Manual of musrSimAna

Kamil Sedlák[1]

[1] *Laboratory for Muon Spin Spectroscopy, Paul Scherrer Institut, CH-5232 Villigen PSI, Switzerland*

"musrSimAna" is an analysis program, which helps the user to analyse and interpret the output of the "musrSim" simulation.

1 Introduction

The purpose of the *musrSimAna* program is to analyse the Root tree, the output of the *musrSim* program. In general, the event-by-event information stored in the Root tree can be easily used only for a very quick and rough tests – e.g. to see, where the muons stop and decay irrespective of whether they were triggered in the M-counter or not, or to have an idea what energy spectrum was deposited in a given counter. Typically, however, one is more interested to study μSR signal amplitude, time-independent background, or where the muons decay only for the "stopped" muons (e.g. muons triggered by M-counter and not detected by a veto counter), or only for triggered muons which actually stopped in a cryostat rather than in a sample. Such a study requires some kind of analysis program, which loops over all events stored in the Root tree (output of *musrSim*), and implements the logic of the required μSR experiment.

One way to do such analysis is to start from a Root command `MakeClass(''MyAnalysis'')`, which creates some sort of skeleton c++ class for the given Root tree. This allows the user to make his/her own and very specific analysis program tailored to his/her needs. On the other hand, it requires special program for each detector set-up.

A second possibility is to use *musrSimAna*, which is intended to be a general analysis program for μSR instruments. At the moment, however, *musrSimAna* is tuned to the needs of continuous muon beam facilities. Some modifications might be necessary for the pulsed muon beams.

As in the case of *musrSim*, the user of *musrSimAna* specifies all parameters required by the analysis in an input text file (steering file). The initial structure of the steering file was taken over from the TDC setup files used by real μSR instruments at PSI [1,2]. This setup file defines the "logic" of a given experiment, namely the coincidences and anti-coincidences of different counters and veto detectors, as well as some timing parameters. The setup file for the case of simulation had to be extended by the definitions of histograms, cuts, fitting, etc. The description of the setup file will be presented in chapter 2-4, the chapter 5 illustrates the whole concept of *musrSimAna* on the example of an existing μSR instrument.

2 The main components of the *musrSimAna* setup file

The parameters for the analysis are stored in the setup file "`RUNxxx.v1190`", whose name typically consists of the run number (`RUN`), some further identifier (`xxx`), and ends with "`.v1190`". One of the following commands can be used to execute *musrSimAna*:

```
$> musrSimAna RUN RUNxxx
$> musrSimAna RUN RUNxxx nographic > data/RUN.RUNxxx.out
```

where the first `RUN` stands for the run number (more precisely, it specifies the *musrSim* output, which is expected to be stored as "`data/musr_RUN.root`") and `RUNxxx` specifies the *musrSimAna* setup file without the ending, which is expected to be stored in the current directory. The output file of *musrSimAna*, containing the result histograms, will be stored as "`data/his_RUN_RUNxxx.v1190.root`".

The syntax of the setup file in *musrSimAna* is based on the experimental one, defined in [2]. At the beginning of this file, some timing parameters are defined:

```
RESOLUTION=100          ... one TDC bin corresponds to 100 ps

MCOINCIDENCEW=50        ... time interval (in TDC bins) to find coincidences with M-counter
PCOINCIDENCEW=50        ... time interval (in TDC bins) to find coincidences with P-counter
VCOINCIDENCEW=100       ... time interval (in TDC bins) to find anti-coincidences with M-counter

MUONRATEFACTOR=0.089    ... will be described in section 3

DATAWINDOWMIN=-2.0      ... data interval (in μs) in which positrons are detected
DATAWINDOWMAX=10.0

PILEUPWINDOWMIN=-10.0   ... the pileup interval (in μs) for muons
PILEUPWINDOWMAX=10.0
```

A good event has exactly one hit in the M-counter within the pile-up window, and exactly one hit in the positron counters within the data window. Both windows are defined relative to t_0, which is the time of the currently analysed M-counter hit.

The coincidence and anti-coincidence logic between the different counters (the detector logic) is also specified in the setup file. An example may look like this:

```
102; "M up";         M; 0.4;  2005; -401;
  1; "B1";           P; 0.4;  2005;   21 -401; B1; 1485; 1515; 50995;
  2; "B2";           P; 0.4;  2005;   22 -401; B2; 1485; 1515; 50995;
 11; "F1";           P; 0.4;  2005; -401 -21 -22; F11; 1485; 1515; 50995;
 12; "F2";           P; 0.4;  2005; -401 -21 -22; F12; 1485; 1515; 50995;
 13; "F3";           P; 0.4;  2005; -401 -21 -22; F13; 1485; 1515; 50995;
 21; "Coinc B1"      K; 0.4;  2005;
 22; "Coinc B2"      K; 0.4;  2005;
401; "Active Veto"   V; 0.1;  2005;
```

In each row, the first number stands for the detector ID (defined in the steering file of the *musrSim*), the second variable is the name of the counter, the third variable identifies the type of the counter (M = muon, P = positron, K = coincidence and V = veto counters, respectively), the forth variable is the threshold in MeV applied to the energy deposited in the counter, the fifth variable is a time delay (in units of bin-width) applied to the given counter, the sixth set of parameters defines which other detectors have to be in coincidence (positive ID) or anti-coincidence (negative ID) with the given counter, and in the case of positron counters, the last four parameters define a special histogram for the given counter. (Note that this histogram is used for a compatibility with the experimental steering file, however there is a different and more powerful way of how to define histograms, which is defined later). The only difference with respect to the experimental steering file is the presence of threshold definition in the fourth column. In the example above, the M-counter is the volume which has ID 102 in the simulation, and a valid hit has to have energy in the counter above 0.4 MeV, and no signal above 0.1 MeV in the "Active Veto" (ID=401) within a time interval defined by `VCOINCIDENCEW`. The detailed description of *.v1190 steering files can be found in chapter 4 and in [2].

