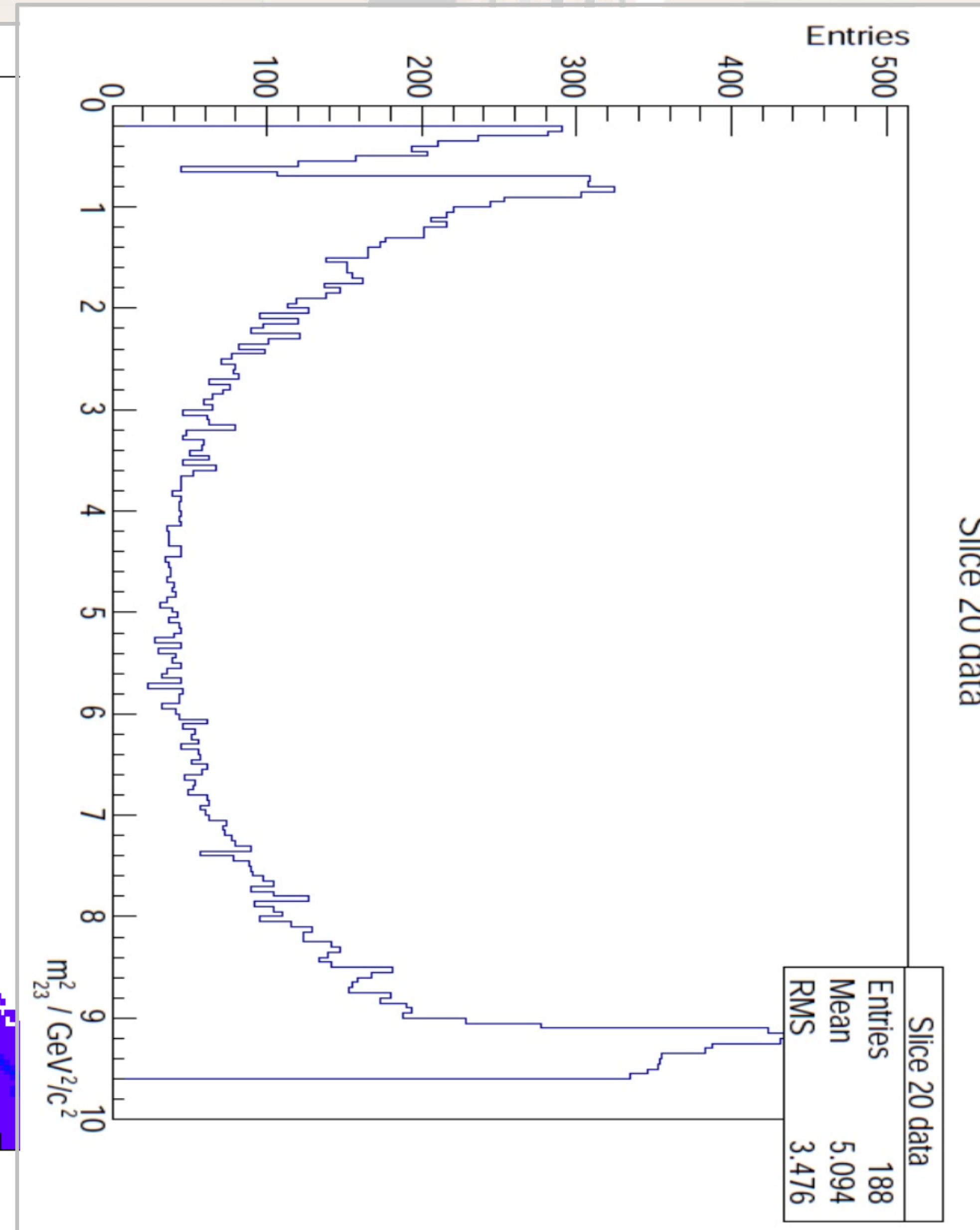
Fit:

