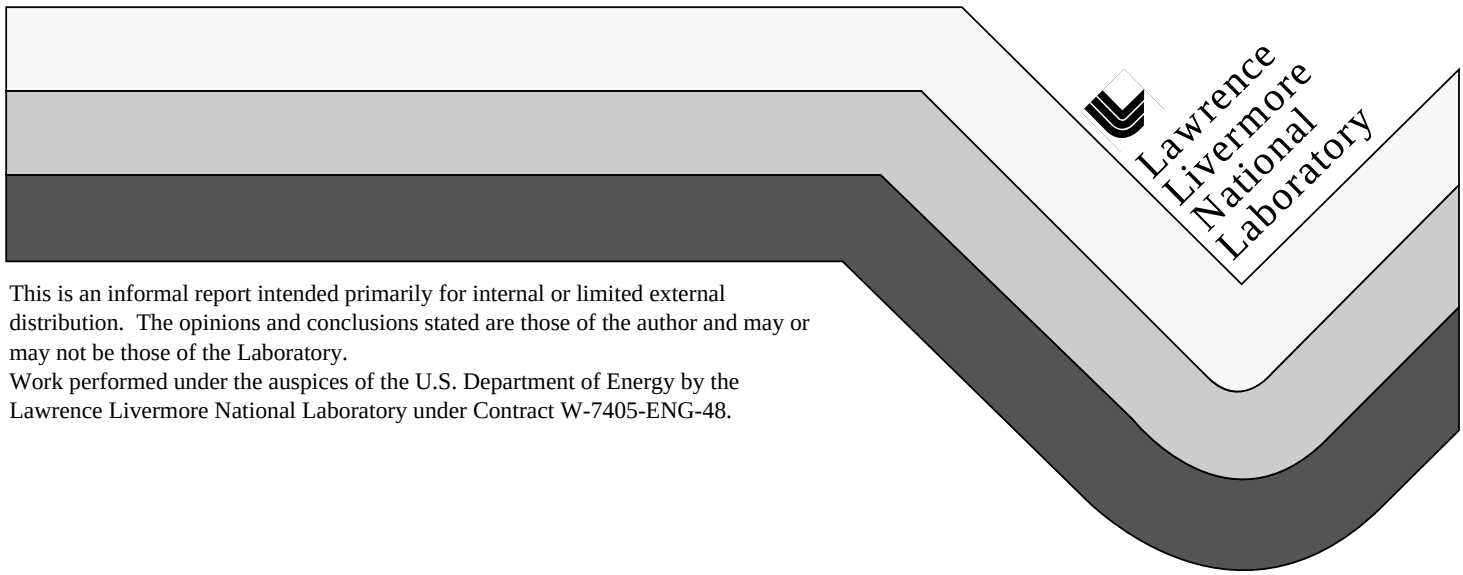
UCRL-ID-127436

Gamma-Ray Identification of Nuclear Weapon Materials

T. B. Gosnell, J. M. Hall, C. L. Ham, D. A. Knapp, Z. M. Koenig,
S. J. Luke, B. A. Pohl, A. Schach von Wittenau, J. K. Wolford

February 3, 1997



Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.

UCRL-ID-127436

February 3, 1997

Gamma-Ray Identification of Nuclear Weapon Materials

Principal Investigator: T. B. Gosnell
Co-Investigators: J.M. Hall, C.L. Ham, D.A. Knapp, Z.M. Koenig, S.J. Luke, B.A. Pohl A. Schach von Wittenau, J.K. Wolford

An FY 1996 LDRD-ER Project
Final Report—96-ERD-062

Contents

Introduction

There has been an accelerating national interest in countering nuclear smuggling. This has caused a corresponding expansion of interest in the use of gamma-ray spectrometers for checkpoint monitoring, nuclear search, and within networks of nuclear and collateral sensors. All of these are fieldable instruments—ranging from large, fixed portal monitors to hand-held and remote monitoring equipment. For operational reasons, detectors with widely varying energy resolution and detection efficiency will be employed. In many instances, such instruments must be sensitive to weak signals, always capable of recognizing the gamma-ray signatures from nuclear weapons materials (NWM), often largely insensitive to spectral alteration by radiation transport through intervening materials, capable of real-time implementation, and able to discriminate against signals from commonly encountered legitimate gamma-ray sources, such as radiopharmaceuticals. Several decades of experience in classified programs have shown that all of these properties are not easily achieved and successful approaches were of limited scope—such as the detection of plutonium only.

This project was originally planned as a two-year LDRD-ER. Since funding for 1997 was not sustained, this is a report of the first year's progress.

The Gamma-Ray Signature Recognition Problem

Gamma-ray signature recognition is straightforward if signals are strong and high-resolution detectors can be used. The high-energy-resolution of these detectors provides unambiguous identification of radionuclides (Fig. 1) using photopeak search algorithms.

However, in many applications the highest resolution detectors, high-purity germanium (HPGe), are not practical because they are bulky, heavy, costly, delicate, and, because of their cryogenic cooling requirement, difficult to support in the field. Newly emerging instruments, using ambient-temperature CZT detectors, have energy resolution inferior to HPGe but are still useful for photopeak identification. Unhappily the current state of the art only produces diminutive CZT crystals—1 cm or less in size. Such detectors require quite strong signals to capture a useful signal from measurements of endurable duration for all but the lowest energy gamma rays. CZT detectors are therefore of greatest use in portable instruments that can be brought in close proximity of an item to be monitored. The considerably greater detection efficiency of HPGe detectors is likely to be best employed in highly sensitive applications where their high cost and cryogenic cooling requirements can be justified.

Monitoring of vehicles at a border crossing is an example where signals may be very weak and highly degraded, due to the contents of a vehicle. A similar example would be the deployment of low-resolution sensors in a Wide Area Tracking System (WATS). In such situations, the signal can be too degraded to produce detectable photopeaks in a HPGe spectrum—even if deployment of such a detector is operationally practical. In these conditions, large-area scintillation detectors have traditionally provided the needed detection efficiency. However, these highly-efficient, but low-energy-resolution detectors introduce their own problem. Because they typically do not produce resolved photopeaks, the spectral data are more challenging to interpret. The most sensitive spectral identification from scintillators is obtained by using the most general approach to gamma-ray signature recognition. That is to analyze the pulse-height spectrum in its entirety. This exploits the entire signal in the case of weak signals in high backgrounds and recognizes the fact that most of the observable counts may lie in the continuum. As traditionally practiced, however, the power of full spectrum analysis is severely degraded if signature alteration occurs because of scattering and absorption by intervening materials. This Lifecycle Plan focuses on a general approach to analyzing such data, while remaining equally useful with higher resolution detectors .

Fig. 2 schematically summarizes the domains of applicability of photopeak analysis and full spectrum analysis for gamma-ray signature recognition as a function of signal strength, detector energy resolution, and signal degradation from scattering and absorption.

A General Approach to Gamma-Ray Signature Recognition

A general gamma-ray signature recognition capability must be able to cope with the following three obstacles that are commonly encountered.

1. Low signal-to-noise ratio. The signals are sparse and background is high—not atypically more than 90% of the total counts recorded.

2. In spite of the fact that we know in detail the gamma-ray emission spectra of the individual *radionuclides* of interest, the exact nature of the gamma-ray signatures from the actual *objects* of interest are unknown. This occurs for three reasons.
   - We don't know the isotopic mixture of the radionuclides in the object.
   - The objects of interest have varied and unknown geometries—leading to geometry-dependent signatures.
   - Various materials will normally lie between the object and the detector, further altering the already geometry-dependent signature.

   Both the object itself and intervening materials cause scattering of radiation, resulting in partial energy degradation of the primary radiation. In the case of intervening materials, already degraded radiation from the object can be further degraded. This scattering, and further scattering within the detector, typically assigns *most detectable events* to a continuum between the photopeaks in the spectrometer's pulse-height spectrum.

3. Signals from commonly encountered natural and man-made gamma-ray sources can compete with signals of interest and cause false alarms.

In spite of these difficulties, NWM spectra still retain unique, but typically subtle, features that can make them recognizable[1].

Full spectrum analysis

The most straightforward approach to full spectram analysis is to cycle through a collection of "templates"—precomputed gamma-ray signatures for nuclear materials expected to be encountered. These templates are compared serially to the observed spectrum. Identification is made by choosing the template that best matches the observed spectrum. This approach works well, even with weak signatures, as long as the signature has not been significantly degraded by scattering and absorption of intervening materials. An example of where such a technique can work quite well is monitoring deplaning passengers at a customs check point. However, in more demanding applications, there are number of drawbacks associated with such an approach.

While other approaches have been investigated, experience has shown that the most useful tool for comparison of full spectra is multiple linear regression. Multiple linear regression minimizes the variance-weighted residual sum of squares (RSS) of a linear combination of pulse-height spectra (such as background and plutonium) presumed to make up the observed spectrum.

1. For completeness, we mention here that in some detection scenarios, particularly nuclear search, background radiation intensity and spectral shape can change very rapidly and complicate the analysis. In passby scenarios, the radiation signature of the object of interest can also vary due to a time-dependent change in source/detector geometry. Such temporal considerations were not within the scope of this LDRD and can be considered independently.

$$RSS = \sum_{i=1}^{N} \left(\frac{y_i - \sum_{j=1}^{M} \beta_j x_{ij}}{\sigma_i} \right)^2 \quad (1)$$

where y_i is the value in the *i*th energy bin of the observed spectrum, and x_{ij} is the value in the *i*th energy bin of the *j*th source spectral component. These components include the spectra from sources expected to be encountered (such as plutonium) and an independently measured background spectrum. The coefficients β_j are free parameters, representing the intensities of the spectral source components and are determined by the minimization process. If the statistical model (the linear combination of component spectra) is an accurate representation of the measured spectrum, then the RSS will follow a χ^2 distribution with $\nu = N - M$ degrees of freedom and the expectation value of the "reduced chi-square" (i.e. χ^2/ν) will be close to unity.

Dealing with unknown spectral signatures

The obvious difficulty of applying multiple linear regression to our signature recognition problem is that we do not know, a priori, the identities of the source spectral components or how they might be altered by radiation transport. These alterations are typically of considerable statistical significance, yielding RSS/ν values in excess of unity. The model is then likely to be rejected by a χ^2 test.

Full spectrum analysis as it is currently practiced. The use of multiple linear regression requires accumulating a collection of radiation signatures for sources likely to be encountered in the field. The most straightforward way to apply this technique is to cycle through the collection, performing the regression with each candidate signature and choosing the one that best matches the observed spectrum. Fig. 2 illustrates such an approach schematically. There are number of drawbacks associated with such an approach.

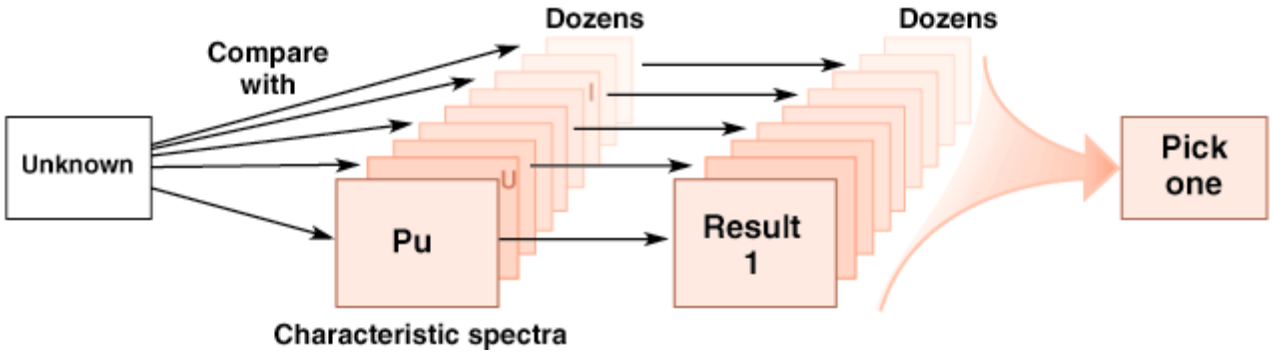


FIGURE 2. An example of full spectrum analysis as currently applied. The observed (unknown) spectrum is compared by multiple linear regression to each of dozens of prospective characteristic signatures, yielding as many results as there are signatures. The"best" result is then picked by some criterion such as minimum value of RSS/ν.

1. *It is inefficient.* Doing multiple fits is time-consuming and taxes the capability of processors suitable for small, portable instruments.
2. *It is rigid and inflexible.* The fits must be done to the signatures as they are. There is no means of adjusting for spectral shape alteration due to scattering.
3. *It is insensitive to weak signals.* Identifying a likely candidate from the signature collection is done not only by the goodness of the match, but by the strength of the signal, such as determined by the value of β_j and its associated uncertainty,.σ_j. The ratio β_j/σ_j defines the signal-to-noise ratio of the candidate signature. To minimize false alarms, β_j/σ_j is typically required to exceed a value of 3.0-5.0 to qualify as a detection. The maximum value of β_j/σ_j occurs when the signature is an exact match to the observed spectrum. Inaccuracies in the signature due to scattering will cause lower values of β_j/σ_j, resulting in false negatives when this value dips below the detection threshold.

A new approach: The single multicomponent signature. The key objective of this project was to identify a small number of spectral components that, taken in linear combination, can form *a single multicomponent signature* that describes the signatures of all NWM and legitimate sources that might be encountered. Once again, we rely on a collection of radiation signatures for sources likely to be encountered in the field. In this case, we increase the size of the collection to include variations on signatures for particular sources in order to sample the effects of likely scattering geometries.

Our approach was to apply principal component analysis to this collection to achieve a linear transformation that produces orthogonal eigenvectors that can reproduce all of the data. Such an analysis is routinely used by statisticians as an automated means to find complex signatures. The principal components have desirable properties that are well-suited for multiple linear regression.