The main output of the simulation is generated in the form of histograms. The histograms are defined in the steering file. For example, the following line defines 1-dimensional histogram of the z-position of where the muons stop and decay:

```
musrTH1D hMuDecayPosZ "Where the muons stop;z[mm];N" 100 -5.  5.  muDecayPosZ
```

This histogram has 100 bins, spans from -5 mm to 5 mm, and the variable filled in the histogram is `muDecayPosZ`. In fact, this line does not create one histogram, but an array of histograms – one histogram for each "condition". An example of five conditions reads:

```
condition 1 oncePerEvent
condition 2 muonDecayedInSample_gen
condition 4 muonTriggered_det
condition 6 goodEvent_det
condition 9 pileupEvent
```

where integer numbers (1,2,4,6 and 9) denote the condition number, and the string at the end describes the condition – e.g. "`muonDecayedInSample_gen`" specifies that muon has stopped and decayed in the sample, and "`muonTriggered_det`" specifies that there had to be a good muon candidate triggered in the M-counter. The ending "`_gen`" indicates that the variable used in the condition was "generated", i.e. it is known in the simulation, but can not be directly measured experimentally. On the other hand the ending "`_det`" specifies that the given condition is based on measurable variables, as e.g. energy deposits in a counter.

There can be up to 30 conditions requested, and for each of them a separate histogram will be filled. In the output file of *musrSimAna*, the histograms corresponding to the previous set of conditions would be saved as `hMuDecayPosZ_1, hMuDecayPosZ_2, hMuDecayPosZ_4, hMuDecayPosZ_6, hMuDecayPosZ_9`. The `hMuDecayPosZ_1` shows where the muons stop irrespective whether they were detected or not. The `hMuDecayPosZ_6` contains "good" muons, i.e. muons that would be saved in the final histograms in the experiment. The `hMuDecayPosZ_9` shows the muons, which contribute to the time-independent background. Note that there are always two muons, μ_1 and μ_2, contributing to the time-independent background, and `hMuDecayPosZ_9` shows the one (μ_1) that started the M-counter. More to this topic is presented in chapter 3. Here we just mention that in order to see the z-coordinate of the second muon (μ_2), which was not triggered by the M-counter but whose decay positron hit the positron counter, the histogram `hpileup_muDecayPosZ_9` must be used:

```
musrTH1D hpileup_muDecayPosZ "Pileup muons;z[mm];N" 100 -5.0 5.  pileup_muDecayPosZ
```

3 Event mixing – an unavoidable complication

The output of *musrSim* is stored in a Root tree named "`t1`". One event corresponds to one simulated muon, and it is saved as one row of the tree. The only information that relates one event (muon) to any other one is the variable `timeToNextEvent`, which is a randomly generated time difference between two subsequent events.

The *musrSimAna* allows one to study the "time-independent background", which is due to mixing of two different events. A simple example of the event mixing affecting the μSR measurement is the following: the first muon, μ_1, hits the M-counter, and subsequently stops and decays in the sample, however the decay positron escapes undetected – most likely because of the limited angular acceptance of positron counters. The second muon, μ_2, arrives around the same time, misses the M-counter, and stops and decays in a collimator wall or elsewhere. Its decay positron hits a positron counter. Thus there are good-looking muon and positron hits, which however arise from two uncorrelated muons. This fake (background) event is experimentally treated as a good event, and contributes to the time-independent background.

Events can be mixed in *musrSimAna*, allowing us to study sources of the time-independent background. Before analysing a given event, arrays of hits are filled for all counters (M, positron, veto, coincidence counters), which store the hits occurring in the future up to the time equal to $3\cdot$ pileup window or $3\cdot$ data window, whatever is larger[a]. There is one such array for every counter. After this initial filling, there might be several hits in every array, originating from one or more events.

The fraction of the time-independent background to good events depends on the incoming muon rate measured by the trigger, possibly in the anti-coincidence with a backward veto detector, if used in the experiment. Typically, the experimentalists set the incoming muon rate (rate of *stopped muons*) to $\sim 30\,000\,\mu/s$. The same should be done in the simulation. However, the rate of stopped muons is known only after the analysis is done, because, for example, many simulated muons stop and decay in collimators or elsewhere in the beam-pipe without producing any signal in the M-counter. Therefore the simulation is normally started with an initial rate of generated muons of $30\,000\,\mu/s$, which in practise can correspond to much lower rate of *stopped muons*. The rate of stopped muons is calculated at the end of the *musrSimAna* execution, and it is printed out in the *musrSimAna* output. The user can use this information and rescale the initial muon rate by changing parameter `MUONRATEFACTOR` in the `*.v1190` setup file. This is done without the necessity to re-run the CPU consuming simulation – only the *musrSimAna* has to be repeated. The complete simulation and analysis chain therefore usually consists of three steps:

1. *musrSim* (takes typically 10 hours).
2. *musrSimAna* with `MUONRATEFACTOR=1.0` (takes typically 10 minutes).

[a] The data and pile-up windows defined by parameters `DATAWINDOWMIN, DATAWINDOWMAX, PILEUPWINDOWMIN` and `PILEUPWINDOWMAX` are applied later on in the analysis.

3. *musrSimAna* with `MUONRATEFACTOR` determined in step 2.

The `MUONRATEFACTOR` specifies a multiplicative factor applied to the variable `timeToNextEvent` (the randomly generated time difference between two subsequent events).

IMPORTANT NOTE: In order to get the pile-up effects correctly analysed by *musrSimAna*, it is probably necessary to run the *musrSim* with no event reweighting (i.e. the command "`/musr/command logicalVolumeToBeReweighted ...`" must **not** be used). All events should/have to be (?) saved in the Root tree (i.e. the command "`/musr/command storeOnlyEventsWithHits false`" must be used).

4 Detailed list of steering file parameters

RESOLUTION=*value*
width of the TDC bin in picoseconds.

MDELAY=*value*
currently not used (probably not needed in the case of simulation).

PDELAY=*value*
currently not used (probably not needed in the case of simulation).

