On the kinematic reconstruction of multiparticle events

A.D.Bukin
Budker Institute of Nuclear Physics, Novosibirsk 630090, Russia

Kinematic reconstruction of multiparticle events with special parametrization of likelihood function is considered. For those cases when approximate form of angular part of likelihood function is acceptable, the analytical solution for optimal general rotation was found, that allows to decrease the total number of free parameters by three rotation angles and strongly simplifies the likelihood function profile.

Key words: data processing; event reconstruction; kinematic fit

1 Introduction

Data processing in high energy physics often contains the procedure of improving the accuracy of experimental measurements of particle characteristics, using some constraints, typically conservation laws and known values of unstable particle masses. Most of all it is done by numerical minimization of the log-likelihood function depending on some number of parameters. The choice of parameters for the events in center-of-mass system can be done so that some subset of parameters defines the rigid construction of outgoing particles while the last three parameters define the rotation of this construction as a whole. Then the task of analytical minimization of the aim function over these three rotation parameters becomes very attractive because of: 1) decrease of CPU time necessary to minimize the function and 2) simplifying the function profile, that decreases the probability of finding some false minimum.

2 Sample of parametrization of the events $e^+e^- \to \eta\gamma \to \pi^+\pi^-\pi^0\gamma$

For conveniency let us consider the particular process $e^+e^- \to \phi \to \eta\gamma \to \pi^+\pi^-\pi^0\gamma \to \pi^+\pi^-\gamma\gamma\gamma$. The final state contains five particles and is described by 15 parameters: 5 momenta and 10 angles (θ and φ angles in the polar system). But these parameters are not independent. There are 5 constraints: 4 energy-momentum conservation laws and a known value of the π^0 mass. So there are only 10 free parameters. In general, if a detector measures the angles of all 5 particles, one can reconstruct momenta of all particles. Of course the accuracy of angular measuments will determine the accuracy of the reconstructed η-meson mass. If there are some additional measured parameters, then the accuracy of η meson mass will increase and the minimum value of the log-likelihood function can be used for rejection of background events.

The possible set of 10 free parameters is the following. The first parameter is an invariant mass M_η of the η-meson. For an e^+e^- collider with equal energies of the initial electron and positron E_0, the total momentum equals zero and the total energy equals $2E_0$. Using this information one can easily derive the energy of the first photon and momentum of the η-meson. Let η-meson move along the Z-axis in some system of reference and π^0 move in the XZ-plane in this system. In the system of η-meson we can define two more free parameters: momentum p_0 of π_0 and angle θ_0 between this momentum and the Z-axis. Now in the system of π^0-meson we can define the angle θ_γ between one of the photons and π^0 momentum and axial angle φ_γ, which determines the rotation of $\pi^0 \to \gamma\gamma$ decay plane with respect to the XZ-plane. The corresponding angles θ_+ and φ_+ define the π^+ momentum vector in the center-of-mass system of $\pi^+\pi^-$. At last the rotation angles ψ_1, ψ_2, ψ_3 allow an arbitrary rotation of this 5-particles construction as a whole.

Preprint submitted to Elsevier Preprint *28 February 1997*

Let us represent the log-likelihood function L as a sum $L = L_E + L_A$, where L_E takes into account the deviations of the particles energies from the measured values (for those particles, whose energies are measured) and L_A takes into account the deviations of particle directions from the measured ones:

$$L_A = \sum_i \frac{\Delta\alpha_i^2}{2\sigma_i^2} \tag{1}$$

Here $\Delta\alpha_i$ is the angle between the direction of the i-th particle momentum in our model and experimentally measured one, σ_i is an estimation of experimental angular accuracy.

For the further illustration let us use the simulated events for the detector SND [1] (Spherical Neutral Detector) at the energy $E_0 = 510$ MeV. This detector has no magnetic field, so the energies of the charged pions are not measured. Hence the energy part of the likelihood function L_E takes into account only the deviations of the measured energies of the three photons from the "theoretical" energies.

Obviously the energies of particles in our model do not depend on the angles ψ_1, ψ_2, ψ_3, hence the energy part of likelihood function L_E does not depend on these angles. There is no profit to minimize separately L_A by numerical methods over $\psi_{1,2,3}$ for every set of 7 parameters M_η, p_0, θ_0, θ_γ, φ_γ, θ_+, φ_+. On opposite there will be great increase of CPU time consumption. In this paper the analytic solution for minimization of the approximate form of L_A over $\psi_{1,2,3}$ is presented.

