DRM Notes

Brant Carlson (brant.carlson@ift.uib.no, bcarlson1@carthage.edu)

July 16, 2015

Contents

1 Introduction

This document describes detector response simulations of ASIM, including contributions from ACES and Columbus. The goal is to produced detector response matrices. The matrices given in this document are preliminary and should not be used for analysis, but can easily be improved. Please contact me with any and all questions.

1.1 Primer on gamma-ray spectra

Data from particle detectors often needs a great deal of interpretation. The spectrum measured by a solid state detector for a simple monoenergetic gamma-ray source like Cesium-137, shown in Figure 1, can be surprisingly complicated depending on the physics involved. There are several possible identifiable features:

Full-energy peak All of the energy of the incident photon is absorbed within the detector.

Compton edge The incident photon Compton-scatters $\sim 180^\circ$ within the detector and escapes, the Compton electron deposits all its energy.

One-escape peak The incident photon pair-produces in the detector, the positron annihilates in the detector, and one of the annihilation photons escapes the detector.

Two-escape peak The incident photon pair-produces in the detector, the positron annihilates in the detector, and both of the annihilation photons escape the detector.

Compton continuum The incident photon Compton-scatters at an angle less than 180° and escapes, the Compton electron deposits some or all of its energy.

Annihilation line The incident photon pair-produces somewhere outside the detector, the positron annihilates, and one of the annihilation photons is detected at full energy.

Backscatter peak The incident photon Compton-scatters at a large angle outside the detector, then enters the detector and deposits all of its energy.

X-ray lines The incident photon is photoelectrically absorbed outside the detector, ejecting an inner electron from the absorbing atom. X-rays are emitted as remaining atomic electrons relax to fill the hole left by the ejected electron. These x-rays can be detected.

X-ray escape lines The incident photon is photoelectrically absorbed inside the detector, ejecting an electron from an atom, but the x-rays released as the atom relaxes back to its ground state escape from the detector, resulting in detection of less than the full energy of the incident photon.

There spectral features will not all appear depending on the energy of the incident photon. In the case of MXGS, we expect a broad distribution of incident photon energies, meaning the measured spectrum will be a combination of some or all of these spectral features and their variation with energy convolved with the incident photon spectrum. Perhaps suffice it to say this is a complicated problem made only more complicated by secondary processes like multiple scattering from non-detector mass, simultaneous energy deposition in multiple detector elements, and incident particles other than photons. The whole problem must also be treated as a function of the direction from which the incident particle arrives.

1.2 Spectrum determination

Given these complications, it is not generally possibly to determine the incident photon energy spectrum directly from the observed “counts” spectrum: the inverse problem cannot be solved. The best that can be done is to solve the forward problem: predict the observed spectrum that would result from a given incident spectrum. This can be done by simulating the response of the detector to particles with a variety of energies. One can then combine these single-energy responses to determine the response of the detector to a spectrum of incident particles.

Typically, this is expressed as a detector response matrix (DRM). Each row of the matrix represents the likelihood of observing a count at a particular energy as a function of the incident particle energy. Alternatively, each column of the matrix represents the spectrum of counts observed for a particular incident photon energy. The response of the detector to a given incident particle spectrum is the simply the matrix-vector product of the DRM with the incident particle spectrum. This DRM is a function of the incident particle direction and of course particle type. There are some subtleties, of course, described below, but this is the fundamental outline of the

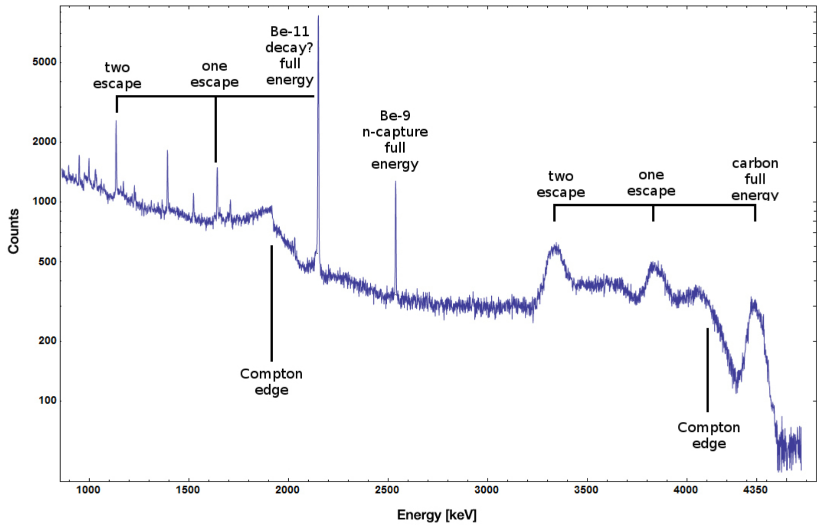


Figure 1: Sample gamma-ray spectra, here from an Americium-Beryllium neutron source measured with a Germanium detector. Various features are speculatively labeled, but many more are visible.

process. The goal of this document is to simulate the detector response and determine the detector response matrices relevant to MXGS.

This goal breaks down into two main parts: detector simulation (Section 2), and detector response matrix construction (Section 3). The technical details of the format and use of the resulting DRMs is described in Section 4.

2 Detector simulations

The simulation of the response of a particle detector is typically done as a Monte Carlo simulation of incident particles and how they might interact with the detector and its support structure. These Monte Carlo simulations avoid the difficulties of accurately calculating the extremely complex integrals over incident particle position, possible interaction locations, possible interaction types, possible product particle geometry, possible product particle interactions, ... etc. Such simulations have a long history in particle physics, and the tools used to construct them are quite mature. The simulation tool used here is GEANT4, produced by CERN. GEANT4 has been used for many years, and is used not only for detector simulations but also for particle accelerator design, radiotherapy, and medical physics, and is in general useful whenever energetic particle properties need to be simulated. A GEANT4 simulation consists of several parts: the relevant laws of physics, the geometry of the simulation, generation of initial energetic particles, and data collection from the "sensitive" portions of the geometry (the detector), all tied together in a single program. Technical details are given in the appendix.

2.1 Relevant laws of physics

GEANT4 has an extremely flexible notion of "the laws of physics." A simulation can be constructed that handles fictitious particles (the "geantino" for example), or uses any number of more realistic models of the familiar physics processes like Compton scattering. Thankfully, GEANT4 includes many pre-defined "physics lists" that combine all of the physics GEANT4 knows about with various approximations suitable for various regimes. Note that this is both good, as a user has some assurance that GEANT physics resembles real physics, and bad because GEANT often includes too much physics (e.g. photo-nuclear reactions, decay of the resulting nuclei, etc.) and therefore can run very slowly and give confusing results. The simulations described in this document use the $\mathrm{QGSP}_{\mathrm{BERTLIV}}$ physics list, which includes GEANT4's treatment of:

photon Compton scattering, pair production, and photoelectric effect, with extensions to include photo-nuclear reactions.

lepton Multiple scattering, ionization, bremsstrahlung, and annihilation, for electrons, muons, and taus, and their antiparticles, with extension to include electro-nuclear interactions.

hadron Relevant physics for charged hadrons (protons, neutrons, pions, kaons, deuterons, etc.), and assorted inelastic processes (e.g. pion absorption).

decay All unstable particles decay with the relevant half-lives, probabilities, and product particles.

Note that the first version of this document used the LHEP physics list, which has been declared obsolete in favor of $\mathrm{QGSP}_{\mathrm{BERTLIV}}$. Comparison of simulations made with LHEP to simulations with the updated physics list are not shown in this document, but the results are essentially identical except for the appearance of x-ray escape lines.

2.2 Geometry

GEANT treats the geometry of parts of the simulation as simple solids (cubes, cylinders, tubes, spheres, etc.), combined with Boolean operations (union, difference, intersection), the "constructive solid geometry" (CSG) approach. This poses some difficulties, as the parts used for MXGS are designed with Creo Elements/Pro (Pro/Engineer), which uses a surface-based representation of parts, the "boundary representation" (BREP) approach. While BREP is very useful (almost all CAD programs use BREP), there is no simple and efficient way to convert from BREP to CSG for use by GEANT.[1] As such, it was necessary to manually construct the geometry for GEANT4 from technical drawings of MXGS parts. This has the decided disadvantage of requiring manual attention whenever the design changes. Hopefully in the future this problem can be solved.

The geometry of the simulation is described by a series of files written in Geometry Description Markup Language (GDML, file extension .gdml), an XML-based format describing sizes, shapes, positions, and materials of elements of the simulation. These files were constructed by hand over the series of several weeks. If this seems like a waste of effort, discussions with the Fermi/GBM have repeatedly suggested that such detailed models of the spacecraft are necessary to ensure accurate simulation results: one small detail may not matter, but taken together, many small details can be quite important. For example, the housing of a single BGO crystal is an aluminum box several millimeters thick but with thinner triangular-shaped regions milled away to reduce mass while retaining stiffness. In reality, the thick regions of the box will block low-energy x-rays, while the thin regions will tend to allow such x-rays to pass. As such, it is not correct to approximate the box as uniformly thick (too many low-energy x-rays blocked), uniformly thin (too few low-energy x-rays blocked), or uniformly intermediate (intermediate amount of x-rays blocked but with the wrong energy dependence). Details like that may not be relevant in the end, but tending to include too much detail is prudent.

Each GDML file has the following sections

[1]And not for lack of searching. The best pathway from Pro/E to GEANT is to use Pro/E to produce a STEP file, convert the STEP file to a GDML file with FastRad (commercial), and read the GDML file with GEANT. there are some non-commercial tools, but none that actually work. I thought for a while that I could construct a chain from Pro/E to BRLCad to a general output format that I could write a program to convert to a format suitable for GEANT4 (BRLCad is also a CSG system), but I decided the problem was too complicated.

define Definitions of constants, positions, and rotations.

materials Definitions of elements and mixtures used to fill detector volumes.

solids Definitions and combinations of shapes to define volumes.

structure Definitions of volumes for simulation, which link solids to materials, and physical volumes, which link volumes to positions and rotations.

setup Identifies the "world volume" for the GDML file, within which all simulation will take place.

GDML files can include each other, so the overall structure is as follows:

columbus.gdml geometry of the Columbus module, includes asim.gdml and aces.gdml

asim.gdml includes mmia.gdml and mxgs.gdml

mxgs.gdml geometry of the outer structure of MXGS, includes codedMask.gdml and instrument.gdml

instrument.gdml geometry of the shielding box, includes bgo.gdml and czt.gdml

aces.gdml includes a crude model of the ACES instrument.

codedMask.gdml generated by code in makeCM.py, describes the geometry of the tungsten shield in the coded mask.

bgo.gdml geometry of the BGO detectors, including support structure.

czt.gdml geometry of the CZT detectors, including support structure.

The drawings used to construct these files are current up to early 2012, with the exception of some of the new thermal and support structure.

The geometry used in the simulations is shown in Figure 2.

