ORNL/CP-100996

Analysis of High-Fold Gamma Data

D.C. Radford[1], M. Cromaz[2] and C.J. Beyer[3]

[1]*Physics Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831*
[2]*Nuclear Science Division, Lawrence Berkeley National Laboratory, Berkeley, California 94720*
[3]*Department of Physics and Astronomy, Vanderbilt University, Nashville, Tennessee 37235*

Abstract. Historically, γ-γ and γ-γ-γ coincidence spectra were utilized to build nuclear level schemes. With the development of large detector arrays, it has became possible to analyze higher fold coincidence data sets. This paper briefly reports on software to analyze 4-fold coincidence data sets that allows creation of 4-fold histograms (hypercubes) of at least 1024 channels per side (corresponding to a 43 gigachannel data space) that will fit onto a few gigabytes of disk space, and extraction of triple-gated spectra in a few seconds. Future detector arrays may have even higher efficiencies, and detect as many as 15 or 20 γ rays simultaneously; such data will require very different algorithms for storage and analysis. Difficulties inherent in the analysis of such data are discussed, and two possible new solutions are presented, namely adaptive list-mode systems and "list-list-mode" storage.

1. INTRODUCTION

Large γ-ray detector arrays such as GAMMASPHERE and EUROBALL now generate three-, four-, and even five-fold Ge-detector coincidence data in useful quantities. The unsurpassed sensitivity of these arrays has allowed observation of γ-ray cascades that are well beyond the detection limits of previous-generation arrays. However, in order to realize this sensitivity, the data must be analyzed in high fold, and the size of data sets generated by these instruments can make it very difficult to extract all of the useful but detailed information contained in them. Furthermore, limited available computer resources have meant that researchers have typically only been able to make simple histograms of data sets of up to 3-fold; for folds higher than three, list-mode storage has had to be used, or the events reduced to fold less than four by applying gates during the sorting process.

General analysis of γ-γ and γ-γ-γ coincidence data to construct complete and consistent nuclear level schemes is facilitated by sophisticated computer programs such as "escl8r" and "levit8r" [1], that can extract the physically interesting numbers from the raw data and present them to the physicist in an easily assimilated manner, keep track of all γ-ray assignments and expected coincidence intensities, and to quickly find and report major discrepancies between a proposed level scheme and the data so that they can be understood and corrected. These programs include an interactive display of the user's proposed level scheme as an integral part of the Graphical User Interface. An equivalent tool for 4-fold analysis ("4dg8r") has recently been developed. The techniques that allow creation and storage of 4-fold histograms (hypercubes) that will fit onto a few gigabytes of disk space, and yet allow access to triple-gated spectra in only a few seconds, are briefly described below in section 3.

The next generation of γ-ray detector arrays is already being discussed; if constructed, an array such as GRETA could produce copious quantities of very-high fold ($F \sim 15$) data that would require analysis in 6 or 7 dimensions. Analysis of these data would present very significant new problems, many of which are discussed below in section 2. The most significant problems have less to do with actual storage of the data than with the time required to generate gated spectra; in a simple-minded list-mode scheme, all the data need to be read and processed when gates are applied. Techniques which overcome these problems will need to be developed; two attempts to do so are described in sections 4 and 5.

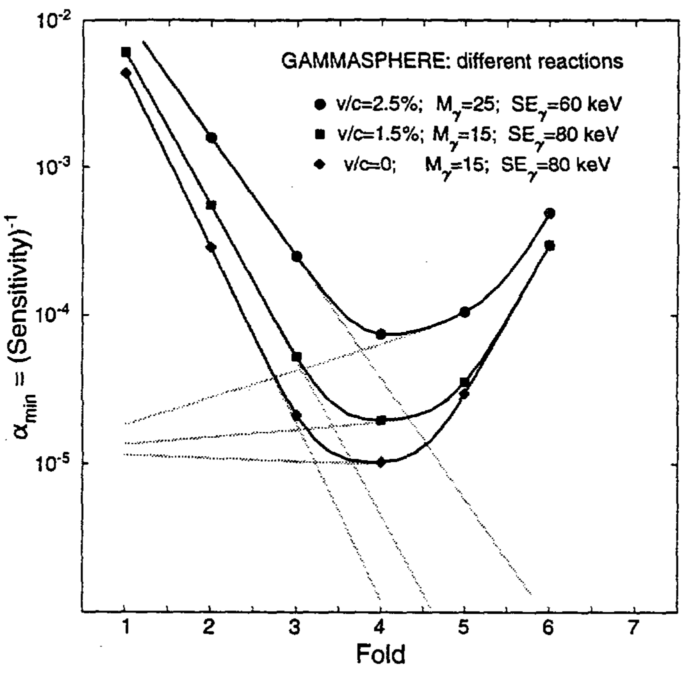

FIGURE 1. Calculated sensitivity limits for GAMMASPHERE used in conjunction with three different values of γ-ray multiplicity, recoil velocity, and average γ-ray energy separation. The calculations are performed as described in ref. [2].

2. DIFFICULTIES INHERENT IN HIGH-FOLD ANALYSIS

As new detector arrays have improved in efficiency and resolving power, their sensitivity has improved tremendously. Higher efficiency allows collection of higher-fold coincidences, and each additional fold allows an increase in the peak-to-background ratio for weak coincidences, typically by a factor of about 4 to 10. This is illustrated in Figure 1, which shows calculated sensitivity limits for GAMMASPHERE used in conjunction with three different values of γ-ray multiplicity and recoil velocity. Note that in all cases, the optimum calculated sensitivity occurs for fold 4, so that to realize the full sensitivity, the analysis must somehow be performed in a four-dimensional space.

