CLAS12 Software Tutorial

Nathan Harrison
Jefferson Lab

CLAS Collaboration Meeting
March 31, 2017
Jefferson Lab

Outline

- Introduction and preliminary set-up (10 min)
- Simulations with GEMC (15 min)
- KPP raw data (5 min)
- Downloading and installing CLARA and COATJAVA (10 min)
- Decoding (10 min)
- Reconstruction (10 min)
- Analysis code (15 min)
- Using an IDE (10 min)
- Analysis studio – Will Phelps (15 min)
- Docker – Nick Tyler (15 min)

Introduction and preliminary set-up

- Instructions are written in sea blue
- Terminal commands are written in black, are indented, and start with a “> “ e.g.

  > echo “hello world”

- Green text means you should substitute the text with your own value

Introduction and preliminary set-up

(1) Open a terminal on your laptop and log into an ifarm CUE machine (requires two steps):

```
> ssh -Y username@login.jlab.org

> ssh ifarm
```

(2) This tutorial will assume you are using tcsh, to check your shell, do:

```
> echo $SHELL
/bin/tcsh
```

to switch to tcsh, do:

```
> chsh -s /bin/tcsh
```

It will also be assumed that your ~/.tcshrc file is empty; if this is not the case then your safest option is to comment everything out. If you made any changes in this step, log out and log in again.

Introduction and preliminary set-up

(3) Select a location with sufficient disk space (e.g. /work or /volatile) and create a new directory in which to work:

> cd /volatile/clas12/username/

> mkdir myDemo

> cd myDemo/

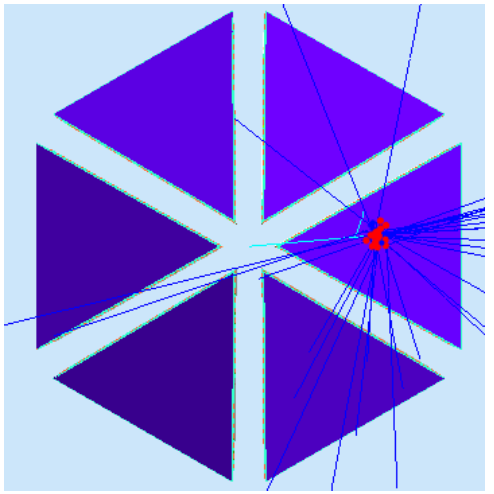
(4) Copy the directory of ancillary files to your working directory and source the provided environment script:

> cp -r /volatile/clas12/nathanh/demo_31mar17 .

> source demo_31mar17/demo-env.csh

* see demo_31mar17/commands.txt for copying/pasting

Simulations with GEMC



(1) Setup the correct GEMC environment with the latest version (4a.0.1):

> source /group/clas12/gemc/environment.csh 4a.0.1

(2) Pass a single e- through EC with graphics:

> gemc demo_31mar17/test.gcard -USE_GUI=2 -BEAM_P="e-, 4*GeV, 20*deg, 5*deg" -N=1 -OUTPUT="evio, single_ele.evio"

(3) Pass 200 e-'s through CLAS12 in batch mode:

> gemc /group/clas12/gemc/4a.0.1/clas12.gcard -USE_GUI=0 -BEAM_P="e-, 4*GeV, 20*deg, 5*deg" -SPREAD_P="0.5*GeV, 15*deg, 25*deg" -N=200 -OUTPUT="evio, ele.evio" -RUNNO=11

(4) Pass 500 $\pi^0 \rightarrow \gamma\gamma$ events through CLAS12 in batch mode:

> gemc /group/clas12/gemc/4a.0.1/clas12.gcard -USE_GUI=0 -INPUT_GEN_FILE="LUND, demo_31mar17/pizero2gg.dat" -N=500 -OUTPUT="evio, pi02gg.evio" -RUNNO=11

* save all the output files for later!

* see gemc.jlab.org for more information

KPP raw data

- KPP raw data is located on tape at /mss/clas12/kpp/data/
- Faster access is temporarily available at /cache/clas12/kpp/data/
- Copy one KPP file to your working directory; run 809, file 902 is a good one:

  > cp /cache/clas12/kpp/data/clas_000809.evio.902 .

Downloading and installing CLARA and COATJAVA

(1) Get the CLARA install script and make it executable:

> wget --no-check-certificate https://claraweb.jlab.org/clara/_downloads/install-claracre-clas.sh

> chmod +x install-claracre-clas.sh

* modify the top of this script to use COATJAVA 4a.2.2

(2) Set the CLARA_HOME environmental variable (note that you are setting it to a directory that does not yet exist):

> setenv CLARA_HOME $PWD/myClara/

(3) Run the install script, this will install both CLARA and COATJAVA. Point the COATJAVA variable to the COATJAVA installation:

> ./install-claracre-clas.sh

> setenv COATJAVA $CLARA_HOME/plugins/clas12/

* more information at claraweb.jlab.org and http://clasweb.jlab.org/clas12offline/distribution/coatjava/

* CLARA and COATJAVA should work on any Mac or Linux system with only one prerequisite – Java version 1.8 or higher