$$I = |T_\omega\, p_\omega\, D_\omega + c_1\, D_0 + c_2\, D_2|^2$$

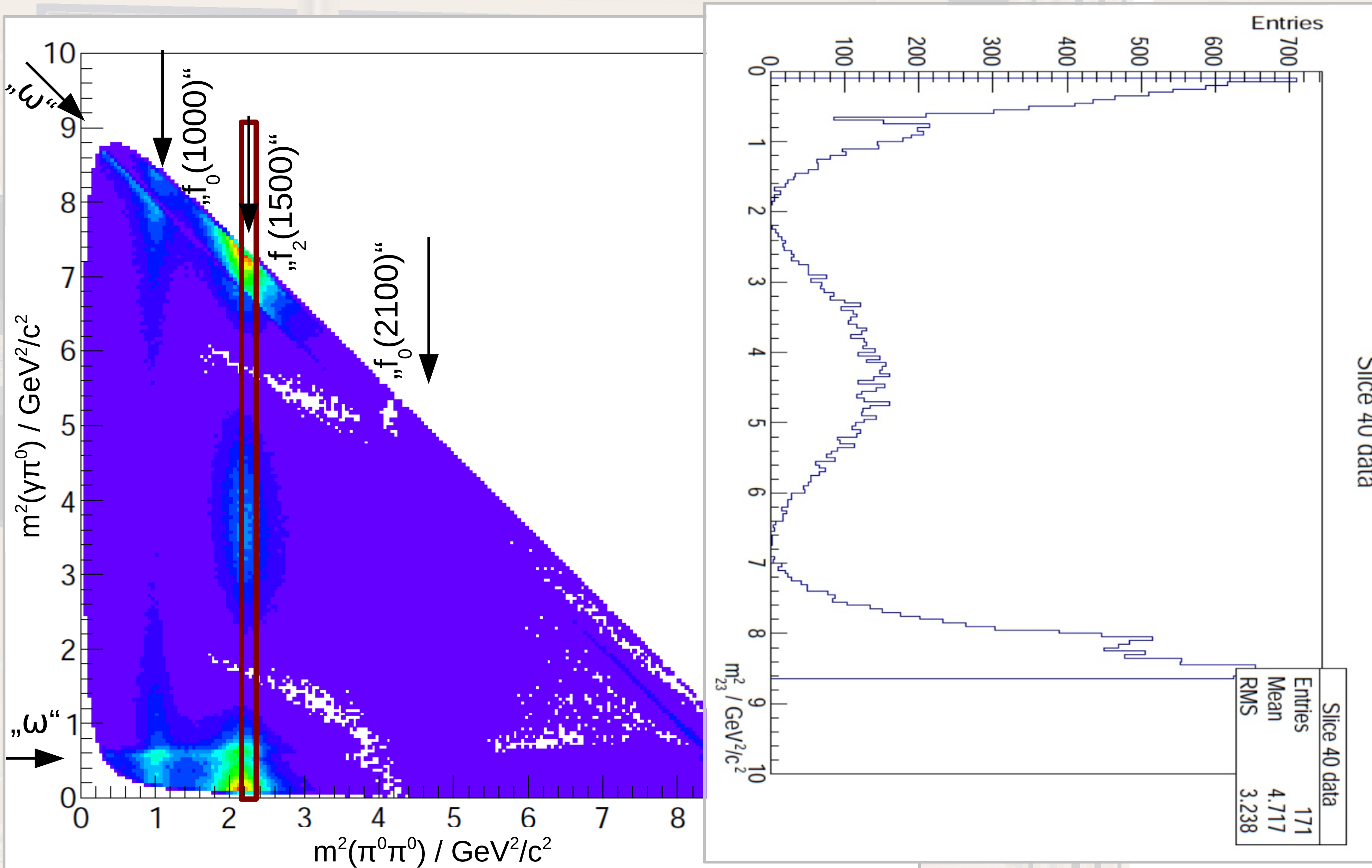
c_1, c_2 complex Parameter

Example „Slice“

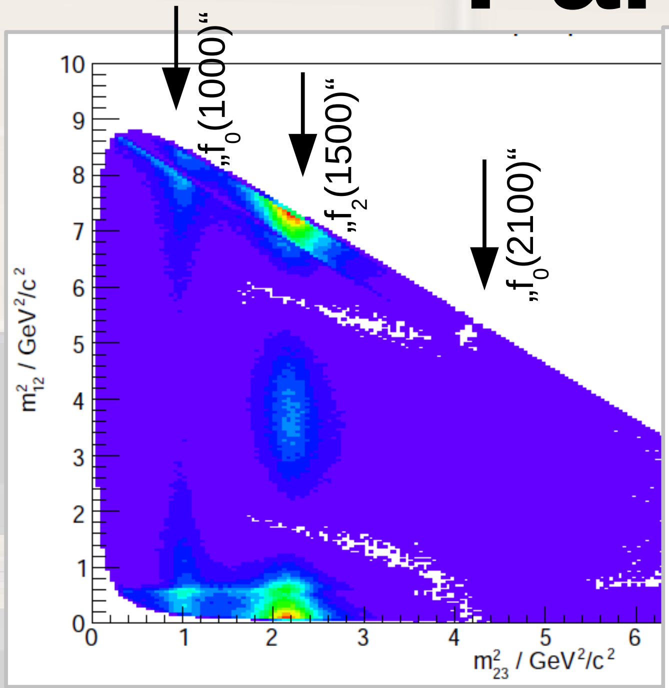


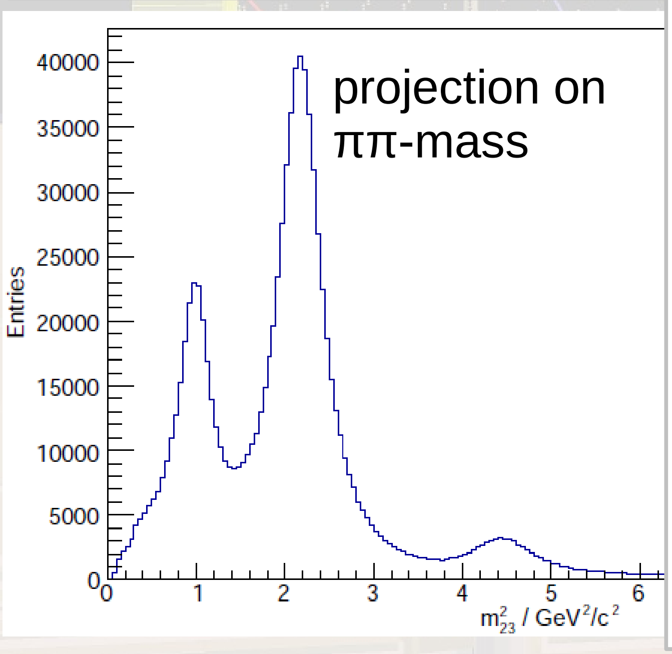


