PyPWA Manual - Simulation

Step-by-step guide to PWA (mass-independent data simulation) using PyPWA
(using the scientific JLab batch farm)
(February 2015)

J. Pond and C. Salgado
Norfolk State University
and
The Thomas Jefferson National Accelerator Facility

All software used by PyPWA can be downloaded from https://pypwa.jlab.org/. Installing PyPWA software needs full internet access i.e go to jlabs1 as ifarm1102 doesn't have internet access.

For your new installation: Create a PyPWA directory in your home directory.

Untar your software.

NOTE: *It is advised that you read the general documentation, in the wiki PyPWA, or the website before proceeding. For an overview of the general formalism consult: C. W. Salgado and D. P. Weygand, Physics Report 537 (2014) 1-58 and references within.*

These are detailed step-by-step instructions for SIMULATION using the Isobar-model based partial waves formalism:

There are two ways to perform simulations:

(A) You select a set of resonances, and for each resonance a set of waves, decaying to the final state that you want to simulate. Then, you will simulate mass distributions and produce sets of events in "gamp" format (raw/generated) and accepted. NOTE: resolution is not yet incorporated into the framework.

(B) After you have fitted the data in a PWA (fitting). You can simulate sets of events in "gamp" format (raw/generated) and accepted) according to the waves and production amplitude (V's) that you obtained. The angular distributions of the simulated events can be then compared with your data to check your fits.

We describe option **(A)** first:

Requirements (You'll need your own software for these steps):

1) Run a full monte carlo simulation (generate + Geant(detector simulation) + Reconstruction(and analysis)) using a flat phase-space generator (you can simulate using your reaction's t distribution). Create a gamp formatted file: **raw_events.gamp** with all the generated events. Create a pass/fail file **events.pf** (with 0 (fail) and 1 (pass) in each line and the same order than the gamp file)**.**

This file will map the accepted events after the full simulation (for weighting of the events without resubmitting new simulation).

2) Create a "**resonances.txt**" file with the resonances you want to simulate. You need to choose a set of resonances and the weight of each resonance.

For each resonance you can pick one or more waves from your wave set (in your keyfiles/ directory). For a given resonance the waves will be given a weight (0 to 1) for the participation in the resonance production.

Determine all waves you want to use and write a *name*.keyfile for each according to the gamp format (see general documentation).
Populate the MAIN/ directory with all your waves as for example: 0-1--1-P_rho.keyfile
(*name* includes wave definition: IGJPCMepsL_(isobar-if any).keyfile)

After this is done you are ready to start running farm jobs.

For example if you want to generate a1_1(1260),a_2(1320), pi_1(1600) and pi_2(1670), with a 27%, 45%, 6% and 22% relative weights. Suppose also that you have four waves P1, D1, P0, D1 and you want to assign only one of the waves to each of the resonances. For example:

#resonances.txt is the control file that simulatorMain uses to get resonances to simulate from waves.
Add resonances starting on line 7 in the format cR (float) wR (list) w0 (float) r0 (float) phase (float)
The sum of all cR for /all/ resonances needs to be 1.0 and the sum of the wR list for /each/ resonance needs to be 1.0
Ex: 1.0 1.0,0.0 1.0 0.5 0.0
One resonance to a line and use the # to comment.
#

```
0.27 1,0,0,0 1260. 300. 0.
0.45 0,1,0,0 1320. 107. 0.
0.06 0,0,1,0 1600. 234. 0.
0.22 0,0,0,1 1670. 259. 0.
```

You are ready to start using the PyPWA software.

[1] Log in to a CUE machine (e.g. ifarm1102). (Note: You may need to contact your hall's scientific computing liason to get access to the Jlab scientific computing farm!)

[2] Create a directory named as you like (for example after your reaction: e.g. Pippimpi0 – called here "MAIN")

[3] Move your files: raw_events.gamp, events.pf and the keyfiles of the waves you want to simulate into that directory.
Copy also, from your home directory, PyPWA/pythonPWA/ into that directory.

Go to the pythonPWA/batchFarmServices directory and

[4] run:

simulation_Install

A GUI called pwa_controls will pop-up (Note: There is a "help" button in the GUI itself).

