FDC-PWA 的GPU实现

Rong-Gang Ping
Jian-Xiong Wang
(IHEP)

Introduction

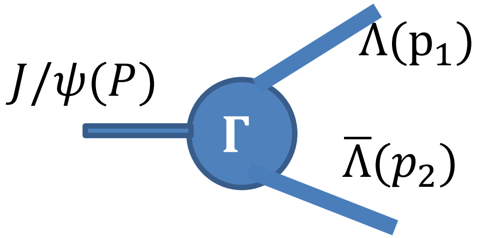
- Feynman Digram Calculation(FDC) is a general-purpose program package for Feyman diagram calculation, created by Jian-Xiong Wang from 1993.**[Jian-Xiong Wang, Nucl. Instr. Meth. Phys. Res. A534, 241(2004)]**
- FDC-PWA: FDC extension for partial wave analysis of charmonium decays(@1999)
- Two versions:

  FDC-pwa3.0-slc6: spin<9/2

  FDC-pwa3.0 : spin≤9/2:

Amplitude in FDC-PWA

- Tensor form of vertex generation by phenomenological Lagrangian (strong intera.)
  - conserve P, C parity, isospin, strangeness, charm, baryon and lepton numbers
  - an example of $J/\psi \to \Lambda\left(\frac{1}{2}^{+}\right)\overline{\Lambda}\left(\frac{1}{2}^{-}\right)$



Effective Lagrangian:

$$\mathcal{L} = \bar{u}(p_1)\,\Gamma\, v(p_2)\,\epsilon_\mu(P)$$

P, C, CPT symmetry transformation: $\mathcal{L}^P = \mathcal{L},\ \mathcal{L}^C = \mathcal{L}\ ,\ \mathcal{L} = \mathcal{L}^\dagger$

Amplitude in FDC-PWA

$$\Gamma = \gamma_\mu \ , \ \ P_\mu , \ \ p_{1\mu} , \ \ p_{2\mu} \ , \ \ \sigma_{\mu\nu} P^\nu \ , \ \ \sigma_{\mu\nu} p_1^\nu \ , \ \ \sigma_{\mu\nu} p_2^\nu \ .$$

C, P conservation vertex: $\Gamma_\mu = f_1 \gamma_\mu + f_2 \sigma_{\mu\nu} p_2^\nu$

- Note from FDC-PWA execution:

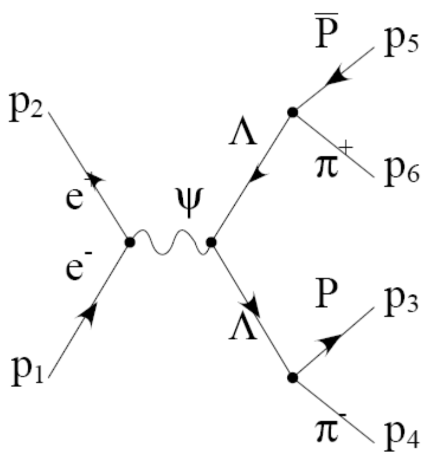


Fig. 1

$$\text{Vertex 5: } \overline{\Lambda}(p_1) \ - \Lambda(p_2) \ - \psi_\mu^0 \ (p_3)$$

$$V_5 = g \ (-2\hat{p_3}\gamma_\mu f_5 + 2\gamma_\mu \hat{p_3} f_5 + \gamma_\mu f_4)$$

where $\hat{p}_3 = p_3^\nu \gamma_\nu$

Amplitude in FDC-PWA

- Lineshape of resonance (spin j)

$$\frac{\mathcal{P}^{(j)}_{\mu_1\mu_2\ldots\mu_j,\nu_1\nu_2\ldots\nu_j}}{p^2 - m_0^2 + i\Gamma m_0}$$

Special case:

1. $j = 1$: $\mathcal{P}^{(1)}_{\mu,\nu} = -\tilde{g}_{\mu,\nu} = g_{\mu,\nu} - \frac{p_\mu p_\nu}{m_0}$
2. $j = 2$:

$$\mathcal{P}^{(2)}_{\mu_1\mu_2;\nu_1\nu_2} = \frac{1}{2}(\tilde{g}_{\mu_1\nu_1}\tilde{g}_{\mu_2\nu_2} + \tilde{g}_{\mu_2\nu_1}\tilde{g}_{\mu_1\nu_2}) - \frac{1}{3}\tilde{g}_{\mu_1\mu_2}\tilde{g}_{\nu_1\nu_2}$$

3. $j = 3$:

$$\mathcal{P}^{(3)}_{\mu_1\mu_2\mu_3;\nu_1\nu_2\nu_3} = -\frac{1}{6}\sum_{P\{\nu_1,\nu_2,\nu_3\}}\tilde{g}_{\mu_1\nu_1}\tilde{g}_{\mu_2\nu_2}\tilde{g}_{\mu_3\nu_3} + \\ + \frac{1}{30}\sum_{P\{\nu_1,\nu_2,\nu_3\}}(\tilde{g}_{\mu_1\mu_2}\tilde{g}_{\nu_1\nu_2}\tilde{g}_{\mu_3\nu_3} + \tilde{g}_{\mu_1\nu_1}\tilde{g}_{\nu_2\nu_3}\tilde{g}_{\mu_3\mu_3} + \tilde{g}_{\mu_1\mu_3}\tilde{g}_{\nu_1\nu_3}\tilde{g}_{\mu_2\nu_2})$$