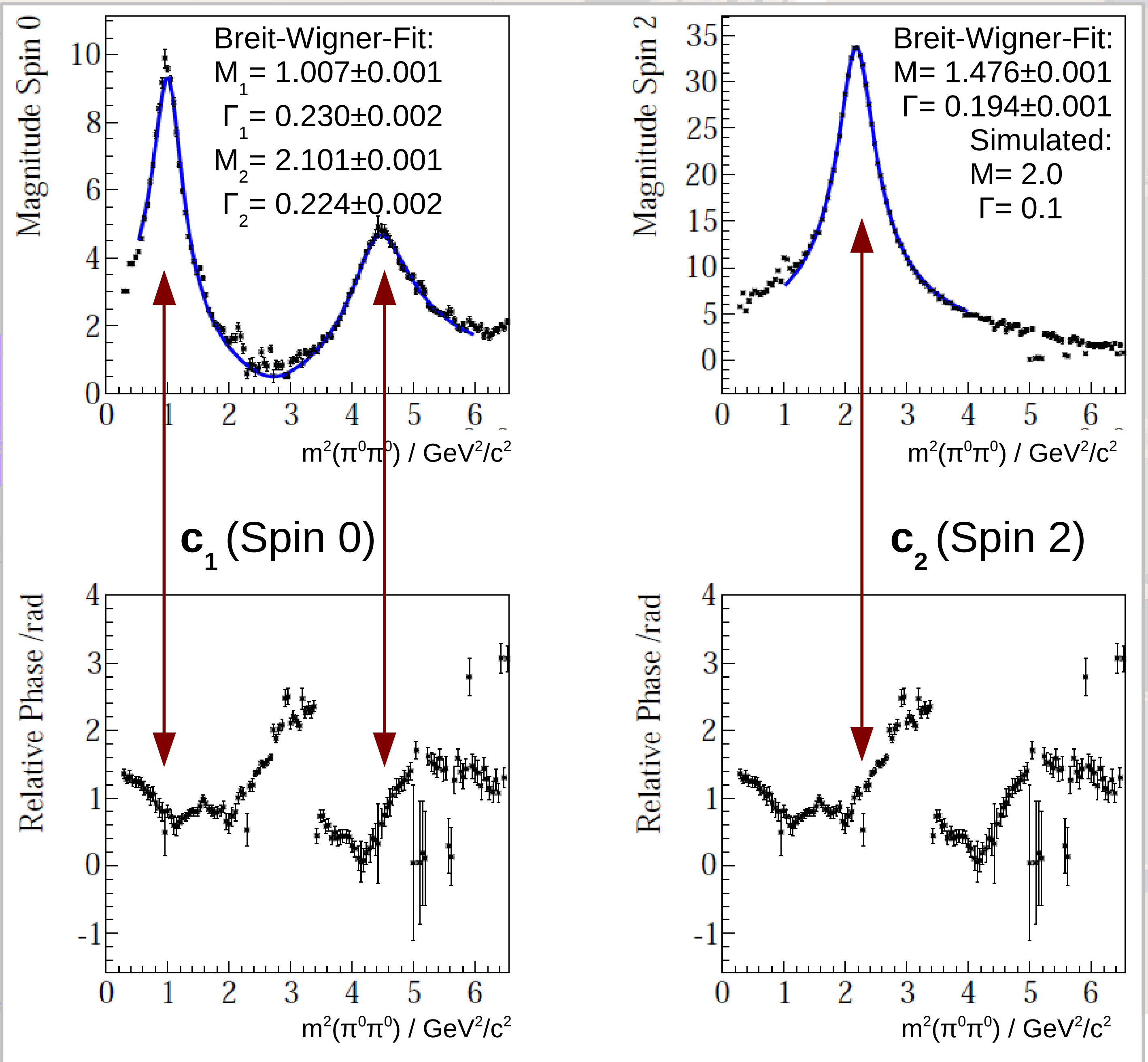
Example „Slice“



Parameter after Fit







ComPWA Status

Present:

- Abstract Interfaces

  → not fixed to methods or models
- Function Tree

  → general efficient calculation
- Wave Extraction closes in on Dalitz-Fit

  (widths still worse)

Planned:

- Parallelization
- Expert Systems for building amplitudes & decay trees
- Meta-Fit control via scripting
- Report generators

This is work in progress!

