CLAS12 Software Guide for Beginners

This is a “bare-bones” guide to get users started with GEMC (Geant Monte Carlo) (version 2.3) and coatjava (CLAS Offline Analysis Tools (Java based)) (version 2.0). For more complete documentation, see gemc.jlab.org and http://clasweb.jlab.org/clas12offline/docs/software/html/index.html.

Nathan Harrison (nathanh@jlab.org)
March 10, 2016

Outline

- Setting up your environment
- Running GEMC
- Running clas12-reconstruction
- Working with evio files

Setting up your environment

Note: I recommend running GEMC on ifarm65 or the batch farm; coatjava can be run anywhere

- To set up the farm environment, run the following commands:

```
source /site/env/syscshrc
module load java_1.8
use groovy
source /site/12gev_phys/production.csh 1.3
setenv COATJAVA /group/clas12/packages/coatjava-2.0
setenv PATH /group/clas12/packages/bos:${COATJAVA}/bin:${PATH}
rehash
```

- On your own machine (mac or linux)
  - check your version of Java JDK (from a terminal, do “java -version” (without the quotes))
  - if version is < 1.8, update to the latest version
    *you need the Java Development Kit (JDK) from oracle.com, NOT the Java Runtime Environment (JRE) from java.com
  - if Groovy is not installed, install it (easiest way on mac is to first install homebrew, then do “brew install groovy”)

- Download and unpack coatjava with the following commands:

```
wget https://userweb.jlab.org/~gavalian/software/coatjava/coatjava-2.0.tar.gz
tar -zxvf coatjava-2.0.tar.gz
```

Running GEMC (use ifarm65 or the batch farm)

- Assuming your environment is set up correctly, do

```
gemc /group/clas12/clas12.gcard -USE_GUI=0 -N=250 -RUNNO=11
```

reads constants from database corresponding to run 11 (for now this is the best value, but it's likely to change)

set to 2 for graphical interface (but it's very slow)

number of events to generate

copy this file, then you can customize:

- which detectors and components to include/remove
- beam particle, momentum, angle, vertex and spread
- name of the output file
- lund file input (see gemc.jlab.org)
- etc... (see gemc.jlab.org)

*in the ftof line, change “original” to “javageom”

*comment out the “INTEGRATEDRAW” line if you don't need the MC true banks (this saves a lot of space)

Running clas12-reconstruction

(must have internet connection to access database, or use the local sqlite db (see documentation))

- To reconstruct (“cook”) the raw evio file produced by GEMC, use your downloaded coatjava package:

```
coatjava/bin/clas12-reconstruction -s DCHB:DCTB:FTOF:ECREC:EB -i out.ev \
-o rec2.0_out.evio -config DATA::mc=true -config CCDB::CALIBRUN=11
```

name of the output file
*the actual output filename will be rec2.0_out.0.evio (note the extra “0”)