3 Approximate form of likelihood function

For small deviations $\Delta\alpha_i$ in (1) likelihood function can be approximated by

$$L_A \approx \sum_i \frac{1 - \cos\Delta\alpha_i}{\sigma_i^2} = \sum_i \frac{1 - \mathrm{e}_i \cdot \mathrm{n}_i}{\sigma_i^2} = \sum_i \frac{1}{\sigma_i^2} - \sum_i \frac{\mathrm{e}_i \cdot \mathrm{n}_i}{\sigma_i^2} \tag{2}$$

Here vector e_i is the unit vector, defining the measured direction of i-th particle, vector n_i is the theoretical unit vector for i-th particle, defined by 10 variable parameters in our model.

First sum in (2) is constant and can be omitted. Likelihood function L_A reaches its minimum value, when the second sum in (2)

$$L_M = \sum_i \frac{\mathrm{e}_i \cdot \mathrm{n}_i}{\sigma_i^2} \tag{3}$$

reaches its maximum value. Vectors n_i can be presented as unit vectors s_i, depending in our model only on the first 7 parameters, transformed with a rotation matrix T, depending only on the angles $\psi_{1,2,3}$: $\mathrm{n}_i = \mathrm{T} \cdot \mathrm{s}_i$. Now we can repeat the task in another way: it is necessary to find the rotation matrix T such that

$$L_M = \sum_i \frac{\mathrm{e}_i^T \mathrm{T} \mathrm{s}_i}{\sigma_i^2} = Tr\left(\mathrm{T} \sum_i \frac{\mathrm{s}_i \mathrm{e}_i^T}{\sigma_i^2}\right) = Tr(\mathrm{T}\ \mathrm{V}) \tag{4}$$

reaches its maximum value, where superscript "b^T" means transposed matrix b.

The analytic solution of this problem was found by the well-known method of Lagrange coefficients (for example, in [2]). All auxilliary transformations were made with REDUCE code [3]. Detailed de-

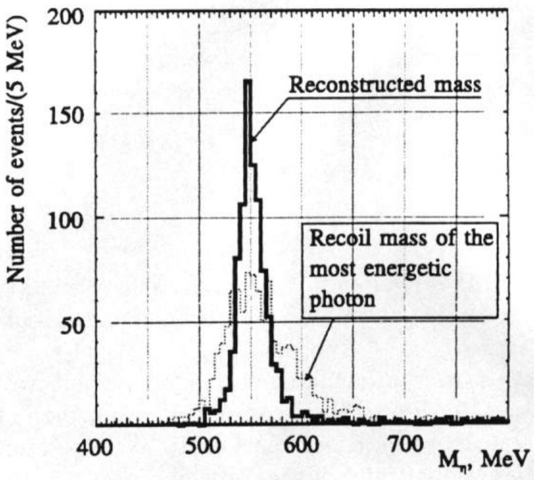

Fig. 1. Distribution of simulated events over parameter M_η ($e^+e^- \to \eta\gamma \to \pi^+\pi^-\pi^0\gamma$). Dashed line shows the distribution of these events over the recoil mass of the most energetic photon

scription of the solution and corresponding subroutine BUROTAT, implemented in Fortran 77, is sent to "Computer Physics Communications".

Fig. 1 demonstrates the effect of kinematic reconstruction of the non-magnetic detector SND [1] events of the considered process.

Measurements of the directions of all 5 final particles and energies of the 3 final photons are fitted. Events are simulated by UNIMOD2 code [4] and reconstructed by SND software package.

4 Conclusion

One of the possible procedures for the kinematic reconstruction of multiparticle events is considered. For special parametrization of an event and suggested approximate angular part of the likelihood function the optimal rotation procedure is derived and implemented in Fortran-77.

This procedure allows to decrease the number of free optimized parameters by three, and hence decrease the CPU consumption by more than two times for the sample process $e^+e^- \to \eta\gamma \to \pi^+\pi^-\gamma\gamma\gamma$ (decrease from 10 to 7 optimized parameters). Even more important there is an increase of detection efficiency for this process by several percents, that characterises the simplification of the likelihood function profile and hence the less probability of finding the false minimum (minimization was performed by MINUIT code).