Biggest ToDo's

more physics cases

Contact

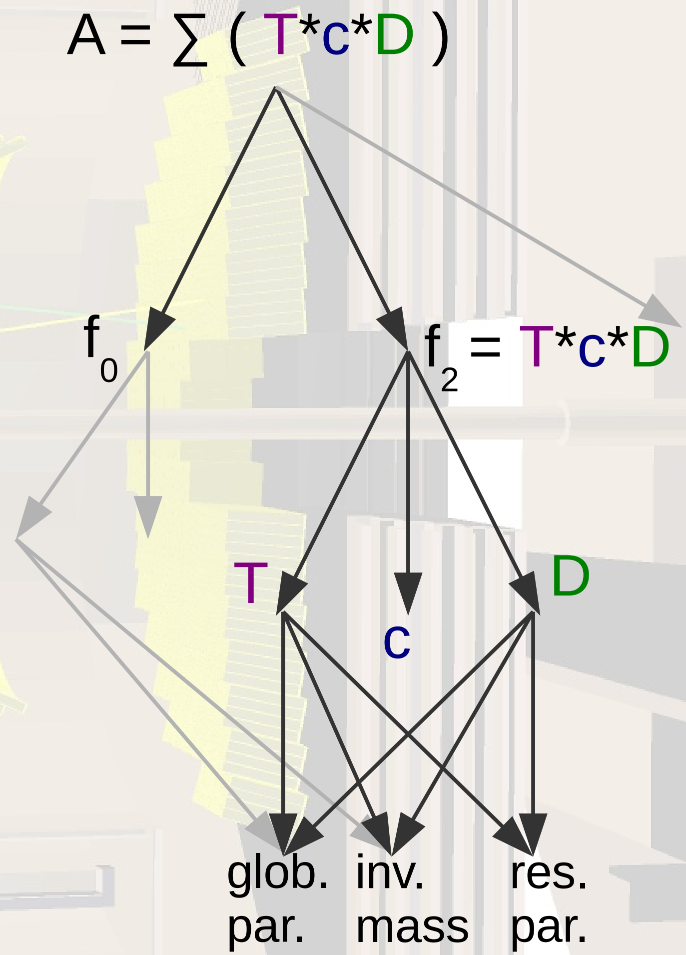
github.com/ComPWA/ComPWA
michel@kph.uni-mainz.de

Beware of unsorted Backup!

FunctionTree

$$I = \left| \sum_n T_n c_n D_n \right|^2$$

T = Breit-Wigner Function
D = D-Wigner Function
c = Complex Factor



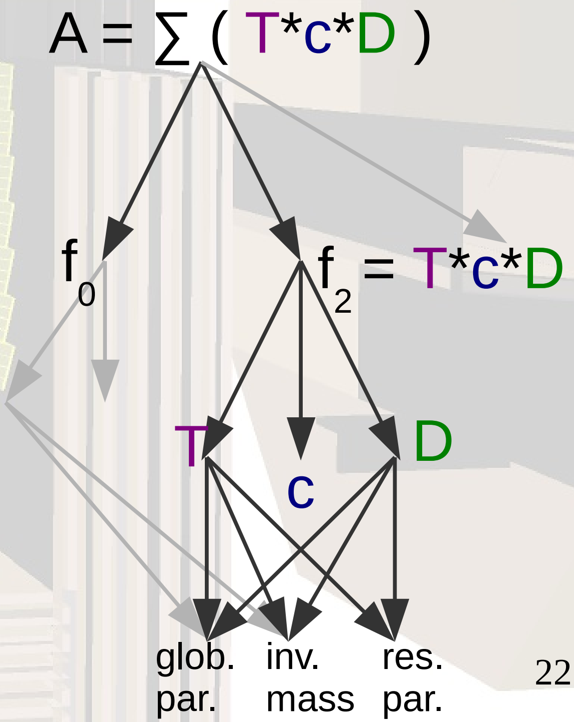
CPU-Time

Scenario (fit variables)	Avg. CPUTime per iteration without Tree	Avg. CPUTime per iteration with Tree	Speedup
1 Intensity	29.96 s	0.57 s	>50
5 Intensitie's	47.74 s	1.65 s	>25
5 BW-Width's	44.19 s	5.01 s	>8

