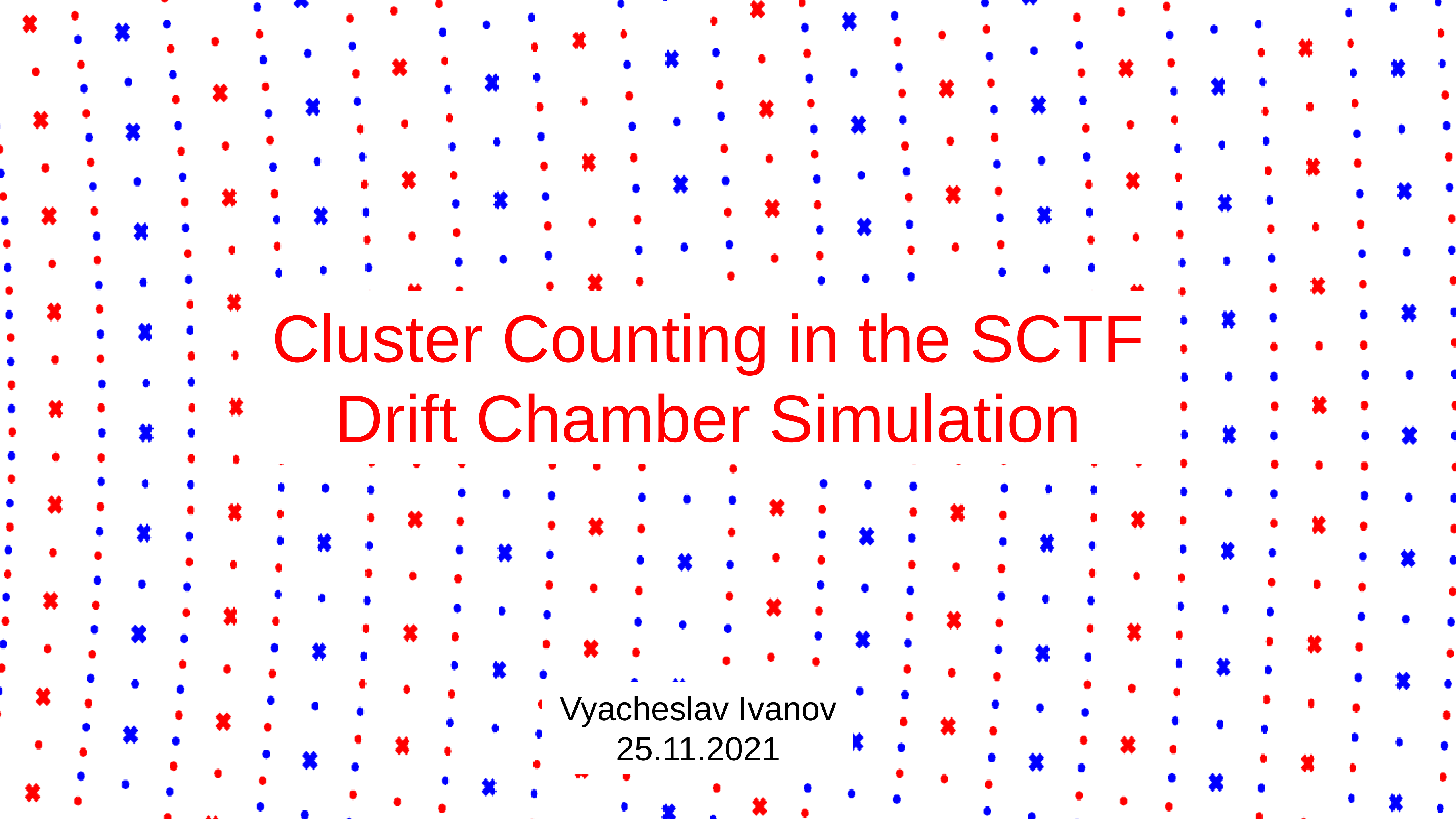
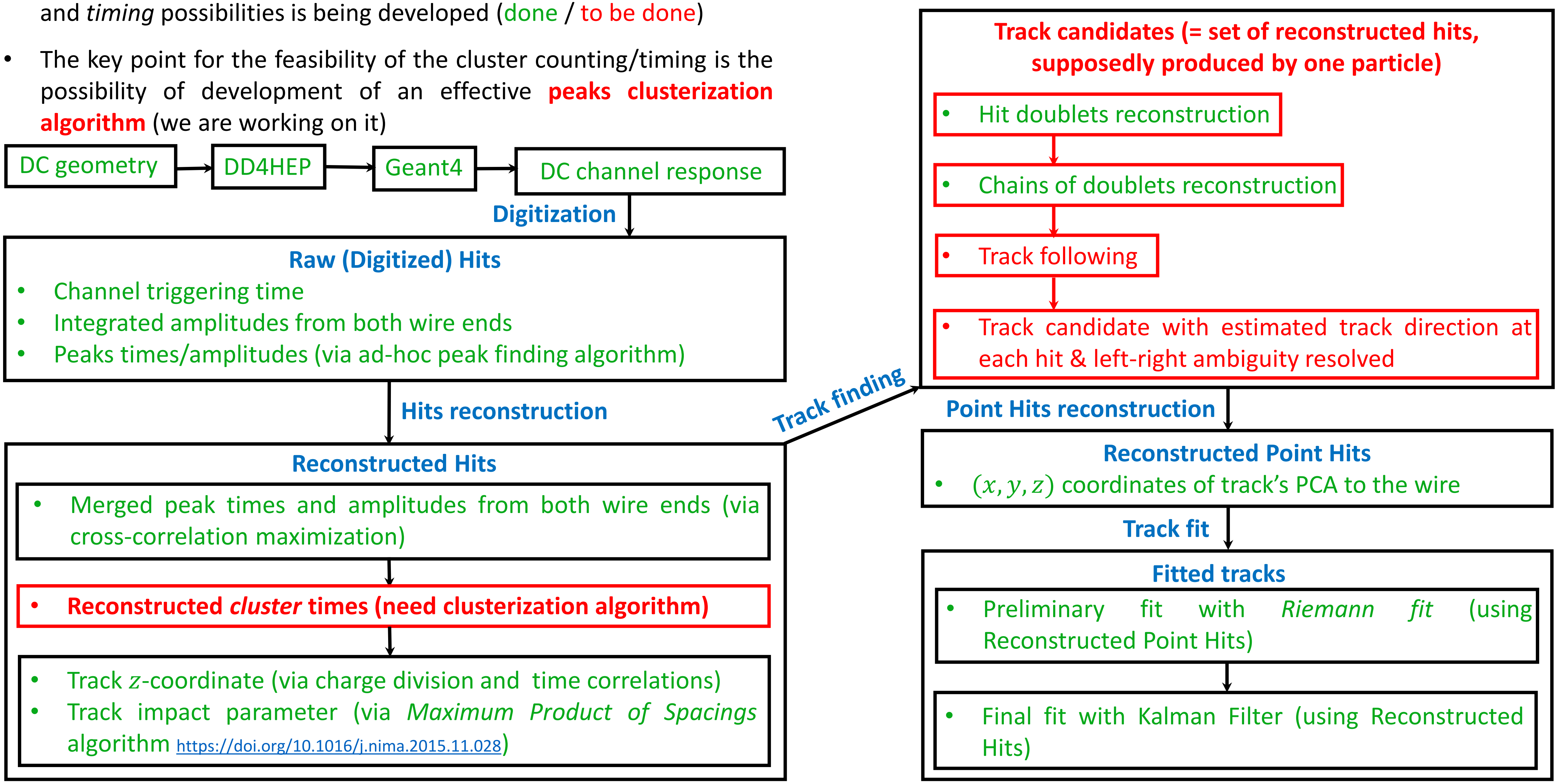


Cluster Counting in the SCTF Drift Chamber Simulation

Vyacheslav Ivanov
25.11.2021

Drift chamber simulation chain

- Simulation of the full stereo drift chamber with *cluster counting* and *timing* possibilities is being developed (done / to be done)
- The key point for the feasibility of the cluster counting/timing is the possibility of development of an effective **peaks clusterization algorithm** (we are working on it)



DC geometry → DD4HEP → Geant4 → DC channel response

Digitization

Raw (Digitized) Hits

- Channel triggering time
- Integrated amplitudes from both wire ends
- Peaks times/amplitudes (via ad-hoc peak finding algorithm)

Hits reconstruction

Reconstructed Hits

- Merged peak times and amplitudes from both wire ends (via cross-correlation maximization)
- **Reconstructed *cluster* times (need clusterization algorithm)**
- Track z-coordinate (via charge division and time correlations)
- Track impact parameter (via *Maximum Product of Spacings* algorithm https://doi.org/10.1016/j.nima.2015.11.028)

Track finding

Track candidates (= set of reconstructed hits, supposedly produced by one particle)

- Hit doublets reconstruction
- Chains of doublets reconstruction
- Track following
- Track candidate with estimated track direction at each hit & left-right ambiguity resolved

Point Hits reconstruction

Reconstructed Point Hits

- (x, y, z) coordinates of track's PCA to the wire

Track fit

Fitted tracks

- Preliminary fit with *Riemann fit* (using Reconstructed Point Hits)
- Final fit with Kalman Filter (using Reconstructed Hits)

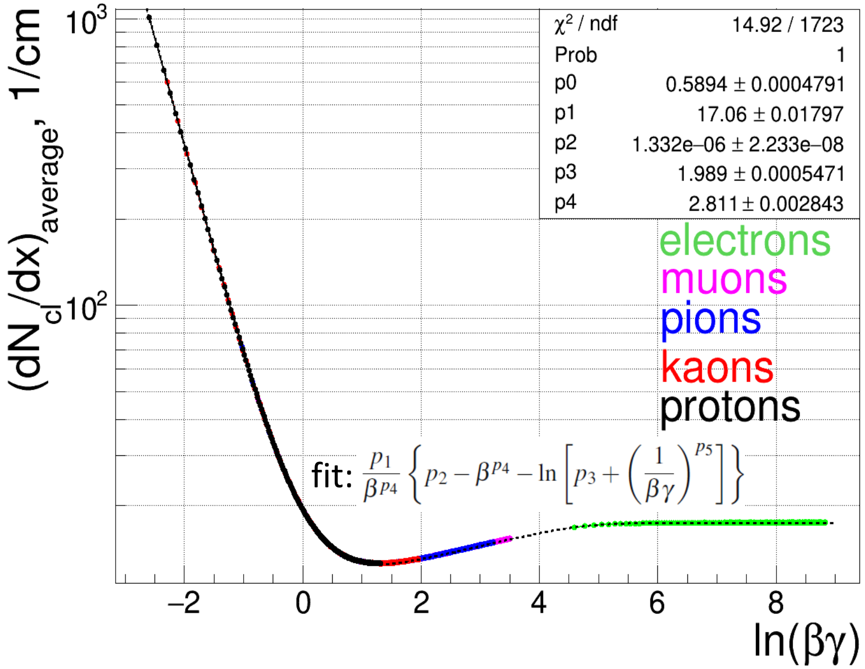
Ionization clusters generation

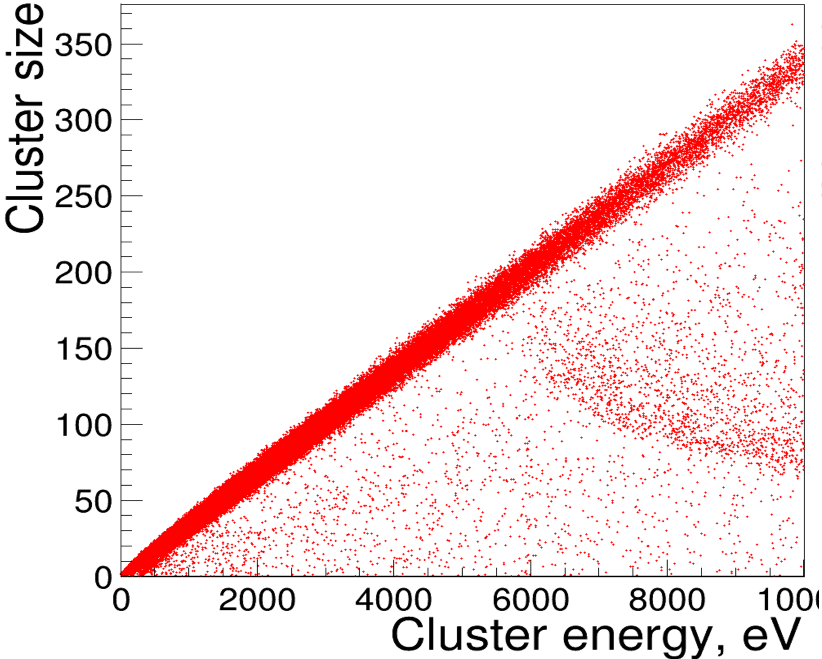
- On the spiral stretched between the beginning and the ending point of the GEANT4 hit (G4Step) we generate the ionization clusters

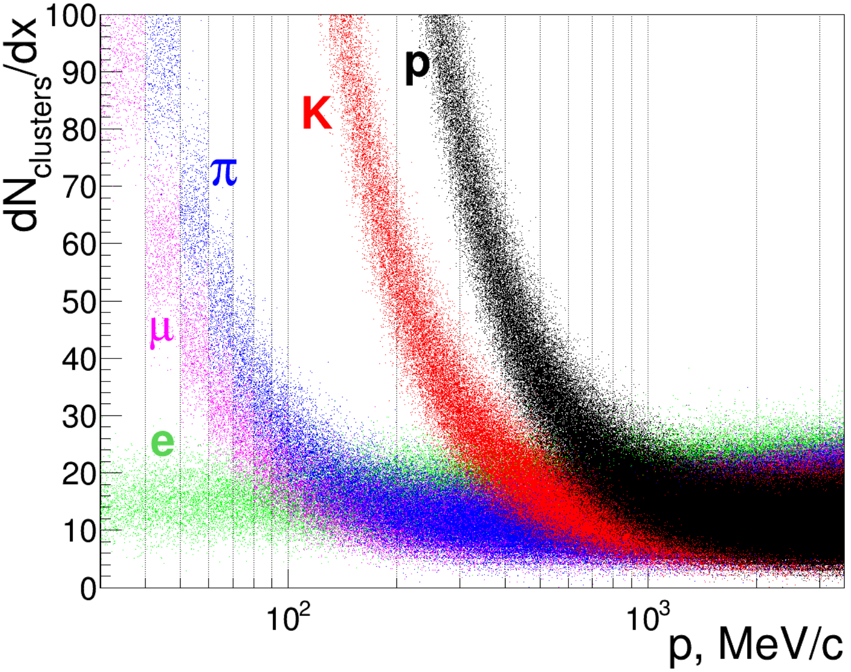
- The cluster positions on track are generated uniformly using the $(dN_{clusters}/dx)_{average}$ curve, obtained from Garfield++

- The energy of each cluster (E_{cl}) is generated according to the energy transfer spectra, predicted by Garfield::TrackHeed

- The number of electrons (cluster size) in each cluster is calculated from the E_{cl}, taking into account the Garfield-predicted average energy of the electron-ion pair production $W = 29.52$ eV: $N_{electrons} = E_{cl}/W + \delta$, where δ is the fluctuation with sigma equals $\sqrt{FN_{electrons}}$, where the $F = 0.19$ is the Fano factor

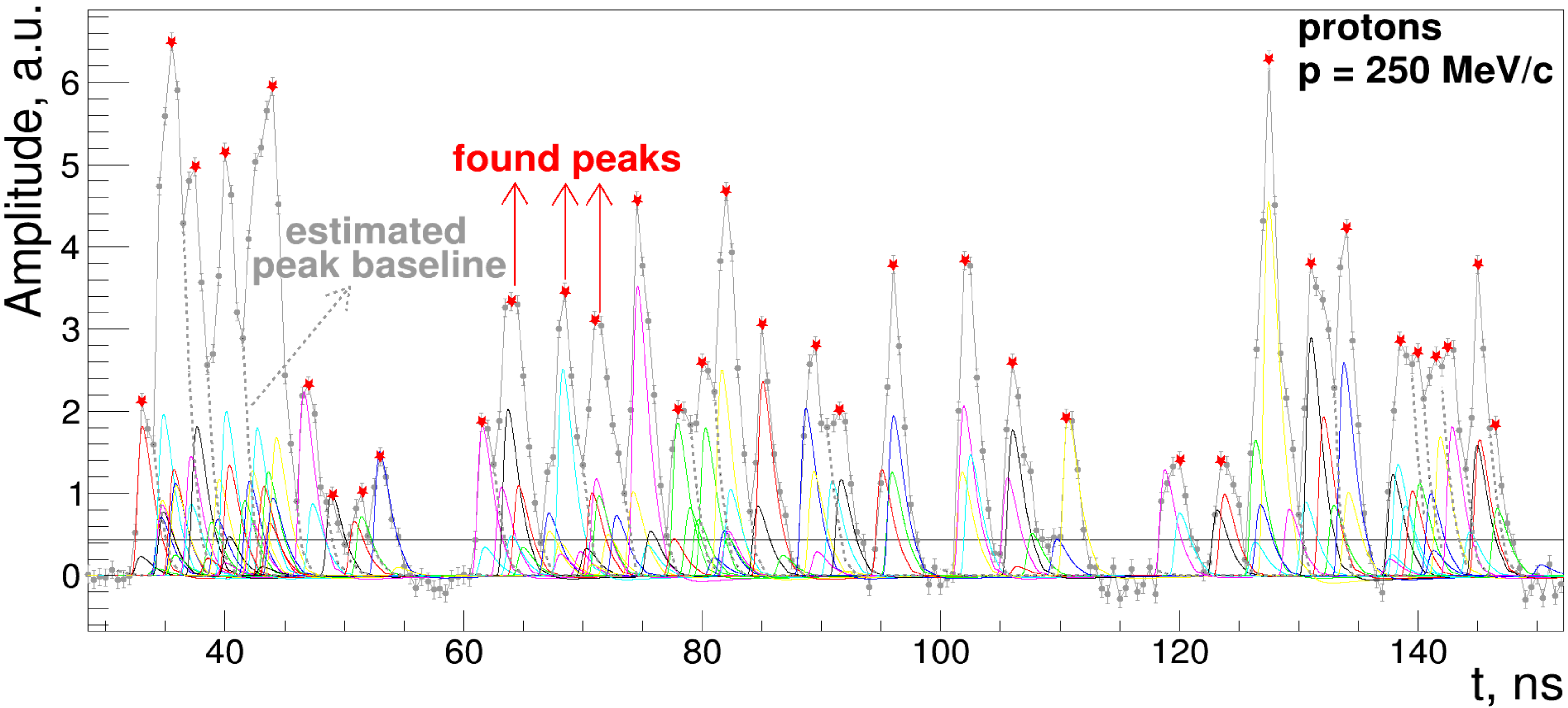






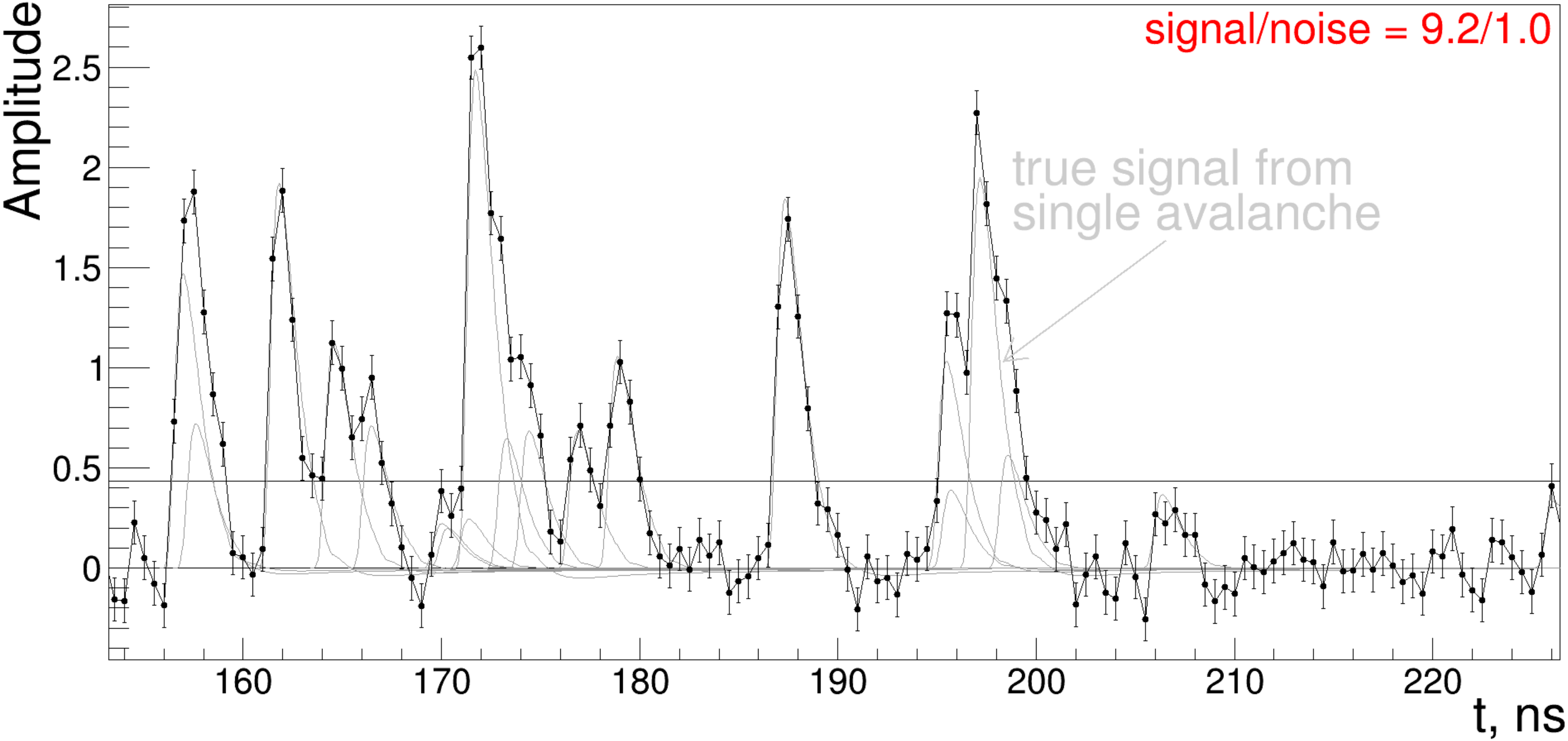
Peak finding algorithm

- We developed an ad-hoc peak finding algorithm, based on the dynamic estimation of the **baseline level**
- **The main question: if this algorithm is suitable for FPGA?**