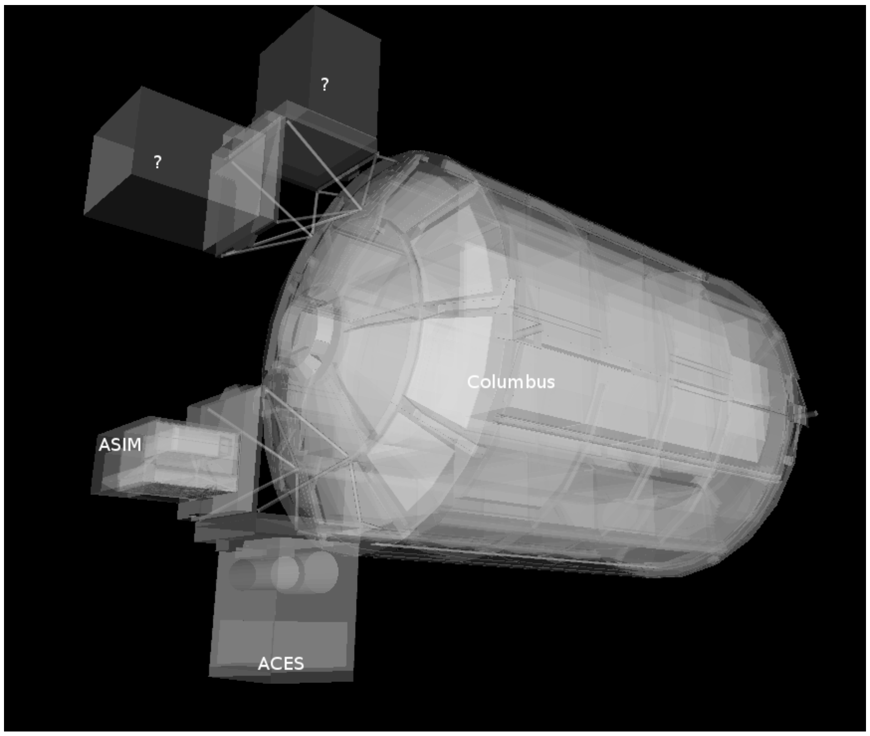


Figure 2: view of the geometry as used in GEANT.

2.3 Primary particle generation

The primary particles to be simulated in this geometry are produced in a beam, incident from a given direction, centered on the detector. The coordinate system used for the simulations has polar angle θ and azimuthal angle ϕ, with $\theta = 0°$ corresponding to particles incident from directly below the space station (i.e. directly into MXGS). $\theta = 90°$, $\phi = 90°$ points toward Columbus.

This raises several questions: what initial directions should be used, and how wide should the beam be made? I don't have good answers yet, this section is unfinished. The initial directions are chosen over a grid with resolution TBD.

2.4 Detector response simulation

Given physics, geometry, and a population of initial particles, the simulation can proceed. As the simulation executes, a particle may be detected if it interacts within the sensitive volume of the detector (i.e. within the BGO bar or CZT wafer volumes). Such an interaction will deposit a certain amount of energy in the detector, and the detectors are designed to produce a signal dependent on this energy deposition. The conversion of energy deposition to signal strength is not straightforward, however. In the case of BGO, the position of the energy deposition will determine the efficiency for scintillation photons to make their way to the photomultiplier tube, while in the CZT layer, the position of the energy deposition may fall on a boundary between multiple pixels. For simplicity, we ignore the details of the physics connecting energy deposition to signal strength and simply record energy deposition. In keeping with the framework described above of a single detector response matrix, we also sum all the energy deposition due to a single incident particle within the BGO and CZT layers. In other words, if an incident high-energy photon pair produces in a CZT wafer, there will be multiple energy deposition events as the electron and positron propagate out of the CZT wafer, and possibly multiple energy deposition events in multiple the BGO bars as the electron, positron, and/or annihilation photons are absorbed. In the simulation, such a process results in two numbers, the total energy deposited in the CZT wafers and the total energy deposited in the BGO bars.

As the simulation continues, many primary particles are simulated, producing many CZT and BGO energy deposition events. These events are recorded and will be used to construct the detector response matrix.

2.5 Simulation control

The structure described above is held together by the main simulation program, mxgsDRM.cc. This program takes a variety of parameters determining the simulation to be executed:

```
./mxgsDRM interactive(0|1) priPDGID(22,11,-11,...) \
    nPriPerE priStartDiskRad(m) priStartDiskRad0(m) \
    theta(deg) phi(deg) Emin(MeV) Emax(MeV) numEnergies \
    outEMin outEMax outNumE outputfileName  \
    ...rest of arguments written as comment to output file...
```

interactive 0 for automatic run, 1 to be given a prompt to issue visualization and simulation commands via the GEANT4 command line.

priPGDID PDG identified for the primary particle (22 for photons, 11 for electrons, -11 for positrons, etc.).

nPriPerE number of primary particles per initial energy bin.

priStartDiskRad maximum radius of beam of incident particles in meters, typically 0.6 m.

priStartDiskRad0 minimum radius of beam of incident particles in meters, typically 0.0 m.

theta polar angle from front of MXGS in degrees for all primary particles[2]

phi azimuthal angle from side of mxgs in degrees for all primary particles[2]

Emin, Emax limits of logarithmic initial energy grid in MeV.

numEnergies number of initial energies to use in logarithmic initial energy grid. For example, Emin = 0.1, Emax = 10, numEnergies = 3 will produce a grid with energies at 0.1, 1, and 1 MeV.

outEMin, outEMax, outNumE limits and number of bins in histogram of energy deposition events in CZT and BGO layers.

outputfileName name of output file to write histograms.

The main program constructs the physics and geometry of the simulation, then produces particles in a beam with the specified geometry. nPriPerE primary particles are produced at each primary energy, and for each primary energy, two histograms counting energy deposition events (CZT and BGO) are accumulated and written to the output file. Sample energy deposition histograms are shown in Figure 3.

In order to achieve an accurate estimate of the detector response matrix, these histograms must each have many events, several thousand at a minimum. As a beam of particles large enough to encompass MXGS, MMIA, ACES, and some of Columbus must be at least 1 m in radius, many particles will not reach the sensitive volume of the detector. As such, around 10^6 initial particles must be simulated at each primary energy (nPriPerE = 10^6). Given a grid of many initial energies, many millions of initial particles

[2]The coordinate system for the simulations has the Columbus module along the $-y$ axis and MXGS along the $-z$ axis, i.e. $-z$ is toward nadir and photons moving in the $+z$ direction will enter the front of MXGS. The theta and phi parameters of the simulation refer to the direction of motion of the photons (i.e. not the direction from MXGS toward the source) in spherical coordinates with theta the polar angle and phi the azimuthal angle, e.g. theta=0 and phi=0 corresponds to photons entering MXGS through the coded mask while theta=45 and phi=90 corresponds to photons that would pass through ACES before entering MXGS.

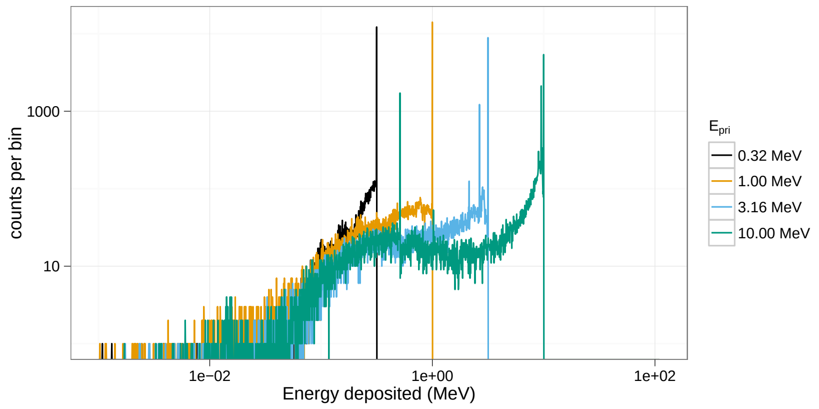


Figure 3: Sample energy deposition histograms for the BGO detector for a variety of primary energies. Note that the energy deposition bins are uniform in logarithmic space. Features like the full energy peak, one- and two-escape peaks, and the 500 keV annihilation line are clearly visible, but the Compton edge feature often seen in such spectra is difficult to identify due to multiple scattering in the complex geometry.

must be simulated in order to construct a single DRM. As the DRM depends on the direction and identity of the initial particles, many DRMs must be created. These simulations therefore take quite a large amount of computer time. As a bare minimum, only running 5×10^5 initial particles per primary energy at a grid of θ with 15° resolution from 0° to 90° (7 θ values) and ϕ with 30 degree resolution from 0° to 180° (7 ϕ values) and a logarithmic grid in energy from 10 keV to 100 MeV with 41 points, just over 10^9 primary particles must be simulated. Test simulations on desktop computers run at an average rate of $\sim$ 500 primary particles per second, so this minimal run corresponds to $\sim$ 20 CPU-days of computer time.

These lengthy run times means running on a supercomputer is beneficial, and thankfully the structure of the simulations poses no obstacle to such simulations. The supercomputer in question is fimm.bccs.uib.no, an 800-core cluster used for a variety of projects. Running a simulation on such a supercomputer entails writing scripts to submit to the job queue. Here these scripts are automatically generated by the program makePBS.py. Changing the parameters in makePBS.py produces a set of .pbs files that can be submitted to the queue. Once submitted, the scripts ensure that the output is placed in a directory of results, ready for processing once the jobs complete. There are a lot of details here that I'm skipping over (copying the simulation to fimm, compiling GEANT on fimm, compiling the simulation on fimm, ensuring the environment is set correctly, submitting the scripts, etc.), but that describes the overall process. See the Appendix for some technical notes.

3 GEANT output processing

As described above, the output of the main simulation program is a file containing two sets of histograms, one set of BGO histograms and one set of CZT histograms. Each histogram describes the number of energy deposition events per energy deposition bin as a function of deposited energy for a single primary energy. These histograms need to be processed to become a true detector response matrix.

First, the histograms must be interpreted in the context of the simulation. The i^{th} entry of a histogram, N_i, refers to the number of times a total energy was deposited in the sensitive detector between E_i^{dep} and E_{i+1}^{dep} (i.e. E_i^{dep} are the bin boundaries of the histogram). N_i can be converted to an effective area by dividing by the total number of primary particles simulated and multiplying by the area over which those primaries were spread: $A_i^{\text{eff}} = \frac{N_i}{n_{\text{pri}}} * \pi * r_{\text{pri}}^2$. This effective area is a function of both the position and the size of the energy deposition bin in question, and typically the size of the energy bin is divided out: $\frac{dA^{\text{eff}}(E^{\text{dep}})}{dE^{\text{dep}}} = A^{\text{eff}}/(E_{i+1}^{\text{dep}} - E_i^{\text{dep}})$. Since this process applies to the histograms generated for each primary energy E^{pri}, the results can be seen as a function also of E^{pri}: $\frac{dA^{\text{eff}}(E^{\text{dep}},E^{\text{pri}})}{dE^{\text{dep}}}$.

This function, determined at the E^{dep} bin centers and each E^{pri}, approximates the true detector response function. ... give examples ... Ideally, the input spectrum would be convolved with this function to determine the detected count distribution. However, the detector does not measure the count distribution, only a sampling from that distribution, binned according to the digitization process during data collection. As such, what we need is not $\frac{dA^{\text{eff}}(E^{\text{dep}},E^{\text{pri}})}{dE^{\text{dep}}}$, but its integral over the output bins. We also need to know $\frac{dA^{\text{eff}}(E^{\text{dep}},E^{\text{pri}})}{dE^{\text{dep}}}$ at all E^{pri}, not just the E^{pri} used in the simulation, requiring some interpolation. Constructing a true DRM from this function subsequently requires assumption of a functional form of input spectrum and integration over some set of E^{pri} bins. This processing, from histogram to function to DRM, breaks down into smoothing of individual histograms, interpolation between histograms, convolution with interpolated histograms, and DRM generation, described as follows.