MCOINCIDENCEW=*value*
time window for the coincidences of the coincidence detectors ("K") with the M-counter. The *value* is given in TDC bins (see `RESOLUTION` above).

PCOINCIDENCEW=*value*
time window for the coincidences of the coincidence detectors ("K") with positron counters. The *value* is given in TDC bins.

VCOINCIDENCEW=*value*
time window for the coincidences of the anti-coincidence detectors ("V") with any other counter. The *value* is given in TDC bins.

MUONRATEFACTOR=*value*
a multiplicative factor which is used to rescale time differences between subsequent muons. Setting *value* larger than 1 artificially prolongs the time difference between two subsequently generated muons, and therefore decreases the incoming muon rate (number of muons per second). This parameter should be changed in order to set the incoming muon rate to a given required value, typically to 30 000 μ/s.
See also variable "INFINITELYLOWMUONRATE".

INFINITELYLOWMUONRATE
If INFINITELYLOWMUONRATE is specified, each event is treated independently of any other event. This corresponds to a situation of infinitely low rate of incoming muons, and no pileup can be observed. The variable "MUONRATEFACTOR" becomes irrelevant when INFINITELYLOWMUONRATE is specified.

DATAWINDOWMIN=*value*
Beginning of the data interval for the positron counters in μs.

DATAWINDOWMAX=*value*
End of the data interval for the positron counters in μs.

PILEUPWINDOWMIN=*value*
Beginning of the pileup interval for the M-counter in μs.

PILEUPWINDOWMAX=*value*
End of the pileup interval for the M-counter in μs.

PROMPTPEAKMIN=*value*
Beginning of the prompt-peak interval in μs. This variable is used only for the condition "`promptPeak`", "`promptPeakF`", etc. , and normally does not need to be specified. It becomes useful if the user wants to investigate, where the prompt-peak originates from (where do the muons, which give rise to the prompt peak, stop). The default value is -0.01 μs.

PROMPTPEAKMAX=*value*
End of the prompt-peak interval in μs, the default value is $0.01\,\mu$s. (See comments for `PROMPTPEAKMIN`.)

MUSRMODE=*string*
Defines the mode of μSR experiment – presently only "D", corresponding to the time differential mode is implemented.

REWINDTIMEBINS=*value*
A technical parameter specifying when a roll-over of all hits has to be done. It is specified in TDC bins, and normally there should be no need to change this parameter.

DEBUGEVENT *eventID debugLevel*
Prints out debug information about the event with the ID *"eventID"*. The higher the *debugLevel*, the more details are printed. (Both *eventID* and *debugLevel* are integers).

CLONECHANNEL *ID1 ID2*
Clones the hits detected in counter *ID1* into a new counter *ID2*. A different (even a lower) threshold can be applied to the cloned counter. This way the same counter can be used as two independent counters – e.g. once as a veto detector for the M-counter, and simultaneously as a coincidence detector for a P-counter. In both cases the energy threshold and time windows are defined independently.

musrTH1D *histoName histoTitle nBins min max variable* [`rotreference` ν_{RRF} ϕ_{RRF}]
Defines a histogram (or more precisely an array of histograms, where the number of histograms in the array is given by the number of conditions, see section 2). The name of the histogram is defined by *histoName* + the number of the condition. The string variable *histoTitle* specifies the title of the histogram, *nBins*, *min* and *max* stand for the number of bins and minimum and maximum of the x-axis of the histogram.
The optional keyword "`rotreference`" signals that the given histogram will be filled in rotating reference frame (RRF) with the frequency of ν_{RRF} and a phase shift of ϕ_{RRF}.
The *variable* stands for the variable that will be filled into the histogram. The *variable* can be any variable from the output Root tree of musrSim (see "Manual of musrSim") (except for the array variables like `det_*[], save*[], odet_*[]`). In addition, it can be also one of the following:

muDecayPosR ... $\sqrt{\mathrm{muDecayPosX}^2 + \mathrm{muDecayPosY}^2}$.

wght ... weight of the event.

det_m0edep ... energy deposited in the M-counter that gives the muon signal.

det_posEdep ... energy deposited in the P-counter that gives the positron signal.

pos_Momentum ... magnitude of the momentum of the decay positron ("generated", not measurable variable).

pos_Trans_Momentum ... transverse momentum of the decay positron.

pos_Radius ... positron radius calculated from the decay positron momentum and magnetic field at the point of decay.

pos_Theta ... polar angle of the decay positron.

pos_Phi ... azimuth angle of the decay positron.

det_time10 ... time difference between the positron and muon counters (measured by the respective counters).

gen_time10 ... the time difference between the muon decay and the first muon hit in the M-counter (i.e. `muDecayTime - muM0Time`).

det_time10_MINUS_gen_time10 ... `det_time10 - gen_time10` in picoseconds.

det_time20 ... very similar to `det_time10`, however taking into account "counter phase shift" defined by `counterPhaseShifts` variable. This gives the user a possibility to sum up backward and forward histograms into one histogram (this of course make sense only in the simulation, where there is "no physics" happening in the sample, just the muon spin rotation).

pileup_eventID ... eventID of the μ_2, where μ_2 stands for the muon, which did not give signal in the M-counter, but whose decay positron gave signal in the positron counter around the time when a different muon (μ_1) triggered the M-counter.

pileup_muDecayDetID ... detector ID, in which μ_2 stopped and decayed.

pileup_muDecayPosZ ... z-coordinate of where the μ_2 stopped and decayed.

pileup_muDecayPosR ... radius of where the μ_2 stopped and decayed.

