About Shower Detector Software

A. Ketikyan, H. Voskanyan, B. Wojtsekhowski

September 1997.

Contents

1 Algorithms

1.1 Shower Clusters Reconstruction

The purpose of the shower clusters reconstruction in each of Preshower and Shower of the shower detector is:

- Define all clusters of fired blocks, which belong to the showers, registered in the detector;
- Calculate parameters of showers in the detector: energy deposition of showers, X and Y coordinates of the shower center;
- Set parameters and identifier of the, so called, "main" cluster.

Cluster in the shower detector is determined as follows:

— Cluster is a group of continuous blocks;

— Cluster can occupy maximum 6 (2×3) blocks in the case of Preshower and 9 (3×3) blocks in the case of Shower;

— Central block is defined as block that has maximum energy deposition.

The "main" cluster in Preshower/Shower is the cluster with the biggest energy, which is coincident with the "golden track", or coincident with some Shower/Preshower cluster. Coincidence of cluster with "golden track" means, that distance between the shower cluster center and crossing point of "golden track" with detector plane is less than a certain magnitude. Coincidence of Preshower and Shower clusters means, that distance between the clusters centers is less than a certain magnitude, in assumption, that both of this points are considered on the same Z–plane.

Shower clusters reconstruction in Preshower and Shower is performed by the following steps:

1. Sort detector fired blocks in order of its energy deposition values decreasing;
2. Pick out the block with maximum energy deposition and all fired blocks in it's arrangement (2×3 blocks for Preshower, 3×3 blocks for Shower), as belong to one cluster;
3. Remove all of blocks, associated in cluster, from following consideration;
4. Repeat steps 2 and 3, while all of fired blocks will be associated into any cluster;
5. Calculate energy deposition, X and Y coordinates of each cluster and sort clusters on its energies decreasing order;
6. Define coordinates of crossing point of "golden track" with detector plane in detector local coordinate system;

7. Analyze geometrical position of the "golden track" point on the detector plane and clusters centers in order to determine the "main" cluster and to set it parameters and identifier.

Energy deposition E, X and Y coordinates of the shower center are calculated by the formulas:

$$E = \sum_{i \in M} e_i \ , \quad X = \sum_{i \in M} e_i \cdot x_i / E \ , \quad Y = \sum_{i \in M} e_i \cdot y_i / E \ ,$$

where: i — number of detector block, included into the cluster; M — set of blocks numbers, included into the cluster; e_i — energy deposition in the block i of detector; x_i, y_i — X and Y coordinates of center of the block i of detector.

Shower clusters reconstruction algorithm described above, has been realized by program **tot_shower** (see sec.3.2).

1.2 Shower Detector Channels Calibration

The problem of shower detector channels calibration is to define a coefficients for the amplitude value transformation to the energy deposition for every Preshower and Shower modules.

Magnitudes of amplitudes from each of shower detector blocks has been equalized preliminary by the high voltage using signals from cosmic rays.

As shower detector is intended for electro-magnetic showers registration and e/π identification, for the calibration constants definition it necessary to have events, where electron was registered by the shower detector.

Total absorption of electron energy is performed in Preshower and Shower, so for electron energy definition the energy deposition in both of these detectors must be summed up.

Calibration constants are fitted by minimization of functional

$$\chi^2 = \sum_{i=1}^{N} \Big[\sum_{j \in M^i_{ps}} C_j \cdot (A^i_j - P_j) + \sum_{k \in M^i_{sh}} C_k \cdot (A^i_k - P_k) - P^i_{kin} \Big]^2 . \qquad (1)$$

Here: $i = 1 \div N$ — number of selected calibration event; j — number of Preshower block, included into the cluster, reconstructed in the i-th event; k — number of Shower block, included into the cluster, reconstructed in the i-th event; M^i_{ps} — set of Preshower blocks numbers, included into the cluster, reconstructed in the i-th event; M^i_{sh} — set of Shower blocks numbers, included into the cluster, reconstructed in the i-th event; A^i_j — amplitude value into the j-th Preshower block in the i-th event; A^i_k — amplitude value into the k-th Shower block in the i-th event; P_j — mean value of pedestal of the j-th Preshower channel; P_k — mean value of pedestal of the k-th Shower channel; P^i_{kin} — particle momentum corrected for kinematic broadening in the i-th event; C_j, C_k — Preshower and Shower blocks calibration constants to be fitted. Total number of these constants is 144 (48 of them for Preshower, and 96 for Shower).

Calibration events (i.e. events where the charged particle, detected in spectrometer, is electron with the biggest probability) are selected on the following conditions:

— Data reconstruction in all of spectrometer detectors packages was successful;

— Only one particle has been detected in the electron arm Scintillators S1 and S2;

— Only one track has been reconstructed in the VDC system;

— Sum of amplitudes of Gas Cherenkov signals is greater than a certain threshold (300 channels);

— Only one cluster has been reconstructed in each of Preshower and Shower;

— Each of reconstructed clusters are coincident with the "golden track".

Calibration events selection is realized by program **shcal_events**, calibration constants calculation — by program **shcal_dofit** (see sec.3.2).

2 For ESPACE User

2.1 ESPACE Variables

In the following table ESPACE variables are introduced, which can be used in user KUMAC file. Also, type of variable (I for integer, R for real), the internal variable corresponding to each of ESPACE variable, it number and a short description are shown in the second, third, fourth and fifth columns accordingly.

In the table, for shortness, the following assumptions are used:

— either **spec_e.preshow** or **spec_e.show** precedes to any string of the first column;

— **spdetector.sp(1).trigger.sh(ish)** (or **spdet.trigger.sh(ish)**, etc.) precedes to any string of the second column, where **spdetector** is a record of structure **/allspectrodet/** (**spdet** is a record of structure **/onespectrodet/**), **ish**=1 for Preshower and **ish**=2 for Shower.

For instance, the first line of the table means that value of external ESPACE variable **spec_e.preshow.mult** is taken equal to value of internal variable **spdetector.sp(1).trigger.sh(1).nhits.adc**, as well as **spec_e.show.mult** is the same as internal variable **spdetector.sp(1).trigger.sh(2).nhits.adc**, etc.

ESPACE var.	T	Internal variable	N	Description
.mult	I	.nhits.adc	3	Number of hits on detector.
.clus	I	.ncluster	4	Number of clusters on detector.
.x_det	R	.x	23	Coordinates of crossing point of golden
.y_det	R	.y	24	track with detector plane in local c.s.
.adc[[i];[i]]	I	.block(i).val(1).adc	47	ADC values of detector blocks.
.adc_c[[i];[i]]	I	.block(i).cor_val(1).adc	48	Corrected ADC values of detector blocks.
.x_array[[k];[k]]	R	.clust(k).x	85	X-coordinates of clusters in detector c.s.
.y_array[[k];[k]]	R	.clust(k).y	86	Y-coordinates of clusters in detector c.s.
.e_array[[k];[k]]	R	.clust(k).energy	87	Energy depositions of clusters in detector.
.m_array[[k];[k]]	I	.clust(k).mult	88	Number of blocks of clusters.
.part_id	I	.iden	89	Identifier of the "main" cluster.
.cl_e	R	.ecl	90	Energy deposition of the "main" cluster.
.cl_x	R	.xcl	91	X-coordinate of the "main" cluster.
.cl_y	R	.ycl	92	Y-coordinate of the "main" cluster.
.rate[[i];[i]]	R	—	100	Don't used.
.adcsum	I	—	258	Sum of ADC values of blocks.
.adcsum_c	I	—	259	Sum of corrected ADC values of blocks.
.ana_adcsum	I	—	260	Don't used.