References

[1] *V.M.Aulchenko et al.* SND — detector for VEPP-2M and ϕ–factory, in: Proc. Workshop on Physics and Detectors for DAΦNE (Frascati, 1991) 1991, pp. 605 – 613.

[2] *G.A.Korn and T.M.Korn.* Mathematical handbook for scientists and engineers. McGraw-Hill, 1968.

[3] *Anthony C. Hearn.* REDUCE user's manual. Rand, 1987.

[4] *A.D.Bukin et al.,* Proc.: Workshop on Detector and Event simulation in High Energy Physics, Amsterdam, April 1991, NIKHEF, 1991, p.79.
Budker INP, 94-20, Novosibirsk, 1994.

OPAL, an Open Physics Analysis Language.

A.G.Shamov,

Budker Institute of Nuclear Physics
630090 Novosibirsk, Russian Federation

OPAL is an open language which includes most important FORTRAN-77 operators and can be easily expanded with arbitrary user-defined statements. The subset of OPAL called XBOOK provides a convenient interface to HBOOK/HPLOT packages of the CERN program library. The OPAL toolkit is rather compact and can be built in any application. Currently OPAL is available under VAX/VMS only.

Key words: Analysis; programming languages; software tools.

1 Introduction

The last years of HEP show that the PAW program [1] is an efficient tool of data analysis. The drawback of it is the necessity to use very different languages: that of KUIP [2] and FORTRAN in the COMIS frame [3].

The paper present the *open* language which includes simple and convenient commands for histogram manipulating together with the most important FORTRAN-77 operators and can be expanded with arbitrary user-defined statements.

The language was called OPAL in honor of PAW and was employed for data analysis and presentation in some experiments [4], in the test beam and prototype experiments software [5], as an input language of the simulation tool [6] and as a general command language of the XonLINE data acquisition system [7].

2 OPAL kit

The OPAL kit is the object (or shareable) library which can be built in any application program. The library includes the initialization procedure, the command line interpreter (CLI), the set of procedures implementing the standard OPAL commands and the interfacing routines allowing an application to establish its specific environment.

An OPAL application can be considered as consisting of a static part designed at coding time and a dynamic one created during the interactive session with OPAL interpreter.

Using the OPAL interfacing routines any static objects (variables, arrays, common blocks, functions and subroutines) can be declared for interactive use at the OPAL level. Dynamically created objects can be accessed from the static parts of an application by their names.

This allows to develop an efficient and flexible tools for off-line and on-line data analysis and simulation.

3 OPAL compiler/interpreter

The OPAL syntax is similar to that of Digital Command Language DCL but allows multilevel command parameters:

Preprint submitted to Elsevier Preprint *28 February 1997*

$command$:=[$label$] [$c\_keyword$] [$qualifier$...] [$parameter$ ...]

$qualifier$:=\$q_keyword$ [= $parameter$]

$parameter$:=$parameter$ [$qualifier$] | $parameter_1$

$parameter_1$:=$parameter_1$ d_1 $parameter_2$ | $parameter_2$

$parameter_2$:=$parameter_2$ d_2 $parameter_3$ | $parameter_3$

$parameter_N$:=[$p\_keyword$ p_k_d] $expression$ | $source\_string$

where *label* - statement label starting with a digit, *keyword* - user specified keyword starting with a letter, d_1, d_2, ... - delimiter set specific for given command, *p_k_d* - parameter keyword delimiter, *expression* - FORTRAN-77 expression (integer, real etc.), *source_string* - character string which is not to be interpreted by the CLI.

Such syntax, with the additional rule that a delimiting symbol enclosed in brackets is not considered as a delimiter, allows to describe all FORTRAN-77 operators with the reasonable limitations and gives a simple way to implement application-specific commands.

To allow more or less free usage of blanks for source line formatting, the following rule is applied:

delimiter := [*blank*] *non_blank_delimiter* [*blank*]

Processing a source line, the CLI extracts the command label, if specified, and the command keyword, looks up the command table and parses command qualifiers and parameters using the corresponding delimiter set. The command keyword can be omitted for assignments and procedure calls (LET and CALL commands).

To define an additional command, an user must call the library routine to specify its parameter structure and other attributes and supply the procedure implementing the command.

The CLI generates an executable machine code for each command processed. Parsing the command line, it generates a code to obtain a value of each parameter specified. The command-dependent part of code generation can be accomplished in the two different ways.

