Miroslav Morháč
Institute of Physics, Slovak Academy of Sciences, Bratislava, Slovakia

TSpectrum class developments

ROOT 2005 Workshop, September 28-30, CERN Geneva

Introduction

•TSpectrum class of the ROOT system is an efficient tool aimed for the analysis of spectra (histograms) from the experiments in nuclear, high energy physics (possibly other types of data)
•it includes non-conventional processing functions of
- o background estimation, elimination
- o deconvolution – resolution improvement
- o smoothing
- o peak identification
- o fitting
- o orthogonal transforms, filtering, enhancement

Aim of the talk

• to present the set of functions implemented in TSpectrum class
• to explain briefly, the principles of the mathematical methods implemented in TSpectrum class
• to explain the meaning of individual parameters in the calls of TSpectrum functions
• to give few examples of how to use the functions in TSpectrum and illustrating the influence of the parameters
• to outline possible improvements, modifications, and extensions to TSpectrum2, 3

Background estimation

Goal: Separation of useful information (peaks) from useless information (background)

- method is based on Sensitive Nonlinear Iterative Peak (SNIP) clipping algorithm
- new value in the channel “i” is calculated

$$v_p(i) = \min\left\{ v_{p-1}(i), \frac{[v_{p-1}(i+p) + v_{p-1}(i-p)]}{2} \right\}$$

where p = 1, 2, …, number_of_iterations. In fact it represents second order difference filter (-1,2,-1)

Function 1:

const char* Background1(float *spectrum, int size, int number_of_iterations)

Parameters

•spectrum-pointer to the vector of source spectrum
•size-length of the source spectrum
•number_of_iterations or width of the clipping window

This function calculates background spectrum from the source spectrum. The result is placed in the vector pointed by spectrum pointer. On successful completion it returns 0. On error it returns pointer to the string describing error.

Example 1– script Background1.c :

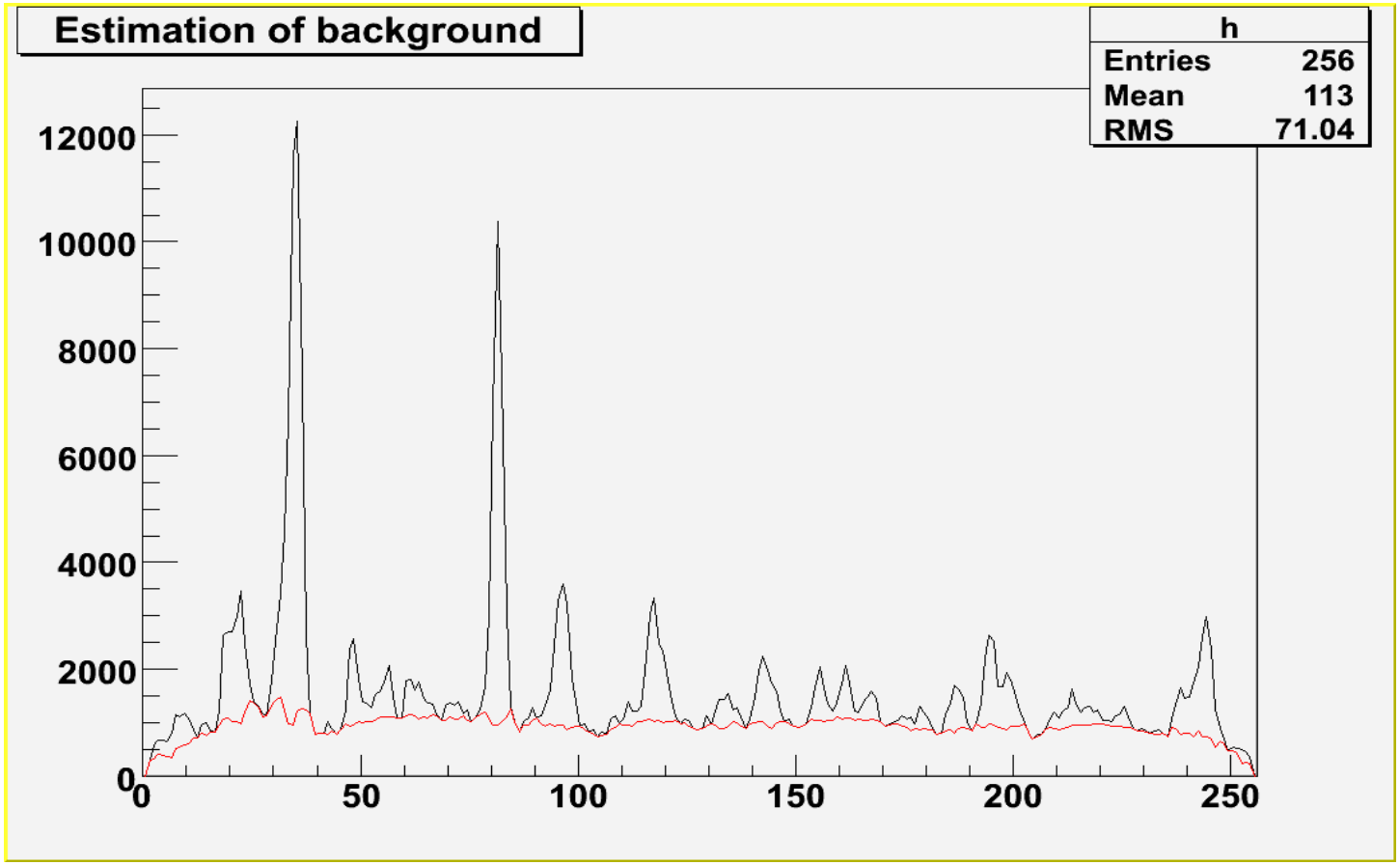


Figure 1 Example of the estimation of background for number_of_iterations=6. Original spectrum is shown in black color, estimated background in red color.

Function 2:

const char* Background1General(float *spectrum, int size,
int number_of_iterations, int direction, int filter_order, bool compton)

Parameters

•spectrum-pointer to the vector of source spectrum
•size-length of the source spectrum
•number_of_iterations or width of the clipping window
•direction- direction of change of clipping window - possible values:
  BACK1_INCREASING_WINDOW
  BACK1_DECREASING_WINDOW
•filter_order-order of clipping filter - possible values:
  BACK1_ORDER2
  BACK1_ORDER4
  BACK1_ORDER6
  BACK1_ORDER8
•compton- logical variable whether the estimation of Compton edge will be included
  BACK1_EXCLUDE_COMPTON
  BACK1_INCLUDE_COMPTON

It represents generalization of the previous function. The meaning of the first three parameters is the same. Moreover one can change the direction of the change of the clipping window, the order of the clipping filter and to include the estimation of Compton edges.

One can notice that in Figure 1 at the edges of the peaks the estimated background goes under the peaks. An alternative approach is to decrease the clipping window from a given value number_of_iterations to the value of one (DECREASING CLIPPING WINDOW) is presented in Example 2.

Example 2 – script Background1General_decr.c :

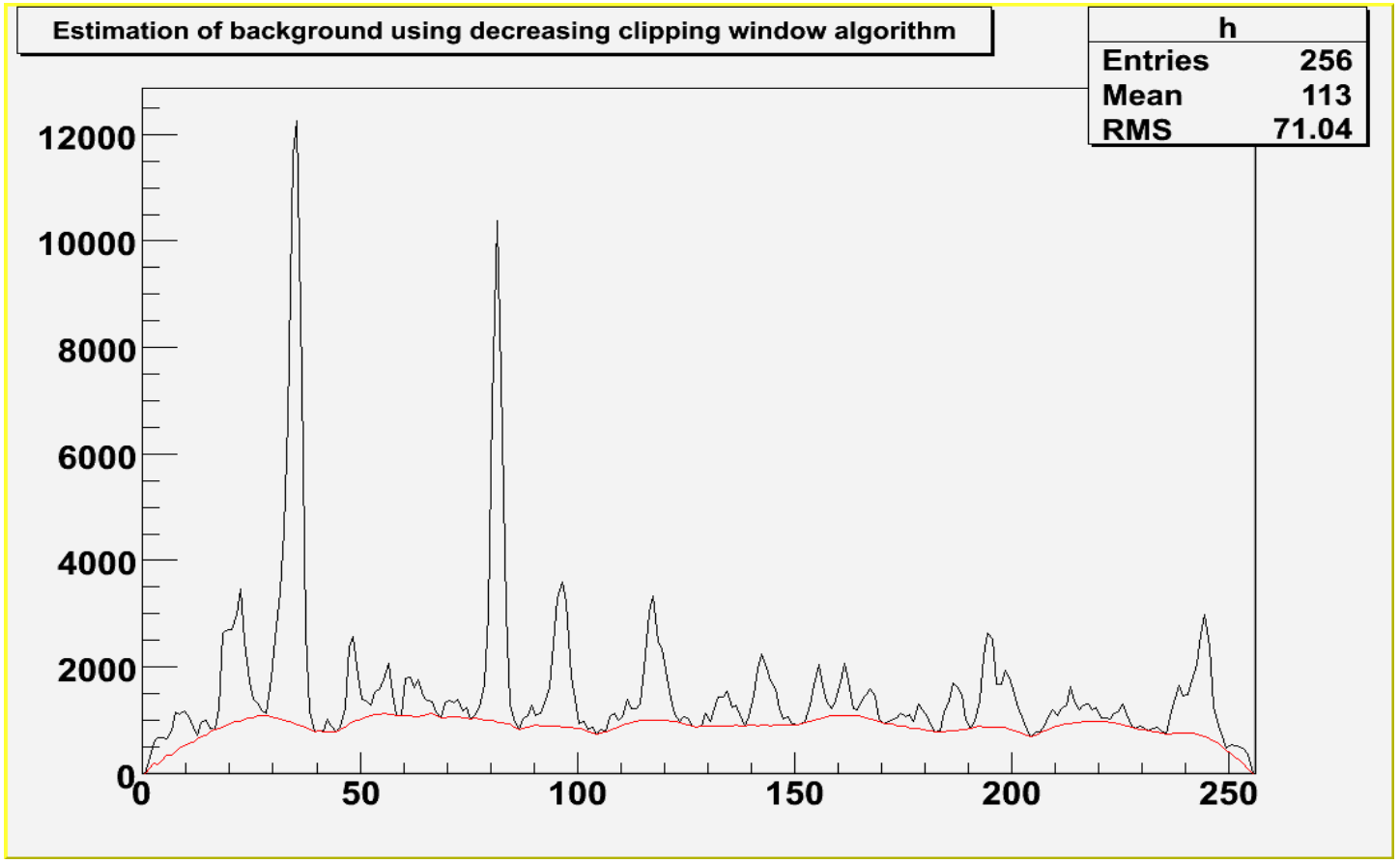


Figure 2 Example of the estimation of background for number_of_iterations=6 using decreasing clipping window algorithm. Original spectrum is shown in black color, estimated background in red color.

•the question is how to choose the width of the clipping window, i.e., number_of_iterations parameter. The influence of this parameter on the estimated background is illustrated in Figure 3.

Example 3 – script Background1General_width.c :

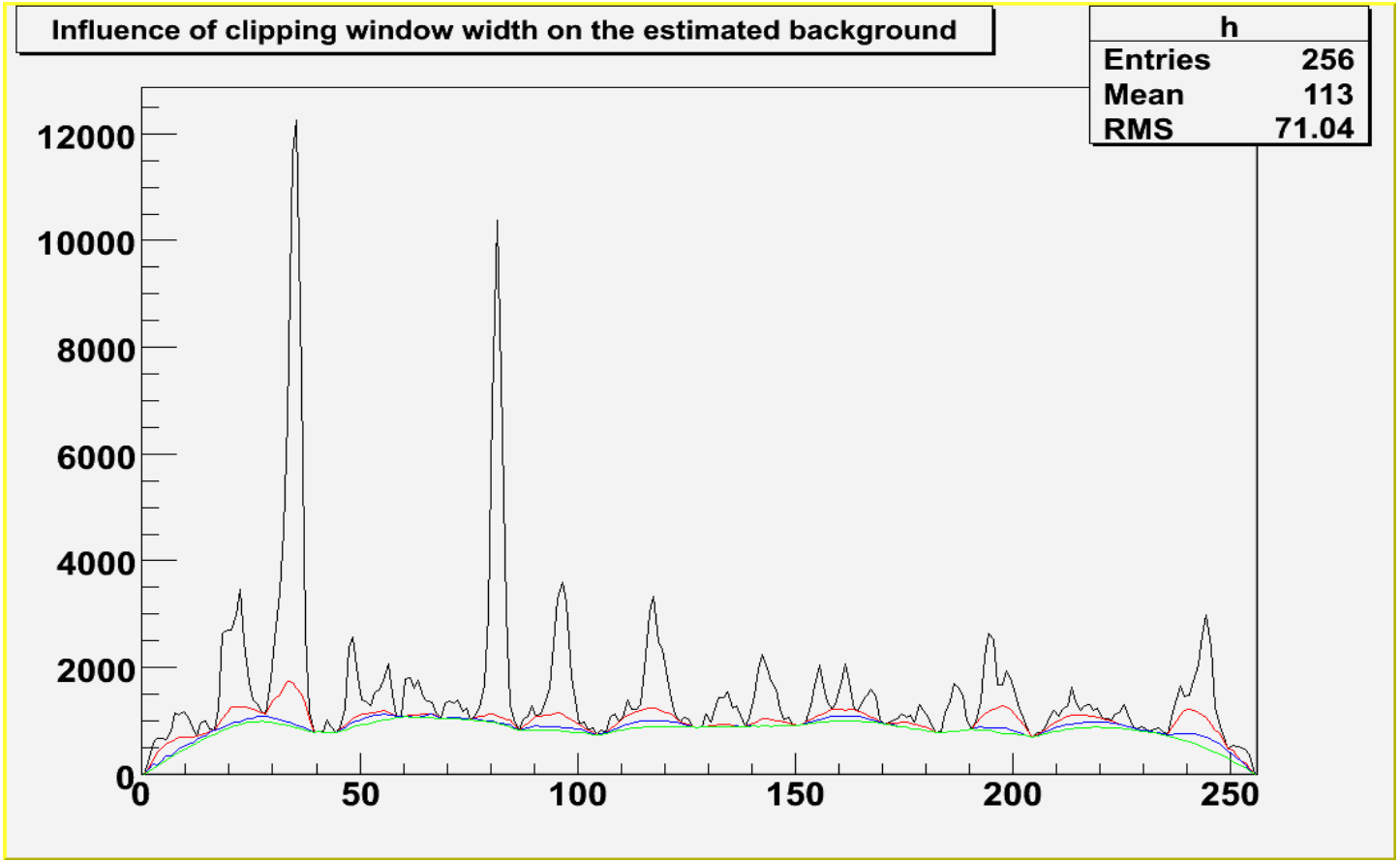


Figure 3 Example of the influence of clipping window width on the estimated background for number_of_iterations=4 (red line), 6 (blue line) 8 (green line) using decreasing clipping window algorithm.

•*in general one should set this parameter so that the value 2*number_of_iterations+1 was greater than the widths of preserved objects (peaks).*

•another example for very complex spectrum Figure 4.

Example 4 – script Background1General_width2.c :

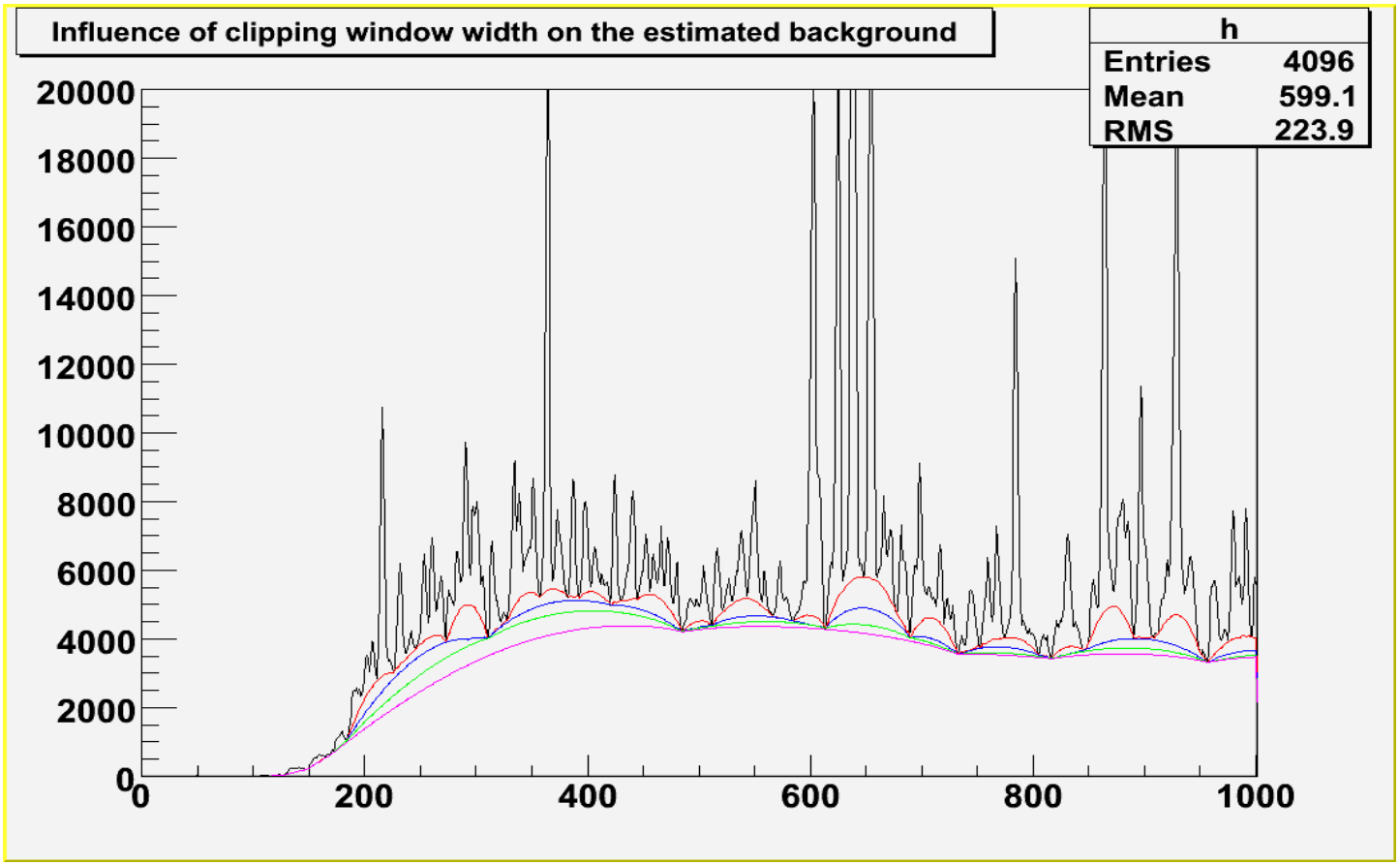


Figure 4 Example of the influence of clipping window width on the estimated background for number_of_iterations=10 (red line), 20 (blue line), 30 (green line) and 40 (magenta line) using decreasing clipping window algorithm.

•second order difference filter removes linear (quasi-linear) background and preserves symmetrical peaks.
•however if the shape of the background is more complex one can employ higher-order clipping filters (see example in Figure 5)

Example 5 – script Background1General_order.c :

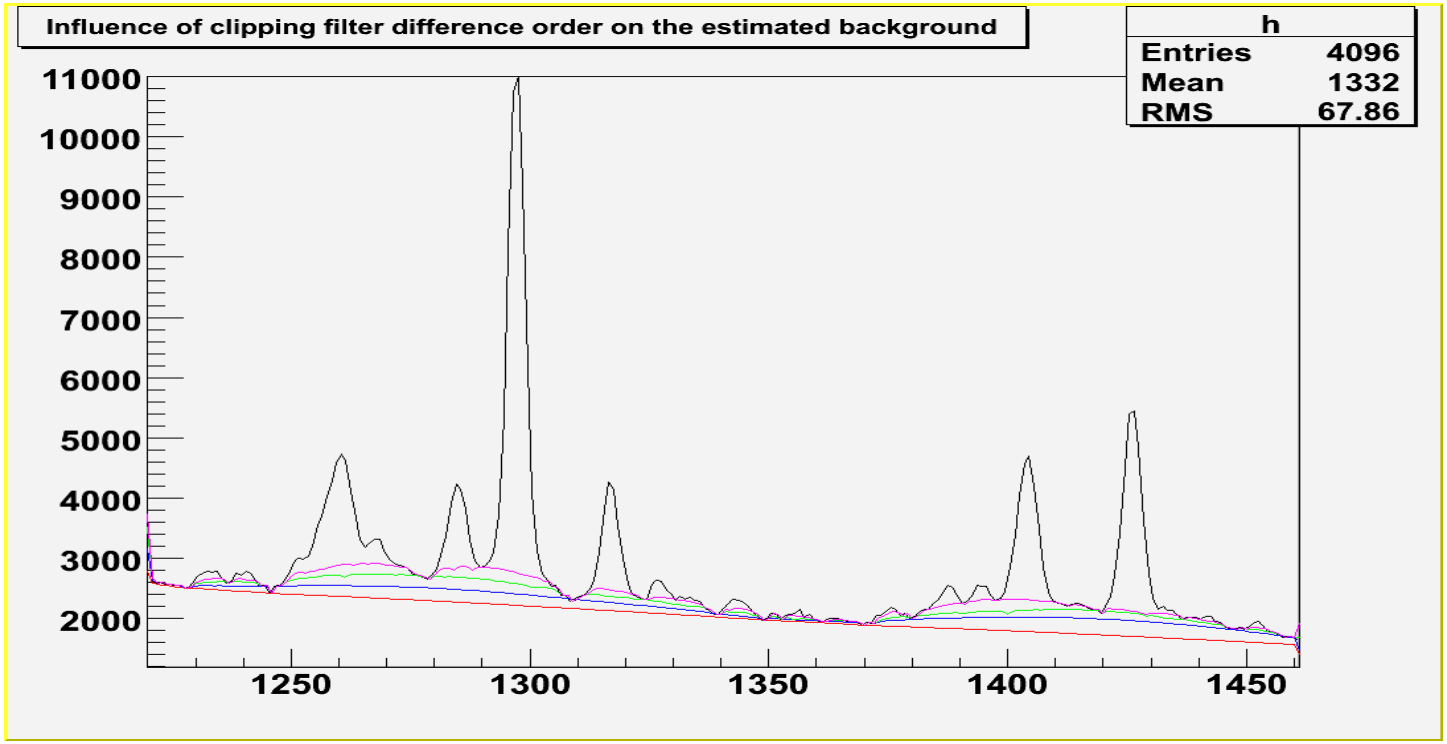


Figure 5 Example of the influence of clipping filter difference order on the estimated background for number_of_iterations=40, 2-nd order red line, 4-th order blue line, 6-th order green line and 8-th order magenta line, and using decreasing clipping window algorithm.