Variables are usually set to -1000 if they can not be calculated (e.g. `det_posEdep` = -1000 if there was no hit in any positron counter).

musrTH2D ***histoName histoTitle nBins min max nBins2 min2 max2 variable***
Similar to *musrTH1D*, but for a 2-D histogram.

humanDecayHistograms ***hist_decay_detID hist_decay_detID_pileup*** id_1 $name_1$ ... id_n $name_n$
This is a special kind of histogram, which converts two histograms (*hist_decay_detID hist_decay_detID_pileup*) into a human-friendly histograms, where detector ID on the x-axis is converted into a string label. The id_i is the detector id, and the $name_i$ is the corresponding label name. If $name_i = name_j$, the corresponding bins of the original histograms will be summed up together into the same bin. The *hist_decay_detID* and *hist_decay_detID_pileup* have to be defined before (by the command `musrTH1D`).

condition ***conditionID conditionName***
Definition of a condition, which is then used when filling histograms. The *conditionID* specifies the number of the condition, and must be between 0 and 30 (0 and 30 are also possible). The *conditionName* is one of the following:

alwaysTrue ... true for every hit in the m-counter (there can be more than one M-counter hit per event).

oncePerEvent ... true once for every event (the first hit in M-counter, if any, is considered).

muonDecayedInSample_gen ... true if muon stopped and decayed in the sample.

muonTriggered_gen ... true if muon passed through the M-counter (irrespective of the deposited energy) – not a measurable variable.

muonTriggered_det ... true if a good muon candidate was found in the M-counter (using coincidences, vetoes, ...). Double hits within the pileup window are excluded.

positronHit_det ... true if a good positron candidate was found in the positron counter. Double hits within the data window are excluded.

goodEvent_det ... true if `muonTriggered_det` and `positronHit_det`.

goodEvent_gen ... true if muon passed through the M-counter, and the muon stopped anywhere (i.e. did not leave the World volume of the simulation). No requirement on the positron is implied, i.e. the positron may or may not be detected. Not a measurable variable.

goodEvent_det_AND_goodEvent_gen

pileupEventCandidate ... M-counter hit and positron counter hit both come from two different events.

pileupEvent ... `pileupEventCandidate` and `goodEvent_det`.

goodEvent_det_AND_muonDecayedInSample_gen

goodEvent_F_det ... `goodEvent_det`, where the positron was detected in the forward detectors defined by the command `counterGrouping`.

goodEvent_B_det ... like `goodEvent_F_det` but for backward positron counters.

goodEvent_U_det ... like `goodEvent_F_det` but for upper positron counters.

goodEvent_D_det ... like `goodEvent_F_det` but for lower positron counters.

goodEvent_L_det ... like `goodEvent_F_det` but for left positron counters.

goodEvent_R_det ... like `goodEvent_F_det` but for right positron counters.

goodEvent_F_det_AND_pileupEvent ... `goodEvent_F_det` and `pileupEvent`.

goodEvent_B_det_AND_pileupEvent ... `goodEvent_B_det` and `pileupEvent`.

goodEvent_U_det_AND_pileupEvent ... `goodEvent_U_det` and `pileupEvent`.

goodEvent_D_det_AND_pileupEvent ... `goodEvent_D_det` and `pileupEvent`.

goodEvent_L_det_AND_pileupEvent ... `goodEvent_L_det` and `pileupEvent`.

goodEvent_R_det_AND_pileupEvent ... `goodEvent_R_det` and `pileupEvent`.

promptPeak ... `goodEvent_det`, and `PROMPTPEAKMIN < det_time10 < PROMPTPEAKMAX`.

promptPeakF ... like `goodEvent_F_det` and `promptPeak`.

promptPeakB ... like `goodEvent_B_det` and `promptPeak`.

promptPeakU ... like `goodEvent_U_det` and `promptPeak`.

promptPeakD ... like `goodEvent_D_det` and `promptPeak`.

promptPeakL ... like `goodEvent_L_det` and `promptPeak`.

promptPeakR ... like `goodEvent_R_det` and `promptPeak`.

Additional conditions may be implemented on request.

draw ***histogramName conditionID***
Plot histogram (for a given condition) at the end of the analysis. Note that all histograms are saved into the output file irrespective whether they are plotted or not.

counterPhaseShifts $ID_1\ \phi_1\ ID_2\ \phi_2\ \ldots\ ID_n\ \phi_n$
Defines relative phase shifts of signal between different positron counters, which is used for calculation variable `det_time20`. ID_i is the ID number of the positron counter, ϕ_i is its phase shift. This gives the user a possibility to sum up backward and forward histograms into one histogram.

counterGrouping ***group*** $ID_1\ ID_2\ \ldots\ ID_n$
This defines a group of detectors, where *group* stands for "B" (backward), "F" (forward), "U" (up), "D" (down), "L" (left) and "R" (right) detectors. This grouping is used in the definition of some conditions.

setSpecialAnticoincidenceTimeWindow ***detectorID timeMin timeMax unit***
This command sets a special anti-coincidence time window for a detector *detectorID*. Normally, the anti-coincidence time window is defined by `VCOINCIDENCEW`, and is the same for all anti-coincidence detectors. However, sometimes it might be interesting to set the anti-coincidence time window differently for a specific detector (e.g. one might test an anti-coincidence of a veto detector with the M-counter for the whole pile-up time window of $\sim 10\,\mu s$. Unlike in the case of `VCOINCIDENCEW`, here the *units* are not TDC bins, but rather time in "nanosecond" or "microsecond".

fit ***histogramName function min max*** $p_1\ \ldots\ p_n$
Fits the histogram by a given function, where *min*, *max* define the range of the fit on the x-axis of the histogram, and $p_1\ \ldots\ p_n$ are (typically, with some exceptions) the initial values of the function parameters. The following functions are currently predefined:

pol0 $= p_0$ ... a constant (1 parameter) - typically used to fit background.

simpleExpoPLUSconst $= p_0 \exp(-x/2.19703) + p_1$

rotFrameTime20 $= p_2 \cos(p_0 x + p_1)$

funct1 $= p_3 \exp((p_4 - x)/2.19703) \cdot (1 + p_2 \cos(p_0 x + p_1))$

funct2 $= p_3 \exp((p_4 - x)/2.19703) \cdot (1 + p_2 \cos(p_0 x + p_1)) + p_5$

funct4 $= p_3 \exp((-x)/2.19703) \cdot (1 + p_2 \cos(p_0 x + p_1)) + p_4$

5 A real-life example: GPD instrument

The simulation of the General Purpose Decay-Channel Spectrometer (GPD) instrument [3] installed at PSI has been exemplified in the *musrSim* manual [4]. Here we analyse the output of this simulation using *musrSimAna*. The run number of this simulation is 201, therefore the steering file names are "`201.mac`" for *musrSim*, and "`201.v1190`" for *musrSimAna*, respectively, and the output file name of *musrSim* is saved as "`data/musr_201.root`". The detector system is very simple with only six counters – M-counter, two backward positron counters and three forward positron counters. The reader is strongly recommended to see the illustration of the GPD geometry in the *musrSim* manual [4].