- They are orthogonal.
- They are automatically sorted from the most to the least significant.
- If the original components are carefully chosen then the principal components have physical significance.
- Most of the variation in the collection is explained by the first few principal components.

These few principal components can be used as a single multicomponent signature (i.e. as regressors in multiple linear regression) and related characteristic spectra will cluster together in the principal component space, providing for signature interpolation. Principal component analysis is discussed in more detail in Appendix A.

Our approach to principal component analysis is illustrated schematically in Fig. 3. The feature extraction step in the illustration is a key element in this project. Generally, we will not retain the spectral data in the original pulse-height bins. We will rebin the data adaptively to a smaller number of bins to emphasize structural features and deemphasize the continuum. This is because one of the most significant drawbacks to the use of multiple linear regression is that the value of the goodness-of-fit estimator, RSS, is disproportionately affected by data in the continuum—a highly variable region notoriously difficult to model accurately. Adaptive binning addresses the problem by reducing the

impact of the continuum on the value of RSS. The continuum between photopeaks is treated as a single large channel; the details of the shape of this region are thrown out while the overall amplitude is preserved. This technique eliminates the necessity of *detailed* modeling of the continuum, although the high sensitivity of the amplitude of the continuum to the source geometry remains. Nonetheless, using this technique, observed spectra and theoretical or template spectra can more quickly and effectively be compared.

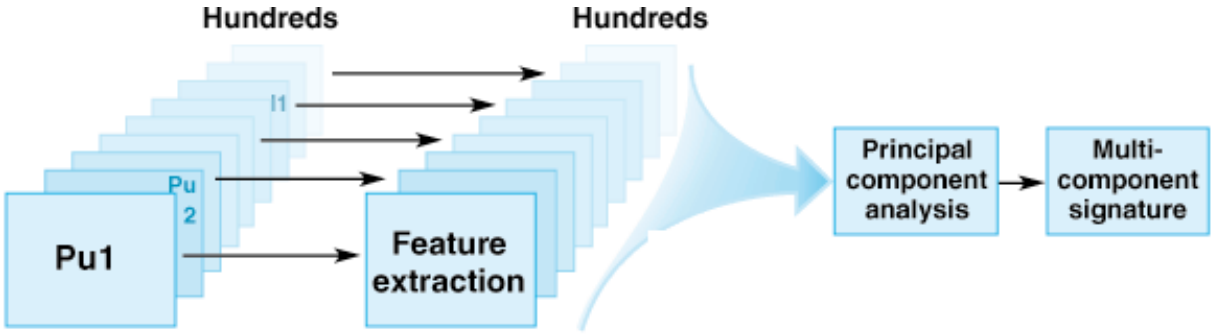


FIGURE 3. Application of principal component analysis to obtain a single multicomponent signature for gamma-ray signature recognition. The features of hundreds of computed characteristic spectra are extracted by adaptive binning prior to the principal component analysis. The analysis yields a single multicomponent signature whose components, in linear combination, can describe spectral shapes of the original characteristic spectra.

Signature recognition of an unknown spectrum is then accomplished simply by a single fit to the multicomponent signature by multiple linear regression as illustrated in Fig. 4.

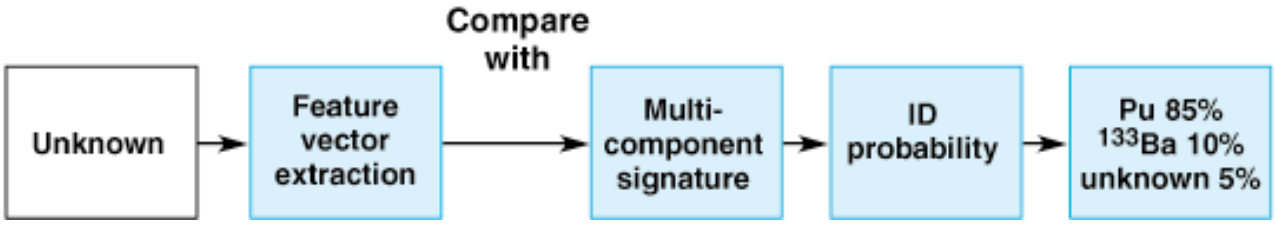


FIGURE 4. Signature recognition with a multicomponent signature. First the spectral features are extracted using the same adaptive binning scheme used for the principal components analysis. The multicomponent signature is compared to the unknown using multiple linear regression. The euclidean distance from the fitted spectrum is used to assign identification probabilities.

We can exploit the fact that the principal component space is in units of standard deviation to assign identification probabilities. The location of the fitted spectrum is projected into the principal components space and its identification probability is determined by its euclidean distance, in standardized units, from its nearest neighbors—clustered in PC space (Fig. 5).

Evidence supporting the concept of using principal components regression for gamma-ray signature recognition. The idea of using principal components regression for gamma-ray signature recognition was pursued briefly a number of years ago by the

principal investigator. However, efforts to exploit these ideas were limited by the cost of experimental measurements of SNM and the cost and immaturity of computational simulations of gamma-ray pulse-height spectra at that time. Exploratory work done using experimentally generated characteristic spectra for individual radionuclides was encouraging and provides more concrete examples of what can be expected. One of the experiments was directed at studying variations in the shape of NaI spectra of ^{60}Co taken through six combinations of scattering and shielding materials. In all, 94 spectra were taken. They are all plotted together in Fig. 6a. The first three principal components accounted for 94% of the variance in the data set and are shown in Fig. 6b.

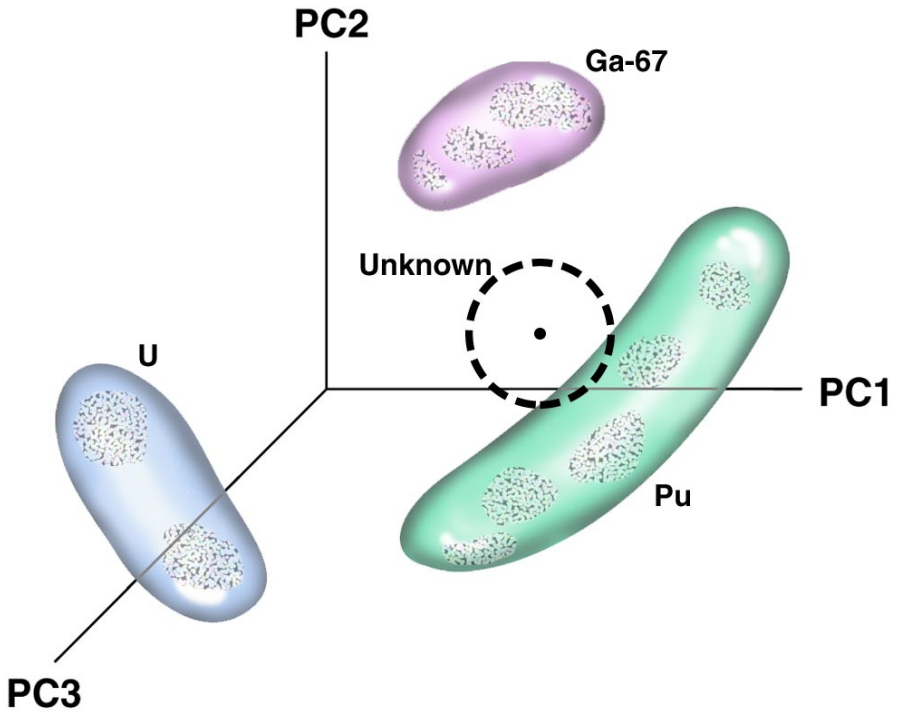


FIGURE 5. Signature identification is accomplished by projecting the unknown spectrum into the principal components space. Its identification probability is determined from its euclidean distance, in standardized units, from its nearest neighbors. In this notional diagram, we shown an unknown spectrum, represented by a black dot. The dashed circle represents a one-standard-deviation uncertainty in its location. This unknown is most likely plutonium, since it overlaps the green domain representing a variety of plutonium spectra.

The projection of 94 spectra into the principal components space is shown in Fig. 7. Clustering of the spectra into regions representing the scattering and absorbing materials produces domains similar to those envisioned in Fig. 5.

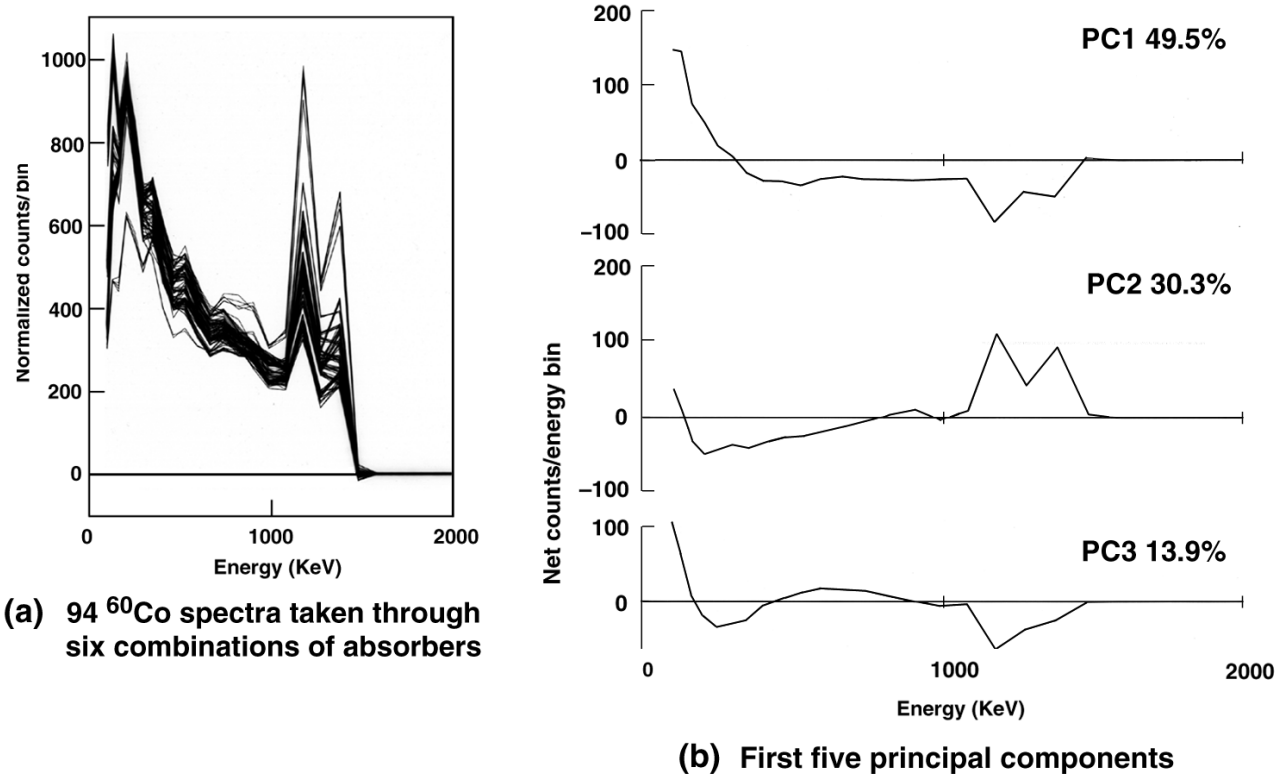


FIGURE 6. Plots of 94 ^{60}Co spectra taken through 6 combinations of absorbers are shown in Fig. 6a. Feature extraction in this case relied only on rebinning the original 256-channel spectra into 32 bins with bin widths increasing in proportion to the energy resolution of the spectrometer. These first three principal components of this data set are shown in Fig. 6b and account for 93.7% of the variance in Fig. 6a.

Project Structure

To produce a multicomponent signature, principal component analysis requires a large set of signature samples. The only practical way to obtain a set of gamma-ray signatures representative of the variety of NWM configurations and industrial and radiopharmaceutical sources is by computer simulation. This occurs for two reasons:

- A pure experimental approach would require a large number of measurements of NWM which has become prohibitively expensive. We have reserved our measurements of NWM to a few items here at LLNL for validation of our computer simulations. Experimental measurements also have the disadvantage of being detector-dependent.
- Simulated radiation signatures have the advantage that the most difficult, time-consuming, and expensive part of the calculations, Monte-Carlo radiation transport, are detector-independent. Detector-dependent response functions can subsequently be folded into the computed flux to obtain the pulse-height distributions for any detector of choice.

