EDOC130

EUROGAM PROJECT

Multi-Dimensional Analysis of γ-ray Data

Edition 1.0
October 21th 1991

S. Flibotte, Uwe J. Huttmeier
Centre de Recherches Nucléaires, Strasbourg
CNRS-IN2P3 France

Abstract

A program has been developed to analyse high-fold γ-ray coincidences. This paper presents the basic concepts used in the program. Few properties of multi-dimensional γ-ray peaks are discussed. Examples of memory and storage requirements are given. The problem of background subtraction is examined.

Prototypes of the program exist for 3- and 4-fold data analysis. A version will be ready for circulation shortly.

I. Introduction

The future generation of high-resolution γ-ray spectrometers, such as Eurogam, Gammasphere or Euroball, are designed for acquisition of high-fold γ-ray coincidences with very high statistics. High spin experiments proposed for these large Ge arrays generally assume a trigger of $N \geq 4$ Ge coincidences. The main idea is that the move to higher-fold coincidences increases the resolving power anywhere from 4 to 7 (or greater, depending on the respective design) for each additional coincidence required.

The analysis of the vast datasets obtained from the new arrays poses a variety of challenging problems. In order to take full advantage of the increase of resolving power, it will be necessary (see below) to analyse the data in a multi-dimensional space of γ-ray energies. Present techniques of sorting events into a two-dimensional histogram have severe limitations when extended to higher dimensions. For example, a four-dimensional histogram with 4096 channels per axis would require storage of $N_c = (4096)^4 \simeq 3 \times 10^{14}$ channels. Assuming each channel is limited to 255 counts, i.e., 1 byte, this would amount to a total of 300 Tb (Tera= 10^{12}) of memory, while presently available large direct-access storage devices provide of the order of 1 Gb (Gigabyte) of memory.

The actual datasets obtained with a device like Eurogam, however, will be considerably smaller than the total number of channels discussed in the example above. Thus most of the channels of a large multi-dimensional histogram would have zero counts, which suggests that other means of data storage may be advantageous. After obtaining a realistic upper limit of the total number of events that may be obtained in a typical γ-ray experiment with a device such as Eurogam, alternative techniques of storing high-fold data will be discussed in detail and analysed using typical examples in different dimensions. The advantage of reduced storage requirements generally comes at the cost of more CPU intensive sorting and searching algorithms. The aim is to estimate the feasibility of such techniques and to deduce the requirements on CPU power, storage criteria and the overall costs. But before going into computational details, a few aspects of multi-dimensional γ-ray analysis shall be highlighted to substantiate why it is worth the effort to go into higher dimensions in the first place.

II. Properties of γ-ray Peaks in n Dimensions

For the following discussion it shall be assumed that a γ-ray peak is adequately represented by a Gaussian in one dimension. A n-dimensional peak is then obtained by multiplication of n Gaussians

$$P(\vec{x},\vec{\mu},\vec{\sigma}) = \prod_{i=1}^{n} \frac{1}{\sigma_i\sqrt{2\pi}} \exp\left[-\frac{1}{2}\Big(\frac{x_i-\mu_i}{\sigma_i}\Big)^2\right],$$

where $\vec{x} = \{x_1,\ldots,x_n\}$ are channel numbers, $\vec{\mu} = \{\mu_1,\ldots,\mu_n\}$ are the centroids and $\vec{\sigma} = \{\sigma_1,\ldots,\sigma_n\}$ are the standard deviations in the respective dimensions. Assuming that the Gaussians have the same widths σ in all dimensions,

$$\sigma = \sigma_1 = \ldots = \sigma_n$$

and arbitrarily setting $\mu = \mu_1 = \ldots = \mu_n = 0$ leads to

$$P(\vec{x},\sigma) = \frac{1}{(\sigma\sqrt{2\pi})^n} \exp\left[-\frac{1}{2}\Big(\frac{x_1^2+x_2^2+\ldots+x_n^2}{\sigma^2}\Big)\right].$$

Thus for peaks with similar energies the width of the n-dimensional Gaussian will be

$$\sigma_n = \frac{1}{\sqrt{n}}\sigma,$$

i.e., the width will be reduced by a factor of $1/\sqrt{n}$. (Note that this will not lead to a bell-shaped peak.) For example, if the FWHM (full width at half maximum) of a one-dimensional peak is chosen as a gate, the corresponding gate on a four-dimensional matrix should only be half as wide. This will lead, in general, to a considerably improved peak-to-background ratio and correspondingly cleaner projected spectra without losing statistics.

