Don’t Panic! Things will be fine.

LMF2Root 2 ... in a Nutshell

Till Jahnke
jahnke@atom.uni-frankfurt.de
IKF, University of Frankfurt

January 12, 2017

Abstract

A short introduction to the new features and config file commands of LMF2Root 2 maximizing your offline-analysis pleasure!

Feel free to add or modify LMF2Root. However, if you do so, please comply with the following:

- If you find a bug (yeah, there are lots of bugs to be found!) report the bug (and maybe the fix) so the ”official” version becomes bug-free at some point.
- If it is just a ”hack” to deal with exactly your problem/dataset etc. **do not distribute this version** (please!).
- If it is something which turned out to be useful in your case and might be of use for others, as well, spend some time to program it properly or to explain it to someone who is in charge of the ”official” LMF2Root-version in order to make him or her implement your idea properly.

Now: enjoy!

Contents

1 LMF2Root 2 - Concept

LMF2Root is a c++ and Root[1] based data analysis package designed to provide functionality for a typical COLTRIMS analysis using delayline detectors. An initial task during an analysis is the extraction of positions of impact from the raw data. LMF2Root does "resorting", also called reconstruction, which reconstructs detector events for multiple hits applications. Detector time-sums are corrected and if a hex-detector is used it can be linearized.

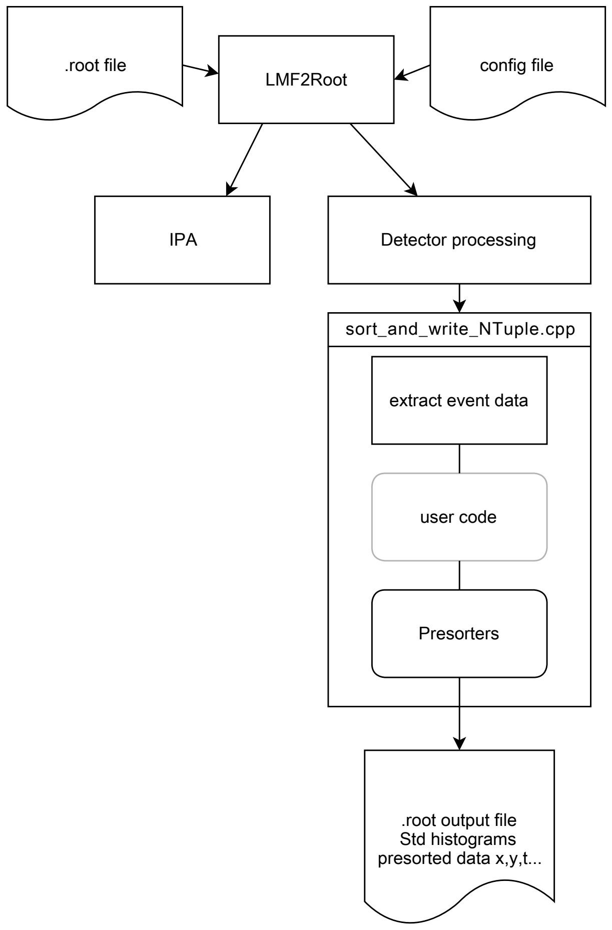

Figure 1: Typical work flow during the first stage of a COLTRIMS analysis.

Since version 2 LMF2Root includes the COLTRIMS analysis helper library (ColAHelL) and the interactive parameter adjustment tool (IPA). ColAHelL provides functions for typical tasks that have to be done during an analysis. The main goal of ColAHelL is to standardize lowlevel functions, which simplifies sharing of user

[1]Root is a data analysis and visualization page developed at CERN. See http://root.cern.ch/drupal/

code and experiment parameters. The main features are classes for obtaining times-of-flight and momenta from raw data. The user should be able, for most standard COLTRIMS-setups, to obtain momenta, energies, typical emission angles etc. and a vast set of standard histograms without adding any user code to the analysis. Predefined presorters can be invoked in order to reduce the amount of raw data.

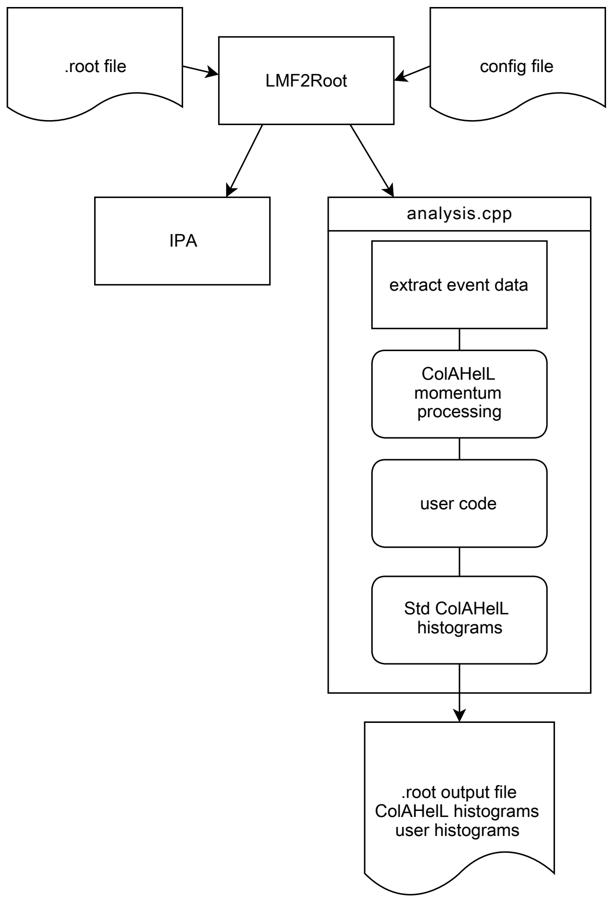

Figure 2: Typical work flow during later stages of a COLTRIMS analysis.

During an intermediate stage of the analysis the user typically needs to fine tune certain parameters. In order simplify that IPA was added to LMF2Root 2. When IPA is started it will collect a user defined number of events into PC-RAM. These events are processed in a user accessible eventloop and user defined histograms can be updated on the fly. Thus, the user can change parameters online and directly view the effect of these changes on the histograms displayed.

A typical analysis consists of two steps. In a first step the raw TDC-data is converted to positions of impact and times-of-flight. During this stage it is in many cases helpful to reduce the amount of initial data by checking if a raw event

”roughly” complies with what is expected to occur due to the reaction under investigation. This process is called ”presorting”. Figure 1 shows a typical work flow for this analysis step. The output of this step is a root-file consisting of the presorted data and a set of standard histograms. The user may add additional code and histograms by editing the file ”sort_and_writeNTuple.cpp” in the LMF2Root VS-project.