Indexes i and k in the table mean the following:

— i shows detector block number and has values $i = 1 \div 48$ for Preshower and $i = 1 \div 96$ for Shower;

— k shows cluster number, reconstructed on detector, and has values $k = 1\div$ **spdetector.sp(1).trigger.sh(ish).ncluster**.

2.2 Calibration Initiation

Preshower and Shower channels calibration procedure has been included into ESPACE. A new option "s" of the Espace/File/scan command added.

Calibration process is initiated, when a File/scan command with option "s" is executed. The first and the last entries given in this command show the range of entries, among which events for calibration will be selected. It can be used by the different ways.

One way is to execute a File/scan command with option "s" before the basic scanning command, and then during the basic scan the new calculated calibration constants will be used (see example below).

Another way is to include "s" option in the basic scanning command. In this way the calibration constants are calculated using the data of all of scanning entries and are used for the next seance of scanning of the same or another data file.

Of course, the second way allows to calculate calibration constants on the big statistics, which gives more quantity of fired shower detector blocks and hence more block constants computed. But for choosing of one of these ways one must take into account CPU time and disk free memory, which are required for calibration performance. The following table may be useful to choose one of the way for calibration and/or to choose the optimal number of entries in the calibration command.

Entries	CPU time	Disk memory
5000	44 sec	≈ 160 KB
10000	1.45 min	≈ 330 KB
20000	3.33 min	≈ 650 KB
50000	8.37 min	≈ 1600 KB
100000	≈ 17 min	≈ 3300 KB

According to us the optimal is approximately 5000–50000 entries.

A simple example of KUMAC file for shower detector calibration and then Shower clusters energies distribution saving, approximately looks as follows.

```
*****************************************************
define/batch on
*********************** Setup files *******************
set/file/header hdr_1690
set/file/output spec_1s1690.hbook
set/file/database db_coin_new
set/file/detmap detmap.config
************* Shower Detector Calibration *************
file/scan e89003_1690.dat FIRST=1 LAST=5000 -s
***************** Scanning and output ****************
spectra/save spec_e.show.cl_e
file/scan e89003_1690.dat FIRST=1 LAST=10000
spectra/write
return
*****************************************************
```

After the necessary files definition commands, the calibration constants calculation is started by the **File/scan e89003_1690.dat FIRST=1 LAST=5000 -s** command, which is performed on two stages.

On the first stage, entries from 1 to 5000 are scanned, and calibration events are selected among these entries. The calibration events are selected such, that the charged particle is electron with the biggest probability. The calibration events data (momentum of particle, fired blocks numbers and corresponding amplitudes values in these blocks) are written into the temporary file **shcal_events.dat_tmp** in the current directory.

On the second stage, the calibration constants of the channels of Preshower and Shower are calculated (using the calibration events data) and saved into the two disk files: **shower_calib.dat** and **shower_calib_r**XXXX**.dat** (where XXXX is the run number of currently data processed data file).

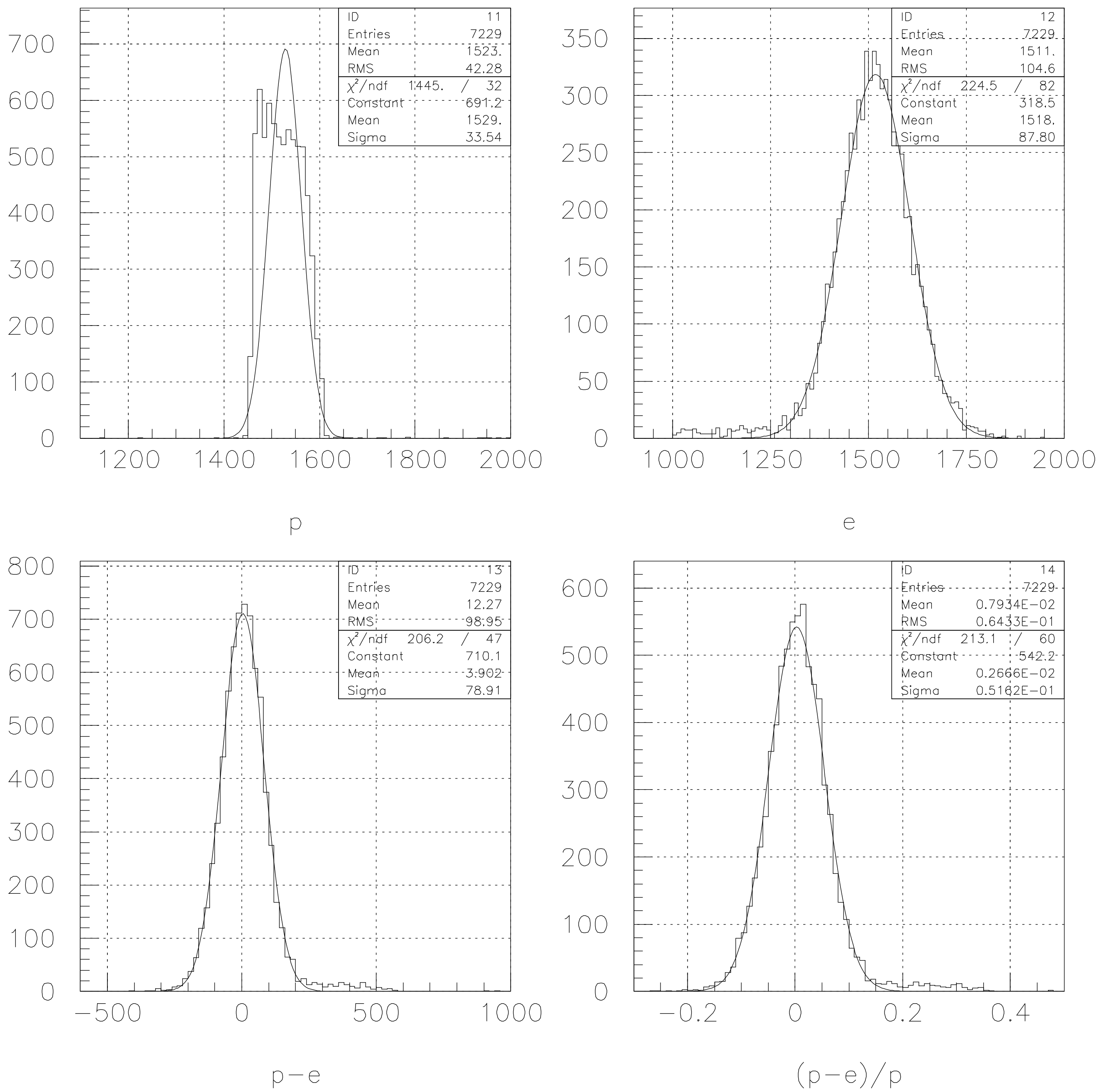

Figure 1: In these distributions: p — particle momentum (MeV/c), e — total energy deposition in shower detector (MeV).

Each of files **shower_calib.dat** and **shower_calib_r**XXXX**.dat** contains the same information (Calibration constants values, mean Pedestal and its Sigma-s values for all Preshower and Shower channels) and occupies 4824 bayt of disk memory.

Two similar calibration constants files creation allows to use constants during any run data processing, by following: 1) If calibration for this run has been done previously, then use the run's own calibration constants; 2) Otherwise, one can use the latest calculated constants (see program **read_show_coef**).

In present it assumed, that **shower_calib.dat** and **shower_calib_r**XXXX**.dat** files, also file of the shower detector channels Pedestals mean values **shower_peds.dat**, are the current directory files (see programs **read_show_coef**, **read_show_peds** and notes to these programs).