Setting $(n-1)$-dimensional reduced gates on a list of γ rays, say in a previously known band, and extracting the resulting one-dimensional spectrum, summed over all the combinations of gates, can of course be done on data stored in event mode on a sequential access medium. The resulting spectrum, however, will in general provide information on possible new gates at which point it would be very desirable to be able to extract the information from the entire dataset stored in a more efficient form. Analysing the dataset in this form will take advantage of the considerable reduction in background due to its scattering into a vast space. The width of the peaks in the final one-dimensional spectrum, however, will still be the original σ, not the reduced width.

In order to fully utilize the additional gain in resolving power in the nth dimension, one could envision a n-dimensional Gaussian peakfitting algorithm to identify all n-tuplets and store the resulting information in some coincidence data base. This approach would take full advantage of the reduced peak width and the resulting increase in resolution. In addition this would provide a means to determine γ-ray energies with a precision previously not achieved in one-dimensional Ge spectra.

Visual analysis in greater than one dimension may also be a powerful tool that would greatly benefit from efficient data storage. Two dimensional planes perpendicular to the diagonal, so-called rotational planes, may considerably facilitate the identification of rotational structures with a constant moment of inertia. Such cuts on a n-dimensional matrix would require the summation over all corresponding planes in $(n-3)$ dimensions. Cuts in two or three dimensions parallel to the diagonal in n-dimensional space or some $(n-i)$-dimensional subspace may provide additional information. In theory these techniques could also be generalised using planes with varying curvature to identify structures with non-constant moments of inertia.

The analysis of multi-dimensional γ-ray spectra is at present still in its infancy. With time one may think of new techniques that could extract the information of interest more effectively. One of the biggest problems are the limitations of the human brain of visualising objects in greater than three dimensions. Approaches using the terminology of three-dimensional geometry will not be adequate to conquer the problem. Conceptionally new mathematical appoaches that make interdisciplinary connections to such fields as differential geometry may be the way of the future (and that future may not be far).

Clearly, efficient storage of the n-dimensional information is essential to perform any sophisticated analysis "on-line" and on a reasonable time scale.

III. Alternative Storage Techniques

a. Estimate of the Data Size

The present bottleneck in the design of data aquisition systems for the large Ge arrays seems to be the current limits of i/o transfer rate to the on-line storage medium. The write speed of Exabyte tapes is at present limited to 250 kb/s, although an improved design with a transfer rate of about 500 kb/s may be available by the time the first array will be in operation (which will be some time in late summer of 91 for Eurogam 45). Assuming that data is written to one tape at a rate of 500 kb/s (or to two tapes at a rate of 250 kb/s) for 5 days would yield about 2×10^{11} bytes on tape, or the equivalent of about 100 Exabyte tapes, each holding the present limit of about 2 Gb. Furthermore, assuming that each event consists of at least about 20 bytes (i.e., a four-fold event with 4×2 bytes for energy, 4×2 bytes for time and some additional hitpattern information) the total dataset would approximately contain $N_e \approx 10^{10}$ events. This appears to be a realistic *upper* limit for a typical dataset of four-fold events involving only Ge detectors.

In order to obtain an estimate of the number of four-fold suppressed events produced in an experiment assume that a typical heavy-ion fusion-evaporation reaction yields about 50000 compound nuclei per second (e.g.: the reaction ^{124}Sn(^{30}Si,5n)^{149}Gd with a beam intensity of 3 particle nA, target thickness of 1 mg/cm^2 and fusion cross section of about 600 mb). For the first phase of Eurogam with 45 detectors the probability of a suppressed four- or higher fold ($F_s \geq 4$) event is estimated to be about 14.3%, resulting in a total of $N_e = 3 \times 10^9$ events after five days. The corresponding probability of $F_s \geq 5$ events is about a factor of 3 smaller, i.e., $N_e = 10^9$ events.