If the command is declared as a 'system mode' command, the code must be generated by the command implementing procedure which is called once after parameter parsing.

If the command is declared as a 'user mode' command, the CLI generates the code to call a command implementing procedure which must analyze parameters specified and perform actions required.

System mode commands are generally more efficient than user mode ones. To generate an executable code, a command implementing procedure can use the special library calls or recursively call the CLI. The user mode command procedures are similar to those of KUIP.

Due to machine code generation, OPAL is approximately 10 time more efficient than COMIS.

4 Example of a command implementation

Let us consider how to implement in the OPAL frame the *select* command of the Kinematical Analysis Language (KAL) developed for the data analysis of the ARGUS experiment [8]:

select particle_type[, ...]

This command starts all loops required to select all combinations of particles of specified types in an event record. To declare the commands one call of the OPAL library is required:

```
call xdcom( SELECT,0,'XP1','SELECT','R9TSD2',CC)
```

This call specifies the external command implementing procedure (SELECT), the command mode and number of parameter description strings ('XP1'), the command keyword ('SELECT') and the parameter description string 'R9TSD2' which defines the parameter repetition count 'R9', the parameter type 'TS' (source string) and the delimiter set 'D2' (commas).

Using the OPAL library, the command implementing procedure must retrieve the particle types and generate the executable code required. For two particles it might look like:

```
call x_cli('DO LV1=1,PDIRL(NPC1)')
call x_cli('IP1=PDIR(LV1,NPC1)')
if(npc1.eq.ncp2) call x_cli('DO LV2=LV1+1,PDIRL(NPC1)')
if(npc1.ne.ncp2) call x_cli('DO LV2=1,PDIRL(NPC2)')
```

call x_cli('IP2=PDIR(LV2,NCP2)')
where NPC1,NCP2 are numerical particle codes, PDIR, PDIRL are functions for accessing particle directory of an event and IP1,IP2 are the particle pointers which must be used by other KAL operators.

5 OPAL interface to HBOOK/HPLOT

OPAL includes the set of commands and the interfacing procedures for the histogram manipulating called XBOOK ('X' can be considered as the cyrillic letter matching the latin 'H').

In the XBOOK frame histograms are identified by names and can be arranged in multidimensional arrays. It provides most part of the histogram service available in PAW and, additionally, allows to control interactively histogram filling process both in the static and dynamic parts of an application.

The last feature is very important for an on-line data analysis. Histogram filling with XBOOK is faster than with normal HBOOK calls.

6 Conclusion

A few years of exploitation of OPAL in BINP in various applications have shown its efficiency and convenience. VAX/VMS version of OPAL is available for HEP community. Porting it to other platforms requires an external support.

References

[1] Brun R. et al., CERN program library Q121.

[2] Brun R., Zanarini P.. CERN program library I202.

[3] Berezhnoi V. et al., CERN program library L210.

[4] S.E.Baru et al., Phys. Rep. 267 (1996) 73-160.

[5] V.M. Aulchenko et al., NIM A379 (1997) 475; ibid p.360; ibid p. 491.

[6] A.G. Shamov, A.R. Buzykaev. LCE, Light Collection Efficiency simulation tool. This proceeding.

[7] I.B.Logashenko, A.G.Shamov. Int. Conf. on Computing in High Energy Physics'95, Rio de Janeiro, Brasil, 1995.

[8] H. Albrecht et al., http://www.Physik.Uni-Dortmund.DE/ARGUS/KAL

LCE, Light Collection Efficiency simulation tool.

A.G.Shamov, A.R.Buzykaev
Budker Institute of Nuclear Physics
630090 Novosibirsk, Russian Federation

LCE code simulates all physical processes essential for light collection in Cherenkov and scintillation detectors including light scattering and re-emission of light by wave length shifters. LCE employs the Open Physics Analysis Language (OPAL) for the detector description and the simulation process control. It allows one to specify an arbitrary dependence of the optical parameters on the wave length and simulate physical processes or their modes which were not foreseen in the program initially. Currently LCE is available under VAX/VMS only.

Key words: Simulation; aerogel; WLS; Cherenkov counter.

1 Introduction

Particle identification systems based on the Aerogel Threshold Counter (ATC) were supposed to be used in all detectors designed to study B-physics in the Υ energy region. This paper describes the Monte Carlo code called LCE and employed for simulation of the ATC systems of KEDR [2,3] and Ba$\bar{\text{B}}$ardetectors [4,6,6].