4. ……

Amplitude in FDC-PWA

- Baryon cases (s_j from FDC-PWA note)

$$s_{3/2}(k,m,\mu,\nu) = \frac{i}{k^2-m^2}\frac{-2}{5}(-\gamma_{\mu_1}\hat{k}\gamma_{\mu_2} + m\gamma_{\mu_1}\gamma_{\mu_2})(\frac{1}{2}(\tilde{g}_{\mu,\mu_2}(k)\tilde{g}_{\mu_1,\nu}(k) + \tilde{g}_{\mu,\nu}(k)\tilde{g}_{\mu_1,\mu_2}(k))$$

$$+\frac{-1}{3}(\tilde{g}_{\mu_2,\nu}(k)\tilde{g}_{\mu_1,\mu}(k)))$$

$$\tilde{g}_{\mu,\nu}(k) = -g_{\mu,\nu} + \frac{k_\mu\ k_\nu}{K^2}$$

$$\begin{aligned} s_{5/2}(k,m,\mu,\nu,\alpha,\beta) = \frac{i}{k^2-m^2}\frac{-3}{7}(-\gamma_{\mu_1}\hat{k}\gamma_{\mu_2} + m\gamma_{\mu_1}\gamma_{\mu_2})(\frac{1}{6}(\tilde{g}_{\nu,\mu_2}(k)\tilde{g}_{\mu,\alpha}(k)\tilde{g}_{\mu_1,\beta}(k) \\ +\tilde{g}_{\nu,\alpha}(k)\tilde{g}_{\mu,\mu_2}(k)\tilde{g}_{\mu_1,\beta}(k) + \tilde{g}_{\nu,\mu_2}(k)\tilde{g}_{\mu,\beta}(k)\tilde{g}_{\mu_1,\alpha}(k) + \tilde{g}_{\nu,\beta}(k)\tilde{g}_{\mu,\mu_2}(k)\tilde{g}_{\mu_1,\alpha}(k) \\ +\tilde{g}_{\nu,\alpha}(k)\tilde{g}_{\mu,\beta}(k)\tilde{g}_{\mu_1,\mu_2}(k) + \tilde{g}_{\nu,\beta}(k)\tilde{g}_{\mu,\alpha}(k)\tilde{g}_{\mu_1,\mu_2}(k)) + \frac{-1}{15}(\tilde{g}_{\mu_1,\mu_2}(k)\tilde{g}_{\alpha,\beta}(k)\tilde{g}_{\mu,\nu}(k) \\ +\tilde{g}_{\mu_1,\alpha}(k)\tilde{g}_{\mu_2,\beta}(k)\tilde{g}_{\mu,\nu}(k) + \tilde{g}_{\mu_1,\beta}(k)\tilde{g}_{\mu_2,\alpha}(k)\tilde{g}_{\mu,\nu}(k) + \tilde{g}_{\mu,\mu_2}(k)\tilde{g}_{\alpha,\beta}(k)\tilde{g}_{\mu_1,\nu}(k) \\ +\tilde{g}_{\mu,\alpha}(k)\tilde{g}_{\mu_2,\beta}(k)\tilde{g}_{\mu_1,\nu}(k) + \tilde{g}_{\mu,\beta}(k)\tilde{g}_{\mu_2,\alpha}(k)\tilde{g}_{\mu_1,\nu}(k) + \tilde{g}_{\nu,\mu_2}(k)\tilde{g}_{\alpha,\beta}(k)\tilde{g}_{\mu_1,\mu}(k) \\ +\tilde{g}_{\nu,\alpha}(k)\tilde{g}_{\mu_2,\beta}(k)\tilde{g}_{\mu_1,\mu}(k) + \tilde{g}_{\nu,\beta}(k)\tilde{g}_{\mu_2,\alpha}(k)\tilde{g}_{\mu_1,\mu}(k))) \end{aligned}$$

$s_{7/2}(k,m,\mu,\nu,\alpha,\beta,\gamma,\eta) =$ … (about 110 terms)

Fit package of FDC-pwa

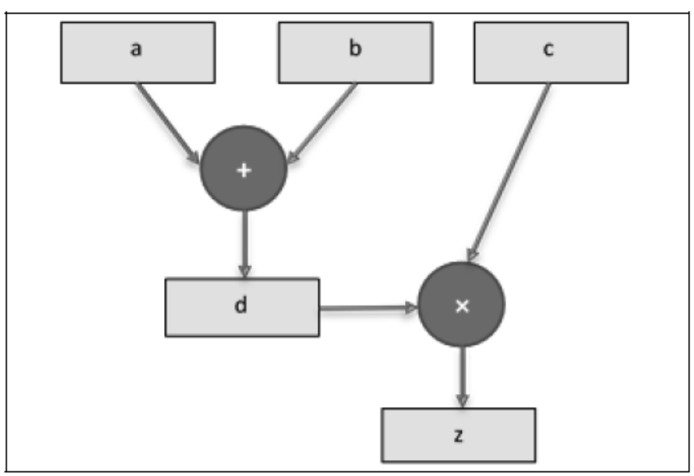
- create model (resonance list) from data hints
- generate tree level Feynman diagram and produce amplitude in Fortran code
- MLLH Minimized with Fumili package
- Normalization factor calculated in the reduce amplitude (save more time)

$$|\mathcal{M}|^2 = \sum_{j=1}^{n_{par}} \sum_{i=1}^{N_{mc}} c_j A_j = c_j a_j, \text{with } a_j = \sum_{i=1}^{N_{mc}} A_j$$