After the shower detector calibration, KUMAC program described before saves spectrum of energy deposition in the Shower, scans the first 10000 entries of the data file, writes spectrum into the disk and exits to the Espace main menu.

Note, that the second File/scan command is executed without "s" option. In this case, neither calibration events selection no calibration constants calculation is made. So, user can include or no calibration command in own KUMAC program in dependence of disk quota, or cpu limit possibilities, or other reasons.

If user don't want make shower detector calibration, he can use either the already existing constants file, or built in program **read_show_coef** ones.

Distributions, histograms of which are shown in Fig.1, have been created for 10000 entries of data run 1690. In this distributions **p** is particle momentum corrected by kinematics and **e** is sum of energy depositions of particle in Preshower and Shower.

Calibration constants, using of which these energy depositions are computed, have been calculated by calibration events, selected among the first 100000 entries of data file.

2.3 Program Messages

During the calibration procedure execution, a few information messages print-out is possible, which perhaps can be useful for the calibration process following. In these messages, list of which is introduced below, the program module name is distinguished by capital letters.

!-READ_SHOW_COEF-Inf-File "shower_calib_rXXXX.dat" found. — Where XXXX is the number of run, data of which is processed currently, and own calibration constants file of which was found.

!-READ_SHOW_COEF-Inf-File "shower_calib_rXXXX.dat" not found. — Own calibration constants file of run number XXXX was not found.

!-READ_SHOW_COEF-Inf-File "shower_calib.dat" found. — Own calibration constants file of run was not found, but a file **shower_calib.dat**, in which the latest computed constants is written, was found.

!-READ_SHOW_COEF-Inf-Data from Data Statement. — Neither own calibration constants file nor a file **shower_calib.dat** was found, and constants was taken from the inner data statement.

!-READ_SHOW_PEDS-Inf-File "shower_peds.dat" found. — Pedestals and its Sigmas values was read from the corresponding file and was set into the corresponding structure of data.

!-READ_SHOW_PEDS-Inf-File "shower_peds.dat" not found. — The pedestals file was not found, and hence, no changes in the data structure was made.

!-SHCAL-Inf-Save coef-s to file "shower_calib.dat". — Calibration constants fitting was successful and constants was saved into the disk file.

!-SHCAL-Inf-Save coef-s to file "shower_calib_rXXXX.dat". — Calibration constants fitting was successful and constants was saved into the own disk file of run number XXXX.

!-SHCAL-Inf-Constants not fitted and not saved ! — Calibration constants fitting was not successful and no constants was saved into any disk file.

!-SHCAL-Inf-Coefficient of block NNN is bad! Fix it old value and repeat fit. — Calibration constant of the block NNN went to minimum (0.00001) or maximum (10) boundary value during fit. This constant value will be fixed to it old value, and fitting will be repeated again.

3 For ESPACE Programmer

3.1 ESPACE Internal Variables

All of internal variables used for shower detector software programming into ESPACE, are formed in form of structure. This structure, named **/shower/** was inserted into FORTRAN include file **espace_type.h**.

In the following tables for every used variables (in the first column) it fortran type (in the second column), program module name where variable is set (the third column) and brief description of variable (in the fourth column) are are shown.

In the table the following designations are used:

— **spdetector.sp(1).trigger.sh(ish)** (or **spdet.trigger.sh(ish)**, etc.) precedes to a variable name in the first column, where **spdetector** is a record of structure **/allspectrodet/** (**spdet** is a record of structure **/onespectrodet/**), **ish**=1 for Preshower and **ish**=2 for Shower.

— In the variable type column **I** means an integer type, **D** — real*8 type, and **L** — logical type.

— Index i shows detector block number and has values $i = 1 \div 48$ for Preshower and $i = 1 \div 96$ for Shower.

— Index n points to the fired blocks of detector and has values $n = 1\div$ **spdetector.sp(1).trigger.sh(ish).nblock.adc**.

— Index j shows number of hit on detector and has values $j = 1\div$ **spdetector.sp(1).trigger.sh(ish).nhits.adc**.

— Index k shows cluster number, reconstructed in detector, and has values $k = 1\div$ **spdetector.sp(1).trigger.sh(ish).ncluster**.

— Index l points to blocks of reconstructed clusters and has values $l = 1\div$ **spdetector.sp(1).trigger.sh(ish).clust(k).mult**.

Variable	T	Where is set	Description
hitbit	I	user_init	Hitbit assigned to the detector.
adc_range	I	user_init	Maximal value of range of ADC amplitudes.
geom.p.x	D	read_geometry	X-coordinate of the detector center.
geom.p.y	D	read_geometry	Y-coordinate of the detector center.
geom.p.z	D	read_geometry	Z-coordinate of the detector center.
geom.nblock	I	read_geometry	Number of detector blocks.
geom.block(i).p.x	D	read_geometry	X-coordinates of detector blocks centers.
geom.block(i).p.y	D	read_geometry	Y-coordinates of detector blocks centers.
geom.block(i).p.z	D	read_geometry	Z-coordinates of detector blocks centers.
cor(i).offset.adc	D	shcal_coef_io	Mean values of channels Pedestals.
cor(i).amplitude.adc	D	shcal_coef_io	Values of calibration constants.
cor(i).offset_sigma.adc	D	shcal_coef_io	Values of Pedestals σ.

The following parts of the structure is overwritten for every hit.

Variable	T	Where is set	Description
nblock.adc	I	read_show_e	Number of blocks that fired.
blocks(n).adc	I	read_show_e	Array of block numbers that fired.
nhits.adc	I	read_show_e	Number of hits on detector.
blocknr(j).adc	I	read_show_e	Pointer to block number for each hit.
hitnr(j).adc	I	read_show_e	Pointer to hit number in a block for each hit.
block(i).fired.adc	L	read_show_e	Flag which shows if block was fired.
block(i).nrhit.adc	I	read_show_e	Number of hits on the block.
block(i).val(1).adc	I	read_show_e	ADC values of detector blocks.
block(i).cor_val(1).adc	D	read_show_e	Corrected ADC values of detector blocks.

Next part of the structure is overwritten after the shower clusters reconstruction in the detector.

Variable	T	Where is set	Description
x	D	tot_shower	Coordinates of crossing point of golden
y	D	tot_shower	track with detector plane in local c.s.
ncluster	I	tot_shower	Number of clusters determined on detector.
clust(k).x	D	tot_shower	X-coordinates of clusters in detector c.s.
clust(k).y	D	tot_shower	Y-coordinates of clusters in detector c.s.
clust(k).energy	D	tot_shower	Energy depositions of clusters in detector.
clust(k).mult	I	tot_shower	Number of blocks inside of clusters.
clust(k).cell(l).cell_num	I	tot_shower	Block number in the particular cluster.
clust(k).cell(l).cell_val	D	tot_shower	Part of the energy inside of the block.
iden	I	tot_shower	Identifier of the "main" cluster.
ecl	D	tot_shower	Energy deposition of the "main" cluster.
xcl	D	tot_shower	X-coordinate of the "main" cluster.
ycl	D	tot_shower	Y-coordinate of the "main" cluster.

3.2 ESPACE Program Modules

In this section programs and fragments of programs, concerning to shower detector software are described. Here, for shortness, the designation **shwr(ish)** is used instead of **spdet.trigger.sh(ish)**, where **ish**=1 for Preshower and **ish**=2 for Shower. (See structures **/onespectrodet/**, **/trig_package/**, **/shower/** in **espace_type.h** file.)

logical function shcal_coef_io (io,fname) (file **read_show_coef.f**) — Program for the shower detector channels Pedestals, its Sigma-s and Calibration coefficients Input/Output from/to the given disk file into/from the corresponding structure. When data input has been required, then program tries to read the coefficients from the given file. If the given file doesn't exist, then program builtin data into the structure is set. These builtin coefficients have been calculated using the data of run 1690.