[5] Fill in the information and save.

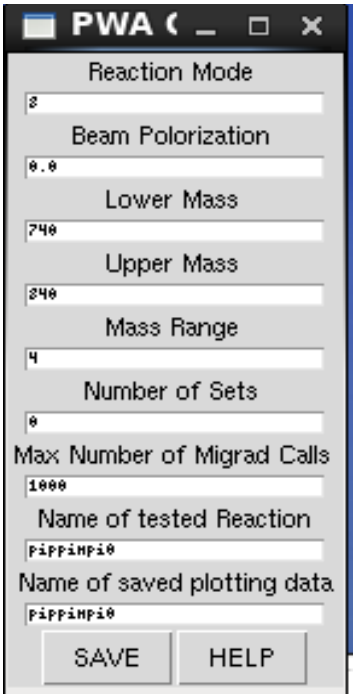


Figure 1: "Pwa_controls" GUI. The HELP button helps you to navigate this GUI.

This action will create your full directory structure needed for SIMULATION. It can take up to 30 minutes of execution (if you have a lot of events). This action prepares the directory structure, re-bin the data, move data to the right directories, transfer some information into numpy format and setup the necessary software.

You will have the following directory structure such that

```
name-of-reaction (“MAIN”)
      |
      |_____________ keyfiles
      |
      |_____________ scripts
      |_____________GUI
      |_____________pythonPWA
      |            |____batchFarmServices
      |            \____pythonPWA
      |
      |_____________simulation
```

and

```
simulation
   |
   |_______overflow
   |_______mass_bin (chosen set)
           |
           |_______ |flat
           |        |_____ events.gamp
           |        |_____ events.num
           |        |_____ events.pf
           |
           |_______|weight
                   |
                   |______ raw
                   |      |____ empty-for-now (events.gamp)
                   |
                   |
                   |______ acc
                  |____ empty-for-now (events.gamp
                   |____                  and events_pf.gamp)
```

---

[6] go to the MAIN/GUI directory and run

PWA_GUI

This is the main GUI for the analysis. It will start with one column an after you make selections two new columns will be opening.

Figure 2: Main PWA GUI.

Then, what you need to do (in this order) is:

[7] *click* **simulation**

A new layer if buttons will open (second column in figure 3),(the next commands will send farm jobs to run the programs: *gamp, genAlpha* for each of the waves and data sets.

[8] *click* **Run Gamp**
[9] *click* ***Gen Alpha***

These actions will: a) create the necessary “waves” in binary format in each correct directory – files called *name*.bamp in the flat/ directory).

Figure 3: Main GUI with selections for the second column.

After clicking command lines will be printed out (a line for each job, one per mass bin and keyfile) as they are being submitted to the batch system. All jobs, (# of mass bins) will all run as separate jobs and not interfere with each other.

b) create the *alphaevents.txt* files in each mass-flat directory.

You can submit the Run Gamp and Gen Alpha simultaneously, they will not interfere with each other.

WAIT until everything is done in the farm.
You need to look at http://scicomp.jlab.org/scicomp/ to check that your jobs are all done and that there were successful (and with Exit Code of 0).

---

[10] *click* **normint**

(these actions calculates raw normalization integrals for each mass bin)

WAIT until everything is done on the farm.
You need to look at http://scicomp.jlab.org/scicomp/ to check that your jobs are all done and that there were successful (and with Exit Code of 0).

[11] *click* **iList**

Create the iList.npy with the calculated intensities for each event in each mass bin. Also rhoAA.npy to be used to calculate intensities.

---

[12] *click* **iMax**

This calculate the maximum intensity in all masses to normalize MC. It is running in your computer is very fast it will print **DONE!.**

NOTE: before proceeding, check that all directories are filled with the corresponding necessary files for your SIMULATION.
The number of events in the files *alphaevents.txt*, *events.gamp* and *events.num* must be the same and the structure should look like this below (for example, for 4 waves in a 1000_MeV mass bin):

```
1000_MeV/
|-- flat
|   |-- 1--1-P.bamp
|   |-- 1--1+P.bamp
|   |-- 2++0-D.bamp
|   |-- 2++1-D.bamp
|   |-- alphaevents.txt
|   |-- events.gamp
|   |-- events.num
|   |-- events.npy
|   |-- rhoAA.npy
|   |-- events.pf
|   |-- iList.npy
|   |-- normint.npy
|   |-- rhoAA.npy
`-- weight
    |-- acc

    `-- raw
```

-- **Simulation for each mass_bin (Mass-independent fit using Minuit)**

from the main GUI:

[13] *click* **Simulator**

All simulation jobs (one for each mass bin) will be submitted to the farm. The time required to run each job depends on number of events and number of waves used.