Unfortunately, the large size of a four-dimensional data-set gives problems with data storage and data access speeds. Until recently, this meant that researchers were generally restricted to 2D and 3D histograms (matrices and cubes), so that a four-dimensional analysis required some form of data selection (gating) during the replay (histogram construction) phase. Furthermore, the complexity of the analysis of these data makes computer-assistance in *examining* the data, for example as in the programs escl8r and levit8r mentioned above, even more imperative than for 2- and 3-fold analyses.

The next generation of detector arrays, such as GRETA, will greatly compound these problems. For example, the sensitivity plot corresponding to Figure 1 but calculated for GRETA would have optimum sensitivity for fold 7, requiring a 7-dimensional analysis.

In order to illustrate some of the problems inherent in 4- and higher-fold analysis, we first define a few useful terms:

- The **Gamma-Ray Multiplicity** M_γ is the number of γ rays produced in a single event, and varies with the reaction used in the experiment. For rare-earth nuclei produced in heavy-ion fusion-evaporation reactions, for example, M_γ is typically $\sim$ 25 to 30.
- The **Detected Fold** F_D in an event is $\sim M_\gamma\epsilon$, where ϵ is the *total* efficiency of the detector array, *i.e.* the efficiency including both photopeak and residual unsuppressed Compton-scattered gamma rays.
- The **Analysis Fold** F_A is the fold or dimensionality in which the data are analyzed. Obviously we require that $F_D \geq F_A$.

- The **Number of Reactions** N_R produced during the experiment depends on beam current, target thickness and total reaction cross-section, and possibly also on any auxiliary detectors that perform some sort of selection on accepted events.

- The **Number of Events** N_E recorded during the experiment varies with the detected fold F_D as

$$N_E[F_D] \approx N_R \ {}^{M_\gamma}C_{F_D} \ \epsilon^{F_D} \ (1-\epsilon)^{M_\gamma - F_D}. \tag{1}$$

where the number of combinations

$${}^{M_\gamma}C_{F_D} = \frac{M_\gamma!}{F_D! \ (M_\gamma - F_D)!}. \tag{2}$$

- The **Number of Counts** N_C in a (possibly imaginary) F_A-fold histogram depends on the analysis fold F_A as

$$N_C[F_A] = \sum_{F_D} N_E[F_D] \ {}^{F_D}C_{F_A}, \tag{3}$$

where the number of combinations

$${}^{F_D}C_{F_A} = \frac{F_D!}{F_A! \ (F_D - F_A)!}. \tag{4}$$

With these definitions, we can now consider examples of typical values for these quantities for three different detector arrays. In a typical high-spin experiment, using a fusion-evaporation reaction to produce rare-earth nuclei, with several pnA of beam current, one can expect a reaction rate of the order of 10^5 per second. A three-day experiment thus produces $N_R \sim 3 \times 10^{10}$ reactions, with a typical γ-ray multiplicity of $M_\gamma \sim 25$.

The second-generation 8π-spectrometer array has a total efficiency of $\epsilon \sim 0.01$, resulting in $N_E[F_D \geq 2] \approx 7 \times 10^8$ events, and $N_C[F_A = 2] \approx 9 \times 10^8$ counts in a two-dimensional histogram (matrix). Due to a small fraction (10%) of 3- and higher-fold events, there are an average of about 1.3 counts generated by each event.

GAMMASPHERE and EUROBALL are third-generation arrays, with efficiencies of $\epsilon \sim 0.15$, resulting in $N_E[F_D \geq 4] \approx 1.6 \times 10^{10}$ events, and $N_C[F_A = 4] \approx 1.9 \times 10^{11}$ counts in a four-dimensional histogram (hypercube). Due to a significant fraction of 5- and higher-fold events, there are now an average of about 12 counts generated by each event.

The next generation of detector arrays, such as GRETA, might have efficiencies approaching $\epsilon \sim 0.6$, resulting in an average detected fold of about 15. Thus we get a useful event for essentially every reaction, $N_E[F_D \geq 7] \approx 3 \times 10^{10}$ (neglecting dead-time losses), and $N_C[F_A = 7] \approx 4 \times 10^{14}$ counts in an imaginary 7-dimensional histogram. The large detected fold means that there are now about 13000 "counts" generated by each event!

Incidentally, it is these large values of ${}^{F_D}C_{F_A}$ that produce "spikes" [3] in incorrectly-produced spectra from the sum of many gates. If each event is unpacked into F_A-tuples before the gates are applied, and many resulting gated spectra added together, then it is possible for many F_A-tuples from one individual event to meet the gating conditions. Thus a single γ-ray energy from a single event may be added into the resulting spectrum many times, producing "spiking". In order to avoid this problem with high-fold data, one should either avoid summing many gated spectra (since an individual $(F_A - 1)$-fold gate on F_A-fold data contains no spikes), or produce the summed spectrum directly from the raw data, without first unpacking into F_A-tuples, and make sure that each energy for each event is added to the spectrum at most once.