After the first stage the user has a .root-file containing the presorted data of interest. During the next steps this file is used as an input to LMF2Root. When a root-file is specified as an input file LMF2Root will run ”analysis.cpp” (instead of ”sort_and_writeNTuple.cpp”) that contains the momentum calculations using ColAHelL and a set of standard histograms. In this file the user needs to add experiment specific analysis steps and histograms. In case ColAHelL is used, the user can access the following properties of the measured particles. If present in the experiment, an array of electrons `e[n]` contains the measured electron data. The electron class consists of the raw data, and physics quantities such as momenta, emission angles and energies as obtained by ColAHelL. If ions were measured (and if the user provided corresponding information in the config file) an array of ions `r[n]` is available just similar to the array of electrons. Furthermore, if occurring in the experiment and defined properly in the config file, a diatomic molecule object `mol`, containing most properties of interest, is accessible to the user. The raw eventdata (after detector reconstruction and presorting) is available in the `CH_evt` object. If information on the reaction was provided in the config file the properties of the reaction for the current event can be accessed in `cur_reaction`, also the name of the reaction is available as a char string called `reaction_dir`. This may be used to create folders for each reaction in the output root-file.

2 List of config file commands

The config file is used to set parameters and define input and output files. Special commands invoke the ”interactive parameter adjustment”-tool (IPA) and set to IPA-parameters. The spectrometer type and geometry is set using the config file and predefined presorters can be triggered. Also, LMF2Root uses C style syntax. Semicolons (;) are reqiured at the end of line and lines can be commented out with // and `/* comments */`.

Note: The order of commands is important. It is a good idea to keep the order of the command the same as in the example config file.

2.1 LMF2Root misc settings

```
parameter 2 (0 = no, 1 = yes)
```

If set to `1` the .lmf-file’s header information is displayed on the screen when reading the .lmf-file.

```
parameter 9
```

Channel offset for rising signal edges: if rising and falling edges were recorded and written to the .lmf-file, the timing information of the rising (i.e. trailing) edge is stored in...?

`parameter 10 (0 = no, 1 = yes)`

Enable multicore support if CPU has multiple cores to increase data processing. Did cause crashes in (really) early versions, so if strange things happen disabling multicore support might help. Multi-core support is only implemented in the first stage of LMF2Root ("sort_and_writeNTuple.cpp").

`parameter 11`

Filling mode of automatically created *detector* histograms:

- `0`: do not create/fill standard histograms (faster).
- `1`: normal operation.
- `x`: create all standard histograms but fill only with every xth event. (faster)

`parameter 15 (0 = no, 1 = yes)`

Perform anti-Moire dithering. This adds a small random number (on the order of the TDC resolution) to all TDC signals to smooth out the data.

`parameter 49 (0 = no, 1 = yes)`

Write a new listmode .lmf-file including sorted TDC data. The filename of the newly created file can be set using the `set_LMF_output_file_name` command.

2.2 Channel demixing

`parameter 50 (0 = no, 1 = yes)`

Demix channel (0 = no, 1 = yes)

`parameter 51`

Channel to demix (channels are counted starting from "0")

`parameter 52`

Demix region (times after this time belong to other channel)

`parameter 53`

Number of channel to write the demixed data to.

2.3 Output file settings

`parameter 54`

Sets the number of channels in the output .lmf-file (in case it is created, see parameter 49). If set to `0` then number of channels of the original .lmf-file is used.

`parameter 55`

Sets the number of hits per channel in the output .lmf-file (in case it is created, see parameter 49). If set to `0` then number of hits of the original .lmf-file is used.

`parameter 56`

Number of events to write to the .root-file (`0` = infinit).

`parameter 57 (0 = no, 1 = yes)`

Start a new .root-file when number of events is (parameter[56]) (0 = no, 1 = yes).

`parameter 58`

Number of events to be read from .lmf-file (`0` = infinit).

`parameter 60`

Projectile detector: max. number of hits/event to be written to the .root-file.

`parameter 61`

Ion detector: max. number of hits/event to be written to the .root-file.

`parameter 62`

Electron detector: max. number of hits/event to be written to the .root-file.

2.4 Detector settings

LMF2Root supports up to three detectors. These are the projectile, the ion (sometimes called recoil) and the electron detector to cover typical experiments using synchrotrons, lasers and ion beams. The following parameters need to be set for each detector used. The projectile detector has the parameter range 1xx, the ion detector the range 2xx and the electron detector the parameter range 3xx.

`parameter x00`

Defines the type of detector: use 0 if the detector is not present, 1 for a DLD (QUAD or Square) anode and 2 for a HEX anode.

`parameter x01`

Defines whether the TDC is common start or common stop. Most likely you are using a HPTDC, which is common start (set to 0). If you are using an old TDC8 common stop is most probably your choice (set parameter to 1).

`parameter x02 - x08`

Provides the TDC channel numbers for the detector's anode and MCP. Unused channels should be set to -1.

`parameter x09 (0 = no, 1 = yes)`

Defines if the MCP signal is replaced during reconstruction.

`parameter x10`

Defines the conversion factor of the u-layer of the detector in mm/ns. Some typical values are:

Type	u-layer	v-layer	w-layer
DLD 40	0.61	0.58	-
DLD 80	0.55	0.52	-
DLD 120	0.42	0.4	-
HEX 40	0.53	0.53	0.51
HEX 80	0.55	0.53	0.52
HEX 90	0.49	0.48	0.48
HEX 90 (2/3)	0.35	0.35	0.35
HEX 90 ”Batman”	?	?	?
HEX 90 ”Batman” (2/3)	?	?	?
HEX120	?	?	?
HEX120 (2/3)	0.38	0.38	0.38

Note: your value might differ, even though it is the same type of detector!

`parameter x11`

Same as parameter $x10$ for the v-layer.

`parameter x12`

Same as parameter $x10$ for the w-layer.

`parameter x13`

Offset of the time sum of the u-layer. The offset is *subtracted* from the time sum value, which results in a time sum peak centered about zero.

`parameter x14`

Offset of the time sum of the v-layer. The offset is *subtracted* from the time sum value, which results in a time sum peak centered about zero.

`parameter x15`

Offset of the time sum of the w-layer. The offset is *subtracted* from the time sum value, which results in a time sum peak centered about zero.

`parameter x16`

When employing a hexagonal delayline anode, on of the layers needs to be shifted in position to match the other two layers. This parameter sets the position offset in ns. The exact value is provided as the auto calibration feature of LMF2Root is used.

`parameter x17`

The time sum value is used to identify whether a certain combination of hits belong to a valid detector event. This parameter sets the width (in ns) of the corresponding gate for the u-layer. The time sum is expected to peak at 0 ns (see parameters x13, x14 and x15). The gate is $0 \pm width$.