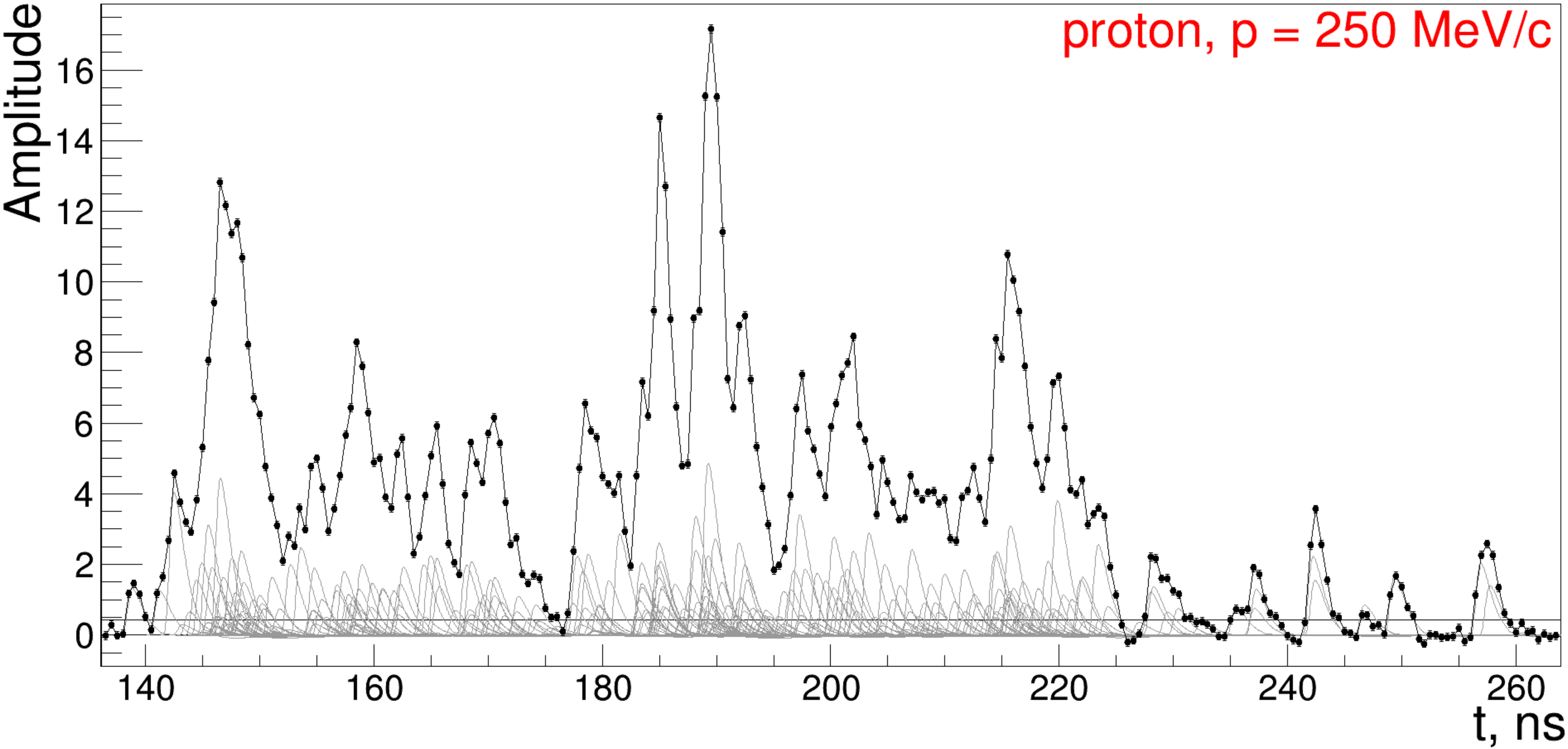


Typical waveform (muons)

- Digitization time step is 0.5 ns (freq. = 2 GHz), signal/noise = 9.2/1.0 (according to V.M. Aulchenko)

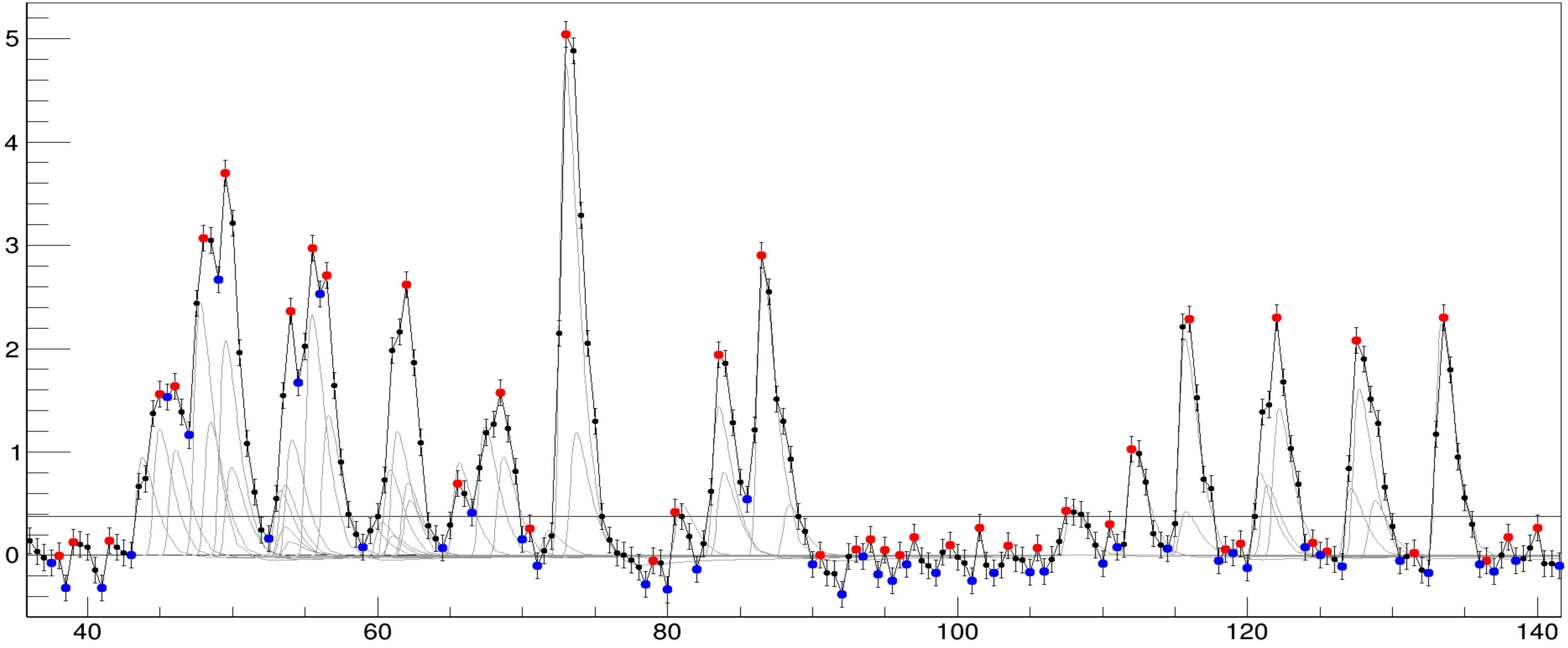


Typical waveform (protons)

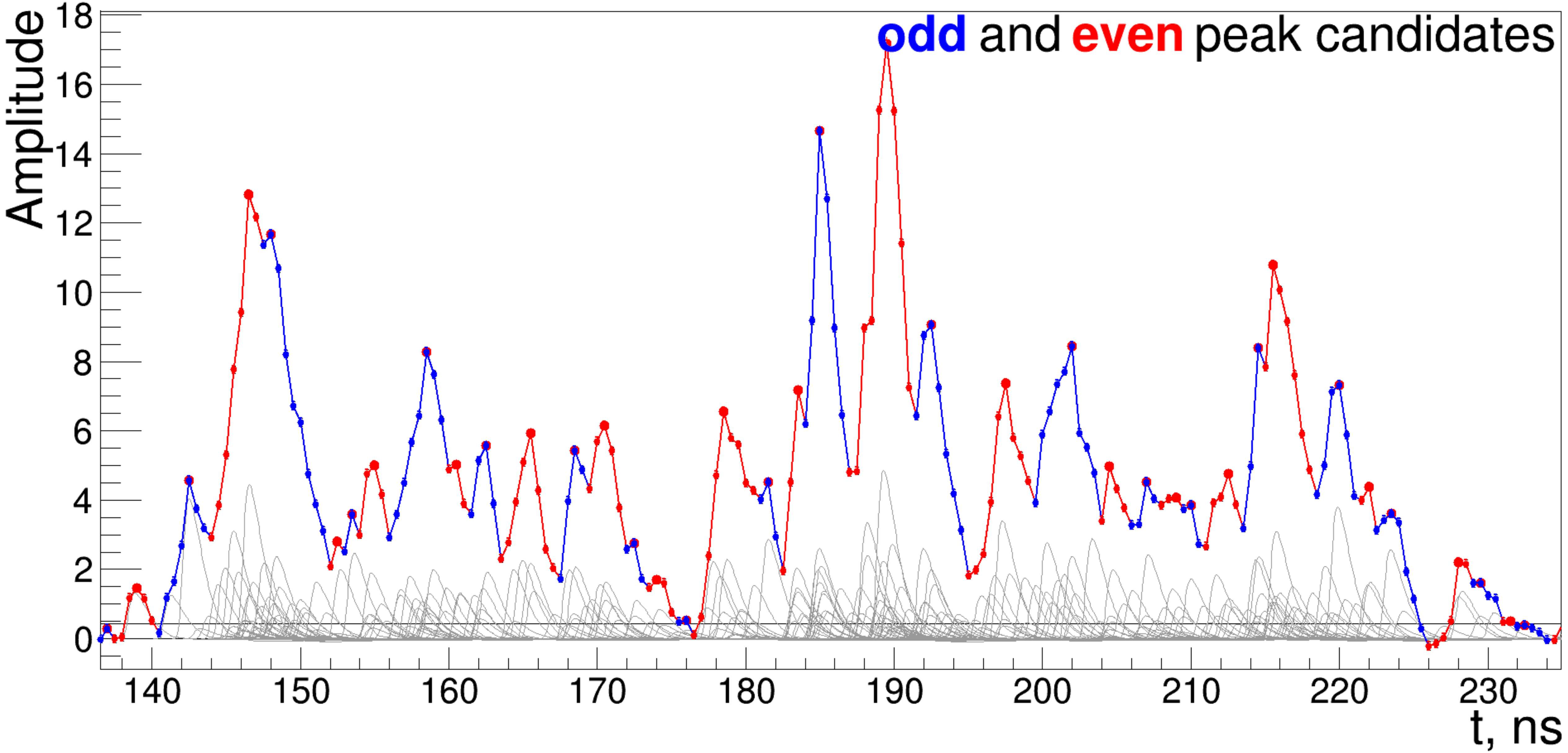


The idea of my peak finding approach

- The waveform contains local minimums and local maximums
- **Each** waveform segment “loc. min. – loc. max. – loc. min” is considered as *peak candidate*
- Peak candidate is identified as *real peak* if it satisfies a *quality criterion.* Currently one peak candidate can give only one real peak
- To calculate the peak quality correctly, one should account for the **baseline** shift, caused by the previous peaks
- Thus, for each peak candidate we should estimate the **baseline** it resides on (“running baseline”)

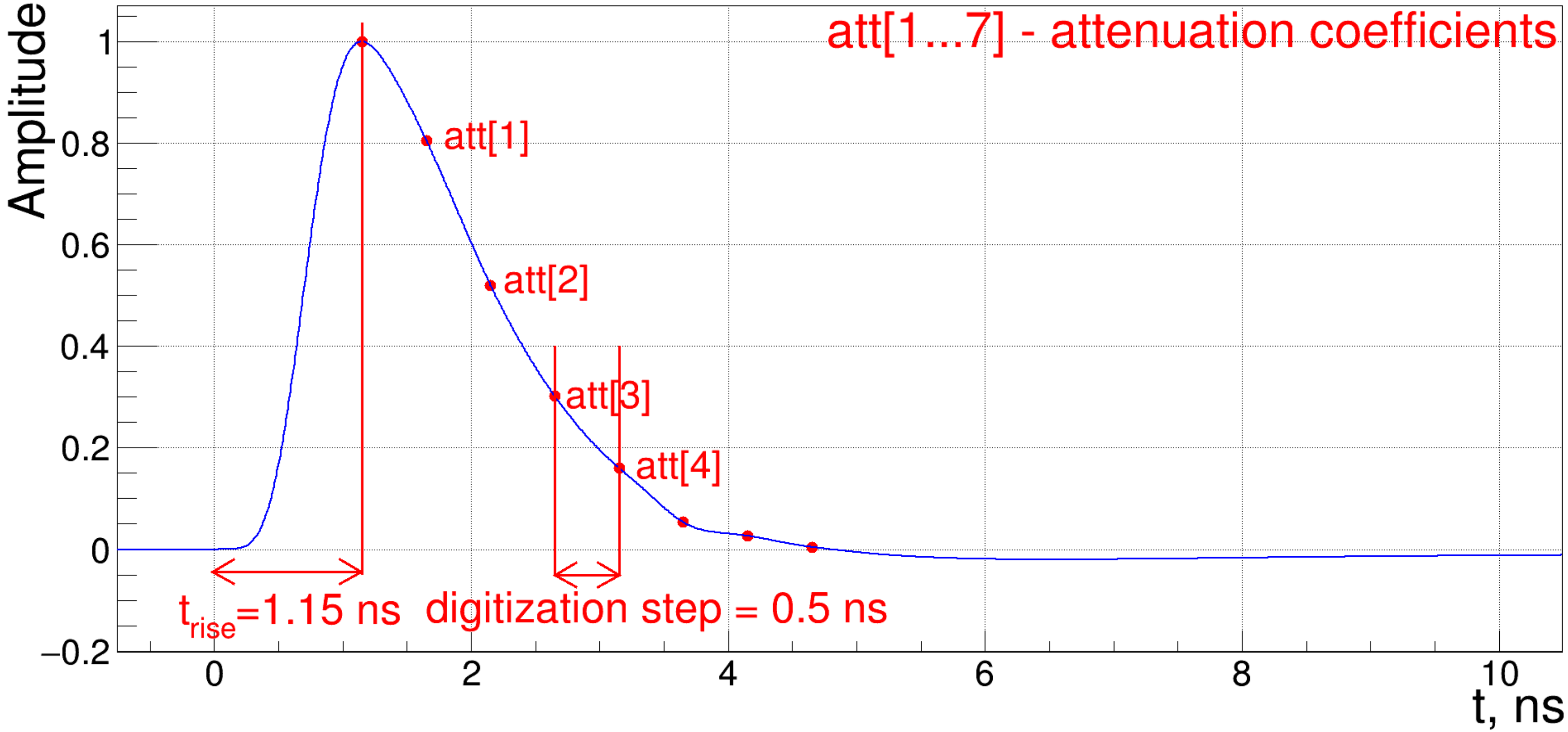


Peak candidates in the waveform (protons)



Signal shape and attenuation coefficients (protons)

- To estimate the **baseline** it is reasonable to benefit from our knowledge of the signal shape
- Signal rise time is ~ 1.15 ns (2 digitization steps)
- We calculate the attenuation coefficients att[1…7] for the 7 steps after the signal maximum
- In the real digitization the hitting to the signal maximum is not exact, this leads to ~5% inaccuracy in measuring of the amplitude of a single peak and to some inaccuracy of the attenuation coefficients