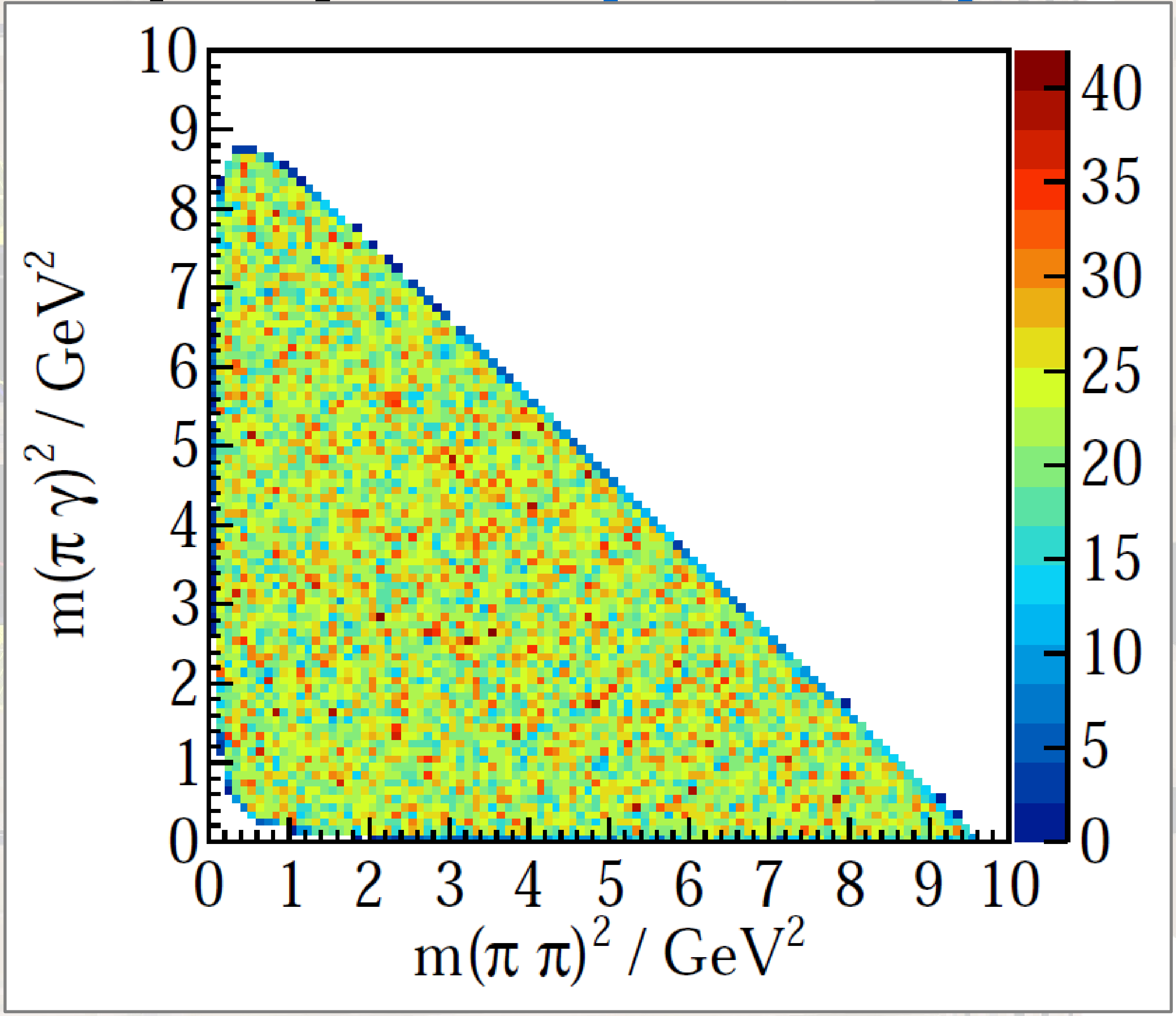
100 000 Events, 7 resonances

each fit repeated:
- 10 times without tree
- 20 times with tree

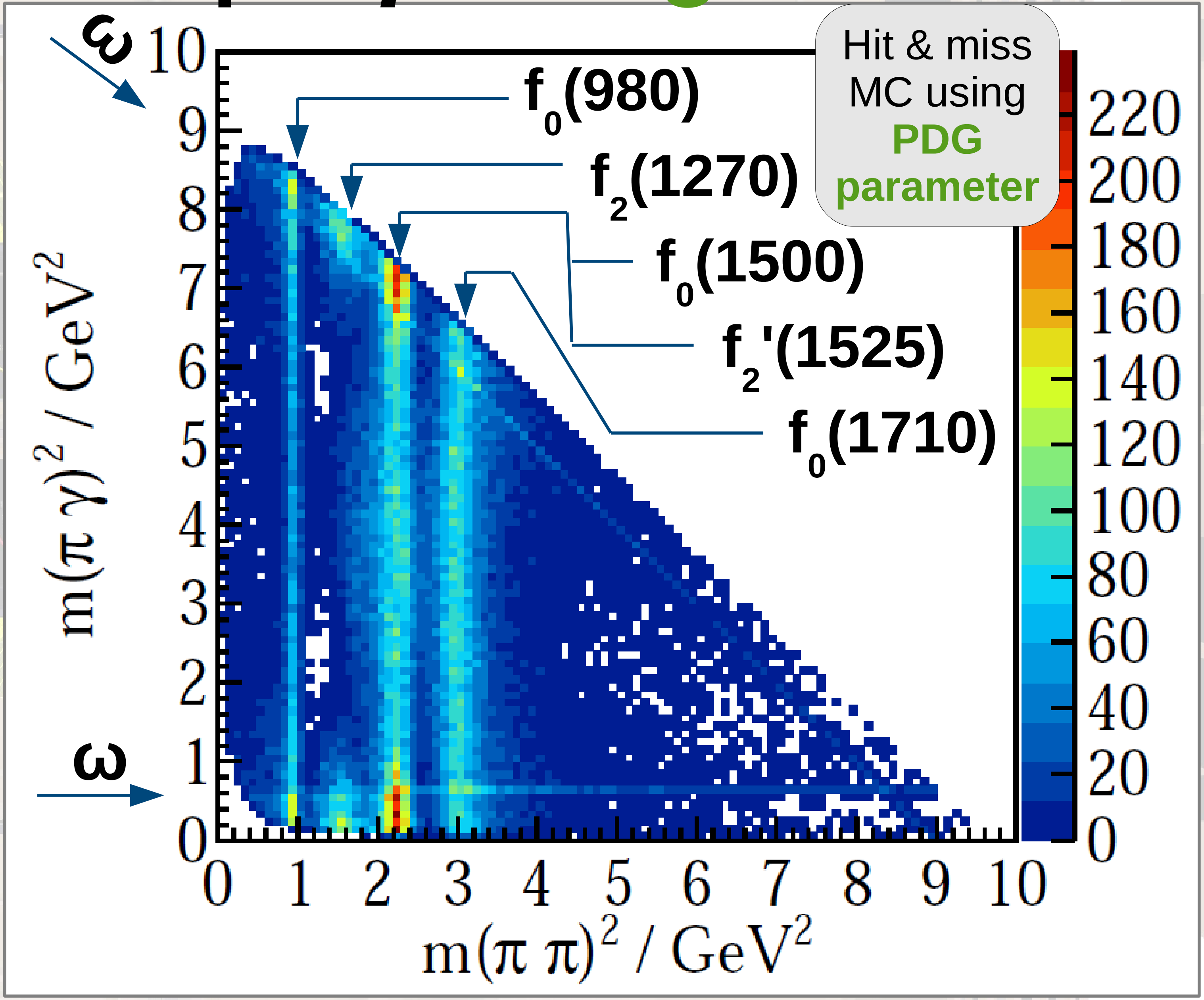
- Speedup strongly dependent on use case
- Simple decay, simple model, few resonances → small tree
- Big trees → save memory by collapsing