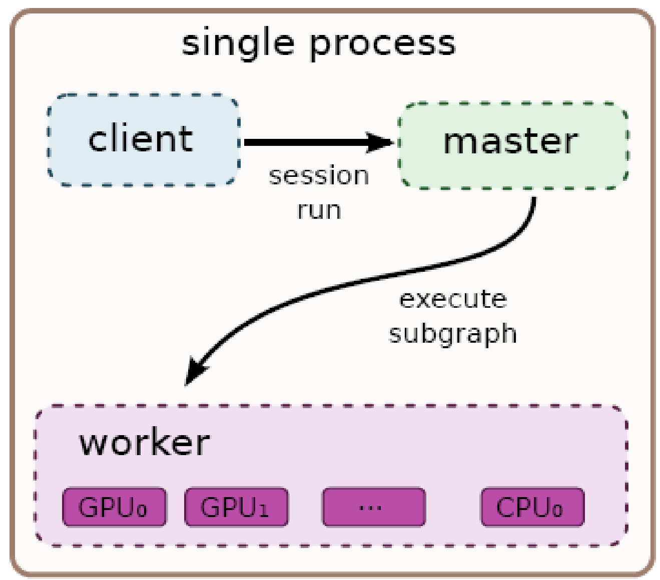
- But loop over the data events event by event
  (most cpu time consuming)
- Mass and width not enter into the fit parameter list
- Access the fit projection (dplot.hbook) and resonance ratio (mplot.info)

Why tensorflow?

- Tensorflow (tf) popular in AI community
- After tensorflow2.x, it has eager execution mode, make it suitable for scientific computation
- Tensor data model applicable to amplitude calculation
- Autograph functionality



$$z = c \times (a + b)$$



Accelerate FDC with GPU-Tensorflow

- Compile amplitudes of fortran codes into a python modules
- Calculate the event amplitude in Tensorflow framework
- MLLH minimized with iMinuit (python version of Minuit)
- Access fit results (signal yields and statistical uncertainty calculated based on resultant covariance matrix)
- Allow user to add mass and width as hyper-parameters in the fit
- Allow for simultaneous fit to multiple samples

Tensor algorithm of amplitude

- $|\mathcal{M}(\mathrm{event_v})|^2 = \bar{\Sigma}_{s_1,\ldots,s_j} \left|\Sigma_k c_k\, a_{\mathrm{v},k}\right|^2$
  $= C_{k,l}\, A_{\mathrm{v},k,l}$ (dumb index rule)
  with $C_{k,l} = c_k\, c_l^*$, $A_{\mathrm{v},k,l} = \bar{\Sigma}_{s_1,\ldots,s_j}(a_{\mathrm{v},k} a_{\mathrm{v},l}^*)$
  c_k : k-th parameter,
  $a_{\mathrm{v},k}$: k-th term of amplitude for event v
- $A_{\mathrm{v},k,l}$ calculated by FDC, and stored in memory (limitation form GPU memory)
- Amplitude reduction in tensorflow

Structure of FDC-TF package

FDC: generate amplitude

FDC python module

TensorFlow:
MLLH calculation

iMinuit:
minimize MLLH

Signal yields, statistical uncertainty.

Matplotlib:
Fill histograms

Dependent modules:

- Numpy, matlibplot, iminuit, Scipy, Tensorflow2.0
- Python version: 3.7

An example of users' script

```
from SMLfit import SMLfit
import tensorflow as tf
import numpy
import os
import warnings
import timeit as tt
import FDC
#here invoke FDC equivalent FDC in Amps

EvtPDL = Script()
EvtPDL.readPdtTable('reson.inp') # pdt.table
EvtPDL.readParaList('fpara.inp') # para.list
EvtPDL.setFinalState(['p','pbar','K^+','K^-'])
EvtPDL.setEvtFileMC(['p4.mc'])
EvtPDL.setEvtFileDT(['p4.dt'])
EvtPDL.setEvtFileBG(['p4.bg'])
EvtPDL.setAddWidth([['5','Gx0',0.005,0.004,0.08]])
#### SMLfit ####
with tf.device("/device:gpu:0"):
   myfit = SMLfit(EvtPDL)
   print('try scan 1 para....')
   myfit.scan(1)
   myfit.exec('migrad')
   myfit.writeParaList('myfit.list',myfit.exec('np_values'),myfit.exec('np_errors'))
   myfit.write_totMCamps('totAmps.npz',0)
   if myfit.exec('valid'): numpy.save('mycov',myfit.exec('np_covariance'))
   print('FVAL= ',myfit.exec('fval'))
   print('values ',myfit.exec('np_values'))
   print('errors ',myfit.exec('np_errors'))
   myfit.writeEvtMass('p4.mc','mij.mc')
   myfit.writeEvtMass('p4.dt','mij.dt')
   myfit.writeEvtMass('p4.bg','mij.bg')
   myfit.writeModeEvtAmps('mmEvt.npz')
   myfit.calSta(0)
```