reconstruction for the forward detector
remove this to get the entire detector

should be consistent with RUNNO on previous slide

name of the file to cook

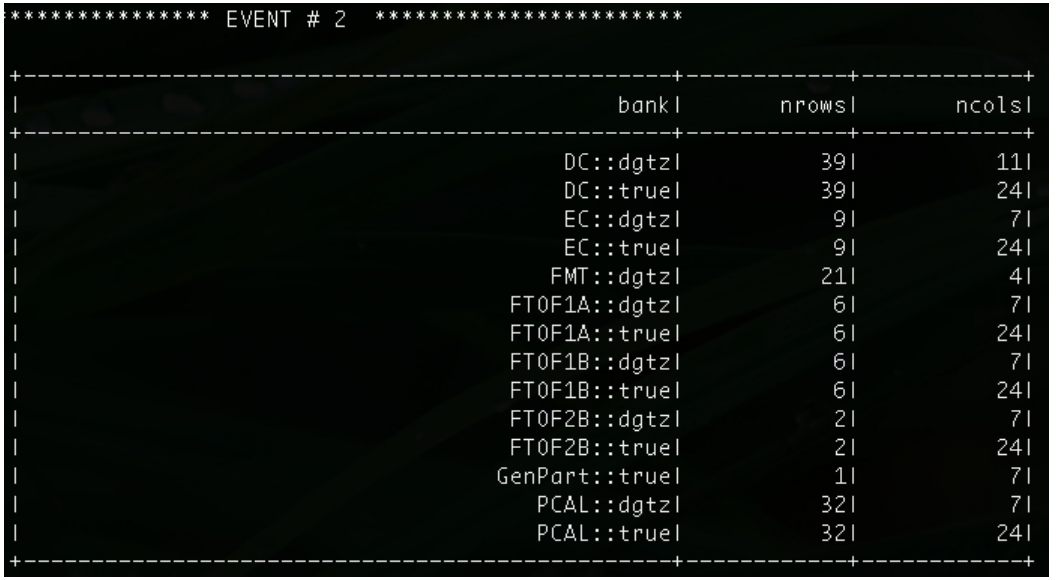
*when running clas12-reconstruction on the batch farm, you need to request a lot of memory (~6 GB) (this increases wait time to use the farm, and is really more than is needed... something to do with Java VM... hopefully this can be improved in the future???)

Working with evio files, part 1

Both GEMC and clas12-reconstruction produce data files in evio format which store information in banks. To browse through the banks, use evio-dump:

coatjava/bin/evio-dump -i filename.evio

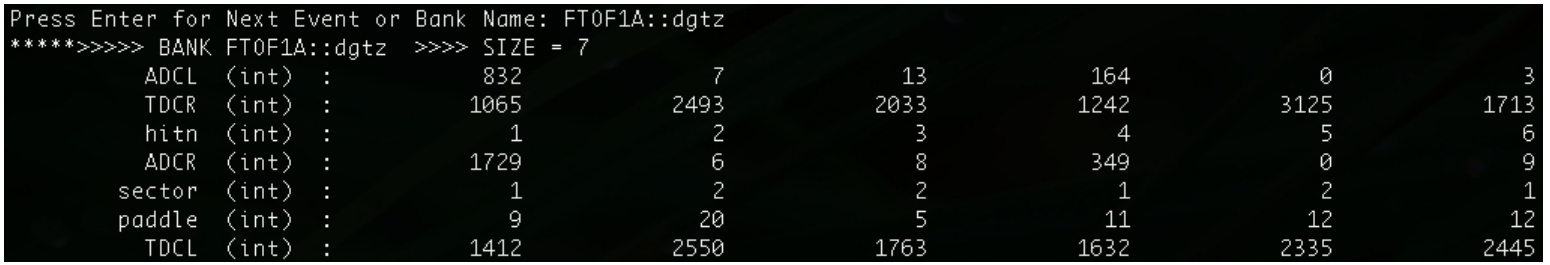
You will see a list of the banks for a given event (press enter to go to the next event):



```
*************** EVENT # 2  ***********************

+-----------------------------------------------+------------+------------+
|                                           bank|       nrows|       ncols|
+-----------------------------------------------+------------+------------+
|                                       DC::dgtz|          39|          11|
|                                       DC::true|          39|          24|
|                                       EC::dgtz|           9|           7|
|                                       EC::true|           9|          24|
|                                      FMT::dgtz|          21|           4|
|                                    FTOF1A::dgtz|           6|           7|
|                                    FTOF1A::true|           6|          24|
|                                    FTOF1B::dgtz|           6|           7|
|                                    FTOF1B::true|           6|          24|
|                                    FTOF2B::dgtz|           2|           7|
|                                    FTOF2B::true|           2|          24|
|                                   GenPart::true|           1|           7|
|                                      PCAL::dgtz|          32|           7|
|                                      PCAL::true|          32|          24|
+-----------------------------------------------+------------+------------+
```

Copy and paste one of the bank names into the prompt and press enter to see the contents of the bank, e.g.