`parameter x18`

Same as parameter $x17$ for the v-layer.

`parameter x19`

Same as parameter $x17$ for the w-layer.

`parameter x20`

In many cases, missing detector signals can be reconstructed, but the reconstruction routine needs to know some details about the detector signals. This parameter sets the expected deadtime of the anode in ns, i.e. the time within which no other signal is expected to occur after a hit.

`parameter x21`

Same as parameter $x20$, but for the MCP signals.

`parameter x22`

A maximum radius (in mm) provided by using this parameter helps reconstruction to omit completely unrealistic hits. Choose a radius that is always a little bigger than the real MCP radius.

`parameter x23`

The max runtime (in ns) is defined by the total signal propagation time across the largest layer of the detector (time for signal to travel from x_1 to x_2). Providing the maximum runtime optimizes performance of the reconstruction.

`parameter x24`

Overall shift (in mm) of the detector to center MCP in x-direction.

`parameter x25`

OVerall shift (in mm) of the detector to center MCP in x-direction.

`parameter x26 (0 = no, 1 = yes)`

Enable auto calibration. Auto calibration provides the parameters the correctly shift and stretch the layers of the anode. Furthermore a new lookup table file for time sum correction is written after running LMF2Root with auto calibration enabled. Auto calibration will need to be run several times until it converges onto the correct values (conversion factor for v and w layer [mm/ns] and position offset w-layer), **Note: only one detector at a time can be auto calibrated!**

```
parameter x27 (0 = no, 1 = yes)
```

Enable advanced reconstruction and resorting.

```
parameter x28 (0 = no, 1 = yes)
```

Enable time sum correction by using lookup table created in a previous run using the auto calibration feature.

```
parameter x29 (0 = no, 1 = yes)
```

When employing a hexagonal anode the position information of the three layers can be used to optimize the detector's linearity. Enable by setting this parameter to 1.

After all parameters are set, the parameters need to be transferred into the execution units of LMF2Root. This is done, as the line

```
feed_parameters_into_sorters;
```

is added to the config file **after** the parameter definitions and **before** the time sum correction table definitions (see next chapter).

2.5 Time sum correction definitions

The *time sums* of a detector correspond to the overall time the signals of a hit travel on the wire of a layer of the delayline detector. They are obtained as:

$$
\begin{aligned}
t_{sum_u} &= t_{u1} + t_{u2} - 2 \cdot t_{MCP} \\
t_{sum_v} &= t_{v1} + t_{v2} - 2 \cdot t_{MCP} \\
t_{sum_w} &= t_{w1} + t_{w2} - 2 \cdot t_{MCP}
\end{aligned}
$$

Ideally the time sum value is a property of the layer itself and of the experimental setup (i.e. cable lengths etc.). It turns out that the time sums depend on the position of impact and thus are not a constant for a positions on the detector. In order to maximize detector performance the resort/reconstruction algorithm is able to create a lookup table (in a first step) and (in later runs) use this lookup table to correct for these deviations from the mean time sum value for different positions of impact on the detector. The lookup table is created by switching on parameters $x26$ for a detector. Please note that only one detector at a time can be calibrated. After processing an input data file with parameter $x26$ being enabled, a file called `correction_table_rec.txt` is created (the corresponding files for electron and projectile detectors are named `..._elec.txt` and `..._proj.txt`). The user must add the following lines to the config file to use the correction lookup table:

```
execute "correction_table_rec.txt";
execute "correction_table_elec.txt";
execute "correction_table_proj.txt";
```

If less than three detectors are used, the line belonging to the missing detector needs to be omitted.

2.6 COLTRIMS/spectrometer settings

In order to use the advanced features of LMF2Root 2 the user needs to provide information on the experimental setup. As a COLTRIMS experiment consists of a position of impact and a time-of-flight measurement, the time-of-flight of each particle needs to be extracted from the raw data. LMF2Root can do this automatically in case enough information is provided. By using the `get_tof_from` command the required information can be specified. Different experimental schemes are supported. A typical scheme consists of a bunchmarker (e.g. a signal from a synchrotron or a photo diode):

```
get_tof_from BM TDCchannel period shift
```

The user needs to set the `TDC-channel` the bunchmarker signal was recorded on. If the signal is periodic a `period` can be set. This results in the electron time-of-flight to be calculated as modulo(period). An (optional) `shift` can be given that is added prior to the modulo operation to the electron time-of-flight. The times-of-flight are calculated as follows in this case:

$$\begin{aligned} e_{tof} &= fmod(e_{MCP} + t_{shift} - t_{BM} + 1000 \cdot BM_{period}, BM_{period}) \\ r_{tof} &= r_{MCP} - e_{MCP} + e_{tof} \\ p_{tof} &= p_{MCP} - e_{MCP} + e_{tof} \end{aligned}$$

For cases where either no electron is recorded or the reference signal's period is large compared to all expected times-of-flight the `period` setting can be omitted resulting in the times-of-flight being calculated in the following manner:

$$\begin{aligned} e_{tof} &= e_{MCP} - t_{BM} \\ r_{tof} &= r_{MCP} - t_{BM} \\ p_{tof} &= p_{MCP} - t_{BM} \end{aligned}$$

In ion/atom collision experiments it is typical to use the projectile detector as a reference for times-of-flight calculations. In this case use the following command:

```
get_tof_from PROJ shift
```

Now the times-of-flight are obtained as (the `shift` is optional and can be omitted):

$$\begin{aligned} e_{tof} &= e_{MCP} - p_{MCP} + t_{shift} \\ r_{tof} &= r_{MCP} - p_{MCP} + t_{shift} \\ p_{tof} &= p_{MCP} - p_{MCP} + t_{shift} \end{aligned}$$

In rare cases no reference signal is available at all. In this case the electron timing signal can be used as a reference yielding rudimentary information e.g. on the ion time-of-flight:

```
get_tof_from ELEC
```

The times-of-flight are then computed as:

$$
\begin{aligned}
e_{tof} &= e_{MCP} - e_{MCP} = 0 \\
r_{tof} &= r_{MCP} - e_{MCP} \\
p_{tof} &= p_{MCP} - e_{MCP}
\end{aligned}
$$

The spectrometer geometry and the applied fields are defined using the following commands:

```
set_spectrometer ARM regions len1_mm E1_Vcm len2_mm E2_Vcm..
```

The ion arm of the spectrometer is addressed as `ION_ARM`, while the electron arm properties are specified by using `ELEC_ARM`. This parameter is followed by the number of electrostatic regions for the given arm. Up to three regions per arm may be set by providing the lengths in mm and the corresponding electric fields in V/cm. Electric fields of 0 V/cm are allowed yielding a drift region. Furthermore, ColAHelL supports retarding regions where (typically electrons!) are decelerated. A retarding region is defined by employing a negative value for the electric field.