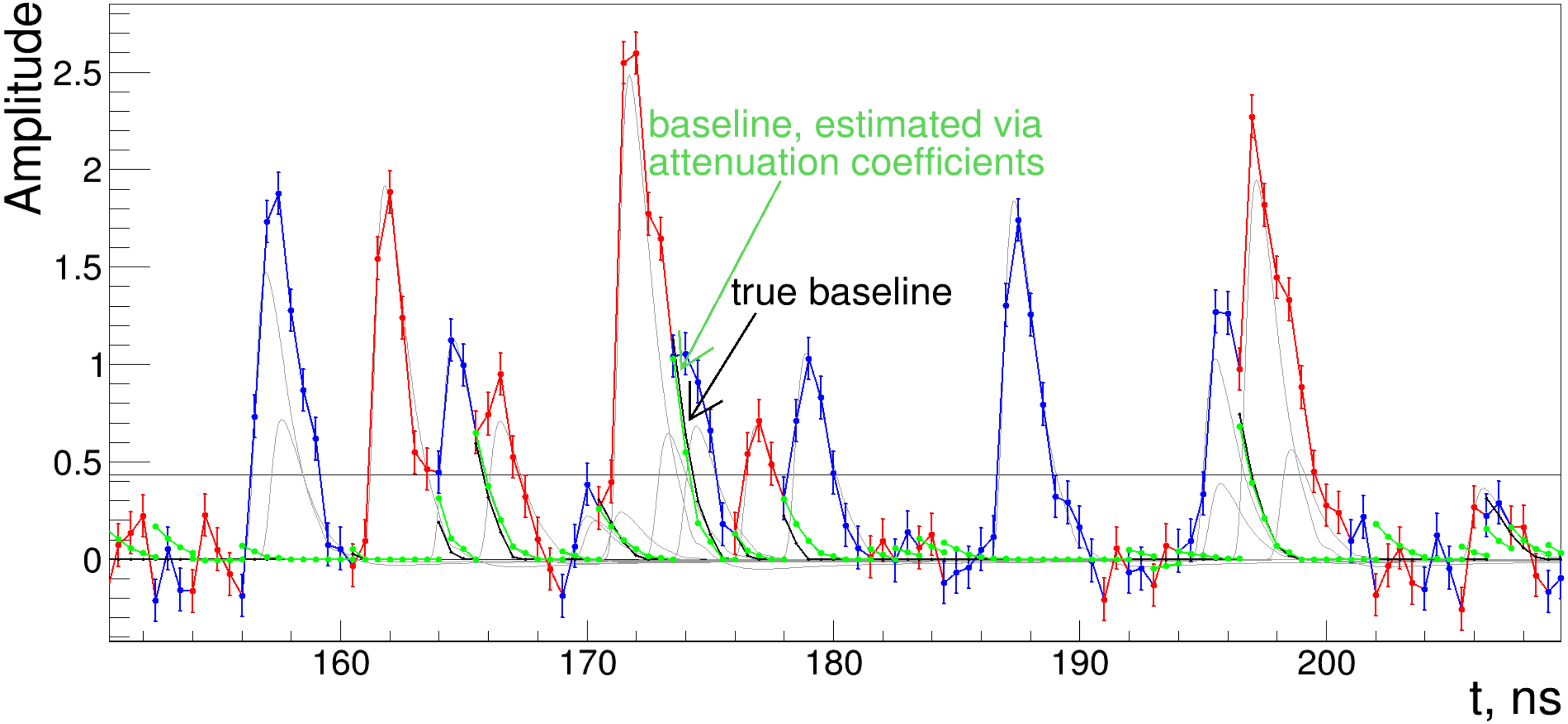


The baseline estimation (muons)

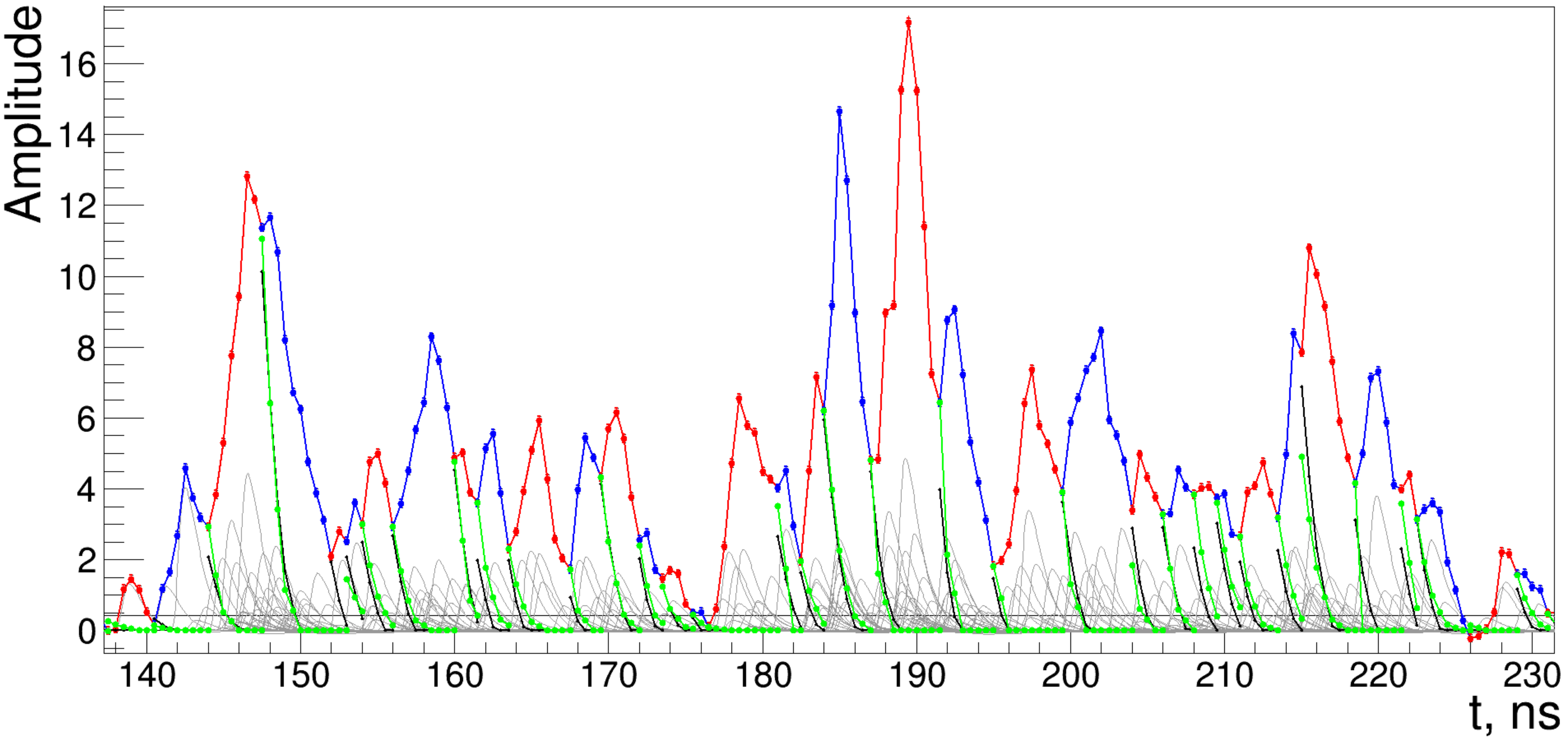
- To estimate the **baseline** of the current peak candidate we consider the previous 3 peak candidates as a real single signal peaks. Their contribution to the baseline is calculated using their amplitudes at their peaks (with subtraction of their baselines) and the attenuation coefficients

- Often due of the overlap of many peaks the signal shape gets deteriorated, and this leads to the wrong baseline estimation (as a rule - underestimation). To overcome this problem we scale the **baseline** with the *coefficient to make it equal to the amplitude at the first point of current peak candidate (= first local minimum)*. The only *exception* is the case of too short peak candidate rise time (1 digitization step). In this case the baseline if scaled to the lower value *first.min-(loc.max-first.min)*

- For control we compare the estimated **baseline** with the **true baseline**, calculated as a sum of the signals of all the avalanches, which reached the maximum before the starting point of the current candidate. In the vast majority of cases the agreement is good enough
- Currently we set **baseline** to 0, if it becomes <0 due to the signals undershoot



The baseline estimation (protons)

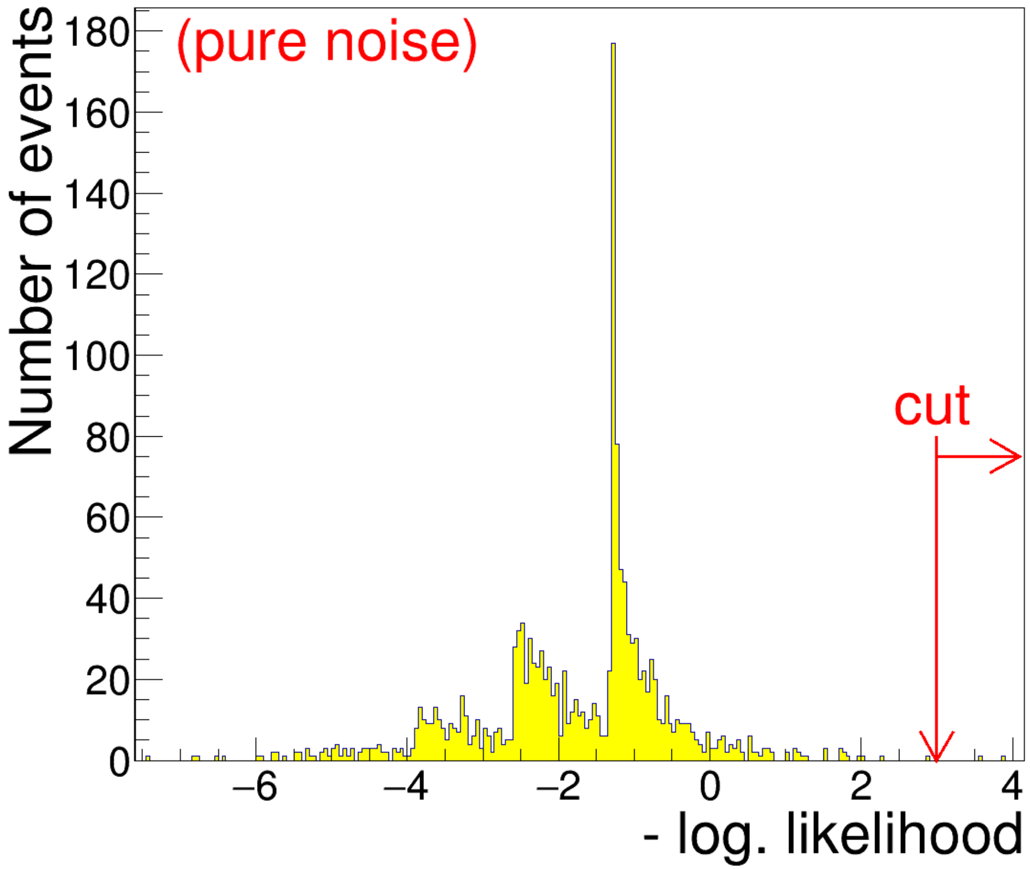


The peak quality estimation

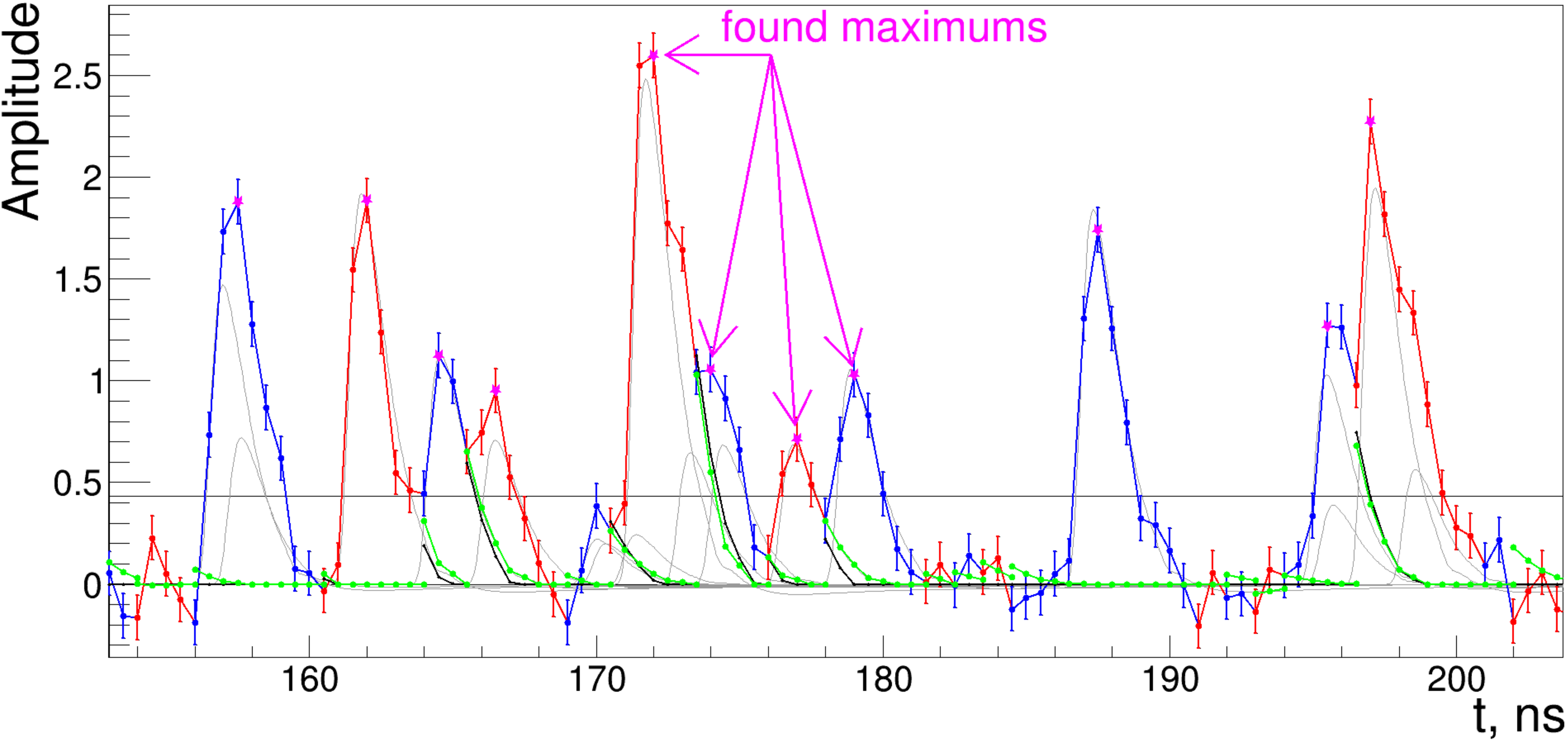
- After subtraction of the baseline (b_i) from the peak candidate amplitudes (a_i) we calculate peak quality q as a (minus) log. likelihood for the hypothesis, that this set of measurements was produced by a random Gaussian noise with known σ_{noise}. The two ending points (local minima) of peak candidates are not used:

$$q = \sum_{i=1}^{N} \frac{(a_i - b_i)^2}{\sigma_{noise}^2} + N \cdot \ln(2\pi\sigma_{noise}^2)$$

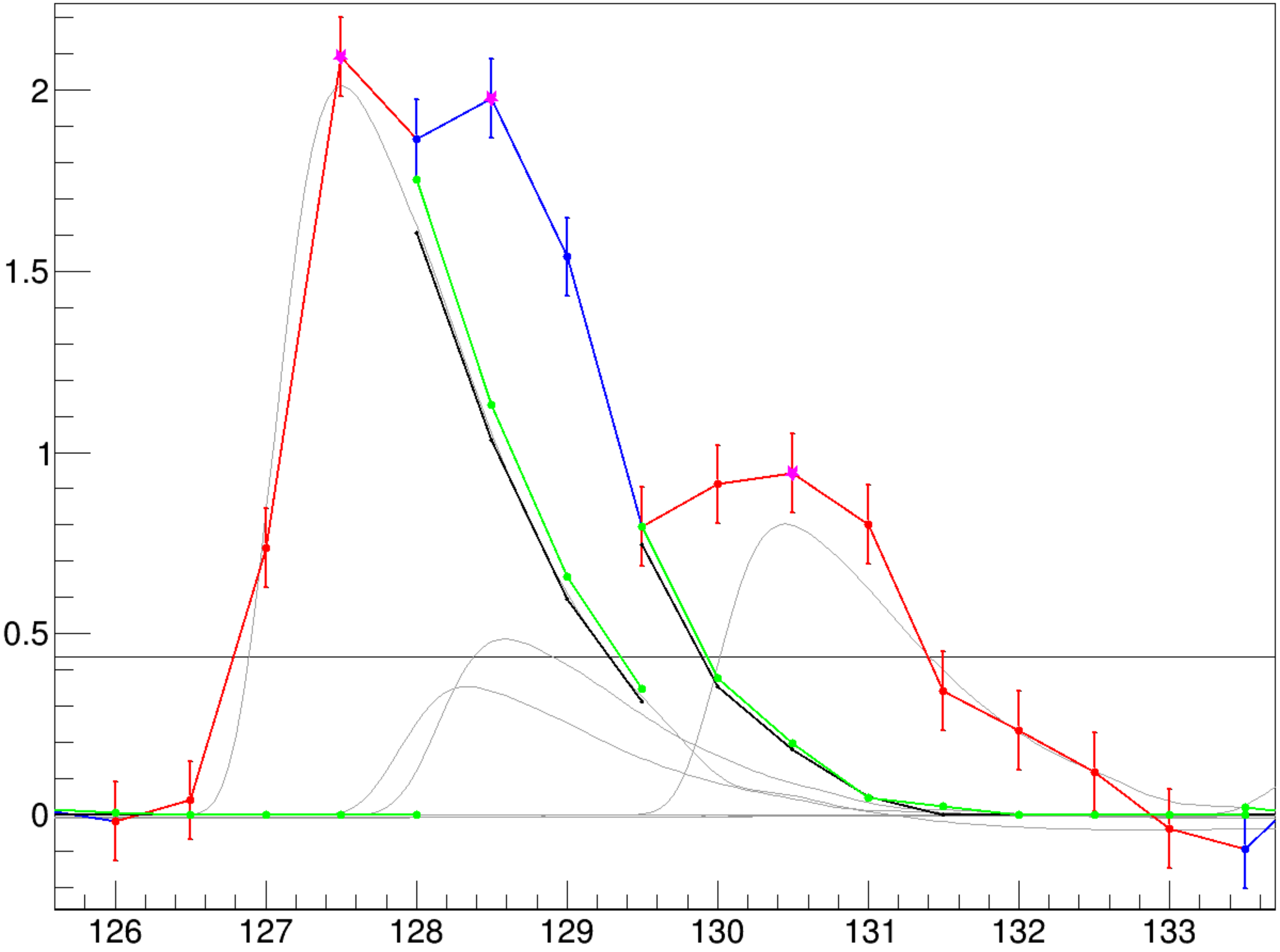
- The peak candidate is accepted as real peak if q>3.0



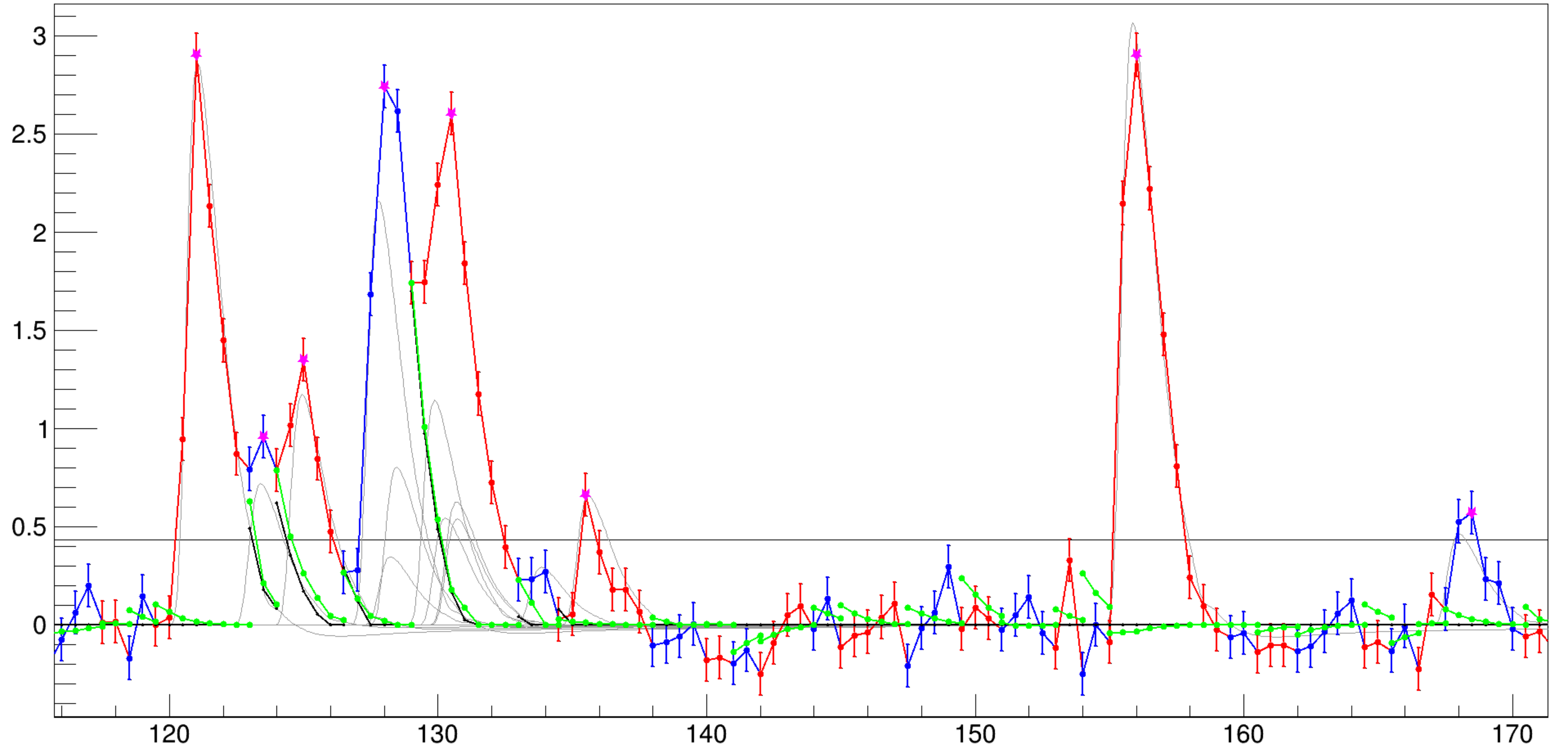
The found “real peaks” (muons)



