The Basics of Skimming CLAS Data

and some finer points of PID

J. Pond[1][2]

[1]Department of Physics
University of Virginia

[2]CLAS
Jefferson Lab

June 19, 2017

Prerequisites

- How to ssh into the JLab ifarm
- Rudimentary Linux/Unix commands (tcsh)
- Basic Python scripting
- C++ and the Cern ROOT libraries

Outline

Introduction

Skimming

Data Skimming is the process of taking an entire data set and skimming for events in a particular subset based on certain criteria and returning them in a more compact and easy to analyze format.

Particle Identification

The most basic skimming type is when one skims for all events of a particular final state. I.e. $\gamma P \rightarrow P\pi^{+}\pi^{-}(\pi^{0})$, in which the user would take every event with a proton and both charged pions, assuming the neutral pion is “missing” (more on that later).

Analysis Cuts

The next step would be higher level analysis cuts. I.e. taking out events that have vertices outside of the target cup, events with particles moving faster than the speed of light, and events with poor timing.

The Interface

Who's in charge?

Auger

- Top level interface of the computing system
- Interfaces with the PBS system on your behalf

Jasmine

- Top level interface of the tape system
- Controls the data flow in and out of the tape system

The Interface cont.

How do they do it?

Auger

- Auger is the program that takes job requests, stages and runs the jobs, and returns the output to where the user expects it to be.
- Auger can talk to Jasmine and get the files a job needs without any extra work from the user.
- The input for Auger is called a jsub file, and the command is simply: user@host$ jsub jsub-filename.

Jasmine

- Here, and in the vast majority of skims, the user should bypass Jasmine and request files from tape directly from the Auger jsub file.

The Infrastructure

What are they in charge of?

- Batch Farm
  - Roughly 200 nodes with varying numbers of job slots, and available resources
  - Entirely abstracted by Auger
- Disk
  - Lustre Disk
    - 1.4 petabyts of available storage
    - Includes the /cache, and /volatile file systems
    - Short term storage and job outputs
  - Other Disks
    - Includes the /home, /group, and /work file systems
    - Long term storage
- Tape
  - $\approx$ 20 petabyts
  - Very long term storage
  - Entirely abstracted by Jasmine
  - Virtual file system rooted at /mss

Basics

Auger Jsub Files

Example

```
PROJECT: g12
TRACK: test
JOBNAME: nStar_skim
OS: centos7
MEMORY: 1GB
COMMAND: source /group/clas/builds/centos7/environment.csh;
/home/jpond/build/bin/nStar_simple_skimmer inFile
INPUT_FILES:
/mss/clas/g12/production/pass1/bos/1-1ckaon1ctrk/56400.A10
INPUT_DATA: inFile
OUTPUT_DATA: outPutFile.root
OUTPUT_TEMPLATE: /u/home/jpond/test_simple.root
```

Breaking it Down

First Lines

- PROJECT: tells Auger what group you're in, since different groups get different access to the farm. Use CLAS projects, like g11, g12, g13, etc.
- TRACK: tells Auger what queue to put your job in, priority, Prod64, or long-job. Test will put you in priority, but it has restrictions. For skimming, use Analysis Track, which goes to Prod64.
- JOBNAME: simple string that identifies that job. No need to make it unique for every job.
- OS: tells Auger what OS to use, as there are more than one. At time of writing centos7 is the most current.
- MEMORY: The amount of RAM Auger will guarantee your job. This has a default value of about 500 MB, but this is smaller than some files might need, so 1 GB is safer.

Breaking it Down

The Command Line

Example

```
COMMAND: source /group/clas/builds/centos7/environment.csh;
/home/jpond/build/bin/nStar_simple_skimmer inFile
```

The command is the most important part. It is everything that is run in the job and it can be written exactly as you would in the terminal on the iFarm. In this example we source the current environment for the clas6 software, and then the skimming code is run on the file inFile.

Breaking it Down

File control

- INPUT_FILES: The /mss stub file(s) corresponding to the file you want from tape.
- INPUT_DATA: The local name of the file(s) from tape. This files are *on the node*, and *in the current working directory*.
- OUTPUT_DATA: The local output file(s) Auger can expect to be produced by your job.
- OUTPUT_TEMPLATE: The template Auger will use when naming the output file(s) when it copies them to disk.

Software

The recommended method of skimming requires 3 major pieces of software:

- A BOS to ROOT skimmer that can run on the Batch farm,
- A ROOT merging program that can run on the Batch farm,
- A job submission automater.

Software cont.

BOS to ROOT

BOS

The proprietary file format of CLAS data that is stored to tape. It is split into "Banks" and contains every possible piece of information one might want about an individual physics event, including the digital instrument signals, as well as PID banks. BOS can only be read and analyzed by tools in the CLAS libraries.

