Analysis Preservation:
CLAS12 Exclusive Phi BSA

Brandon Clary

August 2020

1 Introduction

Purpose of this is to document technical details of this analysis for future analyzers.

2 Data

2.1 RGA data:

Collected data is from the RGA Fall 2018 to Spring 2019 run period. It includes the inbending and outbending runs. Four runs are inbending data are not used, 3 did not have any events to process, and one had no phi events but at the time of the analysis it was a corrupt file until later in the analysis. Nonetheless, including it yielded no change. All outbending runs are used.

File used are physically located on the office computer desktop in JLAB. The directory containing the data is:

inbending:
`/home/bclary/Documents/data/rg-a/phi_skim8/inb/skim8_*.hipo`
outbending:
`/home/bclary/Documents/data/rg-a/phi_skim8/outb/train_skim/epKpKm_RGA_OUT.hipo`

The outbending file was produced by Francois X. Girod for the outbending dataset.

2.2 Simulation:

Data is for inbending field. It was produced using OSG. At the time it was based on gemc 4.3.2 coatjava software version 6.3.5 in July of 2020. Simulation was done with the Type 2 submission and specifying the rga-fall2018 gcard. The gemc.gcard is copied and stored on the office desktop computer as gemc.gcard. Main consideration is that the target position is shifted 3cm upstream or -3cm.

Beam energy specified is 10.6 in the gcard. Since it is on OSG there is no flexibility to have the user change these settings.

Location of files:
`~/Documents/data/sim/phi/10gev/cj6p3p5/final_thesis_simulation/`

These simulation files correspond to the lund files in

`~Documents/data/sim/phi/10gev/cj6p3p5/final_thesis_simulation/lund_files`. There were 998 files at the time with 10k generated events in each file.

3 Generator for Simulation

Lund files created using the generator specified in the pacs phi proposal. See Brandon's thesis for more details on the contents of the generator.

Generator is located at `/w/hallb-scifs17exp/clas12/bclary/CLAS12/phi_analysis/generator/GenPhi.cxx`

It can be run locally on the ifarm, office computer, or the bash farm (where it should be used to generate events in an efficient time frame).

Running it The local environment in which it is run is specified in the GenPhi.cxx file at the top near the header. It is currently able to run with root 6.20.04. To compile the code requires the following command:

```
g++ GenPhi.cxx -o b.exe `root-config --cflags --glibs`
```

The output file is in the standard lund format. Currently it was found that the dipole model best fits the t slope in data. The crucial line is

```
cT = 400*TMath::Power(1-Wth*Wth/W2,0.01)*TMath::Power(W,0.32);
```

The power of 0.01 is new and not the power of 1 as proposed in the PACS document. Talked about in more detail in Brandon's thesis.

3.1 Submit jobs to farm

To submit jobs in parallel requires using the batch farm. Files for this are in `/w/hallb-scifs17exp/clas12/bclary/CLAS12/phi_analysis/generator/farm`

File farp.py creates the jsub files for submission. Comment in there shows that mod 3 is 0.01 is the power used for calculating cT term.

It is best to use .csh files for the batch farm. To run the executable from compiling the GenPhi.cxx file the run.csh is included as an additional file. It sources the clas12 environment and root version of 6.14.04, which the GenPhi.cxx is also compatible with. It copies the GenPhi to the local batch farm node, compiles it, then runs it. Argument is the nth job that is processing. This help to index the files. Submit jobs via jsub *.jsubs.

Note that this generator will likely become obsolete in the near future. In the meantime keep using this one.

3.2 Jobs for OSG and Managing Output

To submit jobs for GEMC go to `https://gemc.jlab.org/web_interface/index.php`. Here the jobs can be submitted for lund files in the volatile directory. Output file location is specified on the website.

Merging the output from the OSG files can be a bit tedious. Use this script here to do it in the background (i.e. using screen functionality). The files are **run-merge.py** and **run-createMerge.sh**.

`/w/hallb-scifs17exp/clas12/bclary/CLAS12/simulation/generators/`

First generate sublist of files from osg by running **run-merge.py** with 2 arguments: OSG job number and the prefix for the output txt files to be used in the next step.

Second step involves **run-createMerge.sh**. Before running make sure the arguements for the merge -o outputfilename inputfiles fits your needs and naming convention. Make sure the coatjava is cited correctly. I didn't make it global because I needed to keep changing it - but it worked for my needs.