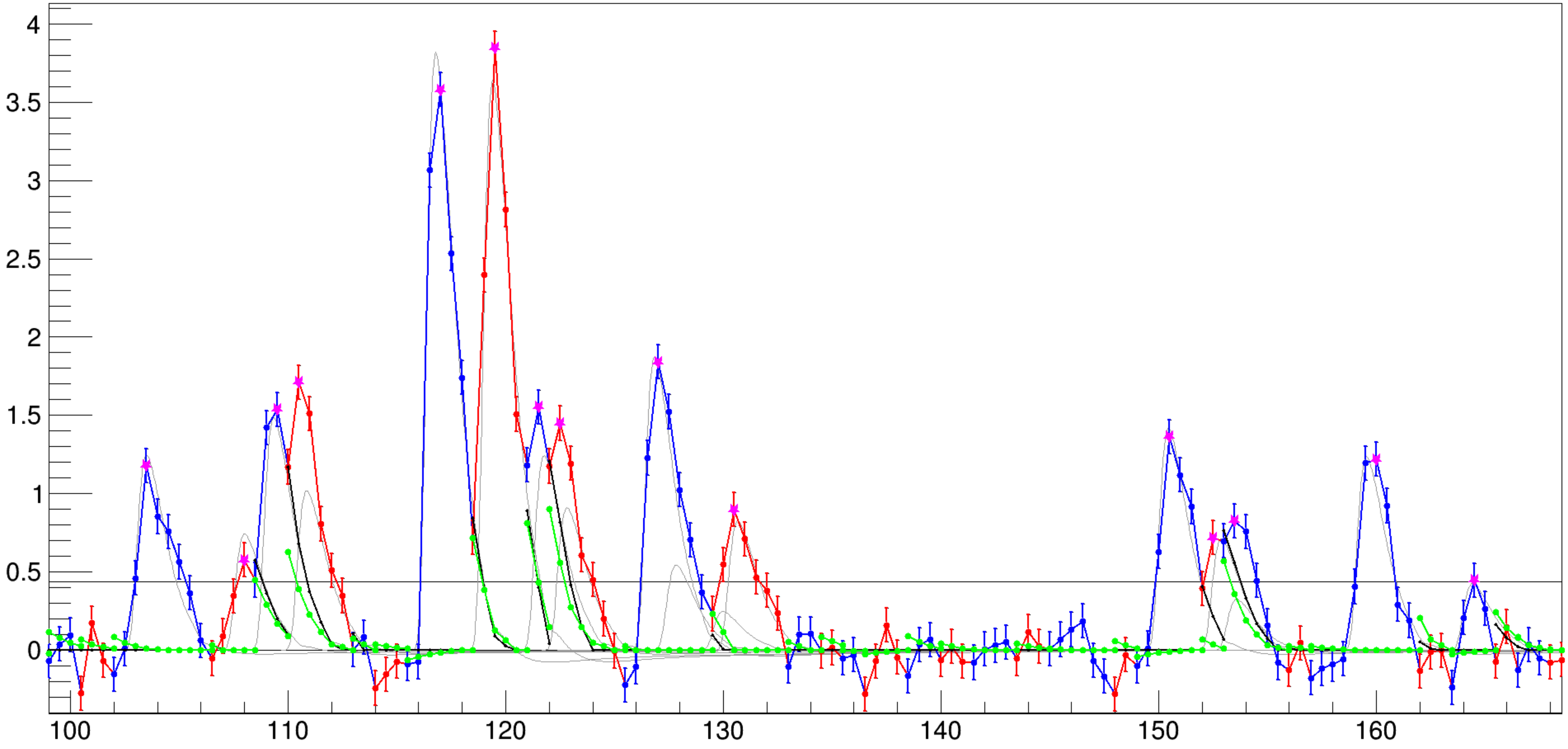
The found “real peaks” (muons)



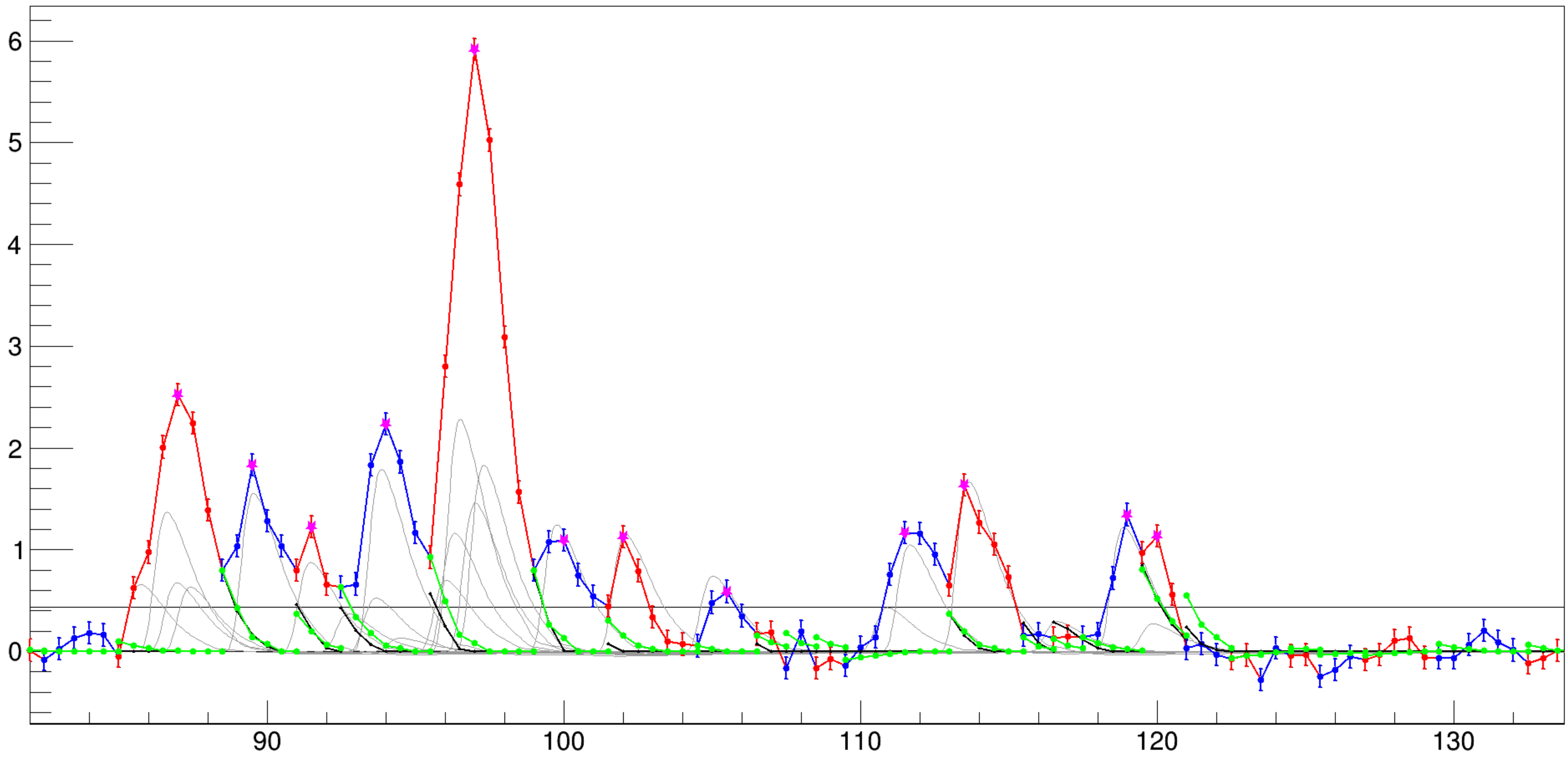
The found “real peaks” (muons)



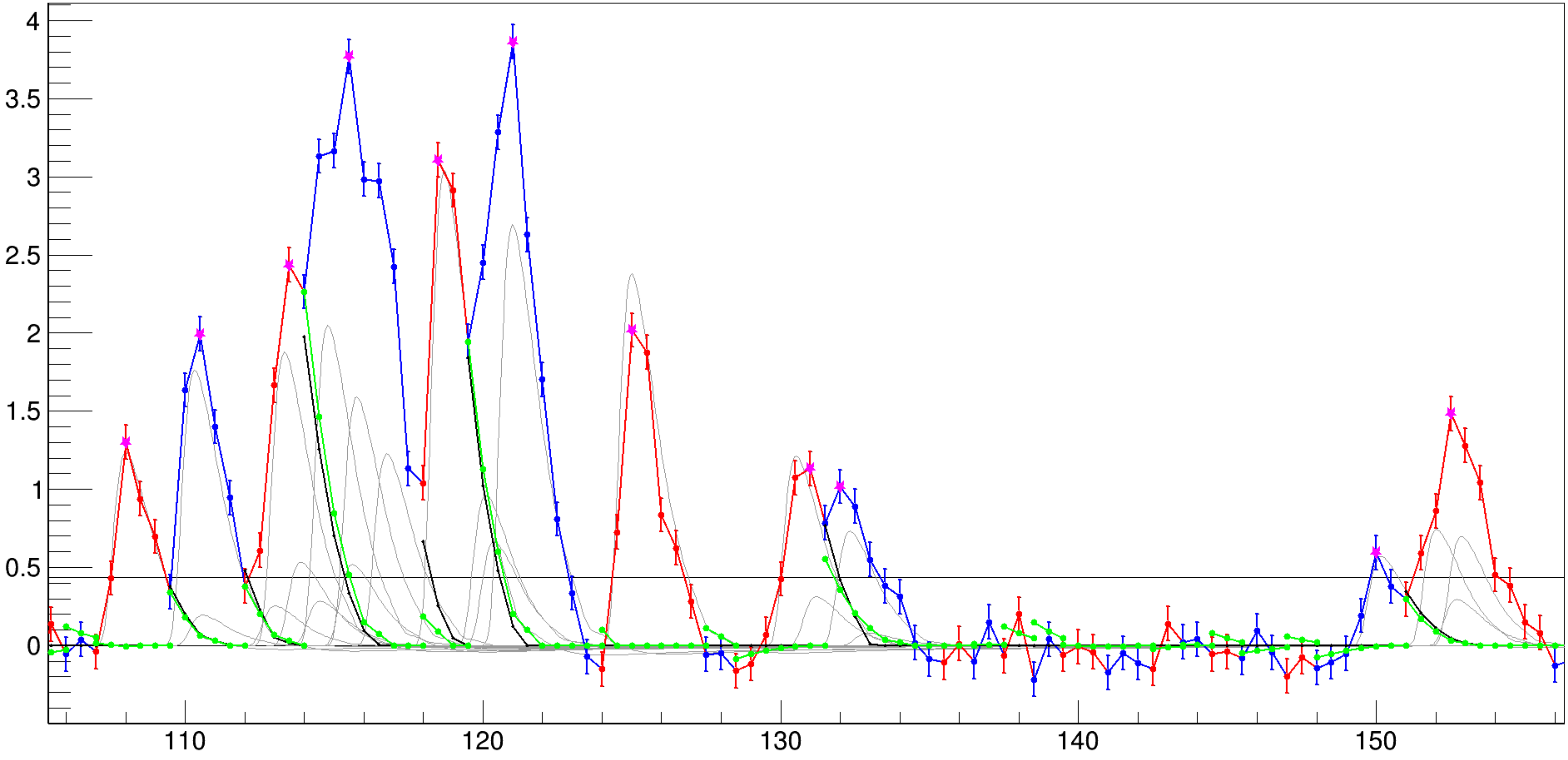
The found “real peaks” (muons)



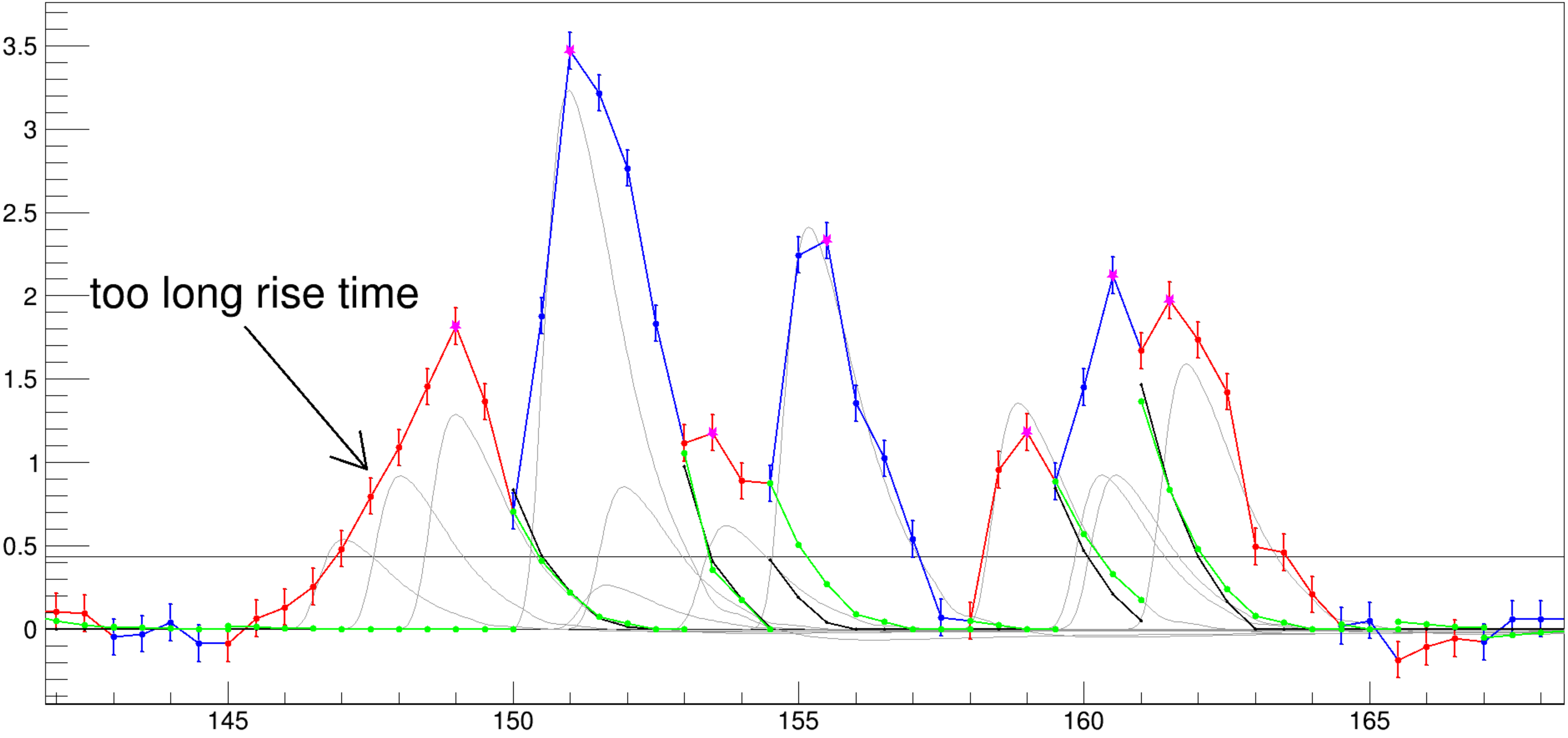
The found “real peaks” (muons)



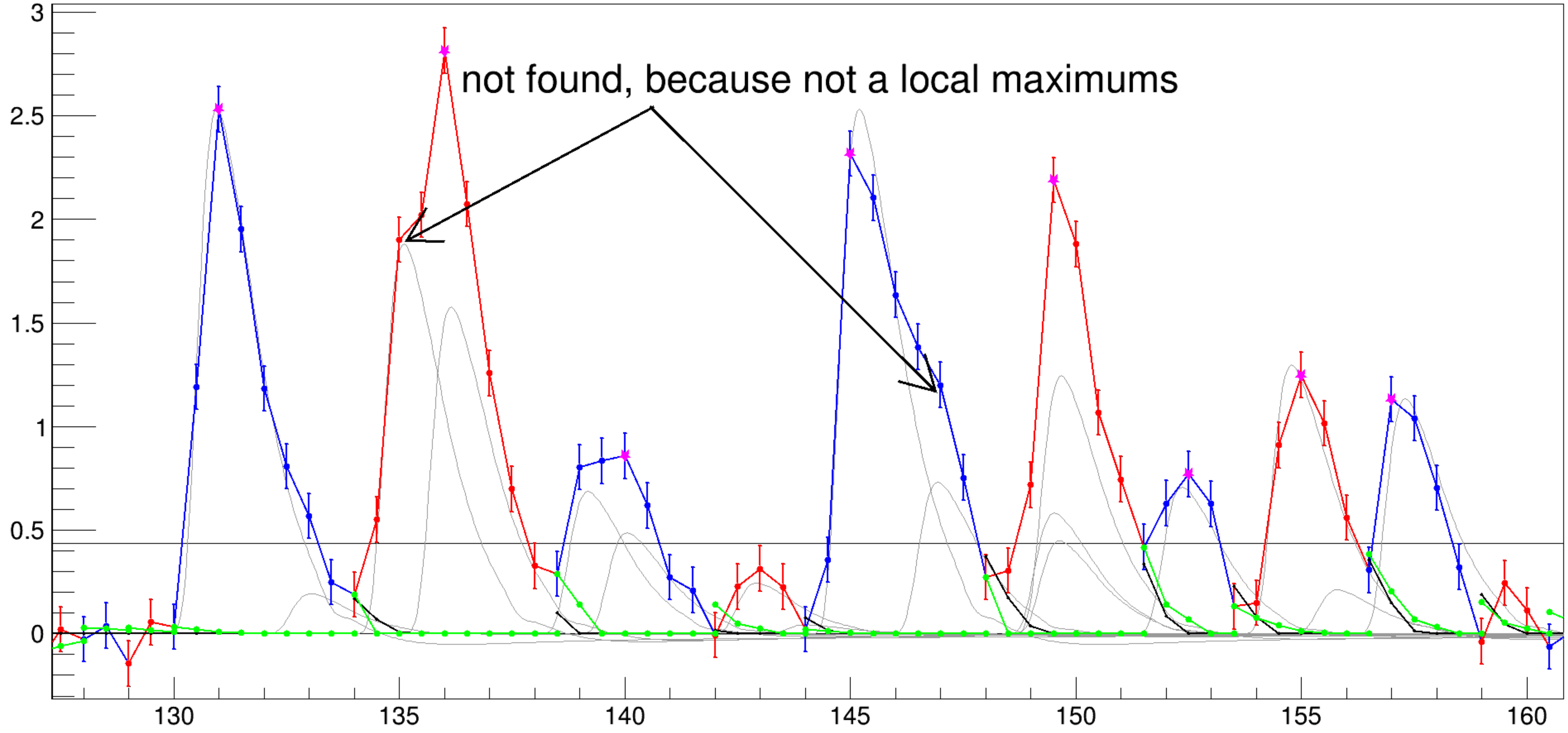
The found “real peaks” (muons)