3.1 Smoothing of single histograms

The simulation results, i.e. histograms such as those shown in Figure 3, are binned with very fine resolution to preserve as much information as possible. The bins are far to fine to be useful, however, with the counts in each bin subject to wild statistical fluctuation. This requires some sort of re-binning or smoothing. Re-binning blurs out spectral features like the 511 keV annihilation line, which we would like to preserve, and smoothing cannot accurately capture such sharp spectral features. Fortunately, such spectral features are limited in number and appear at predictable locations. The only features identifiable in the spectra are the full-energy peak(E^{pri}), the annihilation line (511 keV), and one- and two-escape peaks ($E^{\mathrm{pri}} - 511$ keV, $E^{\mathrm{pri}} - 2 \times 511$ keV). There may or may not also be a contribution from a two-annihilation-photon line (2×511 keV), especially at high primary energies, so this is also included, but x-ray escape lines are not as they are not consistently identifyable. As such, the approach taken here is to smooth the histogram without the spectral lines to give an estimate of the continuum, storing their values separately.

The smoothing technique used is weighted loess smoothing. The loess estimate of a function at a point is the value of a weighted quadratic regression fit to the neighbors of the point in question. The weights are determined by the distance from the point in question and the size of the neighborhood here is taken to be the nearest 11 points. This smoothing effectively dampens out the statistical fluctuations, but note that the resulting points are no longer statistically independent. The result of this procedure is a smoothed estimate of the continuum portion of the histogram.

This background can then be subtracted from the counts in the bins containing spectral lines, giving an estimate of the number of counts in each line. A combination of these peak estimates with the continuum estimate can be compared to the original histogram as shown in Figures 4 and 5. These continuum and spectral line estimates can then be built upon to estimate the overall detector response function.

3.2 Interpolation between histograms

The smoothed single histograms described above only capture the response of the detector to a single primary energy. Multiple smoothed histograms must be interpolated to determine the response at an arbitrary energy between those simulated. This interpolation must include both interpolation of the continuum and of the spectral lines.

The continuum interpolation cannot be done with a typical bilinear method, since bilinear interpolation cannot capture a sharp cutoff that is not aligned to the grid such as the requirement that the maximum energy that can be deposited is the energy of the primary. As such, the interpolation scheme used here is a weighted average of the two histograms with their energy deposition axes scaled such that the full energy peaks align. More mathematically, if $f_1(E^{\mathrm{dep}})$ and $f_2(E^{\mathrm{dep}})$ are the smoothed estimates of the continuum for two nearby primary energies E_1^{pri} and E_2^{pri}, the estimate of the continuum at an intermediate energy E^{pri} is

$$f(E^{\mathrm{dep}}) = f_1(\frac{E^{\mathrm{dep}}E_1^{\mathrm{pri}}}{E^{\mathrm{pri}}})\frac{E_2^{\mathrm{pri}} - E^{\mathrm{pri}}}{E_2^{\mathrm{pri}} - E_1^{\mathrm{pri}}} + f_2(\frac{E^{\mathrm{dep}}E_2^{\mathrm{pri}}}{E^{\mathrm{pri}}})\frac{E^{\mathrm{pri}} - E_1^{\mathrm{pri}}}{E_2^{\mathrm{pri}} - E_1^{\mathrm{pri}}} \tag{1}$$

This essentially interpolates between the two continua along lines radiating out from the origin.

The counts in spectral lines can simply be linearly interpolated.

This interpolation can be tested by interpolating from to the histogram for a known E^{pri} from the histograms from flanking values of E^{pri}. A sample is shown in Figure 6. Though there may seem to be some systematic offsets near full energy, the significance of those offsets is minimal as seen in the bottom panel, and note also that in the actual DRM calculations, the interpolation will be only between neighboring E^{pri}, not over the longer interval in the figure as was done solely for confirmation of the interpolation technique.

There are some small errors for primaries with energies just above 1 MeV due to linear interpolation of annihilation and escape peaks which appear at slightly different rates in the simulation as in the interpolation, but these errors are not washed out in later stages and can be easily be removed if desired.

3.3 Convolution with input spectrum form

The smoothing and interpolation described above all acted on the energy deposition histograms derived from simulations. Construction of a DRM requires integration over bins, both in energy deposition and in primary energy.

As mentioned initially, a DRM is a matrix that converts from an input particle spectrum to a measured counts spectrum. Writing a vector to represent a spectrum either requires some assumption about the interpolation between points or it implies some binning of the spectrum. Interpolation is typically not used as a detector response matrix cannot be written due to interdependence of elements for reasonable interpolation schemes. Binning, on the other hand, requires some assumption of the shape of the primary spectrum within a bin: if one of the numbers in the vector is 10, does that imply 10 primaries uniformly distributed over the bin, or exponentially, or logarithmically, or something else? This is especially important at high energies, where logarithmic bins get very large and the differences between uniform, linear, exponential, and power law spectra are large.

As such, construction of a DRM requires convolution of a primary spectrum with the functions described above. This convolution can be constructed from the smoothing and interpolations described above. There is some subtlety related to the difference between continuum and the spectral line representations, however.

Convolution with the continuum is relatively straightforward and is done over a grid in primary energy over the range of interest. Here the convolution of the primary spectrum with the continuum part of the detector response is simply the sum of the interpolated continua produced by primaries of each primary energy in the grid over the range of interest, weighted by the spectrum and the width of the the primary energy bin. This sum is then normalized over the sum of all the weights used, converting it to a weighted average.

Contributions to the convolution from spectral lines are slightly more complicated. Since spectral lines are arguably delta functions, spectral lines whose positions depend on the primary energy contribute to a particular energy deposition bin by an amount proportional both to the width of the energy deposition bin and to the primary spectrum at the primary energy necessary to put the line in the given energy deposition bin. This calculation has to be done over the entire energy deposition histogram, not simply over the range of primary energies of interest, since the bin sizes are unequal and a grid at full energy will not correctly fill the bins at the two-escape peak, for example. Again, these calculations need to be normalized by dividing by the sum of all the weights used to give a weighted average consistent with the continuum contribution.

Stationary spectral lines (annihilation and $2\times$-annihilation lines) contribute counts given by the weighted average of their interpolated contributions.

Sample convolution outputs are shown in Figure 7.

3.4 Detector resolution

The last step prior to generation of a detector response matrix is to capture the finite resolution of the detector. Every step described above assumes that however much energy is deposited in the detector volume will be measured with perfect accuracy, but this is of course not the case. The generation of scintillation light is probabilistic, as is the resulting release of electrons at the photocathode of the PMT for the BGO detectors. The overall energy measurement essentially involves counting such particles, so the expected behavior is that of Poisson statistics and the width of the distribution of observed energy given deposition of a single energy will be proportional to the square root of the observed energy. Mathematically, FWHM $\propto \sqrt{E}$ where FWHM is the full-width at half maximum (related to the standard deviation in the context of a gaussian as FWHM $= 2.355\sigma$) of a peak observed at energy E. The same statistics are expected to hold for solid-state detectors like the CZT as again the observation essentially involves counting particles generated by a partially-random process.

This $\sigma \propto \sqrt{E}$ dependence is easy to account for given a reference measurement to set the proportionality constant. Given such a reference measurement, observations at a particular energy in the simulated spectra determined above can simply be spread out according to a gaussian distribution with width determined by the proportionality. This can be accomplished by multiplying the spectrum in question by a matrix constructed such that the \$n\$th column in the matrix is a gaussian distribution centered at the \$n\$th energy. The \$i,j\$th entry of the matrix thus contains the integral over the range of energies covered by the \$i\$th bin of a gaussian distribution with mean E_j and standard deviation $\propto \sqrt{E_j}$. Note that some counts will be lost here at low energies, since the gaussian may extend below zero energy, but this is expected as deposition of a small amount of energy might not be detectable purely by chance not be detectable.

The proportionalities used here are set by measurement. For the BGO, FWHM/energy of approximately 15% is found at 0.662 MeV (see the Spectroscopy Test Report, ASIM-UB-UBINT-RP-006). This corresponds to 12% at 1 MeV for a standard deviation of 5%. For CZT, resolution of 10% at 60 keV is appropriate, though the peak is not exactly gaussian so the approach of convolving with a gaussian is perhaps too simple.

3.5 DRM generation

Given convolution of a given primary spectrum with the interpolated detector response simulation results and with the expected resolution of the detector, construction of a DRM requires repeated calculation of that convolution over the required primary energy bins. The resulting detector response may not have the desired energy deposition bins, requiring re-binning by summing such that the counts are properly distributed over the desired bins.

The final conversion that must be made is to normalize the matrix. All of the manipulations described above act on histograms of observed counts, as the histograms are more smooth and their statistics are more easily understood than for representations of functions describing effective area per energy deposited. As such, the matrices must be normalized in the same manner as histograms, i.e. by dividing by the total number of primaries simulated and multiplying by the area illuminated by the primaries. This gives the DRM as measured in effective area, i.e. cm^2. An image plot of a sample square DRM produced is shown in Figure 8.

All of the analysis described above can be repeated for each simulated primary particle direction. If desired, interpolation in direction can be made with the DRM matrix entries.

3.6 Error analysis

Given the complexity of the above analysis, it is not particularly easy to estimate the statistical errors present in the final convolved DRM. However, multiple simulations with the same parameters can be passed through the same processing steps and compared. Figure 9 shows the standard deviation of the DRMs from a set of 5 simulations cm^2, and Figure 10 shows the standard deviation divided by the mean in percentage. The largest percent errors are $\sim$ 10 %, but these occur in regions where the effective area is small. These results seem therefore to be quite accurate, typically to within 3%.

The results above were generated by launching particles at a 0.6 m radius disk centered at the center of the sensitive volume of the detector. Choosing such a small radius increases the fraction of incident particles that will be detected, decreasing statistical fluctions on the result, but raises the possibility of a systematic underestimate of the effective area of the detector since incident particles outside the disk may scatter into the detector. We can assess this effect by re-running the simulations instead launching the particles into an annulus with inner radius 0.6 m and a larger outer radius. These particles will entirely miss the sensitive volume of the detector unless they scatter into it, so a calculation of the effective area on the basis of this set of particles spread over their area will provide an estimate of the size of the error introduced by not including such particles. Such “detector response matrices” are shown in Figures 11 and 12.

Note that these estimates are not particularly precise since the area covered by the annulus is relatively large and relatively few particles will be detected so the statistical fluctuations are relatively large. Despite the relatively large scatter, the scale of the effect is clearly a few cm^2 at most and is most relevant for primary energies above roughly 10 MeV and produces observations with energies typically at or below the positron annihilation line. Similar regions of the DRM including particles directly incident on the detector have around 50 cm^2 area per bin, leaving the effect at the few percent level. For lower primary energies, the effect is 1 cm^2 or less and thus is comparable to or smaller than the statistical uncertainty in the DRM simulations.