8 000 000 of events were simulated (i.e. 8 000 000 of muons were generated 100 cm in front of the GPD sample). In only 949 759 events (11.9% out of the 8 million) there was a signal detected in one or more

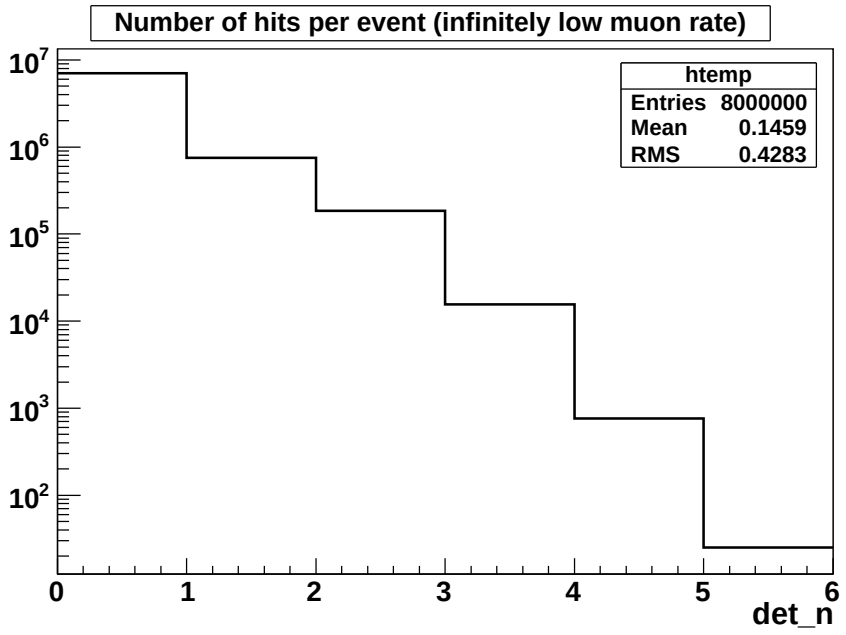

Figure 1. Number of hits in all counters per event, assuming infinitely low incoming muon rate. The same detector may be hit more than once (e.g. if both the muon and its decay positron pass through the M-counter).

counters. The remaining muons stopped somewhere (most often in collimator, as we will see later), decayed, and the decay positron (and any other particles created thereof) missed the counters. This is illustrated in more details in Fig. 1, where number of detector hit per event, assuming infinitely low incoming muon rate, is shown. This plot was created in Root by executing:

```
root [0] TFile* ff=new TFile("data/musr_201.root")
root [1] t1->Print()
root [2] t1->Print("det_n","det_n>0")
```

It has to be pointed out, that the ratio of muons passing through the opening in collimators to the number of all generated muons strongly depends on the beam properties – beam profile, beam convergence, etc. Typically, if we have too broad muon beam, we simulate many "useless" events. However, the other extreme (simulating too narrow beam) can lead to underestimating the time-independent background.

It took approximately 12 hours of the CPU (on PC bought in 2010, where 1 out of 4 processor cores was running) to simulate these 8 000 000 events. Assuming the 30 000 μ/s trigger rate, this corresponds to 26 seconds of real experimental running.

5.1 Where the muons stop and decay

The positions, or more precisely the components of the GPD instrument, where the muons stop and decay, are shown in Fig. 2:

- Figure 2 was generated by Root macro file "Plot201.C".
- The labels on the x-axis are defined in the file `201.v1190` by the command `humanDecayHistograms ...`
- The dashed-line histogram in Fig. 2 shows where the muons stopped and decayed if no preselection criteria are applied on the muons, i.e. if all generated muons are considered. This is histogram "`humanDecayHistograms_1`".
- The full-line histograms show where stopped the muons, for which either the muon itself or its secondary particle (e^+, γ) triggered the M-counter: black histogram stands for all such muons, corresponding to infinitely low incoming muon rate, while the red histogram represents the case for the