The found “real peaks” (muons)

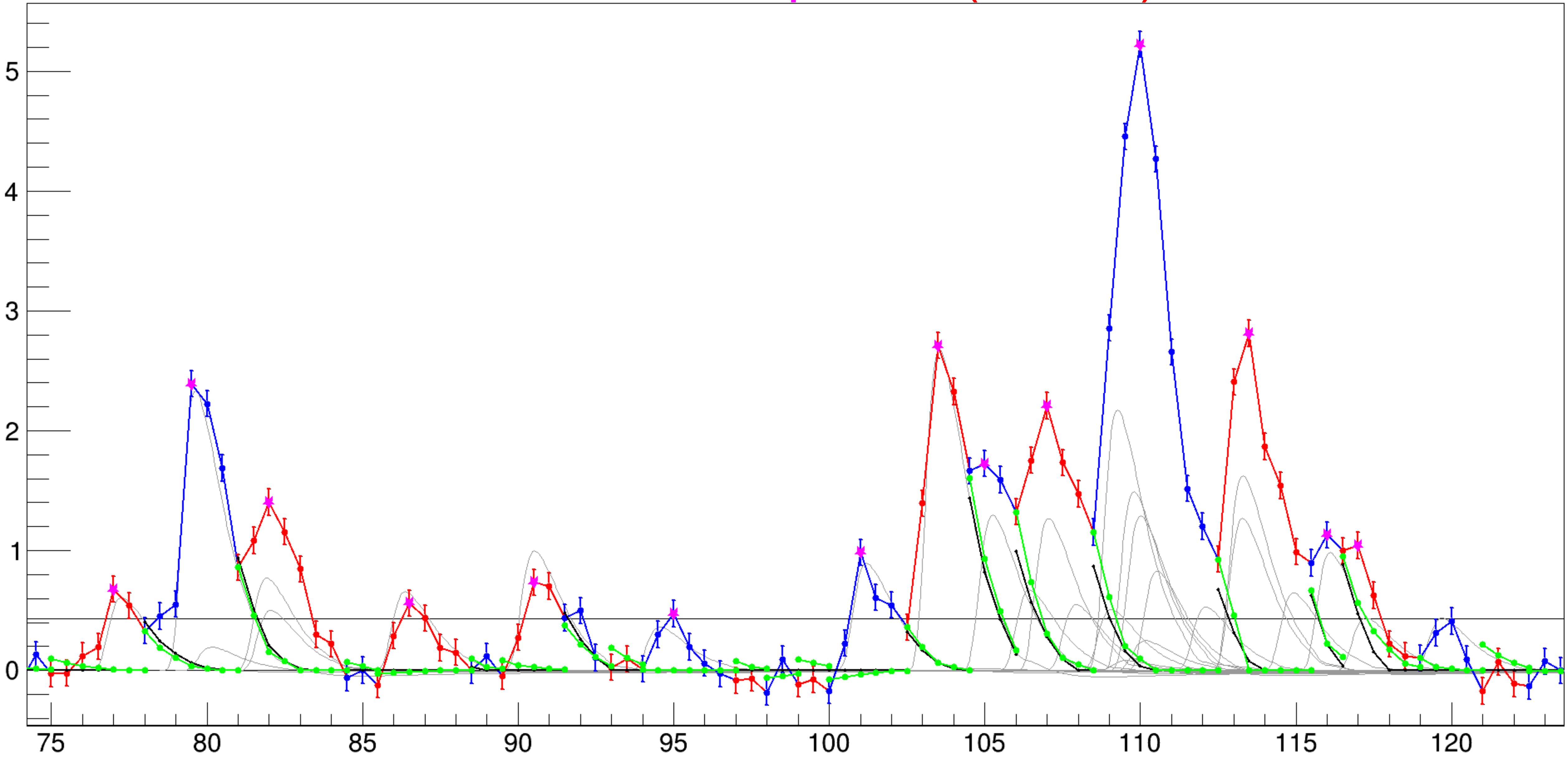


The found “real peaks” (muons)

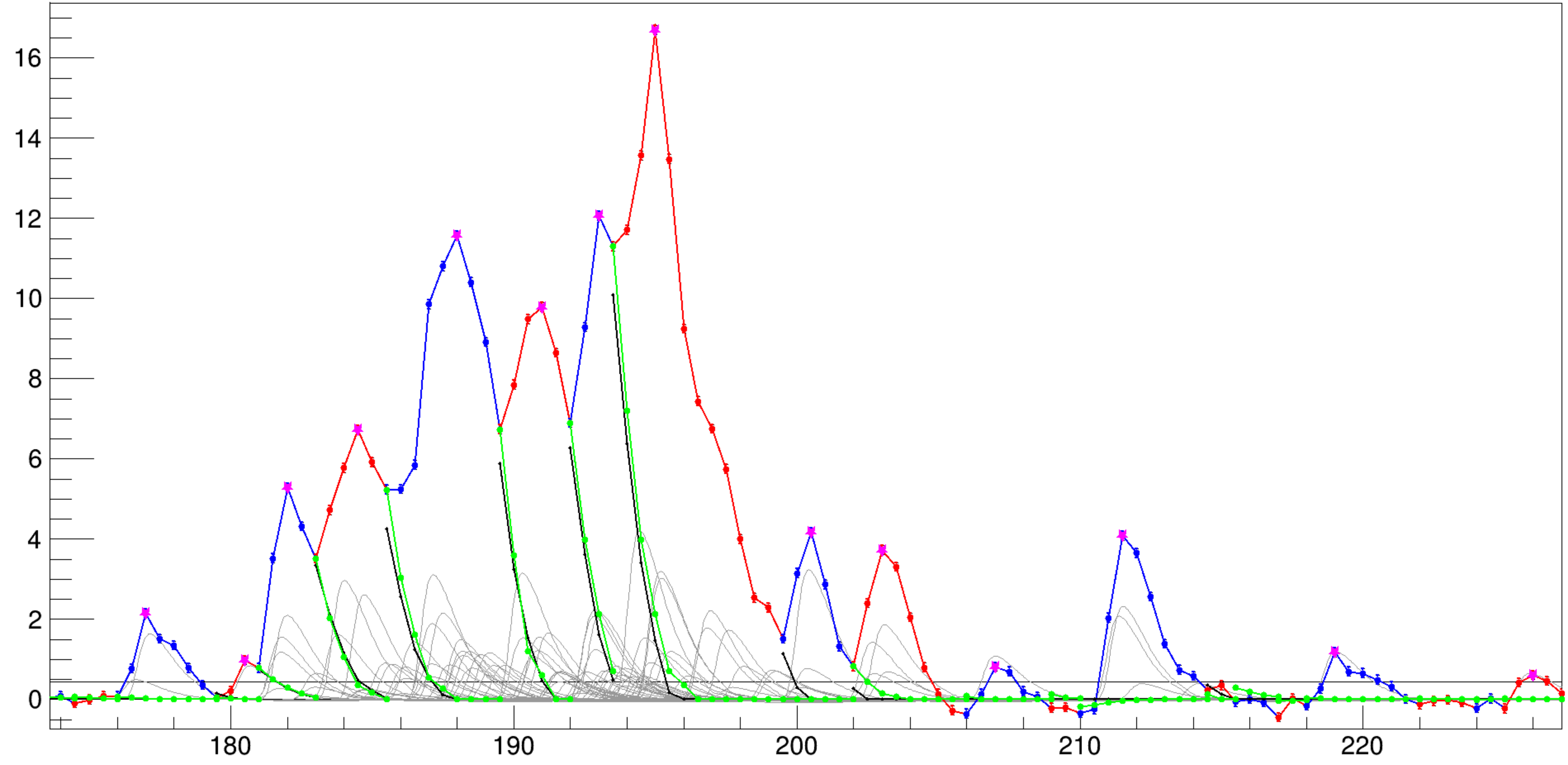


not found, because not a local maximums

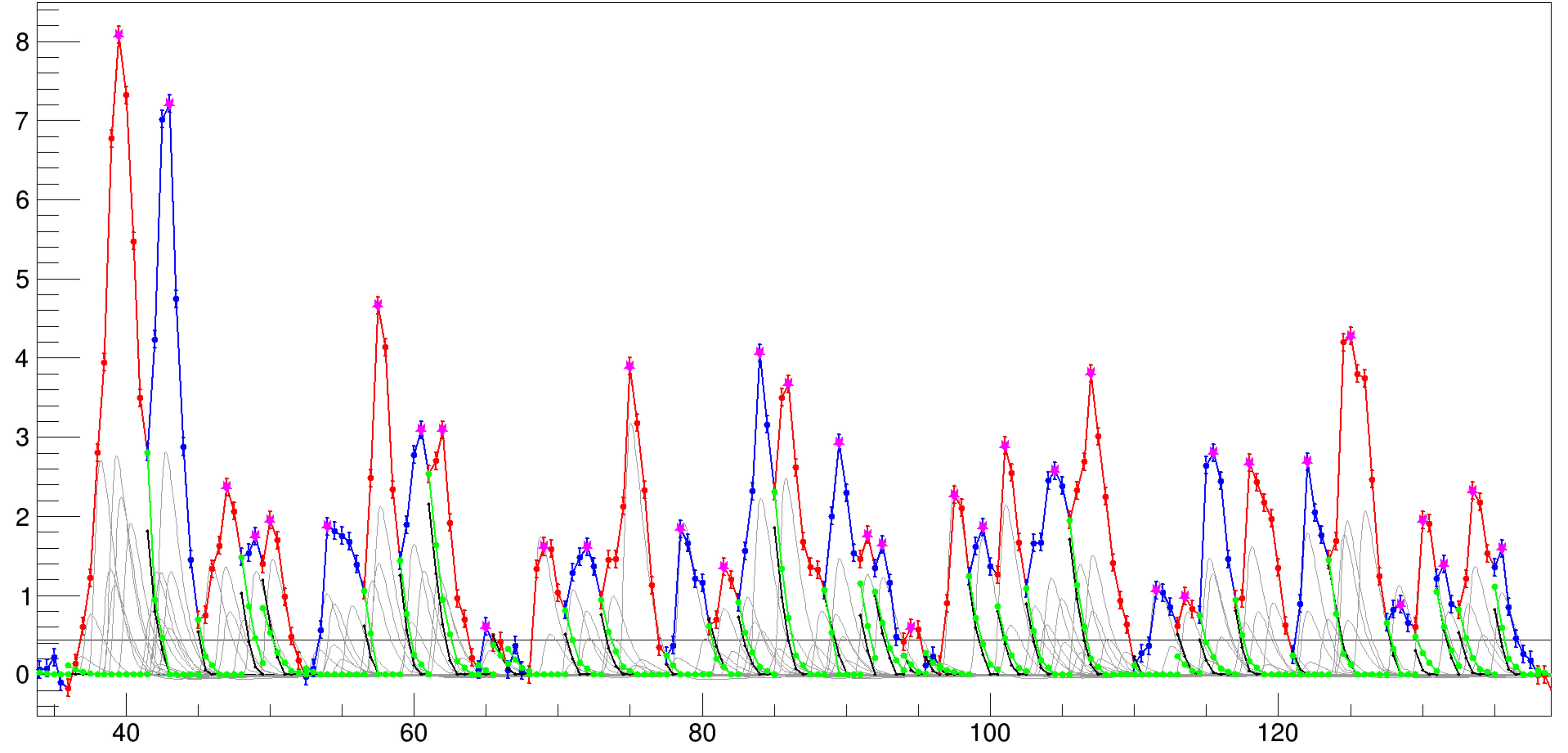
The found “real peaks” (muons)



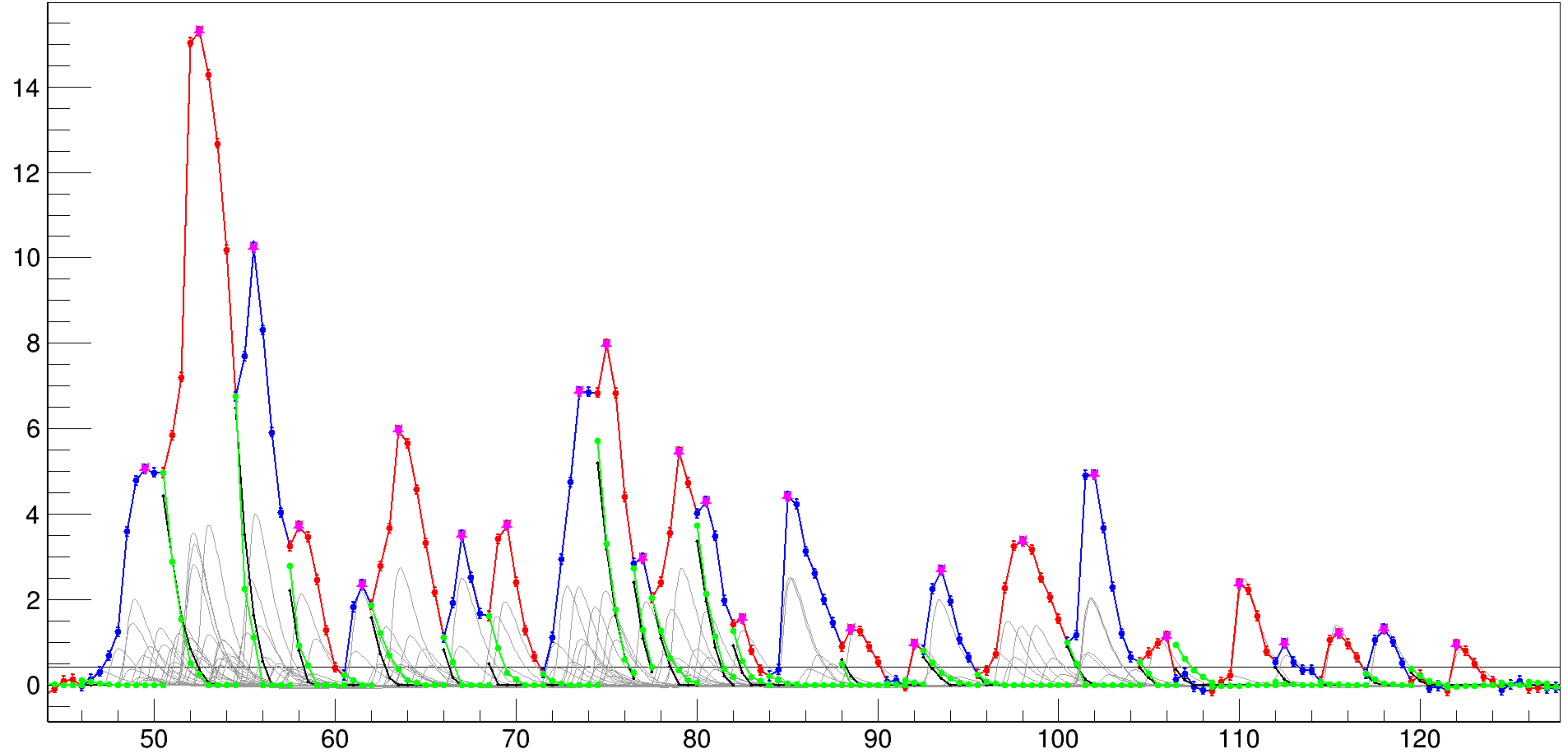
The found “real peaks” (protons)



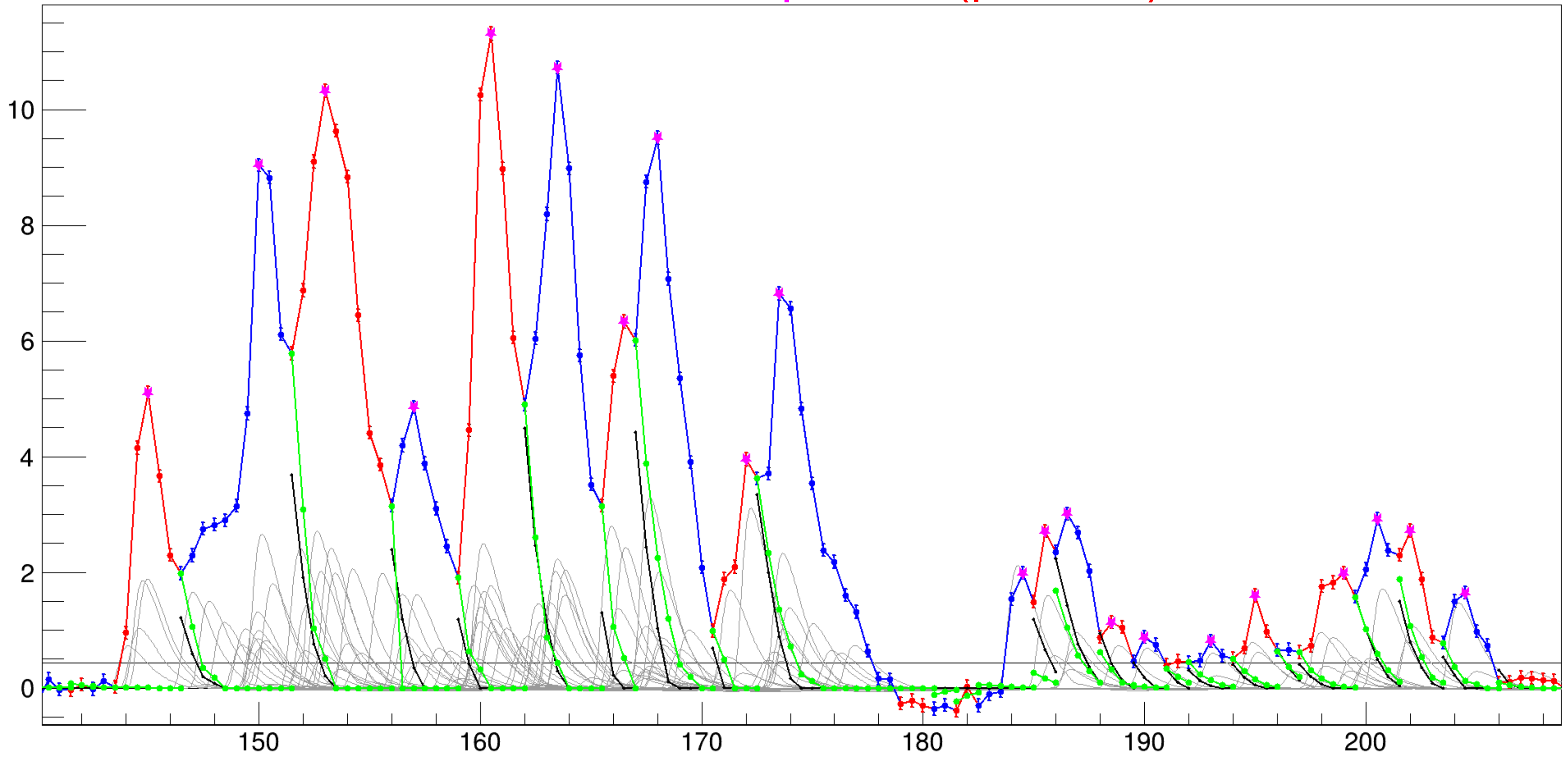
The found “real peaks” (protons)