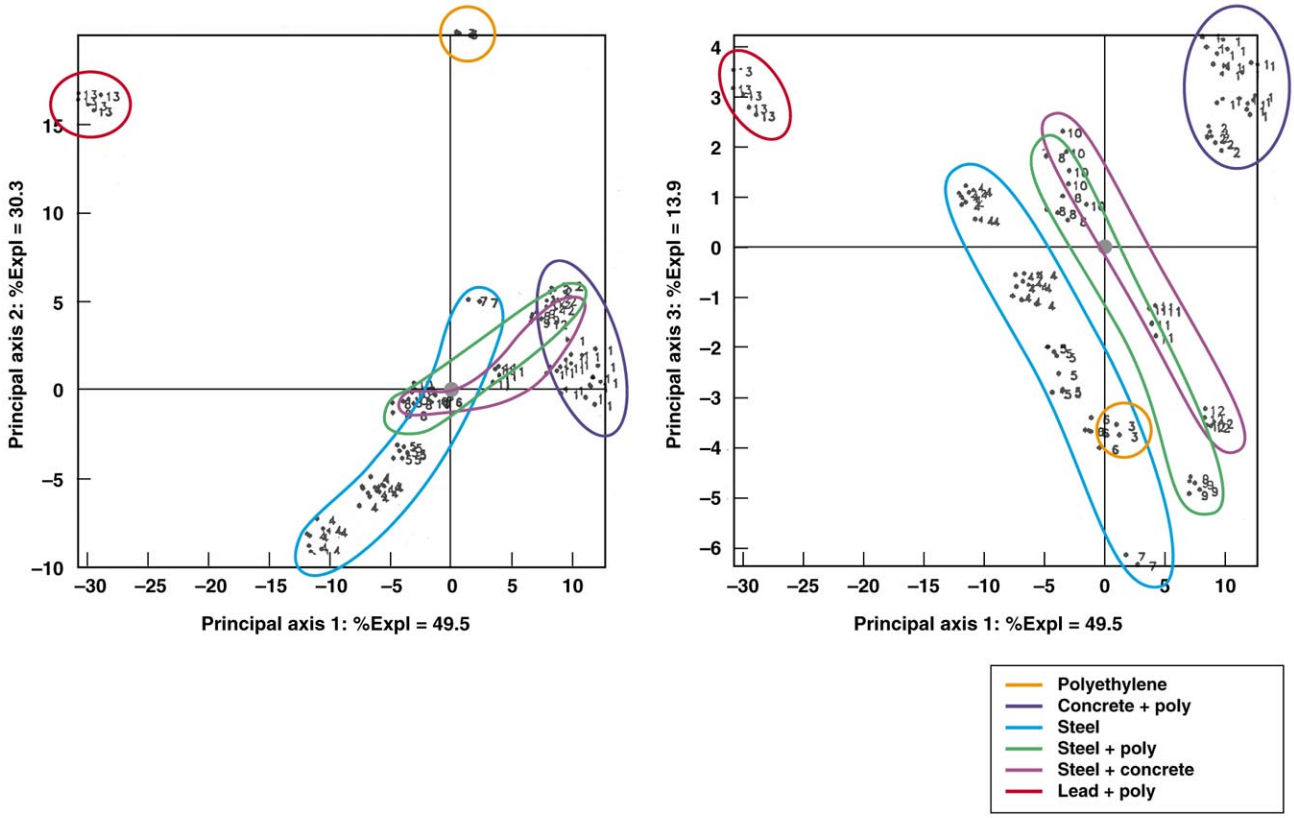


FIGURE 7. Two-dimensional plots showing the 94 ^{60}Co data projected into the space formed by the first three principal components. Clustering of the spectra by the type of intervening material through which the gamma rays passed is clearly evident.

Using an ensemble of computer codes, we have demonstrated our ability to simulate the complex signatures from thick NWM sources to accuracies of 5% or better. This set of codes was satisfactory for producing a few simulations at a time. Producing the signature set for this LDRD required an upgrade of our capability to perform production simulation.

This project proceeded along six lines of effort, with the first five devoted to providing the large data set required for signature recognition algorithm development and a final line of effort for developing and testing the signature recognition algorithm.

1. Develop supporting utility software to enable efficient production calculations
2. Detector calibration/response function determination
3. Validate spectrum synthesis capability
4. Measure representative sample of background variations
5. Production calculations of a representative sample of gamma-ray signatures
6. Develop and test signature recognition algorithms

During FY96 items 1-4 were largely completed. Our first computed radiation signature was produced at the request of former laboratory director John Nuckolls. Considerable thought, planning, and preparation was done on item 6.

Accomplishments in FY 1996

This project was originally intended to span a two-year period since it was realized by the principal investigator that baseline measurements and significant improvements to available software were required to enable implementation of the proposed signature recognition technique. The production calculation of 100's of accurate gamma-ray spectral signatures required an upgrade of our computational capability, in the form of utility software, to transform our disparate codes into a true ensemble. We also ported and tested a multiple linear regression VAX code designed specifically for gamma-ray spectrum analysis. Although the second year's funding was denied, we made substantial progress in several areas to enhance our core capabilities. The most important of these are summarized below.

Develop supporting utility software to enable efficient production calculations.

Radiation Physics Format (RPF)

We developed the RPF to solve a perennial problem of the radiation physicist: how to transfer radiation data, obtained experimentally or computationally in a variety of data structures between a variety of computer codes on a variety of computing platforms. RPF is a set of callable C-language functions based on National Center for Supercomputing Applications' (NCSA) Hierarchical Data Format (HDF)—a platform-independent format designed to handle large, complex sets of scientific data. RPF can easily accommodate data in the wide variety of forms used in this LDRD and other related efforts.

RPF data can originate as instrumental data or computational simulations of experiments from single detectors, detector arrays, and networks of disparate detectors. Total count data can be stored as a time series or as multi-channel scaling. Spectral data can currently be stored as pulse height spectra. Time-stamped list-mode storage for spectral data is planned for a later version. Data can be mixed within a single RPF file. Time series of spectra or gross count data from multiple detectors can be stored within a single RPF file. This would allow, for example, disparate measurements from a series of related experiments or from arrays of detectors from a network. RPF is an important new contribution to our core competency and has broad applicability.

First code was delivered in late FY96 and provides initial RPF functionality. Currently it provides for files containing one data page (header plus data), although implementation in a format conversion utility, can create a two-page RPF file. Full multipage functionality is a minor addition and will be completed in FY97 under separate funding. A format conversion utility, CDF2RPF, was completed to provide conversion of an older file type (CDF) to RPF. This facilitates the ease of use for much of our legacy data. More details on RPF can be found in Appendix B.

Tally Conversion and Formatting (tcf)

Our researchers use two Monte Carlo radiation transport codes, MCNP from LANL and COG from LLNL. These codes produce output in the form of particle flux tallies which the Monte Carlo calculation provides at the positions of simulated radiation detectors. The tallies are energy resolved, and their energy bins are typically structured to accumulate flux with favorable statistics for either spectral line or continuum features, but cannot do so efficiently for both types of features simultaneously. These dissimilar tallies must be combined in a physically correct way prior to further processing steps—such as convolution with a detector response function. We developed *tcf*, a code that automatically parses both MCNP and COG output and combines the line and continuum spectra in a manner that conserves flux. The tool then scales the results to account for problem geometry and source strength and converts to appropriate units.

Tcf produces its output in RPF and other specialty formats. It was implemented to adhere to UNIX programming standards, and has been tested on Hewlett-Packard and Sun workstations. It also runs and gives identical results on the Apple Macintosh.

Future development goals include integration into an abstract execution environment such as KHOROS, refinement of the Macintosh interface, adding a Microsoft windows interface, and more comprehensive screening of the input stream for anomalies. A requirements specification, design description and man page for *tcf* can be found in Appendix C.

Iteratively Reweighted Least Squares (IRLS)

Our final algorithm is intended to initially proceed by fitting of measured spectra to our multicomponent signature by multiple linear regression. We ported (to both Windows-95 and Macintosh platforms) and tested an existing, untested VAX code, written by the principal investigator, especially for gamma-ray spectrum analysis. We will seek other revenue sources for IRLS documentation, publication, and code distribution.

Detector calibration/ response function determination

Detector Response. We made free-field measurements of gamma-ray standards of activity with a variety of high- and low-resolution detectors for use in producing the detector response models needed for signature calculations. Ongoing response model work was curtailed at the end of FY96.

Validate spectrum synthesis capability

NWM Measurements

We made free-field measurements of several nuclear weapon pits of various designs using the same detectors. These spectra were intended for validation of our spectrum synthesis capability and for algorithm testing.

Code Validation

Our researchers use two Monte Carlo radiation transport codes, MCNP from LANL and COG from LLNL. The complex deep penetration radiation signature model prepared for

John Nuckolls was computed using both MCNP and COG and, in spite of their use of different crossection libraries, they produced results that were equivalent within the statistical variance of the calculations (Fig. 8). The remaining spectrum synthesis validation work was curtailed at the end of FY96.

Measure representative sample of background variations

We instrumented a van with both high- and low-resolution gamma-ray detectors and took large number of gamma-ray spectra along the sea shore (low background), at LLNL (medium background), and in downtown Oakland (high/variable background). Details of the instrumentation and data acquisition are described in Appendix D.

Production calculations of a representative sample of gamma-ray signatures

Our first computed radiation signature simulated a specific nuclear smuggling scenario at the request of former laboratory director John Nuckolls. We modeled a 1 Kg source of HEU and demonstrated the effects on the spectrum from a fairly thin lead shield. For comparison to background radiation, we approximated background with a mixture of potassium, thorium, and uranium in soil (Fig. 9).

Overview of project status

The structure and status of this LDRD-ER at the end of FY 1996 is shown in the flow chart in Fig. 10. On scenarios, we decided to look at attempts to smuggle a nuclear weapon pit. We performed calculations to simulate the transport of gamma-rays from a 1 Kg HEU source and demonstrated that the COG and MCNP radiation transport codes produce equivalent results. The data filter is the *tcf* utility described above. Work on a Radiation Physics Format file type is also described above. We measured gamma-ray standards of activity for with a number of detectors of varying efficiency and energy resolution in order to compute their response functions. The detector response function calculations were not completed in FY 1996, as indicated in the figure. As, described above, we did instrument a van and acquire background data for the same detectors.

The work showing equivalency of COG and MCNP results provided partial validation of our simulation capability, as MCNP has been extensively validated for this purpose in the past. For further validation, we acquired test data from three types of nuclear weapon pits. Evaluation of these data were deferred to FY 1997. As noted above, our first characteristic spectrum was the HEU spectrum calculated for former laboratory director John Nuckolls. Production calculations of characteristic signatures was to begin early in FY 1997. Since a fair number of results from the production calculations needed to be available as grist for algorithm development, feature vector extraction, principal components analysis, multiple linear regression, and identification probability work was also to be done in FY 1997.

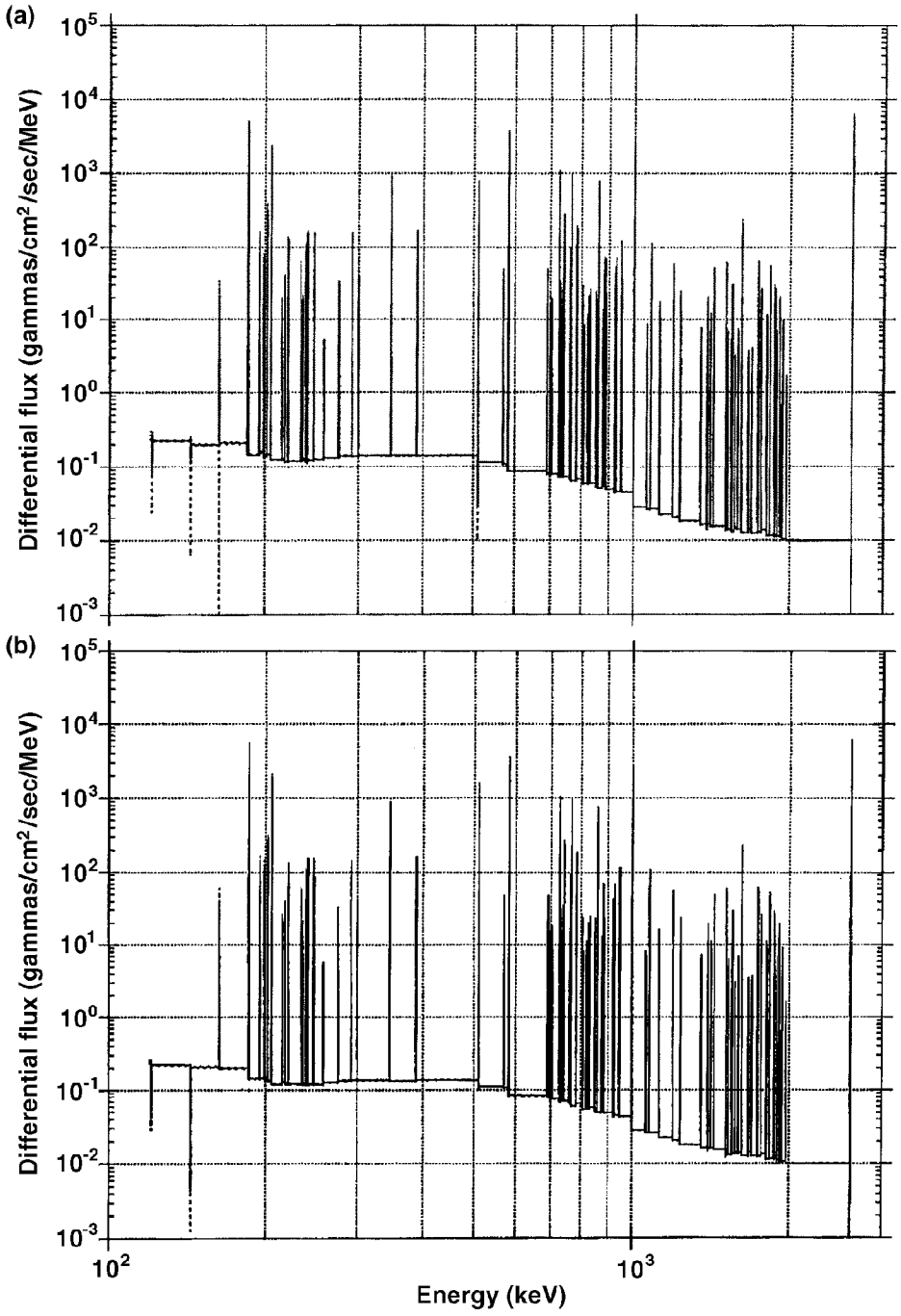


FIGURE 8. The results for transport of gamma rays from shielded HEU from the COG code (a) are equivalent to those for MCNP (b) within the statistical variance of the Monte Carlo process. Notable differences on the plots at low energy reflect the high variance for transport of these weakly penetrating gamma rays.