Performance test

- GPU : Tesla V100-SXM2-32GB
- decay: $\psi' \to pK^-\overline{\Lambda} + c.c.$
  - 5969 data events, and 80,000 PHSP events. 179 parameters in the fit.
- 24 resonances included in the fit

$N^*(1710), N^*(1870), N^*(1720), \Lambda(1810),$
$\Lambda(1800), \Lambda(1670), \Lambda(1600), \Lambda(1405), \mathrm{K}_1(2075)$
$\mathrm{N}^*(2060), \Lambda(2325), \Lambda(1890), \Lambda(1690), \Lambda(1520)$
$\mathrm{K}_2(2250), \mathrm{N}^*(1990), \mathrm{N}^*(2190), \Lambda(2110), \Lambda(1830)$
$\Lambda(1820), \mathrm{N}^*(2250), \Lambda(2100), \Lambda(2020), \Lambda(2350)$

Performance test (cont.)

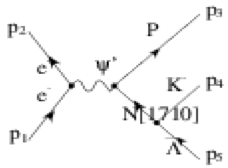


Fig. 1

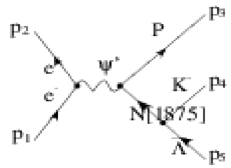


Fig. 2

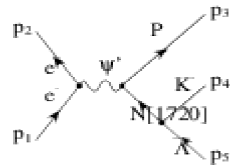


Fig. 3

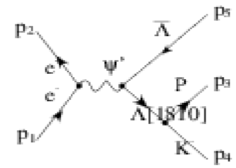


Fig. 4

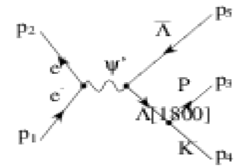


Fig. 5

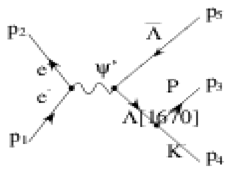


Fig. 6

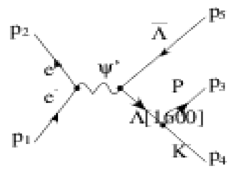


Fig. 7

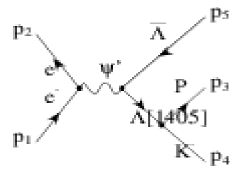


Fig. 8

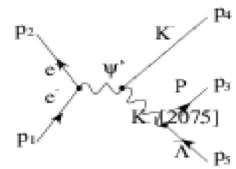


Fig. 9

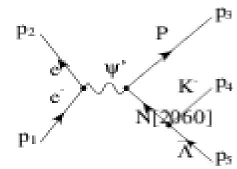


Fig. 10

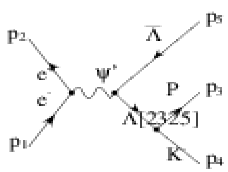


Fig. 11

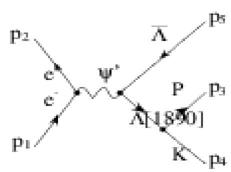


Fig. 12

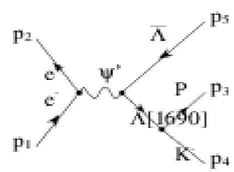


Fig. 13

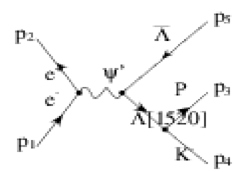


Fig. 14

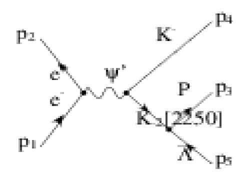


Fig. 15

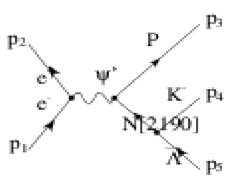


Fig. 16

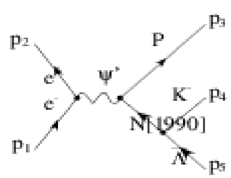


Fig. 17

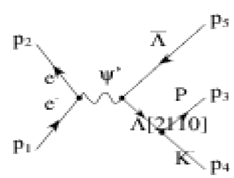


Fig. 18

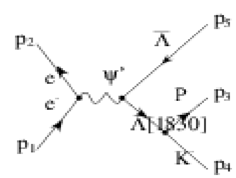


Fig. 19

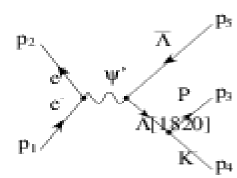


Fig. 20

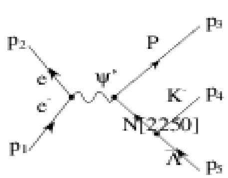


Fig. 21

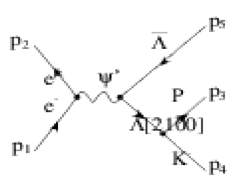


Fig. 22

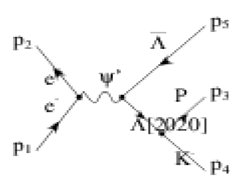


Fig. 23

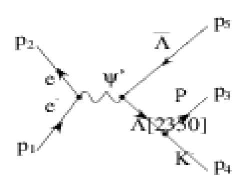
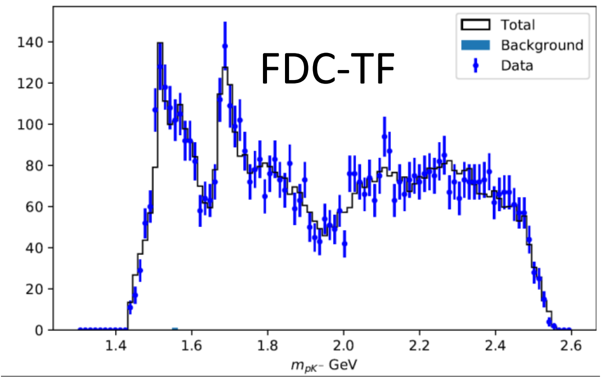