Decoding

(1) Now let's decode our various raw files, note GEMC files and data files are done differently:

> $COATJAVA/bin/evio2hipo -r 11 -t -1.0 -s 1.0 -o ele.hipo ele.evio

> $COATJAVA/bin/evio2hipo -r 11 -t -1.0 -s 1.0 -o pi02gg.hipo pi02gg.evio

> $COATJAVA/bin/decoder -t -0.5 -s 0.0 -i clas_000809.evio.902 -o clas12_000809_a00902.hipo -c 2

Reconstruction

(1) Create “files.list” containing the hipo files to be cooked (file names only, no path):

> ls *hipo > files.list

* suggestion: put the KPP file last since it takes the longest; it can be killed mid-cook with no consequences other than lower statistics. (Advantage of hipo format!)

(2) Launch the CLARA CLI:

> $CLARA_HOME/bin/clara

(3) Try typing “help”, “help set”, “help monitor”, and “monitor params” to get a feel for the CLI.

(4) Set the relevant params and run locally:

> -i /path/to/myDemo/ (directory containing input files)
> -o /path/to/myDemo/ (output directory)
> -f /path/to/myDemo/files.list (file list)
> -t 4 (number of threads)
> run local

Alternatively, you can do this right from tcsh:

> $CLARA_HOME/bin/run-clara -m local -i /path/to/myDemo/ -o /path/to/myDemo/ -f /path/to/myDemo/files.list -t 4

(5) Optional: try submitting a job the the batch farm with “run farm” or “-m farm” and the following additional options: “-fc 16 -t 16 -fm 12 -fd 5 -ft debug”

Analysis Code

(1) Browser the structure of the data files using eviodump (also works on hipo files):

> $COATJAVA/bin/eviodump out_ele.hipo

(2) Take a look at demo_31mar17/electronAnalysis.groovy and try running it:

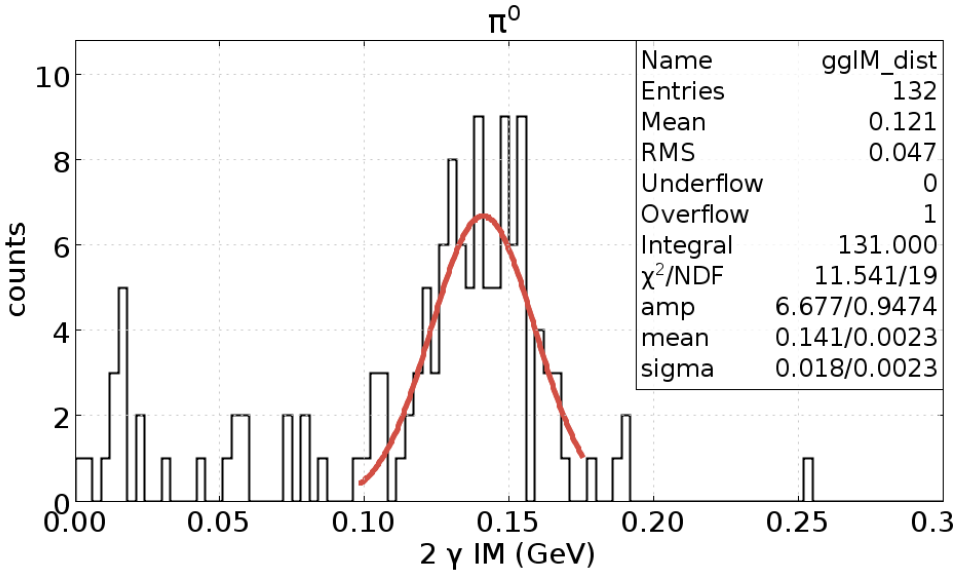
> $COATJAVA/bin/run-groovy demo_31mar17/electronAnalysis.groovy out_ele.hipo

(3) Run demo_31mar17/pi0Analysis.groovy on the pi0 file and the KPP file:

> $COATJAVA/bin/run-groovy demo_31mar17/pi0Analysis.groovy out_clas12_000809_a00902.hipo

> $COATJAVA/bin/run-groovy demo_31mar17/pi0Analysis.groovy out_pi02gg.hipo

(3) Right-click on a canvas to adjust the options or open the fit panel.



Using and IDE

Integrated Development Environments (IDEs) such as Eclipse and NetBeans make writing code much easier and faster. To get started, download Eclipse (preferred) or NetBeans and open it.

(1) Do File → New → Java Project and type in a project name. This will create a new project visible in the Package Explorer on the right.

(2) Right-click on the project in the Package Explorer, navigate to Java Build Path and then Libraries and click on “Add External JARs...”

(3) When the file browser opens, navigate to your COATJAVA directory, and go to the lib/clas/ directory and open the coat-libs-3.0-SNAPSHOT.jar file. Now you can use the COATJAVA tools in your project.

(4) Expand your project in the Package Explorer and right-click on src and select New → Package and name your package. Then right-click the package and select New → Class and name the class.

(5) Create a main method with some simple code. Note some of the options that IDEs offer, such as providing a list of all methods for a given class and helping you import the correct packages.

(6) Click the green play button to run your code.