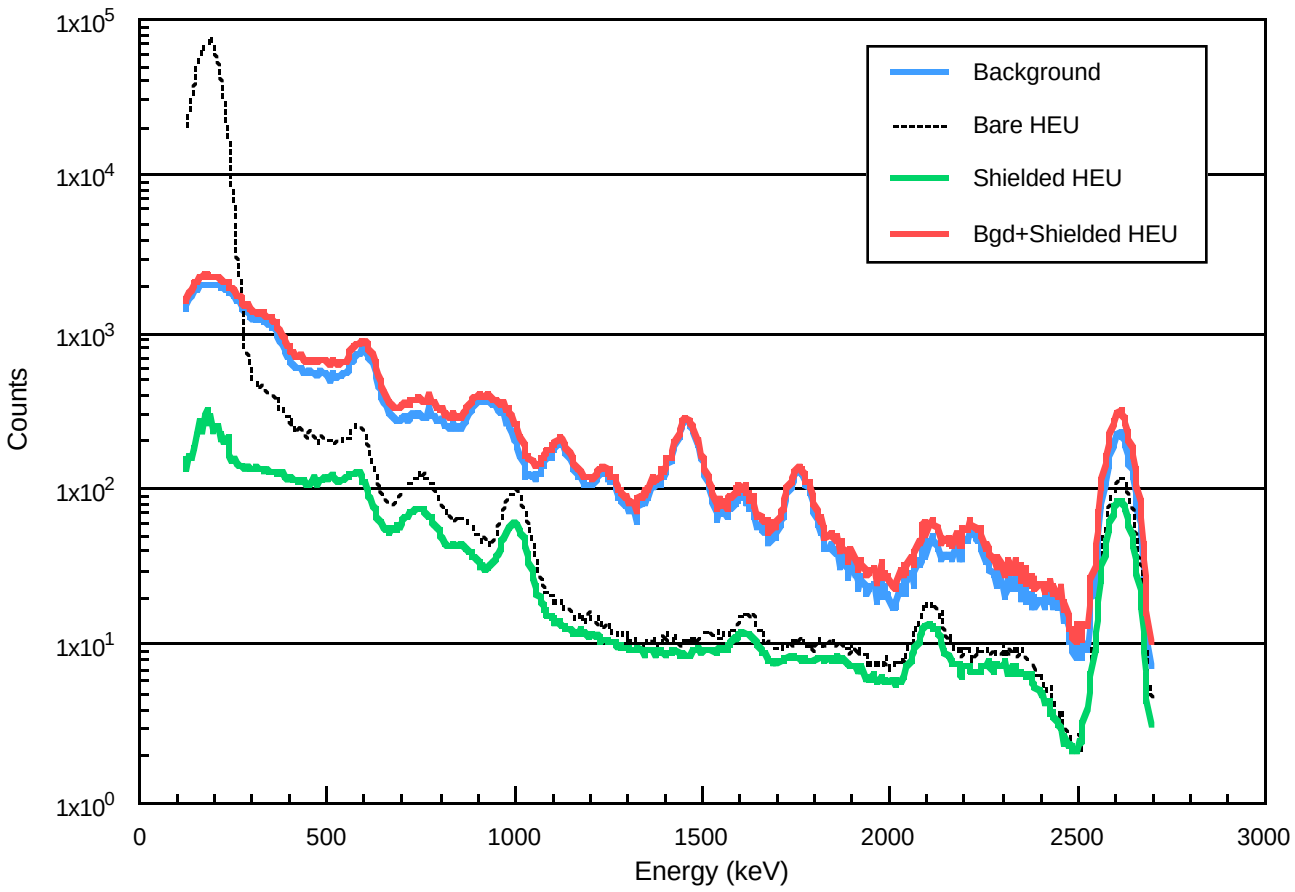


FIGURE 9. Computer synthesized sodium iodide gamma-ray pulse-height spectra from HEU and background radiation. The main spectral feature in the bare HEU spectrum is a photopeak from ^{235}U at 185 keV. When a lead shield is placed between the HEU and the detector (the green spectrum), this peak is greatly attenuated and the principal contribution to the spectrum is due to the isotopic impurity ^{232}U. This uranium isotope decays rapidly through a succession of daughters and enters the thorium decay series. The final radioactive nuclide in this series is ^{208}Tl which contributes the characteristic peak at 2615 kev, its Compton continuum, and escape peaks. Background radiation, shown in blue also has contributions from the thorium series.

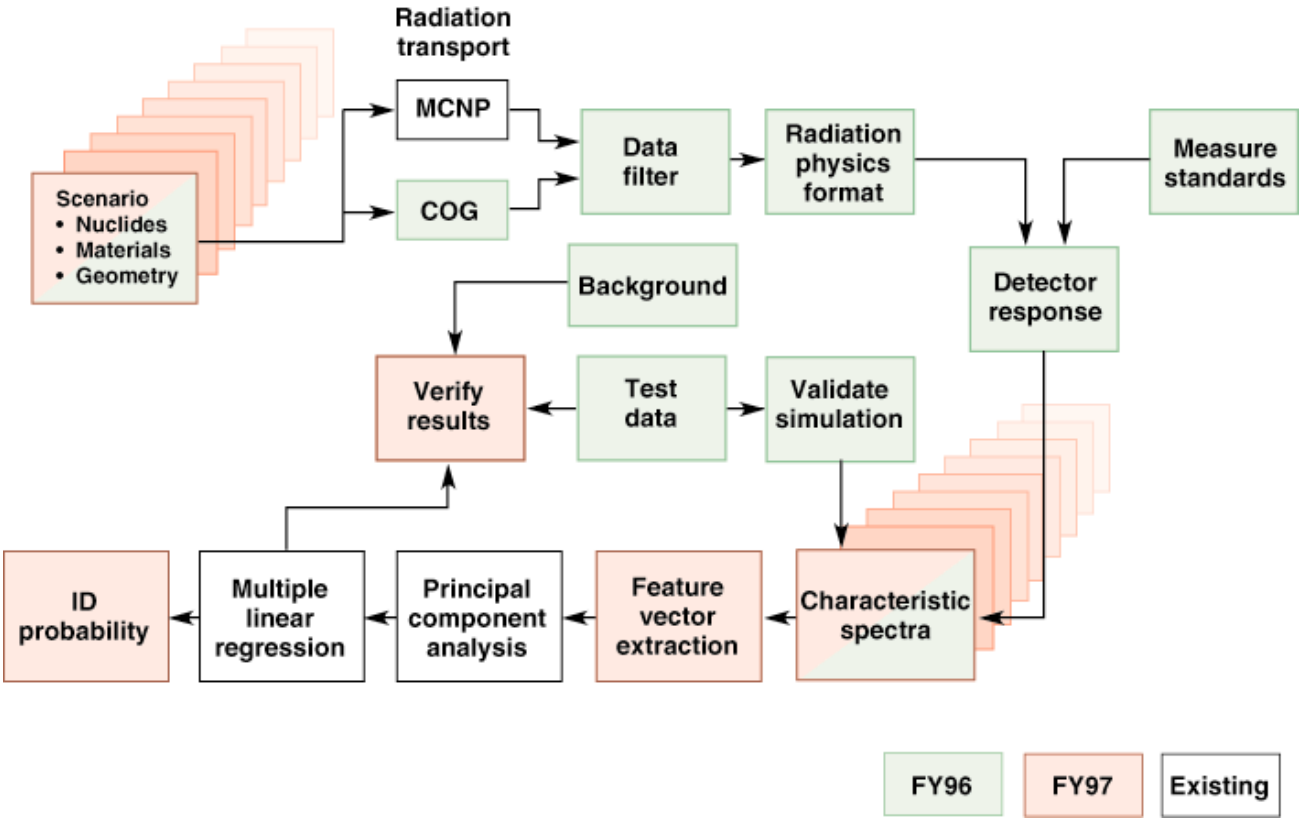


FIGURE 10. Flowchart of the Gamma-Ray Identification of Nuclear Weapon Materials LDRD-ER project. The white boxes indicate existing capabilities. The green boxes indicate capabilities completed in FY 1996. Work planned for FY 1997 is in the peach-colored boxes. The COG computer code is obviously an existing resource. Its green green color indicates work completed that verified its ability to produce gamma-ray transport results equivalent to MCNP.

Appendix A—Principal Component Analysis

The purpose of principal component analysis is to determine independent factors (i.e. principal components) of a multivariate data set in such a way so as to explain as much of the total variation in the data as possible with as few of these factors as possible.

Consider a data set of variables, X_1, X_2, …, X_p. The *i*th principal component of this data set is

$$PC(i) = \sum_{j=1}^{p} w_{ij}X = w_{i1}X_1 + w_{i2}X_2 + \ldots + w_{ip}X_p$$

where the *w*'s are weights to be estimated from the data and the *X*'s are the original variables, such as counts in energy bins, generally expressed in standardized form[1]. This linear relationship clearly describes a coordinate rotation.

The total variation in the data set is given by the sum of the sample variation of the *k* samples:

$$totalvariation = S_1^2 + S_2^2 + \ldots + S_p^2$$

where S_j^2 is the sample variance of X_j, *j*=1,2,…,*p*. The total variation is then a measure of "uncertainty" associated with the observations on all *p* variables. If, as is usually the case, the variables are in standardized form (so that $S_j^2 = 1$ for every *j*), the total variation is simply equal to *p*, the number of variables.

The first principal component, $PC(1)$, is that weighted linear combination of the variables which accounts for the largest amount of the total variation in the data. Notionally, one can imagine a hyperellipsoid that describes the sample variance about the mean values of the *X*'s. The first principal component is the result of a $S_j^2 = 1$ coordinate rotation that establishes a new axis that lies along the principal axis of the ellipsoid. That is, $PC(1)$ is that linear combination of the *X*'s, say

$$PC(1) = w_{(1)1}X_1 + w_{(1)2}X_2 + \ldots + w_{(1)p}X_p$$

1. For principal component analysis, the variables are usually recast in standardized form—so that $S_j^2 = 1$ for every j and the total variation is simply equal to *p*, the number of variables.

where the weights $w_{(1)1}, w_{(1)2}, \ldots, w_{(1)p}$ have been chosen so as to maximize the quantity

$$\frac{varianceofPC(1)}{totalvariation}$$

In other words, no other linear combination of the *X*'s will have as large a variance as $PC(1)$.

[When the *X*'s are in standardized form, the proportion of the total variation in the data accounted for by $PC(1)$ is

$$\frac{varianceofPC(1)}{p}$$

Also, the weights are chosen subject to the restriction $\sum_{j=1}^{p} w_{(1)j}^2 = 1$ in order that the variance of $PC(1)$ will not exceed the total variation.]

The second principal component, $PC(2)$, is that weighted linear combination of the variables which is uncorrelated with $PC(1)$ and which accounts for the maximum amount of the remaining total variation not already accounted for by $PC(1)$. It can be thought of as that direction obtained by a second rotation about $PC(1)$ that aligns along the first minor axis of the variance hyperellipsoid.

In general, the *i*th principal component $PC(i)$ is that linear combination

$$PC(i) = w_{(i)1}X_1 + w_{(i)2}X_2 + \ldots + w_{(i)p}X_p$$

which has the largest variance of all linear combinations which are uncorrelated with all of the previously determined $j - 1$ principal components. It is possible to determine as many principal components as there are original variables. However, in most practical problems, most of the total variation in the data is usually accounted for by the first few components. Furthermore, the analytic goals of parsimony and independence are typically achieved with this method. In other words, a linear combination of the first few principal components can be used to describe the entire data set with considerable accuracy and their linear independence makes them ideal as regressors in multiple linear regression.

Algorithms for extracting the principal components are described extensively in the literature. However, in practice, the analyst usually relies on one of the many excellent commercial multivariate data analysis software packages to perform principal component analysis.

Appendix B—Radiation Physics Format

Requirements

This file format must accommodate neutron (count) data, gamma ray spectral (energy, flux pairs) data, and statistical data. This data can be either collected from or calculated for any number of detectors as either a single count/spectrum or a time series of counts/spectra.

In addition, the data files must also accommodate varying types and amounts of information describing the origin, history, and contents of the data. It is desired that this 'header' information be readable by the user.

These data files must also be flexible/extensible, and portable across various platforms in order to avoid the problem of converting vast quantities of legacy data when modifying/adding information to the files, or changing platforms.

The implementation of these data files will be based upon ANSI C using as few system-dependent extensions as possible in order to promote cross platform compatibility. Current platforms under consideration are UNIX (HP, Sun), PC (DOS and Windows), and Mac (Mac-OS).

Conceptual Solution

A spectral file format that we developed a number of years ago, the CDF file, provided a good starting point for the solution of our problem. A CDF file is an ASCII file that has a variable length and content header consisting of keyword labels followed by value (e.g. 'nchan=4096') delimited by the keywords '::beginheader::' and '::endheader::' followed by the spectral data. The spectral data could consist of up to five columns of information. More than one spectrum could be sequentially stored in one CDF file.

The advantages of this solution are that the header is variable length, variable content, and easy to read, write, and modify by an application and by the user with any text editor. This also leads to the disadvantages of the introduction of non-standard keywords and that the user can easily corrupt the file. Another major disadvantage is the amount of storage space that the ASCII data can consume.

An existing, widely-used, noncommercial basis for the data files was sought to provide the advantages of not having to develop low level routines from scratch, low cost availability, a potentially large user base from which to draw utilities and expertise, and not being tied to a commercial product and the potential fickleness and longevity of any particular company.

The Hierarchical Data Format (HDF) package available from National Center for Supercomputing Applications (NCSA) provides a good basis for the general solution to our problem. (The HDF WWW home page is found at 'http://hdf.ncsa.uiuc.edu/'. The NCSA anonymous ftp server can be accessed across the Web at 'ftp://ftp.ncsa.uiuc.edu'

or directly over the Internet at ‘ftp.ncsa.uiuc.edu’.) Our solution was to Implement an extended CDF file using the HDF paradigm to meet our requirements.