Long ion spectrometers are supported, too. In this case the linear approximation for the conversion of time-of-flight to momentum is assumed. In this case the use needs to set the spectrometer to:

```
set_spectrometer ION_ARM_LINEAR E_Vcm
```

The E-field at the target region needs to be provided as `E_Vcm`.

In many cases a homogenous B-Field is superimposed. This B-field is set by using the command:

```
set_spectrometer B_ns wiggle_period_in_ns B_direction;
```

In this case the strength of the B-field is provided in nanoseconds of the gyration period of an electron. The `B_direction` can be set to 0 or 1 for clockwise or counter clockwise gyration of an electron. If the B-field is known in Gauss it can be set using:

```
set_spectrometer B_Gauss B_in_G B_direction;
```

In many cases the raw measured particle data needs to be shifted or stretched. This can be done for each detector by employing, e.g. for electrons:

```
set_par_elec shX shY shT strX strY strALL
```

Prior to other processing, the electron raw data can be shifted in x,y,t by `shX`, `shY` and `shT` and the detector image is stretched in x and y direction by `strX` and `strY`. Additionally a total stretch of `strALL` can be applied, which simple scales the detector size by `strALL`.

After stretching and shifting the detector image can be rotated by `angle_deg` degrees using:

```
set_rot_elec angle_deg;
```

The corresponding commands for ions and projectiles are:

```
set_par_rec shX shY shT strX strY strALL
set_rot_rec angle_deg;
set_par_proj shX shY shT strX strY strALL
set_rot_proj angle_deg;
```

The momenta obtained from the raw data can be shifted and stretched, as well. Use the following command (e.g. for the electron detector):

```
set_mom_par_elec shX shY shZ strX strY strZ strALL MirX MirY MirZ;
```

`shX`, `shY` and `shZ` will shift the momenta in x-, y- or z-direction, and the momenta are stretched by the factor `strX`, `strY` and `strZ` in the corresponding direction. Furthermore, the data can be mirrored at the x- and the y-axis if - for example - the wires of the detector were switched during the experiment. `MirX`$= 1$ will mirror the momenta in x-direction (i.e. $x \rightarrow -x$), a value of 0 will leave the corresponding direction unaffected.

2.7 Presorters

LMF2Root incorporates a set of predefined presorters to write only a subset of the original .lmf-data to the .root-file. A presorter is invoked by using the `new_presorter` command. The structure of the command is:

```
new_presorter TYPE CHANNEL parameters...
```

`TYPE` is a text string, such as ANY or RECELECTOF, which specifies the presorter to be invoked. `CHANNEL` sets a numerical variable called `reaction` in the output root file, which is used to label different reaction channels. `parameters` are numerical arguments that are specific to each presorter.

2.7.1 ANY presorter

```
new_presorter ANY chan
```

The `ANY` presorter simply writes all events from the input data file to the output file marking these events by setting the ”reaction” in the output event to `chan`. A maximum number of hits, as specified by parameters $60-62$, is written to the output file.

2.7.2 Electron/Ion time-of-flight presorter

```
new_presorter RECELECTOF chan etof_center etof_width rtof_center rtof_width
```

This presorter searches events with a certain combination of recoil and electron times-of-flight, i.e. for a combination of electron and ion hits within:

$$e_{tof\ center} - e_{tof\ width} < e_{tof} < e_{tof\ center} + e_{tof\ width}$$
$$r_{tof\ center} - r_{tof\ width} < r_{tof} < r_{tof\ center} + r_{tof\ width}$$

If such hits are found these are written to the output file marked with the reaction number `chan`. If several electron hits are found within in the time window specified up to `parameter[62]` electrons are written to the output file.

2.7.3 PIPICO presorter

There are several different version of the pipico (Photo-Ion Photo-Ion COincidence) presorter, but in general the pipico presorter allows us to a define gate on the first ion's time-of-flight and the second ion's time-of-flight in coincidence with one another. The number of parameters provided by the user defines what exact implementation of the presorter is going to be used (i.e. the pipico presorter command is overloaded).

```
new_presorter PIPICO chan etof_center etof_width mass1_amu charge1_au
               mass2_amu charge2_au pipico_width
```

This presorter requires the `set_spectrometer` and `get_tof_from` commands to be correctly defined (see section 2.6). The expected times-of-flight of the two ions are obtained from the spectrometer settings (i.e. fields and lengths) and the masses and charges of the two ions. `mass1_amu` is the mass of the first ion in atomic mass units (e.g. nitrogen is 14), `charge1_au` is the first ion's charge in a.u. (e.g. singly charged ion has charge 1). The presorter checks for combinations of electron and ion hits that occur within:

$$e_{tof\ center} - e_{tof\ width} < e_{tof} < e_{tof\ center} + e_{tof\ width}$$
$$fabs(r2_{tof} - t2(r1_{tof})) < pipico_{width}$$

Here $t2$ is a function that calculates the expected time-of-flight of the second ion from the measured time-of-flight of the first ion and the spectrometer properties (using conservation of momentum). In case of nonequal ions both possible combinations of ion properties are checked.

If several electron hits are found within the given time window up to `parameter[62]` electrons are written to the output file. Valid events are flagged as reaction `chan`.

```
new_presorter PIPICO chan etof_center etof_width mass1_amu charge1_au
               mass2_amu charge2_au efield_Vcm acc1_mm pipico_width
```

PIPICO presorter employing a simple spectrometer with just a single acceleration region of length `acc1_mm` (given in mm) with an E-field of `efield_Vcm` (in V/cm). This version does not require `set_spectrometer` to be defined.

The expected times-of-flight for the two ion are obtained from these spectrometer settings and the masses and charges of the two ions. `mass1_amu` is the mass of the first ion in atomic mass units (e.g. nitrogen is 14), `charge1_au` is the first ion's charge in a.u. (e.g. singly charged ion has charge 1). The presorter checks for combinations of electron and ion hits that occur within:

$$e_{tof\ center} - e_{tof\ width} < e_{tof} < e_{tof\ center} + e_{tof\ width}$$
$$fabs(r2_{tof} - t2(r1_{tof})) < pipico_{width}$$

Here $t2$ is a function that calculates the expected time-of-flight of the second ion from the measured time-of-flight of the first ion and the spectrometer properties (using conservation of momentum). In case of nonequal ions both possible combinations of ion properties are checked.If several electron hits are found within the given time window up to `parameter[62]` electrons are written to the output file. Valid events are flagged as reaction `chan`.

```
new_presorter PIPICO chan etof_center etof_width mass1_amu charge1_au
               mass2_amu charge2_au efield1_Vcm acc1_mm
               efield2_Vcm acc2_mm pipico_width
```

Same PIPICO presorter as before but employing a spectrometer with two different acceleration regions of lengths `acc1_mm` and `acc2_mm` (given in mm) with E-fields of `efield1_Vcm` and `efield2_Vcm` (in V/cm). E-field values of 0 V/cm are recognized yielding a drift region.