Input:

io — Option indicating Input or Output action:
  = 0 — Input data into the structure from the given file;
  = 1 — Output data from the structure into the given file.

fname — File name from/to which data must be inputed/outputed.

shwr(ish).geom.nblock — Number of the detector modules.

Output:

shcal_coef_io — Indicates the operation result:
  = $.true.$ — The data have been read/written from/to the given disk file;
  = $.false.$ — Open or Write error on the given file.

Input/Output:

shwr(ish).cof(i).offset.adc — Mean value of the i-th channel pedestal — Output, if **io**=0, and Input, if **io**=1, to/from structure **/shower/**.

shwr(ish).cor(i).offset_sigma.adc — Value of the i-th channel pedestal σ — Output, if **io**=0, and Input, if **io**=1, to/from structure **/shower/**.

shwr(ish).cor(i).amplitude.adc — Value of the i-th channel calibration constant — Output, if **io**=0, and Input, if **io**=1, to/from structure **/shower/**.

Is Called From: **read_show_coef**, **shcal_dofit**.

Note: Program assumes, that **fname** is a name of the current directory file. In order to prevent a repetition of the same calibration coefficients computation by the various users, one can create a common usage directory and save there all created coefficients files. In this case into the inner character variable **path** can be set this directory path name.

subroutine read_show_coef (file **read_show_coef.f**) — Read from the disk file and set into the corresponding data structure the shower detector channels Pedestals, its Sigma-s and Calibration coefficients. Firstly program tries to read coefficients from the **shower_calib_r**XXXX**.dat** file (where XXXX is a run number). If this file doesn't exist, then program tries to read the **shower_calib.dat** file. If this file doesn't exist too, then program builtin data is set into the structure.

Input:

run_number — Number of currently processing run (see **run_info.h**).

Output:

shwr(ish).cor(i).offset.adc — Mean value of the i-th channel pedestal.

shwr(ish).cor(i).offset_sigma.adc — Value of the i-th channel pedestal σ.

shwr(ish).cor(i).amplitude.adc — Value of the i-th channel calibration constant.

Used Subroutine: **shcal_coef_io**.

Is Called From: **read_show_e**.

Note 1: Program assumes, that **shower_calib_r**XXXX**.dat** and **shower_calib.dat** are the current directory files (see also the Note to the **shcal_coef_io** program).

Note 2: XXXX — is a 4 digital run number.

subroutine read_show_peds (file **read_show_peds.f**) — Read from the disk file **shower_peds.dat** and set into the corresponding data structure the mean pedestal values and Sigma-s of the shower detector channels. If file of pedestals doesn't exist, then no changes in the data structure is made.

Input:

shwr(ish).geom.nblock — Number of the detector modules.

Output:

shwr(ish).cor(i).offset.adc — Mean value of the i-th channel pedestal.

shwr(ish).cor(i).offset_sigma.adc — Value of the i-th channel pedestal σ.

Is Called From: **shcal_dofit**.

Note: Program assumes, that **shower_peds.dat** is the current directory file. Pedestals file can be saved into the common usage directory together with the calibration coefficients files (see also the Note to the **shcal_coef_io** program).

logical function read_show_e (raw,wdctr,spdet) (file **read_show_e.f**) — Extract shower detector data from the raw data read out, correct it and set into the corresponding structure **/shower/**. For every block, data correction is made by formula $A^i_{cor} = (A^i_{raw} - P^i) \times C^i$, where: i — block number, A^i_{raw} — raw ADC value from the block, P^i — pedestal value, C^i — calibration constant, A^i_{cor} — corrected ADC value. (So, the i-th block data correction means determination of energy deposition in the block.) For this, the calibration constants reading program **read_show_coef** is called only once for each new run during the first entrance into the **read_show_e**.

<u>Input:</u>

raw — Is a record of structure **/raw_total/** (see **raw_type.h**).
wdctr — Is a record of structure **/word_counters/** (see **raw_type.h**).
spdet — Is a record of structure **/onespectrodet/** (see **espace_type.h**).
run_number — Number of currently processing run (see **run_info.h**).
rawdata_input — Flag, which shows either raw data input is done (if $.true.$), or not (if $.false.$). (See **communications.h**).
raw.shower1(100).block – Array of Preshower fired blocks numbers (**raw_type.h**).
raw.shower1(100).data — Array of amplitudes values from ADC in Preshower fired blocks (see **raw_type.h**).
raw.shower2(100).block — Array of Shower fired blocks numbers (see **raw_type.h**).
raw.shower2(100).data — Array of amplitudes values from ADC in Shower fired blocks (see **raw_type.h**).
raw.shower1 — Preshower read out word counter (see **raw_type.h**).
raw.shower2 — Shower read out word counter (see **raw_type.h**).
spdet.index — Index to the spectrometer records (= 1 for the electron arm records).
spdet.hitbit(64) — Array of detectors hitbits.
shwr(ish).hitbit — Hitbit assigned to the shower detector.
shwr(ish).geom.nblock — Number of the detector modules.
shwr(ish).adc_range — Range of ADC amplitudes values.
shwr(ish).cor(i).offset.adc — Mean value of the i-th channel pedestal.
shwr(ish).cor(i).offset_sigma.adc — Value of the i-th channel pedestal σ.
shwr(ish).cor(i).amplitude.adc — Value of the i-th channel calibration constant.

<u>Output:</u>

read_show_e — Is always $.true.$ when this function is called.
shwr(ish).nhits.adc — Number of hits read out.
shwr(ish).nblock.adc — Number of blocks that fired.
shwr(ish).block(i).fired.adc — Flag, which shows either the block number i was fired (if $.true.$), or not (if $.false.$).
shwr(ish).block(i).nrhit.adc — Number of hits on the i-th fired block.
shwr(ish).block(i).val(1).adc — Raw ADC value from the i-th fired block.
shwr(ish).block(i).cor_val(1).adc — Corrected ADC value (i.e. energy deposition) in the i-th fired block.
shwr(ish).blocks(i).adc — Number of the i-th fired block.
shwr(ish).blocknr(i).adc — Pointer to block number for the i-th hit.
shwr(ish).hitnr(i).adc — Pointer to hit number in a block for the i-th hit.

<u>Used Subroutines:</u> **read_show_coef**, **which_show_block**.