One conclusion that follows from the above considerations is that for analysis folds above four, storage of the data in histograms is no longer feasible. Firstly, the storage space requirements quickly become prohibitive, since the number of channels grows by several orders of magnitude with each fold. Secondly, as the detected fold becomes very large, many counts are produced by each event, so that compression algorithms are not able to make use of a small count density to greatly reduce the histogram size. Lastly, the problem of spiking, together with the need to sum many combinations of gates to obtain useful statistics, requires preserving the data in its raw form, rather than unpacking it into F_A-tuples as would be done in producing a histogram. Thus for $F_A \geq 5$, list-mode storage, where each event is essentially saved as a simple list of γ-ray energies, becomes the preferred storage mode even for very large numbers of events.

List-mode storage for quintuple γ-ray coincidence events has been used at several laboratories for some time [4]. This type of storage typically requires several bytes per event, and the main limitation is that in order to generate gated spectra, one needs to read through entire data-set and process every event. For large quantities of higher-fold data, this quickly becomes unrealistic, and in order to achieve reasonable access times we will require better schemes than those of simple list-mode storage.

3. FOUR-DIMENSIONAL HISTOGRAMS

The "RadWare" programs "escl8r" and "levit8r" [1] are designed to facilitate analysis of γ-γ and γ-γ-γ coincidence data, respectively. One unique aspect of these programs is that they include an interactive display of the current version of the user's proposed level scheme as an integral part of the Graphical User Interface, in order to assist the user in examining the data and extracting γ-ray intensities *etc.* Information about the detector efficiency and resolution, as a function of energy, allows the software to calculate *expected* coincidence spectra, on the basis of the proposed level scheme, for comparison with the corresponding *observed* spectra. It also enables least-squares fitting of the 2- or 3-dimensional data, so that γ-ray energies and intensities can be extracted from the data quickly and easily. In this section, we briefly describe an equivalent tool for 4-fold analysis ("4dg8r") that has recently been developed.

In developing this software for 4-fold histograms (hypercubes), the following goals were considered to be of primary importance:

- It should be able to run on standard workstations, without special requirements such as especially large amounts of RAM or disk storage.
- It should be able to handle a large number of counts (the current version uses up to two bytes per channel).
- The user should be able to extract triple-gated spectra interactively, *i.e.* in a few seconds.
- A typical histogram should occupy less than 9 GB of space on disk.
- Since some file systems have limits on the size of disk partitions and files, and since users may need to spread their hypercube over a number of different disks, it is important that the hypercube be able to be broken into several sub-files.

In order to cover a large energy range with as few channels as possible, the programs allow the use of a non-linear energy dispersion, as for "levit8r" [1]. The current version by default allows up to 1024 channels per side of the hypercube; this can be extended without major modifications. By symmetrizing the hypercube (*i.e.* ordering the four channel indices, $w \leq x \leq y \leq z$) we can reduce the corresponding number of hypercube channels to $(1024 \times 1025 \times 1026 \times 1027)/(4!)$. It should be noted, however, that this number is still in excess of 40 billion channels.

Since the number of channels in the hypercube is much greater than the the number of bytes in which we would like to store it, the only way to do this is to use data compression. The average count density in the hypercube is usually very low (about one count per channel or less), so that good compression ratios are obtainable. Since we require fast random access to the hypercube, the compression/decompression algorithm needs to be very fast, and we also need to compress small pieces of the hypercube individually. For this reason, the hypercube is subdivided into "minicubes", each 4 channels on a side, or $4^4 = 256$ channels total. These minicubes are individually compressed, and occupy a space of between 1 and 512 bytes on disk.

The compression algorithm used is explicitly designed for such small histogram segments, and provides an excellent trade-off between performance and high speed. It consists essentially of storing an optimum number of "bitplanes" of the hypercube, together with addresses of channels which have overflowed this number of bits. Each bitplane doubles the range of counts that can be stored without overflows, at the cost of $256/8 = 32$ bytes (since there are 256 channels in the minicube). Also, each overflowed channel within the minicube can be addressed with a single byte. Thus, once 32 or more overflows have occurred, the overflows occupy as much space as a bitplane, and the algorithm simply allocates that additional bitplane. A one-byte header for each minicube stores enough information for the decompression routine to know how many bitplanes and overflows are stored.

Individual compression of the minicube histogram segments, however, has the drawback that they now have different lengths on disk, so that locating one individual segment for decompression and analysis is no longer simply a matter of skipping over a calculable number of bytes. Rather, tables of pointers, or indices that map minicube numbers to locations on disk, are required, and there will in general be a trade-off between the size of these tables and the average number of interpolations (and hence time) required to find a given minicube. Furthermore, during the histogram construction phase of the analysis, the software needs to read and decompress portions of the hypercube, increment the channels, and recompress/rewrite. Since the count density increases during this process, the size of the hypercube portions will in general grow, and they cannot be written back into the space from which they were read. Thus the data need to be moved and space reallocated, giving rise to additional complexity in the file structure, and in the book-keeping aspects of the program.

Using this design, a package of programs for four-dimensional analysis has been written and included in the "RadWare" software package. In addition to "4dg8r" (the 4D version of "levit8r"), there are programs "4play" for data replay (histogram construction), and "pro4d" for histogram projection to 3D, 2D and 1D. Replay of a typical high-statistics GAMMASPHERE/EUROGAM experiment, on a standard workstation, can be expected to take of the order of 50 hours. Extraction of triple-gated spectra from the hypercube typically takes 1 to 5 seconds per gate, depending on the size of the cube and disk speeds, *etc.*

4. ADAPTIVE LIST-MODE STORAGE

As discussed in section 2, at folds 5 or above histogramming events is no longer practical. The alternative is list-mode based systems, which grow linearly rather than geometrically with fold. It is important to note that in list-mode systems, one does not unfold coincidences as this will cause the size of the database to grow geometrically. Realistically, the number of coincidences in a given experiment with current (or future arrays) will be less than 10^{10} as it is limited by detector efficiency and count rate considerations for Ge detectors. Given this and the capacity of current mass storage systems, space constraints on list-mode systems is no longer a major issue.