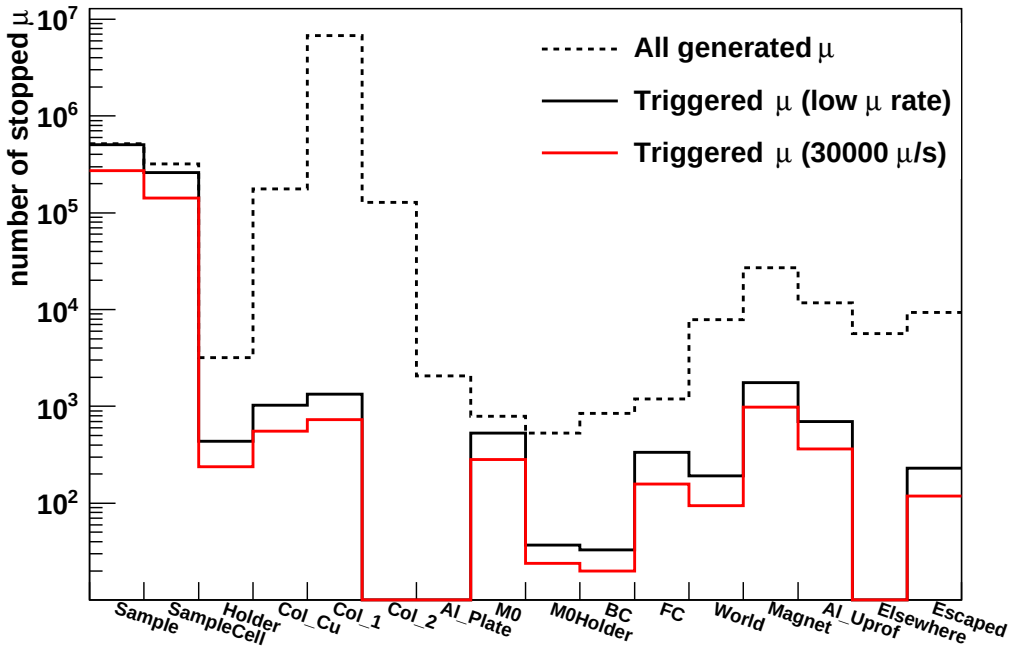

Figure 2. This plot indicates, where the muons stopped and decayed. The dashed histogram shows all generated muons. The full-line histograms show where stopped the muons, for which either the muon itself or its secondary particle (e^+, γ) triggered the M-counter: black histogram stands for all such muons, corresponding to infinitely low incoming muon rate, while the red histogram stands for the incoming muon rate of 30 000 μ/s. 8 000 000 of events were simulated.

30 000 μ/s incoming muon rate. An energy deposit of at least 0.4 MeV in the M-counter is required to fire the trigger. The number of triggered events decreases with the incoming muon rate, because some of the events are rejected due to the 10 μs pileup gate.

The histogram name is in both cases "`humanDecayHistograms_4`", where the black histogram was calculated using the setup file "`201a.v1190`" with the keyword `INFINITELYLOWMUONRATE`, while the red histogram was calculated using the setup file "`201.v1190`" with `MUONRATEFACTOR=0.0965819`.

- The ± 10 μs pile-up gate at the incoming muon rate of 30 000 μ/s rejects approx. 45% of the triggered events. This number can be calulated in Root as the ratio of the "`Integral`" of the red and black histograms in Fig. 2:

```
root [0] TFile* file1 = new TFile("data/his_201_201a.v1190.root")
root [1] humanDecayHistograms_4->Integral()
root [0] TFile* file2 = new TFile("data/his_201_201.v1190.root")
root [1] humanDecayHistograms_4->Integral()
```

- The muon sample fraction (ratio of muons stopped in the sample over all muons that fired the trigger) for the triggered events (full-line histograms) is 65%, and it is practically the same for both infinitely low and 30 000 μ/s incoming rate. This number can be obtained in Root by dividing the first column of histogram `humanDecayHistograms_4` by the sum of all entries in this histogram:

```
root [0] TFile* file = new TFile("data/his_201_201.v1190.root")
root [1] (humanDecayHistograms_4->GetBinContent(1))/(humanDecayHistograms_4->Integral())
```

- The largest fraction of generated muons (dashed-line histogram) stopped in collimators. Only a small fraction of them caused a hit in the M-counter (full-line histograms).

- Despite the high initial muon momentum of 100 ± 3 GeV/c, muons are significantly scattered in the

last 50 cm region of air. This can be clearly seen if the magnetic field is off and a point-like muon beam is used (which can be done by modifying the `201.mac` file) – only 77% of the muons stop in the sample cell or in the sample, while the remaining 23% of the mouns are scattered so much in the air, that they end up in collimators or elsewhere (not shown here).

- "World" in the histogram label means that the muon decayed in the beampipe vacuum or somewhere else in the air (on the fly).

- "Escaped" means that the muon left the simulated instrument (more precisely the "world" volume) prior its decay.

Figure 3 shows the "pile-up events". These are events, in which one muon (μ_1) is triggered by the

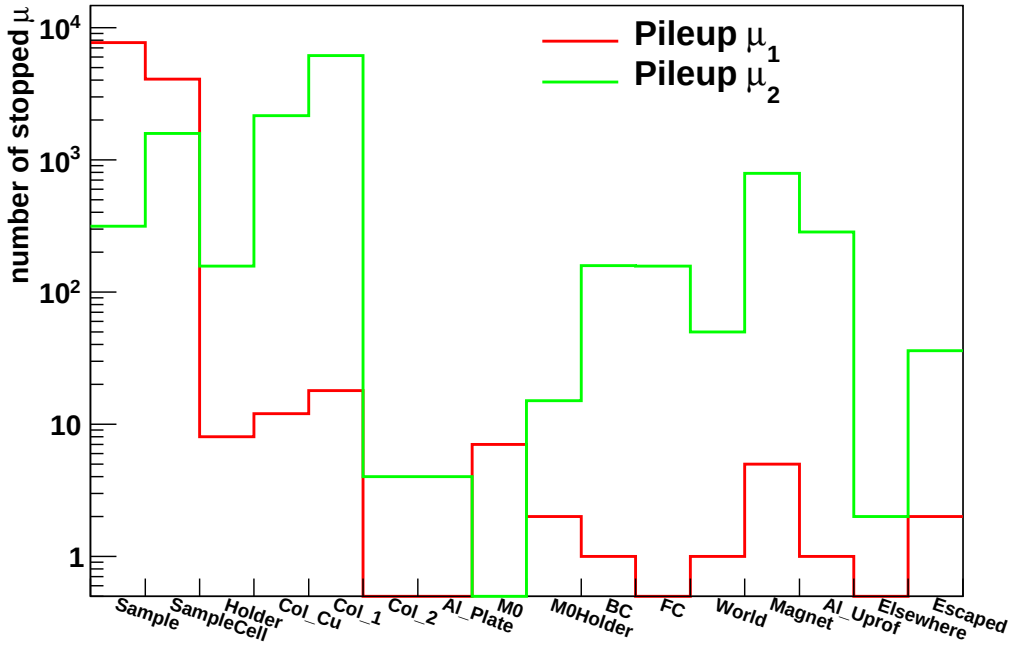

Figure 3. Pile-up events, i.e. the events in which one muon (μ_1) fired the trigger, while the hit in a positron counter is due to a decay positron from a different muon (μ_2). Pile-up events look like a good events, and contribute to the time-independent background.

m-counter, while a positron from a different muon (μ_2) was detected by a positron counter[b]. In addition to this requirement, the decay positron of μ_1 must escape undetected (e.g. it must miss positron counters) and μ_2 must not trigger the m-counter – otherwise the event would be rejected. Pile-up events are the source of the time independent background. Usually μ_1 is a good-looking muon that stops in the sample or in the sample cell (red histogram in Fig. 3), while μ_2 stops and decays at different places, mainly in the collimators (green histogram in Fig. 3).