The expected times-of-flight for the two ion are obtained from these spectrometer settings and the masses and charges of the two ions. `mass1_amu` is the mass of the first ion in atomic mass units (e.g. nitrogen is 14), `charge1_au` is the first ion's charge in a.u. (e.g. singly charged ion has charge 1). The presorter checks for combinations of electron and ion hits that occur within:

$$e_{tof\ center} - e_{tof\ width} < e_{tof} < e_{tof\ center} + e_{tof\ width}$$
$$fabs(r2_{tof} - t2(r1_{tof})) < pipico_{width}$$

Here $t2$ is a function that calculates the expected time-of-flight of the second ion from the measured time-of-flight of the first ion and the spectrometer properties (using conservation of momentum). In case of nonequal ions both possible combinations of ion properties are checked.If several electron hits are found within the given time window up to `parameter[62]` electrons are written to the output file. Valid events are flagged as reaction `chan`.

```
new_presorter PIPICO chan etof_center etof_width mass1_amu charge1_au
               mass2_amu charge2_au efield1_Vcm acc1_mm
               efield2_Vcm acc2_mm efield3_Vcm acc3_mm pipico_width
```

Same PIPICO presorter as before but employing a spectrometer with three different acceleration regions of lengths `acc1_mm`, `acc2_mm` and `acc3_mm` (given in mm) with E-fields of `efield1_Vcm`, `efield2_Vcm` and `efield3_Vcm` (in V/cm). E-field values of 0 V/cm are recognized yielding a drift region.

The expected times-of-flight for the two ion are obtained from these spectrometer settings and the masses and charges of the two ions. `mass1_amu` is the mass of the first ion in atomic mass units (e.g. nitrogen is 14), `charge1_au` is the first ion's charge in a.u. (e.g. singly charged ion has charge 1). The presorter checks for combinations of electron and ion hits that occur within:

$$e_{tof\ center} - e_{tof\ width} < e_{tof} < e_{tof\ center} + e_{tof\ width}$$
$$fabs(r2_{tof} - t2(r1_{tof})) < pipico_{width}$$

Here $t2$ is a function that calculates the expected time-of-flight of the second ion from the measured time-of-flight of the first ion and the spectrometer properties (using conservation of momentum). In case of nonequal ions both possible combinations of ion properties are checked. If several electron hits are found within the given time window up to `parameter[62]` electrons are written to the output file. Valid events are flagged as reaction `chan`.

2.7.4 Electron/Ion time-of-flight with projectile position presorter

```
new_presorter PROJ_RECTOF chan etof_center etof_width rtof_center rtof_width
              proj_x_center proj_y_center proj_radius
```

This presorter searches the event for a certain combination of recoil and electron times-of-flight and projectile positions of impact, i.e. for a combination of electron and ion hits within:

$$e_{tof\ center} - e_{tof\ width} < e_{tof} < e_{tof\ center} + e_{tof\ width}$$
$$r_{tof\ center} - r_{tof\ width} < r_{tof} < r_{tof\ center} + r_{tof\ width}$$

Furthermore a projectile hit with an impact within a radius of `proj_radius` around [`proj_x_center`/`proj_y_center`] is required. If such an event is found it is written to the output file marked as reaction `chan`. If several electron hits are found within in the time window specified up to `parameter[62]` electrons are written to the output file.

2.7.5 PIPICO projectile presorter

The following presorters were designed for typical ion/molecule collision experiments. There are different versions of PIPICO projectile presorter, but in general it writes data when certain conditions on ion/ion-coincidence and specified projectile positions of impact are meet. The number of parameters provided by the user defines what exact implementation of the presorter is going to be used (i.e. the pipico presorter command is overloaded).

```
new_presorter PROJ_PIPICO chan etof_center etof_width mass1_amu charge1_au
              mass2_amu charge2_au pipico_width
              px_center py_center p_radius
```

This presorter needs correctly set spectrometer parameters (see section 2.6. The expected times-of-flight for the two ion are obtained from the spectrometer settings (i.e. fields and lengths) and the masses and charges of the two ions. `mass1_amu` is the mass of the first ion in atomic mass units (e.g. nitrogen is 14), `charge1_au` is the first ion's charge in a.u. (e.g. singly charged ion has charge 1). The presorter checks for combinations of electron and ion hits that occur within:

$$e_{tof\ center} - e_{tof\ width} < e_{tof} < e_{tof\ center} + e_{tof\ width}$$
$$fabs(r2_{tof} - t2(r1_{tof})) < pipico_{width}$$

Here $t2$ is a function that calculates the expected time-of-flight of the second ion from the measured time-of-flight of the first ion and the spectrometer properties (using conservation of momentum). In case of nonequal ions both possible combinations of ion properties are checked. Furthermore a projectile hit with an impact within a radius of `proj_radius` around [`proj_x_center`/`proj_y_center`] is required. If several electron hits are found within the given time window up to `parameter[62]` electrons are written to the output file. Valid events are flagged as reaction `chan`.

```
new_presorter PROJ_PIPICO chan etof_center etof_width mass1_amu charge1_au
                mass2_amu charge2_au efield_Vcm acc1_mm pipico_width
                px_center py_center p_radius
```

PIPICO presorter employing a simple spectrometer with just a single acceleration region of length `acc1_mm` (given in mm) with an E-field of `efield_Vcm` (in V/cm).

The expected times-of-flight for the two ion are obtained from these spectrometer settings and the masses and charges of the two ions. `mass1_amu` is the mass of the first ion in atomic mass units (e.g. nitrogen is 14), `charge1_au` is the first ion's charge in a.u. (e.g. singly charged ion has charge 1). The presorter checks for combinations of electron and ion hits that occur within:

$$e_{tof\ center} - e_{tof\ width} < e_{tof} < e_{tof\ center} + e_{tof\ width}$$
$$fabs(r2_{tof} - t2(r1_{tof})) < pipico_{width}$$

Here $t2$ is a function that calculates the expected time-of-flight of the second ion from the measured time-of-flight of the first ion and the spectrometer properties (using conservation of momentum). In case of nonequal ions both possible combinations of ion properties are checked. Furthermore a projectile hit with an impact within a radius of `proj_radius` around [`proj_x_center`/`proj_y_center`] is required. If several electron hits are found within the given time window up to `parameter[62]` electrons are written to the output file. Valid events are flagged as reaction `chan`.

```
new_presorter PROJ_PIPICO chan etof_center etof_width mass1_amu charge1_au
                mass2_amu charge2_au efield1_Vcm acc1_mm
                efield2_Vcm acc2_mm pipico_width
                px_center py_center p_radius
```

Same PIPICO presorter as before but employing a spectrometer with two different acceleration regions of lengths `acc1_mm` and `acc2_mm` (given in mm) with E-fields of `efield1_Vcm` and `efield2_Vcm` (in V/cm). E-field values of 0 V/cm are recognized yielding a drift region.