3.7 Technical details

The manipulations and processing described above are written in R, a language typically used for statistical computing and visualization. The file procDRM.r, included in the appendix, includes functions to read the output from GEANT, smooth histograms, interpolate between smoothed histograms, convolve spectra with detector response, and make plots. The file is heavily commented and should be self-explanatory given the above description. This report is compiled from a file written in Emacs with "org-mode", and the raw file docs.org includes the code necessary to generate the plots.[3]

4 DRM format/usage

A detector response matrix as simulated, smoothed, interpolated, convolved, and normalized as described above, is thankfully straightforward to use. The DRM in question has associated E^{pri} and E^{dep} bins. The desired primary spectrum is integrated over E^{pri} bins, giving a vector of fluence in each

[3]Org mode is an Emacs file editing mode that is very useful for projects like this. Snippets of code can be directly executed or written to files for compilation/execution elsewhere, results of code execution can be collected automatically and included in the file, and the file itself can be exported to other file formats. The net effect is to produce a single file that contains all the details necessary to reproduce the results.

primary energy bin (counts per area in each bin). The DRM (in cm^2) is then multiplied by this vector, giving the predicted counts in each E^{dep} bin. These deposited energy counts can be directly compared to the measured counts.

Two sets of bins are provided, and more are easy to generate if desired. The first and simplest to describe is 40 bins logarithmically spaced in energy from 0.01 keV to 100 MeV. Bin boundaries are given by in R, in Matlab (or Python/NumPy), and 41 bin boundaries gives overall 40 bins. The second set of DRMs is given in terms of the energies used for the RHESSI detector response matrix, which were hand chosen to be approximately logarithmic but to fall on round numbers, see Table~1. Note that the RHESSI DRM these boundaries were taken from had a 0-20 keV energy bin that we have dropped here.

Table 1: RHESSI-style bin boundaries, one column per decade.

20.0	100.0	1000.0	11000.0
25.0	120.0	1200.0	13000.0
30.0	140.0	1400.0	15000.0
35.0	160.0	1700.0	17500.0
40.0	200.0	2000.0	19999.0
45.0	250.0	2500.0	23000.0
50.0	300.0	3000.0	30000.0
60.0	350.0	3500.0	40000.0
70.0	400.0	4000.0	
80.0	500.0	5000.0	
	600.0	6000.0	
	700.0	7000.0	
	800.0	9000.0	

4.1 results visualization

The detector response matrices produced (see "Accompanying files", below) can be displayed by making a plot of the matrix as an image, see the images subdirectory of the drms.tgz file. These images are present in a variety of ways, either plotted as linear scale or logarithmic, and with the color scale automatically scaled or set to 500 cm^2 as an absolute maximum (useful for comparisons between DRMs). Multi-page combinations of those plots for easy comparison are also provided, in drms$_{\mathrm{theta}}$*$_{\mathrm{phiScan}}$.pdf or drms$_{\mathrm{phi}}$*$_{\mathrm{thetaScan}}$.pdf for scans in phi and theta respectively. Paging through these files gives a good impression of how the effective area changes with direction.

Note that the plots show the effective area as described above with the origin (low energy in to low energy out) in the lower left. This is different from the conventional matrix representation which would place the origin in the upper left (flipping the matrix top-to-bottom) and transposing the matrix. The DRM files use this conventional matrix format. Pay attention to the orientation of the matrix when visualizing, reading, and using these matrices!

5 Appendices

5.1 GEANT4 notes

The GDML files were hand-written to approximate the geometry of the detector as closely as is feasible with the geometric primitives available to GEANT. This is an error-prone process, and geometry errors can lead to simulation errors. One possible way this could happen is if two volumes in the geometry heirarchy overlap, meaning that depending on the way a particle is processed, GEANT could logically decide that the particle is inside either volume. If the two volumes have different materials, the results for that particular particle may differ. Another possible error is if a volume that ought to be fully enclosed within its "mother volume" actually extends beyond the volume. If a particle is inside this region, GEANT might miss this fact and decide that since the particle is not inside the mother volume that it must also not be enside the volume in question and thus decide that paricle interaction is impossible when in fact it ought to occur.

Geometries can be debugged in a variety of ways. The simplest applicable to the DRM code written here is to run the mxgsDRM executable in interactive mode (i.e. with the first command line argument set to 1 instead of to 0), and then when the command line interface appears, run the command /geometry/test/run. This will print out a variety of useful messages, including warnings of overlap, inappropriate enclosure, extension beyond mother volumes, etc. Other methods are described in the GEANT4 User's Guide for Application Developers.

There are some minor overlaps in the geomtery as described here. These overlaps are of order millimeters or microns and thus should not significantly affect the results of a simulation.

5.2 Supercomputer job execution notes

The supercomputer used to run these simulations is fimm.bccs.uib.no. Accounts can be requested at `https://hpc.uib.no/signup/`. Once an account has been issued, GEANT4 must be compiled on the cluster with GDML support and data downloads. For the set of simulations run in June 2015, it was necessary to download and build and install the XercesC++ library (to a prefix like sharedLibs in the user's home directory) in order to support reading from the GDML and the GSL for the histogramming in detector response matrix genertaion (installed to the ~/gsl prefix, since that's where CMake will look, see cmake/FindGSL.cmake in the source directory). It was not necessary to switch the compiler to gcc as it was in June 2012 (see `https://docs.hpc.uib.no/wiki/Application_development_(Fimm)` for info on how to do this if necessary).

After building GEANT4, it is necessary to transfer and rebuild the simulation. This is straightforward transferring with git or rsync and rebuilding with cmake as any GEANT4 application would be built.

The resulting executable should run without issues on either the login node or on the cluster nodes. There may be issues with missing shared libraries on the cluster nodes, but these issues can be rectified by emailing the FIMM help people (hpc-support@hpc.uib.no, probably Saerda Halifu).

With an executable ready to go, the DRM simulations can be executed with the queueing system. Jobs must be submitted with qsub script.pbs, where script.pbs is a job script file. There is a Python script, makePBS.py, to generate these scripts, see especially the writeJobGrid(...) function. Each job script runs one or more DRM simulations covering a chunk of the angle range of interest. The .pbs files generated, once they make it through the queue and run, create output directories for the resulting histograms and populate those directories with the appropriate simulation results.

5.3 Accompanying files

drms.tgz detector response matrices. Example file name: `drm_mats_22_500000_0.60_0.00_45.00_60` signifying a matrix for PDG particle 22 (photons), incident 500000 times on a disk with outer radius 0.60 m and inner radius 0.00 m, 45° degrees from normal, 60° degrees azimuth, with primary energies from 0.01 to 100 MeV, 41 different primary energies. Each column in a corresponds to a given primary energy bin. Each row in a file corresponds to a given bin of deposited energy. The entries in the column are therefore the spectrum of deposited energies produced by primaries with energies in the given bin chosen, measured in cm^2.

plots.tgz plots of the detector response matrices as described above under Results Visualization.

results.tgz simulation output to be processed by the R code described above to produce normalized DRMs.

simulation.tgz GEANT4 simulation code and associated scripts. Files are not completely documented, ask Brant if you have questions.

5.4 procDRM.r

```
# This file includes code to process histograms from GEANT simulations
# into detector response matrices.
#
# WARNING: this file can be manually edited for use directly in R, but
# can also be automatically generated from the appendix in notes/docs.org.
# If you have emacs and org-mode, it is better to edit in emacs and use
# org-babel-tangle to generate the file.  That way the file and the document
# will stay in sync.
#
# Basic usage from the R prompt:
# a <- readDRMs_df("../results/mxgsDRM_1/mats_22_500000_0.60_0.00_0.00_30.00_0.01_1e+0
# f <- drmConvolver(a);
# bins <- 10**seq(-1,2,length.out=40);
# primarySpectrum <- function(e){1/e};
# drm <- makeDRM(f,primarySpectrum,bins,bins);
# image.plot(bins,bins,drm);

# libraries necessary
library(ggplot2); # plotting library
library(reshape); # utilities for rearranging vectors and matrices.
library(fields); # image.plot
#source("~/R/utils.r"); # color maps, multiplot function.

cbpr <- c("#000000", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00",

# from Robin Evans, http://cran.r-project.org/web/packages/rje/index.html
cubeHelix <- function (n, start = 0.5, r = -1.5, hue = 1, gamma = 1, reverse=FALSE)
{
    M = matrix(c(-0.14861, -0.29227, 1.97294, 1.78277, -0.90649,
0), ncol = 2)
    lambda = seq(0, 1, length.out = n)
    l = rep(lambda^gamma, each = 3)
    phi = 2 * pi * (start/3 + r * lambda)
    t = rbind(cos(phi), sin(phi))
    out = l + hue * l * (1 - l)/2 * (M %*% t)
    out = pmin(pmax(out, 0), 1)
    out = apply(out, 2, function(x) rgb(x[1], x[2], x[3]))
    if(reverse){
rev(out);
    }else{
out;
    }
}


# Calculate a confidence interval for probability p in binomial given
# observation of x successes out of n trials, vectorized over x.  default
# confidence level gives 1 sigma error bar if normal approx holds.
# Used for adding error bars to a DRM.
binomCI <- function(x,n,conf.lev=0.6826895){
  #print(c(x,n,conf.lev));
  alpha <- 1-conf.lev;
  p.L <- function(x, alpha){
    y <- x; y[x==0] <- 1;
    ifelse(x == 0, 0, qbeta(alpha, y, n - y + 1));
  }
  p.U <- function(x, alpha){
    y <- x; y[x==n] <- 1;
    ifelse(x == n, 1, qbeta(1 - alpha, y + 1, n - y));
  }

  matrix(c(p.L(x, alpha), p.U(x, alpha)),ncol=2);
}

# construct a matrix that when multiplied by another matrix, gives a matrix
# with row groups summed together.
# used for combining simulation output bins if desired (since they're really small by d
sumRowGroupsMat <- function(nRows,nGrp){
  outer(seq(nRows/nGrp),seq(nRows),function(i,j){ifelse(j/nGrp-i<=0 & j/nGrp-i>-1,1,0)}
}

# read GEANT output.
# effective areas measured in cm^2/keV
# returns a data table with columns for input and output energies, counts and
# effective areas for BGO and CZT layers.  Pay the most attention to the ctsB
# variable, as it is the easiest to understand.  The attributes store relevant
# parameters for later calculation.
readDRMs_df <- function(fn,nPriPerE=1.0,rDisk1=0.6,rDisk0=0.0,combineOutBins=1,defaultI
  f <- file(fn,"rt");
  l <- readLines(f,5);
  close(f);
  priLine <- as.double(strsplit(l[3]," ")[[1]][-1:-3])
  outLine <- as.double(strsplit(l[4]," ")[[1]][-1:-5])

  print("reading file, loading matrices...");
  a <- read.table(fn);
  bdf <- t(data.matrix(subset(a,a$V1=="BGO")[,-1]));
  cdf <- t(data.matrix(subset(a,a$V1=="CZT")[,-1]));

  combMat <- sumRowGroupsMat(dim(bdf)[1],combineOutBins);

  bdf <- combMat %*% bdf;
  cdf <- combMat %*% cdf;

  outLine <- outLine[seq(1,length(outLine),by=combineOutBins)];

  # centers and widths of output energy bins
  om <- (outLine[-1] + outLine[-length(outLine)])/2;
  deo <- (outLine[-1] - outLine[-length(outLine)]);

  bdf <- data.frame(melt(matrix(bdf,nrow=length(om),dimnames=list(om,priLine))));
  cdf <- data.frame(melt(matrix(cdf,nrow=length(om),dimnames=list(om,priLine))));

  e1 <- outLine[findInterval(bdf$X1,outLine)];
  e2 <- outLine[findInterval(bdf$X1,outLine)+1];

  # convert to cm^2/keV
  norm <- pi*(rDisk1**2-rDisk0**2)*100^2/((e2-e1)*1000.0)/nPriPerE;
  x <- data.frame(outE=bdf$X1
  ,inE=bdf$X2
  ,ctsB=bdf$value
  ,areaB=bdf$value*norm
  ,ctsC=cdf$value
  ,areaC=cdf$value*norm
  ,outEBinLow=e1
  ,outEBinHigh=e2);

  x$cts <- if(defaultDet=='bgo'){x$ctsB}else{x$ctsC}
  x$area <- if(defaultDet=='bgo'){x$areaB}else{x$areaC}

  attr(x,"nPriPerE") <- nPriPerE;
  attr(x,"rDisk") <- sqrt(rDisk1**2+rDisk0**2)
  attr(x,"rDisk1") <- rDisk1;
  attr(x,"rDisk0") <- rDisk0;
  attr(x,"norm") <- norm;
  attr(x,"outEBins") <- outLine;
  x;
}

# add columns to DRM data table describing error bars.
# This calculation may take a long time and/or use up all the memory on the
# computer.  Use with caution.
addErrorBars_df <- function(df){
  nPriPerE <- attr(df,"nPriPerE");
  norm <- attr(df,"norm");

  print("calculating BGO error bars...");
  bcis <- binomCI(df$ctsB,nPriPerE);
  print("calculating CZT error bars...");
  ccis <- binomCI(df$ctsC,nPriPerE);
  print("done");

  df$cBmin <- bcis[,1]*nPriPerE;
  df$cBmax <- bcis[,2]*nPriPerE;
  df$aBmin <- bcis[,1]*nPriPerE*norm;
  df$aBmax <- bcis[,2]*nPriPerE*norm;
  df$cCmin <- ccis[,1]*nPriPerE;
  df$cCmax <- ccis[,2]*nPriPerE;
  df$aCmin <- ccis[,1]*nPriPerE*norm;
  df$aCmax <- ccis[,2]*nPriPerE*norm;

  df;
}

# plot the DRM simulation results at the given energies.
# Rounds desired energies down to the nearest value that was simulated.
# geomOnly is useful for those familiar with ggplot for stacking multiple
# plots.
#
# example:
# a <- readDRMs_df(...);
# lineDRM(a,c(1,10,100);
lineDRM <- function(a,e,geomOnly=FALSE){
  ine <- as.double(levels(factor(a$inE)));

  e <- ine[findInterval(e,ine)];

  a <- subset(a,a$inE %in% e);
  a$estr <- sprintf("%.2f MeV",a$inE);
  a$estr <- ordered(factor(a$estr),levels=sprintf("%.2f MeV",sort(unique(a$inE))))

  g <- geom_line(data=a,aes(x=outE,y=cts,group=inE,color=estr));

  p <- ggplot();
  #p <- p + scale_color_brewer();
  p <- p + scale_color_manual(values=cbpr,name=expression(E[pri]));
  p <- p + theme_bw();
  p <- p + scale_x_log10();
  #p <- p + xlim(limits=c(0,5));
  p <- p + scale_y_log10();

  p <- p + xlab("Energy deposited (MeV)");

  p <- p + ylab("counts per bin");

  if(geomOnly){
    g;
  }else{
    p+g;
  }
}

#suggested additional argumens: log="xy", zlim=c(0,500)
plotDRM <- function(inBins,outBins,drm,...){
    image.plot(outBins,inBins,drm,xlab="deposited energy (MeV)",ylab="primary energy (M
}

# makes plot in matrix sense, i.e. with origin at upper left instead of lower left.
plotDRMAsMat <- function(drm,...){
    image.plot(t(drm)[,ncol(drm):1],legend.lab="effective area (cm^2)",legend.mar=4,col
}

# compares two DRM data tables.
# Assumes d1 and d2 are subsets of DRM data frames describing same primary energy.
# d1 is shown in black, d2 is shown in blue.
compareDRMs <- function(d1,d2){
  p <- ggplot() + theme_bw();
  range <- c(0.01,1.1*max(d1$outE[d1$cts>0],d2$outE[d2$cts>0],na.rm=TRUE));
  p <- p + geom_line(data=d1,aes(x=outE,y=cts),col='black');
  p <- p + geom_line(data=d2,aes(x=outE,y=cts),col='blue');
  p <- p + scale_x_log10(limits=range);
  p <- p + scale_y_log10();
  p <- p + xlab("Energy deposited (MeV)");
  p;
}

# take the subset of a DRM data table, preserving attributes.
subsetEADF <- function(df,sel){
  x <- subset(df,sel);
  attributes(x) <- attributes(df);
  x;
}

# interpolate a DRM simulation over bins, separating the continuum (bg) from the spectr
# lines.  i.e. smooth a single DRM to limit noise.
# interpolation BETWEEN multiple DRMs is handled elsewhere.
```