A nice visualisation of where the background-contributing muons μ_2 stop and decay is presented in Fig. 4 (histogram "`hMuDecayMappileup_9`"). In this two dimensional histogram, different components of the GPD instrument, like the lead collimator, the copper collimator and the sample cell, can be recognised. The lead collimator is located at the z-position between -115 mm and -85 mm. Due to the high initial muon momentum of ~ 100 MeV/c, the maximum of muons in Fig. 4 stop quite deep in the lead collimator, at around $z = -103$ mm. This might be a little bit surprising to the μSR scientists who are used to work with the surface muons with momentum of 28 MeV/c.

[b]In fact, the trigger may also be triggered by the decay positron of μ_1 and/or a positron counter may detect directly μ_2, not its decay positron. Such cases are rare, but they are implicitly included in the simulation.

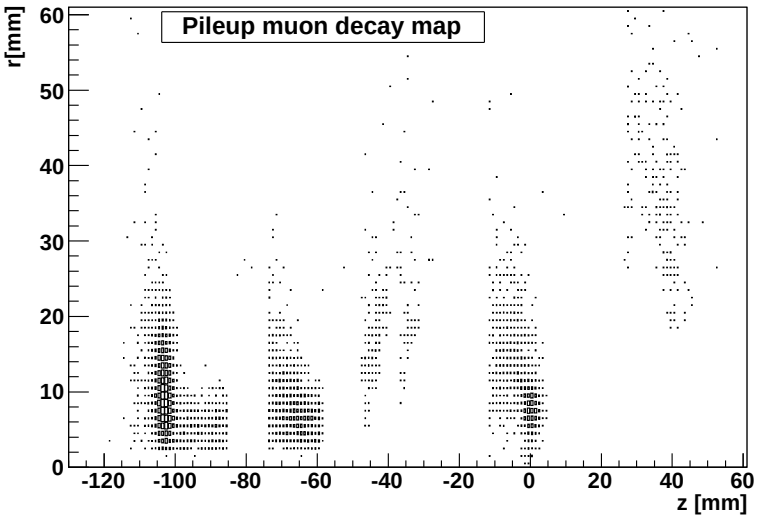

Figure 4. Positions of where the μ_2 stop and decay.

5.2 The μSR signal

Figure 5 shows the μSR spectra for the same run, i.e. for the transverse field of 300 gauss, integrated over

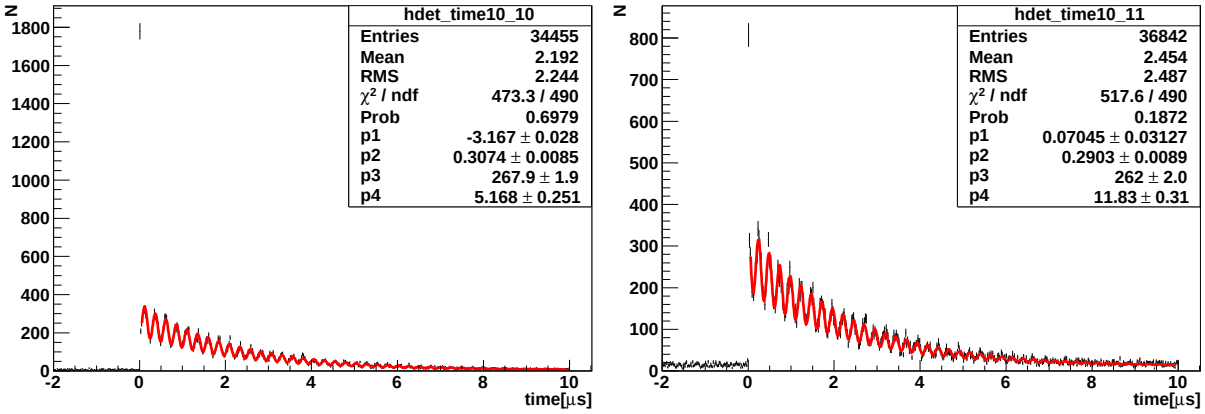

Figure 5. MuSR signal for the run 201 (TF= 300 gauss). The tree forward positron counters are summed up in the left histogram, and the two backward counters in the right histogram.

the three forward positron counters (left histogram called `hdet_time10_10`) and over the two backward positron counters (right histogram called `hdet_time10_11`). Zero on the time axis corresponds to t_0, i.e. time of the m-counter hit. One can see a prompt peak at t_0, time independent background at negative times and an oscillating signal at positive times. The following function has been fitted to the oscillating part of the signal:

$$f = p_3 \cdot e^{-t/2.19703} \cdot (1 + p_2 \cdot \cos(t \cdot p_0 + p_1)) + p_4 \tag{1}$$

The fits were restricted to the time interval of $(t_0 + 0.05\mu s, t_0 + 9.95\mu s)$, and the parameter p_0 was fixed (e.g. not fitted). The fitted amplitude of asymmetry are $p_2 = 0.307 \pm 0.009$ and $p_2 = 0.290 \pm 0.009$ for the forward and backward counters respectively.

Parts of the spectra from Fig. 5 are shown in detail in Fig. 6. The left plot in Fig. 6 shows the signal in the forward counters around t_0, the right plot shows the (time-independent background) signal at negative times in the backward counters.