•sometimes it is necessary to include also the Compton edges into the estimate of the background. In Figure 6 we present the example of the synthetic spectrum with Compton edges.
•the background was estimated using the 8-th order filter with the estimation of the Compton edges and decreasing clipping window.

Example 6 – script Background1General_compton.c :

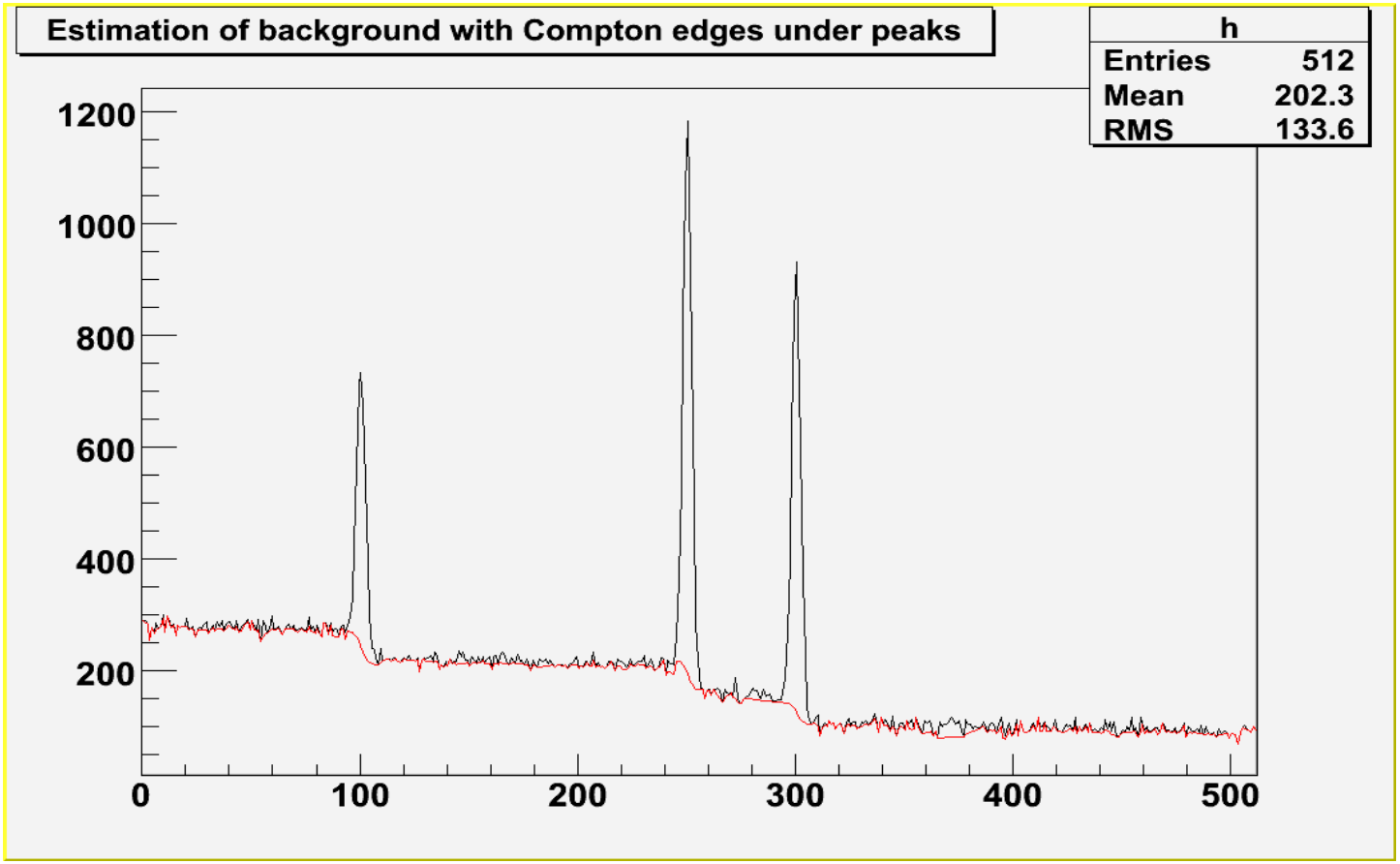


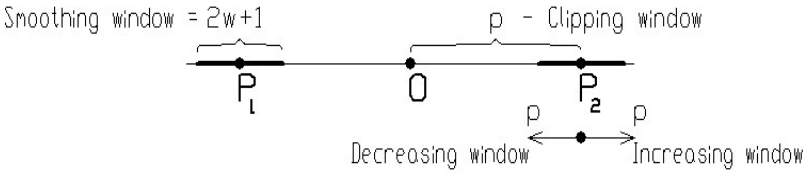
Figure 6 Example of the estimate of the background with Compton edges (red line) for number_of_iterations=40, 8-th order difference filter and using decreasing clipping window algorithm.

References:

[1] C. G Ryan et al.: SNIP, a statistics-sensitive background treatment for the quantitative analysis of PIXE spectra in geoscience applications. NIM, B34 (1988), 396-402.
[2] M. Morháč, J. Kliman, V. Matoušek, M. Veselský, I. Turzo.: Background elimination methods for multidimensional gamma-ray spectra. NIM, A401 (1997) 113-132.
[3] D. D. Burgess, R. J. Tervo: Background estimation for gamma-ray spectroscopy. NIM 214 (1983), 431-434.

Possible extension of TSpectrum background estimation method:

- the estimate of the background can be influenced by noise present in the spectrum. We proposed the algorithm of the background estimate with simultaneous smoothing
- in the original algorithm without smoothing, the estimated background snatches the lower spikes in the noise. Consequently, the areas of peaks are biased by this error.



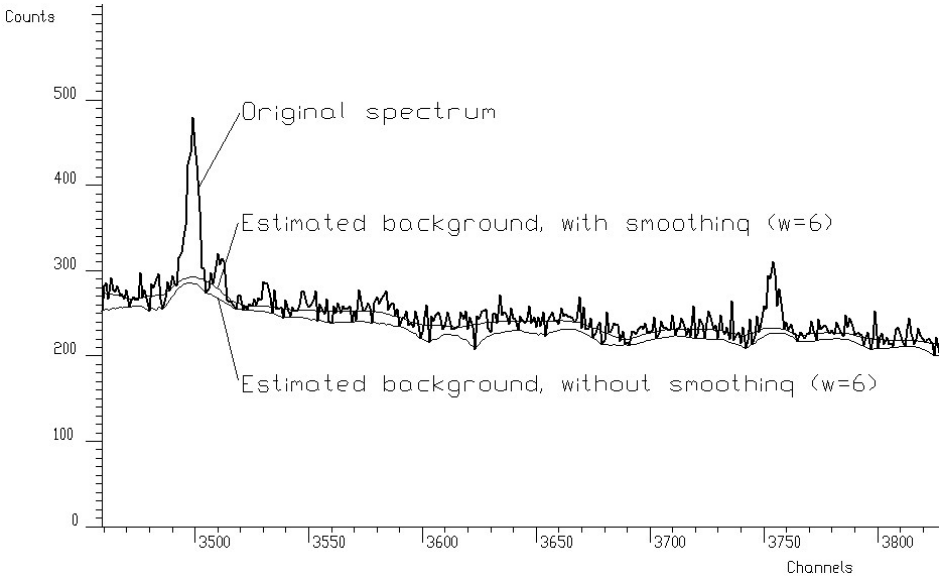


Figure 7 Illustration of non-smoothing and smoothing algorithm of background estimation

Deconvolution

Goal: Improvement of the resolution in spectra, decomposition of multiplets

Mathematical formulation of the convolution system is

$$y(i) = \sum_{k=0}^{N-1} h(i-k)x(k), \quad i = 0,1,2,\ldots\ldots,N-1$$

where h(i) is the impulse response function, x, y are input and output vectors, respectively, N is the length of x and h vectors. In matrix form we have

$$\begin{bmatrix} y(0) \\ y(1) \\ . \\ . \\ . \\ . \\ . \\ . \\ . \\ . \\ y(2N-2) \\ y(2N-1) \end{bmatrix} = \begin{bmatrix} h(0) & 0 & 0 & . & . & . & 0 \\ h(1) & h(0) & 0 & & & & . \\ . & h(1) & h(0) & & & & . \\ . & . & h(1) & & & & . \\ . & . & . & & & & \\ h(N-1) & . & . & & & & 0 \\ 0 & h(N-1) & . & & & & h(0) \\ 0 & 0 & h(N-1) & & & & h(1) \\ . & . & . & & & & . \\ . & . & . & & & & . \\ . & . & . & & & & . \\ 0 & 0 & 0 & . & . & . & h(N-1) \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ . \\ . \\ . \\ x(N-1) \end{bmatrix}$$

•let us assume that we know the response and the output vector (spectrum) of the above given system.
•the deconvolution represents solution of the overdetermined system of linear equations, i.e., the calculation of the vector **x.**
•from numerical stability point of view the operation of deconvolution is extremely critical (ill-posed problem) as well as time consuming operation.
•the Gold deconvolution algorithm proves to work very well, other methods (Fourier, VanCittert etc) oscillate.
•it is suitable to process positive definite data (e.g. histograms).

Gold deconvolution algorithm

$$y = Hx$$

$$H^T y = H^T H x$$

$$H^T H H^T y = H^T H H^T H x$$

$$y' = H' x$$

$$x_i^{(k+1)} = \frac{y_i'}{\sum_{m=0}^{N-1} H'_{im} x_m^{(k)}} x_i^{(k)}, \quad i = 0, 1, \ldots, N-1,$$

$$k = 1, 2, 3, \ldots\ldots\ldots, I,$$

$$x^{(0)} = [1, 1, \ldots\ldots\ldots, 1]^T,$$

where I is given number of iterations

Function 3:

const char* Deconvolution1(float *source, const float *resp, int size, int number_of_iterations)

This function calculates deconvolution from source spectrum according to response spectrum
The result is placed in the vector pointed by source pointer.

Parameters

•source - pointer to the vector of source spectrum
•resp - pointer to the vector of response spectrum
•size - length of the source and response spectra
•number_of_iterations – parameter l in the Gold deconvolution algorithm

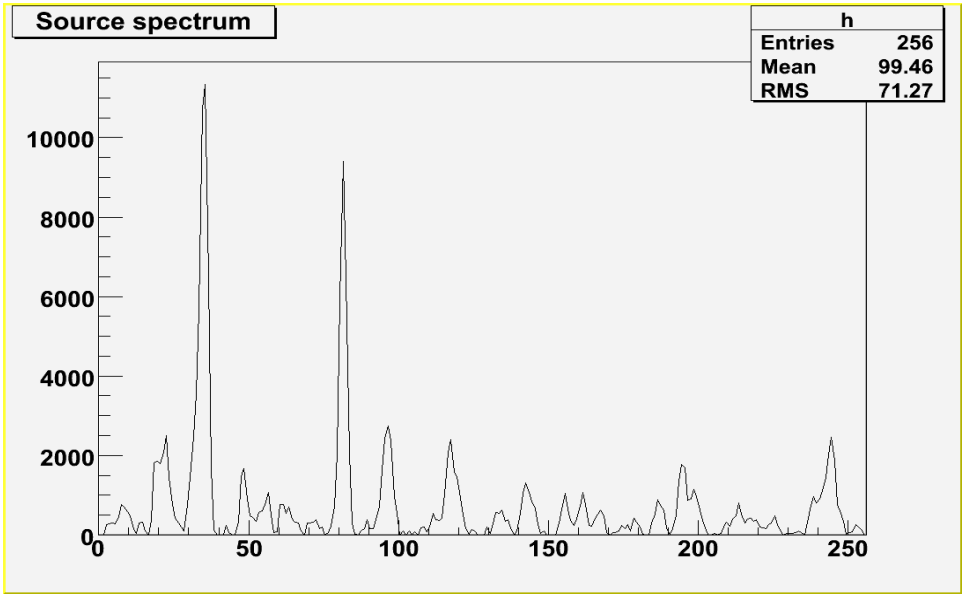


Figure 8 Source spectrum

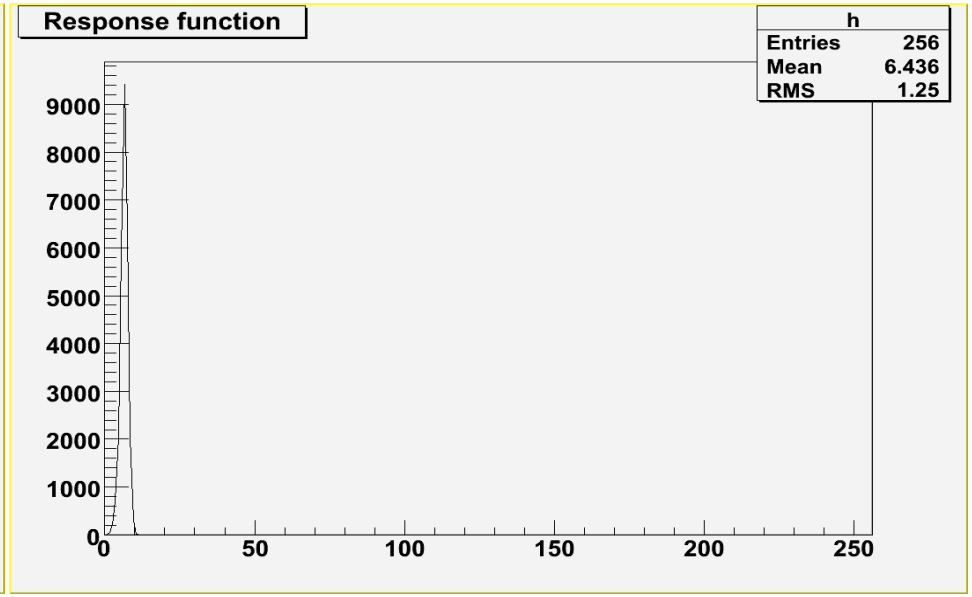


Figure 9 Response spectrum

• response function (usually peak) should be shifted left to the first non-zero channel (bin) (see Figure 9)

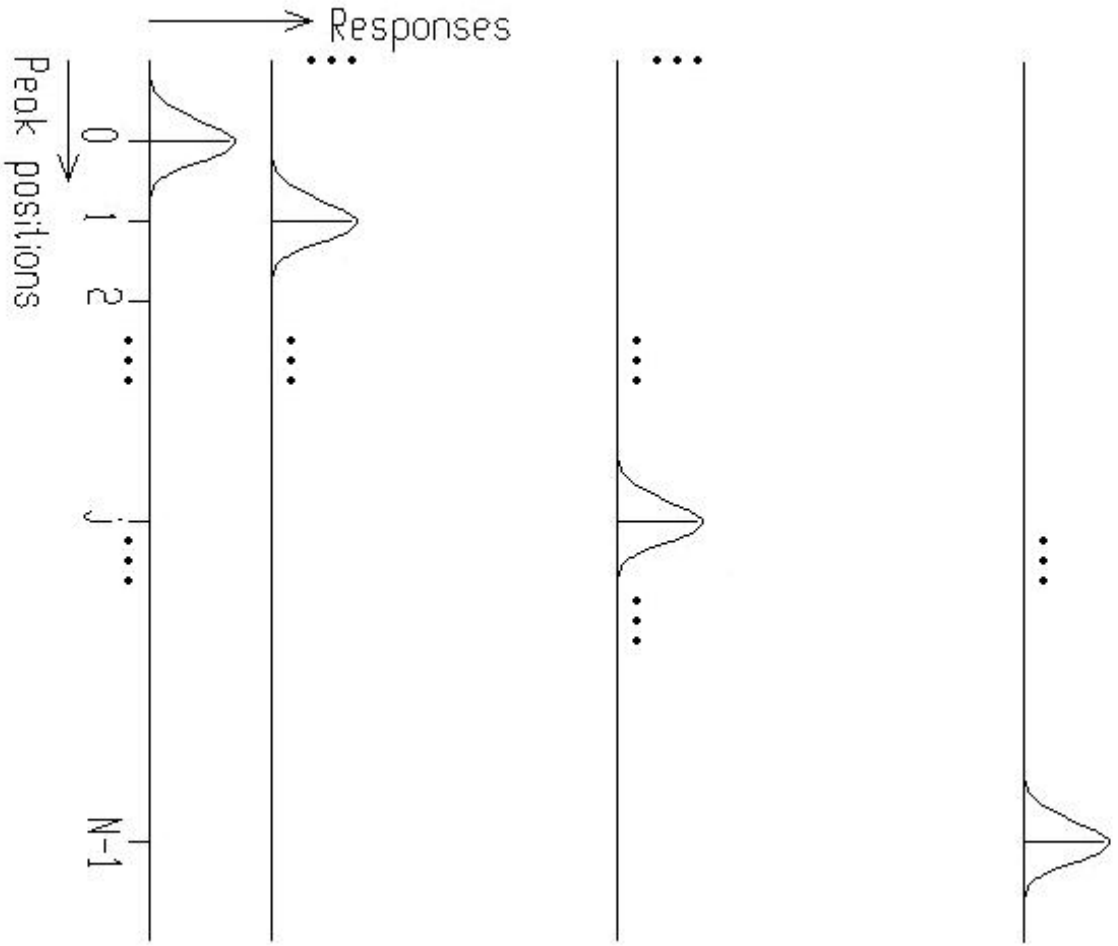


Figure 10 Principle how the response matrix is composed inside of the Deconvolution1 function

Example 7 – script Deconvolution1.c :

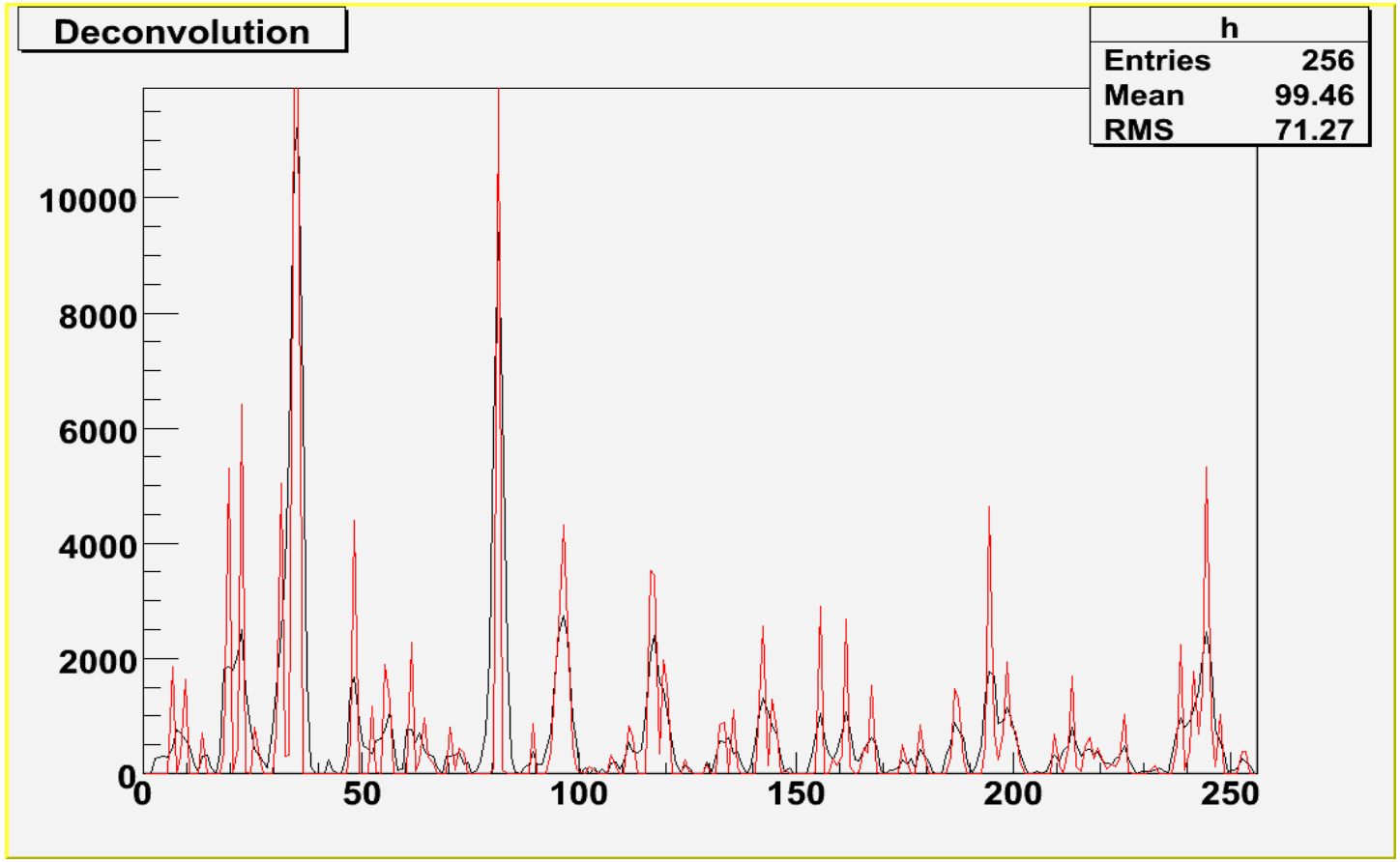


Figure 11 Example of Gold deconvolution. The original source spectrum is drawn with black color, the spectrum after the deconvolution (10000 iterations) with red color,

• further let us study the influence of the number of iterations on the deconvolved spectrum (Figure 12)

Example 8 – script Deconvolution1_iter.c :

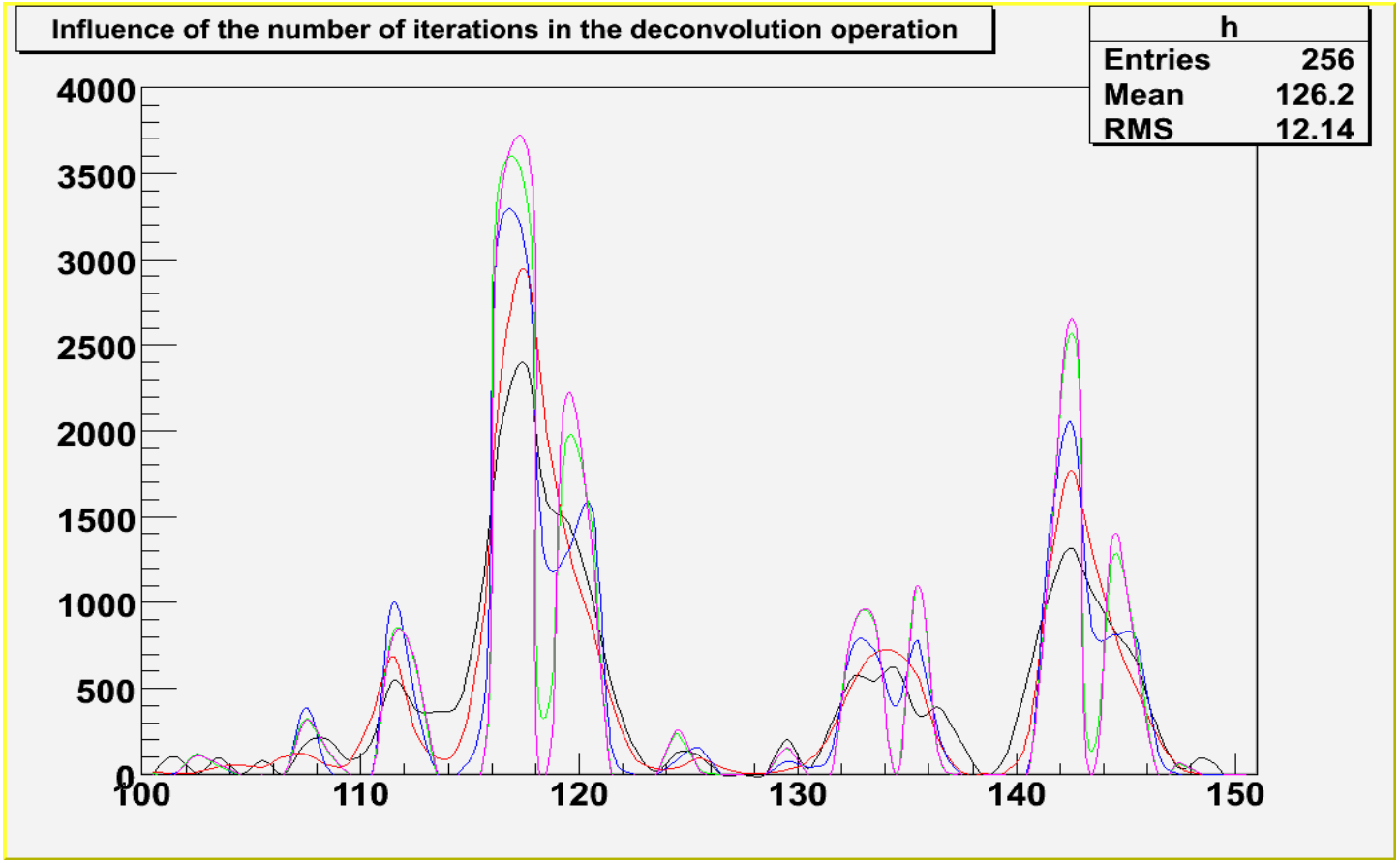


Figure 12 Study of Gold deconvolution algorithm. The original source spectrum is drawn with black color, spectrum after 100 iterations with red color, spectrum after 1000 iterations with blue color, spectrum after 10000 iterations with green color and spectrum after 100000 iterations with magenta color.

