Data Analysis via LMF2Root and ROOT: A Rootorial

Part 1: Installation & General Information

- LMF2Root is available on the network drive
  - Z:\Installable Software\ LMF2Root
  - Or the lastest version is on Achim's website

    http://www.atom.uni-frankfurt.de/czasch/default_files/software/LMF_tools/lmf2root/
- ROOT can be found online at

  http://root.cern.ch/drupal/content/downloading-root
- There are a number of preexisting macros written by the Frankfurt and Auburn groups. These are available on the network drive (Z:\Installable Software\ LMF2Root\macros) and should be copied to the C:\root\macros directory.

Language

- ROOT – Is a program and object-oriented framework for data analysis.
- .root file – This is the file format used by ROOT. It can contain objects like histograms and the raw data.
- .lmf file – List Mode File. This is the file format used by Cobold.
- LMF2ROOT – Is the program we use to convert the lmf to a root. A large amount of the data analysis can also be performed in this program.
- Event – is the set of hits on the particle detector(s) that result from a projectile striking a target.

Language

- Presort – This is a minimal condition (set up by the user) to determine if the data is good or bad and allows us to reduce the size of the data-set.
- Reconstruction – This is Achim's routine that is used to reconstruct hits from incomplete data. This is also called resort but, I will try to avoid that term as it is ambiguous.
- CINT – Is the C/C++ interpreter that is included in ROOT. This allows you to write macros in ROOT.

Data Analysis Work Flow

1) Take Data with Cobold in LMF format
2) Run LMF2Root to convert the lmf file to a ROOT file. This may include a presort and reconstruction on the data.
   sort_and_write_NTuple.cpp
3) View the raw data with ROOT
4) Run LMF2Root with the root file as the input. This will allow you to perform the analysis on the presorted data. analysis.cpp
5) View the analyzed data with ROOT.
6) Rinse and Repeat

Part 2: LMF2Root

Controlling LMF2Root

LMF2Root uses a configuration file in C style syntax.
You can comment out statements with // or you can block comment statements with /* … */ .
All statements must be ended with a semicolon.

Basic Commands used in the configuration file

- Parameter statements allow you to import numbers into LMF2Root.

  Example: parameter 1200 33.25 ;//length acc. recoils [mm]
  This parameter can be accessed in LMF2Root via the variable parameter[1200]

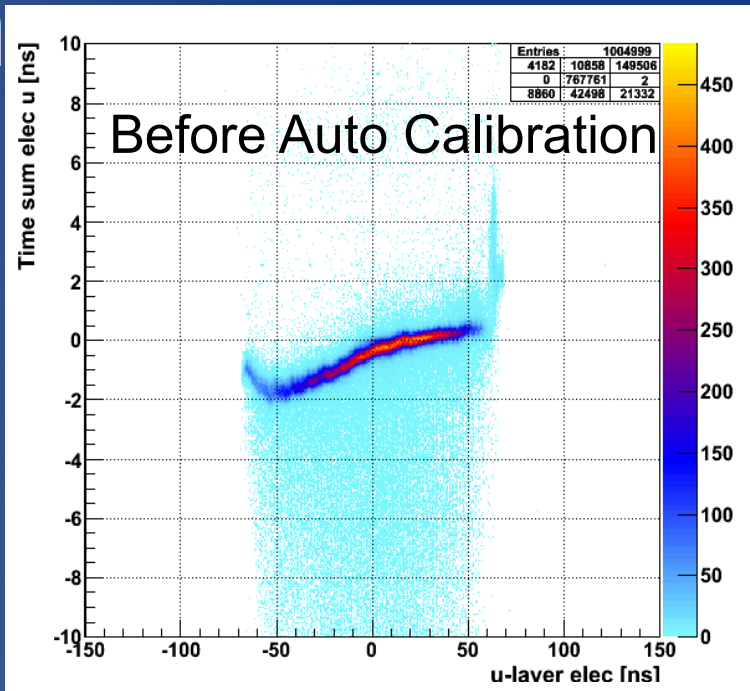
- execute "correction_table_rec.txt";
- set_root_output_file_name
- ReadLMF and readROOTfile