Is Called From: **interpret_event_coin**.

integer function which_show_block (ispectro,ish,tdc_info,ihit) (file **which_show_block.f**) — Define the shower detector block number for the given hit.
Input:
ispectro — Index to the spectrometer arm (= 1 for the electron arm).
ish — Index to the Preshower (if = 1) or the Shower (if = 2) detector.
tdc_info — Flag, which shows either TDC (if $.true.$), or ADC (if $.false.$) hit information is considered.
ihit — The hit number, for which the block number definition is required.
shwr(ish).geom.nblock — Number of the detector modules.
shwr(ish).block(i).nrhit.adc — Number of hits on the i-th fired block.
shwr(ish).blocks(i).adc — Number of the i-th fired block.
Output:
which_show_block — Defined block number for the given hit.
Is Called From: **read_show_e**.

subroutine tot_shower (sptr,spdet) (file **tot_shower.f**) — Program for showers clusters reconstruction in Preshower and Shower. This program determines number of blocks, energy deposition and coordinates of center for every cluster defined in the shower detectors. Also, analyzes geometrical position of the "golden track" point on the detector plane and Preshower and Shower clusters centers in order to determine the "main" cluster and to set it parameters and identifier.
Input:
sptr — Is a record of structure **/onespectrotr/** (see **espace_type.h**).
spdet — Is a record of structure **/onespectrodet/** (see **espace_type.h**).
sptr.particle.fp_h.x, ...fp_h.y, ...fp_h.z — Particle X, Y and Z coordinates in the detector hut coordinate system (in m).
spdet.hitbit(64) — Array of detectors hitbits.
shwr(ish).hitbit — Hitbit assigned to the shower detector.
shwr(ish).geom.p.x, ...p.y, ...p.z — X, Y and Z coordinates giving the detector position (in m).
shwr(ish).geom.block(96).p.x, ...p.y, ...p.z — X, Y and Z coordinates of the detector blocks in detector local coordinate system (in m).
shwr(ish).nhits.adc — Number of hits read out.
shwr(ish).blocks(96).adc — Array of fired blocks numbers.
shwr(ish).block(96).cor_val(2).adc — Energies in fired blocks.
Output:
shwr(ish).iden — Identifier of the "main" cluster:
- $= k \geq 0$ — The $(k+1)$-th cluster corresponds to the "golden track";
- $= -1$ — Shower and Preshower clusters are coincident, but are not coincident with the "golden track";
- ≤ -2 — In other cases.

shwr(ish).ecl, shwr(ish).xcl, shwr(ish).ycl — Energy deposition (in MeV), X and Y coordinates (in m) of the "main" cluster.
shwr(ish).x, shwr(ish).y — X and Y coordinates (in m) of the "golden track" in the detector local coordinate system.
shwr(ish).ncluster — Number of clusters determined.
shwr(ish).clust(i).mult — Number of blocks in the i-th cluster.
shwr(ish).clust(i).x, shwr(ish).clust(i).y, shwr(ish).clust(i).energy — X, Y coordinates in the detector coordinate system (in m) and energy deposition (in MeV) of the i-th cluster.
shwr(ish).clust(i).cell(j).cell_num, shwr(ish).clust(i).cell(j).cell_val — Block number and energy deposition in the j-th block of the i-th cluster.
Used Subroutine: **det_vertex**.
Is Called From: **spectrometer_reconstruct**.

subroutine shcal_events (sptr,spdet) (file **shcal.f**) — Select events for shower detector channels calibration and write these (calibration) events into the disk file. The calibration events are selected such, that the charged particle is electron with the biggest probability (see sec.1.2).
Input:
sptr — Is a record of structure **/onespectrotr/** (see **espace_type.h**).
spdet — Is a record of structure **/onespectrodet/** (see **espace_type.h**).
shcal_flagfit — Flag, which shows either the shower detector calibration required, (if *.true.*), or not (if *.false.*) in the Espace/File/scan command.
sptr.ok — Flag, which shows either data reconstruction in all of spectrometer detectors packages was successful, (if *.true.*), or not (if *.false.*).
shwr(ish).ncluster — Number of clusters determined in the shower detector.
shwr(ish).iden — Identifier used for the "main" cluster selection.
spdet.vdc.ntrack — Number of tracks reconstructed in the VDC system.
spdet.trigger.sc(isc).npad — Number of paddles that fired on the isc-th Scintillator ($isc = 1$ or 2).
spdet.trigger.gas.analog_adcsum — Sum of amplitudes of Gas Cherenkov signals.
sptr.particle.pmom_kin — Particle momentum corrected for kinematic broadening.
shwr(ish).clust(1).mult — Number of blocks in the cluster.
shwr(ish).clust(1).cell(j).cell_num — Blocks numbers in the cluster.
shwr(ish).block(i).val(1).adc — Raw ADC value from the i-th fired block of the cluster.
Output:
shcal_events.dat_tmp — The temporary text file, into which the following data of each of selected calibration events are written: momentum of particle, number of blocks in the Preshower cluster, amplitude values in each of these blocks, number of blocks in the Shower cluster, amplitude values in each of these blocks. This file is used by the calibration constants fitting program **shcal_dofit** later.
Is Called From: **process_data**.

subroutine shcal_dofit (file **shcal.f**) — Calculate the calibration constants using the data of the calibration events selected before by program **shcal_events**. Constants calculation is proceeded by the following.

Firstly, program reads the calibration events and defines the number of hits in each of shower detector blocks. For blocks, hits number in which is less than a number **minblev**, the value of calibration constant is taken from the structure **/shower/** and is fixed, i.e. is not recalculated by the following fitting procedure. (Initial values of all constants into the structure **/shower/** is set before, during the data scanning by program **read_show_coef**.)

Secondly, program reads into the structure **/shower/** the mean pedestal values and Sigma-s of the shower detector channels by program **read_show_peds** calling. Repeated setting of pedestals into the structure is done with the purpose to have the most appropriate (on the time) pedestal values for new calibration. It must provide a user by copying the corresponding file into the corresponding directory (see program read_show_peds).

Then, the calibration constant values are fitted by minimization of functional (1) (see sec.1.2).

If minimization is completed successfully, then the fitted values of calibration constants are set in the structure **/shower/** and are saved into the disk file by calling program **read_show_io**. Otherwise, the old calibration constants values is restored in the structure **/shower/** by calling program **read_show_coef**.

New calibration constants values (including pedestals and its sigma-s values) are saved into both **shower_calib_r**XXXX**.dat** (where XXXX is the run number of data file) and **shower_calib.dat** files. The purpose of creation of two similar calibration constants files is: 1) keep the own constants file for every calibrated run file in order to use it during a future processing without repeated calibration; 2) keep a constants file for cases, when processed run file has not been calibrated previously, and one can use the latest calculated constants (see program **read_show_coef**).

Finally, the temporary file **shcal_events.dat_tmp** is removed and flag of the calibration requirement **shcal_flagfit** is reseted.

Input:

run_number — Number of currently processing run (see **run_info.h**).

shwr(ish).cor(i).amplitude.adc — Initial values of the shower detector channels calibration constants, which was set by program **read_show_coef** before.

shwr(ish).cor(i).offset.adc — Values of the shower detector channels pedestals, by which the new calibration constants are calculated.

Output:

shower_calib.dat and **shower_calib_r**XXXX**.dat** (where XXXX is the run number of data file) — Two similar files, containing the calculated calibration constants.

Used Subroutines: **read_show_peds**, **shcal_coef_io**, **read_show_coef**.

Is Called From: **scan_file** program, when a File/scan command with option “s” has been given.

4 Calibration on Cosmic Ray Data

After the assembly and preliminary test all the modules of the Preshower Detector were tested at TestLab with cosmic ray muons. Wherever it was possible the test were made to simulate closely the final experimental operating conditions, namely

— positions of the modules,

— the electronics and HV supplies,

— all the cables except the delay lines,

— data acquisition.

The test were performed 10 days with the objectives of:

- checking the operation of the modules;
- pulse height equalization from cosmic rays;
- checking the stability during the operation;
- determination of the pulse height resolution for cosmic rays and estimation of the expected energy resolution for electrons.

A system for the shower detector modules calibration on cosmic rays has been developed. This system is useful for preliminary equalization by the high voltage and tracing of amplitudes values from all of shower detector blocks using signals from cosmic rays.

All programs and data files, which are used during shower detector channels calibration on cosmic rays are in **/work/halla/com97/armen/e91026/cosmic** directory.

Cosmic data files **e91026_1445.dat**, **e91026_1446.dat** and **e91026_1447.dat** has been processed by the following.