b. List Storage

As indicated in the example of the introduction, storing of the order of 10^{10} events in a four-dimensional histogram with about 3×10^{14} channels on a direct access medium is at present not only practically impossible but also not very efficient, since on average only one in 3×10^4 channels would have a count. An alternative to a histogram would be to store the data in a list in the form:

$$\{E_1, E_2, \ldots, E_n\}, \tag{A}$$

where the E_i's are the γ-ray energies. The storage requirement can be additionally reduced by storing all equivalent permutations of energies only once; e.g., by requiring

$$E_1 \leq E_2 \leq \ldots \leq E_n, \tag{1}$$

or some equivalent condition. The resulting reduction factor is $(1/n!)$. (This would destroy the information on which event came from which Ge detector, i.e., this would not be suitable for DSAM lifetime measurements.) Each energy parameter E_i requires a minimum of 2 bytes of storage, thus each event will be stored in $2n$ bytes. Further reduction is achieved by creating a set of sublists such that each energy parameter can be expressed in one byte or a fraction of one byte. This is achieved at the expense of additional parameters pointing to the storage location and the length of the variable length sublists. This technique can be generalised to a hierarchy of lists of events, each list containing, in addition to a number of events, one parameter pointing to a sublist, each sublist containing the pointers to a sub-sublist, etc.[PK 90].

At some point, when a sufficient number of events occur more than once, it will be more efficient to store the data in the form

$$\{E_1, E_2, \ldots, E_n, N\}, \tag{B}$$

where N is the equivalent to the number of counts in a histogram. The memory required to store N identical events in a set of sublists in this way is $(n+1)$ bytes, assuming the number of counts, N, is less than 255. Storing the data in this format has the additional advantage of simplifying the sorting and searching operations necessary in most of the following stages of analysis. For instance, subtracting a general background from the data (which will be important to reduce the data to a managable size, see below) is now straightforward if *some* of the events below a given threshhold can be *uniquely* identified as background. However, caution has to be applied since such a process potentially reduces the effective sensitivity of detecting low cross section events. The problems associated with background subtraction will be discussed in detail in Section 4.

Further reduction of the storage space is derived from the fact that with a peak-to-total of $P/T \approx 50\%$ over 90% of the events in $n \geq 4$ dimensions are associated with background. Since this background will be distributed over a very large space, almost all occupied background channels will have only one count (for a realistic dataset, i.e., $N \leq 10^{10}$ events). Thus by storing the number of counts per channel in the form of *one byte* we waste 7 out of 8 bits for over 90% of the events.

This can be improved by means of a compression code that would assign only *one bit* for channels with one count (the most frequent ones), two bits for channels with two counts, etc. The number of bits required to code the number of counts in a channel, using a particular form of the so-called Huffmann coding technique, is one bit per count; that is, for events with more than 8 counts per channel, this priscription will require more than one byte — but such events will be very few and far between. Again, these savings of memory do not come for free but at the expense of a significant increase of CPU time in order to decode (decompress) the coded information.

c. Examples

The tables used in this section are divided into two parts. The upper halfs present the storage of the events in format A ($\{E_1, E_2, \ldots, E_n\}$), i.e., all identical events are stored as often as they occur, but always in some ordered fashion (cf. condition (1) in Section III-b). Format A is very efficient for storing "sparse" datasets with nearly all of the occupied channels having $N = 1$. Due to the fact that the data records have fixed length and that the number of counts does not have to be decoded, format A is particularly suitable for data analysis systems with CPU constraints, e.g., single processor computers.

The lower halfs of the tables show the general requirements on direct-access storage space and physical memory for data storage using format B. For "denser" datasets, with a relatively larger fraction of events with $N \geq 1$, it is clearly more efficient to store identical events only once with an additional multiplicity counter, N, which is stored in encoded form. However, in order to project gated spectra in a reasonable time, it is important that sufficient processing power is available to perform the additional bit-shift and -extracting operations necessary to decode N, which makes format B more suitable for multi-processor machines.

It should be noted that the final decision on the storage format depends critically on the available hardware. For instance, i/o transfer rates from and to the direct-access storage medium could impose a more serious bottleneck on performance than CPU processing power.