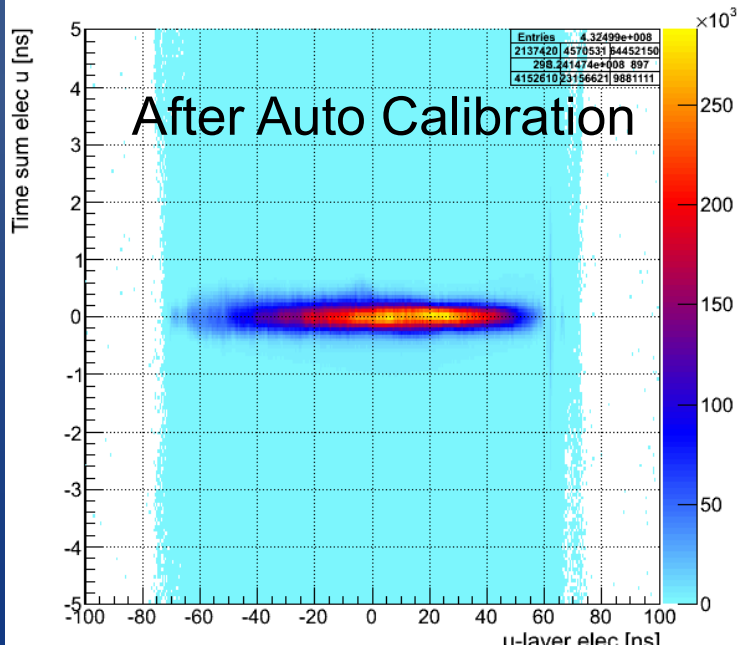
Basic Detector Parameters

```
parameter 300 2          ;//DLD/HEX (0=none; 1=DLD; 2=HEX)   E L E C T R O N - detector
parameter 301 0          ;//common mode (0=start, 1=stop)
parameter 302 2          ;//channel number for electron u1 (the first channel has index zero)
parameter 303 9          ;//channel number for electron u2
parameter 304 4          ;//channel number for electron v1
parameter 305 11         ;//channel number for electron v2
parameter 306 6          ;//channel number for electron w1
parameter 307 13         ;//channel number for electron w2
parameter 308 1          ;//channel number for electron MCP
parameter 309 1          ;//use MCP-signal (0 = no, 1 = yes)
parameter 310 0.5807           ;//conversion factor for u-layer [mm/ns]   0.60583
parameter 311 0.565750         ;//conversion factor for v-layer [mm/ns]   0.59010
parameter 312 0.558350         ;//conversion factor for w-layer [mm/ns]  0.58269
parameter 313 -74.5            ;//offset for timesum u [ns]
parameter 314 -74.25           ;//offset for timesum v [ns]
parameter 315 -76.1            ;//offset for timesum w [ns]
parameter 316 0.725            ;//position offset w-layer [ns]
parameter 317 4.         ;//width of timesum u [ns] (half width at bottom)
parameter 318 4.         ;//width of timesum v [ns] (half width at bottom)
parameter 319 4.         ;//width of timesum w [ns] (half width at bottom)
parameter 320 25.        ;//deadtime anode [ns]
parameter 321 25.        ;//deadtime MCP [ns]
Parameter 32243.5        ;//MCP-radius [mm] (choose always bigger than real radius)
parameter 323 75.        ;//runtime [ns] (max. runtime on largest layer)
parameter 324 0.         ;//1.0 shift detector to center MCP x-direction
parameter 325 0.         ;//0.5 shift detector to center MCP y-direction
parameter 326 0          ;//auto calibration (0 = no, 1 = yes)
parameter 327 1          ;//use resort routine (reconstruction) (0 = no, 1 = yes)
parameter 328 1          ;//use sum correction (0 = no, 1 = yes)
parameter 329 1          ;//use pos correction (0 = no, 1 = yes)
```

Sum and Position Correction Calibration

- Run the sum and position correction on each detector. This auto calibration must be run separately for each detector. Running the auto calibration produces the two text files (one for each detector) with the sum and position correction tables.





Recoil detector:
parameter 226 0 ;//auto calibration (0 = no, 1 = yes)
Electron detector:
parameter 326 0 ;//auto calibration (0 = no, 1 = yes)

Example of the Sum and Position correction table:
//sum correction_points layer u on elec detector:

set_point sum elec u -76.000000 -0.969072;
set_point sum elec u -72.833333 -0.969072;
set_point sum elec u -69.666667 -0.969072;
set_point sum elec u -66.500000 -0.969072;
set_point sum elec u -63.333333 -1.091545;
set_point sum elec u -60.166667 -1.433819;

Sort_and_write_NTuple.cpp

- Raw data from the lmf file is stored in a 2d array called tdc_ns[channel number][hit number]. As the name implies all data in this array is stored as times relative to the trigger.
- Higher level data, such as positions, is stored in a data structure call Ueber
  - Ueber.rec and Ueber.elec stores the data for their respective detectors. Example: Ueber.rec.x[hit number]
  - Useful members of Ueber are:
    - Positions: Ueber.rec.x[1], Ueber.rec.y[1]
    - Ueber.elec.time[1] is the time on the electron MCP (relative to the trigger)
    - Ueber.elec.method[1] is the method used to reconstruct the hit

Sort_and_write_NTuple.cpp

- sort_and_write_NTuple.cpp is called for every event. If you decide that you would like to write a particular event to the resulting root file then set the writeNTuple variable to true.
- The data is written to the root file via an array called NtupleData.