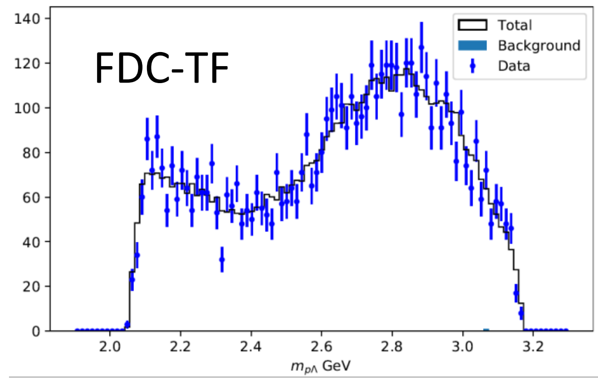
Fig. 24

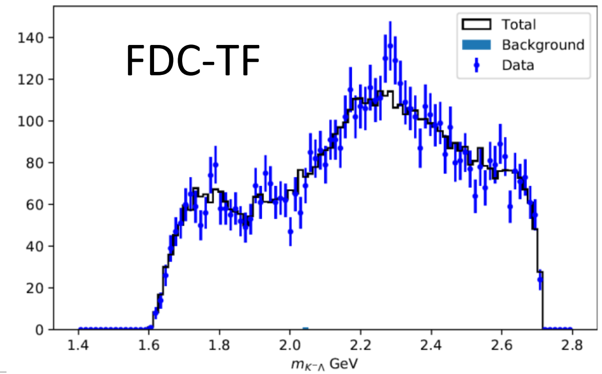
Performance test (cont.)

- $\frac{t_c}{t_g} = \frac{30}{0.07} \approx 430$: $t_c(t_g)$ times cost for CPU (GPU) calculation per iteration

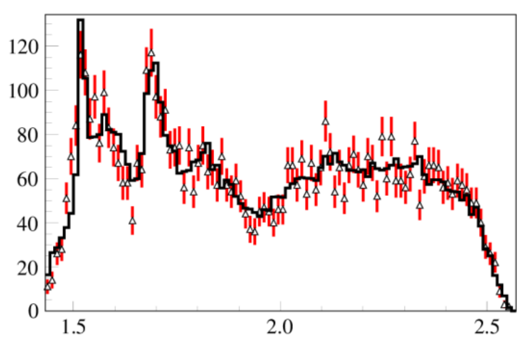
FDC-TF: $-\ln L =$ -789.45

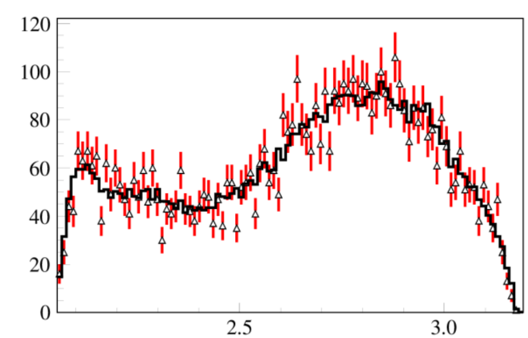


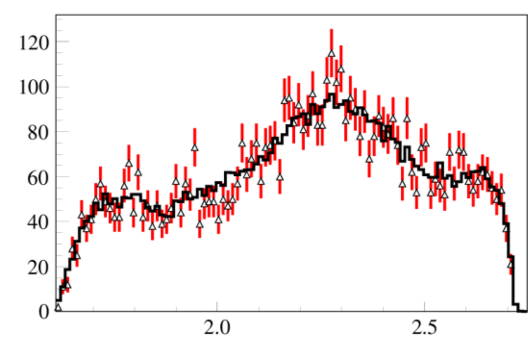




FDC: $-\ln L = -789.10$







Performance test (cont.)

- Check on yields ratios

Mode	FDC	FDC-tf
1	0.034	0.034
2	0.007	0.007
3	0.187	0.187
4	0.069	0.069
5	0.116	0.116
6	0.018	0.018
7	**0.122**	**0.123**
8	0.050	0.050
9	0.051	0.051
10	0.041	0.041
11	0.035	0.035
12	0.040	0.040

Mode	FDC	FDC-tf
13	0.042	0.042
14	0.021	0.021
15	0.056	0.056
16	0.523	0.523
17	0.005	0.005
18	0.614	0.611
19	0.034	0.034
20	0.023	0.023
21	0.003	0.003
22	0.137	0.138
23	0.009	0.009
24	0.011	0.006