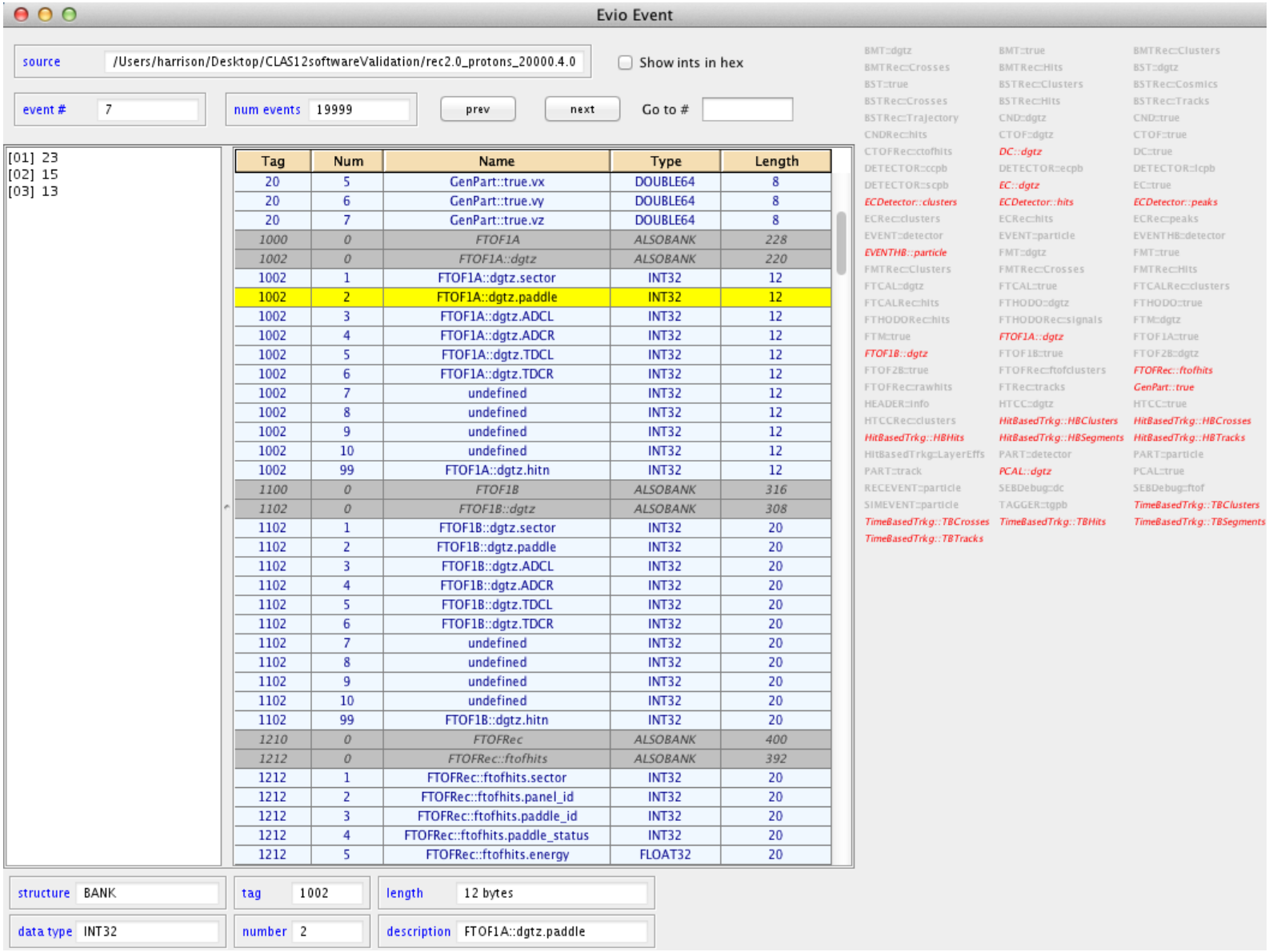


```
Press Enter for Next Event or Bank Name: FTOF1A::dgtz
*****>>>>> BANK FTOF1A::dgtz  >>>> SIZE = 7
          ADCL  (int)  :          832            7           13          164            0            3
          TDCR  (int)  :         1065         2493         2033         1242         3125         1713
          hitn  (int)  :            1            2            3            4            5            6
          ADCR  (int)  :         1729            6            8          349            0            9
        sector  (int)  :            1            2            2            1            2            1
        paddle  (int)  :            9           20            5           11           12           12
          TDCL  (int)  :         1412         2550         1763         1632         2335         2445
```