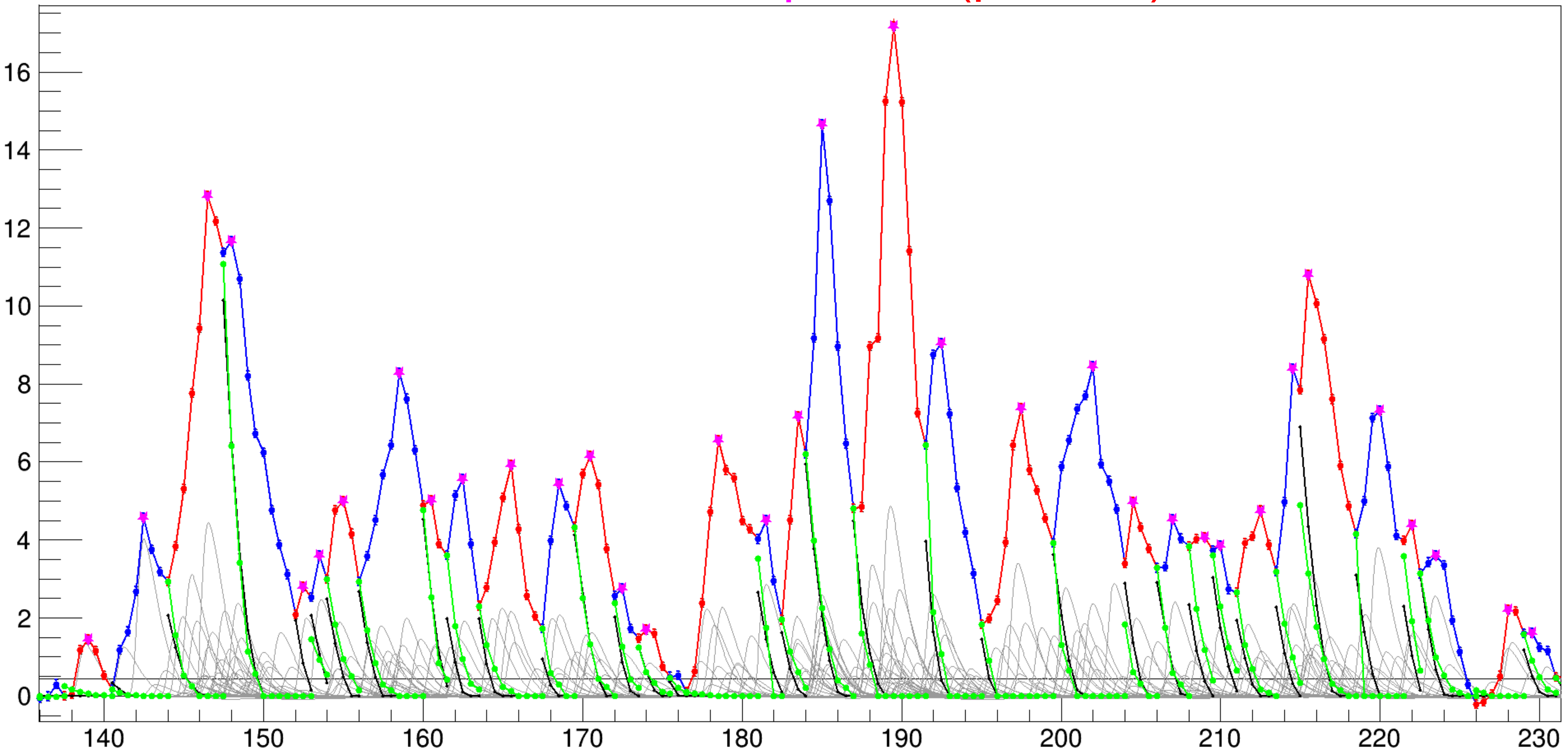
The found “real peaks” (protons)



The found “real peaks” (protons)

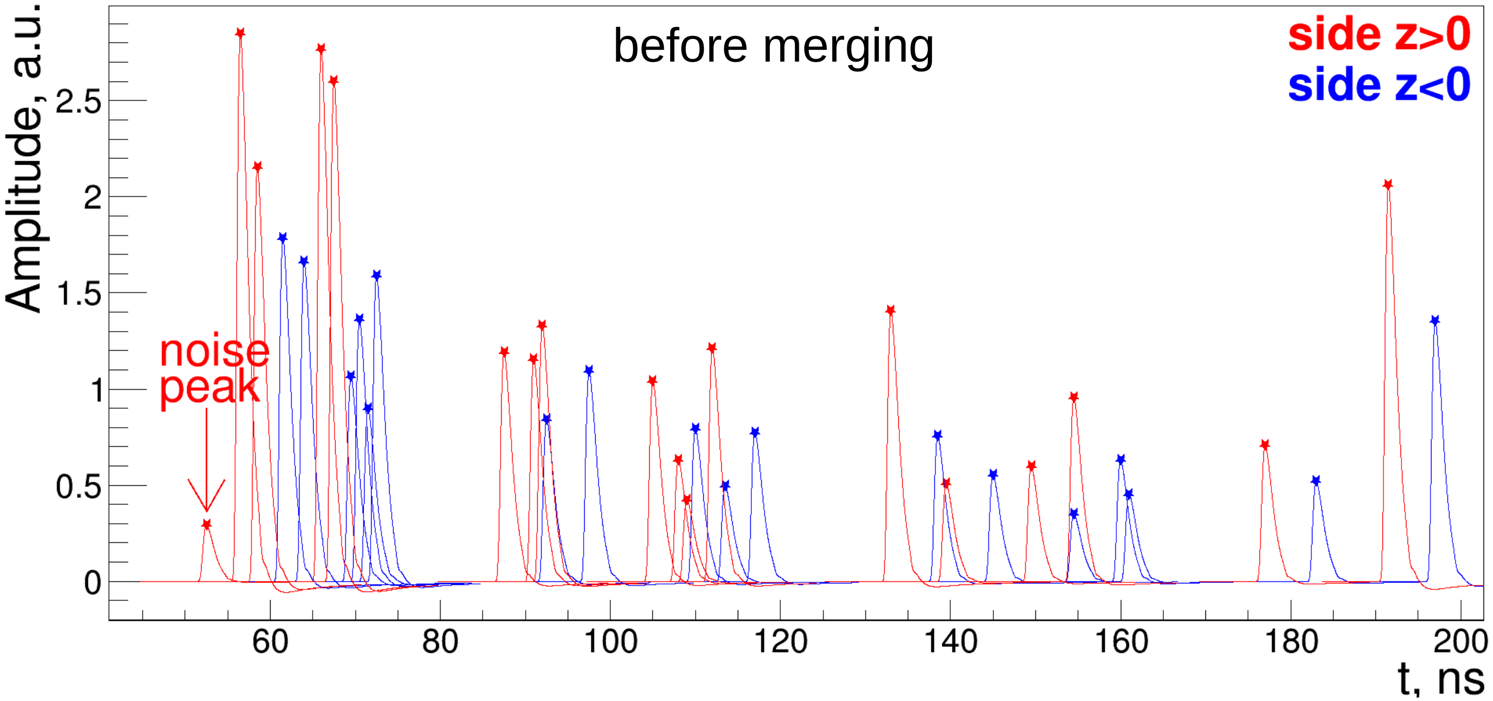


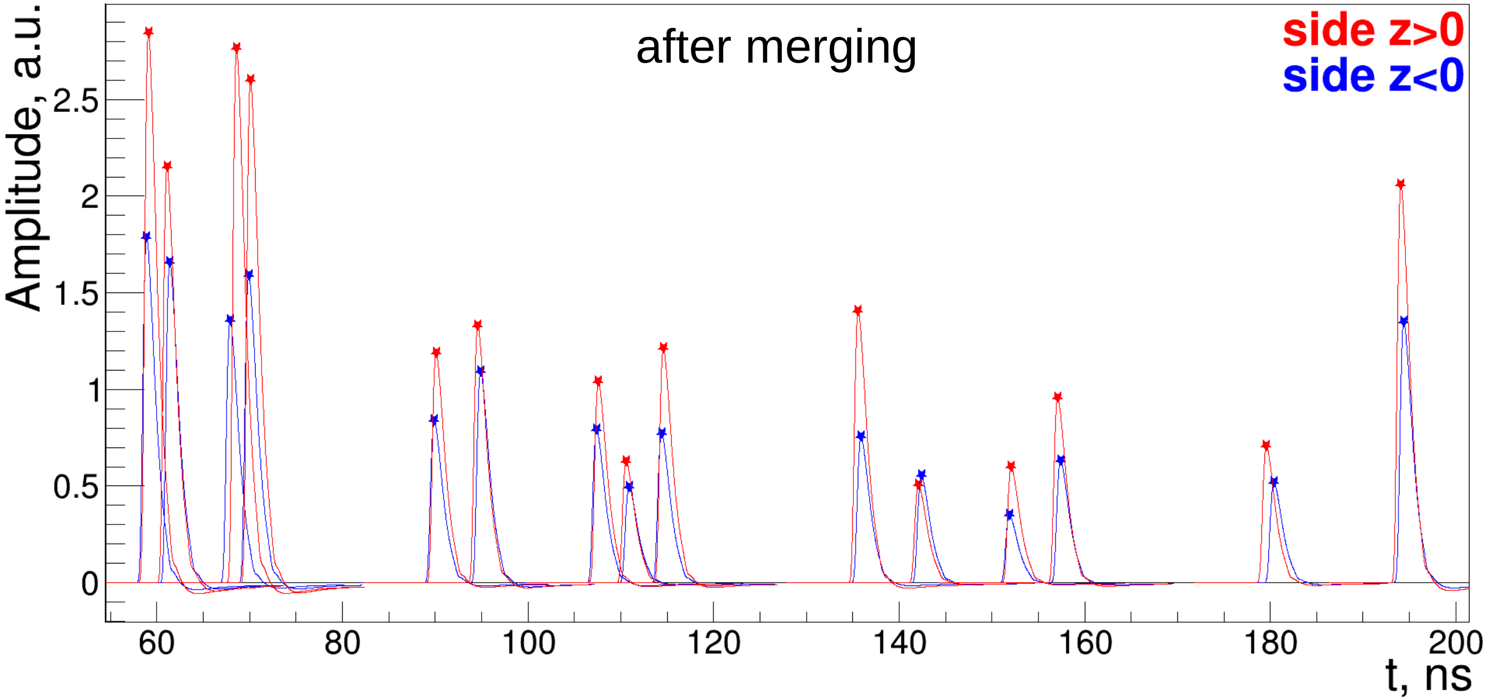
The found “real peaks” (protons)



Merging the waveforms from the wire ends

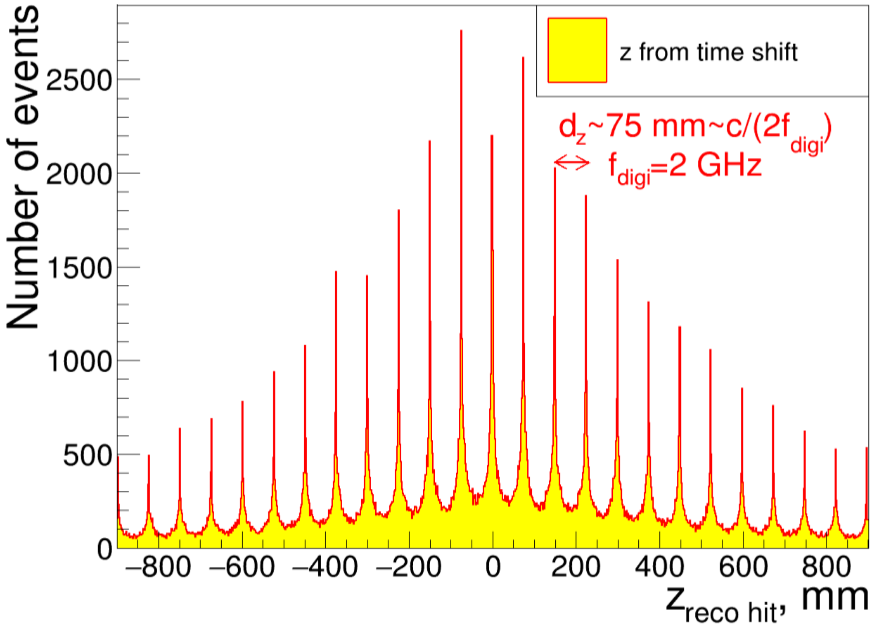
- After the peak finding we merge the waveforms from the wire ends
- This is done by finding the time shift between waveforms via the maximization of their cross-correlation function
- After the alignment of signals in time we find the pairs of peaks with maximum overlap (= dot product) and "merge" them
- The remaining unpaired peaks are considered to be wrong or noise peaks



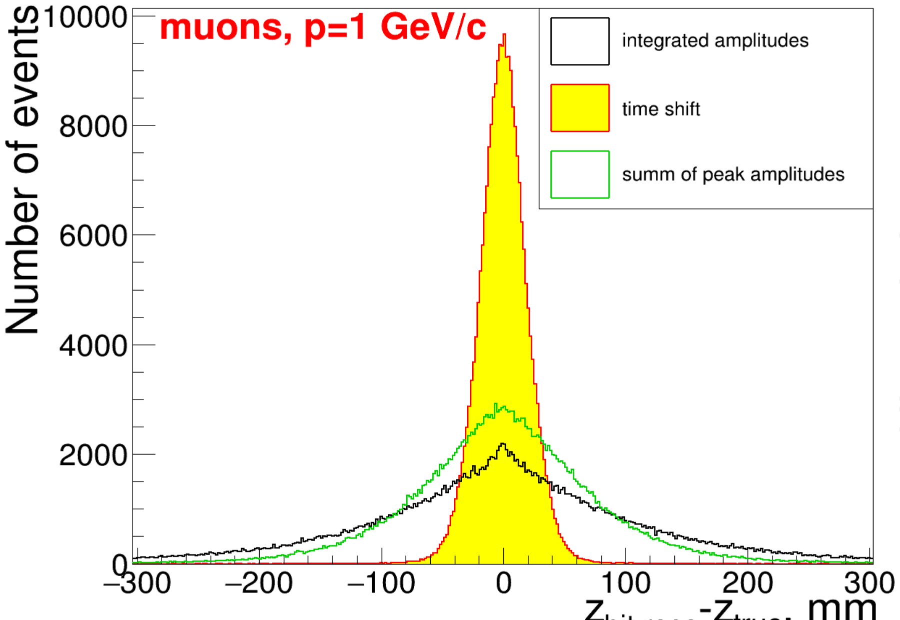


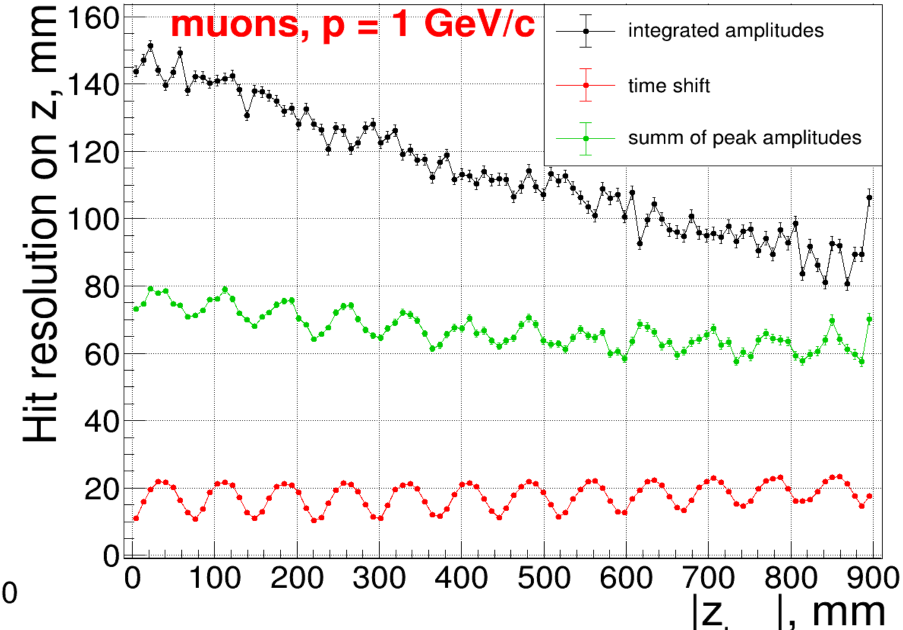
Hit z coordinate measurement

- We can reconstruct the coordinate along the wire via the charge division formula:
  1. Using the **full integrated amplitudes**;
  2. Using the **sum of merged peak amplitudes**.
- Another variant is to use the time shift between waveforms – it gives a better resolution of the order of ~20 mm
- Due to the discreetness of time measurement the z coordinate, reconstructed from the time shift, show a "comb" structure
- Theoretically, the resolution may be further improved if we will save not only the maximums, but also the amplitudes just before and after the maximums. Is it practically possible?



- The discreetness of time measurement also gives a periodic modulation of resolution on z, measured via sum of peak amplitudes: "peak" amplitude is measured not in the actual signal peak, but in different "phase" of the peak, depending periodically on the true z



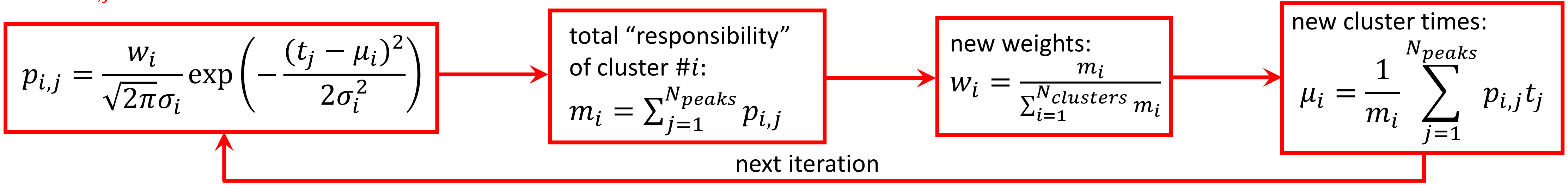


Peaks clusterization algorithm

- The decisive point for the cluster counting and timing is the *possibility* of peaks clusterization

- We consider each merged peak as possible cluster and assign the gaussian to it. The time $\mu_i,\ i = 1,\ldots,N_{clusters}$, of each gaussian is equal to the corresponding peak time $t_j,\ j = 1,\ldots,N_{peaks}$ and sigma equals to the time spread due to diffusion, $\sigma_i = \sigma_{diff}(\mu_i)$). Thus, our model to describe the set of peak times is a gaussian mixture model

- Using the Expectation Maximization (EM) algorithm we iteratively recalculate (until convergence) the positions μ_i of gaussians and their weights w_i using the probabilities $p_{i,j}$ that the peak $\#j$ was produced by the cluster (gaussian) $\#i$:



$$p_{i,j} = \frac{w_i}{\sqrt{2\pi}\sigma_i}\exp\left(-\frac{(t_j-\mu_i)^2}{2\sigma_i^2}\right)$$

total "responsibility" of cluster $\#i$:

$$m_i = \sum_{j=1}^{N_{peaks}} p_{i,j}$$

new weights:

$$w_i = \frac{m_i}{\sum_{i=1}^{N_{clusters}} m_i}$$

new cluster times:

$$\mu_i = \frac{1}{m_i}\sum_{j=1}^{N_{peaks}} p_{i,j}t_j$$

next iteration

- After the convergence was reached, we usually find that some gaussians are "stucked together" (~ "clusterization happened"), some other gaussians have almost zero weights. These effects are the signs of too large model complexity

- To simplify the model, we should choose the cluster to be removed. To make a best choice, we try to remove each one of them and delegate its "responsibility" m_i to its *nearest neighbor*, and calculate the resulting change of the log-likelihood of data description L. We remove the cluster giving the smallest likelihood loss and start clusterization from the very beginning (without removed gauss)

- To find the optimal number of gaussians we use the Akaike information criterion (AIC), which finds the balance between the likelihood of data description and the model complexity

$$L = \sum_{j=1}^{N_{peaks}} \ln\left(\sum_{i=1}^{N_{clusters}} p_{i,j}\right) \qquad AIC = 2N_{clusters} - 2L$$