First, by the ESPACE kumac program **cosmic_cal.kumac** distributions of $A_i - P_i$ are created. Here, $i = 1 \div 144$ is a shower detector channel number, P_i — pedestal value of the i-th channel and A_i — amplitude of the cosmic ray signal on the i-th channel. Cosmic ray events are selected by condition of the unique fired block on the Preshower and the unique fired block on the Shower. Distributions of $A_i - P_i$ are created through the ESPACE variable **adc_c[[i];[i]]** (see 2.1). For this it necessary to contain in the current directory a file **shower_calib.dat**, where calibration constants are equal to 1 for all of channels (see 2.2, 3.2).

Then, by the PAW kumac program **cosmic_cal_paw.kumac** the following is done.

- The mean values $< A_i - P_i >$ for each of detector channel using the fit by Gauss are calculated;
- Calibration constants are calculated by the formula $C_i = N_i / < A_i - P_i >$, where $N_i = 200$ for the Preshower channels, and $N_i = 300$ for the Shower channels;

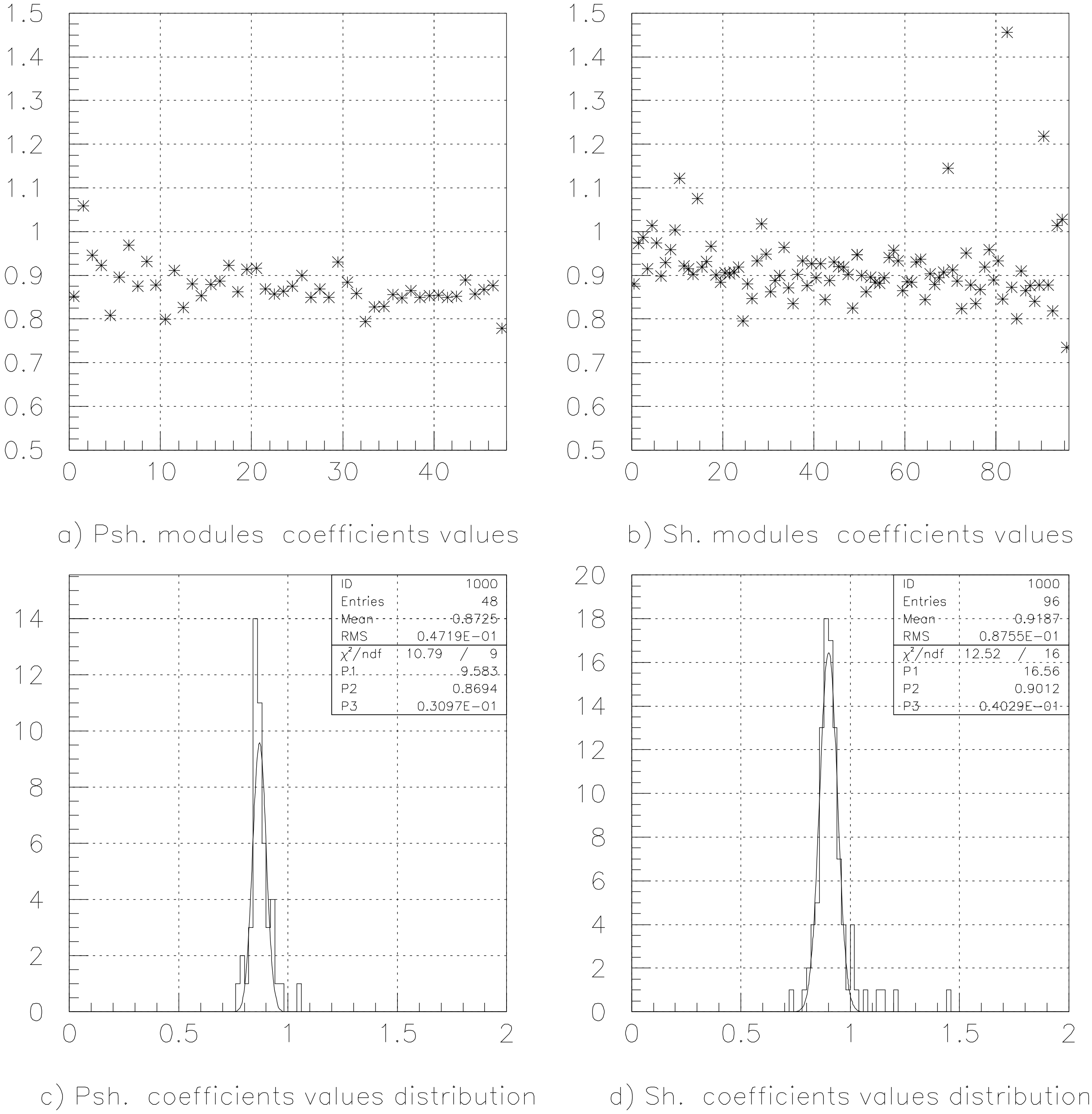

Figure 2: Values of calibration constants computed on cosmic rays data for Preshower (a), and Shower (b) channels *versus* channel number. These coefficients values distributions for Preshower (c), and Shower (d) channels.

- All of distributions histograms, results of fittings, $< A_i - P_i >$ and C_i constants values changing in channels are plotted on the display.

After the execution of **cosmic_cal_paw.kumac** program, the following resulting files are created:

cosmic_cal.ps — a postscript file, containing all of graphics, which are plotted on the display during execution;

cosmic_calib.dat — an ASCII file, containing values of calibration constants C_i, computed on cosmic data, and parameters of Gauss, by which these constants distributions are fitted.

In Fig.2 four pictures among that, which are plotted during execution of program **cosmic_cal_paw.kumac**, are shown. Fig.2(a) and Fig.2(b) show graphics of values of calibration constants computed on cosmic rays data for Preshower and Shower channels, accordingly. These coefficients values distributions in Fig.2(c) for Preshower and Fig.2(d) for Shower allow to estimate an accuracy of shower detector channels amplitudes equalization. These pictures show that channels amplitudes was equalized and stays with the good accuracy for the most of shower detector modules.

5 Shower Detector Channels Stability Control

Shower detector channels stability is controlled through the magnitude of its calibration coefficients values changing in the time during experiment.

Software for the shower detector channels stability control was developed and used for checking of channels stability during two experiments — **e89003** (May–June 1997) and **e91026** (September–December 1997).

All programs and data files, which are used during stability control performance are in the directories:

/work/halla/com97/armen/e89003/sh_calib — for experiment **e89003**, and

/work/halla/com97/armen/e91026/sh_stab — for experiment **e91026**.

The sequence of steps for stability control performing looks as follows:

1. Select a number of run data files distributed uniformly on the creation time in whole of experiment period.

2. Calculate the calibration coefficients of shower detector channels for all of selected data files. Calibration on all of statistic into each of selected files is desirable.

3. View the graphics of calibration coefficients values changing during the time. For any channel, such a graphic of coefficient value versus time (or versus run number) must be around of a horizontal straight line. Any large deflection from this line must be considered separately, in order to determine the reason of the deflection.