An important characteristic of a μSR instrument is the time-independent background. It is usually

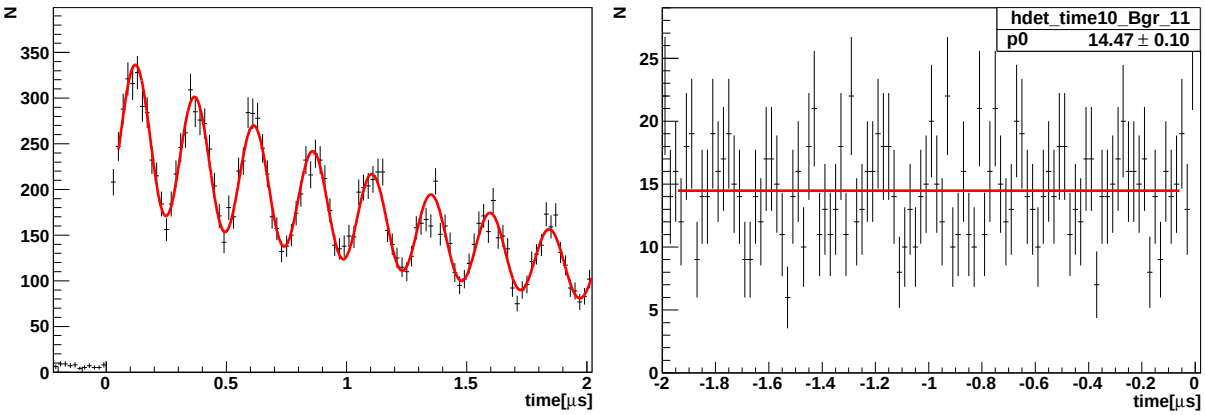

Figure 6. MuSR signal for the run 201 (TF= 300 gauss) – details of Fig. 5. The left plot shows the signal in the forward counters around t_0, the right plot shows the (time-independent background) signal at negative times in the backward counters.

expressed as

$$\mathrm{Bgr} = p_{-}/p_3 \quad , \tag{2}$$

where p_{-} is the fit to the time-independent background, i.e. signal at negative times, and p_3 is the parameter from eq.(1), which specifies what the size of the signal would be at t_0 in the absence of oscillations. In the case of backward counters $\mathrm{Bgr}_{\mathrm{backw}} = 14.47/262 = 5.5\,\%$, in the case of forward counters $\mathrm{Bgr}_{\mathrm{forw}} = 6.88/267.9 = 2.6\,\%$.

Note that the histogram on right hand side of Fig. 6 is labelled "`hdet_time10_Bgr_11`", not "`hdet_time10_11`". In fact, the two histograms are identical, as one can see in the setup file `201.v1190`. The only difference is in the fitting – the same data stored in both histograms are fitted by different functions in different time ranges.

5.3 The μSR signal from individual counters

Figure 7 shows the observed signal in the forward counter No. 11 (FW11). Originally, the histogram F11

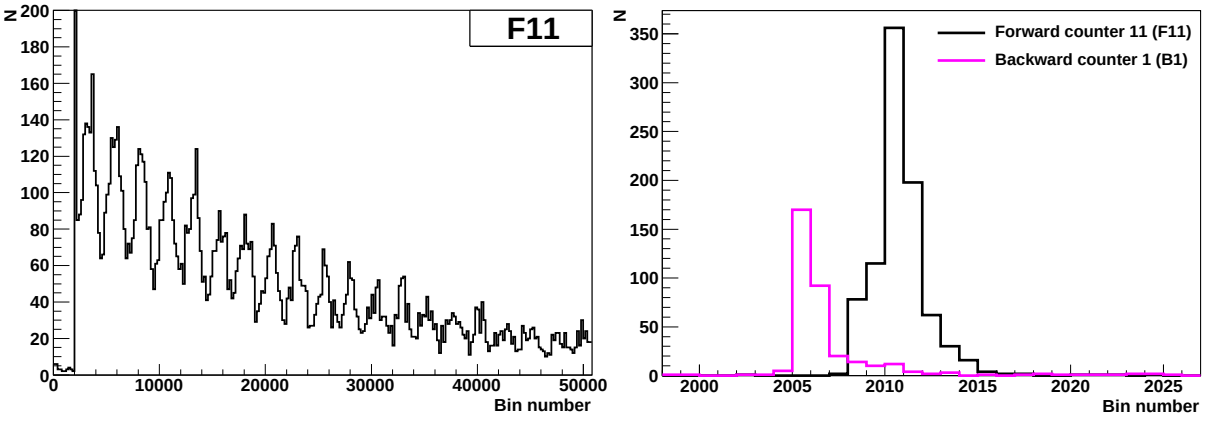

Figure 7. μSR signal in the forward positron counter No. 11 (run 201, TF= 300 gauss). The left plot shows the (rebinned) signal in the counter, the right plot shows the detail of the *prompt peak*, i.e. the region around t_0 in the same counter (black line), compared with the prompt peak in the backward positron counter No. 1 (magenta line).

was defined with the bin width of 100 ps. The number of bins was 50995, covering the time interval of approx. (-0.2 μs, 4.9 μs). In the left hand side plot, however, the histogram was rebinned (200 bins were summed up into 1 bin). The right hand side plot shows the detail of the *prompt peak*, i.e. the region around t_0, of one forward and one backward positron counters, prior to the rebinning.

5.4 Conclusion of the GPD analysis example

The purpose of the example analysis of the GPD simulation was to illustrate the potential of *musrSim* and *musrSimAna* programs to investigate features like time-independent background, sample muon fraction, prompt peak, ... This information can be used in design and optimisation of μSR instruments.

References

1. T. Prokscha *et al.* "A novel VME based μSR data acquisition system at PSI", Physica **B 404**, (2009) 1007-1009.
2. "TDC Manual – Setting up the required logic", http://lmu.web.psi.ch/facilities/electronics/TDC/set_logic.html
3. http://lmu.web.psi.ch/facilities/gpd/gpd.html
4. K.Sedlak *et al.*, "Manual of musrSim".