  Example: NTupleData[0] = Ueber.rec.x[1];

  The root file format requires you to name each “Branch” of the data.

  Example: Hist->NTupleD(0,"Data","CH4_DATA", "recoil_X:next_thing_in_the_NTupleData_array:and_so_on", 32000, NTupleData);

Analysis.cpp

- This function is called for every event in the root file. Running this function is optional. One could choose to do all of the analysis is ROOT (more on this later).
- In the default version of LMF2Root this is essentially and empty function.
- It is possible to write another root here also.

Histograms

- 1D Histograms
- Hist->fill1(id_number,"name",variable_to_be_plotted, Weight, "Title", X_Total_number_of_bins, X_Lower_bound, X_Upper_bound., "X axis title", ”Root file directory”)
- Example: Hist->fill1(255,"rec1tof_ntuple",rec1_tof,1.,"Recoil TOF #1",50050,-100.,100000.,"rec1_tof [ns]","ntuple");
- 2D Histograms
- Hist->fill2(id_number,"name", X_variable_to_be_plotted, Y_variable_to_be_plotted, Weight, "Title", X_Total_number_of_bins, X_Lower_bound, X_Upper_bound, "X axis title", Y_Total_number_of_bins, Y_Lower_bound, Y_Upper_bound, "Y axis title", ”Root file directory”)
- Example: Hist->fill2(250,"rec1xy_ntuple",rec1_x,rec1_y,1.,"Recoil position #1",400,-70.,70.,"rec1_x [mm]",400,-70.,70,"rec1_y [mm]","ntuple");
- 3D Histograms are also available and the syntax scales in the obvious manner.

Part 3: ROOT

Root Macros

- The macros are written in C/C++ and are executed by the CINT interpreter but, they can also be compiled. Compiled macros will run much faster (on the order of a thousand times faster).
- The main set of macros are located in MoritzLutzAchimTill.C and Joshua_Macros.C
- To load and unload a macro file you can type in .L or .U followed by the file name.
- The rootlogon.C macro runs when you start ROOT.
- The fileopen.C macro runs when you open a root file.
- There is a help function that will give you a basic list of available macros. It can be accessed by typing help()

Useful Macros

polar(ErrorBars=true) (use this when the histogram is already in pad)
(Detects ranges automaticly. Will plot fit, when histogram was fitted before. When giving range in cos(theta), then it will mirror the distribution to make a full circle.)
Circle() (draws a circle at center 0,0)
prox(Nbr+-Bins=-1) (if nbr+-bins is >-1 then it draws the projection in a separat canvas online with +- nbrbins around the mouse)
proy(Mbr+-Bins=-1) (same thing here for projection to Y)
Int() (gives U the nbr of entries interactively)
gaus() (makes a gaus fit in a 1D histo interactively)
Canv(int tilehor=1, int tilever=1) (creates a Canvas with nbr of hor and nbr of ver subpads)
pipico2(M1 amu, M2 amu, charge1 au, charge2 au, EField V/cm, AccReg mm)
(Draw PIPICO-lines to histogram currently displayed)
fit_step(height, sharpness, offset1, offset2) (Fits a step function to a 1d histogram)
pipico_3accel(MassP1_amu, ChargeP1_au, MassP2_amu, ChargeP2_au, EField1_Vpcm, LenReg1_mm, EField2_Vpcm, LenReg2_mm, EField3_Vpcm, LenReg3_mm)
(this will draw a pipico line, for two equal particles for and a spectrometer with up to three acceleration regions of any lenght)
hist2ascii(FileName) (Converts the current Histogram to ASCII)
save() (saves the current histogram as a pdf in a sub-directory call graphs)

ROOT Session and CINT

- The ROOT Session window supports tab completion.
- It also allows you to overload functions. Example:
  circle() (draws a circle at center 0,0)
  circle(x,y) (draws a circle with radius at center x,y)
  circle(center=0,0,radius) (draws a circle with radius at center)

The END