Boosted deconvolution

1. Set the initial solution $\boldsymbol{x}^{(0)} = [1,1,...,1]^T$
2. Set required number of repetitions R and iterations L
3. Set the number of repetitions $r = 1$
4. Using Gold deconvolution algorithm for $k = 0,1,..., L-1$ find solution $\boldsymbol{x}^{(L)}$
5. If $r = R$ stop calculation, else
   a.apply boosting operation, i.e., set

$$x^{(0)}(i) = \left[x^{(L)}(i) \right]^p; \quad i = 0,1,..., N-1$$

   and p is boosting coefficient >0

   b. $r = r + 1$

   c. continue in 4.

Function 4:

const char* Deconvolution1HighResolution(float *source, const float *resp, int size, int number_of_iterations, int number_of_repetitions, double boost)

This function calculates deconvolution from source spectrum according to response spectrum using boosted Gold deconvolution algorithm. The result is placed in the vector pointed by source pointer.

Parameters

•source - pointer to the vector of source spectrum
•resp - pointer to the vector of response spectrum
•size - length of the source and response spectra
•number_of_iterations - parameter L in the boosted Gold deconvolution algorithm
•number_of_repetitions - parameter R in the boosted Gold deconvolution algorithm
•boost – boosting coefficient p (should be >0, recommended range <1, 2>)

Example 9 – script Deconvolution1_hr.c :

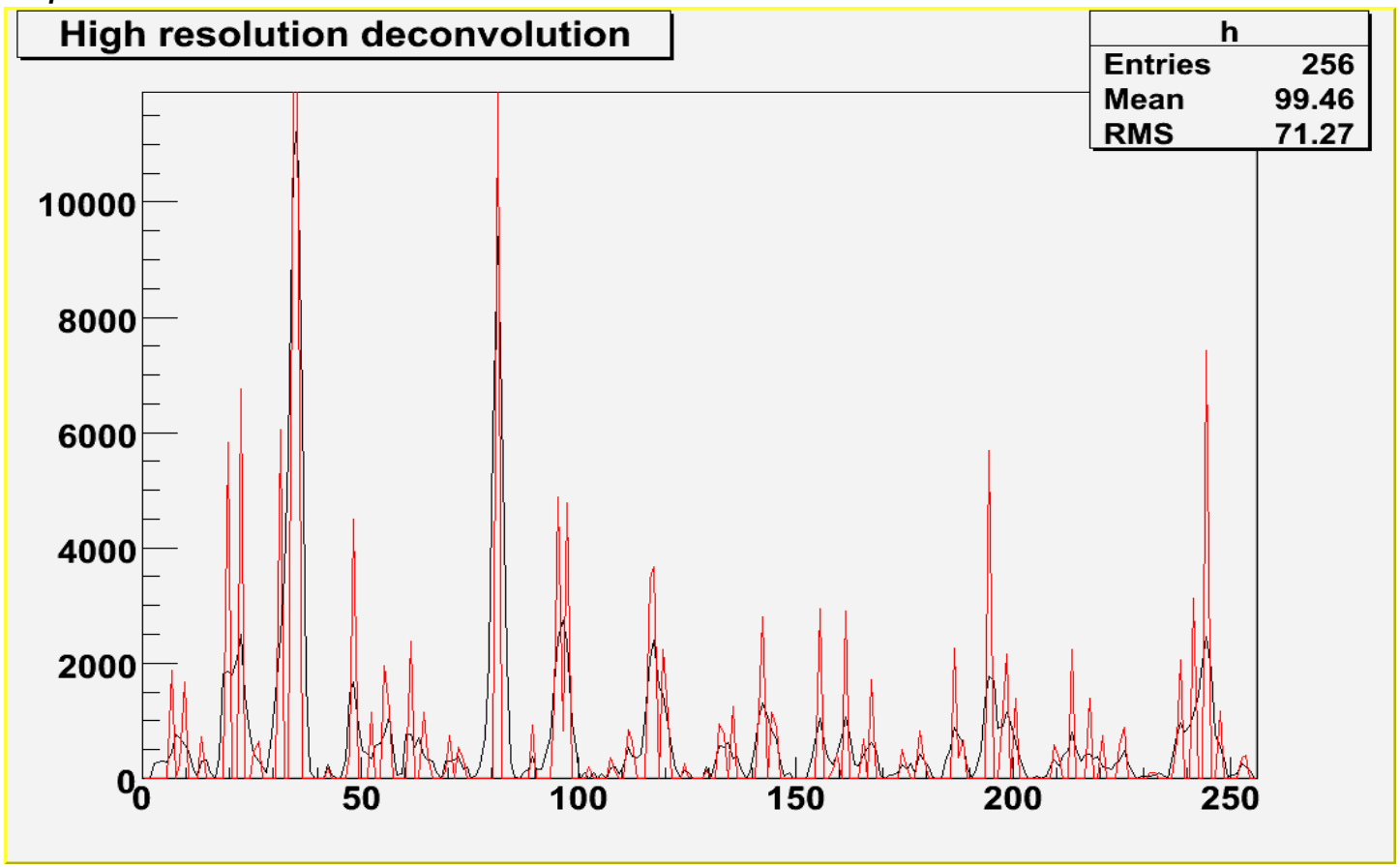


Figure 13 Example of boosted Gold deconvolution. The original source spectrum is drawn with black color, deconvolved spectrum (200 iterations, 50 repetitions, p=1.2) with red color.

•for relatively narrow peaks in the above given example the Gold deconvolution method combined with boosting operation (Deconvolution1HighResolution function) is able to decompose overlapping peaks practically to delta - functions.

•in the next example we have chosen a synthetic data (spectrum, 256 channels) consisting of 5 very closely positioned, relatively wide peaks (sigma =5), with added noise (Figure 14).

•thin lines represent pure Gaussians (see Table 1); thick line is a resulting spectrum with additive noise (10% of the amplitude of small peaks).

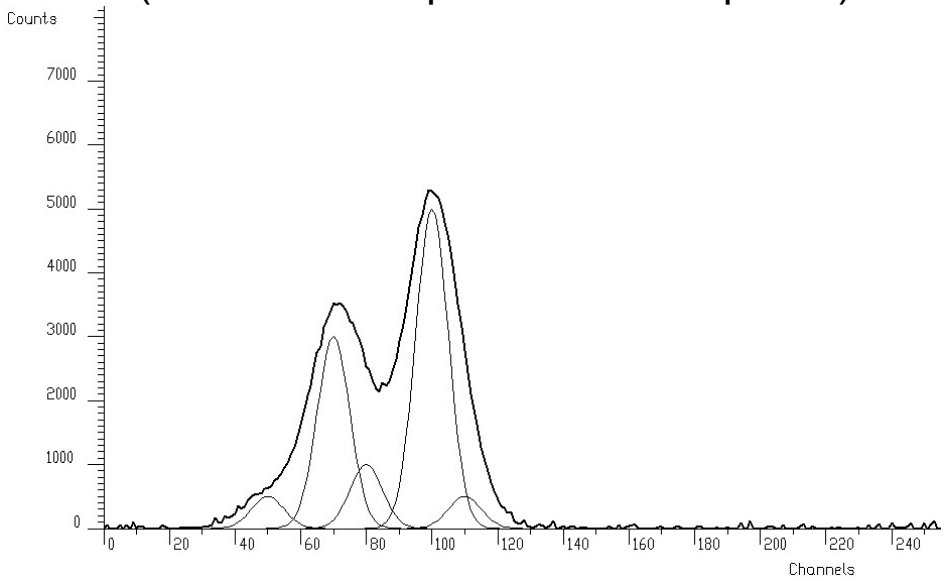


Figure 14 Testing example of synthetic spectrum composed of 5 Gaussians with added noise

Peak #	Position	Height	Area
1	50	500	10159
2	70	3000	60957
3	80	1000	20319
4	100	5000	101596
5	110	500	10159

Table 1 Positions, heights and areas of peaks in the spectrum shown in Figure 14

• in ideal case, we should obtain the result given in Figure 15. The areas of the Gaussian components of the spectrum are concentrated completely to delta -functions

• when solving the overdetermined system of linear equations with data from Figure 14 in the sense of minimum least squares criterion without any regularization we obtain the result with large oscillations (Figure 16).

• from mathematical point of view, it is the optimal solution in the unconstrained space of independent variables. From physical point of view we are interested only in a meaningful solution.

• therefore, we have to employ regularization techniques (e.g. Gold deconvolution) and/or to confine the space of allowed solutions to subspace of positive solutions.

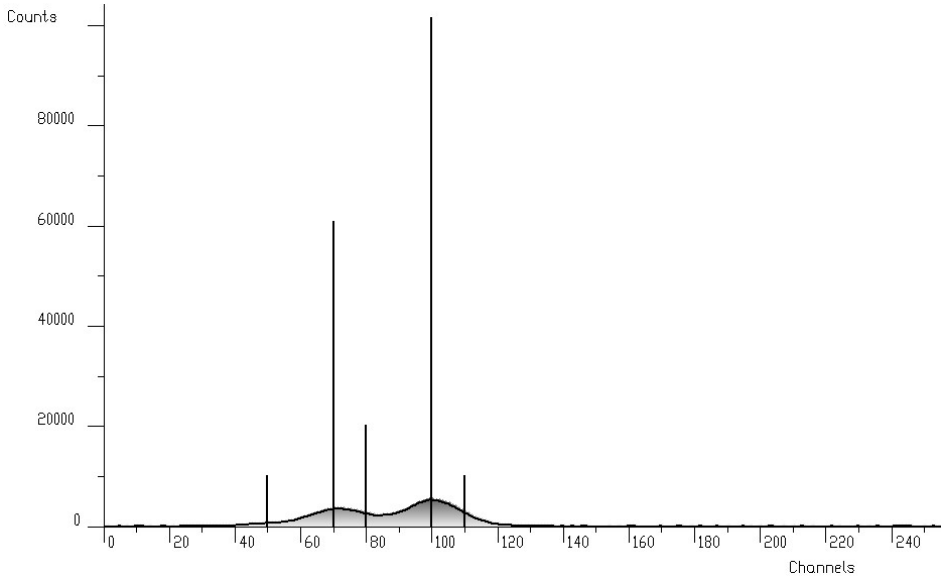


Figure 15 The same spectrum like in Figure 14, outlined bars show the contents of present components (peaks)

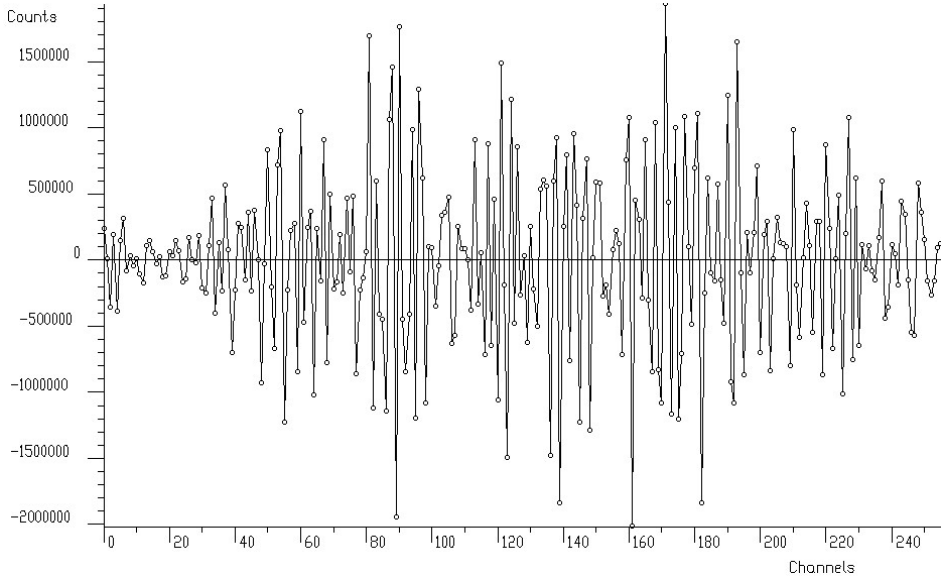


Figure 16 Least squares solution of the system of linear equations without regularization

Example 10 – script Deconvolution1_wide.c :

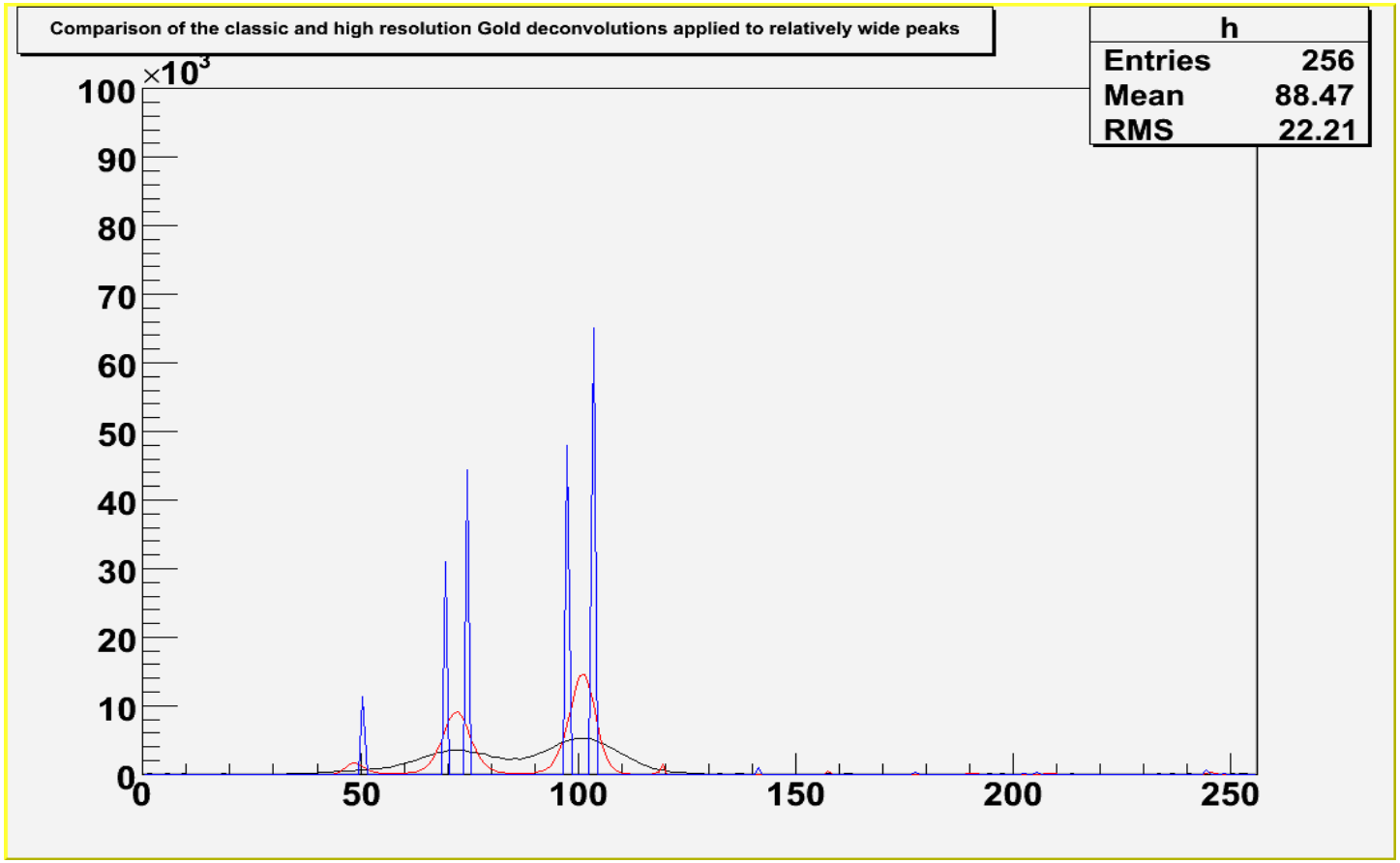


Figure 17 The original source spectrum is drawn with black color, deconvolved spectrum using classic Gold deconvolution (10000 iterations) with red color, deconvolved spectrum using classic high resolution Gold deconvolution (200 iterations, 50 repetitions, p=1.2) with blue color.

Decomposition - unfolding

$$y(i) = \sum_{k=0}^{N-1} h(i,k)x(k), \quad i = 0,1,2,\ldots\ldots,N-1$$

$$\begin{bmatrix} y(0) \\ y(1) \\ . \\ . \\ . \\ y(N_x-1) \end{bmatrix} = \begin{bmatrix} h(0,0) & h(0,1) & . & h(0,N_y-1) \\ h(1,0) & h(1,1) & . & h(1,N_y-1) \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ h(N_x-1,0) & h(N_x-1,1) & . & h(N_x-1,N_y-1) \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ . \\ x(N_y-1) \end{bmatrix}$$

Function 5:

const char* Deconvolution1Unfolding(float *source, const float **resp, int sizex, int sizey, int number_of_iterations)

This function unfolds source spectrum according to response matrix columns. The result is placed in the vector pointed by source pointer.

Parameters

•source - pointer to the vector of source spectrum
•resp - pointer to the matrix of response spectra
•sizex - length of source spectrum and # of columns in response matrix
•sizey - length of destination spectrum and # of rows in response matrix
•number_of_iterations – parameter I in the Gold deconvolution algorithm

Note!!! sizex must be >= sizey After decomposition the resulting channels are written back to the first sizey channels of the source spectrum.