Working with evio files, part 2

Another way to browse through the banks is with ced (CLAS12 Event Displayer), do:

coatjava/bin/ced
then click Events → Open Event File... and open you evio file
then click Views → Evio Event and browse through the GUI:



Working with evio files, part 3

For a description of what a variable name means, browse through the xml files in coatjava/etc/bankdefs/clas12/ e.g. FTOF.xml:

```
<bank name="FTOF1A" tag="1000" info="Simulated FTOF Pannel 1a Hit Information">
  <section name = "true" tag = "1001" num = "0" info = "Digitized information">
      <column name="pid"     num="1"  type="int32"   info="ID of the first particle entering the sensitive volume"/>
      <column name="mpid"    num="2"  type="int32"   info="ID of the mother of the first particle entering the sensitive volume"/>
      <column name="tid"     num="3"  type="int32"   info="Track ID of the first particle entering the sensitive volume"/>
      <column name="mtid"    num="4"  type="int32"   info="Track ID of the mother of the first particle entering the sensitive volume"/>
      <column name="otid"    num="5"  type="int32"   info="Track ID of the original track that generated the first particle entering the sensitive volume"/>
      <column name="trackE"  num="6"  type="float64" info="Energy of the track"/>
      <column name="totEdep" num="7"  type="float64" info="Total Energy Deposited"/>
      <column name="avgX"    num="8"  type="float64" info="Average X position in global reference system"/>
      <column name="avgY"    num="9"  type="float64" info="Average Y position in global reference system"/>
      <column name="avgZ"    num="10" type="float64" info="Average Z position in global reference system"/>
      <column name="avgLx"   num="11" type="float64" info="Average X position in local reference system"/>
      <column name="avgLy"   num="12" type="float64" info="Average Y position in local reference system"/>
      <column name="avgLz"   num="13" type="float64" info="Average Z position in local reference system"/>
      <column name="px"      num="14" type="float64" info="x component of momentum of the particle entering the sensitive volume"/>
      <column name="py"      num="15" type="float64" info="y component of momentum of the particle entering the sensitive volume"/>
      <column name="pz"      num="16" type="float64" info="z component of momentum of the particle entering the sensitive volume"/>
      <column name="vx"      num="17" type="float64" info="x component of primary vertex of the particle entering the sensitive volume"/>
      <column name="vy"      num="18" type="float64" info="y component of primary vertex of the particle entering the sensitive volume"/>
      <column name="vz"      num="19" type="float64" info="z component of primary vertex of the particle entering the sensitive volume"/>
      <column name="mvx"     num="20" type="float64" info="x component of primary vertex of the mother of the particle entering the sensitive volume"/>
      <column name="mvy"     num="21" type="float64" info="y component of primary vertex of the mother of the particle entering the sensitive volume"/>
      <column name="mvz"     num="22" type="float64" info="z component of primary vertex of the mother of the particle entering the sensitive volume"/>
      <column name="avgT"    num="23" type="float64" info="Average time"/>
      <column name="hitn"    num="99" type="int32"   info="Hit1 Number"/>
    </section>

   <section name = "dgtz" tag = "1002" num = "0" info = "True Values">
    <column name = "hitn"  num = "99"  type = "int32"  info = "hit number"/>
    <column name = "sector"  num = "1"  type = "int32"  info = "sector number"/>
    <column name = "paddle"  num = "2"  type = "int32"  info = "paddle number"/>
    <column name = "ADCL"  num = "3"  type = "int32"  info = "ADC Left"/>
    <column name = "ADCR"  num = "4"  type = "int32"  info = "ADC Right"/>
    <column name = "TDCL"  num = "5"  type = "int32"  info = "TDC Left"/>
    <column name = "TDCR"  num = "6"  type = "int32"  info = "TDC Right"/>
  </section>
</bank>
```