Step 2 is made by using the shower detector channels calibration procedure in ESPACE, as it described in sec.2.2. Example of such an ESPACE kumac program text

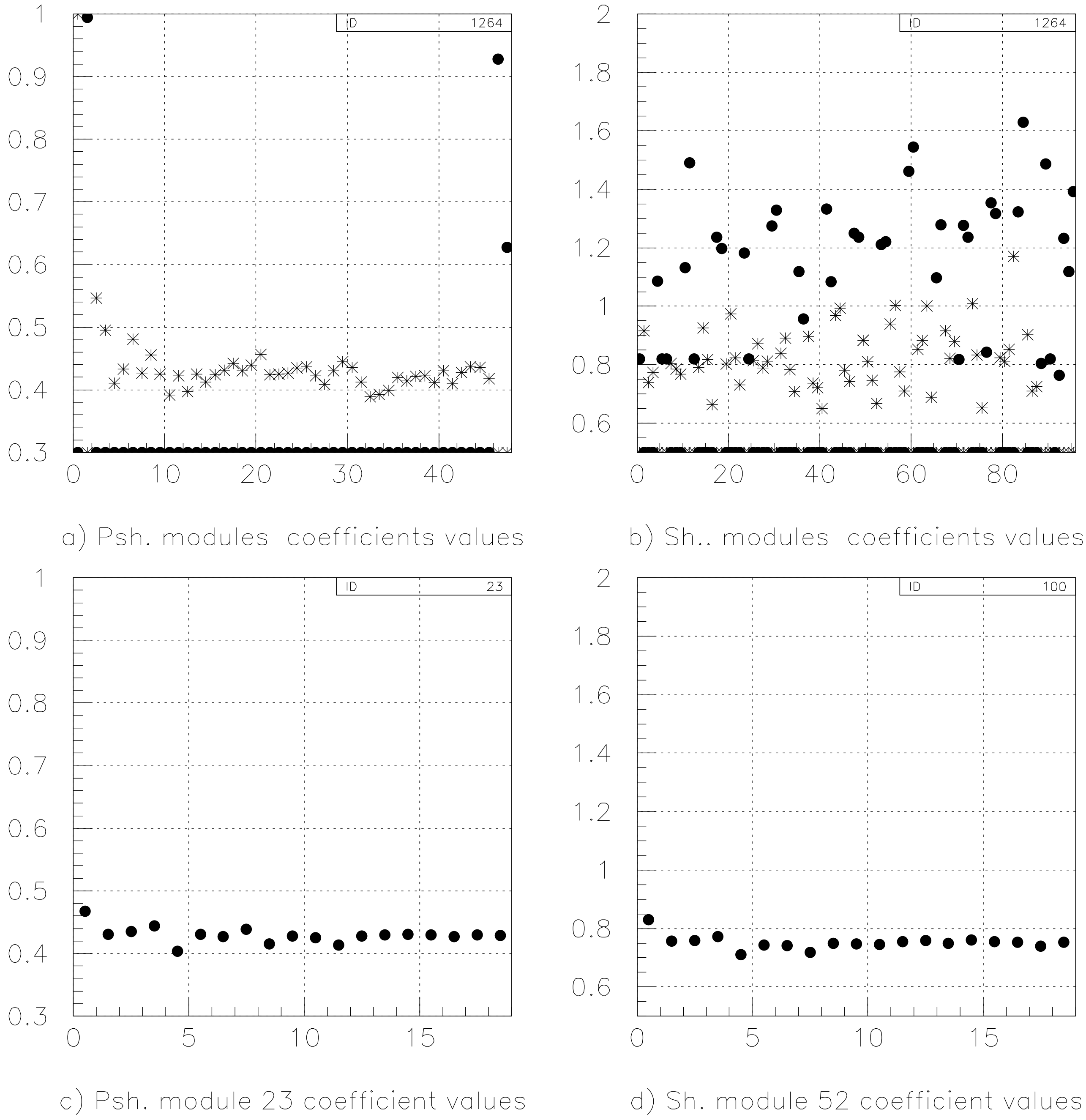

Figure 3: Values of calibration constants computed by data of run 1264 for Preshower (a), and Shower (b) channels *versus* channel number. ('$*$'-s are corresponding to the calibrated modules coefficients, and '$\bullet$'-s, to the uncalibrated). Graphics of constants values changing during the time for the certain channels: module 23 of Preshower (c), and module 52 of Shower (d) (constant value *versus* the sequential number of run in the given interval of run numbers).

is in **shcal.kumac** file. In result of this program execution the computed calibration constants files **shower_calib_r**XXXX**.dat** (where XXXX is the run number of data file) for all of scanned data files are created. Besides, **shcal_stat.**XXXX files containing an information which shows which channels was calibrated and which channels was not calibrated by data of run XXXX, for each of calibration data files are created. (About calibrated and uncalibrated modules see ESPACE program **shcal_dofit** in sec.3.2).

For the step 3 realization a PAW kumac program **sh_stab.kumac** has been used, which:

- Gets the least and the greatest numbers of runs, between which numbers of calibration data runs are found. These run numbers interval bounds must be given by user either as a parameters of PAW macro, or from terminal as an answer of program question.

- Shows graphics of calibration constants values of all channels of shower detector. Two graphics (one for Preshower and one for Shower) are plotted for every run, by data of which calibration constants have been calculated. On these graphics, the points, corresponding to the calibrated and uncalibrated modules coefficients are signed by the different markers (an '$*$' for calibrated, and a '$\bullet$' for uncalibrated). For each of run number XXXX in the given interval, calibration constants are read from the **shower_calib_r**XXXX**.dat** files. Calibrated and uncalibrated modules are identified by statistic information in the **shcal_stat.**XXXX files. Calibration constants file and statistic information file are read by calling the fortran routine **consrd.for**.

- Shows graphics of calibration constants values changing during the time. For this, for every shower detector channel, a graphic of constant value versus the sequential number of run (by data of which this constant has been calculated) is plotted.

After the execution of **sh_stab.kumac** program, a postscript file **sh_stab.ps** containing all of graphics, which are plotted on the display during execution, is created.

Fig.3 contains a four graphics among that, which are plotted during execution of program **sh_stab.kumac**.

Fig.3(a) and Fig.3(b) show graphics of values of calibration constants computed on beam data of run 1264 for Preshower and Shower channels, respectively. Du to the small statistic (or no statistic) on the boundary modules, these modules have not been calibrated, and its calibration constants values stay the same as its initial values. In these graphics '$*$'-s are corresponding to the calibrated modules coefficients, and '$\bullet$'-s, to the uncalibrated.

Fig.3(c) and Fig.3(d) show graphics of constants values changing during the time for two of more statistical modules in central part of Preshower (module 23) and Shower (module 52), respectively. Along the X-axis sequential numbers of runs in the given interval of run numbers are shown. Along the Y-axis — values of calibration constants, computed by data of each of run.

These graphics was obtained after the processing the 19 selected run files for the shower detector channels stability consideration during the **e91026** experiment. The same processing has been done over the data of 30 run files, selected among the **e91026** experiment data files. Quality of shower detector channels stability during both of experiments is good.

Note 1: For **shcal_stat.**XXXX file creation (where XXXX is the run number of data file) some additions in my own version of ESPACE program **shcal_dofit** (see sec.3.2) have been made.

Note 2: Versions of programs for experiment **e91026** data processing are described before. Versions of programs for experiment **e89003** data processing are a little different.

6 e/π Identification

6.1 How to use the "Gas Cherenkov" and "Total Shower" detectors for e/π rejection.

The following instruction has been elaborated recently and represents a general description of how to use "Gas Cherenkov" (GC) and "Total Shower" (TS) detectors of the Hall A electron spectrometer for identification electrons and pions. The work is in progress, so we hope to develop user oriented more convenient program by taking into account possible suggestions and corrections about current code. Also, we plane to add an ESPACE variable, which value would determine particle type.

Present version consists of several stages to reach consequentially definite goals:

1. Energy calibration of "Total Shower" detector.

2. The determination of parameters ($pshe$, μ_e, σ_e, μ_π, σ_π) of shower detector by means of electrons and pions rejected by "Gas Cherenkov" detector.