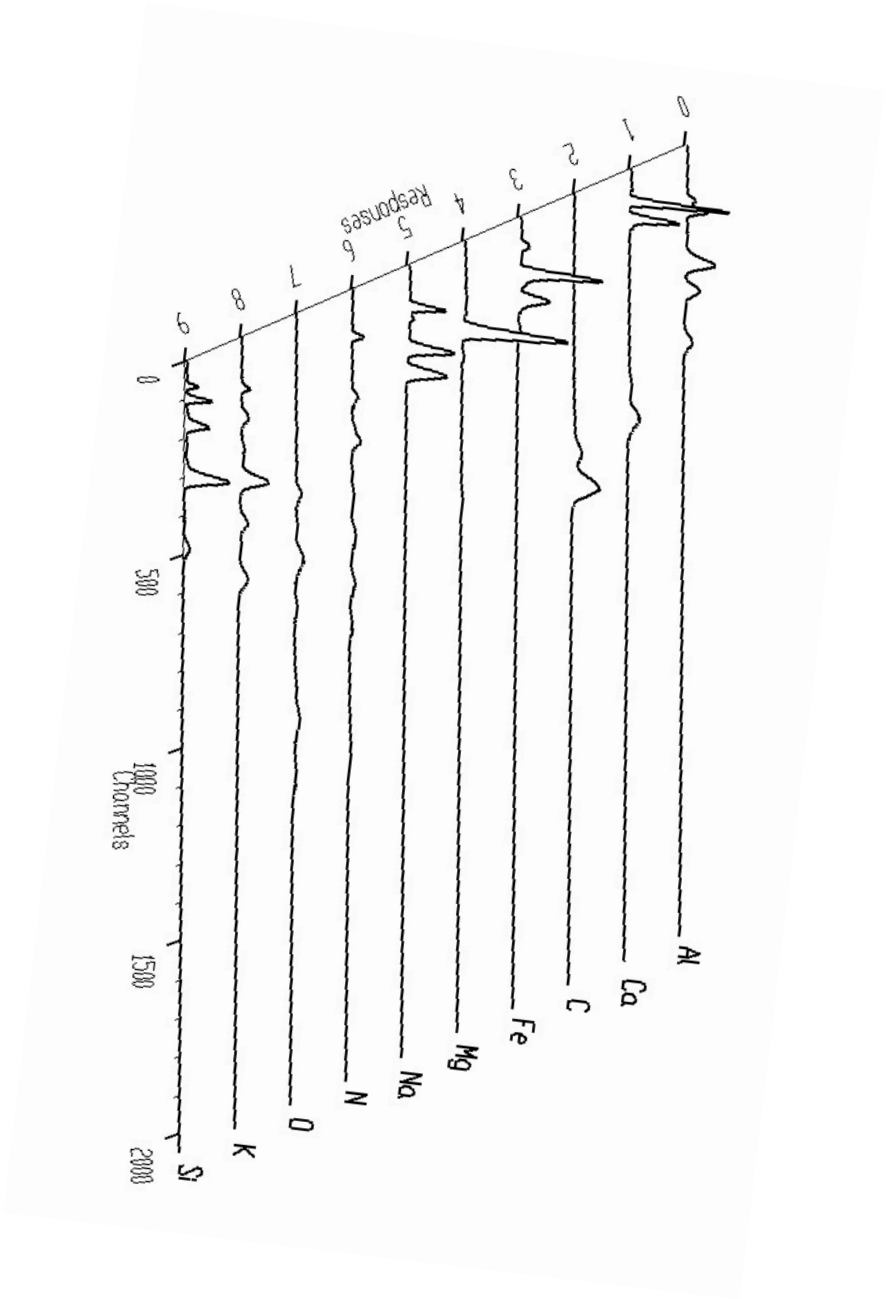


Figure 18 Response matrix composed of neutron spectra of pure chemical elements

Example 11 – script Deconvolution1Unfolding.c :

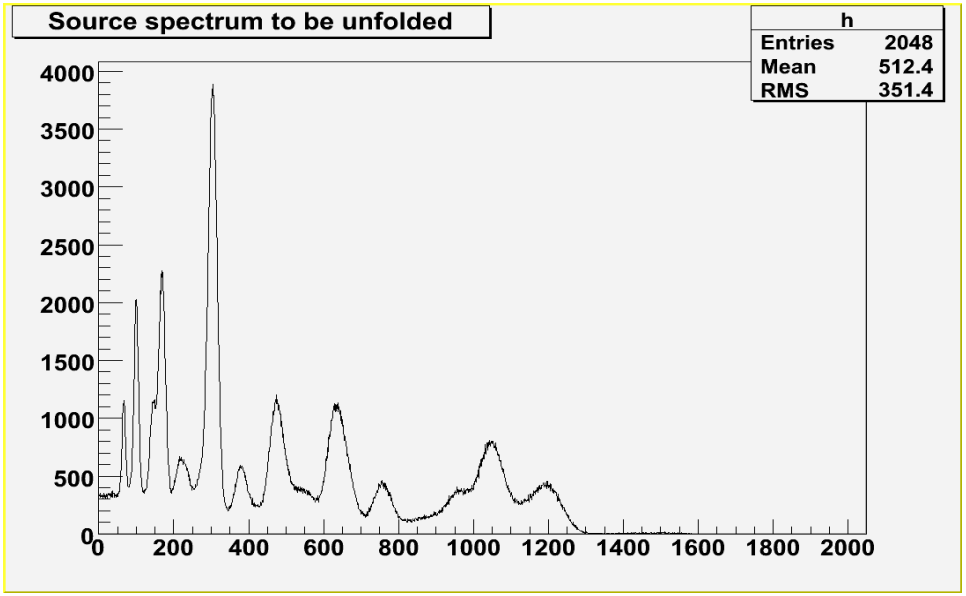


Figure 19 Source neutron spectrum to be decomposed

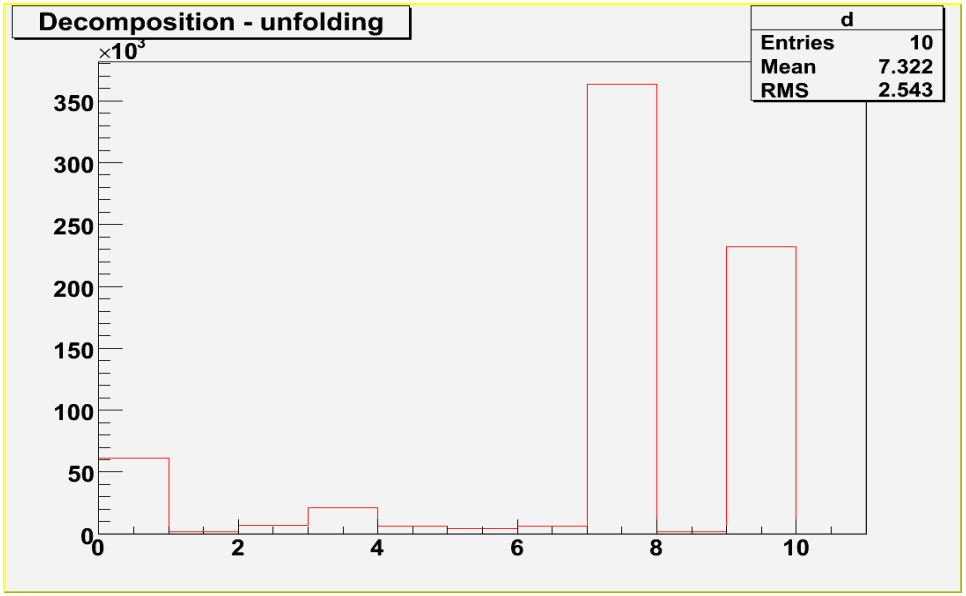


Figure 20 Spectrum after decomposition, contains 10 coefficients, which correspond to contents of chemical components (dominant 8-th and 10-th components, i.e. O, Si)

References:

[1] Gold R., ANL-6984, Argonne National Laboratories, Argonne Ill, 1964.

[2] Coote G.E., Iterative smoothing and deconvolution of one- and two-dimensional elemental distribution data, NIM B 130 (1997) 118.

[3] M. Morháč, J. Kliman, V. Matoušek, M. Veselský, I. Turzo.: Efficient one- and two-dimensional Gold deconvolution and its application to gamma-ray spectra decomposition. NIM, A401 (1997) 385-408.

[4] Morháč M., Matoušek V., Kliman J., Efficient algorithm of multidimensional deconvolution and its application to nuclear data processing, Digital Signal Processing 13 (2003) 144.

Possible new TSpectrum deconvolution methods:

•one-fold Gold deconvolution and its boosted version (see e.g. M. Morháč, Deconvolution methods and their applications in the analysis of gamma-ray spectra, ACAT2005, May 22-27, Zeuthen Germany)

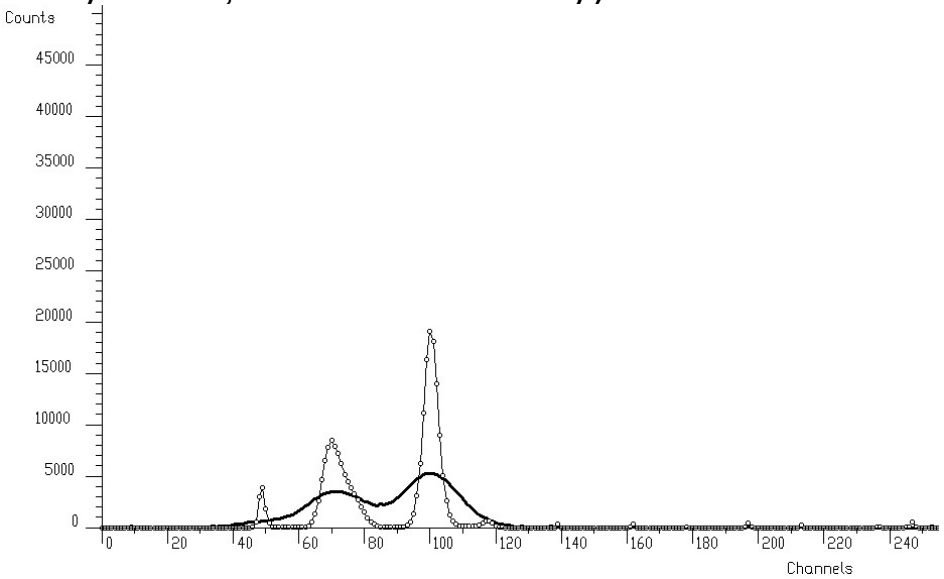


Figure 21 Original spectrum (thick line) and deconvolved spectrum using one-fold Gold algorithm (10000 iterations, thin line)

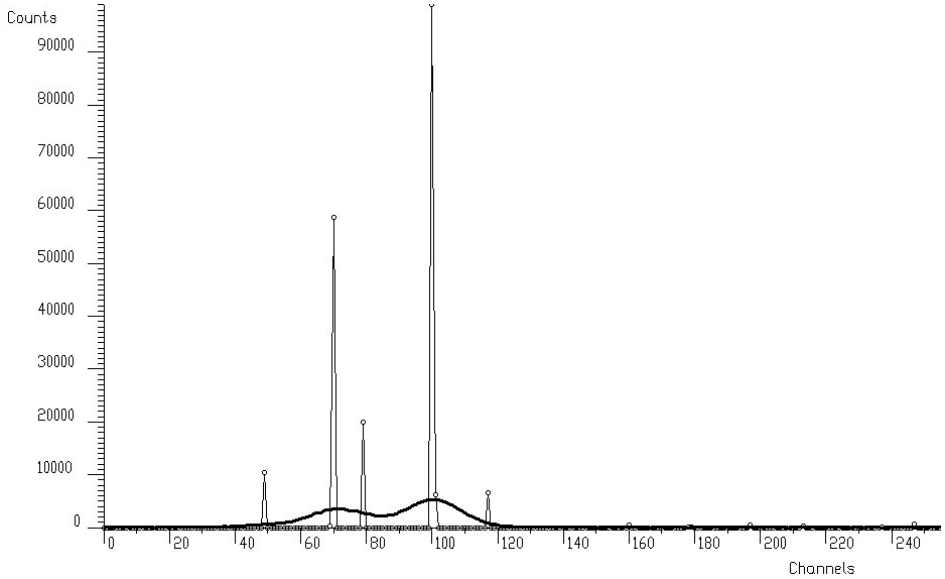


Figure 22 Illustration of deconvolution via boosted Gold deconvolution

Peak #	Original/Estimated (max) position	Original/Estimated area
1	50/49	10159/10419
2	70/70	60957/58933
3	80/79	20319/19935
4	100/100	101596/105413
5	110/117	10159/6676

Table 2 Results of the estimation of peaks in spectrum shown in Figure 22

•Richardson – Lucy deconvolution and its boosted version -see e.g.
Lucy L.B., A.J. 79 (1974) 745.
Richardson W.H., J. Opt. Soc. Am. 62 (1972) 55.

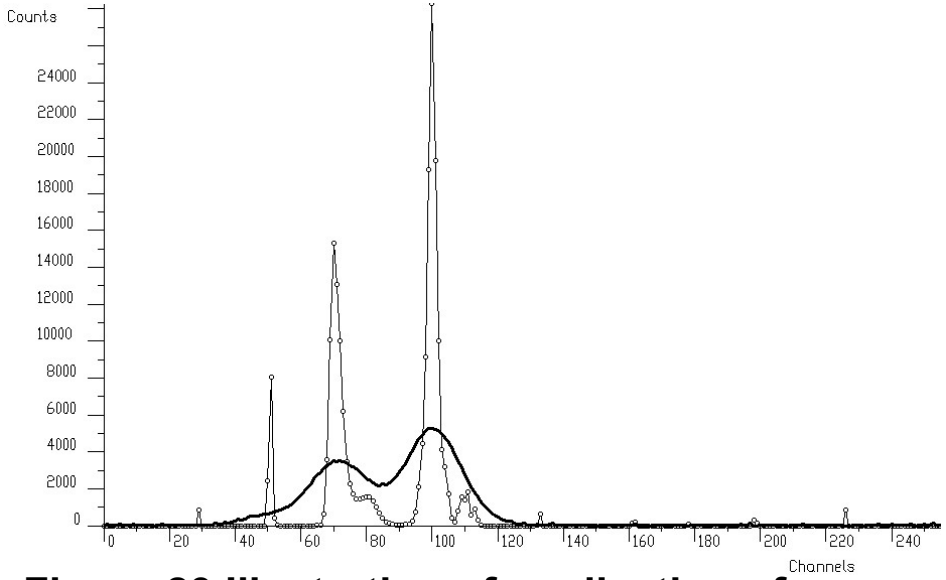


Figure 23 Illustration of application of Richardson-Lucy algorithm of deconvolution

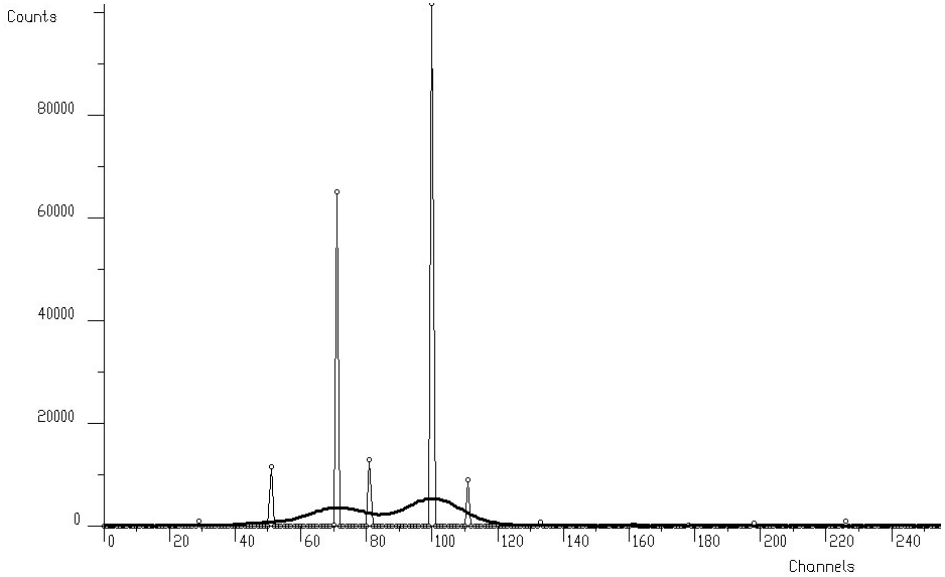


Figure 24 Spectrum deconvolved using boosted Richardson-Lucy algorithm of deconvolution

Peak #	Original/Estimated (max) position	Original/Estimated area
1	50/51	10159/11426
2	70/71	60957/65003
3	80/81	20319/12813
4	100/100	101596/101851
5	110/111	10159/8920

Table 3 Results of the estimation of peaks in spectrum shown in Figure 24

Smoothing

Goal: Suppression of statistical fluctuations

•there exist enormous number of filtration techniques
•we mention Markov smoothing algorithm, which can be employed for the identification of peaks in noisy spectra
•the algorithm is based on discrete Markov chain, which has very simple invariant distribution

$$U_2 = \frac{p_{1,2}}{p_{2,1}} U_1, U_3 = \frac{p_{2,3}}{p_{3,2}} U_2 U_1, \ldots, U_n = \frac{p_{n-1,n}}{p_{n,n-1}} U_{n-1} \ldots U_2 U_1$$

U_1 being defined from the normalization condition $\sum_{i=1}^{n} U_i = 1$

n is the length of the smoothed spectrum and

$$p_{i,i\pm1} = A_i \sum_{k=1}^{m} \exp\left[\frac{y(i \pm k) - y(i)}{y(i \pm k) + y(i)}\right]$$

is the probability of the change of the peak position from channel I to the channel i+1.

A_i is the normalization constant so that $p_{i,i-1} + p_{i,i+1} = 1$ and m is a width of smoothing window.

Function 6:

const char* TSpectrum::Smooth1Markov(float *source, int size, int aver_window)

This function calculates smoothed spectrum from the source spectrum based on Markov chain method. The result is placed in the vector pointed by source pointer.

Parameters

•source - pointer to the vector of source spectrum
•size - length of source spectrum
•aver_window - width of averaging smoothing window

Reference:

[1] Z.K. Silagadze, A new algorithm for automatic photopeak searches. NIM A 376 (1996), 45**1.**

Example 12 – script Smoothing1.c :

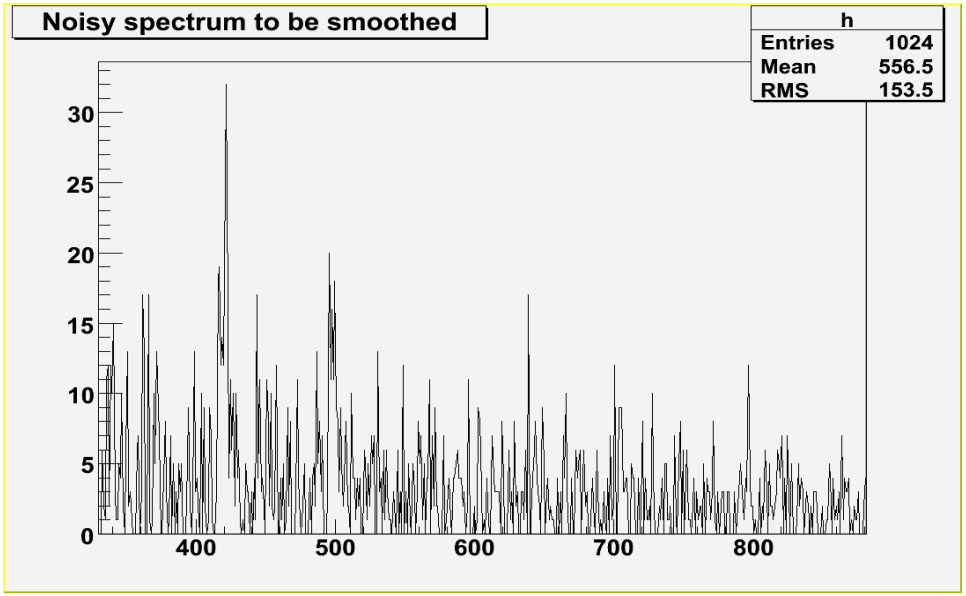


Figure 25 Original noisy spectrum

Example 13 – script Smoothing2.c :

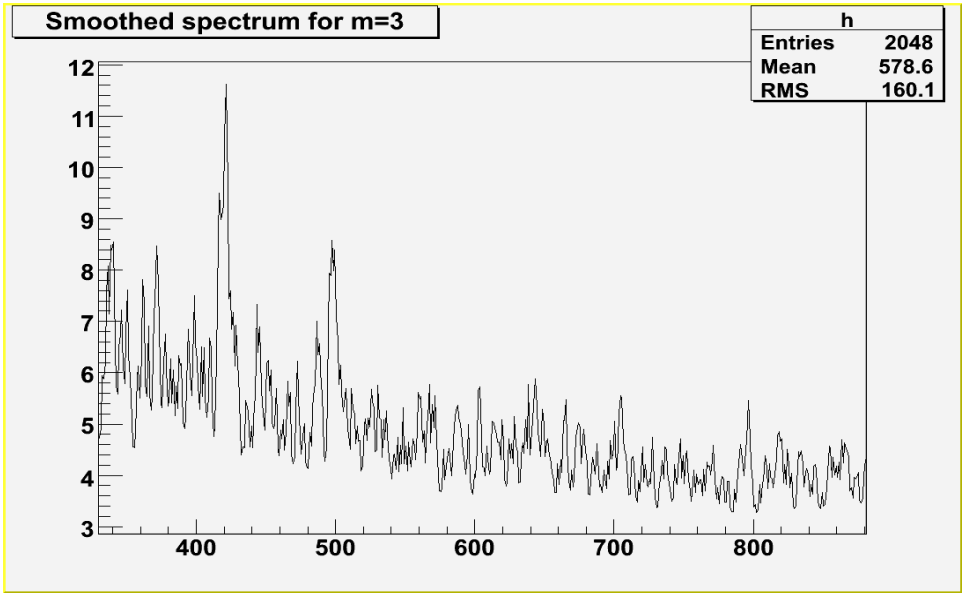


Figure 26 Smoothed spectrum m=3

Example 14 – script Smoothing3.c :

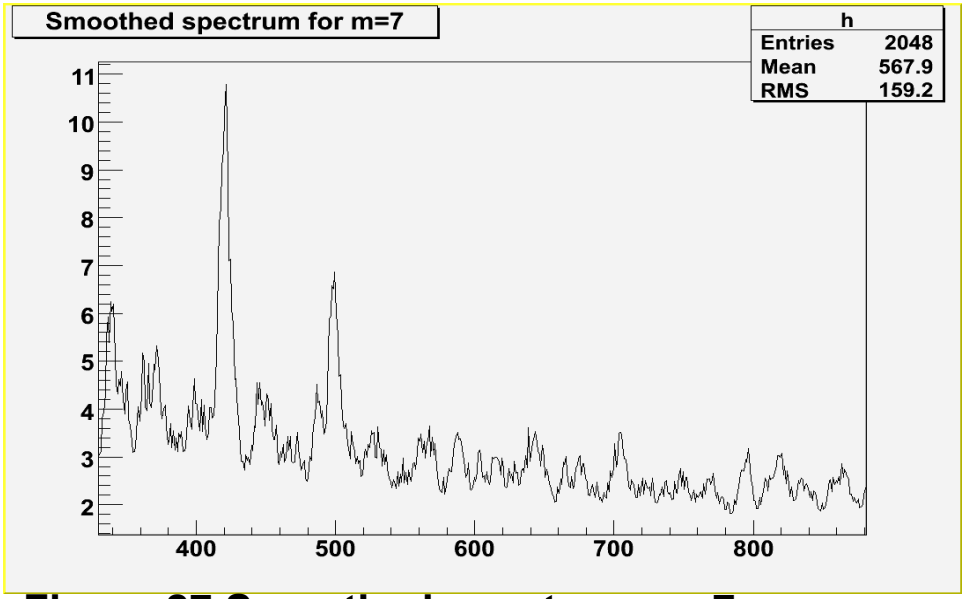


Figure 27 Smoothed spectrum m=7

Example 15 – script Smoothing4.c :

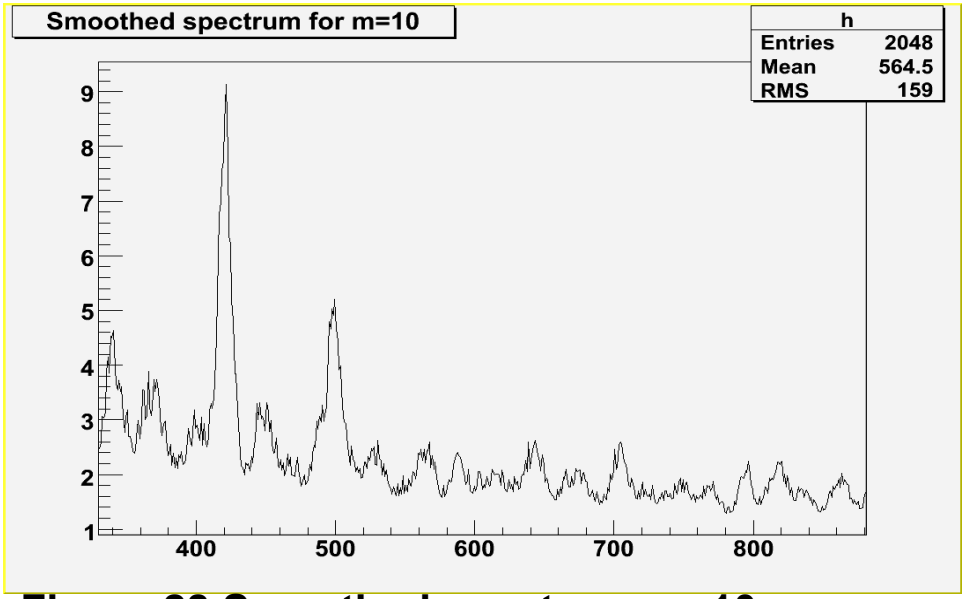


Figure 28 Smoothed spectrum m=10

Peaks searching

Goal: to identify automatically the peaks in a spectrum with the presence of the continuous background and statistical fluctuations - noise.