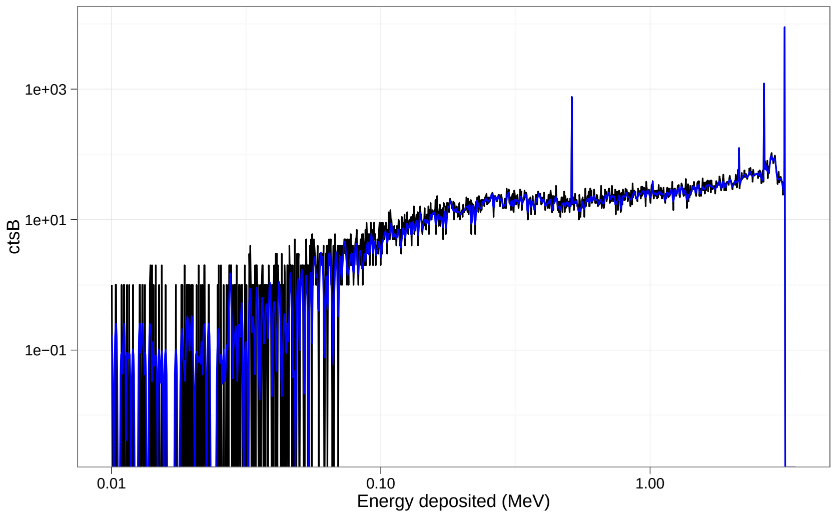
```
interpolateDRMbg <- function(df,e){
  inEs <- unique(df$inE);
  e <- inEs[findInterval(e,inEs)]; # round down to nearest simulated E.
  dfa <- attributes(df);
  df <- subset(df, df$inE == e); attributes(df) <- dfa;
  outE <- df$outE;
  outEb <- attr(df,"outEBins");

  cts <- df$cts;

  eBinMid <- outE[findInterval(e,outEb)];
  eBinMin <- outEb[findInterval(e,outEb)];

  me <- 0.510998903;
  moveLines <- c(e,e-me,e-2*me); # moving lines: full-energy, one-escape, and two-escap
  statLines <- c(me,2*me); # static lines: annihilation, and twice-annihilation.

  # combine lines, ignoring annihilation and escape if primary energy too low.
  lines <- if(e>2*me){c(moveLines,statLines);}else{c(e);}

  # indices in lines array of static and moving lines.
  mvidxs <- if(e>2*me){c(1,2,3)}else{c(1)};
  stidxs <- if(e>2*me){c(4,5)}else{c()};

  # drop bins containing lines from DRM simulation results
  toDrop <- findInterval(lines,outEb);
  outEd <- outE[-toDrop];
  ctsd <- cts[-toDrop];
  outEdd <- outE[toDrop];
  ctsdd <- cts[toDrop];

  # drop everything at or above the full-energy peak.
  ctsd <- ctsd[outEd<=eBinMid];
  outEd <- outEd[outEd<=eBinMid];

  # do weighted LOESS smoothing.  span is the fraction of the dataset to use,
  # and the control variable allows for extrapolation (not really used, but may
  # prevent NA from popping up.  Weights are calculated as in poisson error
  # bars, to prevent bias.
  f <- loess(ctsd~outEd,weights=1/(ctsd+1),span=11/length(ctsd),control=loess.control(s
  # function to evaluate loess smoothed continuum, stripping out NA's and negatives.
  bg <- function(oE){
    ans <- ifelse(oE>eBinMin,0,predict(f,oE));
    ans[ans<0 | is.na(ans)] <- 0;
    ans;
  }

  # subtract continuum contribution from spectral lines.
  peakCts <- ctsdd - bg(lines);

  list(e=e,bg=bg,sl=statLines,slc=peakCts[stidxs],ml=moveLines,mlc=peakCts[mvidxs],outI
}

# add spectral line contributions (lines, lcts) to a histogram containing the
# continuum (outEb, cts).
addLines <- function(outEb,lines,lcts,cts){
  ctr <- 1;
  for(ee in lines){
    i <- findInterval(ee,outEb);
    cts[i] <- cts[i] + lcts[ctr];
    ctr <- ctr + 1;
  }
  cts
}

# convert the results of DRM bg interpolation/smoothing back to a DRM data table (histo
interpDRMtoDF <- function(ntrp){
  bg <- ntrp$bg(ntrp$outE);
  cts <- addLines(ntrp$outEb,ntrp$sl,ntrp$slc,bg);
  cts <- addLines(ntrp$outEb,ntrp$ml,ntrp$mlc,cts);

  data.frame(outE=ntrp$outE,cts=cts,inE=ntrp$e);
}

# interpolation helper function, takes two bg interpolations and an energy, and
# makes a linear interpolation along lines radiating out from the origin.
interpolateH_bg <- function(i1,i2,e){
  f1 <- i1$bg;
  f2 <- i2$bg;
  oE <- i1$outE;

  pf1 <- (i2$e-e)/(i2$e-i1$e);
  pf2 <- 1-pf1;

  x <- oE/e;
  pf1*f1(x*i1$e)+pf2*f2(x*i2$e);
}

# given a DRM data table and two energies to use as interpolation base points,
# interpolate in between.
interpolateDRMs_givenE <- function(df,e1,e2,e){
  # do background/line interpolation at input energies.
  i1 <- interpolateDRMbg(df,e1);
  i2 <- interpolateDRMbg(df,e2);

  # fractional contributions
  pf1 <- (e2-e)/(e2-e1);
  pf2 <- 1-pf1;

  # do background interpolation between energies.
  cbg <- interpolateH_bg(i1,i2,e);

  outE <- i1$outE;
  outEb <- attr(df,"outEBins");

  # add stationary spectral lines to background.
  cbg <- addLines(outEb,i1$sl,i1$slc*pf1,cbg);
  cbg <- addLines(outEb,i2$sl,i2$slc*pf2,cbg);

  # interpolate moving lines to final position, intensity.
  n <- max(length(i1$ml),length(i2$ml));
  mlines <- pf1*c(i1$ml,rep(0,n-length(i1$ml)))+pf2*c(i2$ml,rep(0,n-length(i2$ml)));
  mlcts <- pf1*c(i1$mlc,rep(0,n-length(i1$mlc)))+pf2*c(i2$mlc,rep(0,n-length(i2$mlc)))

  # add moving lines to background.
  cbg <- addLines(outEb,mlines,mlcts,cbg)

  dfa <- attributes(df);
  dfa$names <- c("outE","inE","cts");
  dfa$row.names <- seq_along(outE);
  d <- data.frame(outE=outE, inE=e, cts=cbg)
  attributes(d) <- dfa;
  d;
}

# wrapper for interpolateDRMs_givenE that determines energies automatically.
interpolateDRMs <- function(df,e){
  es <- unique(df$inE);
  if(e<min(es) || e>max(es)){
    print("out of range!");
    return(0);
  }
  e1 <- es[findInterval(e,es)]
  e2 <- es[findInterval(e,es)+1]
  interpolateDRMs_givenE(df,e1,e2,e);
}

# construct a function that convolves a DRM with a given primary spectrum
# between two given primary energies.
drmConvolver <- function(df){
  inEs <- unique(df$inE);
  ntrps <- lapply(inEs,function(e){interpolateDRMbg(df,e)})
  outE <- ntrps[[1]]$outE;
  outEb <- ntrps[[1]]$outEb;
  bgs <- lapply(ntrps,function(ntrp){ntrp$bg(outE)});

  # interpolation of background and static lines
  # returns a histogram of interpolated counts containing background + static line cont
  linBSL <- function(e){
    j1 <- findInterval(e,inEs);
    j2 <- j1+1;
    bg <- interpolateH_bg(ntrps[[j1]],ntrps[[j2]],e); # background interp.
    lns <- ntrps[[j2]]$sl;
    if(length(ntrps[[j2]]$slc)>0){ # static line interp.
      p1 <- (inEs[j2]-e)/(inEs[j2]-inEs[j1]);
      p2 <- 1-p1;
      lcts <- p1*c(ntrps[[j1]]$slc,rep(0,2-length(ntrps[[j1]]$slc))) + p2*ntrps[[j2]]$s
      addLines(outEb,lns,lcts,bg);
    }else{
```

```
    bg;
  }
}

#background and static line convolution
# returns a histogram summed from many results from linBSL
# taken from many primary energies, weighted by the given $sf \propto dN/dE$ spectrum
# grid in input energies is taken from output energies.  reasonable, but not required
bslConv <- function(e1,e2,sf){
  oEs <- outE[outE>=e1 & outE<=e2];
  bwds <- diff(outEb)[outE>=e1 & outE<=e2];
  wts <- sf(oEs)*bwds;
  wts <- wts/sum(wts);
  # sum over primary energies given by output bins between energy limits.
  bg <- Reduce(function(v,i){v + linBSL(oEs[i])*wts[i]}, seq_along(oEs),rep(0,length
}

# moving line convolution
mlConv <- function(e1,e2,sf){
  print(c(e1,e2));
  #poEs <- outE[outE>=e1 & outE<=e2];  # unused?
  #pbwds <- diff(outEb)[outE>=e1 & outE<=e2];  # unused?
  oEs <- outE;
  bwds <- diff(outEb);

  # extract info from the interpolations for the range needed
  #sapply here simplifies down to a matrix, i.e. mlm[,1]= ntrps[[1]]$ml
  i1 <- findInterval(e1,inEs);
  i2 <- findInterval(e2,inEs)+1;
  mlm <- sapply(ntrps[i1:i2],function(xx)c(xx$ml,rep(0,3-length(xx$ml))));
  mlcm <- sapply(ntrps[i1:i2],function(xx)c(xx$mlc,rep(0,3-length(xx$mlc))));