2 Physical processes

LCE code simulates following physical processes:

Cherenkov light radiation,
light refraction on media boundaries,
Rayleigh scattering,
light absorbtion in radiators and light guides,
light reflection from detector walls of various types,
light absorbtion on detector walls,
light absorbtion and re-emission by Wave Length Shifters (WLS)
and photon detection.

All optical parameters involved can arbitrary depend on the light wave length. Wall absorbtion coefficients can arbitrary vary in space. A few different models are foreseen for the photon detection simulation. The main part of the physical algorithm is used since the work [1].

3 Geometry representation

Unlike to many other simulation tools, the main geometrical entity of LCE is *a wall* but not *a block*.

Walls are pieces of plains or second order surfaces. To describe a wall one must state the inequality for its surface and the set of additional inequalities (*bounds*) selecting the required piece of the surface.

A space regions limited by walls and having constant optical parameters is called *a block*. A block may have an arbitrary shape, it is not necessarily closed nor convex. A photons is located inside the block, if it crosses one of the block's walls on its way. A photon travels from block to block though windows on their walls which must be specified explicitly.

Such approach seems adequate to the nature of the problem is rather efficient and can be implemented easily.

4 Input language

The LCE is an example of the OPAL-based application. OPAL [7] is an Open Physical Analysis Language which includes the most important FORTRAN-77 operators and can be expanded with arbitrary application-specific statement. An application can define the set of objects (variables, arrays and callable procedures) which are available at OPAL level.

The LCE declares as OPAL variables the most urgent data including the photon wavelength $\%WL$, its current coordinate $\%P$ and direction of motion $\%D$, its statistical weight $\%PW$ etc..

The commands most frequently used in detector description are shown below in various possible formats:

```
  BLOCK[\NAME = bname] -
 [REFRACTORY_INDEX = ri[(%WL)] -
 [ABSORBTION_LENGTH = al[(%WL)]] -
 [SCATTERING_LENGTH = sl[(%WL)][\LAW = slaw(%D)]] -
 [WLS_NAMES = wlsname[,wlsname..]]
   NESTED BLOCKS bname[...]
   WALL[\NAME = wname] surface -
   [BOUNDS = surface[,...]] -
   [REFLECTION_TYPE = type[\LAW = rlaw(%WL)]] -
   [ABSORBTION_COEFFICIENT = abs[(%WL)]]  END
   WALL[\NAME = wname] surface -
   [BOUNDS = surface[,...]] -
   [REFLECTION_TYPE = type[\LAW = rlaw(%WL)]] -
   [ABSORBTION_COEFFICIENT = abs[(%WL)]]
     WINDOW [surface] TO = bname.wname
     WINDOW [surface] -
       TO = DETECTOR[\PROCEDURE = dp[(%WL,%PW)]]
   END_WALL
END_BLOCK
```

All the command and parameter keywords can be reasonably abbreviated. The surface inequalities denoted as *surface* can be specified as expressions containing dummy variables x, y and z. The examples are $(x-xc)^{**}2+(y-yc)^{**}2 < radius^{**}2$, $x > 0$ etc. The inequality sign specifies the front side of a surface. The other way to specify surfaces is referencing of walls by their names.

The following types of reflection are foreseen: *Fresnel*, *Lambert*, *mirror*, *isotropic* and *userdefined*. To implement its own reflection type, an user must create an OPAL procedure for the photon reflection and specify it with the $\backslash LAW$ qualifier.

The default scattering law (isotropic) and the detection procedure can be modified in the same way.

In any place were a numeric value is required, one can substitute an arbitrary expression with possible dependence on the wave length and, for some parameters, on the photon position.

The Wave Length Shifters referenced in $BLOCK$ commands can be described with the special command WLS. The use of *nested blocks* makes the detector description simpler and allows to simulate such thing as multiple bubbles of air in a scintillator.

The other LCE commands and the standard OPAL commands facilitate a detector description ($COPY$, $MOVE$, $ROTATE$, $DO-ENDDO$) allows to specify the light generation procedure, control the simulation process ($GENERATE$, $STOP$, $ACCOUNTING$), book, fill and display histograms required ($BOOK$, $FILL$,$PLOT$) etc..