See /hpcfs/bes/gpupwa/pingrg/fdc/pkl2/process/fort_dbg

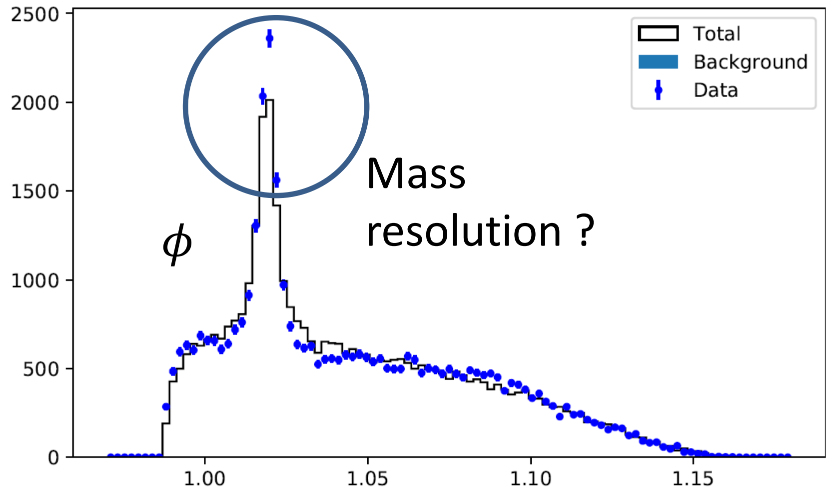
Performance test (cont.)

- $J/\psi \rightarrow K^+K^-p\bar{p}$
  - 38,709 data events, 222,256 PHSP events
  - 55 parameters for 7 diagrams involved in the fit

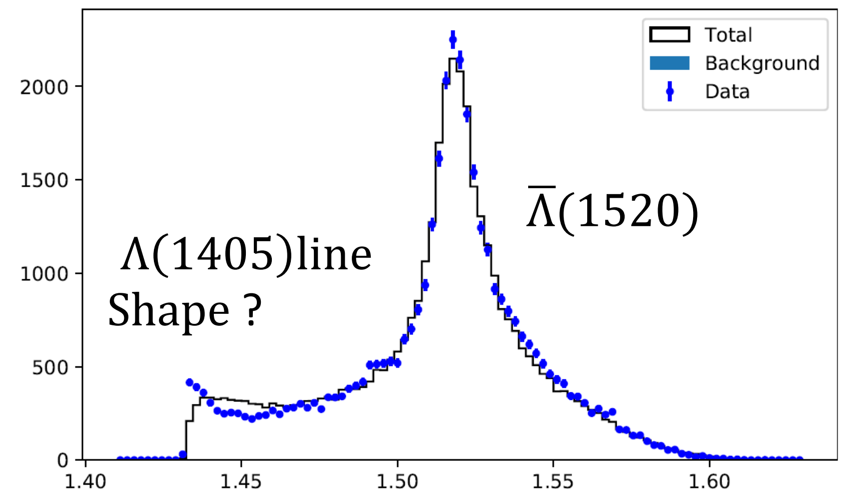
$$
\begin{aligned}
J/\psi \rightarrow\ & \phi\, p\bar{p}\ (0^+, 2^+) \\
\rightarrow\ & f_0\, p\bar{p}(1^-) \\
\rightarrow\ & \Lambda(1405)\bar{\Lambda}(1405) \\
\rightarrow\ & \Lambda(1405)\bar{\Lambda}(1520) \\
\rightarrow\ & \Lambda(1520)\bar{\Lambda}(1405) \\
\rightarrow\ & \Lambda(1520)\bar{\Lambda}(1520)
\end{aligned}
$$

- speed: $\sim$ 48 iterations/second, spend 8 minutes for one fit

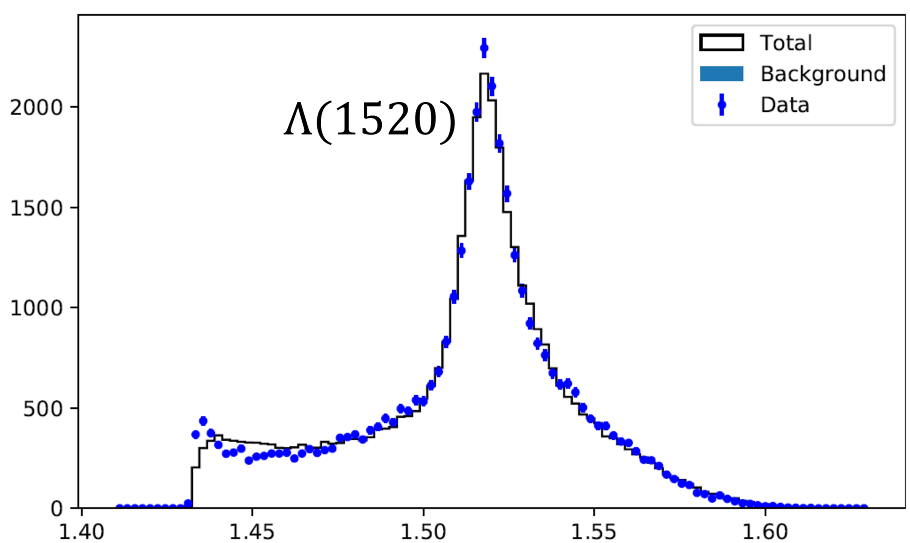
Performance test (cont.)