  # make functions interpolating linearly in spectral line positions, cts.
  # note that there are always 3 moving lines (since the arrays above are padded),
  # so nrow(mlm) = 3.
  fs <- lapply(1:nrow(mlm),function(i){approxfun(mlm[i,],mlcm[i,],yleft=0,yright=0)})

  # use functions to determine counts in each output bin, weighted by primary
  # spectrum and bin width.  Note that primary spectrum is evaluated at the
  # primary energy necessary to put the line in question at the output bin in
```

```
    # question.
    me <- 0.510998903;
    f1 <- fs[[1]](oEs)*(oEs>e1 & oEs<e2)*sf(oEs)*bwds;
    f2 <- fs[[2]](oEs)*(oEs+me>e1 & oEs+me<e2)*sf(oEs+me)*bwds;
    f3 <- fs[[3]](oEs)*(oEs+2*me>e1 & oEs+2*me<e2)*sf(oEs+2*me)*bwds;

    n1 <- sum(sf(oEs)*bwds*(oEs>e1 & oEs<e2));
    n2 <- sum(sf(oEs+me)*bwds*(oEs+me>e1 & oEs+me<e2));
    n3 <- sum(sf(oEs+2*me)*bwds*(oEs+2*me>e1 & oEs+2*me<e2));

    a1 <- if(n1>0){f1/n1}else{0};
    a2 <- if(n2>0){f2/n2}else{0};
    a3 <- if(n3>0){f3/n3}else{0};

    a1+a2+a3;
  }

  list(f=function(e1,e2,sf){ bslConv(e1,e2,sf)+mlConv(e1,e2,sf); },
       fb=function(e1,e2,sf)bslConv(e1,e2,sf),
       fm=function(e1,e2,sf)mlConv(e1,e2,sf),
       inE = inEs,
       outE = outE,
       outEb = outEb,
       attrs = attributes(df));
}

# convolution with detector resolution
detResConvMat <- function(eOutb,relErrAt1MeV=0.05){
  el <- head(eOutb,-1);
  eu <- tail(eOutb,-1);
  em <- (el+eu)/2.0;
  w <- sqrt(em)*relErrAt1MeV;
  outer(seq_along(el),seq_along(el), function(i,j)(pnorm(eu[i],em[j],w[j])-pnorm(el[i]
}

# like sumRowGroupsMat, but properly accounting for splitting of bins.
# used to decrease resolution of output bins to match desired outputs.
# assumes bins that split have uniform density, which isn't a terrible approximation
# since before splitting the bins are really small.
rebinDRMMat <- function(eInb,eOutb){
  eInbl <- head(eInb,-1);
  eInbu <- tail(eInb,-1);
  eOutbl <- head(eOutb,-1);
  eOutbu <- tail(eOutb,-1);
  outer(seq_along(eOutbl),seq_along(eInbl),
function(i,j)pmax(0,(pmin(eInbu[j],eOutbu[i])-pmax(eInbl[j],eOutbl[i]))/(eInbu[j]-eInb
}

makeDRM <- function(dCr,spect,inEBins,outEBins,relErrAt1MeV=0.05){
  elow <- head(inEBins,-1);
  ehigh <- tail(inEBins,-1);
  norm <- pi*(dCr$attrs$rDisk1**2-dCr$attrs$rDisk0**2)*100^2/dCr$attrs$nPriPerE
  mat <- mapply(dCr$f,elow,ehigh,MoreArgs=list(spect));
  rebinMat <- rebinDRMMat(dCr$outEb,outEBins);
  if(relErrAt1MeV>0){
      resConvMat <- detResConvMat(dCr$outEb,relErrAt1MeV);
      norm * rebinMat %*% resConvMat %*% mat;
  }else{
      norm * rebinMat %*% mat;
  }
}

applyDrmConvolution_makeDF <- function(dCr,e1,e2,spect){
  data.frame(cts=dCr$f(e1,e2,spect), outE=dCr$outE, inE=e1);
}

drmDiffPlot <- function(d1,d2){
  diff <- data.frame(outE=d1$outE,dc=d1$cts-d2$cts,type="delta(cts)");
  diffsig <- data.frame(outE=d1$outE,dc=(d1$cts-d2$cts)/sqrt(d1$cts+1),type="delta(cts
  diff <- rbind(diff,diffsig);

  range <- c(0.01,1.1*max(d1$outE[d1$cts>0],d2$outE[d2$cts>0],na.rm=TRUE));
  #range[1] <- range[2]-0.1;
  p <- ggplot() + theme_bw();
  p <- p + geom_line(data=diff,aes(x=outE,y=dc),col='black');
  p <- p + scale_x_log10(limits=range);
  p <- p + facet_wrap(~ type, ncol=1,scale = "free_y");
  p <- p + xlab("Energy deposited (MeV)");
  p <- p + ylab("Difference in cts");
  p;
```

```
}

testInterpBG <- function(df,e){
  ntrp <- interpolateDRMbg(df,e);
  e <- ntrp$e;
  dfs <- subset(df,df$inE==e);
  p1 <- compareDRMs(dfs,interpDRMtoDF(ntrp));
  p2 <- drmDiffPlot(dfs,interpDRMtoDF(ntrp));
  multiplot(p1,p2,cols=1);
  print(e);
}

testInterp <- function(df,e){
  eIn <- unique(df$inE);
  ei <- findInterval(e,eIn);
  e1 <- eIn[ei-1];
  e2 <- eIn[ei+1];
  e <- eIn[ei];
  print(c(e1,e2,e));
  ntrp <- interpolateDRMs_givenE(df,e1,e2,e);
  dfs <- subset(df,df$inE==e);
  p1 <- compareDRMs(dfs,ntrp);
  p2 <- drmDiffPlot(dfs,ntrp);
  multiplot(p1,p2,cols=1);
  print(e);
}

# for example, running R in the directory with the simulation results files:
# makeDRM_batch(fns=list.files("resultsDir",pattern="^mats_22_.*.txt",include.dirs=FALS
#               nPriPerE=500000,
#               rDisk=0.6,
#               bins=10**seq(-1,2,length.out=40),
#               primarySpect=function(e){1/e})
makeDRM_batch <- function(fns,nPriPerE,rDisk1=0.6,rDisk0=0.0,bins,primarySpect,det='bg
  for(fn in fns){
    print("****************************");
    print(sprintf("working on %s...",fn));
    print("reading simulation results...");
    a <- readDRMs_df(fn,combineOutBins=2,nPriPerE=nPriPerE,rDisk1=rDisk1,rDisk0=rDisk0
    print("making DRM convolver...");
    f <- drmConvolver(a);
    print("making DRM...");
    drm <- makeDRM(f,primarySpect,bins,bins);

    print("writing output...");
    dn = dirname(fn);
    fn = basename(fn);
    outDRMFn <- sprintf("%s/drm_%s",dn,fn);
    write.table(drm,outDRMFn,row.names=FALSE,col.names=FALSE)

    print("plotting...");
    plotFn <- sprintf("%s/drmPlot_logScale500Max_%s.pdf",dn,fn);
    pdf(plotFn,width=8,height=8);
    plotDRM(bins,bins,drm,log='xy',zlim=c(0,500),main=fn);
    dev.off();
    plotFn <- sprintf("%s/drmPlot_logScale_%s.pdf",dn,fn);
    pdf(plotFn,width=8,height=8);
    plotDRM(bins,bins,drm,log='xy',main=fn);
    dev.off();
    plotFn <- sprintf("%s/drmPlot_linScale_%s.pdf",dn,fn);
    pdf(plotFn,width=8,height=8);
    plotDRM(bins,bins,drm,main=fn);
    dev.off();

    print("done!");
  }
}


# needs some additional setup to be run in the REPL...
# library(runParallel)
# cl <- makeCluster(4) # number of CPUs
# registerDoParallel(cl)
# then makeDRM_batch_parallel(...)
makeDRM_batch_parallel <- function(fns,nPriPerE,rDisk1=0.6,rDisk0=0.0,bins,primarySpec
    foreach(fn=fns) %dopar% {
source("../procDRM.r");
makeDRM_batch(c(fn),nPriPerE,rDisk1,rDisk0,bins,primarySpect,det)
    }
}
```

```
# note dropping 0-20 keV bin since it's not relevant to include all the way down to ze
# note also division by 1000 to convert to MeV.
rhessiBins <- c(20.0, 25.0, 30.0, 35.0, 40.0, 45.0, 50.0, 60.0, 70.0, 80.0, 100.0,
120.0, 140.0, 160.0, 200.0, 250.0, 300.0, 350.0, 400.0, 500.0, 600.0,
700.0, 800.0, 1000.0, 1200.0, 1400.0, 1700.0, 2000.0, 2500.0, 3000.0,
3500.0, 4000.0, 5000.0, 6000.0, 7000.0, 9000.0, 11000.0, 13000.0,
15000.0, 17500.0, 19999.0, 23000.0, 30000.0, 40000.0) / 1000.0;

logBins <- 10**seq(-2,2,length.out=41);
```