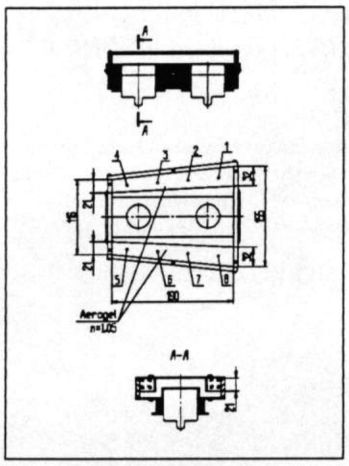

Fig. 1. One of the Ba$\bar{\text{B}}$arATC prototypes.

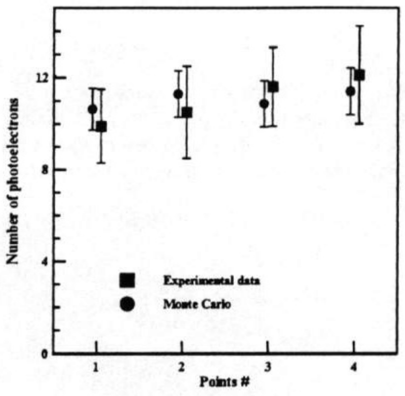

Fig. 2. LCE simulation VS beam test data.

One of the Ba$\bar{\text{B}}$arATC prototypes simulated with the LCE code is shown in fig.1, the LCE predictions are compared with the beam test data in fig. 2 (paper [6]).

5 Conclusion

After several years of intensive use, LCE can be stated as an efficient and convenient simulated tool. It was carefully tested on the configurations with known analytical solutions. Its predictions for ATC prototypes are in agreement with the experimental results.

References

[1] A. Aksenov et al., Int. Symp. on Position Detectors in High Energy Physics, Dubna, Russia, 1988. D1.13-88-172, p.313.

[2] A. Onuchin et al., 5th Int. Conf. on Instr. for Colliding Beams Physics, Novosibirsk, Russia, 1990 (World Scientific 1990), p.208.

[3] A.Buzykaev et al., Nucl. Instr. an Meth. A379 (1996) 453-456.

[4] A.Buzykaev et al., Ba$\bar{\text{B}}$arNote 170, 1996.

[5] K.Arisaka et al., Nucl. Instr. an Meth. A379 (1996) 460-464.

[6] D. Boutigny et al., Ba$\bar{\text{B}}$arNote 290, 1996.

[7] A.G.Shamov. OPAL, an Open Physics Analysis Language. These proceedings.

Event reconstruction algorithm for BGO endcap calorimeter of CMD-2 detector

Kazanin Vasilii
Budker Institute of Nuclear Physics, Russia,
Novosibirsk 630090, Lavrentieva Avenue 11
CMD-2 collaboration

An event reconstruction algorithm for BGO endcap calorimeter of CMD-2 detector is described. Detector operates on VEPP-2M electron-positron collider with c.m. energy range from 0.36 to 1.4 GeV. The calorimeter consists of two endcaps, built of rectangular BGO crystals. Crystal size is $25*25*150$ mm^3 that corresponds to 13.5 radiation length for normal incidence. Total number of crystals in the calorimeter is 680. The presence of 1T magnetic field caused the use of vacuum phototriods for light readout. They can operate in high magnetic field, but their current gain is about 10 only, instead of 10^5 – 10^7, typical to PMT. The low gain and relatively low lightoutput for BGO crystals cause substantial electronic noise level of about 1 MeV per channel in average. The contribution of electronic noise to energy resolution is especially important for CMD-2 calorimeter due to low energy of photons to be detected. Maximum photon energy is 700 MeV, while in the most interesting reactions, such as radiative decays of light vector mesons, photons have typical energies of 100 MeV. Due to these reasons a special algorithm with detailed noise consideration is needed. Algorithm has been realized as a set of subroutines included into general CMD-2 event reconstruction program. The main feature of this algorithm is that selection criteria are tuned individually for each crystal as a function of measured electronic noise. Noise measurement is performed during data taking approximately once per week. Two selection criteria for triggered crystals are used in the algorithm: absolute energy deposition and ratio of signal to electronic noise sigma. Soft selection marks crystal hits. Reconstruction of event in the calorimeter begins with the formation of cluster nuclei. Cluster nucleus is formed of standing side by side crystals, which have passed hard selection. Afterwards cluster is built from cluster nucleus by adding all neighbouring crystal hits. To prevent information loss all crystal hits which don't belong to any cluster are saved separately. Energy, position and other parameters of clusters are calculated after each stage. Reconstruction program uses HEPDB data base to extract calibration data, noise data and parameters of calorimeter data storage. Reconstruction code is used on SGI Super Server Challenge-L. The algorithm has been tested with simulated events and has been applied to elastic electron-positron scattering and cosmic events. Preliminary energy resolution for endcap calorimeter is 6% for 510 MeV electrons produced by elastic scattering. Algorithm provides good resolution and stability.