WAIT until everything is done in the farm.
You need to look at http://scicomp.jlab.org/scicomp/ to check that your jobs are all done and that there were successful (and with Exit Code of 0).

The simulation will produce the following files:
(for each mass bin)

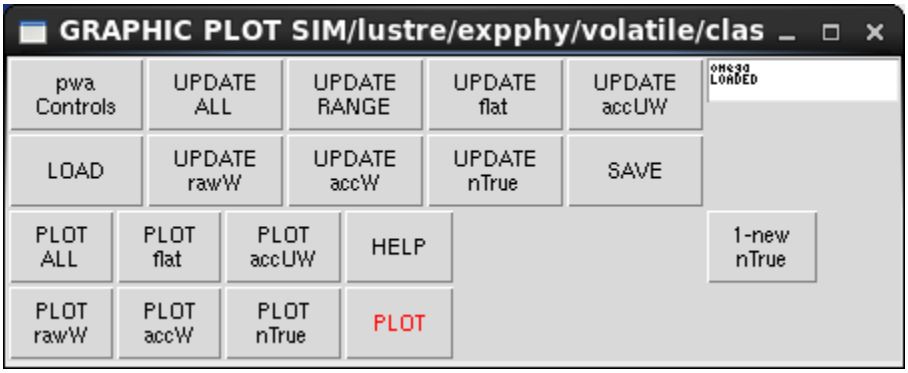
weight/raw/events.gamp
Generated events (weighted) according to resonance/wave set intensity.
weight/raw/events_pf.gamp
Accepted eventsno-weighted (phase space)
weight/acc/events.gamp
Accepted events (weighted) according to resonance/wave set intensity.

flat/wnList.npy
List of weights (intensity normalized) for each event.
flat/calcVvalues.npy
Calculated values of the production amplitudes (V's).
flat/nTrueListR.npy
nTrue values (total and for each wave).

-- **Produce plots (for total and for each wave)**
From the main GUI go **back** and

[14] *click* **Graphic Plot**

This will open a GUI (takes a few seconds).



If it is the first time looking at your results you need to
Click "UPDATE RANGE", "UPDATE flat", "UPDATE accUM", "UPDATE rawW", and "UPDATE accW and nTrue(in that order) and then "SAVE"

Select what you want to plot (one or more distributions from the panel. All are plotted as function of mass bin.)
"PLOT" will plot all selected distributions. (see full description in documentation).

YOU ARE DONE WITH YOUR FIRST PWA FIT!

Now, we will describe option **(B)**:

Working from step 1 of the “Requirements” section (Monte Carlo).
Instead of the resonances.txt, you will need to move the Vvalues.npy files from each mass bin to the created (below) mass bin in the new directory structure. You also need to be sure to have the same keyfiles you used in your fit into the keyfile/ directory (see below).

Execute [1] and [9]

Be sure that you have the correspondent *"bamp"* files to your waves, and the *Vvalues.npy* files on each of the mas bin directories.
As for example, for waves $1^{-}1^{-}P$ and $1^{-}1^{+}P$ you will have:

```
1000_MeV/
|--Vvalues.npy
|-- flat
|   |-- 1--1-P.bamp
|   |-- 1--1+P.bamp
|   |-- alphaevents.txt
|   |-- events.gamp
|   |-- events.num
|   |-- events.npy
|   |-- rhoAA.npy
|   |-- events.pf
|   |-- iList.npy
|   |-- normint.npy
|   |-- rhoAA.npy
`-- weight
    |-- acc

    `-- raw
```

Proceed as in option **(A)** from **[10]**.