Recall that the OSG files are distilled and only have the REC::* banks. You'll need to run the simulation using another approach which is significantly involved and is not necessarily encouraged. Generally speaking it will require generating lund files, processing with GEMC (requires specific input files), then decoding, and reconstruction.

4 Analysis Code

4.1 Where to Find the Base Code

Code is written in Groovy programming language. The place to find it is:
`/w/hallb-scifs17exp/clas12/bclary/CLAS12/analysis_code_fork0`

It is also found on GitHub at
`https://github.com/drewkenjo/analysis_code`

But the version of the code used is found at...
`https://github.com/kuantumlad/thesis_code`

It should be forked to use. Using the code is relatively simple and documented with examples in the example directory. Note that the version in the /work/ path is modified and out of sync with the GitHub repo. These are minor differences.

Currently the groovy version used is 2.5.7 and in `/opt/groovy/2.5.7/bin/groovy` of the office computer.

Python >= 2.7 and ROOT is also required to run the analysis code as it will write out .root files for analysis. The version of ROOT for the analysis was root-6.20.04. Additionally, j2root is required.

4.2 Main Code

The core of the phi analysis is with the **monitorPhi.groovy** file. To run it requires groovy and the coatjava run-groovy feature. The command line to

processes a file is

```
run-groovy monitorPhi.groovy /path/to/hipo/file/to/analyze.hipo
```

Details on how to process it are in the **monitorPhi.groovy** file. This includes specifying:

- beam
- $field\_type$ (inbending is "inb" for outbending is "outb")
- gen_asy used for the checking bin effect of on asy
- number of sigma for exclusivity cuts
  - epkpkmX missing energy
  - epkpX
  - epkmX
  - ekpkmX
- specifiy electron, proton, positive and negative Kaon cuts from following classes.
  - new ElectronFromEvent()
  - new HadronID($hadron\_id : ID$)

The flow of the analyis is understood by going through the code. Basically it requires Event Builder PID and all particles detected in the Forward detector. Histograms are built to monitor the code. The output of this file includes two ROOT files. The first is a the root file containing a TTree with all events for the BSA measurement. The other ROOT file contains all the histograms, the prefix of this file is $monitor\_phi_{*}root$.

4.3 Additional Files For monitor_phi.groovy

As mentioned cuts are done on various variables based on the mean and sigma of the fits to the distributions. The mean and sigma are stored in text files located here `/w/hallb-scifs17exp/clas12/bclary/CLAS12/analysis_code_fork0/projects/exclusive_phi/epkpkm_top/`

The files names are determined based on the field type - "inb" or "outb"

- "excl_phi_me_limits_pidtype1_"+field_type+".txt"
- "excl_phi_mm2_pro_limits_pidtype1_"+field_type+".txt"
- "excl_phi_mm2_kp_limits_pidtype1_"+field_type+".txt"
- "excl_phi_mm2_km_limits_pidtype1_"+field_type+".txt"

4.4 Additional Environment Requirements

Additionally to run this requires ROOT and j2ROOT written by Andrey Kim to create and save the histograms to a ROOT file while running. Root version is 6.20.04 to use to the j2root package. More details to come.

4.5 Processing Output File from monitor_phi.groovy

Various projects are recorded in the `/w/hallb-scifs17exp/clas12/bclary/CLAS12/analysis_code_fork0/projects`. Of specific interest to the BSA / exclusive phi analysis are the directories

- phiasy
- monMC
- compareFields
- exclusive_phi/epkpkm_top

Next the specific script/code in each directory relevant to the analyis project will be breifly discussed.

4.5.1 exclusive_phi/epkpkm_top Directory

Relevant code in here includes

1. monitor-phi.py -i monitor_phi*.root -o outname
2. mon-phi-selection.py -i monitor_phi*.root -o outname
3. mon-phi-event.py -i monitor_phi*.root -o outname
4. phimassV2.py -i input_from_monitorAsy.py
5. mon-phimass.py -i input_from_monitorAsy.py
6. makeFinalPlots.py -i input.root

(1) Provides overview of phi analysis and plots content in the monitor_phi*.root file which is the input for the python script. Here * should be interpreted as the name you may give the file.
(2) monitor-phi-selection.py covers those plots to help how to cut on possible phi events
(3) montiro-phi-event.py covers the distributions in the phi peak to see what they look like (4) phimassV2 plots the kpkm mass histograms produced from the script monitorAsy.py
(5) Same as (4) but also plots more phi masses and allows for better modification of the fit parameters. Also produces the .txt files containing the signal to background ratio needed for bsa sideband subtraction techniques
(6) makeFinalPlots is responsible for making the missing energy and missing mass squared distribution and produces the .txt files used for selecting the phi events mentioned earlier.