The common problems connected with correct peak identification are
•non-sensitivity to noise, i.e., only statistically relevant peaks should be identified.
•non-sensitivity of the algorithm to continuous background.
•ability to identify peaks close to the edges of the spectrum region. Usually peak finders fail to detect them.
•resolution, decomposition of doublets and multiplets. The algorithm should be able to recognize close positioned peaks.
•ability to identify peaks with different sigma

General peak searching algorithm based on smoothed second differences – implemented in Search1 function in old versions of ROOT (up to the Version 3.04)

•it is based on smoothed second differences (SSD) that are compared to its standard deviations

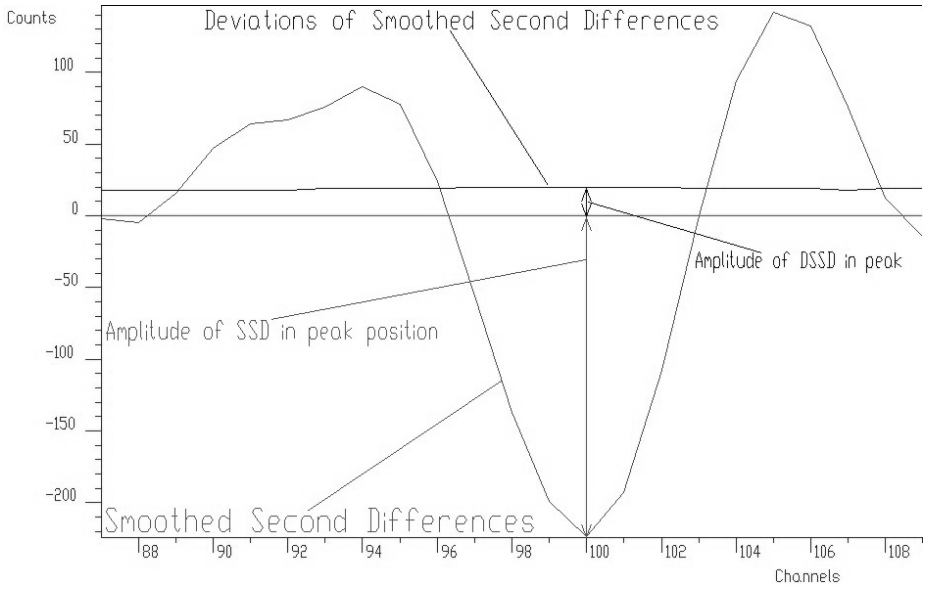


Figure 29 Smoothed Second Differences (SSD) and its standard deviation in the vicinity of peak

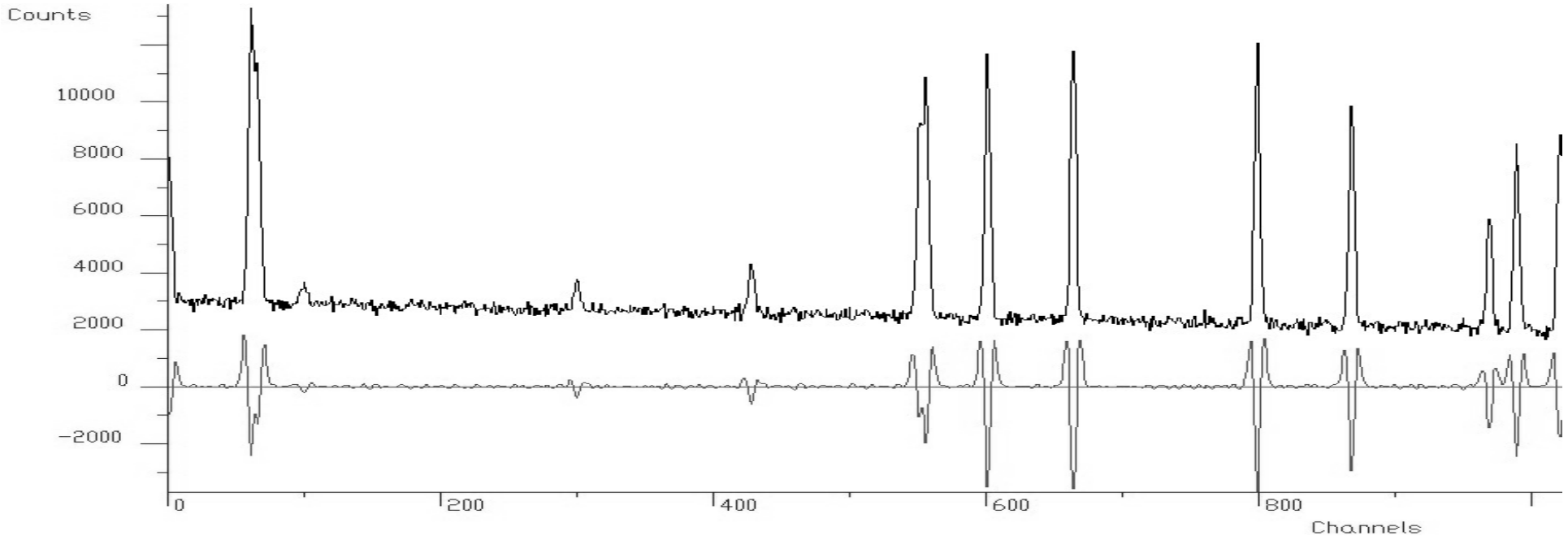


Figure30 Synthetic spectrum and its SSD spectrum

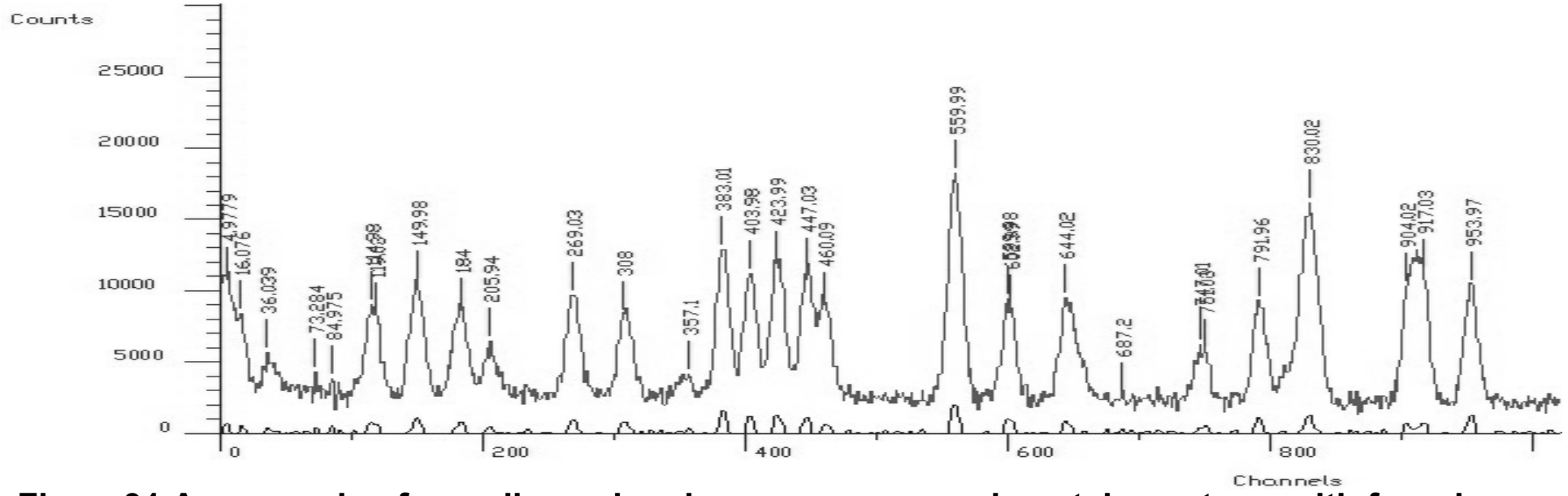


Figure31 An example of one-dimensional gamma-ray experimental spectrum with found peaks and its inverted positive SSD spectrum

High resolution peak searching algorithm

•it is based on the above presented Gold deconvolution algorithm
•unlike SSD algorithm before applying the deconvolution algorithm we have to remove the background using one of the above presented background elimination algorithms
•if desired, before applying the peak searching algorithm the Markov smoothed spectrum (described in previous section) can be calculated as well

Function 7:

Int_t Search(TH1 * hin, Double_t sigma, Option_t * option, Double_t threshold)

This function searches for peaks in source spectrum in hin The number of found peaks and their positions are written into the members fNpeaks and fPositionX. The search is performed in the current histogram range.

Parameters

•hin - pointer to the histogram of source spectrum
•sigma - sigma of searched peaks
•option - if option is not equal to "goff" (goff is the default), then a polymarker object is created and added to the list of functions of the histogram. The histogram is drawn with the specified option and the polymarker object drawn on top of the histogram. The polymarker coordinates correspond to the npeaks peaks found in the histogram. A pointer to the polymarker object can be retrieved later via: TList *functions = hin->GetListOfFunctions(); TPolyMarker *pm = (TPolyMarker*)functions->FindObject("TPolyMarker")
•threshold: (default=0.05) peaks with amplitude less than threshold*highest_peak are discarded. 0<threshold<1

Example 16 – script Search1.c :

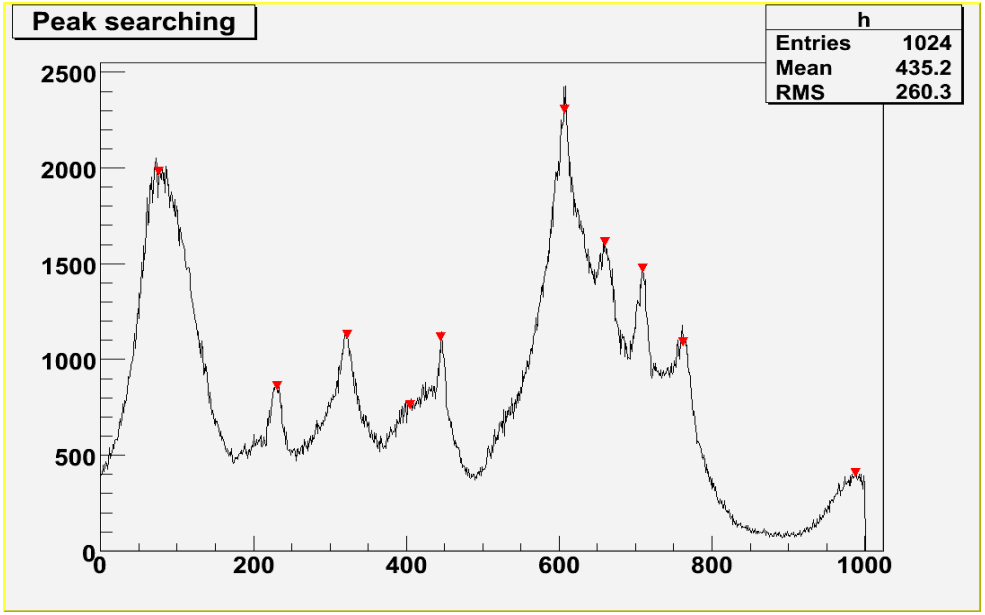


Figure 32 An example of one-dimensional synthetic spectrum with found peaks denoted by markers

Function 8:

Int_t Search1HighRes(float *source, float *dest, int size, float sigma, double threshold, bool background_remove,int decon_iterations, bool markov, int aver_window)

This low level function searches for peaks in source spectrum. It is based on deconvolution method. First the background is removed (if desired), then Markov spectrum is calculated (if desired), then the response function is generated according to given sigma and deconvolution is carried out. On success it returns number of found peaks.

Parameters

•source - pointer to the vector of source spectrum
•dest – resulting spectrum after deconvolution
•size - length of source and destination spectra
•sigma-sigma of searched peaks
•threshold-threshold value in % for selected peaks, peaks with amplitude less than threshold*highest_peak/100 are ignored
•background_remove-logical variable, true if the removal of background before deconvolution is desired
•decon_iterations-number of iterations in deconvolution operation
•markov-logical variable, if it is true, first the source spectrum is replaced by new spectrum calculated using Markov chains method
•aver_window - width of averaging smoothing window

Note

Search1HighRes function provides users with the possibility to vary the input parameters and with the access to the output deconvolved data in the destination spectrum. Based on the output data one can tune the parameters. Search function calls Search1HighRes function with default values background_remove=true, decon_iterations=3, markov=true and aver_window=3.- pointer to the vector of source spectrum.

Example 17 – script Search1_hr1.c:

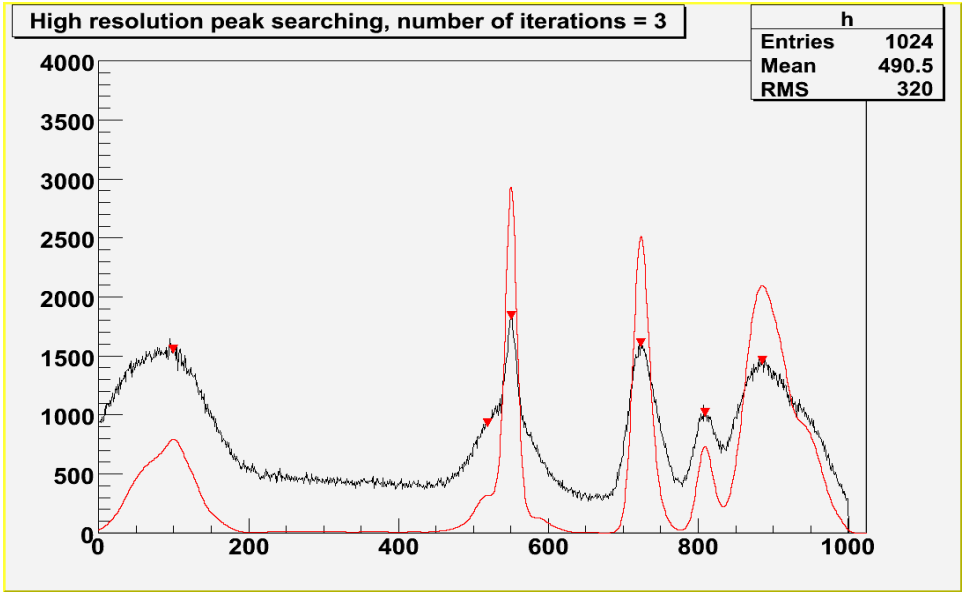


Figure 33 One-dimensional spectrum with found peaks denoted by markers, 3 iterations steps in the deconvolution

Example 18 – script Search1_hr2.c:

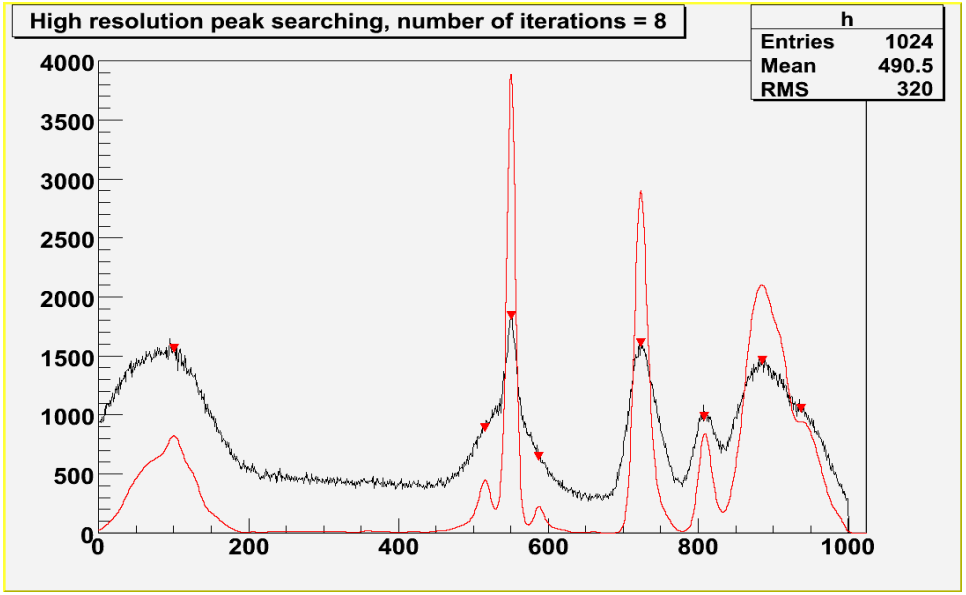


Figure 34 One-dimensional spectrum with found peaks denoted by markers, 8 iterations steps in the deconvolution

Peak #	Position	Sigma
1	118	26
2	162	41
3	310	4
4	330	8
5	482	22
6	491	26
7	740	21
8	852	15
9	954	12
10	989	13

Table 4 Positions and sigma of peaks in the following examples

Example 20 – script Search1_hr4.c:

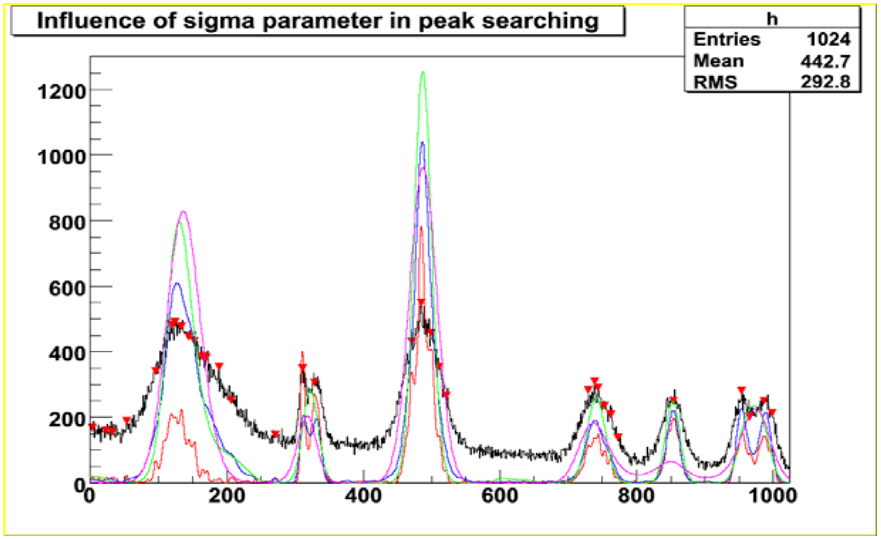


Figure 36 Influence of sigma (3-red, 8-blue, 20-green, 43-magenta), num. iter.=10, sm. width=3

Example 19 – script Search1_hr3.c:

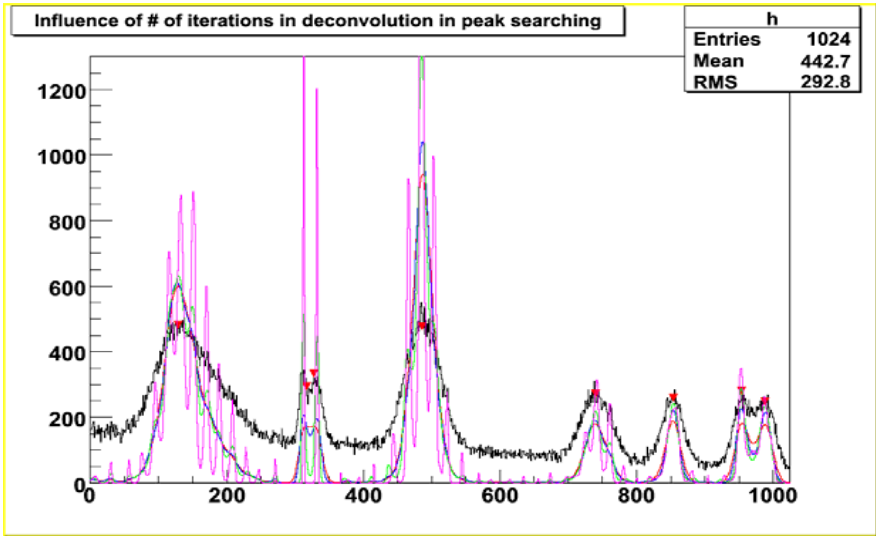


Figure 35 Influence of number of iterations (3-red, 10-blue, 100- green, 1000-magenta), sigma=8, smoothing width=3

Example 21 – script Search1_hr5.c:

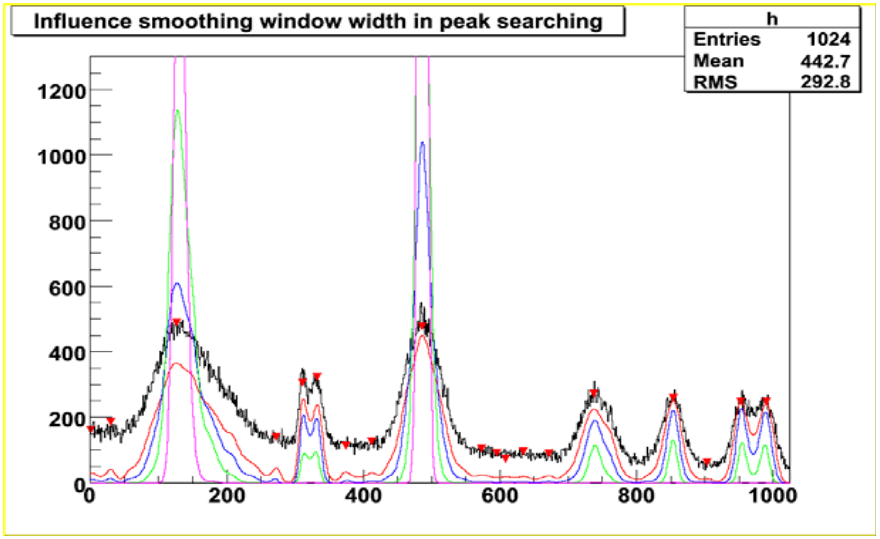


Figure 37 Influence smoothing width (0-red, 3-blue, 7- green, 20-magenta), num. iter.=10, sigma=8

References:

[1] M.A. Mariscotti: A method for identification of peaks in the presence of background and its application to spectrum analysis. NIM 50 (1967), 309-320.
[2] M. Morháč, J. Kliman, V. Matoušek, M. Veselský, I. Turzo.:Identification of peaks in multidimensional coincidence gamma-ray spectra. NIM, A443 (2000) 108-125.
[3] Z.K. Silagadze, A new algorithm for automatic photopeak searches. NIM A 376 (1996), 451.