Specifications for the Actual Solution

The latest incarnation of our spectral data file format solution was developed on a Hewlett Packard 9000 model 735 running HP-UX version 9.01 and implemented using ANSI C standard code within the HDF version 4.0r2 framework. It is a binary a variation on the CDF file solution in which there is a variable length, variable content header describing the HDF file content, and one or more sets of data with a variable length, variable content header describing the data.

The headers can be thought of as metadata since they are data about data and are implemented as HDF vgroups. The data file header consists of a variable number of vdatas that describe the data. The vdata’s name can be thought of as its’ description or variable name. The vdata’s class is its’ category. And the vdata’s data is its’ value. For example, one header vdata entry might have a name of “location”, a class designation of "experiment_identification," and a data value of “measurement lab.”

Data will be represented by default as 32-bit integers or 64-bit floating point numbers depending upon their requirements unless specified otherwise by the user. Floating point numbers will automatically be converted from the native format of the host machine to the standard HDF format of IEEE 32- or 64-bit floating point format as required. This is to promote interplatform portability, a feature which is hindered by storing data using the ‘native format’ option. Neutron data are stored as scientific data set (SDS) arrays of rank 3 to allow fields for time, neutron count, and detector. Gamma spectra are also stored as SDS arrays of rank 4 to allow fields for time, energy, flux, and detector. Calculated gamma spectra can have additional rank for relative error, absolute error, if desired.

RPF Header Viewer and Editor

The use of the HDF model forces a compromise by the user who wants the advantages of an ASCII header. An HDF data file is binary. Tests performed on an annotated HDF data file show that the ASCII strings of the annotations are “visible” and comprehensible with listing utilities such as UNIX more and with text editors such as emacs. Unfortunately, although the annotations are in order, they are spread throughout the HDF data file rather than being contiguous in any one place in the file. If you want to “hand edit” the string, you can. However, if you substitute another string with a different length, the whole HDF data file becomes unreadable.

The solution to this problem is to create utilities for viewing, extracting, and replacing the header information. The viewing utility will simply open the HDF data file, extract the header information, and display it. The utility for extracting the header will write the header information to an ASCII file. This file can be “hand edited” by the user with his/her favorite text editor. The utility for replacing the header information will take this or any other “valid”ASCII header file plus an HDF data file and replace the header information of that HDF data file with that found in the ASCII header file. This will provide the user with the flexibility desired and in a more timely manner while minimizing demands on the implementer.

None of these utilities have yet been implemented. Current thinking is that, when funding is secured, these utilities will be implemented in the highly portable language, Java.

Keywords

There will be a standard set of keyword labels with specific definitions. This list will evolve with time and will be made available in documented form. Some header information updates in existing HDF data files may be required depending upon the scope and nature of the evolution. This uniformity of labels and usage is essential for the successful integration of utilities and applications.

Keywords

Description

It is imperative for the portability of data between our applications to have a standard set of keywords in the header information section(s) of our HDF files. This file contains the data dictionary for the keywords that will be used to describe the fields in our HDF files. The initial list of keywords was developed by a committee made up of the authors of this report.

In order for the HDF spectral file header format and its associated software to be flexible and yet maintain their extensibility with minimal backtracking and reworking, the definitions of the header keywords and their uses must be strictly controlled. It is expected that the list of header keywords will evolve with use over time. Procedures to add, modify, delete, and exclude header keywords are described below. These procedures, although draconian in appearance, are necessary to maintain the integrity and utility of the RPF format. Maintaining a tight rein on the keywords from the beginning will lessen the chaos that keyword changes could cause in the future.

Remember, not all keywords will appear in each HDF header. Only those keywords relevant to each particular header will be included in the HDF file.

Adding/modifying keywords

Changes and additions may be made to the accepted keyword list according to the following 6-step procedure.

1. Submit written request for data dictionary modification to the person in charge of the keyword data dictionary. Initially and currently the keyword czar will be the HDF file designer, Cheryl Ham. For modifications of existing keywords, this request should include the a list of the current keyword(s) to be modified, a list of the corresponding keyword(s) that the requester would like to be used, and a justification for each of the requested changes. To incorporate new keywords, the request should include a list of the new keyword(s) to be added, a complete definition of each of the new keywords, and justification for each of the requested new keywords.
2. The keyword czar will submit the written request to the appropriate subset of the local community of HDF file users for feedback. They will have seven calendar days to provide oral and/or written feedback. Initially and currently, the members of the keyword review committee are the individuals that developed the initial keyword list.

3. The keyword czar will consider the initial request and all the feedback, resolve conflicts, and render a ruling upon the request—plus generate the required update to the HDF spectral file accepted keyword data dictionary.
4. A detailed written ruling on the request to add/modify keywords will be disseminated to the local HDF file user community. This will include information on the original request, the feedback received, and the ruling rendered, plus any other relevant information such as the potential keyword data dictionary entry. The local user community will have seven calendar days to submit a written appeal to the keyword czar.
5. They keyword czar will resolve any appeals by repeating the described process as required.
6. They keyword czar will make a detailed entry into the keyword data dictionary logbook and update the keyword data dictionary as required.

Deleting/excluding keywords

There will be occasions that certain keywords and the information they represent will be deleted or excluded from the accepted list of keywords. The following list describes the procedure to delete or exclude a keyword from the accepted header keyword list.

1. Submit written request for data dictionary modification to the person in charge of the keyword data dictionary. Initially and currently the keyword czar will be the HDF file designer, Cheri Ham. This request should include the a list of the current keyword(s) to be deleted or excluded and a justification for each of the requested changes.
2. The keyword czar will submit the written request to the appropriate subset of the local community of HDF file users for feedback. They will have seven calendar days to provide oral and/or written feedback.
3. The keyword czar will consider the initial request and all the feedback, resolve any and all conflicts, and render a ruling upon the request, plus generate the required update to the HDF spectral file rejected keyword data dictionary.
4. A detailed written ruling on the request to delete/exclude keywords will be disseminated to the local HDF file user community. This will include information on the original request, the feedback received, and the ruling rendered, plus any other relevant information. The local user community will have 7 calendar days to submit a written appeal to the keyword czar.
5. They keyword czar will resolve any appeals by repeating the described process as required.
6. They keyword czar will make a detailed entry into the keyword data dictionary logbook and update the keyword data dictionary as required.

Modifying the keyword change procedure

It its allowed that the procedures for changing the header keyword list might require modification at some point in time.

1. Submit written request for procedure modification to the person in charge of the keyword data dictionary. Initially and currently the keyword czar will be the HDF file

designer, Cheryl Ham. The requester should include a description of and a justification for each of the requested changes, additions, and/or deletions.

2. The keyword czar will consider the request, and if deemed necessary will submit the written request to the appropriate subset of the local community of HDF file users for feedback. They will have seven calendar days to provide oral and/or written feedback.
3. The keyword czar will consider the initial request and all the feedback, resolve any and all conflicts, and render a ruling upon the request, plus generate the required update to the Modifying the keyword change procedure.
4. A detailed written ruling on the request to add/modify keywords will be disseminated to the local HDF file user community. This will include information on the original request, the feedback received, and the ruling rendered, plus any other relevant information. The local user community will have 7 calendar days to submit a written appeal to the keyword czar.
5. They keyword czar will resolve any appeals by repeating the described process as required.
6. They keyword czar will make a detailed entry into the keyword data dictionary logbook and update the keyword change procedure as required.

HDF spectral file accepted keyword data dictionary

This data dictionary contains a list of the accepted HDF spectral file keywords, their defined data type and a detailed description of the information that they contain. Dates will be stored as an ANSI standard formatted string ‘mm-dd-yyyy’. Times will be stored as an ANSI standard formatted string ‘hh:mm:ss XXX’. This list is ordered to group keywords by functionality as opposed to alphabetically.

File Creation Keywords:

The following three keywords preserve information on the initial creation of the HDF spectral data file.

date_file_written char *

This ASCII string contains the original date that this file was written stored in ANSI standard format.

time_file_written char *

This ASCII string contains the original time that this file was written stored in ANSI standard format.

original_filename char *

This ASCII string contains the original file name used when this file was written.

Experiment Identification Keywords:

The following nine keywords contain basic information to identify the experiment which resulted in the HDF file.

location char *
This ASCII string contains a description of the location where the experiment was performed.

latitude char *
This ASCII string contains the latitude of the location where the experiment was performed.

longitude char *
This ASCII string contains the longitude of the location where the experiment was performed.

experimenters_names char *
This ASCII string contains the names of the people involved in conducting the experiment.

experiment_id char *
This ASCII string contains the experiment identification label.

run_number int32
This is the run number which resulted in the data attached to this header.

number_of_bins int32
This contains the first dimension of the data block associated with this header. This would be equivalent to the number of channels for a gamma-ray spectrum.

number_of_units int32
This contains the second dimension of the data block associated with this header. This would be equivalent to the number of spectra in a time series of gamma-ray spectrum.

number_of_boxes int32
This contains the third dimension of the data block associated with this header. This would be equivalent to the number of experiments stored in the HDF file.

Hardware Identification Keywords:

These nine keywords identify the experimental hardware and their parameters. The use of high_voltage, superfine_gain, fine_gain, coarse_gain, and shaping_time are dependent upon the type of instrument used.

instrument_type char *
This ASCII string describes the type of instrument used in the experiment.

instrument_id char *
This ASCII string contains the instrument serial number or other identification.

high_voltage double
This describes the high voltage adjustment used.

superfine_gain double
This describes the superfine gain adjustment used.

fine_gain double
This describes the fine gain adjustment used.

coarse_gain double
This describes the coarse gain adjustment used.

shaping_time double
This describes the shaping time used.

detector_type char *
This ASCII string describes the type of detector used in the experiment.

detector_id char *
This ASCII string contains the detector serial number or other identification.

Experiment Time Keywords:

The following five keywords contain timing and location information for the experiment.

count_start_date char *
This ASCII string contains the date when the experiment's data collection was started stored in ANSI standard format.

count_start_time char *
This ASCII string contains the time when the experiment's data collection was started stored in ANSI standard format.

count_live_time double
This contains the live time of the experiment measured in decimal seconds.

count_dead_time double
This contains the dead time of the experiment measured in decimal seconds.

count_clock_time double
This contains the elapsed clock time of the experiment measured in decimal seconds. It is equal to the count_dead_time plus the count_dead_time.

Energy Calibration Keywords:

These thirteen keywords describe various energy calibration parameters.

energy_calibration_type char *
This describes the type of energy calibration used. Valid options are none, normal, polynomial, and binned. None means that no calibration is available. Normal uses the keywords intercept, slope, and quadratic to calculate the energy calibration via the formula energy value = [quadratic * bin_number * bin_number] + [slope * bin_number] + [offset]. Polynomial uses the number_of_energy_coeffs coefficients stored in energy_calibration_coeffs to calculate the energy calibration using a higher order polynomial equation. The intercept is stored in energy_calibration_coeffs[0]. The slope is stored in energy_calibration_coeffs[1]. The quadratic term is stored in energy_calibration_coeffs[2], and so on. The spectrum option stores the number_of_energy_bin_edges bin edges in the energy_calibration_spectrum array.

intercept double
This is the intercept for the normal mode of energy calibration.

intercept_error double This is the intercept error for the normal mode of energy calibration.

slope double
This is the slope for the normal mode of energy calibration.

slope_error double
This is the slope error for the normal mode of energy calibration.

quad double
This is the quadratic factor for the normal mode of energy calibration.

quad_error double
This is the error in the quadratic factor for the normal mode of energy calibration.

number_of_energy_coefs int32
This is the number of energy coefficients for the polynomial calibration option.

energy_calibration_coef double *
This contains the array of actual coefficients for the polynomial calibration option. The intercept is stored in energy_calibration_coef[0]. The slope is stored in energy_calibration_coef[1]. The quadratic term is stored in energy_calibration_coef[2], and so on.

energy_calibration_coef_error double *
This array contains the error in the corresponding coefficients for the polynomial calibration option.

number_of_energy_bin_edges int32
This is the number of energy bin edges required for the energy calibration. This value should equal [number_of_bins + 1].

energy_calibration_spectrum double *
This contains the array of energy bin edges for the data associated with this header. energy_calibration_spectrum[0] contains the leftmost bin edge. The remaining [number_of_energy_bin_edges - 1] bin edges are the energies of the right bin edges.

energy_calibration_filename char *
This ASCII string contains the name of the file from which energy calibration information was obtained for the data associated with this header.

Efficiency Calibration Keywords:

These seven keywords describe various peak efficiency calibration parameters.

efficiency_calibration_type char *
This describes the type of efficiency calibration used. Valid options are none, gunnink, polynomial, and spectrum. None means that no calibration is available. Parameter uses the number_of_efficiency_coeffs parameters stored in efficiency_calibration_params to store the efficiency calibration information. The spectrum option stores the efficiency calibration information in the efficiency_calibration_spectrum array.

number_of_efficiency_coefs int32
This is the number of efficiency coefficients required for the efficiency calibration.

efficiency_calibration_coef double *
This array contains the actual coefficients for the parameterized calibration option.

efficiency_calibration_coef_error double *
This array contains the error in the corresponding coefficients for the parameterized calibration option.

number_of_efficiency_bin_edges int32
This is the number of energy bin edges required for the energy calibration. This value should equal [number_of_bins + 1].

efficiency_calibration_spectrum double *
This array contains the bin edges for the efficiency calibration spectrum.

efficiency_calibration_filename char *
This ASCII string contains the name of the file from which efficiency calibration information was obtained for the data associated with this header.