The challenge in list-mode systems is to quickly access events which meet a specific gating condition. Exhaustively testing each coincidence in a multi-gigabyte data set is not acceptable if one wishes to set gates interactively. To accomplish this, effective list-mode systems require two elements. First, events must be sorted so that coincidences likely to be accessed in a given query are "close" together in the database. For the purposes of high-fold gamma-ray spectroscopy, "close" refers to the Euclidean distance between coincidences in the hypercube. As data is fundamentally represented in computer memory (or on disk) as a one-dimensional array, implicit in this sorting is an approximate method to map the channel numbers $(x_1, x_2, x_3, \ldots, x_{F_D})$ to an integer key which may then be ordered. The second requirement for a list-mode system is an indexing mechanism for the sorted list of coincidences, so that groups of interesting coincidences can be accessed quickly. Unlike histograms, where the location of an interesting element can be calculated directly from its indices, list-mode systems require a set of indices so that entry points into the database can be specified.

These two goals of effectively sorting and indexing multi-fold coincidences can be met with an adaptive list-mode system. This is based on a data structure called a kd-tree, introduced by Bentley [5] for multi-key databases. The index for such a database is given by a binary tree where each node of the tree represents a volume of the hypercube. Using this data structure, one is able to partition into volumes whose size depends on the density of counts in that part of the hypercube. This adaptive list-mode system allows for greater granularity in densely populated areas of the cube, thus yielding greater performance.

Such a system has been implemented by Mario Cromaz at the Lawrence Berkeley National Laboratory, in the software package *blue*. It provides facilities for sorting into and taking gates on multi-fold list-mode data sets. Adaptive list-mode systems have good performance characteristics enabling one to take $(F_D - 1)$-fold gates on F_D-fold data sets in a few seconds. Gating time tends to be independent of fold but strongly dependent on the total counts in the list. This is a consequence of the ordering and indexing mechanism chosen.

5. LIST-LIST-MODE DATA STORAGE AND ANALYSIS

As discussed above in section 2, GRETA data would require $F_A \approx 7$, and should produce $N_E[F_D \geq 7] \sim 10^{10}$ events in a typical high-spin experiment. The large detected fold $< F_D > \sim 15$, together with the (relatively!) low analysis fold, means that there are now an average of about 13000 septuples generated by each event. This presents a real problem; since we need to be able to apply $F_A - 1$ gates on the data set, and each event can meet those gating conditions in ${}^{F_D}C_{F_A - 1} \sim 10^4$ possible ways, it seems that we need to read and inspect each and every event to see if it meets the gating conditions. Even the adaptive list-mode system of the previous section will suffer from this; it is primarily designed for setting $(F_D - 1)$-fold gates on F_D-fold data, and for the very-high-fold data discussed here almost all of the indices would have to be checked for matching events. In this section, a new solution to this problem is presented, a solution which has been entitled "list-list-mode".

Firstly, it is worth pointing out that a relatively small subset of the full data set produces most of the counts in a gated spectrum. The distribution of detected fold $N_E[F_D]$ is given by equation (1) above. For $\epsilon = 0.6$ and $M_\gamma = 25$, the resulting distribution has a mean at $F_D \approx 15$, and is rather wide. If we apply a random 6-fold gate to to that distribution, 75% of the counts in the resulting spectrum come from events with $F_D \geq 17$, which comprise only 27% of the total number of events. Indeed, 55% of the counts come from events with $F_D \geq 19$, or only 3% of the events. Thus if we save only events with $F_D \geq 17$, we lose very few statistics, but save a factor of 3 or 4 in storage requirements. The number of events remaining would be $N_E[F_D \geq 17] \sim 3 \times 10^9$, with an average fold of $< F_D > \sim 19$, and corresponding to $N_C[F_A = 7] \sim 8 \times 10^{13}$ 7-fold combinations.

We now address the question of how such events are to be stored on disk. We first look at one simple scheme that illustrates the principles involved, but does not perhaps give the optimum storage. More complicated compression schemes, that further reduce the space required to store the data set, could be added later, perhaps at the cost of a minor increase in processing time when the data are read back.

We assume that in each event there are, on average, about 18 energies, each distributed over 2048 channels. Let us order the energies in ascending channel number, so that an event consists of the set of channel numbers $(x_1, x_2, x_3, \ldots, x_{F_D})$, with $x_1 \leq x_2 \leq \ldots \leq x_{F_D-1} \leq x_{F_D}$. Then the average *difference* between the consecutive energies, $x_i - x_{i-1}$, is about 2048/18, or 114 channels. Since one byte (8 bits) can store numbers between zero and 255, we can use a byte to store each of the differences between the consecutive energies in the event, with some extra conventions to handle the special cases where a difference is 255 or more. In this way, an 18-fold event can usually be stored in 18 to 20 bytes. Different events will have different lengths, depending on the detected fold, and on the number of extra-large energy differences that require additional bytes.