ROOT

ROOT is at its core a nuclear and particle physics C++ library. It has useful features for the entire data analysis process. In this case we mean the ROOT data structure called a TTree, which can be saved directly as a .root file. This is a very efficient data structure and has many advantages over keeping everything in BOS.

Software cont.

BOS to ROOT cont.

The goal of the skim is to take the CLAS data that is in BOS, take only the events you want, i.e. a single final state, or a single momentum and energy region, etc. and return the events you want in a ROOT TTree. This constitutes a single program and has a minimum of two parts:

1. A main function that can read BOS files and loop over the events and write to the ROOT TTree.
2. A function that takes a single clasEvent (which is a C++ Class from the CLAS6 libraries), and processes it for compliance with the skim you are taking.

An annotated example written in C++ will be available with this presentation.

Software cont.

Corrections

Another task that will have to completed at some point in your analysis, and might as well get done now, are the event corrections. These corrections are implemented in the CLAS6 libraries and are based on analysis done by the run teams. For the g12 run the corrections include:

1. Energy Loss corrections,
2. Beam corrections,
3. Particle momentum corrections.
4. Fiducial Cuts and TOF knockouts (Discussed later)

An annotated example written in C++ will be available with this presentation.

Software cont.

Merging

This is a matter of personal taste, and it is the taste of the author to have a policy of one job per file on Tape. This will extend the total amount of time spent waiting in Auger queues, but it has great advantages to the user in terms of automating submissions.
This means a program, not unlike the skimmer, is needed to concatenate the .gt. 100,000 small root files. This is done in two stages. Once into about 200 files on the iFarm, and then once more, once the files have been moved to the lab.

An annotated example written in C++ will be available with this presentation.

Software cont.

Automated Job Submission

Job automation for skimming has three main parts:

1. Finding files in the /mss file system
2. Dynamically creating a Jsub text file on disk
3. Calling the Jsub command on the text file

Luckily all of these functions are straightforward in Python, using the OS, and Subprocess modules.

An annotated examples written in Python will be available with this presentation.

Basics

PART bank

The most basic kind of PID is the most simple. Each BOS event contains a PART bank which tells you what kinds of particles are in the event.

Example

```
ClasEvent evt;
if (evt.N(PiPlus)==2 && evt.N(PiMinus)==1 && evt.N(Neutron)==1){
   /*More Analysis*/
}
```

In this example we have a ClasEvent object called evt with the member "N" which returns the number of that kind of particle in the PART bank. This is the first level of PID cuts.

Missing Particles

Neutral particles have very low acceptance because they are unaffected by the magnets in the detector. This means channels with a neutral particle will have low statistics, and more than one is basically pointless.
To help the statistics we let a neutral particle go “missing,” as in, we do not detect it in our initial PID cuts, but instead impose PID cuts on the left over “missing” 4-momentum, X.

Example

$$\gamma P \to \pi^+\pi^- X$$

We know X is probably a π^0 in this example, but we won’t know that for sure until we do PID on the X spectrum.

Missing Particles cont.

The most important missing PID cut is the simple invariant mass cut.

Example

```
if (sqrt(mm2Pi0)> 0.11 && sqrt(mm2Pi0)< 0.17){
   /*More Analysis*/
}
```

For our example the only neutral particle in this range is the neutral pion, what we want. This does not confirm our PID, but it is a great start. More advanced analysis cuts are to come.

Basics

Analysis cuts are like PID cuts in that they remove data from the set that you do not want to analyze, but instead they focus on removing corrupted events and background, to isolate signal events. Here we will give an introduction to two main kinds of cuts we call:

1. Hardware Cuts:
   - Cuts data based on the known dimensions of the experiment,
   - Drift chamber supports, target dimensions, and bad time of flight panels
2. Isolation Cuts:
   - Cuts data to give more detailed isolation of signal particles,
   - Timing, momentum, and invariant mass.

Drift Chamber Supports and TOF panels

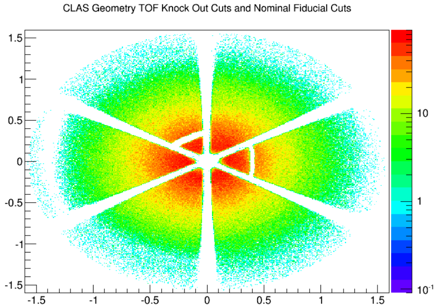
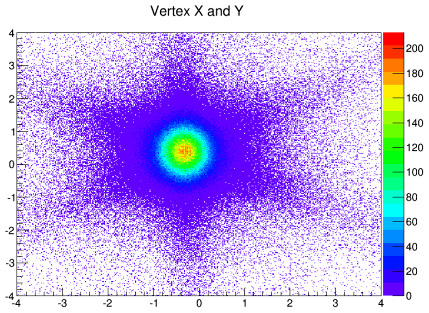
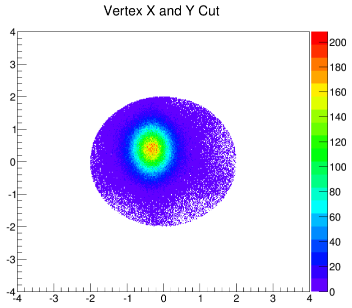


Figure: Using the Fiducial cuts and the TOF knockout cuts has cut out events that have drift trajectories that pass through the supports of the drift chambers themselves, which you don't want, as well as the events passing through areas with dead panels that did not work during the run.