3. The determination of the identifications characteristics of the "Gas Cherenkov" detector for electrons and pions identified by "Total Shower" detector on the basis of parameters, which have been found on step #2.

4. The determination of the identification characteristics of the "Total Shower" detector by using identification characteristics of the "Gas Cherenkov" detector, obtained on step #3.

In "Total Shower" detector particles are identified by their energy deposition in the detector, so one needs to have precise values of the calibration coefficients, i.e. it is better to perform calibration for given data file. Then sometime the calibration has to be done as mentioned. Electrons are defined by summing up amplitudes provided by all PMT-s of "Gas Cherenkov" detector. So one needs to have calibration coefficients for each PMT. At present time due to the absence of calibration code for "Gas Cherenkov" detector the calibration coefficients from database are used. In case one can get them by PMT amplitude spectrum analysis.

Our analysis of the "PreShower" and "Shower" detectors responses have shown that it is convenient to use the function dependent on momentum of particles (P), energy deposition in "PreShower" detector (E_{presh}), integral energy deposition in "Total Shower" detector (E_{tot}) given as

$$fen = \frac{E_{tot}}{P} \times \frac{\lg(E_{presh})}{\lg(pshe)}$$

where $pshe$ is the average energy deposition of electrons with momentum $P_e = P_{centr}$.

As soon as the dependence $pshe(P)$ analytically is not known and due to the incompleteness of data, we recommend to use step #2 procedures once the value of P_{centr} gets change.

On step #2 the mean values and the corresponding standard deviations of energy deposition for electrons and pions with $P = P_{centr}$ must be found as well (μ_e, σ_e, μ_π, σ_π).

To separate electrons and pions the response of "Gas Cherenkov" detector was used. The parameters μ_e, σ_e, μ_π, σ_π are needed in step #3 to extract "pure electron" ($|\mu_e - E_{tot}| < \sigma_e/2$) and "pure pions" ($|\mu_\pi - E_{tot}| < \sigma_\pi/2$) events by implementing certain conditions on E_{tot}. Besides of that on step #3 the mean value of the fen function ($meanfen$) and standard deviation of fen ($sigmafen$) are defined for electrons.

And finally, on step #4, first of all the conditions to chose electrons and pions are defined as

$meanfen - \alpha \times sigmafen < fen < 2$ — for electrons

$0 < fen < meanfen - \alpha \times sigmafen$ — for pions

where α is task dependent, and second of all the identification characteristics of "Total Shower" detector are investigating.

Calibration process is initiated, when a File/scan command with option "s" is executed.

....

File/scan/ [dir]/[run].dat FIRST=1 LAST=50000 -s

....

The first and last entries given in this command show the range of entries, among which events for calibration will be selected.

Using KUMAC file, which containing following commands:

....

define/cut/1d spec_e.gas.adcsum_c-1 0.0 10000.0

....

define/logical cer_elec spec_e.gas.adcsum_c-1

```
define/logical cer_pion!spec_e.gas.adcsum_c-1
..................
Define/variable eshow:(spec_e.show.cl_e+spec_e.preshow.cl_e)
..................
set/spectrum/bins/x1 100
set/spectrum/windowx1 0. 2500.
spectra/save eshow-1 cer_elec
spectra/save eshow-2 cer_pion
spectra/save spec_e.preshow.cl_e-1 cer_elec
..................
```

and analysis spectrum, one can obtain parameters $pshe$, μ_e, σ_e, μ_π, σ_π.

Using KUMAC file, which containing following commands:

```
...................
Define/variable eshow:(spec_e.show.cl_e+spec_e.preshow.cl_e)
Define/variable epi:(spec_e.show.cl_e+spec_e.preshow.cl_e)/spec_e.p_kin
Define/variable ddd:(log10(spec_e.preshow.cl_e))/log10(pshe)
...................
define/cut/1d spec_e.u1.clus-1 0.5 1.5
define/cut/1d spec_e.v1.clus-1 0.5 1.5
define/cut/1d spec_e.u2.clus-1 0.5 1.5
define/cut/1d spec_e.v2.clus-1 0.5 1.5
define/cut/1d spec_e.beta-1 0.7 1.3
define/cut/1d eshow-1 [μe − σe/2] [μe + σe/2]
define/cut/1d eshow-2 [μπ − σπ/2] [μπ + σπ/2]
define/cut/1d spec_e.preshow.part_id-1 -0.5 0.5
define/cut/1d spec_e.show.part_id-1 -0.5 0.5
...................
define/logical goodel eshow-1
define/logical goodpi eshow-2
define/logical partid1 spec_e.preshow.part_id-1&&spec_e.show.part_id-1
define/logical no_cosmic spec_e.beta-1
define/logical one_cluster_e spec_e.u1.clus-1&&spec_e.v1.clus-1&&_
      spec_e.u2.clus-1&&spec_e.v2.clus-1
...................
set/spectrum/bins/x1 10
set/spectrum/window/x1 0. 10000.
spectra/save spec_e.gas.adcsum_c-1
spectra/save spec_e.gas.adcsum_c-2 _
      one_cluster_e&&partid1&&no_cosmic&&goodel
spectra/save spec_e.gas.adcsum_c-3 _
```

```
    one_cluster_e&&partid1&&no_cosmic&&goodpi
set/spectrum/bins/x1 200
set/spectrum/window/x1 0. 2.
spectra/save fen-4 one_cluster_e&&partid1&&no_cosmic&&cer_elec
```

(where instead of $[\mu_e - \sigma_e/2]$, $[\mu_e + \sigma_e/2]$, $[\mu_\pi - \sigma_\pi/2]$ and $[\mu_\pi + \sigma_\pi/2]$ must be set real numbers, computed through the parameters μ_e, σ_e, μ_π, σ_π, obtained before), one can obtain efficiency identification electrons η_e and pions η_π by formulas

$$\eta_e = \frac{number\ of\ events\ in\ spectrum\ 2}{number\ of\ events\ in\ spectrum\ 1}$$

$$1 - \eta_\pi = \frac{number\ of\ events\ in\ spectrum\ 3}{number\ of\ events\ in\ spectrum\ 1}$$

and calculate $meanfen$ and $sigmafen$ by analysis spectrum 4.

Using KUMAC file, which containing following commands:

```
..................
define/cut/1d fen-1 (meanfen-sigmafen/2) 2.
define/cut/1d fen-2 0. (meanfen-sigmafen/2)
..................
define/logical sh_elec fen-1
define/logical sh_pion fen-2
..................
spectra/save eshow-1 one_cluster_e&&partid1&&no_cosmic
spectra/save eshow-2 one_cluster_e&&partid1&&no_cosmic&&cer_elec
spectra/save eshow-3 one_cluster_e&&partid1&&no_cosmic&&sh_elec
spectra/save eshow-4 _
    one_cluster_e&&partid1&&no_cosmic&&cer_elec&&sh_elec
```

one can obtain efficiency identification electrons λ_e and pions λ_π by formulas

$$\lambda_e = \frac{n_{cerel}^{shel} + n_{shel} \times (\eta_\pi - 1)}{n_{cerel} + n \times (\eta_\pi - 1)}$$

$$\lambda_\pi = \frac{n_{cerel} - n_{cerel}^{shel} - \eta_e \times (n - n_{shel})}{n_{cerel} - \eta_e \times n}$$

where: n — number of events in spectrum 1;
n_{cerel} — number of events in spectrum 2;
n_{shel} — number of events in spectrum 3;
n_{cerel}^{shel} — number of events in spectrum 4.