Geometry Keywords:

These nine keywords describe the geometry of the experiment.

source_detector_distance double
This is the closest approach [a.k.a. fact to face] distance from the source to the detector in meters. It is assumed to be the closest approach [a.k.a. face to face] distance unless the geometry_description field to describes otherwise.

source_detector_distance_error double
This is the error in the source detector distance.

geometry_description char *
This ASCII string describes relevant experimental geometry information.

percent_solid_angle double
This is the percent solid angle subtended by the detector as it is viewing the source.

geometric_correction double
This is the geometric correction.

number_of_wall_materials int32
This is the number of wall materials described.

wall_material char *
This array of strings contains the description of the wall material.

wall_thickness double *
This is the array of wall thickness values.

thickness_error double *
This is the array of errors associated with the wall thickness value.

Source Description Keywords:

These seven keywords describe the source used in the experiment.

source_description char *
This ASCII string describes the source used in the experiment.

source_id char *
This is the serial number or other identifier for the source used in the experiment.

source_material char *
This ASCII string describes the source material.

declared_enrichment double
This is the declared enrichment value.

declared_enrichment_error double
This is the error associated with the declared enrichment value.

measured_enrichment double
This is the measured enrichment value.

measured_enrichment_error double
This is the error associated with the measured enrichment value.

Absorber Description Keywords:

These three keywords describe the absorber(s) used in the experiment.

number_of_absorbers int32
This is the number of absorber layers which are described in the absorber data block.

absorber_material char*
This array of strings describes the absorber material.

absorber_thickness double *
This array contains the thickness of the absorber material layers.

Collimator Description Keywords:

These three keywords describe the collimator(s) used in the experiment.

number_of_collimators int32
This is the number of collimators which are described in the collimator data block.

collimator_type char *
This array of strings describes the types of collimator used. Currently acceptable collimator types include none, cylindrical, and complex.

collimator_description char *
This array of strings describes the collimator(s) used.

Shield Description Keywords:

These three keywords describe the shield(s) used in the experiment.

number_of_shields int32
This is the number of shields which are described in the shield data block.

shield_type char *
This array of strings describes the types of shields used. Currently acceptable shield types include none, cylindrical, and complex.

shield_description char *
This array of strings describes the shield(s) used.

Comment Keywords:

This keyword field allows for user comments.

comments char *
This is an ASCII string that can include newlines, tabs, etc., in which the user may make additional comments.

THDF spectral file rejected keyword data dictionary

There will be occasions that certain keywords and the information they represent will be excluded from the accepted list of keywords. This data dictionary contains a list of the rejected HDF spectral file keywords, their description of the information, and why they were rejected.

count_end_time
It was decided by consensus by the attendees at the LDRD meeting on 12/15/96 (who also happened to be the initial members of the keyword review committee) that this field was not necessary since the information that it would have contained can be calculated from other header fields.

Appendix C—tcf requirements specification, design description, and man page

tcf Software Requirements Specification—Version 2.2

1.0 Introduction

1.1Purpose

The purpose of this document is to present in a precise and easily understood manner, all system requirements deemed necessary for the tally configuration and formatting (*tcf*) system, which replaces the SELECTOR/GADCDF code series. Upon review, this document shall become a baseline defining a complete set of high-level system requirements for the system developer.

Each requirement is described in a manner that may be tested by a prescribed method. Requirements apply at the system level, i.e. they may be assigned to software, hardware, and/or operators.

1.2 Scope

The *tcf* system is an intermediate processor between the MCNP1 and COG2 Monte Carlo particle transport programs and the data library (RPF format). *tcf* will also produce output in a format readable by the GADRAS3 program. The functions of *tcf* are limited to:

1. Importing photon flux tallies from MCNP or COG.
2. Combining (taking the union of) the tally energy bin structures of two tallies.
3. Scaling the tallies by some constant factor to account for geometric and source strength parameters in the physical problem.
4. Formatting the output so that it may be read by the GADRAS code, or placed into a data repository in HDF format for future use in GADRAS or other tools.

1.3 Definitions and Acronyms

1.3.1 Acronyms

ANSI	American National Standards Institute
COG	(*not* an acronym)
GADRAS	Gamma Detector Response and Analysis Software
HDF	Heirarchical Data Format (NCSA)
LANL	Los Alamos National Laboratory
MCNP	Monte Carlo Neutron and Photon transport code (LANL)
NCSA	National Center for Supercomputer Applications
RPF	Radiation Physics Format4

1.3.2 Definitions

Tally

Quantity representing a particle flux and statistical variance in an interval of energy or bin, as output by a particle transport code. Also somtimes used to refer to a list of such quantities.

Variance

When applied to the results of Monte Carlo particle transport calculations represents the spread in values of the particle tally, or more precisely the variance σ^2 is given by $\sigma^2 = \langle N^2 \rangle - \langle N \rangle^2$

1.4 References

1. Briesmeister, J., *MCNP—A General Monte Carlo Code for Neutron and Photon Transport*, Los Alamos National Laboratory, LA-7396-M, Rev. 2+addenda, (1986, 1988, 1991).
2. Buck, R. et al., *COG Users Manual, Lawrence Livermore National Laboratory,* UCID M-221-1, July 1994.
3. Mitchell, Dean J., *GADRAS-95 User's Manual*, Sandia National Laboratories, Systems Research Center 5900, August 9, 1995.
4. Ham, Cheryl L., *Radiation Physics Format Tools Reference Manual*, Lawrence Livermore National Laboratory, August 29, 1996

2.0 General Description

The *tcf* system supports the Gamma Ray Signature Recognition Project. It is one of a group of programs that convert files containing scientific data into and out of Radiation Physics Format (RPF), a dialect of the NCSA Heirarchical Data Format (HDF). In addition to this conversion, *tcf* scales photon flux tallies and combines them in a physically correct manner. And in addition to RPF output, *tcf* also creates output in a format directly readable by the GADRAS system.

2.1Project Perspective

The *tcf* system is a standalone tool for the creation of composite photon tallies given the output of a Monte Carlo photon transport problem. It will render output in convenient formats for other programs to use as input.

2.2 Software Functions

2.2.1 Importing – the *tcf* system accepts input in the form of the output files from an MCNP or COG execution.

2.2.2 Purifying – the *tcf* system corrects the Monte Carlo continuum tally, removing the flux due to lines contained in the line tally. It similarly removes the continuum between line bins from the line tally. It thereby produces “pure” line and continuum components.

2.2.3 Scaling – the *tcf* system allows constant scaling of tallies to account for changes in solid angle, source strength and active detector area.

2.2.4 Combining – the *tcf* system combines MCNP or COG line and continuum type tallies into a single tally which is a function of the union of the two energy domains.

2.2.5 Converting – the *tcf* system converts units for use by the GADRAS code system.

2.2.6 Formatting – the *tcf* system formats the results of the combining and scaling operations in a form compatible with one of:

1. The legacy GADCDF program
2. An HDF/RPF data repository, or
3. The GADRAS program directly

2.3 User Characteristics

The users of the *tcf* system will in general be domain experts, with experience in judging the overall accuracy of the system's output.

3.0 Specific Requirements

Specific requirements are listed in this section; they are:

1. Functional Requirements
2. Performance Requirements
3. Design Constraints
4. Attributes
5. External Interface Requirements

3.1Functional Requirements

Fun100

The *tcf* system shall have the ability to accept photon flux tally input from the output of calculations done with the MCNP Monte Carlo transport program.

Fun110

The *tcf* system shall have the ability to accept photon flux tally input from the output of calculations done with the COG Monte Carlo transport program.

Fun 120

The *tcf* system shall have the ability to produce output in the Radiation Physics Format (RPF) dialect of the NCSA Heirarchical Data Format (HDF)

Fun 130

The tcf system shall have the ability to produce output in a format which is compatible with the GADRAS detector modelling program

Fun 140

The *tcf* system shall have the ability to produce output in a format which will facilitate photopeak analysis.

Fun 150

The *tcf* system shall permit the scaling of tallies to reflect a change in photon source strength.

Fun 160

The *tcf* system shall permit the scaling of tallies to reflect a change in detector solid angle.

Fun 170

The *tcf* system shall permit the scaling of tallies to reflect a change in active detector area.

Fun 180

The *tcf* system shall accept input data in the same format used by the previous system.

3.2 Performance Requirements

Perf 100

The *tcf* system shall be capable of processing tallies at a rate such that the total process execution time exceeds by no more than a factor of 2.5 times the time required for data input and output alone.

3.3Design Constraints

Alg 100

The *tcf* system shall include an algorithm for combining tally energy bin structures which has been derived from a reverse engineering analysis of the previous system, i.e. the GADCDF program.

Std 100

Source code for the *tcf* system shall conform to the ANSI Standard for the C programming language.

Port 100

The *tcf* system shall compile on any system which supports the ANSI Standard C Language

Port 110

The *tcf* system shall compile, link and execute on any system which supports the POSIX 1003 standard for operating system and object libraries.

3.4 Attributes

Attr 100

The *tcf* system shall be used to process classified data up to and including the Secret Restricted Data level.

3.5 External Interface Requirements

ExtInt 100

The *tcf* system shall allow execution from a UNIX™ system command line prompt.

ExtInt 110

The *tcf* system shall allow execution from a simulated UNIX™ system command line prompt under the Apple Macintosh™ operating system.

ExtInt 120

The *tcf* system shall permit future execution from within the Khoros 2.0 graphical programming environment.

tcf Software Design Description—Version 1.5

1.0 Introduction

1.1 Purpose

The purpose of this document is to provide a description of the software structure, interfaces and data for the *tcf* (tally combining and formatting) system, which replaces the SELECTOR/GADCDF code series. The short length and scope of this document is justified by the limited size and complexity of this system. This document is not intended to satisfy the full requirements of the IEEE 1016-1987 Software Design Description Standard.

1.2 Scope

The *tcf* system is an intermediate processor between the MCNP[1] and COG[2] Monte Carlo particle transport programs and the data library (RPF format). *tcf* will also produce output in a format readable by the GADRAS[3] and CDF2RDF[4] programs. The functions of *tcf* are limited t

1. Importing photon flux tallies from MCNP or COG.
2. Scaling the tallies by some constant factor to account for geometric and source strength parameters in the physical problem.
3. Optionally providing output in a format equivalent to the VAX SELECTOR code.
4. Combining (taking the union of) the tally energy bin structures of two tallies.
5. Converting tally to proper units for use by the GADRAS code.
6. Optionally writing the tallies into an RPF library archive using an agreed upon HDF/RPF file structure.
7. Formatting the output so that it may be read directly by GADRAS, or placed into a data repository for future use in GADRAS or other tools.

1.3 Definitions and Acronyms

1.3.1 Acronyms

ANSI	American National Standards Institute
COG	(not an acronym)
GADRAS	Gamma Detector Response and Analysis Software
HDF	Heirarchical Data Format (NCSA)
LANL	Los Alamos National Laboratory
MCNP	Monte Carlo Neutron and Photon transport code (LANL)
NCSA	National Center for Supercomputer Applications
RPF	Radiation Physics Format[5]

1.3.2 Definitions

Tally
Quantity representing a particle flux and statistical variance in an interval of energy or bin, as output by a particle transport code. Also sometimes used to refer to a list of such quantities.

Variance
When applied to the results of Monte Carlo particle transport calculations represents the spread in values of the particle tally, or more precisely the variance σ^2 is given by $\sigma^2 = \langle N^2 \rangle - \langle N \rangle^2$

1.4 References

1. Briesmeister, J., MCNP—*A General Monte Carlo Code for Neutron and Photon Transport*, Los Alamos National Laboratory, LA-7396-M, Rev. 2+addenda, (1986, 1988, 1991).
2. Buck, R. et al., *COG Users Manual, Lawrence Livermore National Laboratory,* UCID M-221-1, July 1994.
3. Mitchell, Dean J., *GADRAS-95 User's Manual, Sandia National Laboratories,* Systems Research Center 5900, August 9, 1995.
4. Ham, Cheryl L., *RPF Spectral File Keyword Data Dictionary,* Lawrence Livermore National Laboratory, January 12, 1996
5. Ham, Cheryl L., *Radiation Physics Format Tools Reference Manual,* Lawrence Livermore National Laboratory, August 29, 1996

2.0 General Description

The *tcf* system supports the Gamma Ray Signature Recognition Project. It is one of a group of programs that convert files containing scientific data into and out of Radiation Physics Format (RPF), a dialect of the NCSA Heirarchical Data Format (HDF). In addition to this conversion, *tcf* scales photon flux tallies and combines them in a physically correct manner. Besides RPF output, *tcf* also creates output in a format directly readable by the GADRAS system.

2.1 Project Perspective

The *tcf* system is a stand-alone tool for the creation of composite photon tallies given the output of a Monte Carlo photon transport problem. It will render output in convenient formats for other programs to use as input.

2.2 Software Functions

2.2.1 Importing - the *tcf* system accepts input in the form of the output files from an MCNP or COG execution.

2.2.2 Purifying - the *tcf* system corrects the Monte Carlo continuum tally, removing the flux due to lines contained in the line tally. It similarly removes the continuum between line bins from the line tally. It thereby produces “pure” line and continuum components.

2.2.3 Scaling - the *tcf* system allows constant scaling of tallies to account for changes in solid angle, source strength and active detector area.

2.2.4 Combining - the *tcf* system combines MCNP or COG line and continuum type tallies into a single tally which is a function of the union of the two energy domains.

2.2.5 Converting - the *tcf* system converts units for use by the GADRAS code system.

2.2.6 Formatting - the *tcf* system formats the results of the combining and scaling operations in a form compatible with one of:

1. The legacy GADCDF program
2. An HDF/RPF data repository, or
3. The GADRAS program directly

2.3 User Characteristics

The users of the *tcf* system will in general be domain experts, with experience in judging the overall accuracy of the system's output.

3.0 Design Description

This section describes the decomposition of the *tcf* system into specific processes and data stores. Each process description includes input and output, as well as the operations which transform the former into the latter. The data flow diagrams in Appendix A graphically describe the relationships between the individual processes and data entities.