Target Dimensions



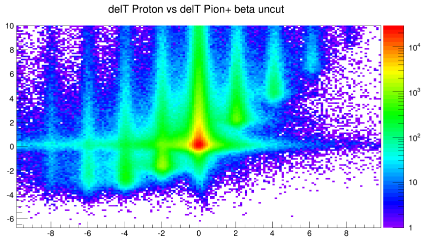
(a)



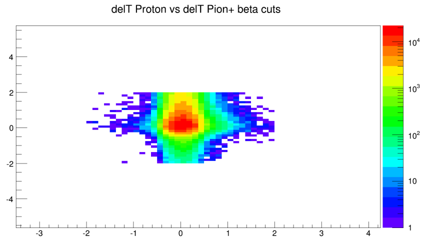
(b)

Figure: Since we know the radius and length of the target, we can cut out any events with a vertex outside of its dimensions.

Timing



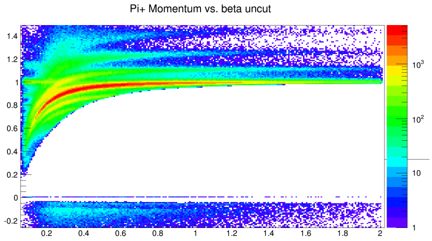
(a)



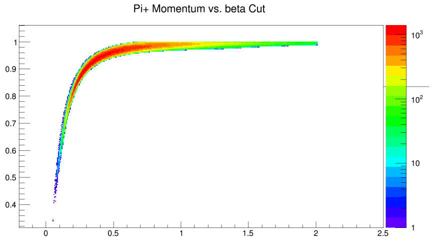
(b)

Figure: The beam from the accelerator comes in pulses, or "Beam Buckets" that are only 2 ns apart, this means there can be some cross interactions with photons from the wrong bucket, which throws off the timing calibration of all the other parameters. We can cut out these events by cutting around the center bucket which becomes visible when the Δ T of two different particles are plotted against each other, as here.

Momentum



(a)



(b)

Figure: Here we've taken two cuts. One that insists that the β of the particles be between 0 and 1, to be physical. You can also see in (a) that there are bands around the main one, these are particles that have been misidentified by their momentum, so by maintaining a maximum difference between the measured and calculated β we isolate the correct band.

Invariant Mass

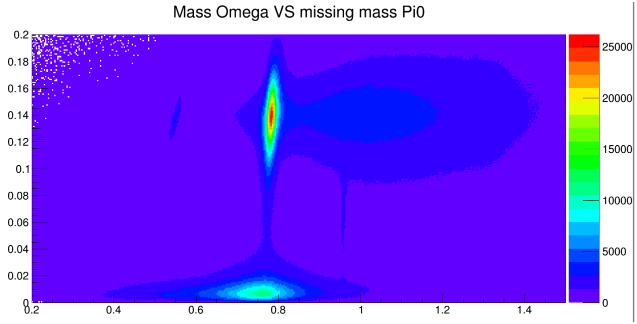


Figure: Here we want to take a more detailed look at the invariant mass of the missing 4-vector, so we spread the I mass of a composite particle, the ω meson, out as a function of the missing mass that decayed from it. In this way we can see the range of X that we want really clearly and isolate the events where X is in that range.

Invariant Mass

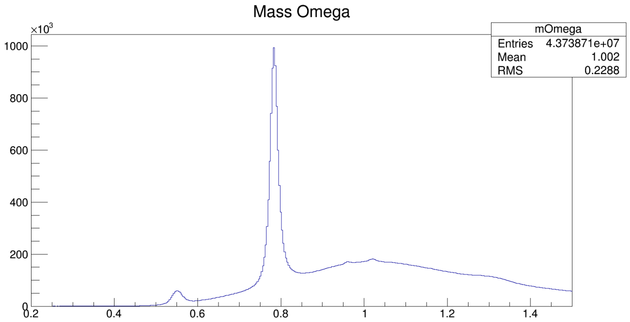


Figure: At the end of all these cuts we can look at the mass spectrum of our example and see the large omega signal clearly visible past the background.