The notation used in the examples will utilise the following quantities:

N - the total number of events of the dataset
N_c - the number of channels of a histogram
ρ - the average density of events
l - the number of lists per dimension
n_l - the total number of lists
k - the number of elements per list per dimension
N_e - the total number of elements per list
M_e - the memory required per event (in bits)
M_l - the memory required for all lists
M_p - the memory required for pointers
M_t - the total memory required

The average density of events, ρ, is determined from

$$\rho = \frac{N}{N_c},$$

where the number of channels of a n-dimensional histogram, reduced by all equivalent permutations of channel numbers, is given by

$$N_c = \binom{m+n-1}{n} \approx \frac{m^n}{n!} \quad \text{for large } m,$$

where m is the number of channels per dimension and the total number of lists is given by

$$n_l = \binom{l+n-1}{n}$$

l	k	n_l	M_p	N_e	ρN_e	M_e	M_l	M_t
256	8	2.8M	8.4M	512	21	9	68 Mb	76 Mb
128	16	358k	1.1M	4096	171	12	92 Mb	93 Mb
64	32	46k	1.4k	32768	1371	15	117 Mb	117 Mb
256	8	2.8M	8.4M	512	21	≥ 10	75 Mb	83 Mb
128	16	358k	1.1M	4096	171	≥ 13	98 Mb	99 Mb
64	32	46k	1.4k	32768	1371	≥ 16	120 Mb	120 Mb

Table 1: **Example 1** — $N = 60 \times 10^6$ events in $\mathbf{n} = \mathbf{3}$ dimensions with $m = 2048$ channels, i.e., $\rho = 1/24$; top: Format $\{E_1, E_2, \ldots, E_n\}$, bottom: Format $\{E_1, E_2, \ldots, E_n, N\}$.

The memory necessary to store the pointers to all lists is $M_p = n_l \times n$; this is a lower limit and will be larger if additional information, such as the length of the list or equivalently, the number of elements in the list are to be stored. The number of bytes to store all events in lists is given by $M_l = n_l \times \rho \times m \times M_e$; hence the memory required to store the entire dataset is $M_t = M_l + M_p$. Larger numbers in the tables are rounded off and the notation used is k = 10^3, M = 10^6, G = 10^9 and T = 10^{12} .

The first example deals with a set of $N = 60 \times 10^6$ existing triples and higher fold coincidence data that will be used to test some of the algorithms discussed below. Storage space and physical memory requirements are shown in Table 1 for the two different storage formats. The values of M_l and M_t in the lower half of all tables are approximate and a lower limit of one bit for N is assumed in the computation.

The number of lists is kept sufficiently large in this example in order to reduce the number of elements per list, since this will considerably reduce the sorting time per list. As a direct result, each element in a list requires less than one byte of storage; hence a portion of bits is not utilised at all. Thus the total memory requirement can be reduced significantly by using only a fraction of a byte of memory for each energy parameter.

Un upper limit for the number of lists is given by the condition that the physical memory should be large enough to store at least the list of pointers and all the elements of at least two lists in order to efficiently play back the data. Most computer systems used for the analysis of coincidence γ-ray data are presently equipped with sufficient physical memory to efficiently analyse two-dimensional matrices up to 4096 channels per dimension. Actual numbers are found in the range of 10 – 100 Mb. This is taken as an upper limit for M_p; cases where n_l yields much greater values for M_p would require correspondingly larger DRAM (Direct Random Access Memory).

The second example deals with a hypothetical scenario for Eurogam. Here it is assumed that the total number of events is $N = 3 \times 10^9$ four-fold coincidences (cf., Section III-a). Table 2 shows the respective figures for storing the entire dataset in $n = 4$ dimensions. Optimum storage is obtained for $l = 256$ and $k = 8$ with about 5.2 Gb (or ≈ 5.6 Gb for format B) of total memory. However, this would require a 733 Mb of pointers which could not be stored in the physical memory of currently readily available computer systems. This number is reduced significantly for $l = 128$ and $k = 16$ with an increase of less than 20% of total memory.