3.1 Top Level Process Decomposition

The top level of the process decomposition, which identifies the external interfaces to *tcf* is described below as single process involving two sources of input, which by default are placed onto the standard input stream, and one stream of output (which likewise defaults to the standard output stream.) This same information is represented graphically in Appendix A.

Input

Table C-1. *tcf* input

File	Name	Description
MCNP tally output	MPNP's text output file OUTx—typically the last file in dump sequence—is used	Tally information for photon fluxes into simulated detectors
COG tally output	COG's text output file x.out	

Processing

CONSTRUCT line tally data structure
CONSTRUCT continuum tally data structure

READ processing options from execute line

IF running in verbose mode THEN
    WHILE stream not empty
        READ tally from stream
        ASSIGN tally to be “line-like”, “continuum-like”, or neither (ignore)
    END WHILE
    READ multipliers from console
ELSE (terse mode assumed)
    READ first tally into “line-like” tally structure
    READ second tally into “continuum-like” tally structure
END IF

PREPROCESS tallies:
    REMOVE continuum tally flux from “line-like” tally => line tally
    REMOVE line tally flux from “continuum-like” tally => continuum tally

IF “selector” formatted output desired THEN
    PRINT header information
    PRINT line tally
    PRINT continuum tally
    EXIT
END IF

CONSTRUCT composite tally data structure

COMBINE line and continuum tallies into composite
    CONSTRUCT primitive line tally data structure
    CONSTRUCT primitive continuum tally data structure
    MERGE line and continuum tally energies into line tally primitive
    COPY line tally primitive energies into continuum tally primitive and composite tally
    REBIN the line tally into the line tally primitive
    REBIN the continuum tally into the continuum tally primitive
    ADD the line and continuum tally fluxes to find the composite tally flux

SCALE tallies using multiplier(s)

CONVERT tally to GADRAS units

```
IF RPF formatted output desired THEN
    OPEN RPF file
    WRITE RPF header information
    WRITE line tally
    WRITE continuum tally
    WRITE composite tally
    CLOSE RPF file
END IF

PRINT composite tally onto output stream in GADRAS format

END
```

Output

Table C-2. *tcf* output

File	Name	Description
1. Composite tally data	stdout	Combined, scaled and converted line an continuum tallies from MCNP calculation in GADRAS format
2. Header, line tally, and continuum tally	stdout	Preprocessed line and continuum tallies in SELECTOR format
3. Header, line tally, continuum tally, and composite tally data	Disk file	Same as 1, except also includes composite tally. All tallies output in RPF format

3.2 Data Decomposition

The *tcf* data are contained in several data structures which are described in the tables below. All structures are defined in ANSI C.

3.2.1 Header Data Structure

```
#define PROB_ID_LENGTH      3
#define LINE_LENGTH         80

struct headerspec
{
    char probID[PROB_ID_LENGTH][LINE_LENGTH];
    double range ;
    double offset ;
    double gammaYieldMult ;
    double solidAngleMult ;
    double timeMult ;
    double faceAreaMult ;
    double compositeMult ;
    char timeStamp[80] ;
} ;
```

3.2.2 Tally Data Structure

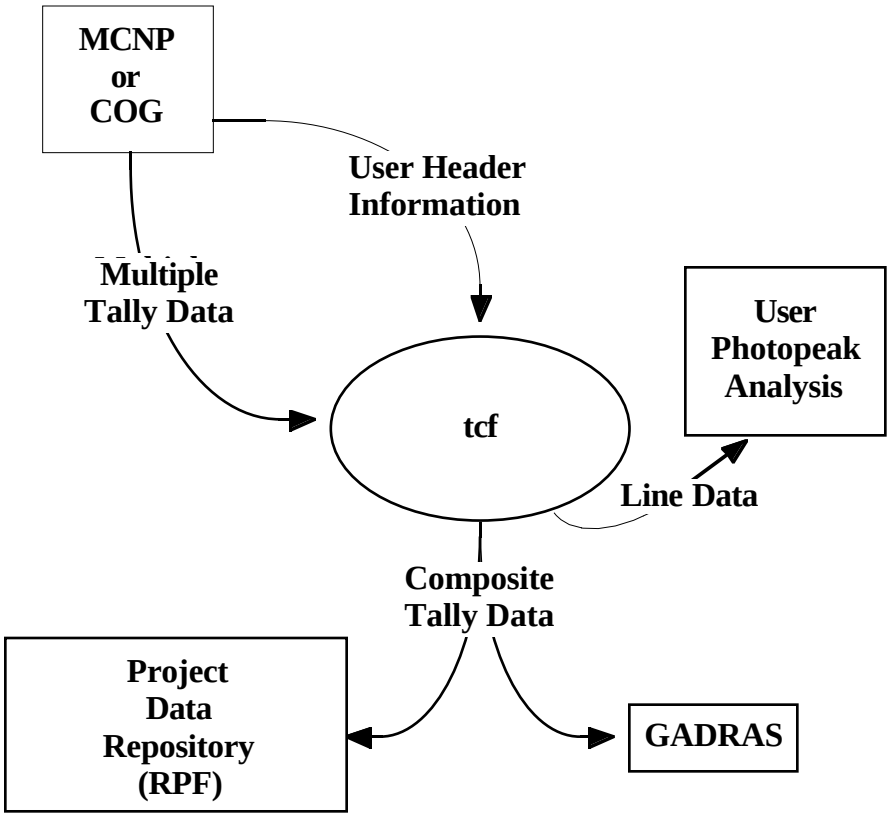
```
struct tallyspec
{
    int length ;          /*  Tally length     */
    int arraySize ;       /*  Array size       */
    double *energy ;      /*  Photon energy    */
    double *flux ;        /*  Particle flux    */
    double *relError ;    /*  Relative error   */
    double *absError ;    /*  Absolute error   */
} ;
```

Instances:

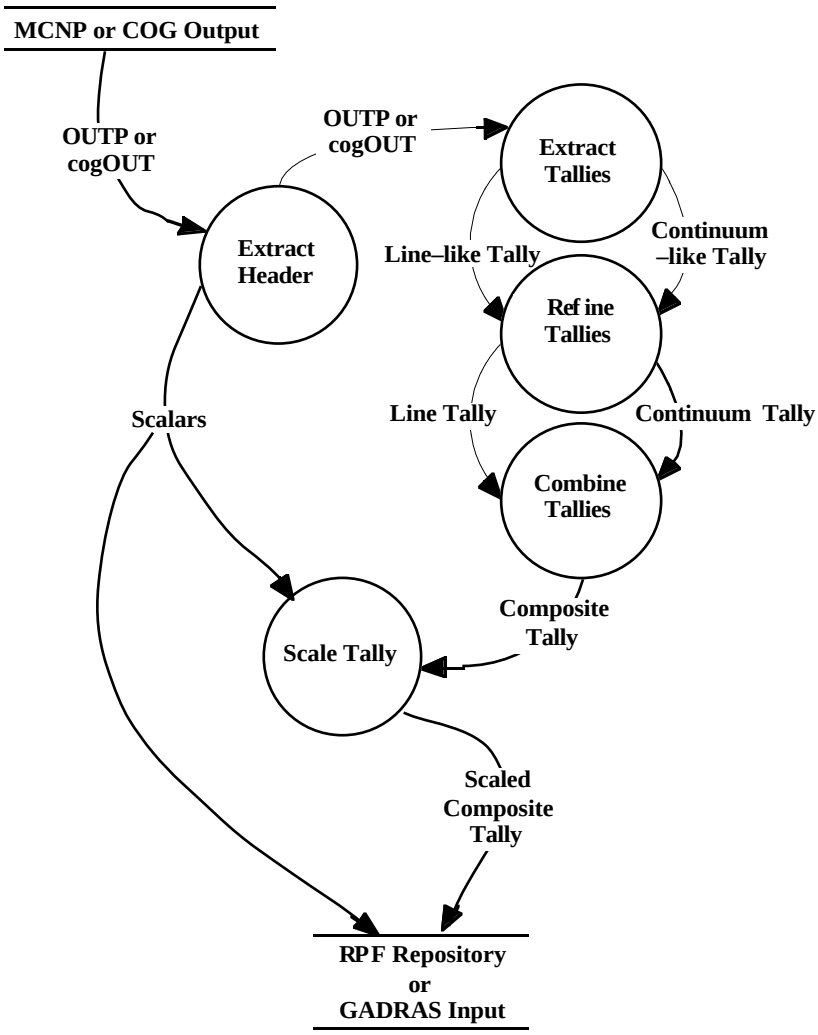
1. Line tally
2. Continuum tally
3. Composite tally
4. Primitive (temporary) tallies for line and continuum

tcf Software Design Description—Appendix A

This section describes graphically how the *tcf* tally data are processed.



tcf Context Diagram
(Level 0 Data Flow Diagram)



tcf Level 1 Data Flow Diagram

tcf Software Design Description—Appendix B

This appendix presents a reverse engineered description of the legacy SELECTOR program in pseudocode form. Portions of this description were reused in the design of tcf, specifically the TallyIO and TallyOp modules.

PseudoCode for SELECTOR (reverse engineered)

Assumptions:

1. Input data will consist of MCNP OUTP type files
2. OUTP type files will contain particle tallies
3. Transported particles are photons

Program Statement

Constants and variable definitions

Get required scalar info from file containing header
    Prompt for filename
    Open file containing header info
    Try to read everything in the header
    If header found
        echo complete header on stdout
    Endif

Find the “cut” energy and save

Check status of file containing tallies
    Prompt for filename
    Open file
    Exit if error
<TOP>
Get name of output file
    Prompt for filename
    Open file
    Exit if error

Attempt to get problem title by lookup in the MCNPCALC database
    Open database file
    Look for tally file name
    If found
        read problem title
    Else
        prompt for new file name or press on
    Endif

```
Write the header into the output file
    Write the existing header down to the last line into the output file without change
    Prompt for RANGE and OFFSET and write them
    Prompt for various tally multipliers and write them
    Prompt for additional user comments and write them
    Write the closing lines

Prompt for point source option flag

If point source not chosen
    Prompt for dual tally option flag
    If dual tally chosen
        Prompt for x-ray bin size
    Endif
Endif
<READ LOOP>
Initialize indices i and igroup
Begin loop to read in and write out tally data: loop forever
    Read a line
    If line contains TALLY or CELL and not IMPORTANCE or FLUCTUAT then
        {Skipping down to top of tally data}
        Increment igroup
        Write igroup to stdout as DATA GROUP
        If line contains TALLY then  {expecting 8 lines to follow)
            Begin loop on j=1…9 to read and echo lines to stdout
                Read a line
                Echo ≤80 chars from line to stdout
                If line contains ENERGY
                    go to <SELECT0>
                Endif
            End loop
        Endif
        If line contains CELL then {expecting 1 line to follow)
            Begin loop on j=1…8 to read and echo lines to stdout
                Read a line
                Echo line to stdout
                If line contains ENERGY
                    go to <SELECT0>
                Endif
            End loop
        Endif
        If line contains DETECTOR LOCATED then {expecting ?  lines to follow}
            Begin loop on j=1…8 to read and echo lines to stdout
                Read a line
                Echo line to stdout
                If line contains ENERGY
```

```
                        go to <SELECT0>
                    Endif
                End loop
            Endif

<SELECT0>       {Energy sentinel found; Do we want this data?}

            Write partial prompt “Do you want to select this group...”
            If dual tally flag = ‘y’
                Complete prompt “...for x-rays”
                If answer is yes
                    Set dual tally flag = ‘d’
                    go to <SELECT1>
                Endif
            Endif
            If point source flag = ‘y’
                Complete prompt “...for Luisa Hansen option only”
                If answer is yes
                    Set jflag = 1
                    Set luisa = 1
                    Go to <SELECT1>
                Endif
                If first flag set
                    Go to <SELECT3>
                Endif
                Complete prompt “...for fine bins direct only”
                If answer is yes
                    Set jflag = 1
                    Go to <SELECT1>
                Endif
<SELECT3>
                Complete prompt “...for coarse bins scattered and direct”
                If answer is yes
                    Set jflag = 1
                    Set point source flag = ‘p’
                    Go to <SELECT1>
                Endif
                Go to <SELECT1>
            Endif
            If point source flag = ‘p’
                (Do point source subtraction)
                Go to <CLOSE FILE>
            Endif
            If igroup is even number and kflag = 2
                Complete prompt “...for x-rays”
                set kflag = 0
```

```
                    If answer is yes
                        Set kflag = 1
                    Endif
                Endif
                Complete prompt “...for current CDF”
                set jflag = 0
                If answer is yes
                    Set jflag = 1
                Endif
<SELECT1>
                If jflag or kflag = 1
                    If jflag = 1
                        Increment outer loop index i
                        (Set first elements of “a” arrays for energy, flux, & error)
                        {Use “cut” value for energy(1) }
                    Endif
                    If kflag = 1
                        Set {x index} inx = 1
                        (Set first elements of “x” arrays for energy, flux, & error)
                    Endif
                    {Begin loop to read tally lines}
                    While (get line ≠ “TOTAL”)
                        If read error
                            Go to <Set Index>
                        Endif
                        Read a tally line
                        Set origflx = flux
                        Scale flux by multipliers
                        Scale abserr by multipliers
                        If jflag = 1
                            Increment outer loop index i
                            (Set i’th elements of “a” arrays for energy, flux, & error)
                        Endif
                        If kflag = 1
                            Increment x index inx
                            (Set inx’th elements of “a” arrays for energy, flux, & error)
                        Endif
                    End while
                    If Luisa Hansen option set
                        Close output file
                        Reopen output file with status = “new” {i.e. overwrite}
                        Write header
                        Loop for j=2…i
                            Write jth element of “a” arrays into output file {5 column}
                        End loop
                        Unset Luisa Hansen option
```

```
                    Set jflag = 0
                    Set i = 0
                    Set first flag = 0
                    Go to <READ LOOP>
                Endif

<SET INDEX>
                If kflag = 1
                    Set kflag = 2
                Endif
                If first flag = 0
                    If i > 0
                        Set first flag = 1
                    Endif
                    Set peaktot = i
                Else
                    Set scattot = i
                Endif
            Endif
        Endif
End loop

<EOF>
        If peaktot = 0
            Announce “No groups selected”
            Go to <CLOSE FILE>
        Endif

If Dual tally flag set
        (Subtract out the x-ray flux)
        Write the “a” flux quantities into output file
        Go to <CLOSE FILE>
Endif

If scattot = 0[No scattering => no cleanup needed]
        Go to <BYPASS>
Endif

Prompt for whether to “clean up” file {i.e. to have non-photopeak data moved}
If answer not ‘n’
        (Clean up file)
Endif

<BYPASS>
Loop over i=1…peaktot
        If kflag = 2
```

```
            For ith element, add x-ray flux to the flux
        Endif
        Write out ith element of “a” arrays into output file
End loop

If scattot = 0
        go to <CLOSE FILE>
Endif

If kflag = 2
        (Subtract x-rays)
Endif

Loop over i = peaktot+1, scattot
        Write ith element of “a” arrays into output file
End loop

<CLOSE FILE>
Close output file
Prompt for more CDF files to be produced from this input
If answer is yes
        Go to <TOP>
Endif

Close any opened input files

End
```

tcf man Page

NAME

tcf - combine and format the tally output from Monte Carlo radiation transport codes MCNP and COG into various output formats