Possible new TSpectrum peak searching method:

Sigma range peak searching algorithm

- the proposed algorithm is to some extent robust to the variations of sigma parameter.
- however, for large scale of the range of sigma and for poorly resolved peaks this algorithm fails to work properly.
- if we knew how the sigma changes in dependence on position we could set up the response matrix and employ the algorithm like Deconvolution1Unfolding
- usually this is not the case and therefore we have simultaneously identify peaks and to adjust their sigma

Outline of the algorithm

For every sigma the algorithm comprises two deconvolutions. The principle of the method is as follows:

a. For $\sigma_i = \sigma_1$ up to σ_2

b. We set up the matrix of response functions (Gaussians) according to Figure 38. All peaks have the same $\sigma = \sigma_i$. The columns of the matrix are mutually shifted by one position. We carry out the Gold deconvolution of the investigated spectrum.

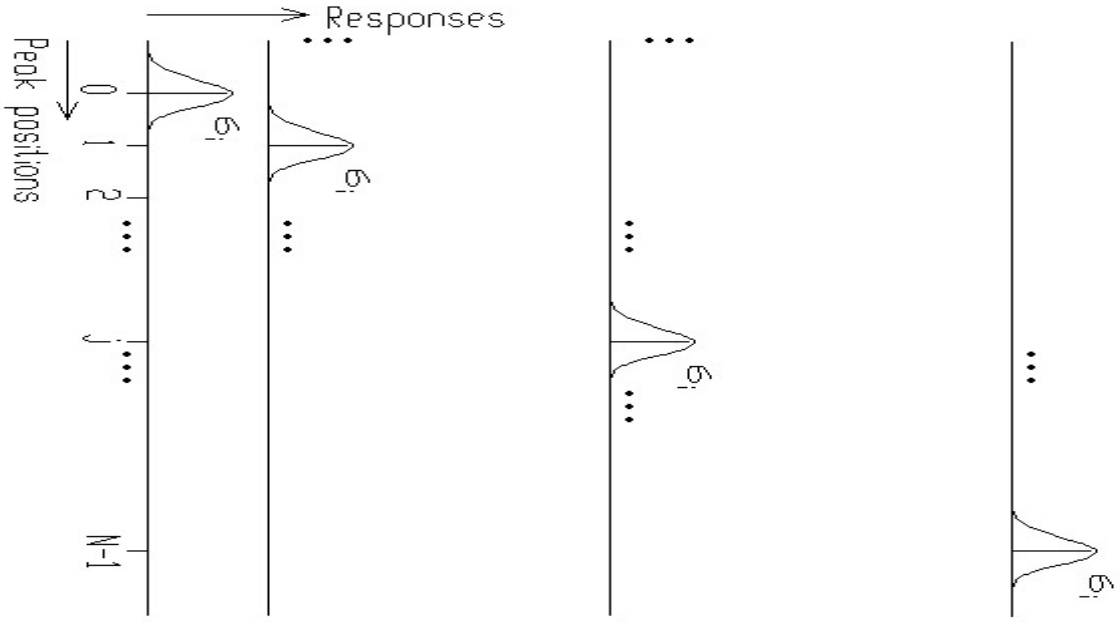


Figure 38 Response matrix consisting of Gaussian functions with the same σ

c. In the deconvolved spectrum, we find local maxima higher than given threshold value and include them into the list of 1-st level candidate peaks.

d. Next, we set up the matrix of the response functions for the 2-nd level deconvolution. There exist three groups of positions:

-positions where the 1-st level candidate peaks were localized. Here we generate response (Gaussian) with σ_i

-positions where the 2-nd level candidate peaks were localized in the previous steps $\sigma_1 \leq \sigma_k < \sigma_i$ (see next step). For each such a position, we generate the peak with the recorded σ_k

-for remaining free positions where no candidate peaks were registered, we have empirically found that the most suitable functions are the block functions with the width $\pm 3\sigma_i$ from the appropriate channel

The situation for one 2-nd level candidate peak in position j_2 with σ_k and for one 1-st level candidate peak in the position j_1 (σ_i) is depicted in Figure 39. We carry out the 2-nd level Gold deconvolution.

e. Further, in the deconvolved spectrum we find the local maxima greater than given threshold value.

We scan the list of the 2-nd level candidate peaks:
- if in a position from this list there is not local maximum in the deconvolved spectrum, we erase the candidate peak from the list.

We scan the list of the 1-st level candidate peak
- if in a position from this list there is the local maximum in the deconvolved spectrum we transfer it to the list of the 2-nd level candidate peaks.

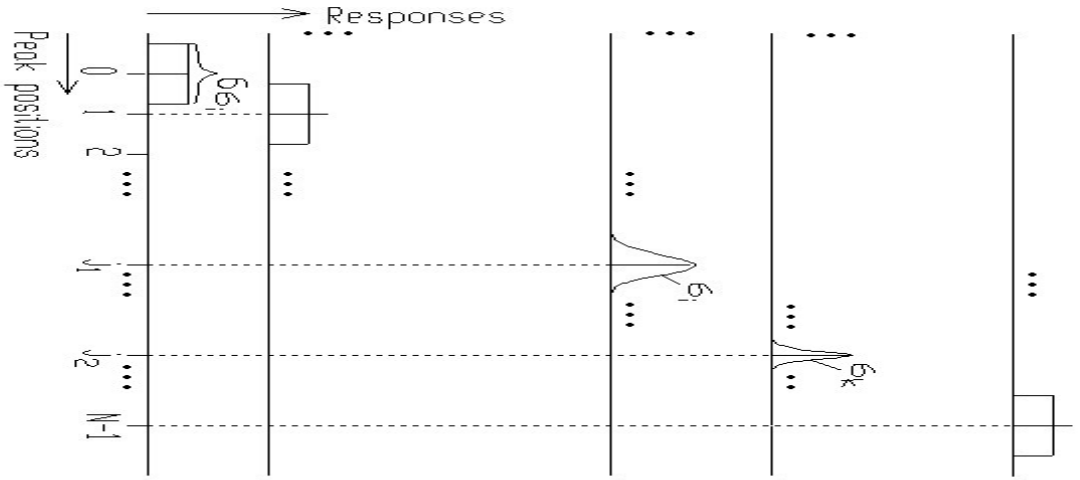


Figure 39 Example of response matrix consisting of block functions and Gaussians with different σ

f. Finally peaks that remained in the list of the 2-nd level candidate peaks are identified as found peaks with recorded positions and σ

- the method is rather complex as we have to repeat two deconvolutions for the whole range of σ
- in what follows we illustrate in detail practical aspects and steps during the peak identification. The original noisy spectrum to be processed is shown in Figure 40. The σ of peaks included in the spectrum varies in the range 3 to 43. It contains 10 peaks with some of them positioned very close to each other.
- as the SRS algorithm is based on the deconvolution in the first step we need to remove background (Figure 41).

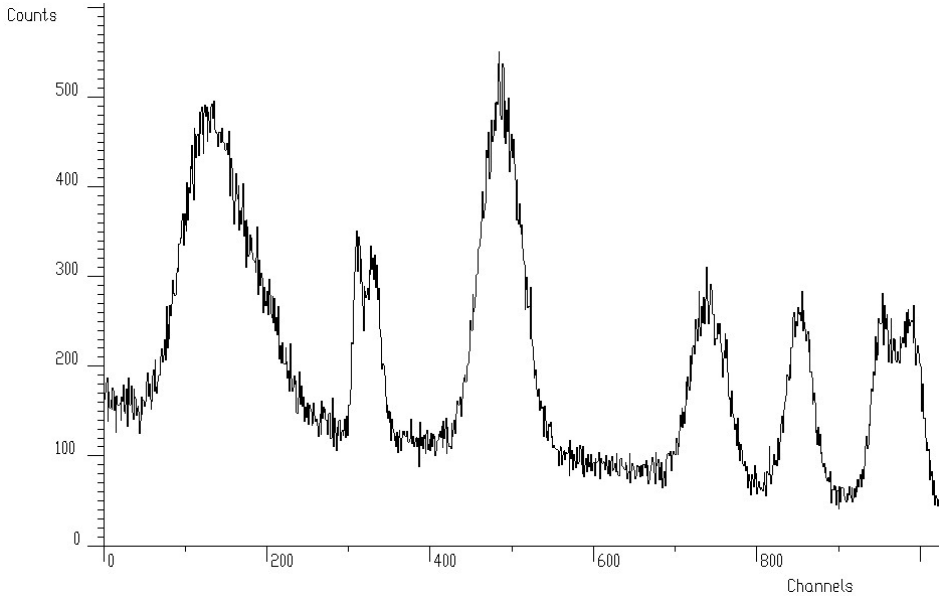


Figure 40 Noisy spectrum containing 10 peaks of rather different widths

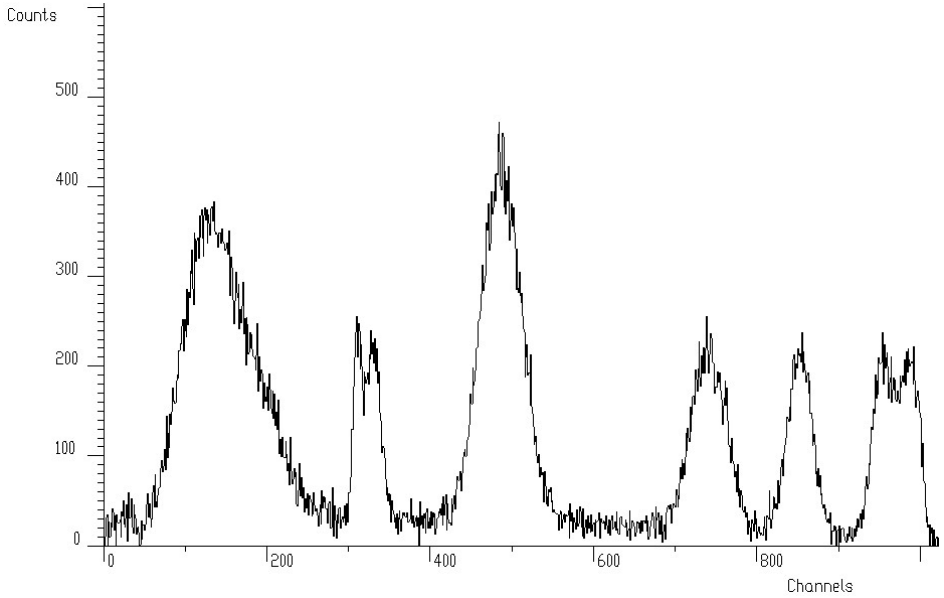


Figure 41 Spectrum from Figure 40 after background elimination

- at every position, we have to expect any peak from the given σ range (3, 43). To confine the possible combinations of positions and σ we generated inverted positive SSD of the spectrum for every σ from the range.

- we get matrix shown in next Figure. We consider only the combinations with non-zero values in the matrix.

- the SRS algorithm is based on two successive deconvolutions. In the first level deconvolution, we look for peak candidates. We changed σ from 3 up to 43.

- for every σ we generated response matrix and subsequently we deconvolved spectrum from Figure 41. Again, we arranged the results in the form of matrix given in Figure 43.

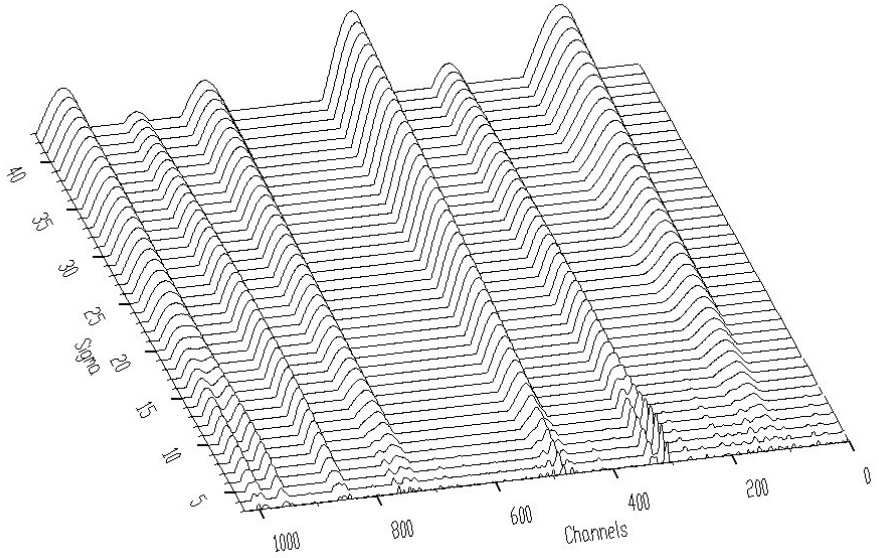


Figure 42 Matrix of inverted positive SSD

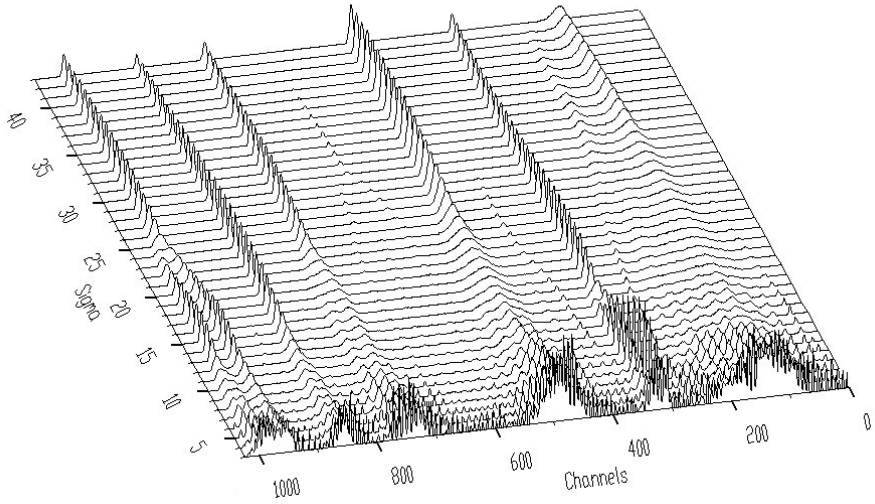


Figure 43 Matrix composed of spectra after first level deconvolution

- successively according to the above-given algorithm from these data, we pick up the candidates for peaks, construct the appropriate response matrices and deconvolve again the spectrum from Figure 41.

- the evolution of the result for increasing σ is shown in Figure 44.

- from the last row of the matrix from Figure 44, which are in fact spikes, we can identify (applying threshold parameter) the positions of peaks. The found peaks (denoted by markers with channel numbers) and the original spectrum are shown in Figure 45.

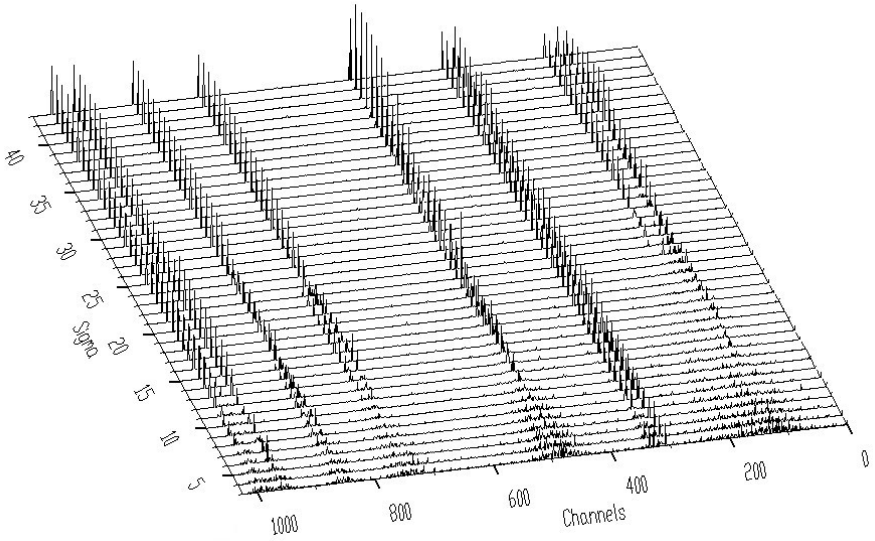


Figure 44 Matrix composed of spectra after second level deconvolutions

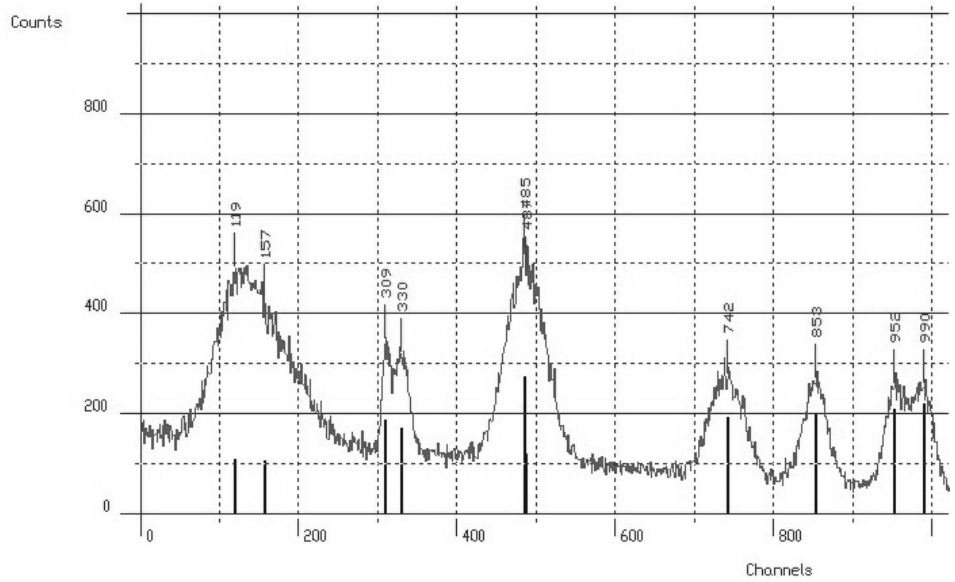


Figure 45 Original spectrum with found peaks denoted by markers

- in Table 5 we present the result of generated peaks and estimated parameters. In addition to the peak position the algorithm estimates even the sigma of peaks. It is able to recognize also very closely positioned peaks.

- however, the estimate of the parameters is in some cases rather inaccurate mainly in poorly separated peaks of the spectrum. The problem is very complex and sometimes it is very difficult to decide whether a lobe represents two, eventually more, close positioned narrow peaks or one wide peak.
- the algorithm is rather complex and thus time consuming

Peak #	Generated peaks (position/sigma)	Estimated peaks (position/sigma)
1	118/26	119/23
2	162/41	157/26
3	310/4	309/4
4	330/8	330/7
5	482/22	485/23
6	491/24	487/27
7	740/21	742/22
8	852/15	853/16
9	954/12	952/12
10	989/13	990/12

Table 5 Results of the estimation of peaks in spectrum shown in Figure 45

Fitting

Goal: to estimate simultaneously peak shape parameters in spectra with large number of peaks

- peaks can be fitted separately, each peak (or multiplets) in a region or together all peaks in a spectrum. To fit separately each peak one needs to determine the fitted region. However it can happen that the regions of neighboring peaks are overlapping. Then the results of fitting are very poor. On the other hand, when fitting together all peaks found in a spectrum, one needs to have a method that is stable (converges) and fast enough to carry out fitting in reasonable time
- we have implemented the nonsymmetrical semiempirical peak shape function [1]
- it contains the symmetrical Gaussian as well as nonsymmetrical terms.

$$f(i,\mathbf{a}) = \sum_{i=1}^{M} A(j)\left\{\exp\left[-(i-p(j))^2/2\sigma^2\right] + \frac{1}{2}T\ \exp\left[\left(i-p(j)\right)/B\sigma\right].erfc\left[\left(i-p(j)\right)/\sigma + \frac{1}{2B}\right] + \frac{1}{2}S\ erfc\left[\left(i-p(j)\right)/\sigma\right]\right\}$$

where T and S are relative amplitudes and B is slope.

- algorithm without matrix inversion (AWMI) allows fitting tens, hundreds of peaks simultaneously that represent sometimes thousands of parameters [2], [6].

Function 9:

const char *TSpectrum::Fit1Awmi(float *source, TSpectrumOneDimFit * p, int size)

This function fits the source spectrum using AWMI algorithm. The calling program should fill in input parameters of the TSpectrumOneDimFit class. The fitted parameters are written into class pointed by TSpectrumOneDimFit class pointer and fitted data are written into source spectrum.