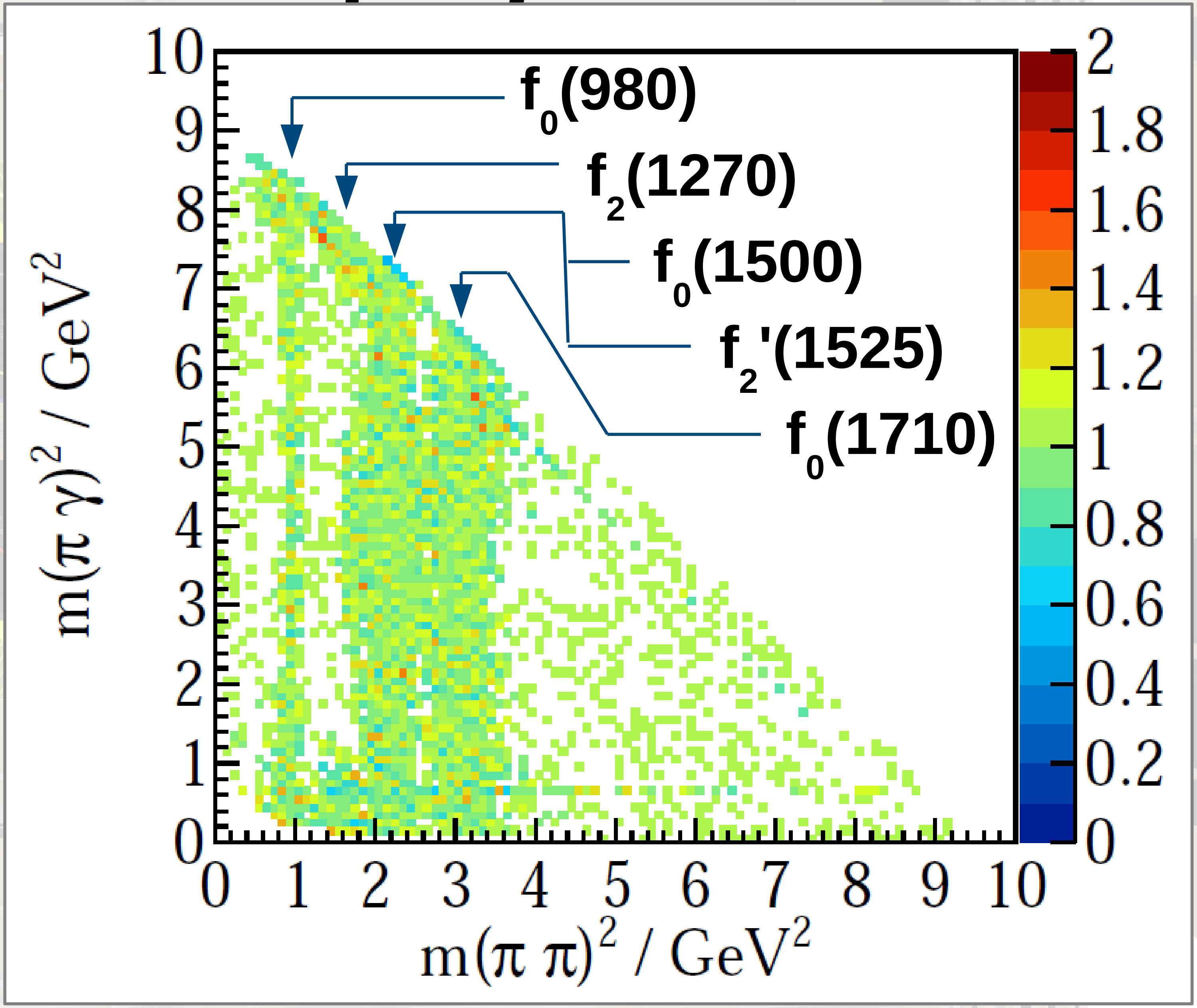
J/ψ → γπ⁰π⁰ phasespace



$J/\psi \rightarrow \gamma\pi^0\pi^0$ generated



$J/\psi \to \gamma\pi^0\pi^0$ ratio



Geneva
(Grid Enabled Evolutionary Algorithms)

- Evolutionary & Swarm Algorithms
- Gradient Descent
- Simulated Annealing

About Geneva:

Geneva is available as Open Source software (AGPL v3) from **http://www.launchpad.net/geneva,** and is also supported commercially by Gemfony scientific **(http://www.gemfony.eu)**

- Single-Thread, Multi-Thread & Networked Mode
- Simple, yet highly configurable User Interface
- Same problem description for all algorithms
  => results can be exchanged freely between algorithms

Geneva
(Grid Enabled Evolutionary Algorithms)