In this list-list-mode scheme, we allocate different event-storage files for each of the different possible *event lengths* (in bytes). For example, for the data set discussed above, with $N_E[F_D \geq 17] \sim 3 \times 10^9$, we would have 14 or so different event lengths, from 17 up to about 30 bytes. The average length would be about 19 bytes, so the full data set could be stored in less than 60 GB.

The problem then becomes how to quickly find those very few events that meet the conditions of a 6-fold gate. That is, how do we find the needle of interesting events in the haystack of $\sim 3 \times 10^9$ events which fail to meet our gating conditions, without reading through all 60 GB to check each event in turn?

Imagine that for each channel x we could pick out the subset $\{e_x\}$ of all events that involve that channel x, that is, those events for which at least one of the detected energies is $E_\gamma = x$. Then, for a quadruple gate (w, x, y, z), for example, we want to specify four channels of the interesting events, so we want the subset

$$\{e_{wxyz}\} = \{e_w\} \bigcap \{e_x\} \bigcap \{e_y\} \bigcap \{e_z\} \tag{5}$$

One simple way to achieve this is to give each event a unique identifying "event-ID", that lets us find it on disk very quickly, so that once an event has been determined to be of interest, it can be accessed with a single read operation. The most obvious scheme to do this is to give consecutive IDs sequentially to each of the events in each event file, since every event in one file has the same length, and we can therefore easily compute the number of bytes that we should seek over in order to access one particular event of interest. Then, for each of the 2048 channel numbers, we create a *list of event-IDs*, containing the IDs of all events that involve that particular channel number. For example, the one-hundredth such list would then correspond to the set of event-identifiers for all events $\{e_{100}\}$ that involve channel 100. The event-ID of each 18-fold event would be put into 18 such event-ID-lists.

These event-ID-lists are the second list in "list-list-mode", and are stored on disk using the same idea of storing the *difference* between consecutive IDs, rather than the IDs themselves. In this way, we again require about one byte per number, or about 50 - 60 GB total. The average length of the 2048 different event-ID-lists is about 25 MB.

If the average single gate is about 3 channels wide, out of the total of 2048 channels, then on average a single gate accepts about $18 \times 3 \div 2048 = 2.6\%$ of all events. Here the 18 comes from the average event fold, which is the number of chances each event has to meet the gate. A random five-fold gate would be expected to accept about

$$18!/(18-5)! \times (3 \div 2048)^5 = 6.9 \times 10^{-9} \tag{6}$$

of all events. In this example, that corresponds to only about 20 events, for a total of about $(18-5) \times 20 = 260$ counts. Gates on strong coincidences would of course produce many more events; this random-gate estimate corresponds essentially to a "background" gate.

Once these event-ID-lists have been generated, then it is simple to see how to use them to create gated spectra. Consider, for instance, the above example of a five-fold gate that is three channels wide in each of five dimensions. By reading 15 event-ID-lists, we can easily construct five ordered lists of IDs of all events that match the gate on the five different dimensions. By looking for IDs that appear in all five of these lists, we can find those few events that meet the five-fold gate. Then it takes very little time to locate the events in the energy-list files, read them from disk, and create the gated spectrum. When the sum of many five-fold gates is required, it is a simple matter to ensure that each energy in each event is used only once, thus avoiding the problem of spiking.

Sums of gates are usually constructed by taking all combinations of multi-dimensional gates from one or more sets of one-dimensional gates. For example, five-fold gates for a superdeformed band of 12 transitions may be created by taking the ${}^{12}C_5 = 792$ five-fold combinations of the twelve individual gates. In such a case, again assuming that each one-dimensional gate is three channels wide, we can find the interesting events simply by reading a total of 36 different event-ID-lists, and searching for IDs common to at least five of the twelve one-dimensional gates.

The total space required, for this example data set, would be about 120 GB, one-half for the event energy-lists themselves, and the rest for the event-ID-lists for each channel. Even using today's technology, this would be quite easy to do by using fast disk arrays, and/or by spreading the data over about 10 workstations and programming them for parallel processing over a local-area network. Since the time required to construct multi-dimensionally-gated spectra is essentially determined by the time spent seeking and reading from disk, and the event-ID-lists can be made contiguous on disk so that the required number of disk seeks is minimized, access to gated spectra would be very fast. We can estimate that a single five-fold gated spectrum could require as little as 5 seconds to create, and the sum of 792 five-fold gates on a (weak) cascade of twelve transitions would take about 30 seconds total. This time will decrease as the speed and capacity of available disks increases. Within five years from now, this should be technologically trivial, or nearly so.

One potential difficulty, however, has been glossed over in the above discussion, namely that of background subtraction. To perform a complete and accurate background subtraction, all folds of gate from one up to $F_A - 1$ need to be accessed [6]. Clearly, folds 1 through 3 can be dealt with by using histogrammed projections of the data set, but higher folds may require reading a greater subset of the events, which could significantly slow down the construction of gated spectra. The CPU time for calculation of the background could also be significant. Clever, sophisticated background-subtraction algorithms may need to be investigated to determine the full extent of this problem.

6. TEST OF LIST-LIST-MODE

The concept of list-list-mode storage and analysis has been tested with an artificial pseudo-data set, generated by Monte-Carlo methods, and resembling an extremely weak (0.005%) band added to a flat background.