Parameters

•source - pointer to the vector of source spectrum
•p-pointer to the TSpectrumOneDimFit class
•size - length of source spectrum

```
class TSpectrumOneDimFit{
public:   int    number_of_peaks;      //input parameter, should be>0
int    number_of_iterations;           //input parameter, should be >0
int    xmin;                           //first fitted channel
int    xmax;                           //last fitted channel
double alpha;                          //convergence coefficient, input parameter, it should be positive number and <=1
double chi;                            //here the function returns resulting chi square
int    statistic_type;                 //type of statistics, possible values FIT1_OPTIM_CHI_COUNTS (chi square statistics with
                                       // counts as weighting coefficients), FIT1_OPTIM_CHI_FUNC_VALUES (chi square statistics
                                       //with function values as weighting coefficients [3]),FIT1_OPTIM_MAX_LIKELIHOOD [4]
int    alpha_optim;                    //optimization of convergence coefficients, possible values FIT1_ALPHA_HALVING,
                                       //FIT1_ALPHA_OPTIMALI
 int    power;                         //possible values FIT1_FIT_POWER2,4,6,8,10,12 (applies only for awmi algorithm)
 int    fit_taylor;                    //order of Taylor expansion, possible values FIT1_TAYLOR_ORDER_FIRST,
                                      //FIT1_TAYLOR_ORDER_SECOND (applies only for awmi algorithm)
double position_init[MAX_NUMBER_OF_PEAKS1];    //initial values of peaks positions, input parameters   double
position_calc[MAX_NUMBER_OF_PEAKS1];           //calculated values of fitted positions, output parameters   double
position_err[MAX_NUMBER_OF_PEAKS1];            //position errors
bool   fix_position[MAX_NUMBER_OF_PEAKS1];     //logical vector which allows to fix appropriate positions (not fit). However they
                                               //are present in the estimated functional   double
amp_init[MAX_NUMBER_OF_PEAKS1];                //initial values of peaks amplitudes, input parameters   double
amp_calc[MAX_NUMBER_OF_PEAKS1];                //calculated values of fitted amplitudes, output parameters   double
amp_err[MAX_NUMBER_OF_PEAKS1];                 //amplitude errors
bool   fix_amp[MAX_NUMBER_OF_PEAKS1];          //logical vector which allows to fix appropriate amplitudes (not fit). However
                                               //they are present in the estimated functional
double area[MAX_NUMBER_OF_PEAKS1];             //calculated areas of peaks
double area_err[MAX_NUMBER_OF_PEAKS1];         //errors of peak areas
double sigma_init;                             //sigma parameter, meaning analogical to the above given parameters
double sigma_calc;
double sigma_err;
bool   fix_sigma;
double t_init, double t_calc,  double t_err,  bool   fix_t;
double b_init, double b_calc,   double b_err,   bool   fix_b;
double s_init, double s_calc,  double s_err,   bool   fix_s;
double a0_init;   //backgroud is estimated as a0+a1*x+a2*x*x
double a0_calc,   double a0_err,   bool   fix_a0, double a1_ini,   double a1_calc,   double a1_err,   bool   fix_a1;
double a2  init,   double a2  calc,   double a2  err,   bool   fix  a2;};
```

Example 22 – script Fit1_awmi.c:

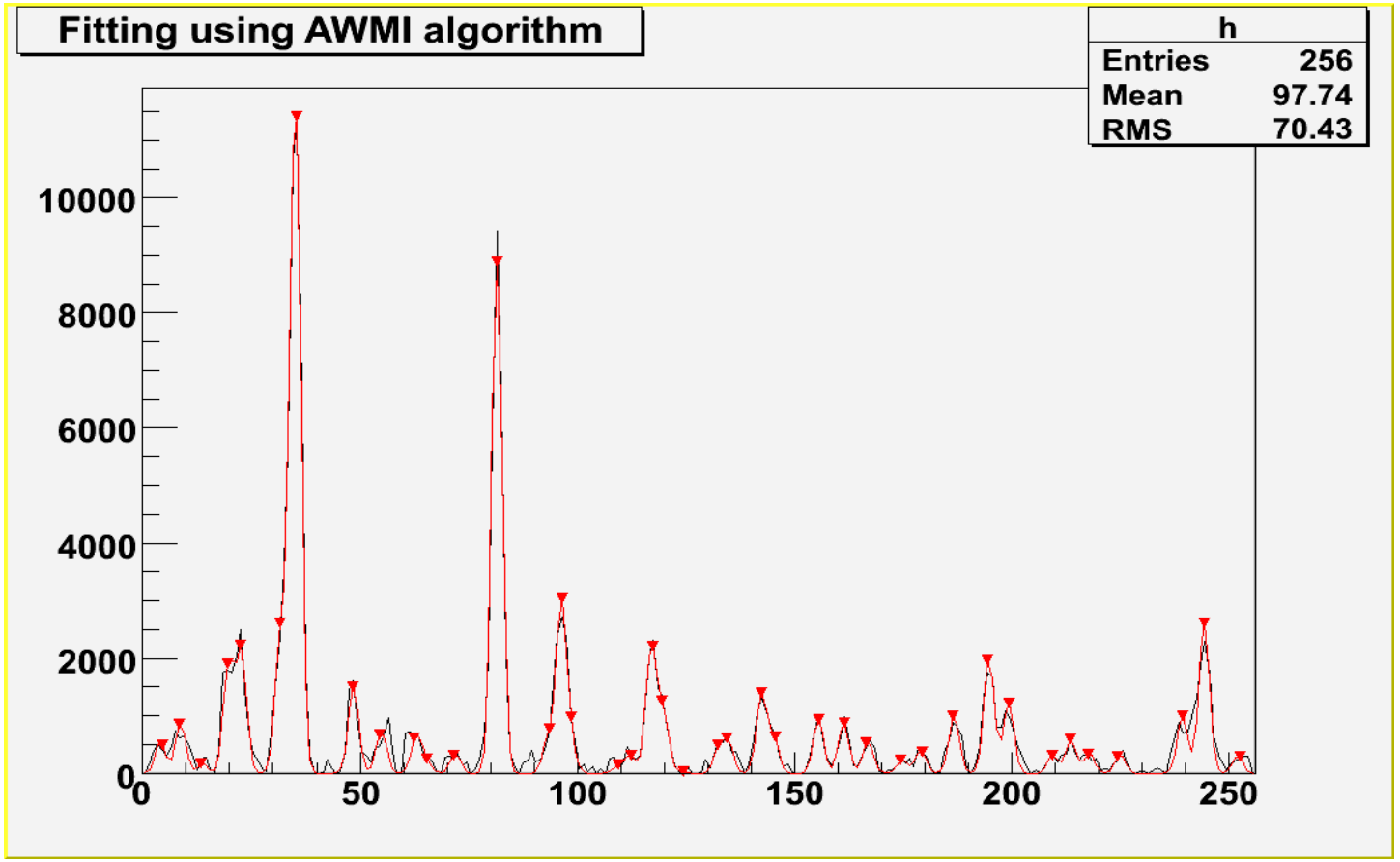


Figure 46 Original spectrum (black line) and fitted spectrum using AWMI algorithm (red line) and number of iteration steps = 1000. Positions of fitted peaks are denoted by markers

Function 10:

const char* Fit1Stiefel(float *source, TSpectrumOneDimFit * p, int size)

This function fits the source spectrum using Stiefel-Hestens method [5]. The calling program should fill in input parameters of the TSpectrumOneDimFit class. The fitted parameters are written into class pointed by TSpectrumOneDimFit class pointer and fitted data are written into source spectrum.

Parameters

•source - pointer to the vector of source spectrum
•p-pointer to the TSpectrumOneDimFit class (see Fit1Awmi function)
•size - length of source spectrum

Example 23 – script Fit1_stiefel.c:

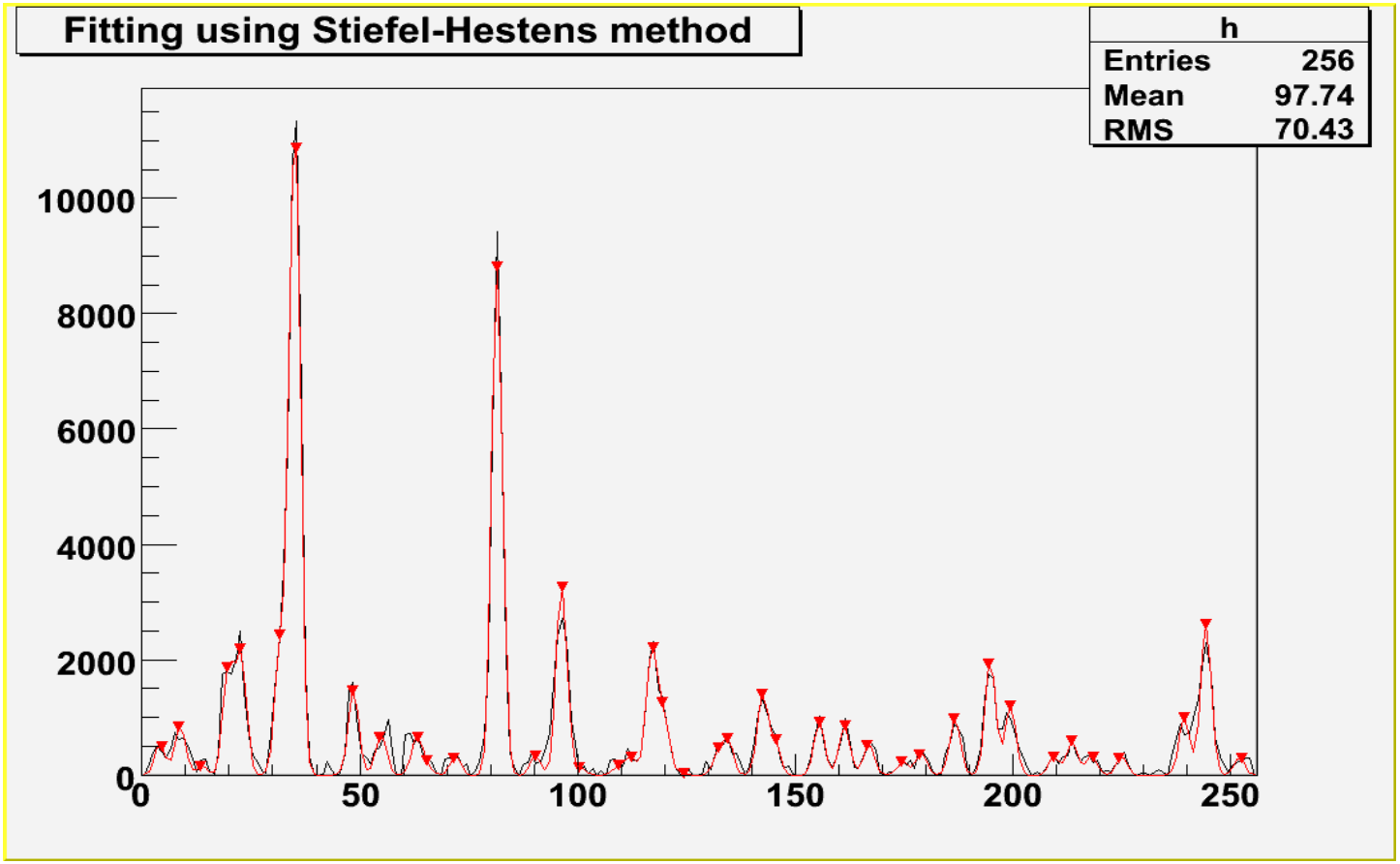


Figure 47 Original spectrum (black line) and fitted spectrum using Stiefel-Hestens method (red line) and number of iteration steps = 100. Positions of fitted peaks are denoted by markers

•converges faster than AWMI method

References:

[1] Phillps G.W., Marlow K.W., NIM 137 (1976) 525.

[2] I. A. Slavic: Nonlinear least-squares fitting without matrix inversion applied to complex Gaussian spectra analysis. NIM 134 (1976) 285-289.

[3] T. Awaya: A new method for curve fitting to the data with low statistics not using chi-square method. NIM 165 (1979) 317-323.

[4] T. Hauschild, M. Jentschel: Comparison of maximum likelihood estimation and chi-square statistics applied to counting experiments. NIM A 457 (2001) 384-401.

[5] B. Mihaila: Analysis of complex gamma spectra, Rom. Jorn. Phys., Vol. 39, No. 2, (1994), 139-148.

[6] M. Morháč, J. Kliman, M. Jandel, Ľ. Krupa, V. Matoušek: Study of fitting algorithms applied to simultaneous analysis of large number of peaks in -ray spectra. Applied Spectroscopy, Vol. 57, No. 7, pp. 753-760, 2003.

Possible modifications:

•extend the fitting algorithms for the possibility to fit peaks with different sigma

Transform methods

Goal: to analyze experimental data using orthogonal transforms

•orthogonal transforms can be successfully used for the processing of nuclear spectra
•they can be used to remove high frequency noise, to increase signal-to-background ratio as well as to enhance low intensity components [1], to carry out e.g. Fourier analysis etc.
•we have implemented the function for the calculation of the commonly used orthogonal transforms as well as functions for the filtration and enhancement of experimental spectra

Function 11:

const char *TSpectrum::Transform1(const float *source, float *dest, int size, int type, int direction, int degree)

This function transforms the source spectrum according to the given input parameters. Transformed data are written into dest spectrum.

Parameters

•source - pointer to the vector of source spectrum. Its length should be equal to the “size” parameter except for inverse FOURIER, FOUR-WALSH, FOUR-HAAR transforms. These need “2*size” length to supply real and imaginary coefficients.
•dest-pointer to the vector of dest data, its length should be equal to the “size” parameter except for direct FOURIER, FOUR-WALSH, FOUR-HAAR. These need “2*size” length to store real and imaginary coefficients
•size – basic length of source and dest spectra (should be power of 2)

•type-type of transform

Classic transforms:

TRANSFORM1_HAAR
TRANSFORM1_WALSH
TRANSFORM1_COS
TRANSFORM1_SIN
TRANSFORM1_FOURIER
TRANSFORM1_HARTLEY

Mixed transforms:

TRANSFORM1_FOURIER_WALSH
TRANSFORM1_FOURIER_HAAR
TRANSFORM1_WALSH_HAAR
TRANSFORM1_COS_WALSH
TRANSFORM1_COS_HAAR
TRANSFORM1_SIN_WALSH
TRANSFORM1_SIN_HAAR

•direction-transform direction (forward, inverse)

TRANSFORM1_FORWARD
TRANSFORM1_INVERSE

•degree-applies only for mixed transforms [2], [3], [4], range $\langle 0, \log_2 size - 1 \rangle$

Function 12:

const char *TSpectrum::Filter1Zonal(const float *source, float *dest, int size, int type, int degree, int xmin, int xmax, float filter_coeff)

This function transforms the source spectrum. The calling program should fill in input parameters. Then it sets transformed coefficients in the given region (xmin, xmax) to the given filter_coeff and transforms it back. Filtered data are written into dest spectrum.

Parameters

•source, dest, size, type, degree – meaning is the same like in Transform1 function
•xmin-low limit of filtered region
•xmax-high limit of filtered region
•filter_coeff-value which is set in filtered region

Function 13:

const char *TSpectrum::Enhance1(const float *source, float *dest, int size, int type, int degree, int xmin, int xmax, float enhance_coeff)

This function transforms the source spectrum. The calling program should fill in input parameters. Then it multiplies transformed coefficients in the given region (xmin, xmax) by the given enhance_coeff and transforms it back . Processed data are written into dest spectrum.

Parameters

•source, dest, size, type, degree – meaning is the same like in Transform1 function
•xmin-low limit of enhanced region
•xmax-high limit of enhanced region
•enhance_coeff-value by which the coefficients in enhanced region are multiplied

Example 24 – script Transform1.c:

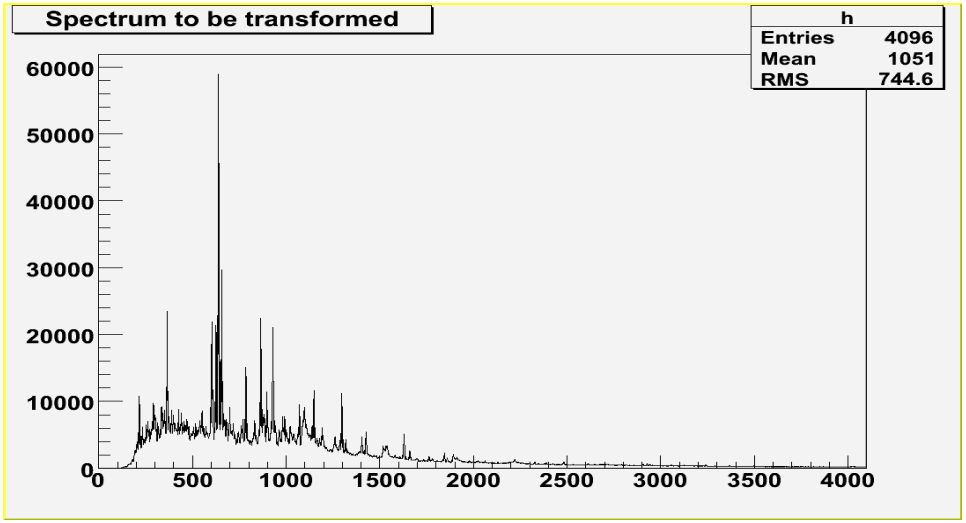


Figure 48 Original gamma-ray spectrum

Example 25 – script Transform2.c:

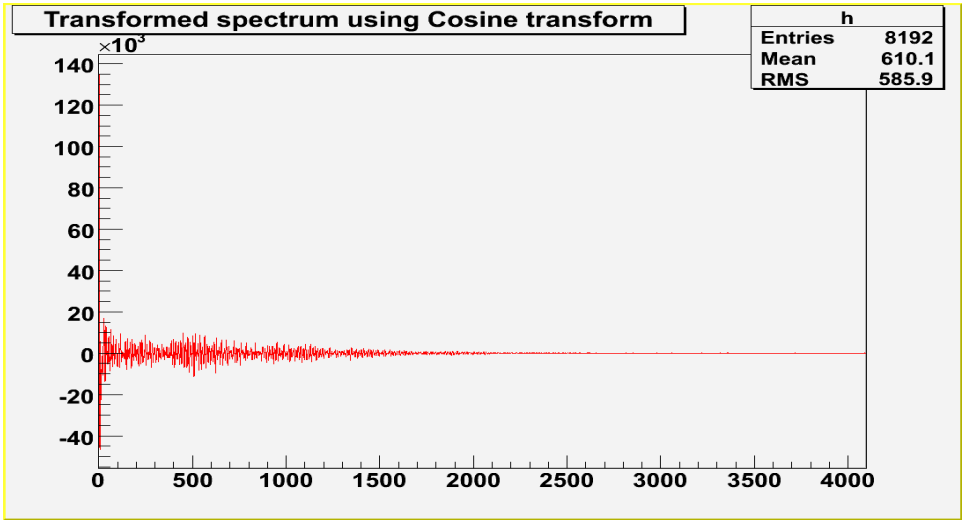


Figure 49 Transformed spectrum from Figure 48 using Cosine transform

Example 26 – script Transform3.c:

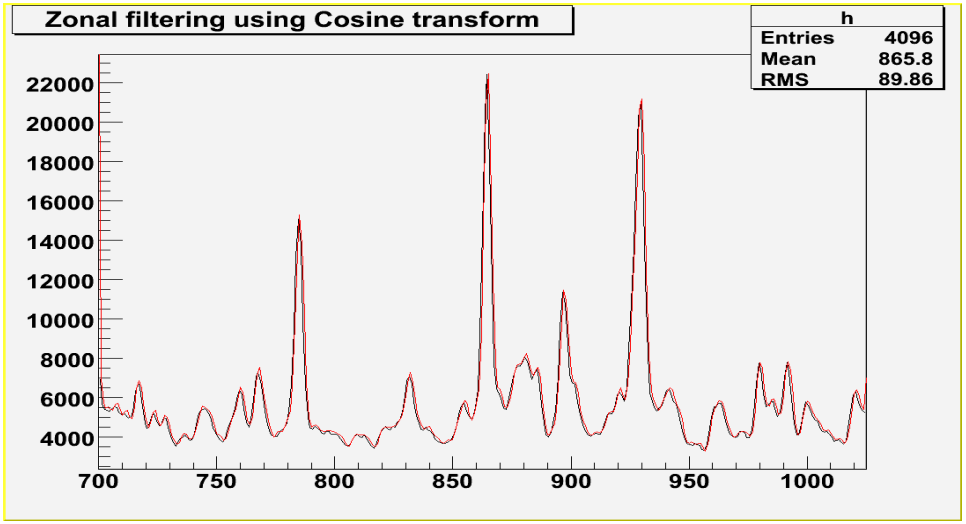


Figure 50 Original spectrum (black line) and filtered spectrum (red line) using Cosine transform and zonal filtration (channels 2048-4095 were set to 0)

Example 27 – script Transform4.c:

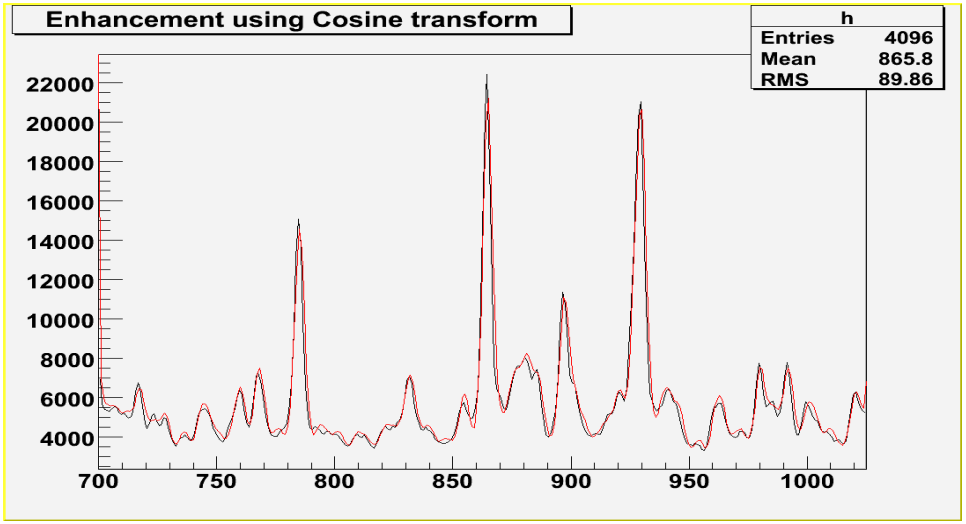


Figure 51 Original spectrum (black line) and enhanced spectrum (red line) using Cosine transform (channels 0-1024 were multiplied by 2)

References:

[1] C.V. Hampton, B. Lian, Wm. C. McHarris: Fast-Fourier-transform spectral enhancement techniques for gamma-ray spectroscopy. NIM A353 (1994) 280-284.

[2] Morháč M., Matoušek V., New adaptive Cosine-Walsh transform and its application to nuclear data compression, IEEE Transactions on Signal Processing 48 (2000) 2693.

[3] Morháč M., Matoušek V., Data compression using new fast adaptive Cosine-Haar transforms, Digital Signal Processing 8 (1998) 63.

[4] Morháč M., Matoušek V.: Multidimensional nuclear data compression using fast adaptive Walsh-Haar transform. Acta Physica Slovaca 51 (2001) 307.