Working with evio files, part 4

A thorough analysis of evio files can be done with groovy scripts.

```
import org.jlab.evio.clas12.*;
import org.jlab.clas.physics.*;
import org.jlab.clas12.physics.*;
import org.root.histogram.*;
import org.root.pad.*;
import org.root.func.*;
import java.lang.Math;

// open your evio files
EvioDataChain reader = new EvioDataChain();
reader.addFile("rec2.0_pip_0.0.evio");
reader.addFile("rec2.0_pip_1.0.evio");
reader.open();

// create new kinematic fitter, with beam energy 11.0 GeV
GenericKinematicFitter fitter = new GenericKinematicFitter(11.0, "X+:X-:Xn");

// define a histogram
H2D h_YvsX = new H2D("h_YvsX", "h_YvsX", 100, -5000, 5000, 100, -5000, 5000);

int eventCounter = 1;

// loop over events
while(reader.hasEvent() == true)
{
println "";
println "Event " + eventCounter + ":";
eventCounter++;

EvioDataEvent wholeEvent = reader.getNextEvent();
PhysicsEvent genEvent = fitter.getGeneratedEvent(wholeEvent);
PhysicsEvent recEvent = fitter.getPhysicsEvent(wholeEvent);

// loop over generated particles
for(int i = 0; i < genEvent.count(); i++)
{
Particle myParticle = genEvent.getParticle(i);
println "generated particle " + i + " has PID = " + myParticle.pid() + ", p = " + myParticle.p() + ", theta = " + Math.toDegrees(myParticle.theta()) + ", phi = " + Math.toDegrees(myParticle.phi());
}

// loop over reconstructed particles
for(int i = 0; i < recEvent.count(); i++)
{
Particle myParticle = recEvent.getParticle(i);
println "reconstructed particle " + i + " has PID = " + myParticle.pid() + ", p = " + myParticle.p() + ", theta = " + Math.toDegrees(myParticle.theta()) + ", phi = " + Math.toDegrees(myParticle.phi());
}

// read info from a bank
if(wholeEvent.hasBank("FTOF1A::true"))
{
EvioDataBank FTOF1A_true = wholeEvent.getBank("FTOF1A::true");
for(int i = 0; i < FTOF1A_true.rows(); i++)
{
double avgX = FTOF1A_true.getDouble("avgX", i);
double avgY = FTOF1A_true.getDouble("avgY", i);
h_YvsX.fill(avgY, avgX);
}
}

} // end loop over events

// define a canvas and draw the histogram
TGCanvas can = new TGCanvas("can", "can", 500, 500, 1, 1);
can.cd(0);
can.draw(h_YvsX);
```