The data had fold F_D distributed between 10 and 20, with a mean at $F_D \approx 14$. Twenty million events were generated. Over 99.995% of the energies had a flat, random distribution over channels 0 to 1023; about 1000 of the events, however, contained a regularly-spaced cascade of up to thirteen transitions. Members of this cascade were "detected" with a probability of 0.7, except that at higher energies the band was cut off over some distribution of "spins". Events containing the band had the same fold distribution as background events, with the remaining energies having the same flat distribution as the background events.

Twenty million events is not a large data set; nevertheless, if these data were to be unpacked into five-folds, they would generate 74 billion quintuples. The events were stored in 15 different files, depending on the event length in bytes, and required 280 MB for the energy-lists and another 280 MB for the event-ID-lists.

The list-list-mode gating scheme was tested with up to 8-fold gates, three channels wide in all dimensions. Background was not subtracted for any of the spectra. Spectra were generated for single F_{A-1}-fold gates, and also by taking all combinations of F_{A-1}-fold gates from a list of 8 one-dimensional gates on different peaks of the band. Examples of the gated spectra produced are shown in Figures 2 and 3.

The time required to construct the gated spectra was measured as a function of analysis fold and the number of summed gates. For low analysis fold, a large fraction of the events met the gating conditions, so that many of the event-list records had to be read from disk, resulting in very long gating times. As the analysis fold was increased, fewer events were accessed, and the time required dropped to about 5 seconds for a single five-fold gate ($F_A = 6$). Above this, for $F_A \geq 7$, the larger number of event-ID-lists that needed to be read dominated the gating time, so that it increased linearly with analysis fold.

The time required for setting gates is, as expected, dominated by disk access times. If the data have been previously accessed and cached in RAM, the gating times decrease greatly. For this reason, in the above timing tests, care was taken to ensure that any previously-accessed data had been flushed from cache memory. Summaries of the time taken and the number of events accessed, as a function of analysis fold, are shown in Figure 4. These tests were performed on a Pentium-II 300 MHz computer, running linux, with the data stored on a local SCSI disk.

6. CONCLUSIONS

High-fold γ-ray coincidence data such as that provided by GAMMASPHERE and EUROGAM often require analysis in four or even five dimensions, so that four-dimensional histograms (hypercubes) can be a powerful tool in extracting as much of the physics information as possible. Software to create and analyse such histograms using standard computer resources available to most physicists has been developed, and the design of the programs and data structures has been briefly presented here. They allow creation of hypercubes of at least 1024 channels per side (corresponding to a 43 gigachannel data space) that will fit onto a few gigabytes of disk space, and extraction of triple-gated spectra in a few seconds.

A new generation of detector arrays, such as GRETA, could produce copious quantities of very-high fold data that would require analysis in 6 or 7 dimensions. Some of the new problems presented by such data have been discussed. The most significant problems have less to do with actual storage of the data than with the time required to generate gated spectra. Two new approaches which attempt to overcome these problems have been presented, namely adaptive list-mode systems and "list-list-mode" storage.

Even with well-organised data structures and very fast gating techniques, the analysis of high-fold coincidence data can be an overwhelming task due to the tremendous quantity of detailed information that they represent. Computer-assistance, in the form of clever programs to keep track of level schemes and help in examining gated spectra, is vitally important. Such programs are now available for 4-fold. By using techniques such as list-list-mode and adaptive list-mode systems, software for higher-fold analysis should not be too difficult to create, even for the very high folds generated by GRETA.

Research at Oak Ridge National Laboratory is sponsored by the U.S. Department of Energy under contract number DE-AC05-96OR22464 with Lockheed Martin Energy Research Corporation.

REFERENCES

1. Radford, D.C., *Nucl. Instr. and Meth. in Phys. Res.* **A361**, 297 (1995).
2. Radford, D.C., *Proceedings of the International Seminar on the Frontier of Nuclear Spectroscopy, Kyoto, October 1992*, ed. Y. Yoshizawa, H. Kusakari and T. Otsuka, World Scientific, 1993, p. 229.
3. Beausang, C.W. *et al., Nucl. Instr. and Meth. in Phys. Res.* **A364**, 560 (1995).
4. See, for example, Flibotte, S. *et al., Nucl. Instr. and Meth. in Phys. Res.* **A320**, 325 (1992).
5. Bentley, J.L., *Commun. ACM* **18**, 509 (1975)
6. Radford, D.C., *Nucl. Instr. and Meth. in Phys. Res.* **A361**, 306 (1995).