The expected times-of-flight for the two ion are obtained from these spectrometer settings and the masses and charges of the two ions. `mass1_amu` is the mass of the first ion in atomic mass units (e.g. nitrogen is 14), `charge1_au` is the first ion's charge in a.u. (e.g. singly charged ion has charge 1). The presorter checks for combinations of electron and ion hits that occur within:

$$e_{tof\ center} - e_{tof\ width} < e_{tof} < e_{tof\ center} + e_{tof\ width}$$
$$fabs(r2_{tof} - t2(r1_{tof})) < pipico_{width}$$

Here $t2$ is a function that calculates the expected time-of-flight of the second ion from the measured time-of-flight of the first ion and the spectrometer properties (using conservation of momentum). In case of nonequal ions both possible combinations of ion properties are checked. Furthermore a projectile hit with an impact within a radius of `proj_radius` around [`proj_x_center`/`proj_y_center`] is required. If several electron hits are found within the given time window up to `parameter[62]` electrons are written to the output file. Valid events are flagged as reaction `chan`.

```
new_presorter PROJ_PIPICO chan etof_center etof_width mass1_amu charge1_au
               mass2_amu charge2_au efield1_Vcm acc1_mm
               efield2_Vcm acc2_mm efield3_Vcm acc3_mm pipico_width
               px_center py_center p_radius
```

Same PIPICO presorter as before but employing a spectrometer with three different acceleration regions of lengths `acc1_mm`, `acc2_mm` and `acc3_mm` (given in mm) with E-fields of `efield1_Vcm`, `efield2_Vcm` and `efield3_Vcm` (in V/cm). E-field values of 0 V/cm are recognized yielding a drift region.

The expected times-of-flight for the two ion are obtained from these spectrometer settings and the masses and charges of the two ions. `mass1_amu` is the mass of the first ion in atomic mass units (e.g. nitrogen is 14), `charge1_au` is the first ion's charge in a.u. (e.g. singly charged ion has charge 1). The presorter checks for combinations of electron and ion hits that occur within:

$$e_{tof\ center} - e_{tof\ width} < e_{tof} < e_{tof\ center} + e_{tof\ width}$$
$$fabs(r2_{tof} - t2(r1_{tof})) < pipico_{width}$$

Here $t2$ is a function that calculates the expected time-of-flight of the second ion from the measured time-of-flight of the first ion and the spectrometer properties (using conservation of momentum). In case of nonequal ions both possible combinations of ion properties are checked. Furthermore a projectile hit with an impact within a radius of `proj_radius` around [`proj_x_center`/`proj_y_center`] is required. If several electron hits are found within the given time window up to `parameter[62]` electrons are written to the output file. Valid events are flagged as reaction `chan`.

2.8 ColAHelL commands

By using ColAHelL a manifold of typical "lowlevel" analysis-tasks are performed automatically. As the user provides information on the reaction under investigation ColAHelL is able to calculate momenta from given raw data and create typical standard histograms.

```
use_ColAHelL
```

This command enable ColAHelL for your analysis. Furthermore this command is the ultimate reason to change the LMF2Root parser to case-insensitive...

In order for ColAHelL to work, the user needs to share details on the masses and charges of the involved ions. A "reaction" can be defined by introducing a reaction-block to the config file. A reaction block contains information on the reaction properties and allows the user to fine tune momenta by adding "set_par" statements. A rudimental set of conditions can be defined for low level background suppression etc.

A reaction block starts with:

```
reaction chan type name
```

As it is assumed that presorted data is used, the property `chan` is used to connect a reaction to the reaction channel found previously by the presorter. For example, if a presorter was used to store data the complies with a pipico-presorter for N_2 molecules tagging these events as reaction x setting `chan` to x will connect that data to the ColAHelL reaction block. The user need to specify the `type` of the reaction. Possible types are so far: `ELEC`, `ION`, `ION_ELEC`, `DIATOMIC` and `DIATOMIC_ELEC`. By giving a name, the spectra, that are generated automatically are stored in corresponding folders inside the .root-file.

A reaction block definition ends with:

```
end_reaction
```

In order to define the ion properties of a reaction the `mass_and_charge` command is used:

```
mass_and_charge ionnum mass_amu charge_au
```

This command sets ion number `ionnum` (first hit = 0 and so on) to be of mass `mass_amu` and charge `charge_au`. For a diatomic reaction both ions need to be defined.

A reaction block may contain a new set of `set_par` commands. As, for example, a jet offset needs to be addressed differently for different channels, these commands override the general `set_par` statements.

Rudimental background suppression is supported, as events can be invalidated if (or if not) certain user defined conditions are fulfilled.

```
invalidate_if TYPE VAR min max
invalidate_ifnot TYPE VAR min max
```

Different `TYPES` of conditions are available defining which property `VAR` to address:

Using the type `ELEC` the user can access the following the properties of the measured electrons:

- X: position of impact x-direction
- Y: position of impact y-direction
- TOF: time-of-flight
- PX: momentum x-direction
- PY: momentum y-direction
- PZ: momentum z-direction
- P: overall momentum
- PHIX: ϕ angle (in deg) in the lab frame with respect to x-axis (typically photon beam direction)
- PHIY: ϕ angle (in deg) in the lab frame with respect to y-axis (typically jet direction)
- PHIZ: ϕ angle (in deg) in the lab frame with respect to z-axis (typically ”time-of-flight” direction)
- PHIPOS: angle (in deg) on the detector
- THETAX: θ angle (in deg) in the lab frame with respect to x-axis
- THETAY: θ angle (in deg) in the lab frame with respect to y-axis
- THETAZ: θ angle (in deg) in the lab frame with respect to z-axis
- ENERGY: kinetic energy of the electron (in eV)

The `ION`-type gives access to the same properties of the measured ions:

