Performing a nearline sort

Nigel Warr

May 2012

Contents

1 Introduction

The purpose of this document is to show how to perform a simple sort of Miniball data at CERN. I distinguish between "online" (performed by Marabou in real time), "offline" (performed by the user at their home institute as part of the full analysis for a publication) and "nearline" (sorts performed at CERN after data has been accumulated, to see how the experiment is going). This document only considers the nearline sorting.

There are three steps involved:

- Use offl_root_med to produce .root files from the .med files.
- Use rt_ana to loop through the data in the root files and produce a single root file containing a tree of the useful data.
- Use g_clx_ana to analyse this tree and generate the plots for the Coulex analysis.

2 Step 1: offl_root_med

For each .med file you want to include in the analysis, you have to run offl_root_med separately. There are actually two versions of this program, one for Coulex runs and the other for calibration runs. For Coulex analysis we always want the one for Coulex runs, which is *offl_root_med_run*. On pcepis40 and pcepis41 this program is installed in /usr/bin. You can find out which version is installed there using the command:

```
rpm -qf /usr/bin/offl_root_med_run
```

which will give an output like:

```
offl_root_med-20110712-1.el6.i686
```

The 20110712 is a date (12^{th} July 2011) but it is also a tag in my git tree, so if you know this number, I can figure out exactly which version you used.

Note that on pcepis40 and pcepis41, the /mbdata directory of the DAQ should be mounted via nfs, so you can read the .med files directly from there.

The syntax for offl_root_med is somewhat complicated, so there is a simple script *sort_data.sh* which calls it. This starts of with the definitions of where the executable and the calibration files are and what the basenames of the .med and .root files are. Then it loops over the specified runs. If the .root file already exists, that run is skipped, so delete or rename any file you want to overwrite.

For example:

```
/usr/bin/offl_root_med_run \\
      /mbdata/miniball/cern-110812/72Kr_104Pd_042.med \\
      -1 \\
      72Kr_run042 \\
      ../calfiles/DGF/Cal_dgf_2011.dat \\
      ../calfiles/Mesytec/Cal_adc_keV_2011.dat dummy.tdc \\
      ../calfiles/Angles/config_2011_jan_23rotated.dat \\
      6 \\
      0.75
```

where we use the executable in /usr/bin to process the .med file which is on the nfs-mounted directory. The "-1" indicates we process all events. We create an output "72Kr_run042.root" and use the "Cal_dgf_2011.dat" calibration file for the DGFs, "Cal_adc_keV_2011.dat" for the Mesytec MADC-32s and the "config_2011_jan_23rotate.dat" configuration for the angles. There is no TDC calibration (i.e. the file "dummy.tdc" does not exist). The window is 6 μs wide and the reference point is 0.75 μs.

At the end of this step, you should have one .root file for each .med file that you want to include in the analysis.

3 Step 2: rt_ana

The second step is to run *rt_ana*, which creates the *g_clx* tree. This is a root script, which could, in theory, be interpreted, but that would be far too slow, so we compile it instead. Older versions of this code required the user to hardcode the list of .root files to process in the source code and recompile, but now it reads the list from the file *rt_ana.dat*.

So the first step is to create a file *rt_ana.dat* with the list of .root files to process (i.e. those created by *offl_root_med_run* in the first step), one filename per line. Then do the following in root:

- gSystem→Load("rt_ana.so");
- rt_ana x;
- x.Loop("outputname.root");
- .q

where *outputname.root* is the filename you want for the output file. If the file already exists, it will be overwritten! If you leave out the filename, the default is *rt_ana.root*.
This should give you a root file with a *g_clx* tree and a few histograms.

4 Step 3: g_clx_ana

Once you have the *g_clx* tree, you can use the *g_clx_ana* code to generate Doppler-corrected background-subtracted histograms.

To do this, you need to write a script with some code like:

```
g_clx_ana x("72Kr_104Pd_data.root");
x.SetAp(72); // Projectile mass A = 72
x.SetAt(104); // Target mass A = 104
x.SetMeanBetaProjectile(0.035); // Mean beta for projectiles
x.SetMeanBetaTarget(0.035); // Mean beta for target recoils
x.SetCDDistance(30.6); // target-CD distance in mm
x.SetCDRotation(215.5); // Angle of CD
x.SetBackgroundFactor(-0.25); // Factor for background subtraction
                              // (must be negative)
x.Loop("72Kr_104Pd_ready.root");
```

Once you've done this, you get a root file *72Kr_104Pd_ready.root*. In it there is a 2-D histogram *pp1h*. Start root with the filename on the command line (so that it is automatically opened as *_file0*) and turn on the toolbar (View→Toolbar). Then click on the cutting tool (the scissors icon) and create a cut for the projectile. Then select "SaveAs" by left-clicking on the cut and save it as "pcut.root" (the option bit is left blank). Repeat for the target recoils, which should be saved to "tcut.root".

```
root 72Kr_104Pd_ready.root
TCanvas *c = new TCanvas();
c->ToggleToolBar();
c->ToggleEventStatus();
pp1h->Draw("colz");
...
click on the scissors and define a region for a cut, double clicking
on the last point to end
...
SaveAs -> pcut.root
...
same for target recoils as tcut.root
.q
```

Then you can rerun g_clx_ana again just as above, but this time it will read in the cuts.

The interesting histograms are *coulexTdc* and *coulexPdc*, which are the Doppler-corrected background-subtracted γ-ray spectra gated on the target and projectile cuts, respectively.