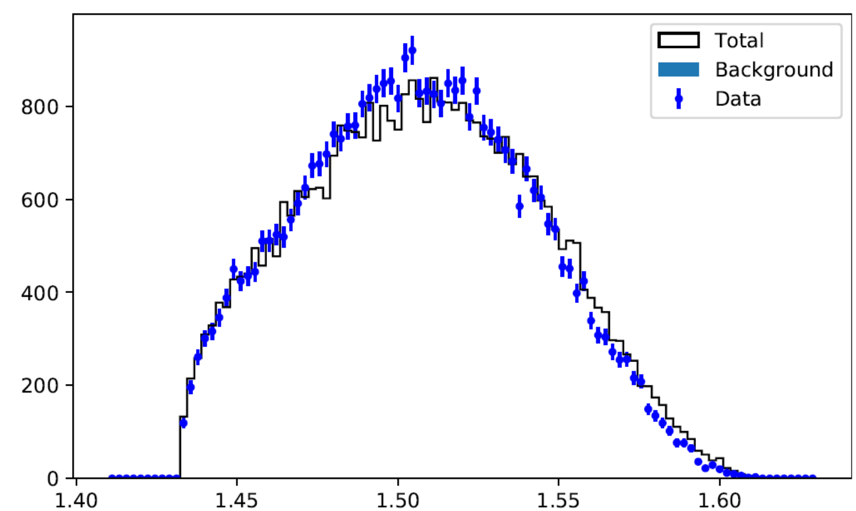
$M(K^+K^-)$ GeV



$M(\overline{p}K^+)$ GeV



$M(pK^-)$ GeV



$M(\overline{p}K^-)$ GeV

Meson final state

- $J/\psi \to \phi\eta\eta$

  14 resonance,

  91 parameters

  $N_{data} = 9873,$
  $N_{mc} = 182238,$
  $N_{bg} = 343$

  $t_g = 38$minuites

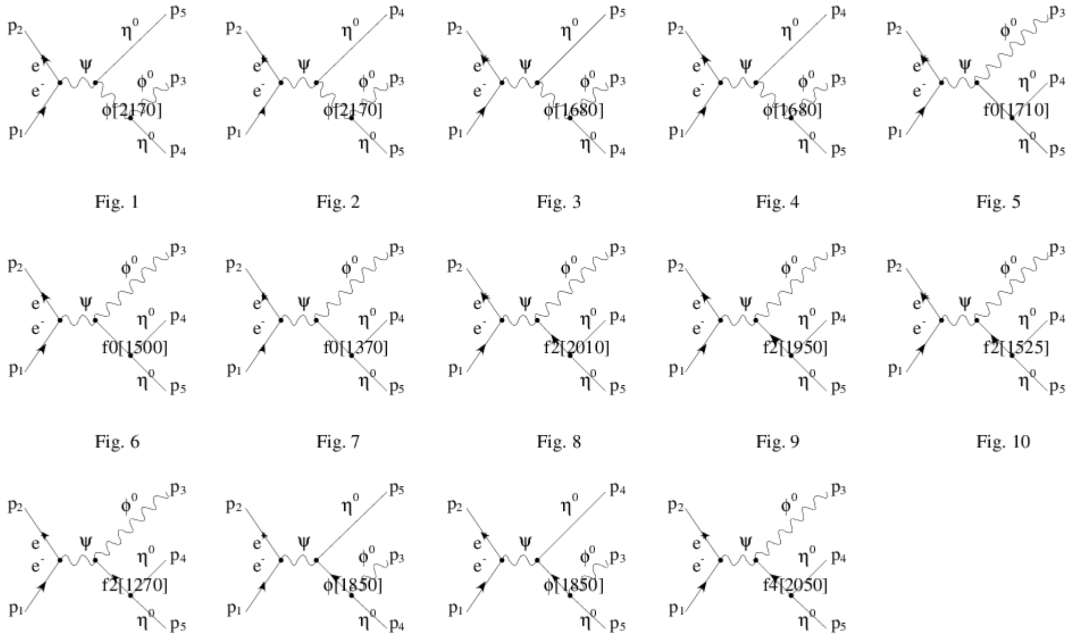
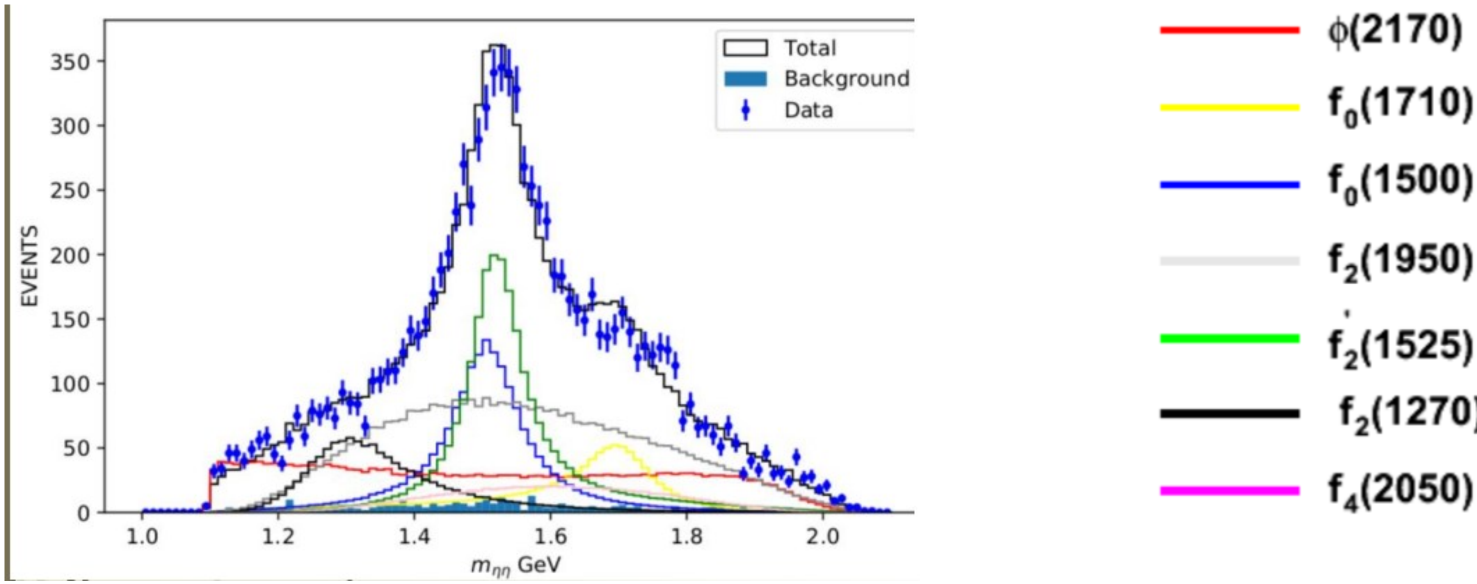
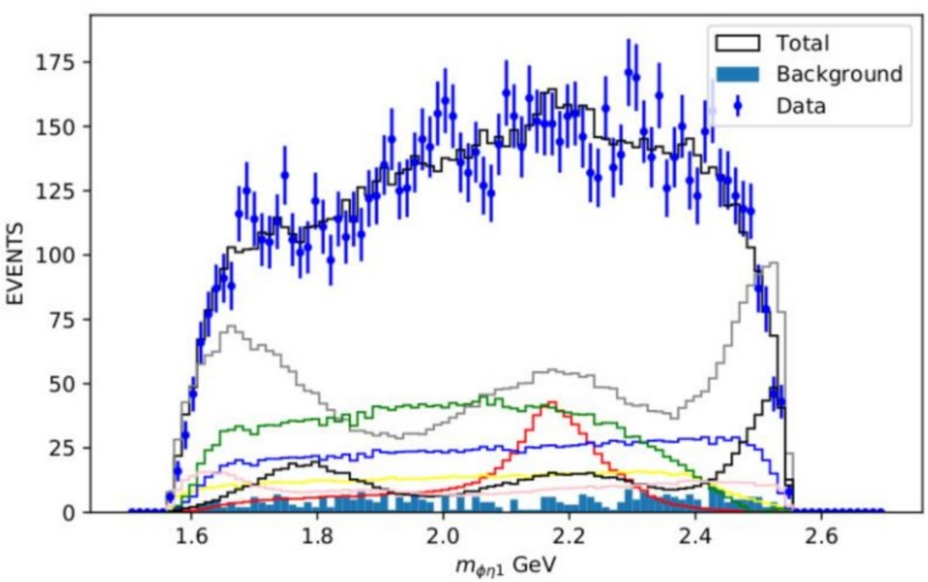
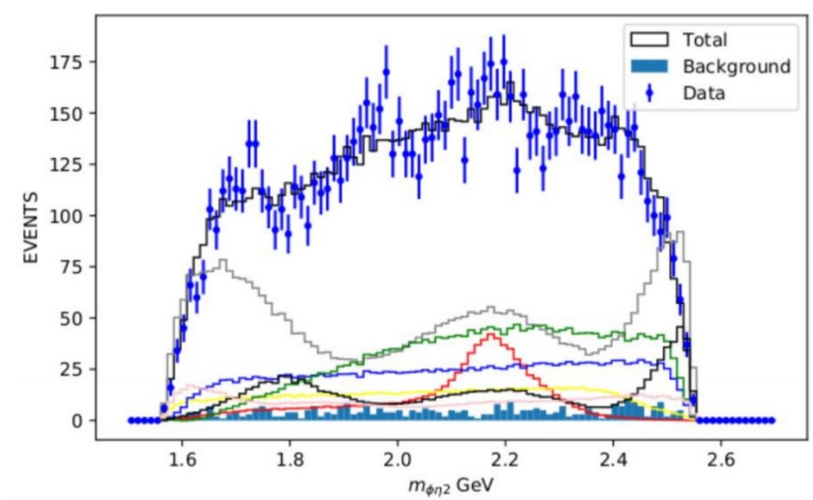


Fig. 1, Fig. 2, Fig. 3, Fig. 4, Fig. 5, Fig. 6, Fig. 7, Fig. 8, Fig. 9, Fig. 10, Fig. 11, Fig. 12, Fig. 13, Fig. 14

$J/\psi \to \phi\eta\eta$







Summary and outlook

- A GPU-TesorFlow package is created to accelerate the FDC calculation
- Allow add mass and width as parameters in the fit, and high speed performance achieved.
- improvement in the future:
  - ➢ design toymc app.
  - ➢ future FDC-paw4.0 release:

    hyperon parity-violation decay, extension to any parent particle decays, FDC-python module

生命短暂，分波需要FDC来助战!