SYNOPSIS

tcf [-v] [-m mult] [-s] [-h] [-l rpflib] [-f filename] [< input] [> output]

DESCRIPTION

tcf accepts ascii output files produced by the MCNP or COG transport codes. It extracts the particle tallies. It scales these tallies according to geometric or physical constants provided by the user. It "refines" the tallies by removing spurious flux, making them represent the flux due exclusively to the continuum or to spectral lines as appropriate. It will optionally produce output in the CDF format, identical to the legacy SELECTOR code. Otherwise, it will combine the tallies into a single spectrum, representing both continua and lines. It will then convert to units and output format compatible with the GADRAS analysis code. It will alternatively create output in the RPF dialect of NCSA HDF.

OPTIONS

-f file -- explicit input file designation

Attempt to use the file "file" as input. This option is required whenever the -v (verbose) option is given, to disambiguate the input data stream from interactive user commands. Otherwise standard UNIX I/O redirection is preferred.

-h -- help

Print a brief paragraph explaining the available options and their corresponding uses.

-l rpflib -- create/append to RPF library

Output the line, continuum and combined tallies in RPF format. RPF is a dialect of NCSA HDF, developed by C. L. Ham at LLNL.

-m mult -- apply aggregate multiplier

Scale tallies by a single quantity "mult", representing the product of all component multipliers present in the transport problem. The user must compute this quantity before execution to use this option. This option obviates the need to enter component multipliers individually during processing.

-s -- SELECTOR
Process the tallies only as far as the legacy SELECTOR code would. This includes extraction and pre-processing to create true line and continuum flux tallies, as well as scaling. Produce output in the CDF format, identical to the legacy SELECTOR code.

-v -- verbose
Process tallies in verbose mode. This option permits the user to select from an arbitrary number of tallies which may be present in the transport code output. It prompts the user to decide, for each tally it encounters, whether the data represent a desired line tally, continuum tally, or neither. It also prompts for each tally multiplier individually, and well as the problem range and offset.

NOTES

tcf will stop processing if it detects a format irregularity in the input tally file[s]. Such irregularities may include legitimate transport code output tokens which were simply not anticipated during development.

PROBLEMS

tcf does not check to verify that it is processing photon tallies. Other types of tallies can be present in problems which transport neutrons or electrons in addition to or instead of photons.

In the unlikely event that line and bin edge energies should exactly coincide, the tcf flux correction will give an incorrect result for the affected line and bin.

DIAGNOSTICS

tcf gives contextual information about the transport problem, as well as each tally label as it processes the tally. It gives timing statistics at the end of execution.

tcf flags if it encounters format problems in reading a tally, or if it fails to find the expected number of tallies in the input stream (≥ 2).

If the user requests RPF library output on a platform where RPF sup port is unavailable, tcf will advise the user and quit before processing, suggesting another option for output.

SEE ALSO

Offline documentation for: rpf, gadras, selector, mcnp, cog

Appendix D—Overview of Background Radiation Measurements

Introduction

In this report we describe several measurements of naturally occurring gamma ray and neutron background radiation. These measurements were mad in several different locations. The locations were varied to determine the possible breadth of the variation of the radiation background.

In this document we describe the experimental philosophy of the background measurements. In this discussion we include information about the location where the background measurements were performed. Next we discuss details of the experimental measurements. This discussion includes details of both the hardware and software used to perform the measurements.

Experimental Philosophy

We performed a series of measurements with several detectors, which included gamma ray detectors (both High Purity Germanium detectors (HPGe) and Sodium Iodide detectors (NaI)) and neutron detectors. The idea of the series of measurements was to establish an understanding of the nature of background radiation in various locations; in essence to get a baseline "database" of the "natural" background radiation. If a baseline background radiation is determined it is then possible to establish the possible presence of a "non-natural" source of radiation.

We performed background measurements in three different locations. Other locations were considered but time did not allow for all of the measurements to be performed; we will discuss other possible location for background measurements.

The first series of measurements were made on the Lawrence Livermore National Laboratory site, we will call this a semi-urban site. The location allowed us to understand the background in a setting that had very little man-made development. This location also had the distinct advantage of being close to our laboratory so it served as our final shake-down for the equipment.

The second series of measurements were made in the financial district in San Francisco, we will call this an urban site. This location is distinct from the semi-urban site because of the presence of large buildings made of concrete. The presence of the minerals in the concrete and other construction materials serves as a perturbation to the natural background. Therefore, the background measurements will be slightly different than the LLNL background data.

The third location was taken at Cypress Point on the Monterey Peninsula. This is a very different location than the other two locations because there only slight man-made development. In addition, the seawater will give different types of nuclear background. Other possible locations for background measurements in California are Carlsbad Caverns because of the presence of naturally occurring radioisotopes and Yosemite National Park because of the large amount of granite in the park.

Experimental Procedure

In this section we will describe the experimental procedure used in the background measurements. All of the hardware, both the electronics and detectors were installed in a 1994 Chevrolet G30 Extended Body Sportvan. The van had been purchased for a different project and we were able to use it for very little cost. The van was equipped with gasoline fueled generator for the electronics racks and computer equipment.

Hardware

We used five different detectors for the measurement of the background radiation. These detectors were:

- 8% HPGe Detector for gamma rays
- 50% HPGe Detector for gamma rays
- 5 by 2 inch NaI Detector for gamma rays
- INS NaI Detector for gamma rays
- Neutron Detector

The HPGe detectors and the 5 by 2 inch NaI detector were standard gamma ray detectors. The INS NaI was a non-standard 2 inch by 6 inch NaI detector with a built-in Am source for internal calibration (this detector was removed in later runs because it was thought the Am source would contaminate the background measurement). The neutron detector was a custom built detector made with four 1 inch ^{3}He tubes with the use of one-inch polyethylene moderator. The signal out of each of the ^{3}He tubes were summed together to increase the response of the neutron detector.

The electronics used for these measurements were standard off the shelf electronics available from Ortec, Inc. and Canberra, Inc. A schematic diagram of the electronics is shown in Fig. D-1. The linear amplifier were relatively important for proper measurements. The amplifiers for the various detectors were:

- 50% HPGe - Canberra 2025
- 8% HPGe - Ortec 672
- 5 by 2 inch NaI - Ortec 572
- INS NaI - Ortec 572
- Neutron Detector - Ortec 572

The unipolar output of the linear amplifiers were inputted into directly into the ADCs. The bipolar output was used to generate the signal which was used for the scalars. In addition, a pulser was used to determine the dead-time in the HPGe electronic circuit. The deadtime for the other detectors (and the HPGe detectors) was measured directly. This dead-time was determined to approximately 10 percent. This dead-time was due mostly to the ADC itself.

The ADCs which were used were ORTEC AD413, which is quad 8K CAMAC ADC. In addition, a ORTEC HM413 histogramming memory was used to reduce the overall dead-time by reducing the interrupts from the computer. The CAMAC crate control was

a Jorway 73A which was interfaced to a APPLE Macintosh PowerPC 8100 through the SCSI port.

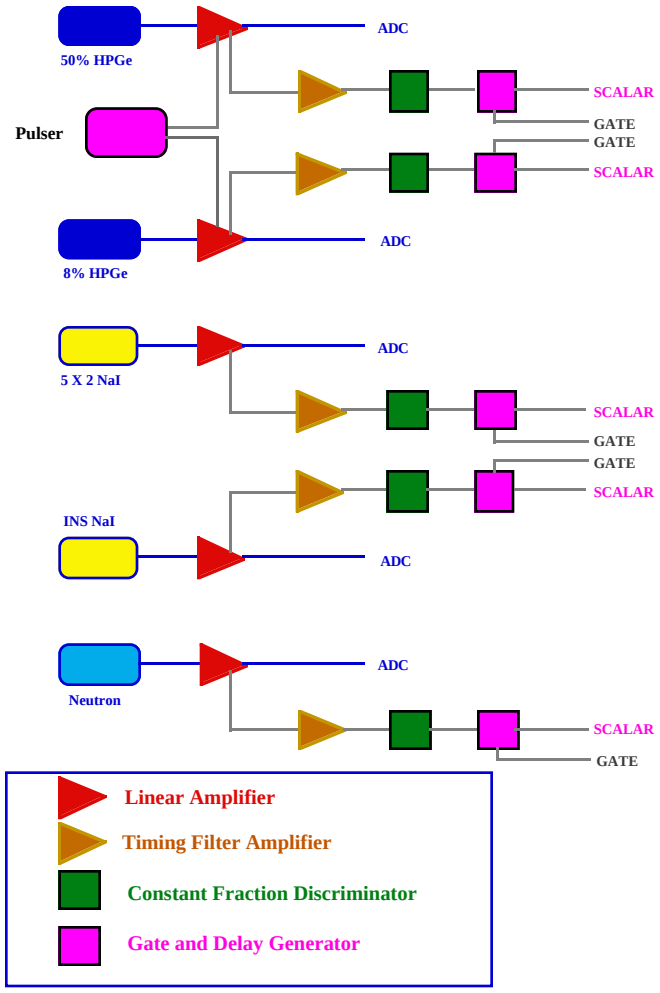


FIGURE 1. Schematic Diagram of electronic used for background measurements.

Software

The software for the data acquisition was a commercial package from Sparrow, Inc. called KAMAX; used version 5.2.1 of this product. This software product is a fairly

flexible which always the user to design an instrument using CAMAC modules. The user interface which we designed for these measurements is shown in Fig. D-2.

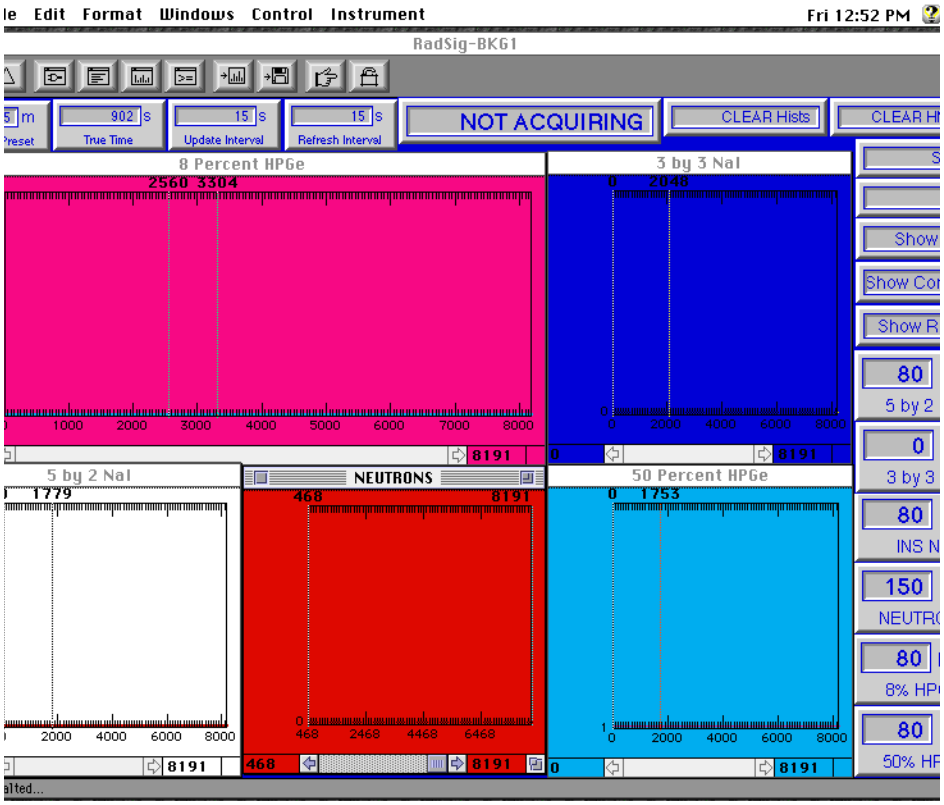


FIGURE 2. Graphical User Interface for the background measurements.

The data were collected in two ways. First, the data were collected so that the counts various energy was displayed in the form of histograms; these data are the sum total of the data. Second, the data were collected in “event” mode, where that data was time-stamped every 1/4 second. The data in this form gives a snapshot of the background every 1/4 second.