- X: position of impact x-direction
- Y: position of impact y-direction
- TOF: time-of-flight
- PX: momentum x-direction
- PY: momentum y-direction
- PZ: momentum z-direction
- P: overall momentum
- PHIX: ϕ angle (in deg) in the lab frame with respect to x-axis (typically photon beam direction)
- PHIY: ϕ angle (in deg) in the lab frame with respect to y-axis (typically jet direction)
- PHIZ: ϕ angle (in deg) in the lab frame with respect to z-axis (typically ”time-of-flight” direction)
- PHIPOS: angle (in deg) on the detector
- THETAX: θ angle (in deg) in the lab frame with respect to x-axis
- THETAY: θ angle (in deg) in the lab frame with respect to y-axis
- THETAZ: θ angle (in deg) in the lab frame with respect to z-axis
- ENERGY: kinetic energy of the ion (in eV)

For the case of a `DIATOMIC` additional properties can be checked:

- PSUMX: sum momentum of the two ions x-direction
- PSUMY: sum momentum of the two ions y-direction
- PSUMZ: sum momentum of the two ions z-direction
- PSUM: sum momentum of the two ions
- PSUM_PHIX: ϕ angle (in deg) of the sum momentum of the two ions in the lab frame with respect to x-axis
- PSUM_PHIY: ϕ angle (in deg) of the sum momentum of the two ions in the lab frame with respect to y-axis
- PSUM_PHIZ: ϕ angle (in deg) of the sum momentum of the two ions in the lab frame with respect to z-axis
- PSUM_THETAX: θ angle (in deg) of the sum momentum of the two ions in the lab frame with respect to x-axis
- PSUM_THETAY: θ angle (in deg) of the sum momentum of the two ions in the lab frame with respect to y-axis
- PSUM_THETAZ: θ angle (in deg) of the sum momentum of the two ions in the lab frame with respect to z-axis
- PRELX: relative momentum of the two ions x-direction

- PRELY: relative momentum of the two ions y-direction
- PRELZ: relative momentum of the two ions z-direction
- PREL: relative momentum of the two ions
- PREL_PHIX: ϕ angle (in deg) of the realtive momentum of the two ions in the lab frame with respect to x-axis
- PREL_PHIY: ϕ angle (in deg) of the realtive momentum of the two ions in the lab frame with respect to y-axis
- PREL_PHIZ: ϕ angle (in deg) of the realtive momentum of the two ions in the lab frame with respect to z-axis
- PREL_THETAX: θ angle (in deg) of the realtive momentum of the two ions in the lab frame with respect to x-axis
- PREL_THETAY: θ angle (in deg) of the realtive momentum of the two ions in the lab frame with respect to y-axis
- PREL_THETAZ: θ angle (in deg) of the realtive momentum of the two ions in the lab frame with respect to z-axis
- KER: kinetic energy release of the molecular breakup (in eV)

The latest version of ColAHelL supports automatic generation of molecular frame electron angular distributions. This feature works - so far - only for diatomic molecules. Within the reaction definition the command

```
MFPAD_linear EE\_min EE\_max KER\_min KER\_max
```

will generate histograms for linearly polarized light. `EE_min` and `EE_max` can be used to restrict the plotting to certain electron energy ranges and, correspondingly `KER_min` and `KER_max` will restrict the kinetic energy release.

See section 3 for definitions of all molecular frame angles.

2.9 IPA commands

The interactive parameter adjustment tool (IPA) can be used fine tune data on the fly. As IPA is invoked a user-defined amount of events is stored in the PC RAM for fast access. These events can are processed in the IPA eventloop (located in `IPA.cpp`) ColAHelL and/or user code. After processing user defined histograms are plotted. The parameters that are used for processing (e.g. spectrometer parameters) can be modified online and the impact of these modifications on the user defined histograms can be viewed instantly. Please refer to the IPA-manual (work in progress...) for details. In order to use IPA the user must add the following statement to the config file:

```
invoke_ipa num_events channel
```

If done so, `num_events` events are copied into RAM prior to invoking the IPA user interface. If a `channel` is given, only events belonging to that reaction channel (i.e. have this reaction) are considered.

Apart from the predefined parameters IPA consists of up to 16 user defined parameters. These are set by using the command:

`add_ipa_parameter name val min max step`

A user parameter with a `name` is created. Its initial value is `val`. It changes in steps of `step` and is limited to `min` and `max`.

In several experiments the breakup of diatomic molecules is examined. For these cases a predefined bonus-IPA-tool is available! The IPA PIPICO-tool shows a pipico-histogram and adds lines of up to 4 simulated molecular breakup channels. Start it be adding the following line to the config file:

`ipa_pipico_tool mom_con bins tmin tmax`

All parameters are optional. If not set default values are used. The parameters are: `mom_con` - if defined as ”true” only events where the positions of impact of the two measured ions have a relative angle of approx. 180° are considered in the histogram. `bins`, `tmin` and `tmax` specify the properties of the PIPICO histogram.

2.10 File I/O

The final section of the config file includes the output file name and the list of input files to be processed. As of version 2 of LMF2Root the input files can be specified using wildcards. A ”*” replaces multiple characters, a ”?” a single character. For example all .lmf-files in the current directory are read by using `readLMF "*.lmf"`.

`set_LMF_output_file_name "NAME"`

Not very typical, but LMF2RRoot can write a .lmf-file that consists of the sorted TDC data. This command sets the corresponding filename. Check parameter 49 for enabling that feature.

`set_root_output_file_name "NAME"`

Defines the output filename. The suffix ”.root” is added to the filename automatically.

`readLMF "NAME"`

Invokes read in of the specified .lmf-file.

`readLMF`

If `readLMF` is used without a filename all .lmf-files that are found in the current directory are processed. After processing LMF2Root waits if further .lmf-files are added to the directory in order to process these, as well. ”lmf” needs to occur somewhere in the filename, it does not have to be its suffix. The current working directory may be specified by using:

`cd "dir"`

Sets the current working directory.

`readROOTfile "NAME"`

Invokes read in of the specified .root-file.

3 ColAHelL angles and coordinate frames

A ColAHelL vector is defined by (x,y,z). The standard ColAHelL coordinate frame of the momenta in photoionization experiments is:

- x-axis: photon propagation direction
- y-axis: jet direction (jet velocity is directed along the positive y-axis)
- z-axis: time-of-flight direction of the spectrometer

Make sure that your data complies with these definitions by rotating of mirroring your detectors accordingly. The polarization of directions of photons are defined correspondingly: linear, horizontal polarization is parallel to y, linear, vertical polarization is along z, the polarization plane of circularly polarized light is the y/z-plane.