The third example shown in Table 3 deals with a Eurogam dataset of $N = 3 \times 10^9$ events

l	k	n_l	M_p	N_e	ρN_e	M_e	M_l	M_t
256	8	183M	733 Mb	4096	56	12	4.5 Gb	5.2 Gb
128	16	12M	47 Mb	65536	894	16	6.0 Gb	6.0 Gb
64	32	766k	3 Mb	1M	14305	20	7.5 Gb	7.5 Gb
256	8	183M	733 Mb	4096	56	≥ 13	4.9 Gb	5.6 Gb
128	16	12M	47 Mb	65536	894	≥ 17	6.4 Gb	6.5 Gb
64	32	766k	3 Mb	1M	14305	≥ 21	7.9 Gb	7.9 Gb

Table 2: **Example 2** — $N = 3 \times 10^9$ events in $\mathbf{n} = \mathbf{4}$ dimensions with $m = 2048$ channels, i.e., $\rho = 4.08 \times 10^{-3}$; top: Format $\{E_1, E_2, \ldots, E_n\}$, bottom: Format $\{E_1, E_2, \ldots, E_n, N\}$.

l	k	n_l	M_p	N_e	ρN_e	M_e	M_l	M_t
128	16	309M	1.5 Gb	1.0M	3.5	20	7.5 Gb	9.0 Gb
64	32	10.4M	52 Mb	34M	111	25	9.4 Gb	9.4 Gb
32	64	377k	1.9 Mb	1.1G	3559	30	11.3 Gb	11.3 Gb
128	16	309M	1.5 Gb	1.0M	3.5	≥ 21	7.9 Gb	9.5 Gb
64	32	10.4M	52 Mb	34M	111	≥ 26	9.8 Gb	9.8 Gb
32	64	377k	1.9 Mb	1.1G	3559	≥ 31	11.6 Gb	11.6 Gb

Table 3: **Example 3** — $N = 3 \times 10^9$ events in $\mathbf{n} = \mathbf{5}$ dimensions with $m = 2048$ channels, i.e., $\rho = 9.93 \times 10^{-6}$; top: Format $\{E_1, E_2, \ldots, E_n\}$, bottom: Format $\{E_1, E_2, \ldots, E_n, N\}$.

to be analysed in 5 dimensions. Figures for $l = 256$ and $k = 8$ are no longer included in Table 3, since the entire list of pointers would require close to 50 Gb of memory in this case. Recommended parameters yielding a pointer table that can be stored in DRAM are $l = 64$ and $k = 32$ resulting in a total memory size of 9.4 Gb for format A, or 9.8 Gb for format B respectively.

IV. Background Subtraction

The examples discussed in the previous section, although upper estimates for Eurogam, show that the memory requirements to store the entire dataset are quite large. A significant reduction of the storage requirements is obtained by eliminating some of the events associated with background. We shall procede by obtaining an estimate of the number of photopeak events in n dimensions compared to the number of events in the background.

In order to estimate the fraction of the events that will be distributed over the respective background regions in n dimensions, a realistic assumption of the peak-to-total ratio (P/T) for the large volume Ge counters with 70% – 80% efficiency used, e.g., in Eurogam, is needed. Source measurements with the new detectors give numbers approaching $P/T \approx 0.60$. However, this does not include degradation of P/T due to events where a neutron and a γ ray or two γ rays (pile-up) are detected at the same time in the Ge cristal. Note that peak events will also be lost in the case of a γ ray being detected in the Ge detector and simultaneously a neutron hitting the associated BGO suppressor, but this should have no effect on P/T. Due to these two factors the following calculations will assume a $P/T = 0.50$

P	B	f_p	prob(%)	N_P	N_B	N_c
4	0	1	6.25	81	0	231
3	1	4	25.0	27	1k	0.7
2	2	6	37.5	9	1M	$< 10^{-2}$
1	3	4	25.0	3	1G	$< 10^{-5}$
0	4	1	6.25	0	1T	$< 10^{-7}$

Table 4: Probabilities of detecting an event in all possible 4-dimensional full energy peak(P)/background(B) combinations. f_p is a permutation factor, N_p and N_B are the number of peak and background channels, respectively, and N_c is the average number of counts per channel.

for a 1 MeV γ ray.

The present example assumes a four-fold coincidence of four γ rays with energies near 1 MeV and $E_1 \leq E_2 \leq E_3 \leq E_4$. Table 6 shows all combinations of P γ rays in the peak and B γ rays in the background. In order to estimate the average number of counts per channel, a dataset of $N = 3 \times 10^9$ events and an average γ-ray multiplicity of $M_\gamma = 20$ is assumed, i.e., four-fold coincidences will be distributed over $\binom{20}{4} = 4845$ points in in four dimensions, yielding about 3×10^5 counts per coincidence peak. Only about 6% of all events will be in the four-fold full energy peak, which corresponds to about 231 counts per channel if the peak in each dimension is 3 channels wide and the counts are uniformly distributed. The remaining 94% are backgound events, of which about 70% have less than 10^{-2} counts per channel and 25% have about 0.7 counts per channel on average.