Possible extensions to potential TSpectrum2, TSpectrum3 classes:

Background estimation examples:

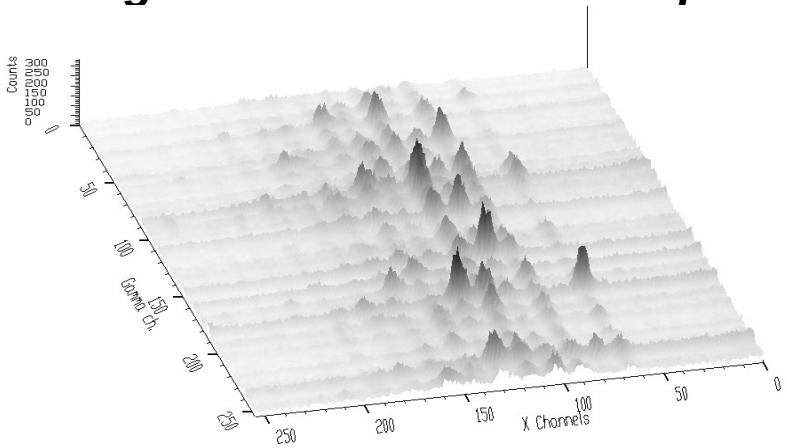


Figure 52 Original two-dimensional gamma-X-ray spectrum

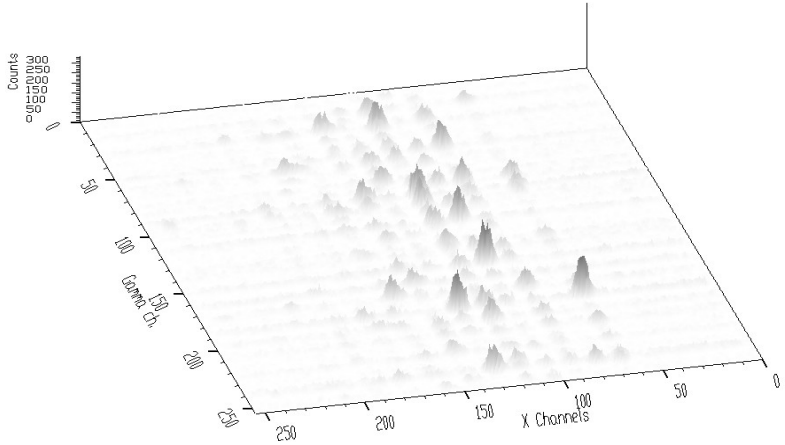


Figure 53 Spectrum from Figure 52 after background elimination

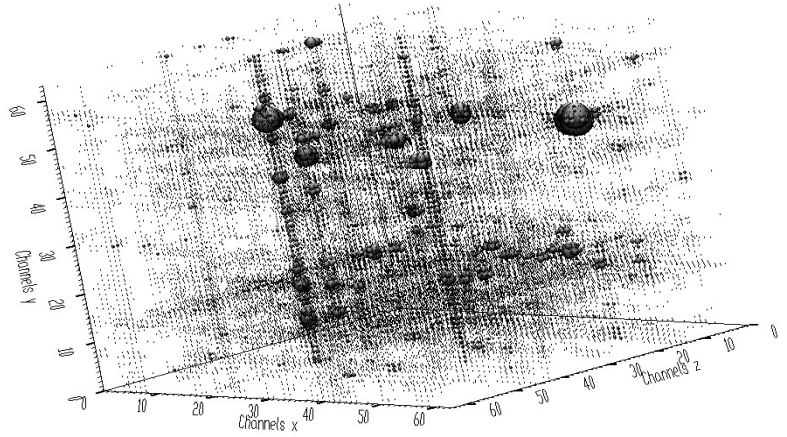


Figure 54 Original three-dimensional gamma-gamma-gamma-ray spectrum

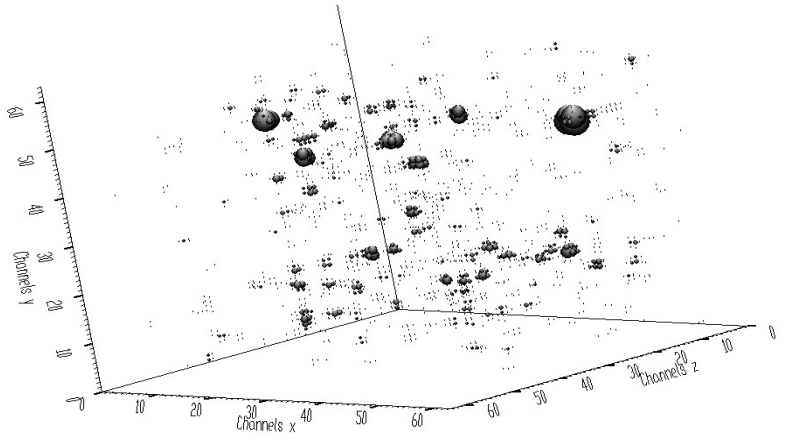


Figure 55 Spectrum from Figure 54 after background elimination

Deconvolution:

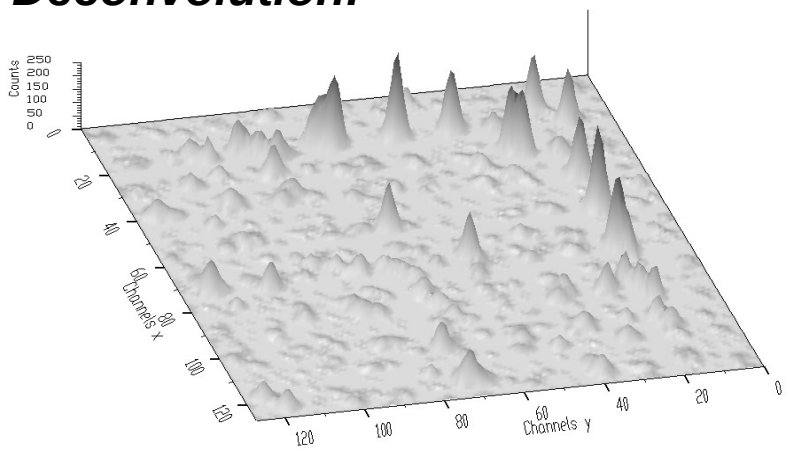


Figure 56 Part of original experimental two-dimensional gamma-gamma-ray coincidence spectrum

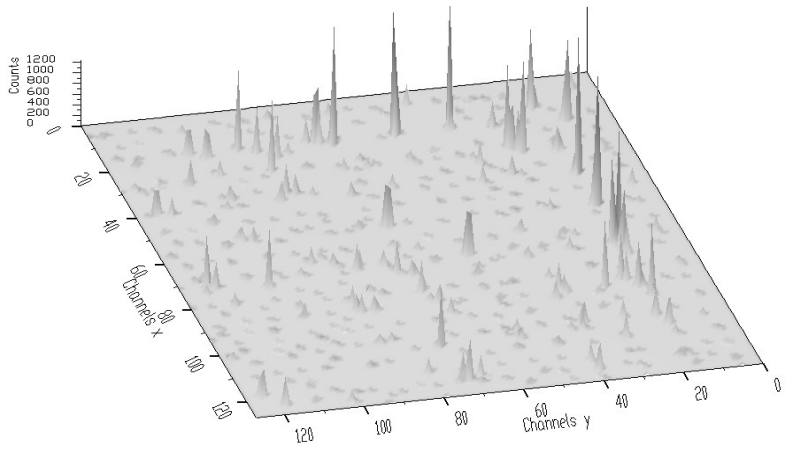


Figure 57 Spectrum from Figure 56 after boosted Gold deconvolution

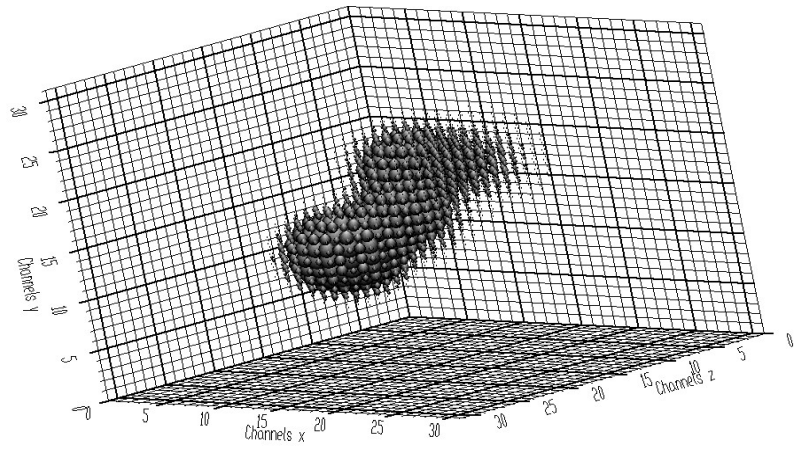


Figure 58 Three-dimensional synthetic spectrum

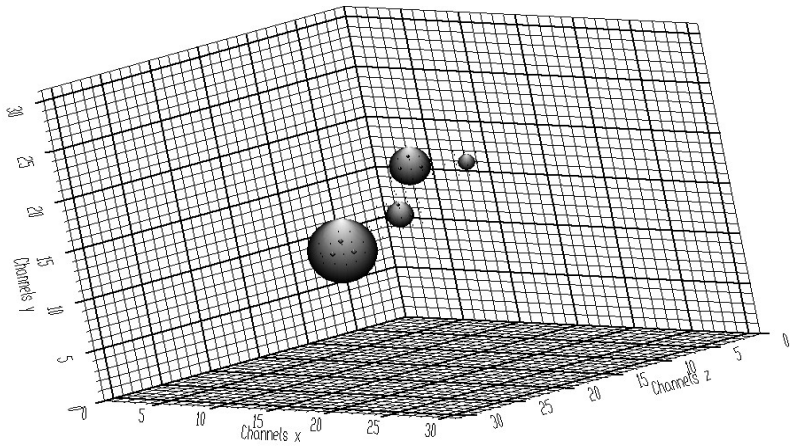


Figure 59 Spectrum from Figure 58 after boosted Gold deconvolution

Peak searching:

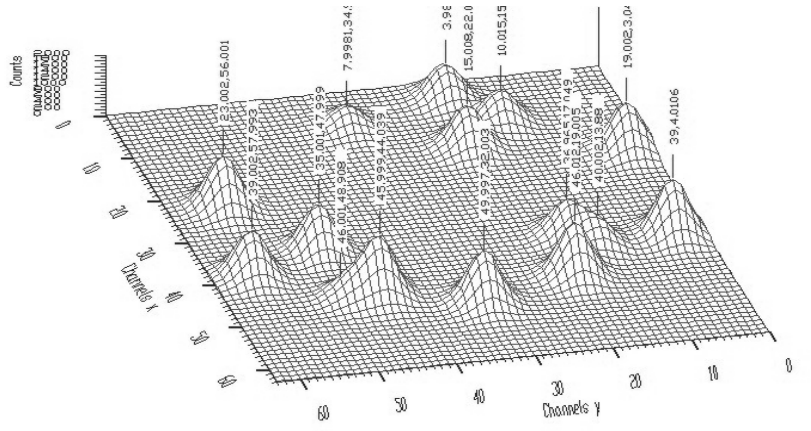


Fig 60 Result of the search using high resolution method based on Gold deconvolution

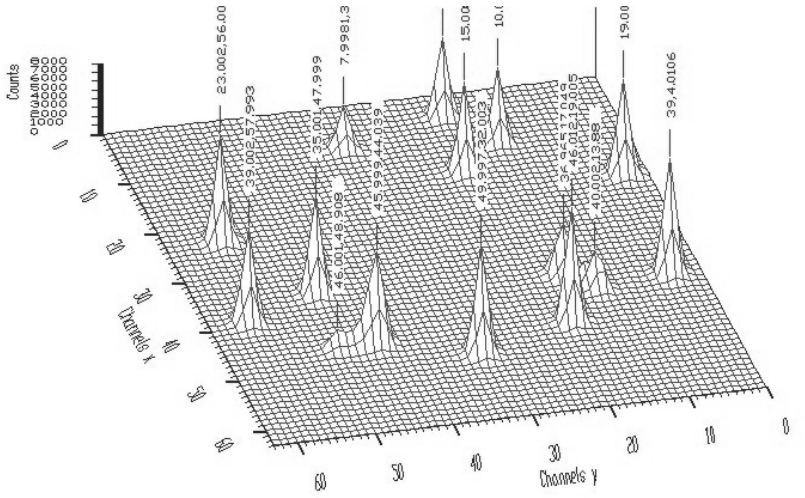


Figure 61 Deconvolved spectrum of the data from Figure 60

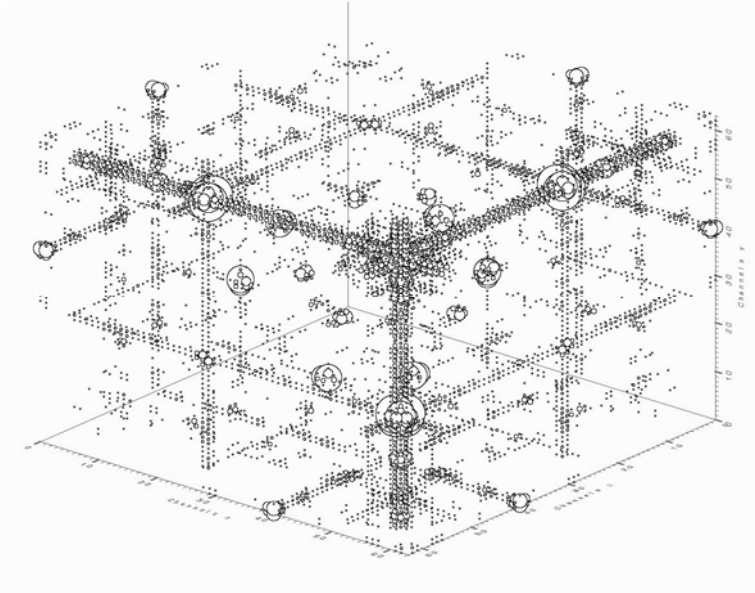


Figure 62 Three-dimensional gamma-gamma-gamma-ray spectrum

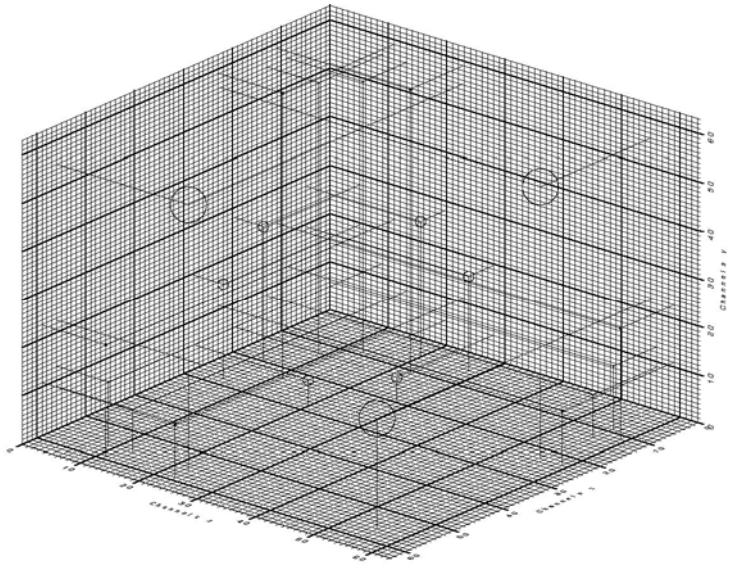


Figure 63 Identified peaks in the spectrum from Figure 62

Fitting:

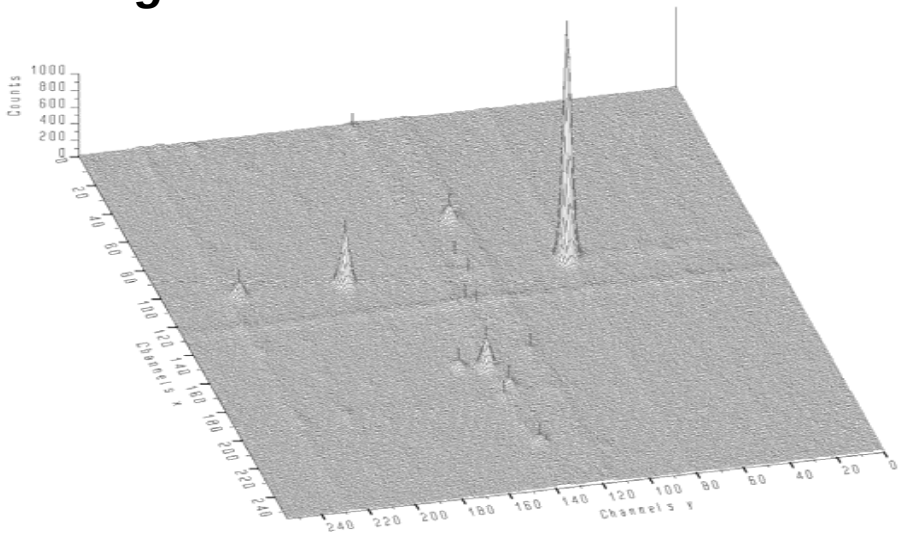


Figure 64 Two-dimensional gamma-gamma-ray spectrum with identified peaks

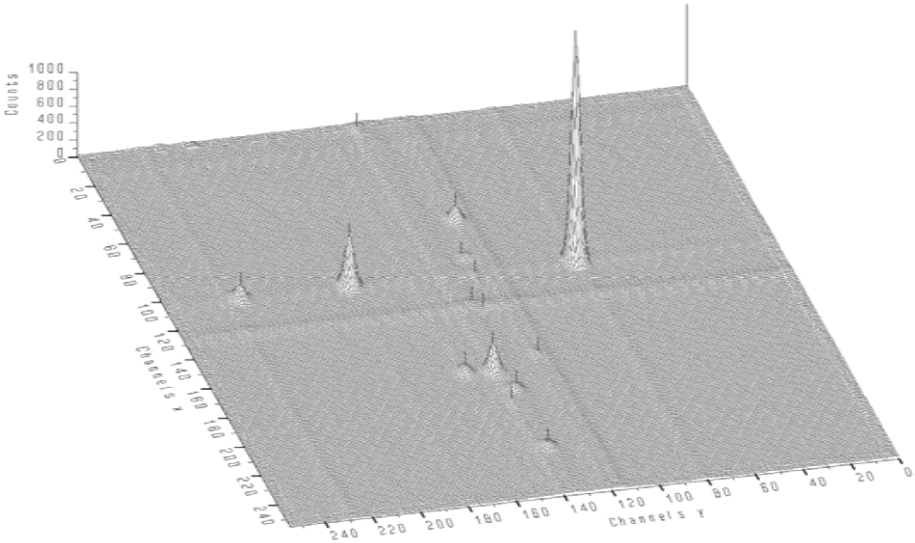


Figure 65 Fitted spectrum of the data from Figure 64

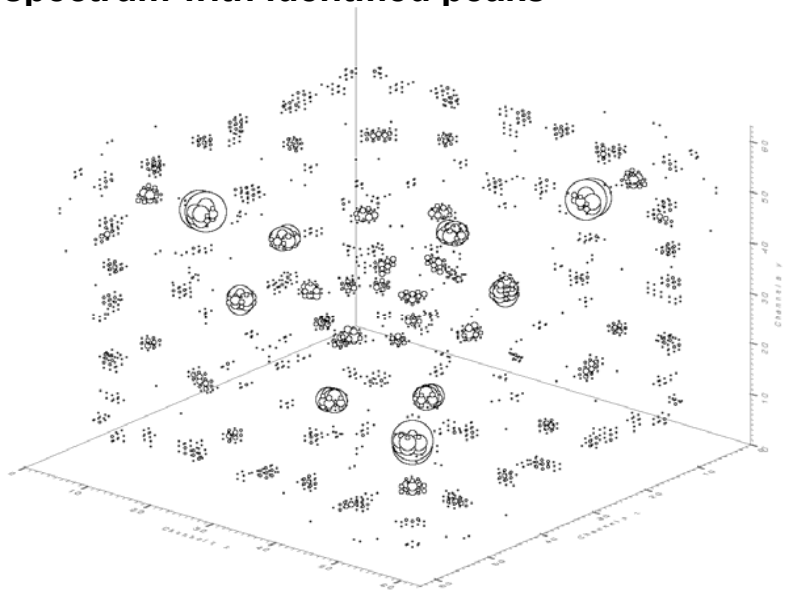


Figure 66 Three-dimensional gamma-gamma-gamma-ray spectrum after background elimination

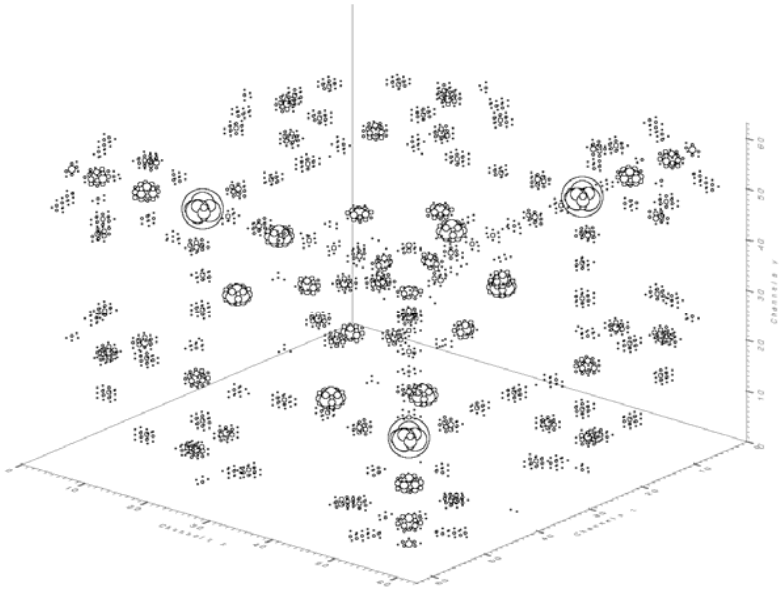


Figure 67 Fitted spectrum of the data from Figure 66

Conclusions

•we have presented a set of various methods of spectra analysis implemented in ROOT system
•presented methods are accompanied by a set of illustrative examples. File of example scripts will be available.
•we have suggested possible extensions and modifications of TSpectrum class as well as potential extensions to higher dimensions (TSpectum2, TSpectrum3 classes)
•the author would be grateful for any remarks, comments, suggestions concerning the material presented in this document
•the author would like to express thanks to Rene Brun for his assistance in the implementation of the TSpectrum class as well as for his help and suggestions during the development of peak searching method

Additional information concerning the presented algorithms can be found at sites:
http://www.fu.sav.sk/nph/projects/DaqProvis
http://www.fu.sav.sk/nph/projects/ProcFunc