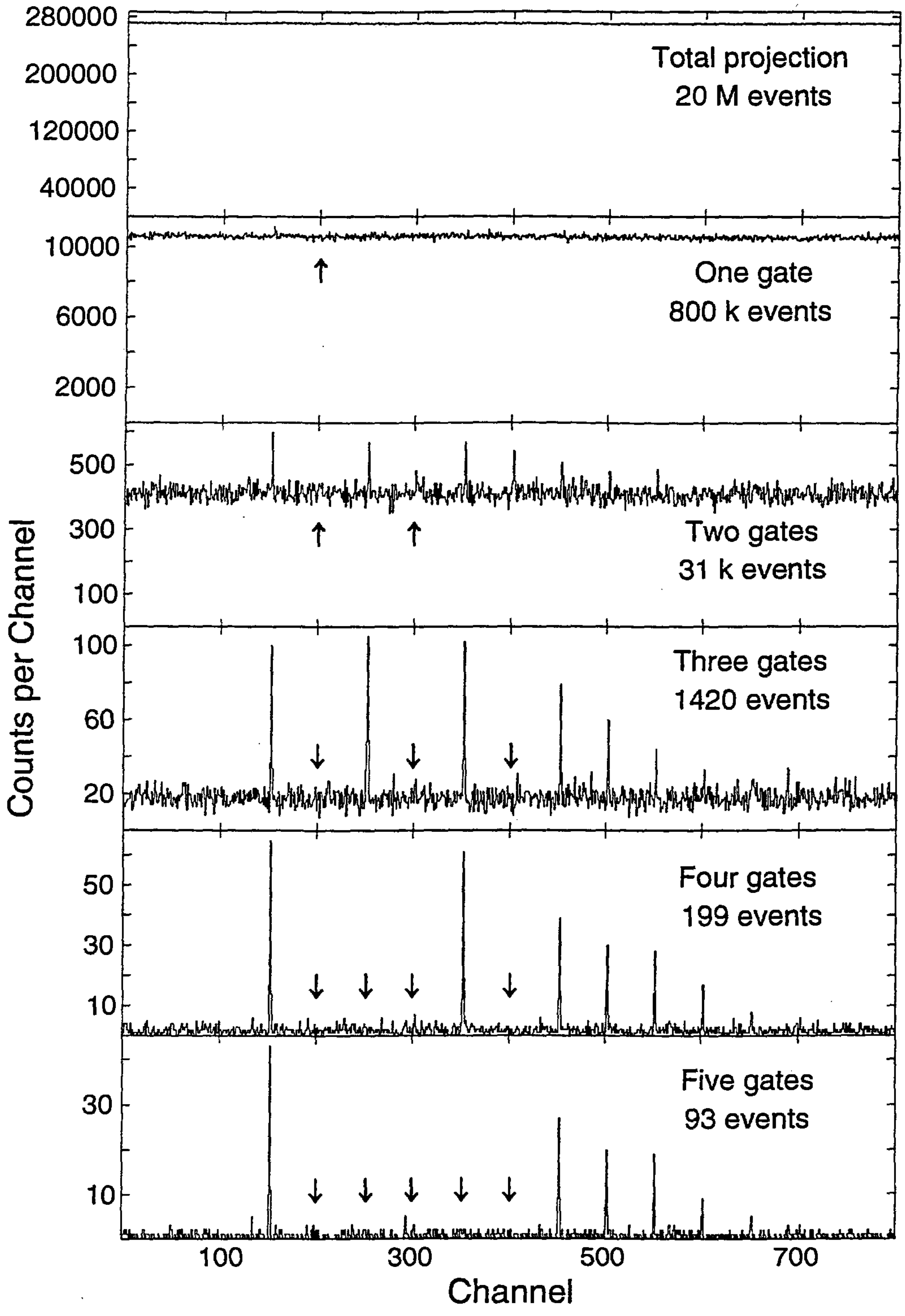

FIGURE 2. Spectra from the set of 2×10^7 events of pseudo-data stored in list-list-mode. Shown are spectra generated by placing a single multi-fold gate, of dimension 1 to 5, on the regularly-spaced "band". The number of events that met the gating conditions, and therefore needed to be read from the energy-lists, is also given for each spectrum. The arrows show the locations of the one-dimensional gates.