The above example should serve only as a rough estimate. Events from an actual experiment will certainly not be evenly distributed, and the P/T will depend on the energy of a given γ ray. Nevertheless, the point of the present exercise is that the majority of the events associated with background will have less than one count per channel, should be fairly well isolated and thus can be subtracted.

However, some care must be taken when eliminating events with a number of counts less than some threshhold (say 1) from the dataset, since this may potentially reduce the sensitivity for events with very low cross section. The effect of background subtraction shall therefore be discussed in detail in the following example.

Assume we have $N = 3 \times 10^9$ four-fold coincidence events to be analysed in four dimensions. Let the cross section of a given superdeformed band of interest be $\sigma_{sd} = 0.01 \times \sigma_t$, where σ_t is the total cross section. This will thus yield $N_{sd} = 3 \times 10^7$ events associated with superdeformed transitions. With a $P/T = 0.50$ we obtain

$$(P/T)^4 \times N_{sd} = 1.88 \times 10^6$$

events in all four-dimensional peaks associated with the superdeformed band. An average γ-ray multiplicity of $M_\gamma = 20$ will result in $\binom{20}{4}$ superband peaks. Assuming that all transitions have the same intensity, this corresponds to 387 counts in each four-dimensional peak.

The highest transitions presently known in the yrast superband of ^{149}Gd (to pick an example at random) has an intensity of about 6×10^{-2} of the strongest transition in this band. By subtracting all events with $N_c = 1$ we would thus limit our sensitivity to roughly 3×10^{-3}, which is still a factor of 20 better than the present limit. Extrapolating the intensity distribution as a function of spin of the highest states in this band [BH 90], the above limit would still enable us to observe the $78 \rightarrow 76\hbar$ transition — with an intensity of 1140 counts, obtained by summing the spectra of all $\binom{20}{3}$ possible combinations of three-dimensional gates. This is about $10\hbar$ higher than the highest spin state observed at present.

The savings in storage space of such background-subtracted lists are dramatic. Since the number of counts of the average background event will be quite a bit less than one, on average only about 6% of the events will have $N_c \geq 1$. Thus the above technique would reduce the dataset by over 90%, i.e., even a dataset consisting of $N = 10^9$ five-fold events could be stored in five dimensions requiring less than 1 Gb of memory, which easily fits on a larger harddisk.

On the other hand one may savely assume that a peak with as little as 100 counts in the final spectrum could be clearly identified in four-fold data. Utilising the unsubtracted dataset this would yield a sensitivity of 2.3×10^{-4} of the strongest transition in the superband (corresponding to the $88 \rightarrow 86\hbar$ transition). Thus by eliminating all events with $N_c = 1$ we have effectively reduced our sensitivity by about one order of magnitude.

Losing about one order of magnitude of sensitivity is clearly not desirable for most experiments. The situation improves considerably when the background subtraction of the n-fold dataset is performed in $(n-1)$ dimensions. One could for instance tag each event in consecutive order, subsequently mark all events with $N_c = 1$ in the $(n-1)$-dimensional lists and then resort all events except the previously marked ones in n dimensions. In general this will increase the sensitivity by about a factor of n over the subtraction procedure in n dimensions.

Another possibility would be to subtract the background below a certain threshhold for a given sub-space, e.g., for all events with $E_\gamma \leq 1\,\mathrm{MeV}$ and/or all events which are not in coincidence with any previously known transitions of the nucleus of interest. In most cases, such a sub-space would be sufficiently large in order to reduce the entire dataset by a significant fraction.

V. Sorting and Searching

Having stored the entire dataset in list format, with or without background subtraction, has provided us with a much more efficient way of storage. But before one can commence with the data analysis, efficient algorithms for extracting the information of interest are required.