Нет с.252
см. с.52 pt. Abstracts –
B401

2 Data acquisition system architecture

The readout and trigger electronics of the system (Fig. 1) makes use of the KLUKVA [4] hardware, the bus-based standard that was specially developed in BINP by the beginning of 90-th.

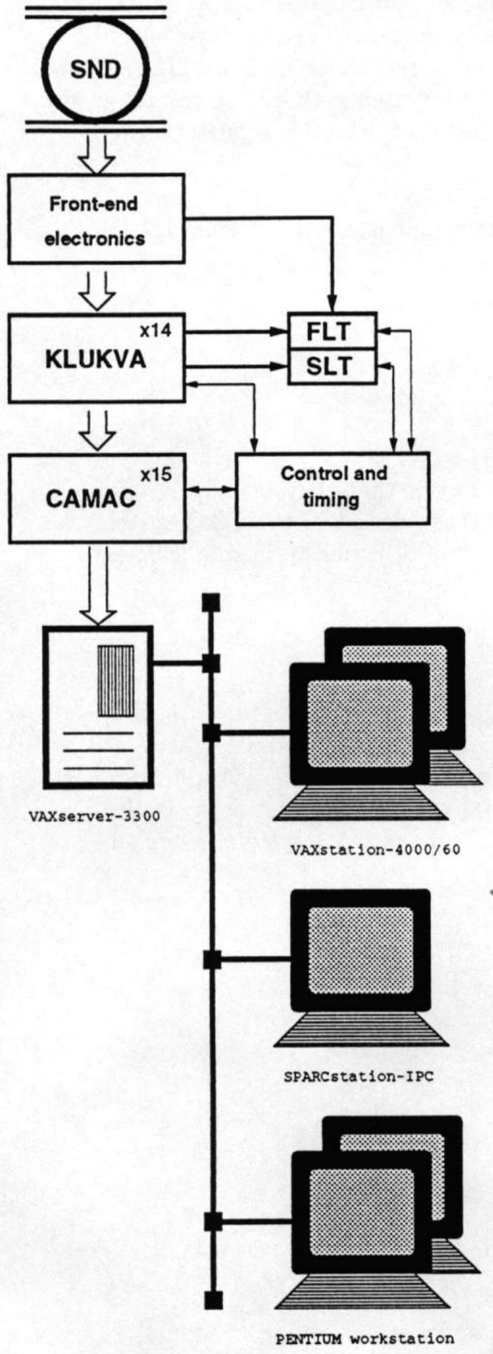

Fig. 1. SND DAQ system architecture

The DAQ system has a three-level trigger scheme. The first two levels FLT and SLT are implemented in hardware. They are tightly integrated with the readout electronics by

KLUKVA timing and control scheme. The third level trigger is a special fast computer code running at the data acquisition computers. The KLUKVA system performs automatic zeroes suppression and event fragments buffering in their internal RAMs. The full event building is done by the front-end DAQ computer which reads the event fragments from these RAMs through KLUKVA to CAMAC interfaces.

The computing environment of the experiment is composed from the Local Area VAX cluster of three VAXes running VMS operating system and SPARC and PENTIUM workstations running UNIX operating systems, all connected via 10 Mbit/sec Ethernet LAN. The resources of this cluster are shared by the DAQ and OFF-LINE systems. One VAX computer is dedicated for the special use as a front-end interface to the experimental equipment hosted in KLUKVA and CAMAC racks.

The experimental data are stored by the experimental data management system ART [5], that was specially developed for the SND experiment.

3 Software

One of the specifics of the HEP experiments with detectors is that they usually last for quite a long period of time(up to 10 years). This is why the design of the DAQ system software must be based on the principles of the scalability and flexibility in mind. These goals at the SND DAQ system were achieved by building the system which:

- is distributed;
- has several(vertical) layers;
- is functionally-modular.