4.5.2 monMC Directory

Files here are used to compare the results from data and simulation. These scripts require two files as inputs, check the main method of the script to determine input order. The inputs are the outputs from the monitorPhi.groovy script.

1. mon-data-mc.py
2. mon-mc.py
3. mon-sim-phi.py

(1) produces histograms that overlap data and simulation events within the phi mass range. Other plots produced as well. Good for validating the phi model / generator
(2) mon-mc.py looks at various distributions related to simulation, includes the generated distributions.
(3) mon-sim-phi.py aids in checking the reconstructed false asymmetry in simulated results.

4.5.3 compareFields Directory

Files here are used to compare the distribution for each particle and physics kinematics from the inbending and outbending datasets. These scripts take two files as inputs to make histograms showing the difference between the two.

1. mon-compare-fields.py -iinb monitor_phi_inbdata.root -ioutb outbending.root -o outname
2. mon-cf-phi-event.py -iinb monitor_phi_inbdata.root -ioutb outbending.root -o outname
3. mon-cf-phi-selection.py -iinb monitor_phi_inbdata.root -ioutb outbending.root -o outname

(1) specifically shows the distributions for events within the defined phi peak
(2) includes the resolution plots to compare measure - calc values from inbending and outbending fields. In principle can be used for comparing simulation with inb and outb fields. (3) File is meant to compare the variables that are cut on to select phi events from the two field settings.

4.5.4 phiasy Directory

This is the heart of the analysis for the BSA. Extracting the BSA for the exclusive phi analysis is done using the following python script files. These files are listed in the order they should be ran in.

1. monitorAsy.py phi_mass_results_*.root

2. mon-phi.py -i output_from_monitorAsy.root -o outname

3. showBckAsy.py -o outputname

4. mon-phi-method2.py -i output_from_monitorAsy.root -o outname

(1) monitorAsy.py will read in a ROOT TTree object to create histograms for the distribution in phi trento, KpKm invariant mass spectrum, etc., which are needed for the other files. Some items to consider before running it are to specify the

- field setting with $field\_setting$ as inb, outb, or inbNoutb
- mass binning variation, which is 11 for the final analysis with bin_var. This is a txt file that one can create.
- overlapping bin mass range, needed for systematic check
- bin ranges for kinematic variables. First create files with arbitrary ranges to run, these ranges will be changes as mentioned below.

At the end of this python script are functions that will divide the dataset into bins in terms of the kinematic variables of interest, as well as mass bins such that there are equal number of entries in each bin. This code should be ran twice to properly get these bin values. First time to write the txt files which are then read in the second time with the correct limits. After monitorAsy.py is done running all the histograms needed for the remaining scripts are available in the output file. The txt files are called bin_limits_w_inbNoutb.txt for example.

(2) After running monitorAsy.py the output file has the correct histograms for phi trento for each helicity state. This .root is processed by mon-phi.py. Check that the correct field setting, mass bin variation, beam polarization (bp), are specified. The output of this is a pdf file and txt file. The pdf file contains the relevant histograms to check to determine the bsa step by step. The methods responsible for this are plot_slices and plot_page_bins. The asymmetry in each kpkm mass bin, as specified by the mass bin variation, is determined. The 10 bsa points as a function of phi trento are saved to a txt file as labeled by f_out and f_out2 in the plot_asy method. The measured $sin(\phi)$ moment in each kpkm mass bin is saved to asy_vs_trento_method1_imkpkm_bin is written out for each mass bin.

(3) Lastly to determine the signal asymmetry using the sideband subtraction technique the last file to run is the showBck.py script. Make sure the correspond field_setting, binvar, and signal to background ratio values are in there (sb_ratio). The asy_per_mass variables is the measured BSA in each KpKm mass bin and used for sideband subtraction to get the signal asymmetry. This method is discussed in detail in B. Clary's thesis. The bin_info is a python dictionary with specific information pertaining to making the plots. The final asy. moment is determined using the plot_asy method. Variations of this

are in the .py script to determine the final asymmetry moments for special cases.

(4) The second method, fitting the kpkm distribution in bins of phi trento for each helicity state is done using file mon-phi-method2.py. This has to be executed after the monitorAsy.py but not the other scripts. Output is a pdf with the necessary info to determine the asymmetry moment.

5 Remarks

The discussed files are to used in determining the moments of the sin(phi) from the exclusive vector phi meson channel. More details can be found in Clary's thesis.