Working with evio files, part 4

A thorough analysis of evio files can be done with groovy scripts.

```
import org.jlab.evio.clas12.*;
import org.jlab.clas.physics.*;
import org.jlab.clas12.physics.*;
import org.root.histogram.*;
import org.root.pad.*;
import org.root.func.*;
import java.lang.Math;

// open your evio files
EvioDataChain reader = new EvioDataChain();
reader.addFile("rec2.0_
reader.addFile("rec2.0_
reader.open();

// create new kinematic
GenericKinematicFitter

// define a histogram
H2D h_YvsX = new H2D("h

int eventCounter = 1;

// loop over events
while(reader.hasEvent()
{
println "";
println "Event " + even
eventCounter++;

EvioDataEvent wholeEven
PhysicsEvent genEvent =
PhysicsEvent recEvent =

// loop over generated
for(int i = 0; i < genE
{
Particle myParticle = g
println "generated part
}

// loop over reconstruc
for(int i = 0; i < recE
{
Particle myParticle = r
println "reconstructed
}

// read info from a ban
if(wholeEvent.hasBank("
{
EvioDataBank FTOF1A_tru
for(int i = 0; i < FTOF
{
double avgX = FTOF1A_tr
double avgY = FTOF1A_tr
h_YvsX.fill(avgY, avgX)
}
}

} // end loop over even

// define a canvas and
TGCanvas can = new TGCa
can.cd(0);
can.draw(h_YvsX);
```

run this code with: coatjava/bin/run-groovy filename.groovy

sample output:

…

Event 9993:

generated particle 0 has PID = 211, p = 6.94579247450511, theta = 27.43991691113383, phi = 16.336739080263495

reconstructed particle 0 has PID = 211, p = 6.970181459595072, theta = 27.433028920755145, phi = 16.322397410012414

Event 9994:

generated particle 0 has PID = 211, p = 3.398543940286258, theta = 30.947530277926862, phi = 26.314366484406264

reconstructed particle 0 has PID = 211, p = 3.419796880845797, theta = 30.972688628848815, phi = 26.26770642233741

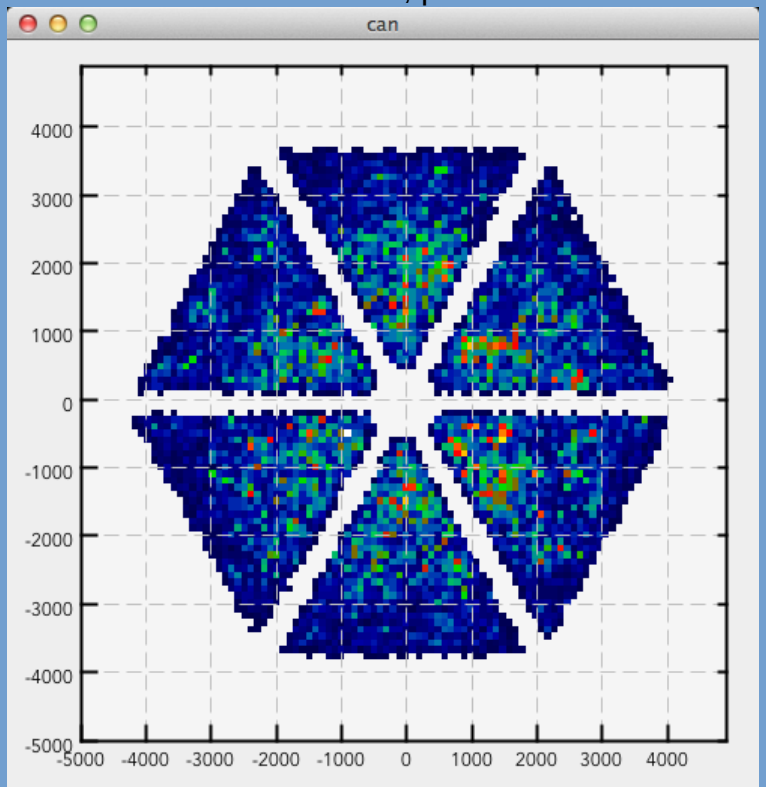
Event 9995:

generated particle 0 has PID = 211, p = 10.764975539054932, theta = 20.465776846939846, phi = -174.36618606279447

reconstructed particle 0 has PID = 211, p = 10.770962703054385, theta = 20.469946025858903, phi = -174.41547767318835

...

*See the documentation for examples of how to make this more efficient!



Some remarks:

- There are some issues with running groovy on the batch farm, this is being investigated

- Indexing to link tracks to hit numbers in the detector banks is currently not available but is in progress