The polar angle θ `.Theta()` is given with respect to the z-axis and ranges from 0 to π. 0 is along the z-axis. `.Theta_deg()` is the polar angle in degrees, ranging from 0 to 180, 0 is along z-axis. `.Cos_Theta()` returns $cos(\theta)$, ranging from -1 to 1 with 1 being along the z-axis. The azimuthal angle ϕ `.Phi()` is the angle around the z-axis, i.e. the angle in the x/y plane. It is defined relative to the x-axis and ranges from -p to p. 0 is along the x-axis. `.Phi_deg()` is the azimuthal angle in degrees, ranging from -180 to 180, 0 is along x-axis.

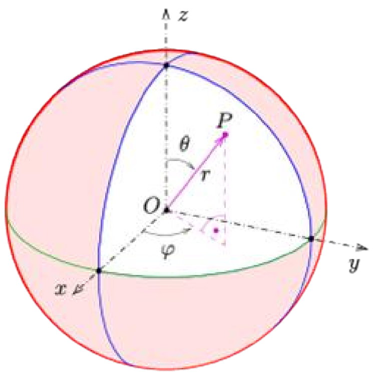

Figure 3: Spherical coordinates as used by ColAHelL. (Figure courtesy of Wikipedia, User:Ag2gaeh.

3.1 Molecular frame, diatomic molecules, circularly polarized light

The molecular frame is defined by the photon propagation direction and the moelcular axis.

For example:

```
CH_vector photon_dir = CH_vector(1.0,0,0);
```

```
   Coordinate_System mol_frame = Coordinate_System(photon_dir, mol→rel→mom);
```

In the molecular frame the molecule lies within the x/z-plane. The first mass (mass 0) given in the diatomic-definition is lying within positive x half plane. The electron momentum in the molecular frame is given by:

```
CH_vector e_MF = mf.project_vector(e[0] →phy→mom);
```

With this definition the mfpad3D()-macro can be directly used defining a histogram:

```
H2D:(e_MF.Phi_deg(), e_MF.Cos_Theta());
```

Using the standard ColAHelL coordinate frame, in order to align the molecule within the polarization plane, the following condition is needed:

```
if(fabs(mol→rel→mom.Angle_deg(photon_dir)-90) < 5)
```

(,,The direction of the first mass (red atom) is supposed to be within 5 in the laboratory y/z-plane (i.e. 90 with respect to laboratory x-axis).")

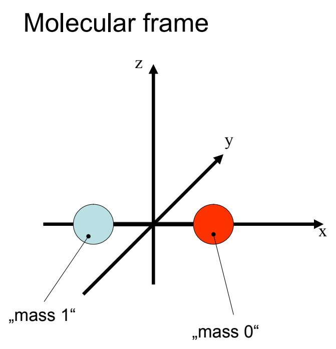

Figure 4: Molecular frame for the circular light case.

3.2 Molecular frame, diatomic molecules, linearly polarized light

The molecular frame is defined by the photon propagation direction and the moelcular axis.

For example:

```
CH_vector photon_dir = CH_vector(0,1.0,0);
   Coordinate_System mol_frame = Coordinate_System(mol→rel→mom,pol_dir);
```

In the molecular frame the molecule is along the z-axis. The electron momentum in the molecular frame is given by:

```
CH_vector e_MF = mf.project_vector(e[0] →phy→mom);
```

With this definition the mfpad3D()-macro can be directly used defining a histogram:

```
H2D:(e_MF.Phi_deg(), e_MF.Cos_Theta());
```

Using the standard ColAHelL coordinate frame, in order to align the molecule along the polarization axis, the following condition is needed:

```
if(mol→rel→mom.Angle_deg(pol_dir) < 10)
```

(,,The direction of the first mass (red atom) is supposed to be within a cone of 10 around the polarization direction (i.e. y-axis).")

In order to align the molecule perpendicularly to the polarization axis, the following condition is needed:

```
if(fabs(mol→rel→mom.Angle_deg(pol_dir)-90) < 10)
```

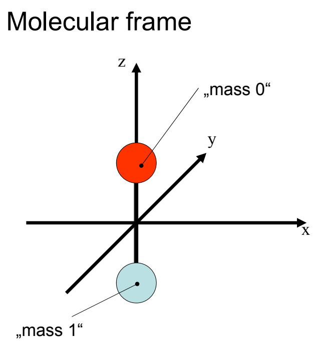

Figure 5: Molecular frame for the linear light case.

4 Example config files

Coming soon...

5 Detector calibration and resort

The new resort routine does many things fully automatic. All spectra needed for calibration are also generated automatically. Here is a is list of things that need to be done in order to calibrate a detector and correctly use the resort feature:

1. For all detectors: Turn off resort, turn off auto calibration.

2.a) Set all scale factors (called ”conversion factors” in the config.txt) to the same value (e.g. 1.0). You may use an appropriate value close to the ones provided in chapter 2.4.
2.b) Set position offset of w-layer (i.e. parameter $x16$) to 0.
2.c) Set time sum offsets (parameters $x13 - x15$) to 0.
2.d) Set maximum runtimes on the three layers (parameters $x23$ to $x25$) to 200.
2.e) Set position shift to 0 (parameters $x26$ and $x27$).

3. Run LMF2Root and look at the histograms of the raw data.
3.a) Located the integrated time sum value of each layer and their widths. Use these values to set parameters $x13 - x15$ and $x17 - x19$.
3.b) If the size of the your detector does not correspond to the real MCP size, multiply all three scale factors by a common factor to make the diameter fit.

4. Run LMF2Root again.
4.a) Check time sums (now at zero?) and detector diameter (now roughly size of real MCP?).
4.b) Find the center of the MCP (not the distribution on the MCP, this is important!) and set these values in the config file (parameters $x26$ and $x27$).

5. Rerun LMF2Root again.
5.a) Check that MCP is centered now.
5.b) Look at the histogram for $u_1 - u_2$, $v_1 - v_2$ and $w_1 - w_2$ (if present). Use a logarithmic y-axis to do so. In order to find the maximum runtime on the layers (parameters $x23$ to $x25$) the largest values of $fabs(u_1 - u_2)$, $fabs(v_1 - v_2)$ and $fabs(w_1 - w_2)$ must be spotted. Example: the u-layer has values of $(u_1 - u_2)$ from -127.3 to 110.8. In this case parameter $x23$ should be set to e.g 130. Find the corresponding values of the other layers and set parameters $x24$ and $x25$ accordingly.
5.c) The following steps need to be perform for each detector individually - only one detector at a time:
Switch on auto-calibration and run LMF2Root. As LMF2Root terminates it will provide new values for the scale factors of layers v and w and the w-offset. Copy these values to the config file and repeat step 5.c). Redo 5.c) a third time if you want to be super precise.

6) Turn off auto-calibration. Turn on resorting, sum correction and position correction. Pray. Done.