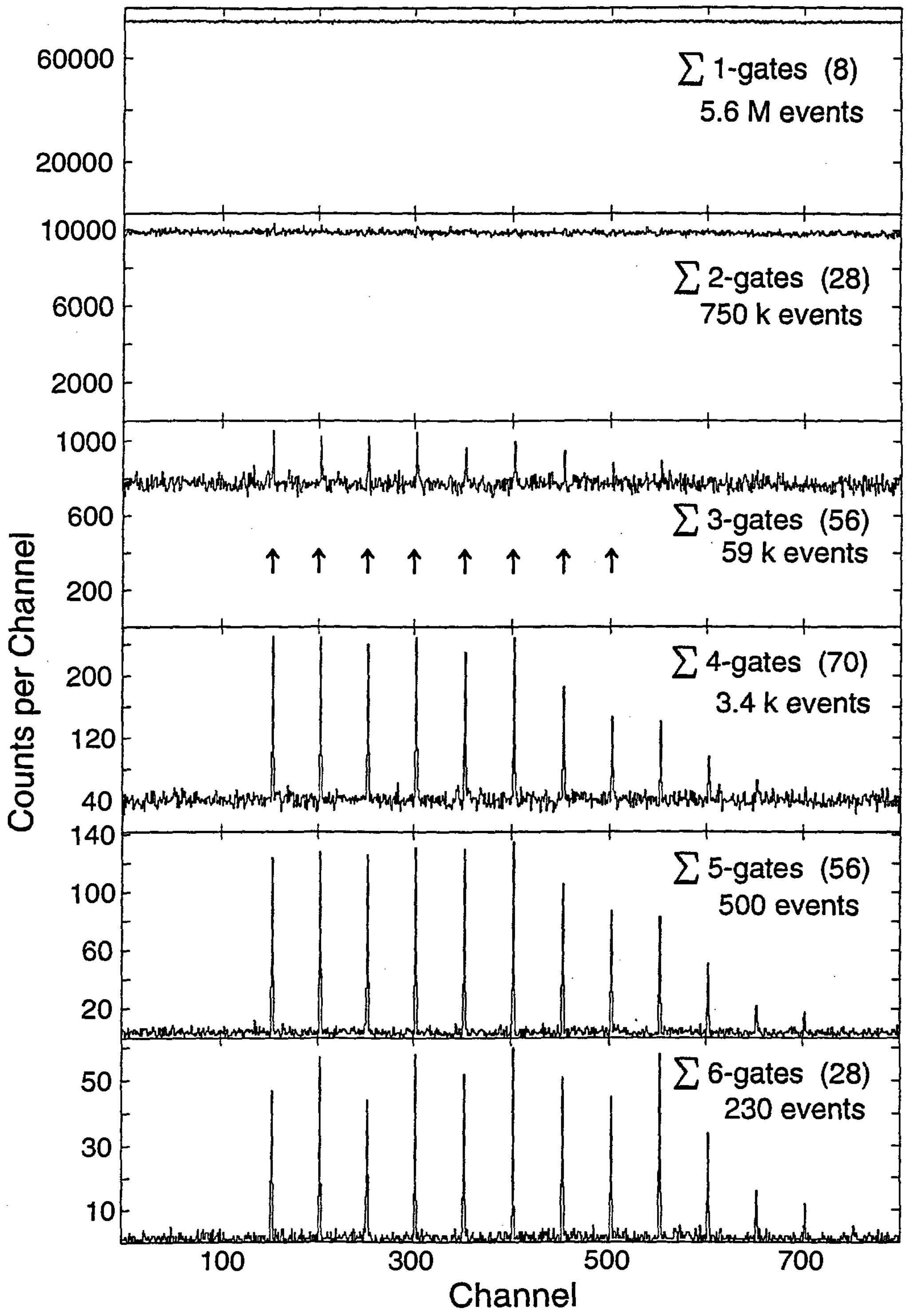

FIGURE 3. Pseudo-data spectra generated by placing all possible combinations of F-fold gates, where F ranges from 1 to 6, from a list of eight one-dimensional gates, on the regularly-spaced "band". The number of events that met the gating conditions, and therefore needed to be read from the energy-lists, is also given for each spectrum, as is the number of F-fold gate combinations, ${}^{8}C_{F}$ (shown in parentheses.) The arrows show the locations of the one-dimensional gates.

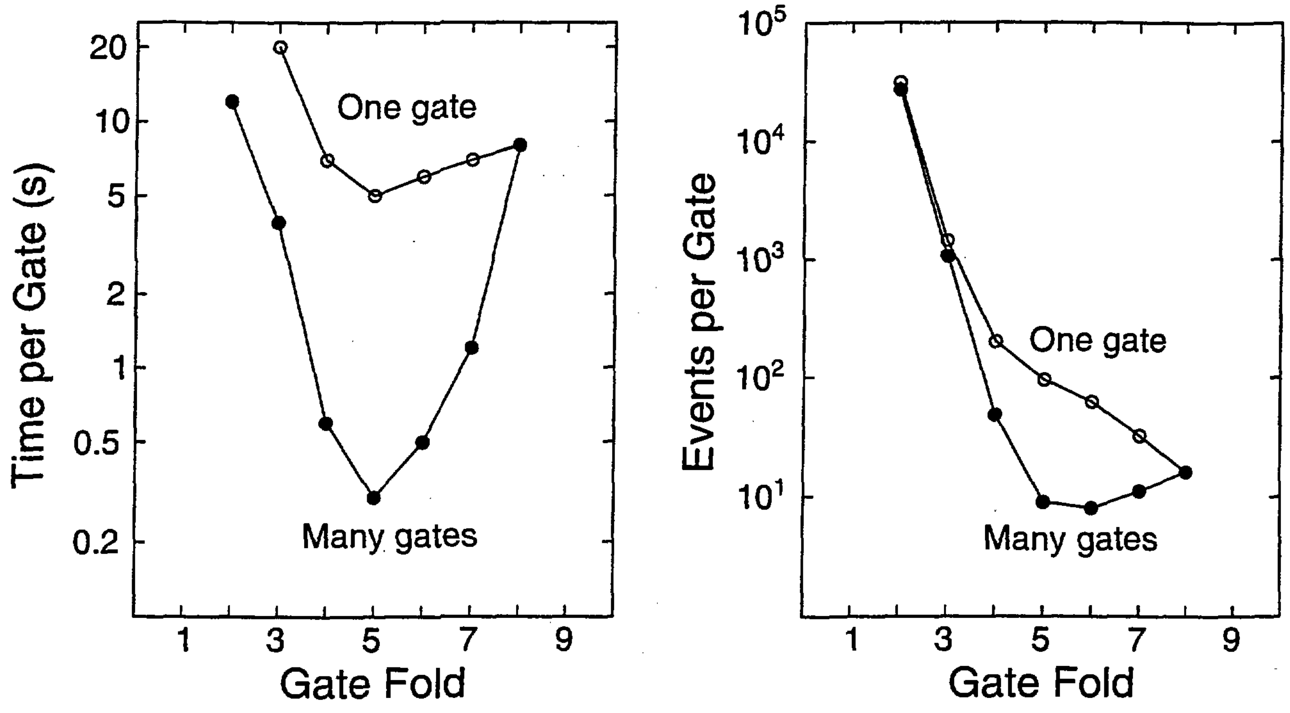

FIGURE 4. Time taken and number of accepted events as a function of gating fold, for tests of list-list-mode storage. Open symbols are for single multi-fold gates, while solid symbols correspond to the case of summing all possible combinations from a list of eight one-dimensional gates.