The following 3 layers were identified in the DAQ system architecture: Service, Application, and Interactive. Each layer is represented by a set of programs or processes each one implementing a separate DAQ function. The software "map" of the DAQ system layers and their interaction to each other is shown in the Fig. 2.

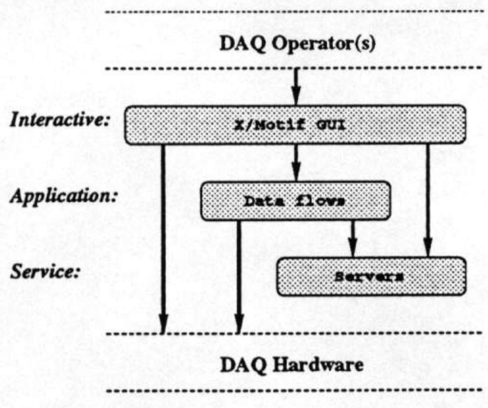

Fig. 2. The layers in the DAQ software

The Service layer has two general classes of functions. Firstly, it provides access to the DAQ system resources and information (formatted detector events, scalers, systems status, etc.). And secondly, it provides communication tools for the system components. The following distributed services have been developed and incorporated into the DAQ system:

- IMAN - status and control information management system [6];
- DEM - distributed buffers manager [7];
- RPD - distributed run parameters data base [8];
- ART - experimental data management system;

In addition, two services were derived from the CERN MODEL [9] project: OSP and EMU.

The Application layer includes the following core functions of the DAQ system, implemented as permanently running processes or groups of them:

- run control;
- detector events readout, formatting, rejection (TLT), and recording;
- full events reconstruction for some portion of events;
- scalers readout and recording;
- luminosity monitoring;
- communication with the accelerator control system;
- on-line electronics control.

TLT is implemented as a fast computer code running on the data acquisition computers. The trigger code was derived from SND OFF-LINE [10] reconstruction software. The trigger checks and rejects events background before recording. In addition TLT identifies collinear events of Bhabha scattering and two gamma annihilation. These events are used to monitor collider luminosity.

This Interactive layer serves as an interface between DAQ system operators and the system itself. An operator is given the ability to run about 80 interactive programs and scripts organized in a hierarchical set of menus. The following classes of functions are implemented here:

- management of the Application and Service layers;
- interaction with the run control system;
- run parameters manipulation in the corresponding data bases;
- sub-detectors calibration;
- electronics tests;
- detector control;
- various visualization programs.

The DAQ code was written on C and FORTRAN-77 programming languages.

4 Conclusion

The current version of the data acquisition system allows to process 1 kByte detector events with a maximum rate of 100 Hz before the TLT. During the first two years of experiments with SND there were acquired about 5 inverse picobarn of experimental data. About $5 \cdot 10^7$ of raw events were recorded.

In the nearest future the hardware platform of the system will be essentially upgraded due to planned increasing of the VEPP-2M luminosity [11]. The flexibility of the already existing software, achieved by using of the long-term solutions in the system design, will allow to minimize the code modifications.

References

[1] A.N.Skrinsky, Proceedings of Workshop on Physics and Detector for DAΦNE, INFN, Frascati, 1995.

[2] V.M.Aulchenko et al., Proc. Workshop on Phys. and Detectors for DAΦNE, Frascati, April 1991, p. 605

[3] V.M.Aulchenko et al., Proc. The 6th Inter. Conf. on Hadr. Spectroscopy "Hadron 95", Manchester, UK, 1995

[4] V.M.Aulchenko et al., Data acquisition system for new detectors in INP. Proceedings of the International Conference on Calorimetry in High Energy Physics, FNAL, 1990

[5] A.A.Korol, Prep. BINP 94-62 (in Russian)

[6] Information management system, SND internal note, BINP.

[7] Distributed Events Manager, SND internal note, BINP.

[8] Run parameters data base, SND internal note, BINP.

[9] D.M.Sendall at al., MODEL: A Software Suite for Data Acquisition, CERN. DD division report, DD/89/26, 1989.

[10] K.Hanssgen, V.N.Ivanchenko, this Proceedings.

[11] A.N.Filippov at al., High energy accelerators. Hamburg, 1992.