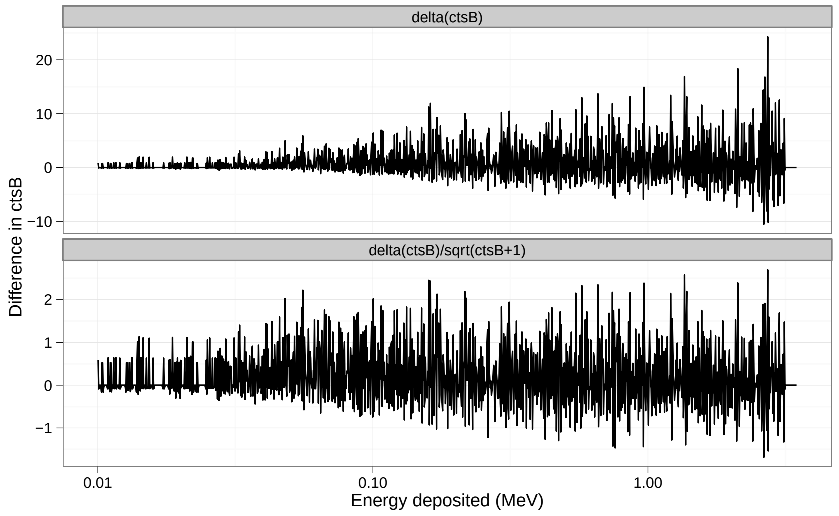


Figure 4: Top panel: Energy deposition histogram for 3.16 MeV photons at normal incidence on the BGO detector (black), compared to its smoothed counterpart (blue). The bottom panels show the results of subtracting the smoothed histogram from the original, both in absolute counts and in standard deviations.

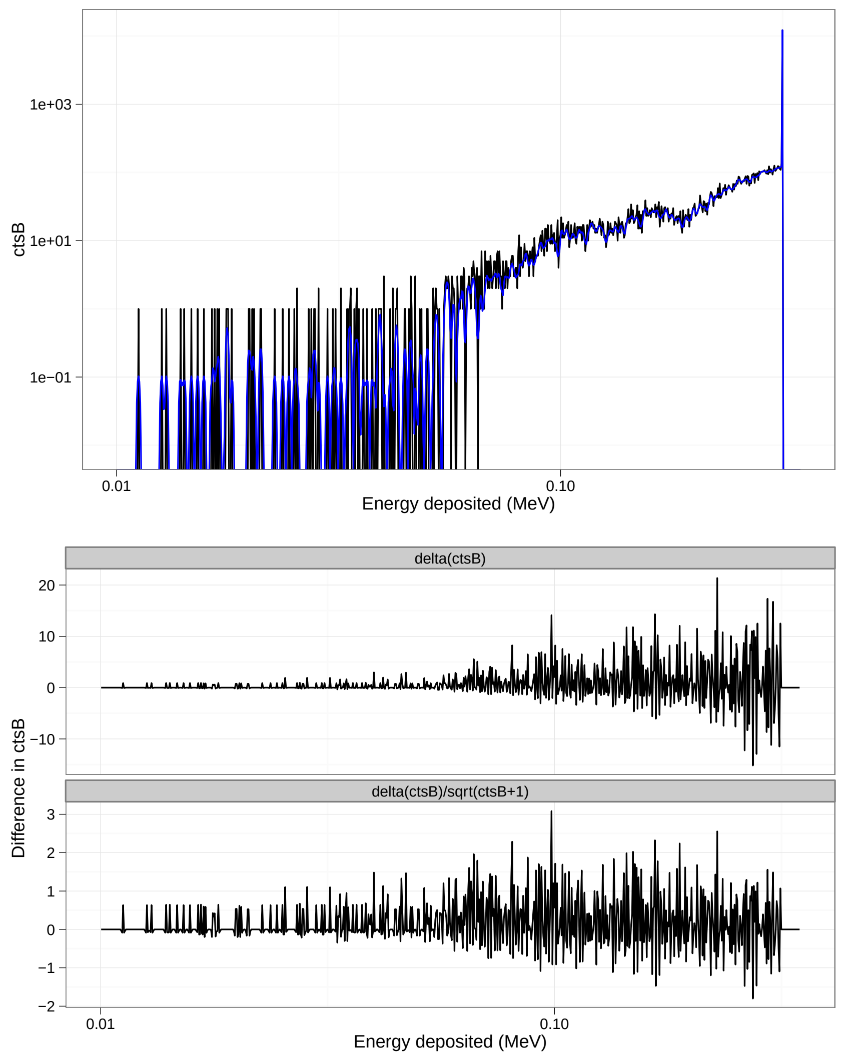
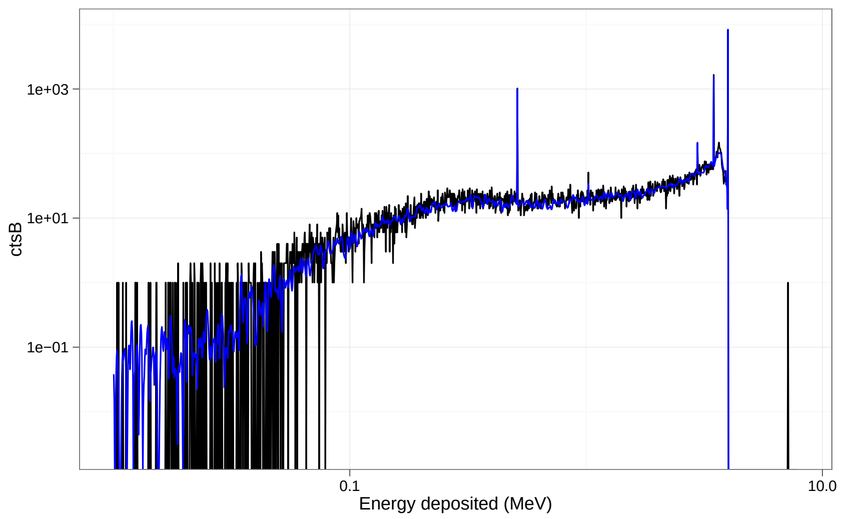


Figure 5: Top panel: Energy deposition histogram for 0.32 MeV photons at normal incidence on the BGO detector (black), compared to its smoothed counterpart (blue). The bottom panels show the results of subtracting the smoothed histogram from the original, both in absolute counts and in standard deviations.



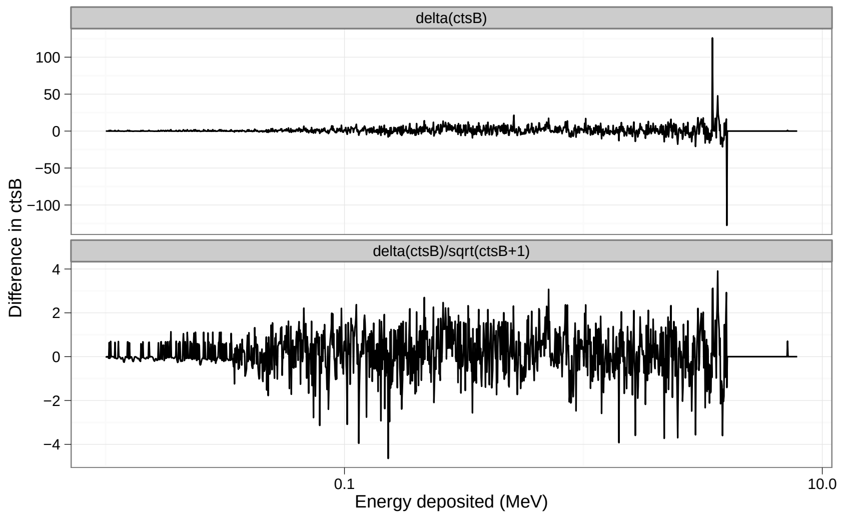


Figure 6: Top panel: Energy deposition histogram for 3.98 MeV photons at normal incidence on the BGO detector (black), compared to an interpolation to the histogram for 3.98 MeV photons based on histograms for 3.16 MeV and 5.01 MeV primary photons (blue). The bottom panels show the results of subtracting the interpolated histogram from the original, both in absolute counts and in standard deviations. Note the occasional energy deposition event with more energy than the primary. The additional energy likely comes from neutron capture or decay of an excited state nucleus, but such events are rare enough to be disregarded.

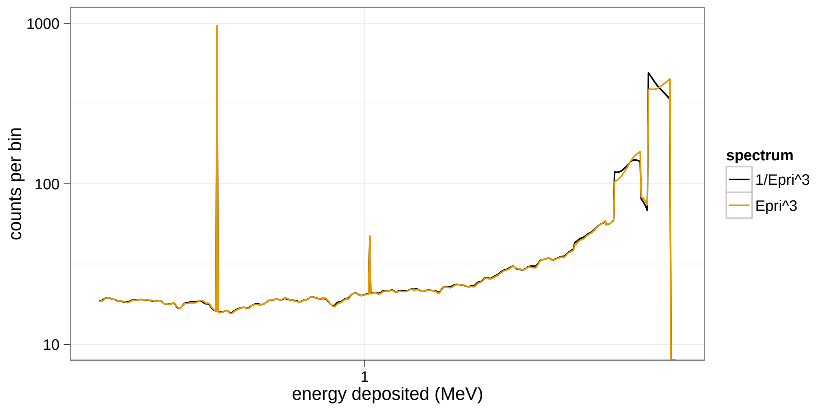


Figure 7: Sample convolution of two input spectra with detector response interpolations. The two input spectra are taken to be only nonzero for 3.6 MeV $< E^{\mathrm{pri}} <$ 4 MeV, with one $\propto 1/E^{\mathrm{pri}\,3}$ and $\propto E^{\mathrm{pri}\,3}$ as labeled.