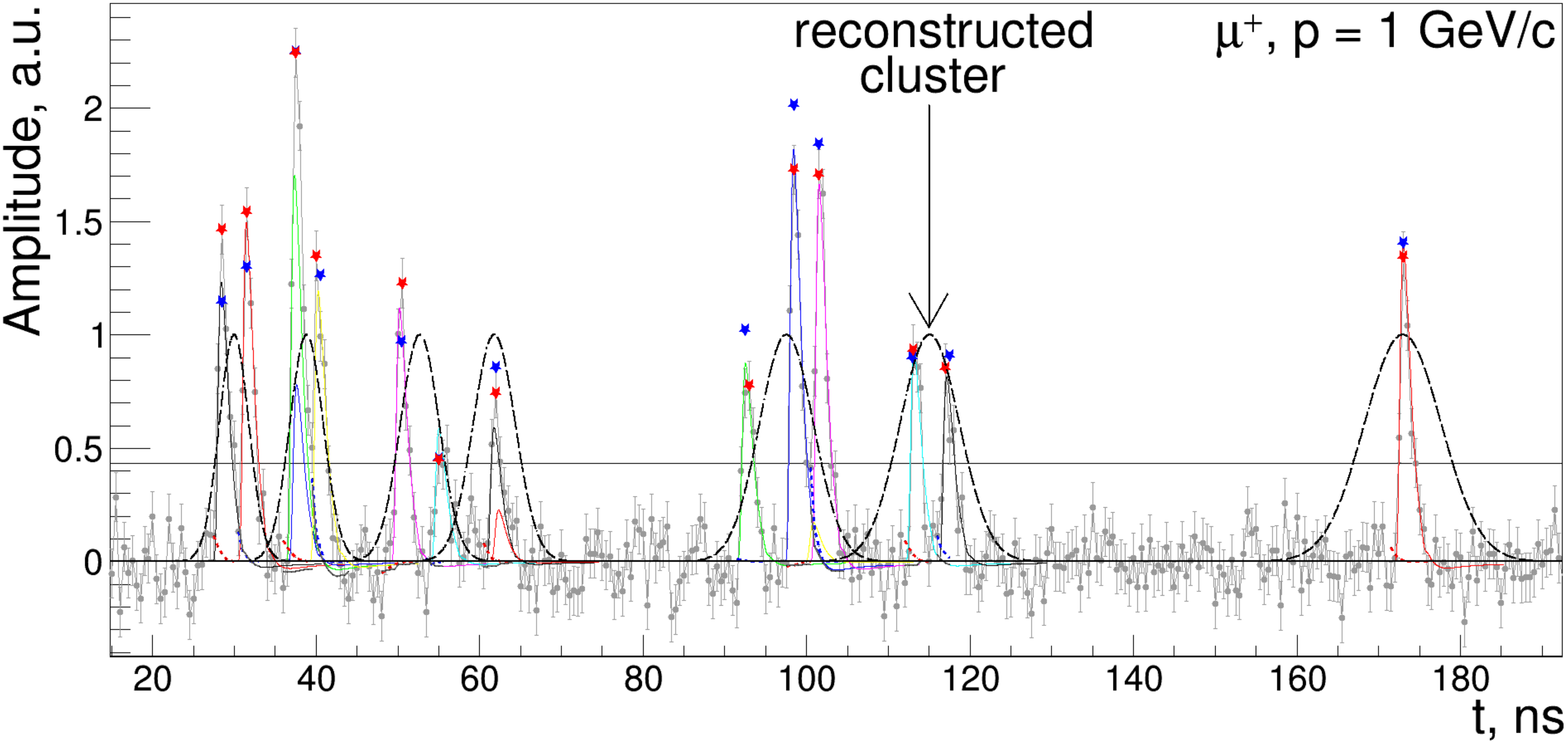
- Current version of algorithm shows poor results for the cluster counting (~50% efficiency for m.i.ps). Improvements are necessary

- Ideas: use the described algorithm for *cluster timing*, and try to estimate the most probable number of clusters within each gaussian using the peak amplitudes (= separate cluster *timing* and cluster *counting* tasks); use the information from all the cells on the track (e.g. the distribution of time separations between peaks);

- **I have a standalone code (only C++ and ROOT) to generate the set of waveforms (~"track"), to find the peaks and to clusterize them. I can share it if anybody would like to try his own ideas for the clusterization algorithm. The joint efforts are necessary, any ideas are welcome!**

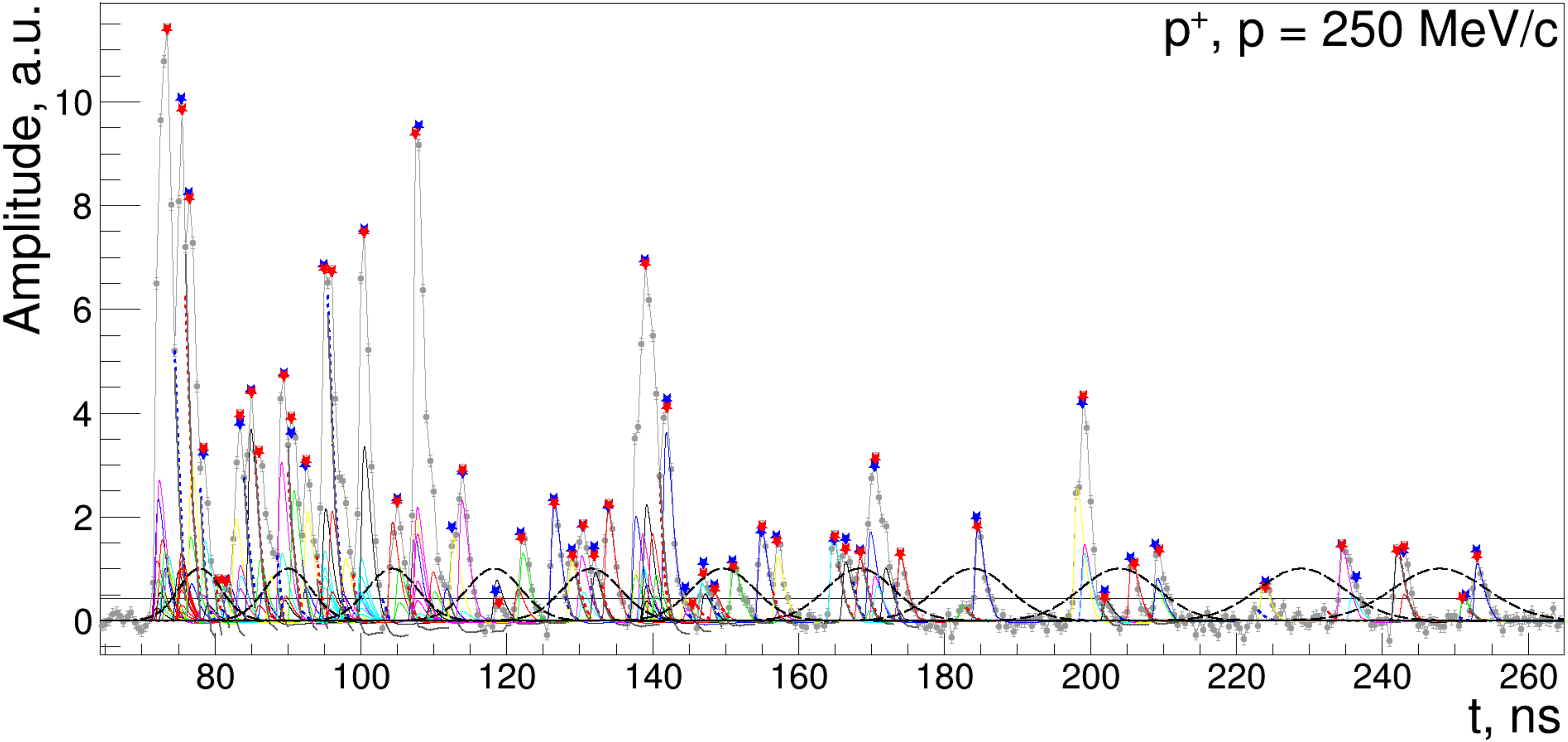
Peaks clusterization algorithm

- Peaks clusterization for muons:



Peaks clusterization algorithm

- Peaks clusterization for protons (p = 250 MeV/c):



Track impact parameter debiasing

- After the peaks clusterization we can use a set of cluster times to reduce the bias of the track impact parameter, caused by the discreetness of ionization
- The Maximum Product of Spacings algorithm has been proposed for this purpose https://doi.org/10.1016/j.nima.2015.11.028
- The idea is to find the impact parameter, which makes a sample of cluster positions along the track most similar to a sample from a uniform distribution. This is achieved by the maximization of the geometrical mean of spacings between clusters:

$$y_i = \frac{\sqrt{\rho_i^2 - b^2}}{\sqrt{(d_{cell}/2)^2 - b^2}}$$

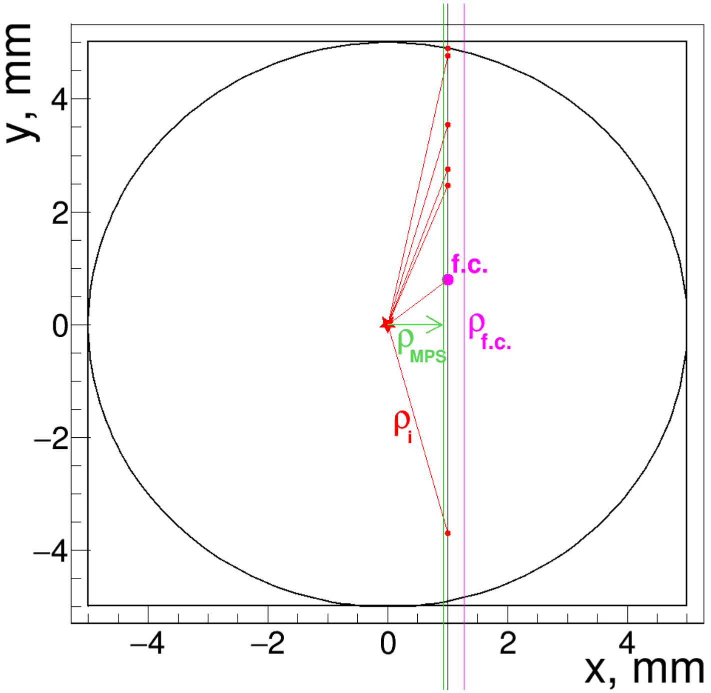
Normalized spacings between clusters:

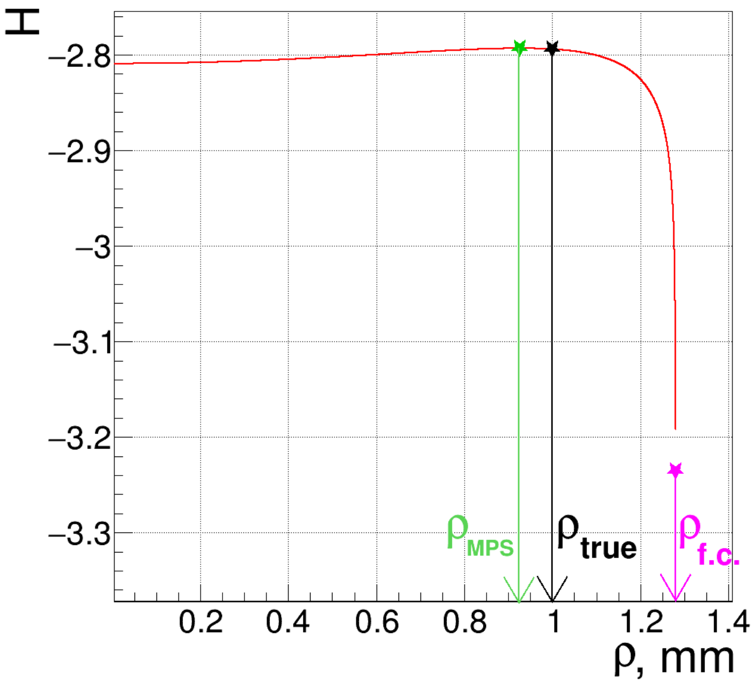
$$D_i = y_i - y_{i-1}$$

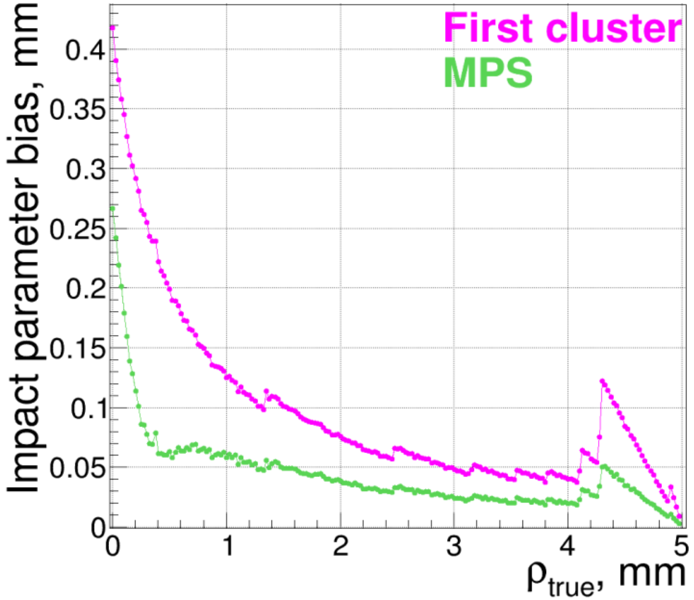
$$H(b) = \frac{1}{N_{clusters} + 1} \sum_{i=1}^{N_{clusters}} \ln(D_i)\,,$$

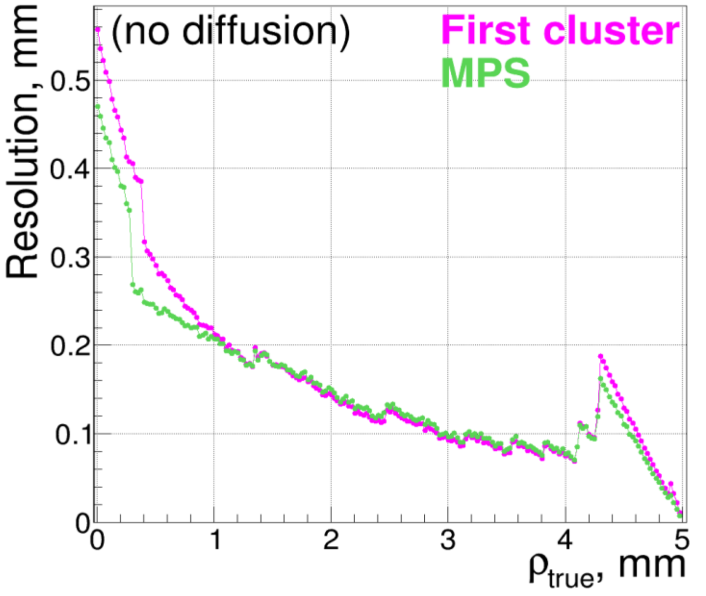
$$\rho_{MPS} = \arg\max H(b)$$

- In the stand-alone implementation of this algorithm we see a serious reduction of impact parameter bias, but a very small effect on resolution



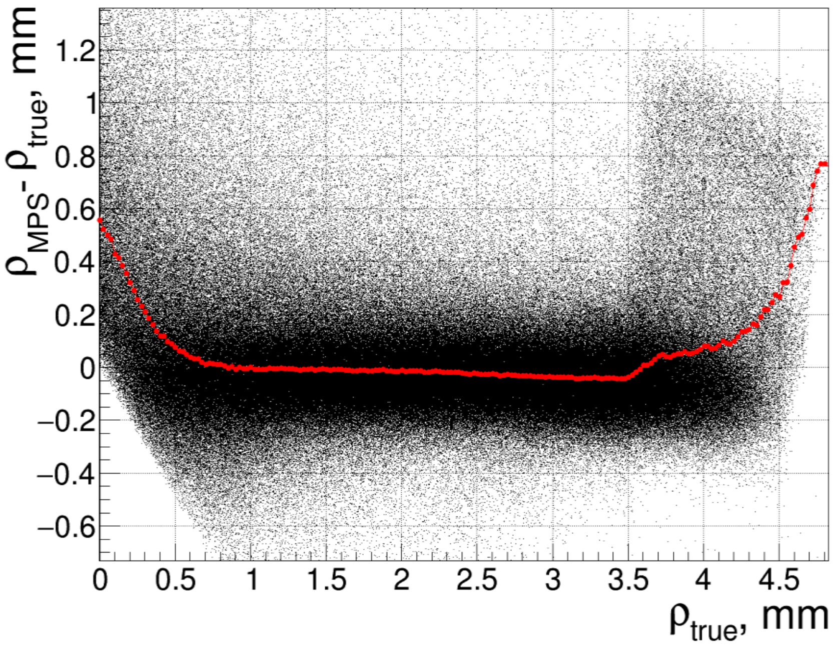


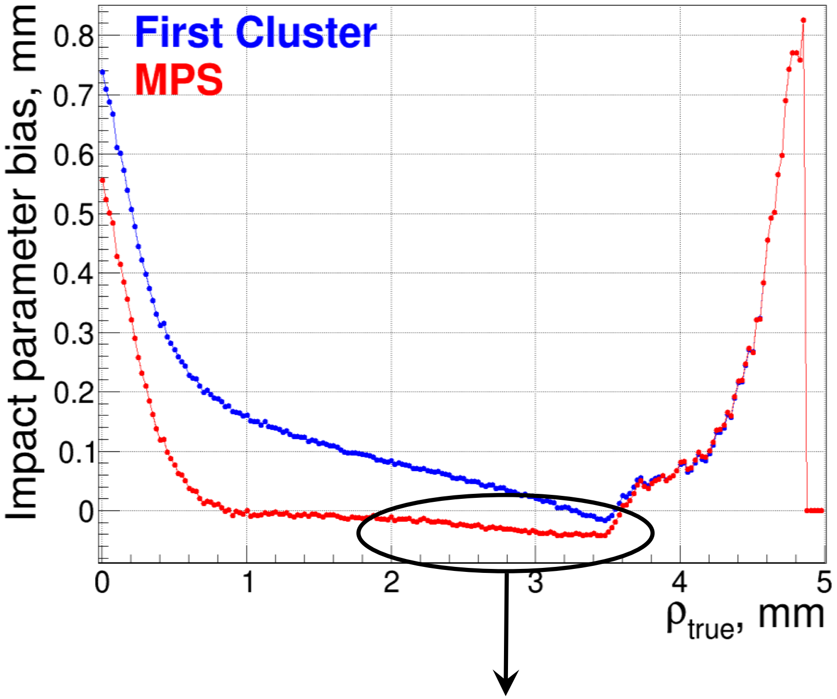




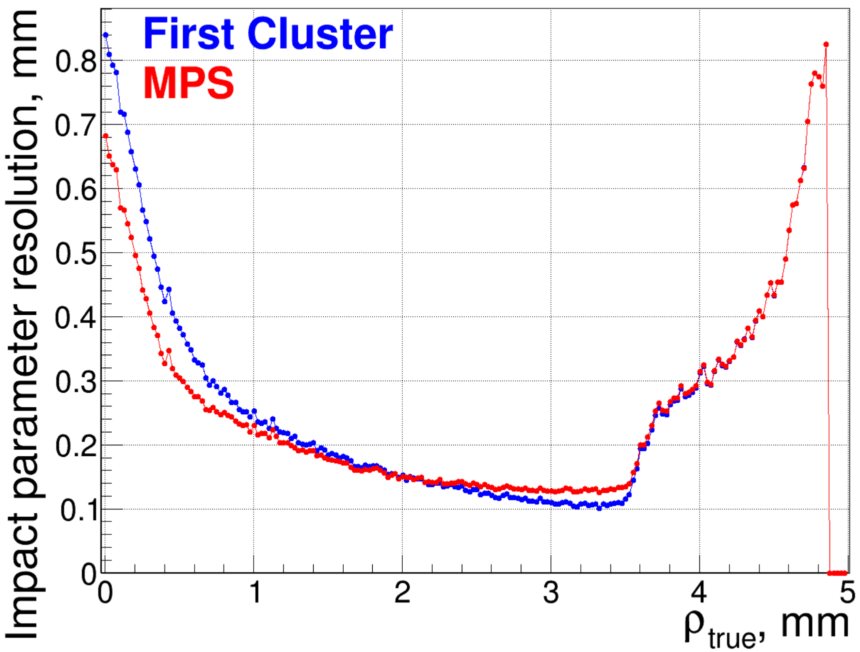
Track impact parameter debiasing

- In the full simulation the debiasing effect is also seen (despite the questionable work of clusterization algorithm)





some negative bias, probably due to circular isochrones assumption in my time-to-rho relations