The two most elementary one-dimensional spectra generally required in the analysis of γ-ray spectroscopy data are a total projection spectrum and spectra in coincidence with one or several γ rays. The first case would require one pass through all the lists and, whether or not the events in a list are sorted, has no effect on CPU time. On the other hand, the process of extracting a subset of the events in a list, say events that are associated with a given γ-ray gate, can be done much more efficiently when all the elements of the list are sorted in some form.

It is proposed that the process of converting the data from event format, the way they exist on tape, to ordered list format is done in two stages. The first stage is composed of the following steps:

- read the data from tape, event by event,
- recalibrate each energy parameter to the desired dispersion,
- rearrange each event into the form $E_1 \leq E_2 \leq \ldots \leq E_n$,
- extract the parameters pointing to the appropriate list,
- condense each event keeping only the relevant bits for the list elements,
- write the event in condensed form into its corresponding list.

Events in a list will still be in random order and in this form each event will require a total of $(M_e - 1)$ bytes of storage, each parameter requiring $\log_2 k$ bits, since we have not yet utilised the multiplicity counter of the events. (A useful statistics at this point may be to extract the spectrum of the number of events per list.)

The second stage (Format B) will be to read in all the events from a list, each as a simple bitstream, sort the elements of the list in say increasing order (i.e., $E_i \leq E_j$), subsequently scan the ordered list for all equal events and finally write the events to memory in the converted form $\{E_1, E_2, \ldots, E_n, N\}$, where N is the multiplicity, i.e., the number of times an event occured.

With the data already sorted into considerably smaller lists, the problems associated with efficient sorting and searching of the entire dataset are already significantly simplified. Efficient algorithms for sorting 'keys' are discussed by [DK 73] and are quite applicable for the sorting step in Stage 2 above. For example, sorting a list containing 100 events with a program like Heapsort would require about $16N \times \log_2 N = 11000$ instructions per second. On a 10 MIPS computer the 12M lists of a four-dimensional matrix could be sorted in less than four hours, excluding the time required for i/o operations (which will constitue a considerable overhead). The best sorting algorithm depends to some extend on the number of events per list, which will considerably fluctuate as a function of energy and will also on the particular experiment; however, the time spent on developing of the most suitable algorithm should be optimised under the constraint that sorting is something that will in general be done only once with a given dataset.

In the case that the total number of dimensions, n, and the dimension of a list, k, is small and the number of events per list is relatively large, an alternative algorithm can be quite efficient. The basic idea is to sort the data into an n-dimensional array with k elements per dimension. The total number of integer instructions per list is in this case approximately k^n, independent of the number of events per list.

VI. Sortware Tests

Programs were developed in C++ for three- and four-fold data analysis. Various algorithms were tested using a data set of nearly 50×10^6 three-fold and about 3×10^6 four-fold events.

The hardware consists of a single-processor Sun SPARCstation2 with a performance rating of about 28 MIPS and 6.2 Mflops (max. in single precision), a 1.2 Gb Seagate hard disk with a mean access time of 15 ms and SCSI-2 interface with a tested data transfer rate of about 4Mb/s for read - and 500 kb/s for write operations. In the case of the three-fold data, a good compromise of storage space reduction and fast access time was achieved for data storage in 5984 lists, each list containing 512 sub-lists using 4 bits per energy parameter hence providing 4096 channel resolution per dimension. Extracting a one-dimensional histogram of all events in coincidence with gates on two γ transitions from these lists is achieved on a time scale of seconds, depending on the width of the gates and the overall size of the data set.

For further details see *A Brief Manual for high-fold data Analysis.*

Appendix A: Hardware Prices

Physical Memory: 10 kFF per 16 Mb (avec remise)

SUN SPARCstation2: 80 kFF (b/w monitor, avec remise)

Seagate Wren7 harddisk: 25 kFF for 1.2 Gb unformatted incl. SCSI-2 interface

(Note: In the present case about 18% of disk capacity was lost due to formatting, about 8% loss due to file structure with about 400000 files max.)

Acknowledgements

I am grateful for stimulating discussions with many collegues, including Bernard Haas, Jean-Pierre Vivien, Norbert Kaiser, Johann Bartel, Marianne Dufour and Peter Kutt.

References

[PK 90] Peter Kutt, *private communication.*

[BH 90] Bernard Haas, *private communication.*

[DK 73] Donald Knuth, *The Art of Computer Programing*, Vol.III.