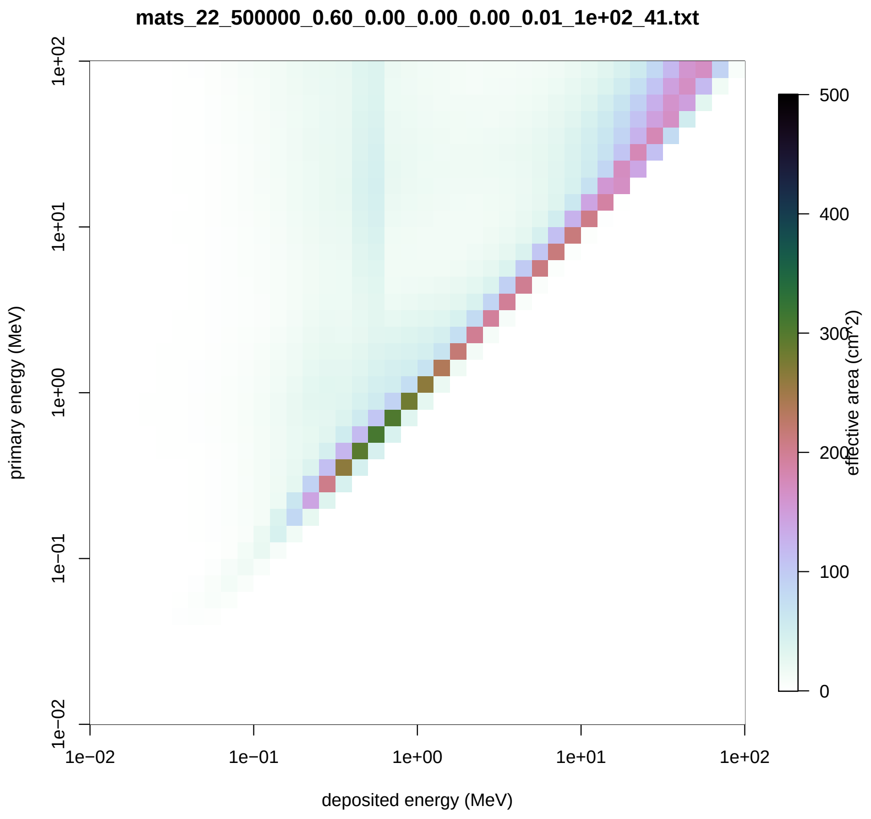


Figure 8: Sample DRM as calculated for normal incidence photons on the BGO detector with a $1/E^{\mathrm{pri}}$ spectrum. The color indicates the effective area for counts in the given deposited energy bin for primaries in the given primary energy bin in cm^2. Scanning across the plot at a particular primary energy gives the shape of the deposited energy spectrum for the given primary energy.

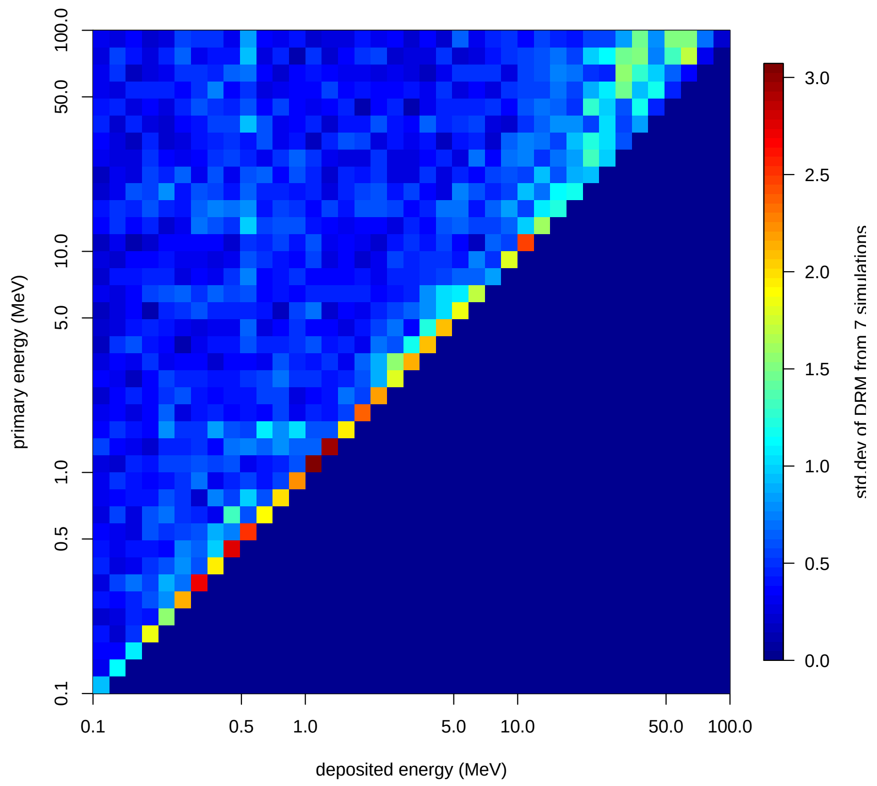


Figure 9: Standard deviation of DRM elements over 7 identical simulations. This is a representation of the statistical error in a single DRM, here executed with 5×10^5 primary particles per primary energy.

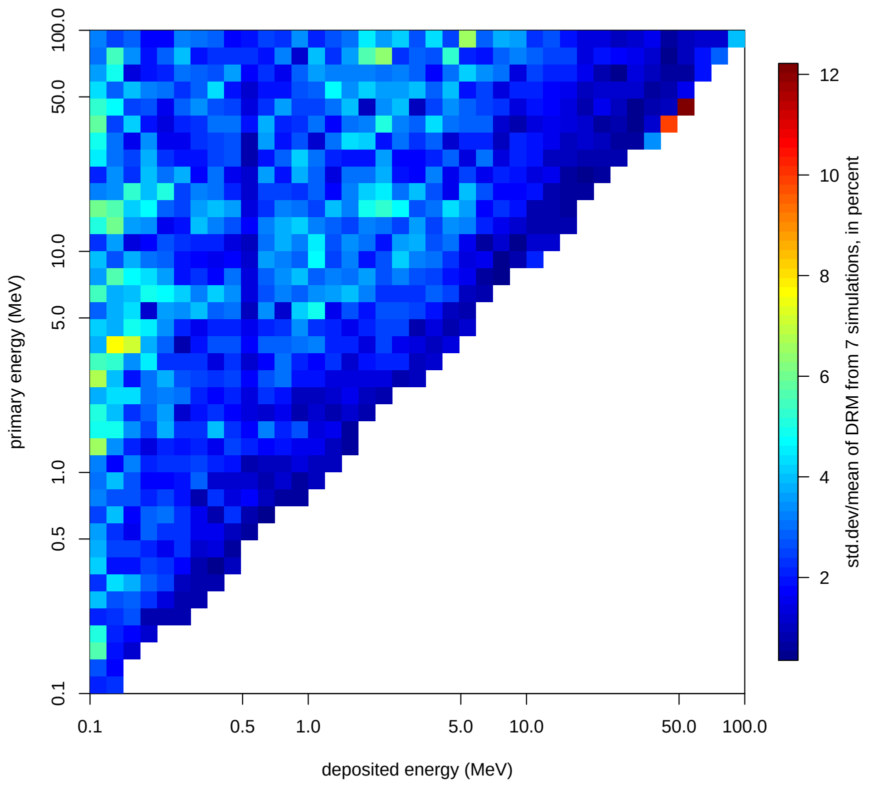


Figure 10: Like Figure 9, but showing standard deviation over mean of DRM elements, expressed as a percent. This is a representation of the statistical error in a single DRM relative to the DRM itself.

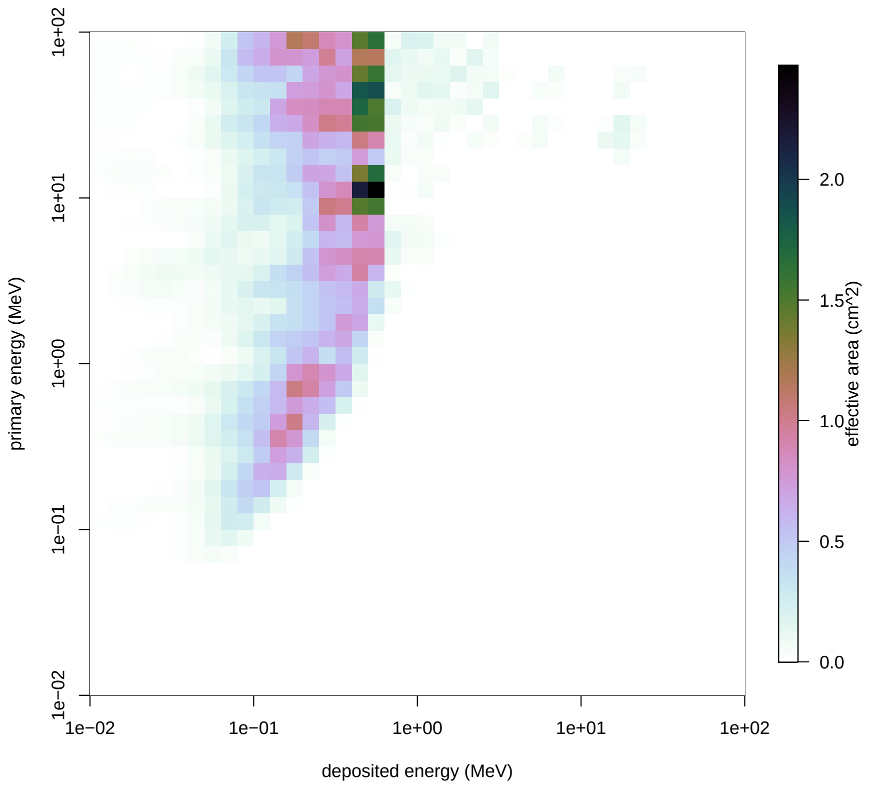


Figure 11: The detector response matrix but only for particles incident that would not directly hit MXGS. This plot essentially shows an estimate of the magnitude of the underestimate of the detector response matrix calculated only including particles that would have directly hit MXGS. The simulations shown here had incident particles that passed MXGS and illuminated the end of the Columbus module closest to MXGS, i.e. theta=45, phi=-90.

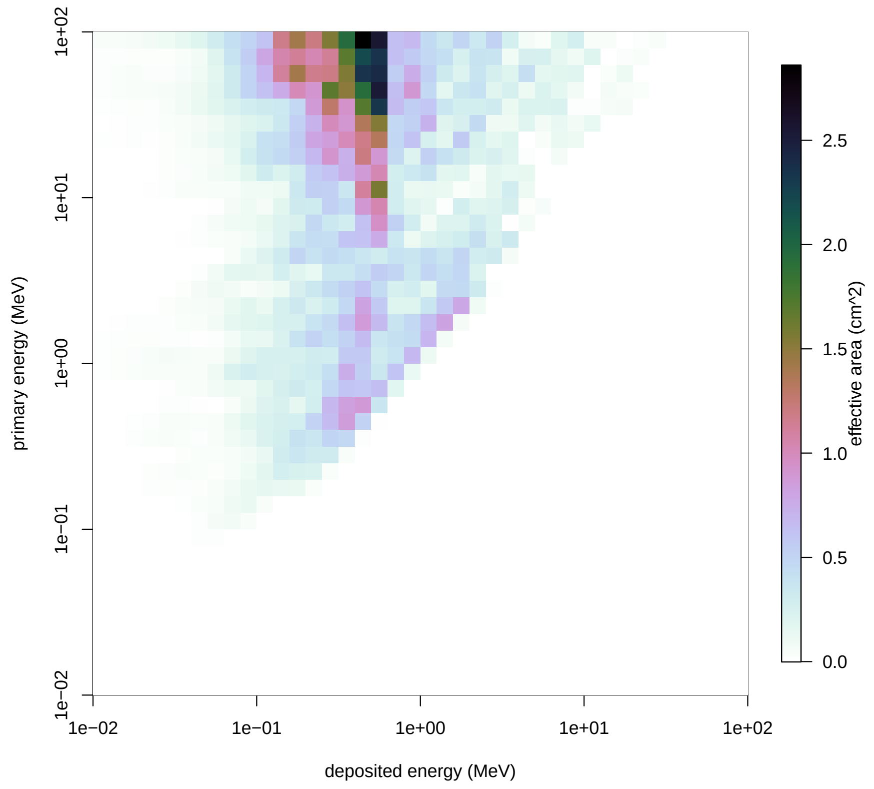


Figure 12: Like Figure 11 only with particles directed so as to pass through ACES on their way to MXGS, i.e. theta=45, phi=90.