Particle identification (very preliminary)

- We compare the signal/background separation powers for the following identification variables:

1) $\boldsymbol{dE/dx}$ with the truncation factor $\eta = 0.7$;

2) dN_{peaks}/dx – number of merged peaks is used instead of number of clusters;

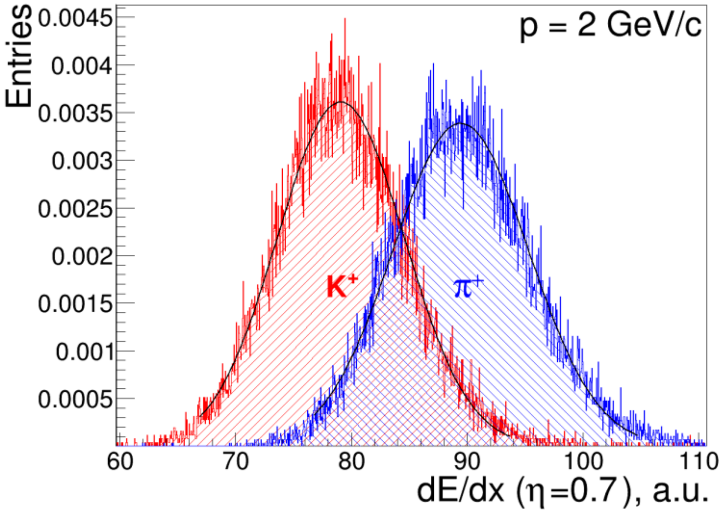
3) $dN_{clust.\,MCTruth}/dx$ – the true number of clusters in each cell is used, to see the theoretical limit of PID efficiency;

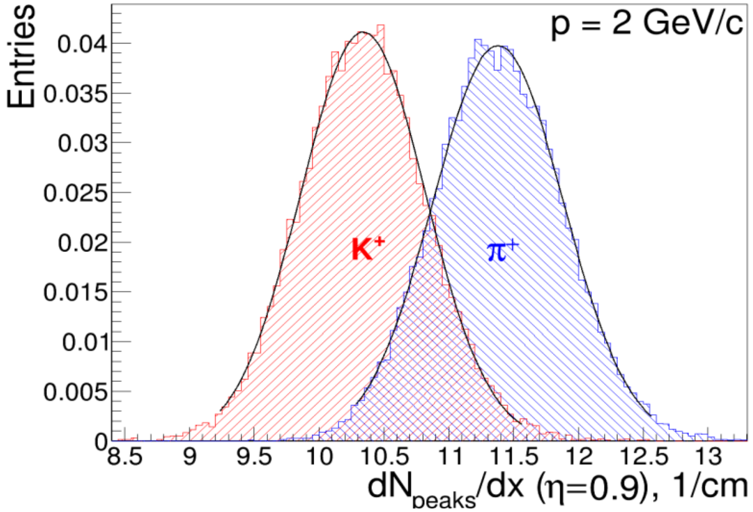
4) $dN_{clust.\,reco}/dx$ – the number of clusters is taken from the clusterization algorithm.

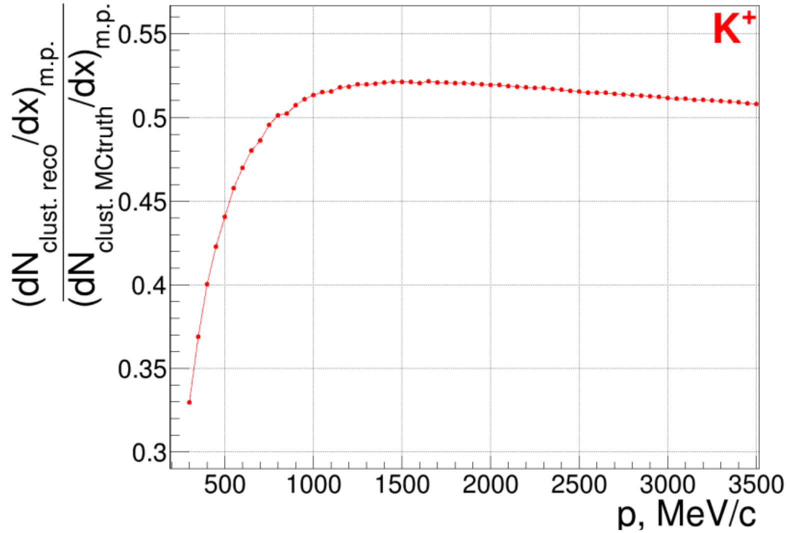
- Separation power: $N_\sigma = \dfrac{|\mu_1 - \mu_2|}{(\sigma_1 + \sigma_2)/2}$

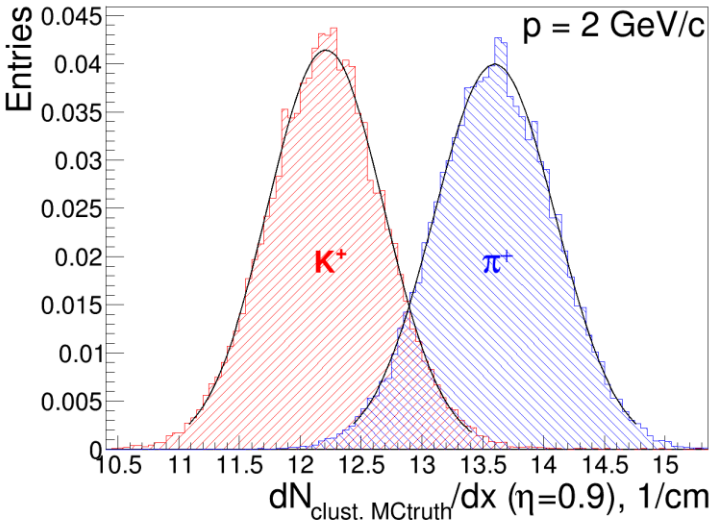
$\mu_{1,2}$ - most probable values

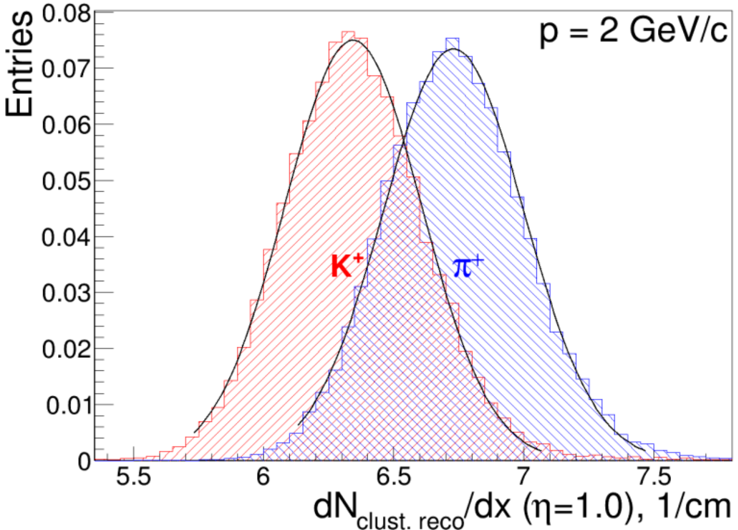
$\sigma_{1,2} = FWHM/\sqrt{2\ln(2)}$

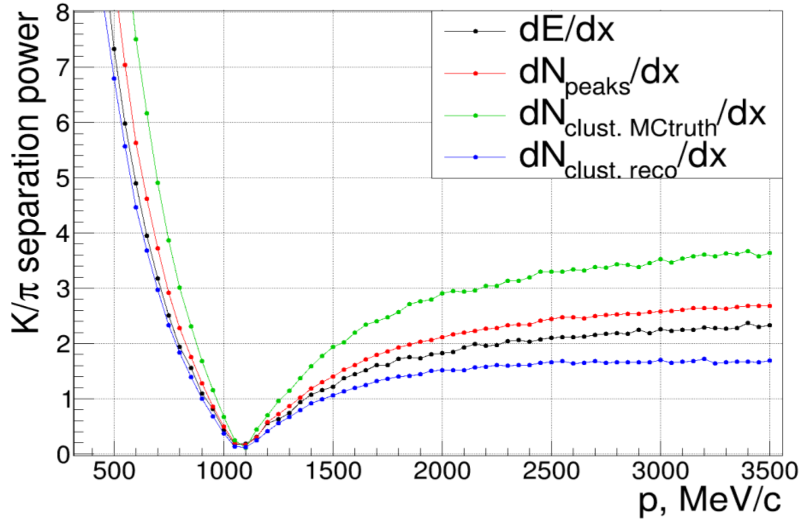




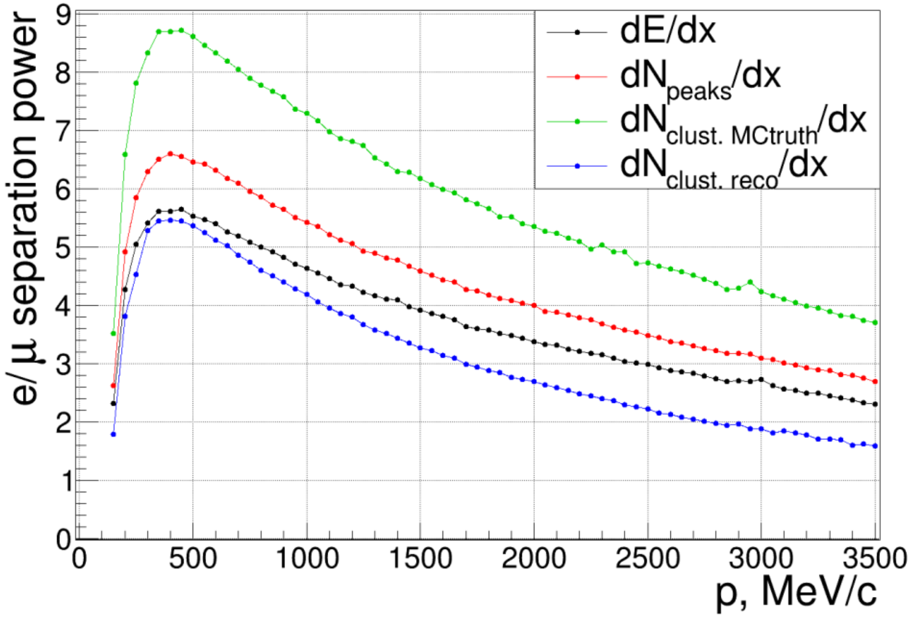


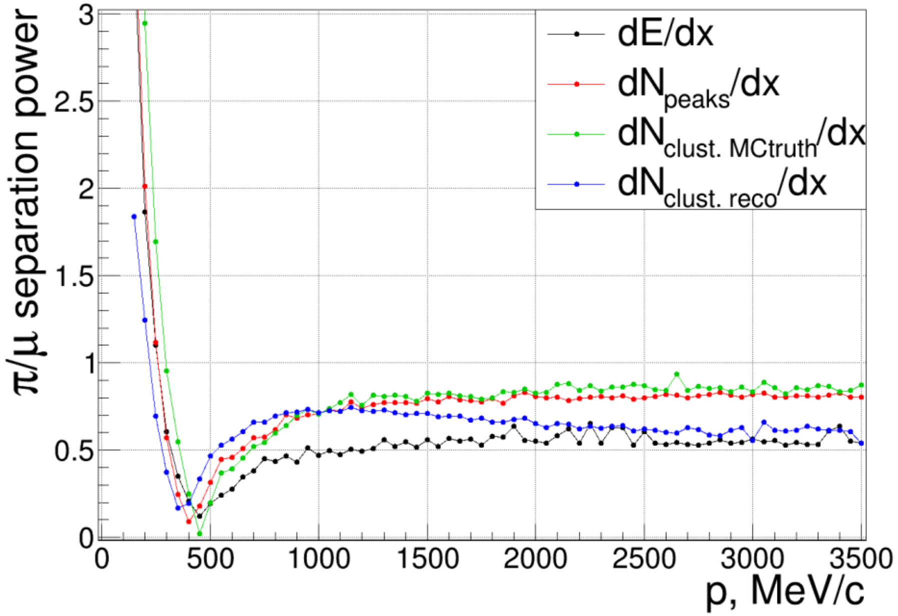


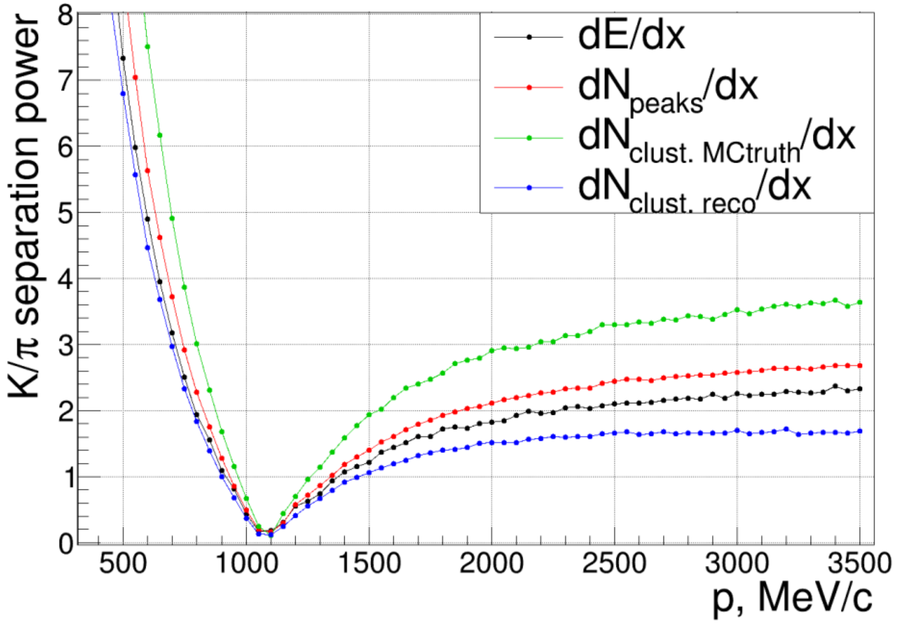


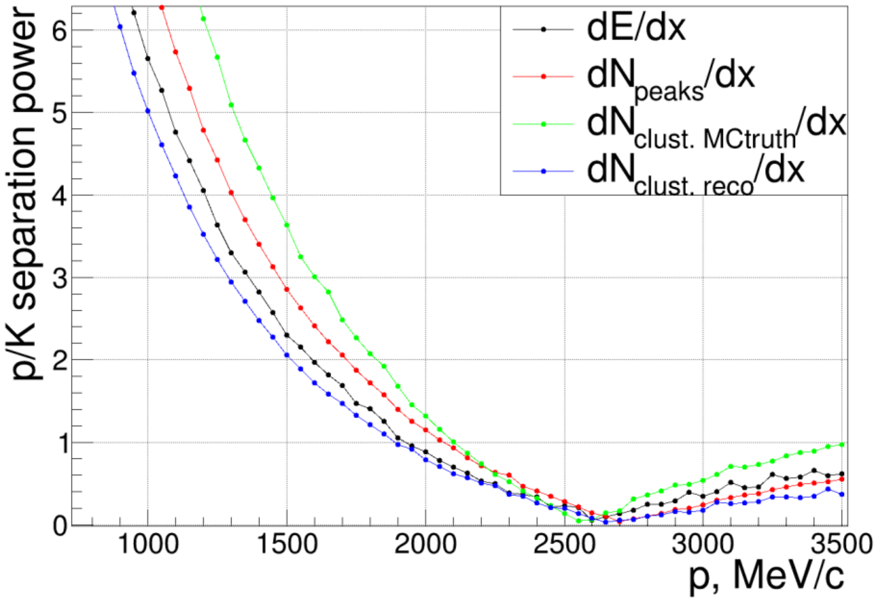


Particle identification (very preliminary)

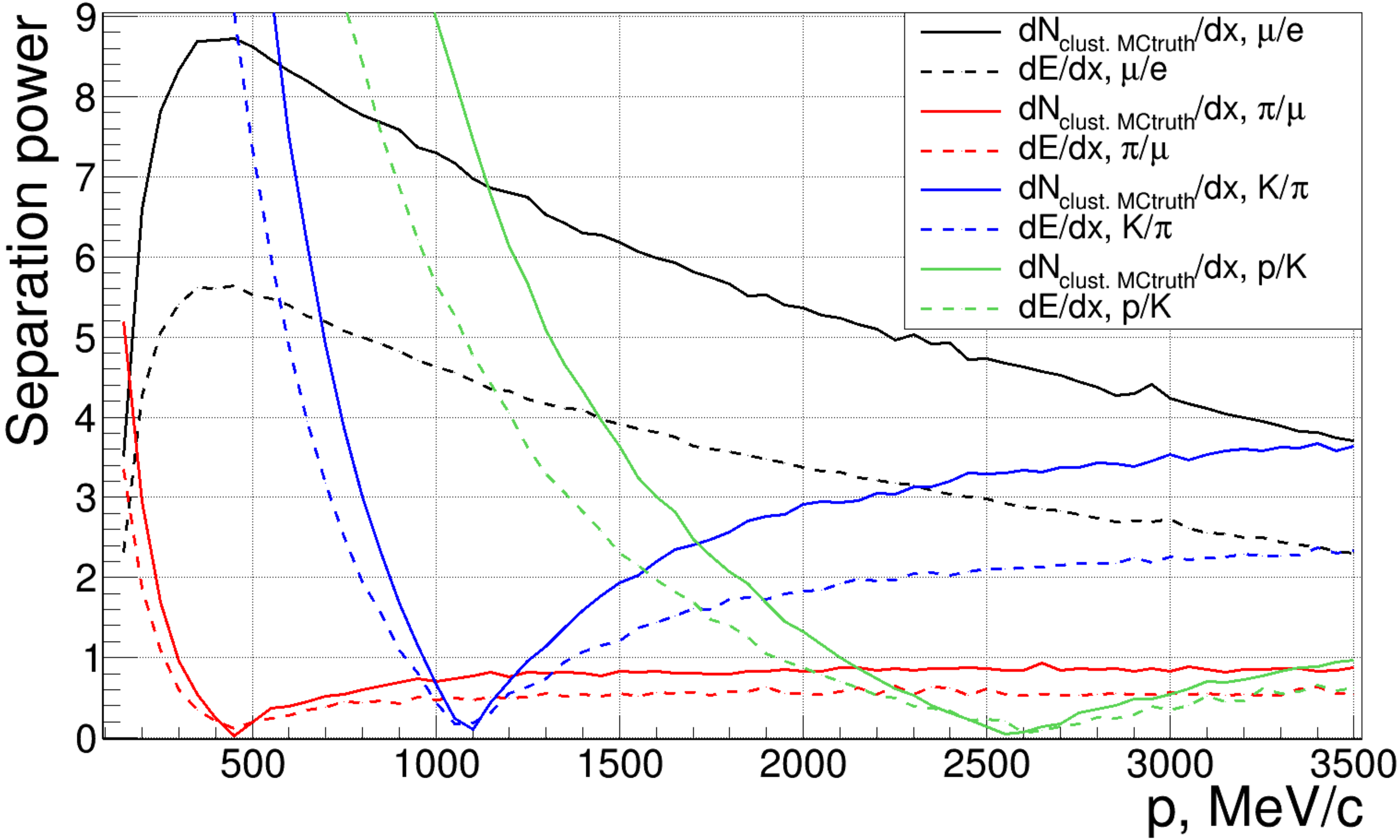








Particle identification (very preliminary)



Plans

- Estimate the effect of impact parameter debiasing on track parameter resolutions (nearest)

- Try to improve the clusterization algorithm

- Understand, if my peak finding algorithm (or its simplified version) is suitable for FPGA, if not – use another one

- Finish the development of full simulation/reconstruction chain, especially track finding algorithm (next half year)

- Push the full simulation chain in the master branch of the Aurora SCTF detector simulation package