SAND2021-11839

SANDIA REPORT
SAND2021-11839
Printed September 2021

Sandia National Laboratories

AXIOM Unfold 0.7.0, Users Manual

Gregg A. Radtke

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico
87185 and Livermore,
California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	orders@osti.gov
Online Ordering:	http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce National Technical Information Service
5301 Shawnee Rd.
Alexandria, VA 22312

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.gov
Online Ordering:	http://classic.ntis.gov/help/order-methods

ABSTRACT

The AXIOM-Unfold application is a computational code for performing spectral unfolds along with uncertainty quantification of the photon spectrum. While this code was principally designed for spectral unfolds on the Saturn source, it is also relevant to other radiation sources such as Pithon. This code is a component of the AXIOM project which was undertaken in order to measure the time-resolved spectrum of the Saturn source; to support this, the AXIOM-Unfold code is able to process time-dependent dose measurements in order to obtain a time-resolved spectrum.

This manual contains a full description of the algorithms used by the method. The code features are fully documented along with several worked examples.

ACKNOWLEDGMENTS

The author acknowledges the entire AXIOM team for the discussions and feedback to help with forming the input and output formats, and determining the features to prioritize in the code development process. More specifically, he acknowledges Ben Ulmen and Tim Webb for helping to understand the experimentally-measured dose data, and for Russ Depriest and Aaron Olson for generating the needed response functions and the falloff functions. The author also acknowledges Wes Fan for the many discussions about radiation physics and unfold methods in general which were helpful in the development of the method and its implementation.

Contents

1 Introduction **10**

1.1 Spectral unfold . . . 10
1.2 Photon spectrum . . . 10
1.3 Calculated dose . . . 11
1.4 Average photon energy . . . 12
1.5 Sampling dose uncertainty . . . 13

2 Spectrum unfolds **14**

2.1 Parametric spectrum unfold . . . 14
2.2 Yogi unfold method . . . 15
2.3 Nonlinear optimization . . . 16
2.4 Time-resolved spectral unfolds . . . 17

3 Software implementation **18**

3.1 Running AXIOM-Unfold . . . 18
3.2 Input deck . . . 18
3.2.1 Shot information . . . 21
3.2.2 Spectrometer data . . . 21
3.2.3 Falloff . . . 23
3.2.4 Locations of interest . . . 24
3.2.5 Unfold . . . 25
3.2.6 Numerics . . . 27
3.2.7 Output . . . 29
3.2.8 Uncertainty quantification . . . 30
3.3 Output files . . . 30

4 Example spectrum unfolds **31**

4.1 Basic unfold . . . 31
4.2 Yogi unfold . . . 37
4.3 Unfold with uncertainty . . . 40
4.4 Time-dependent unfold . . . 48

5 Conclusion **54**

List of Figures

3.1 Overall structure of input deck. 19
3.2 Location of interest input block. 24

4.1 Input YAML file for the basic example. 32
4.2 Log file for the basic example. 33
4.3 Spectrum plot for the basic example. 36
4.4 Dose statistics plot for the basic example. 36
4.5 Input YAML file for the Yogi example. 37
4.6 Spectrum plot for the Yogi example. 39
4.7 Input YAML file for the UQ example. 41
4.8 Spectrum uncertainty plot for the UQ example. 46
4.9 Dose statistics plot for the UQ example. 47
4.10 Input YAML file for the time-dependent example. 49
4.11 Spectrum plot for sixth time bin of the time-dependent example. 52
4.12 Dose statistics plot for the sixth time bin of the time-dependent example. 53

List of Tables

3.1 Required python packages for AXIOM-Unfold at the time of writing. 18
3.2 Input parameter data types. 20
3.3 Units used in the code. 21
3.4 Shot information input parameters. 21
3.5 Spectrometer data input parameters. 21
3.6 Falloff input parameters 23
3.7 Locations of interest input parameters. 24
3.8 Unfold input parameters. 25
3.9 Numerics input parameters. 27
3.10 Output input parameters. 29
3.11 Uncertainty quantification input parameters. 30
3.12 Output files 30

4.1 Dose statistics data file for the basic example. 34
4.2 Spectrum parameters data file for the basic example. 34
4.3 Unfolded spectra data file for the basic example. 35
4.4 Optimization objectives data file for the basic example. 35

4.5 Spectrum parameters data file for the Yogi example. 38
4.6 Unfolded spectra data file for the Yogi example. 38
4.7 Dose samples data file for the UQ example. 42
4.8 Computed doses data file for the UQ example. 42
4.9 The dose statistics data file for the UQ example. 43
4.10 The spectrum parameters data file for the UQ example. 44
4.11 The spectrum parameter statistics data file for the UQ example. 44
4.12 The unfolded spectra data file for the UQ example. 44
4.13 The spectrum statistics data file for the UQ example. 45
4.14 The dose statistics data file for the time-dependent example. 50
4.15 The spectrum parameters data file for the time-dependent example. 51
4.16 Unfolded spectra data file for the time-dependent example. 51

This page intentionally left blank.

Nomenclature

Variable	Definition
C	Logarithmic spectrum curvature
D	Radiation dose [Gy]
E	Photon energy [MeV]
L	Number of detectors in the spectrometer
M	Number of spectral bins in the binned spectrum
n	Power law exponent in the Bremsstrahlung production factor
r	Distance from source [cm]
R	Response function [Gy/(photon cm^2)]
S	Normalized photon spectrum [MeV^{-1}]
α	Line intensity coefficient
β	Branching ratio
δ	Dirac delta function
Θ	Heaviside step function
Λ	Logarithmic curvature penalty
μ	Direction cosine between the detector surface normal and source line of sight
ϕ	Photon flux [photons/cm^2]
Φ	Photon flux distribution [photons/cm^2/Mev]
$\mathbf{\Omega}$	Direction from the source to the dose detector with respect to axis

1 Introduction

This chapter provides the definition of many concepts relevant to performing spectral unfolds, including: the definition of a spectral unfold, the photon spectrum, measured and calculated doses, and average photon energy. In addition, we describe a process for generating dose samples from uncertainty data measurements for producing spectral unfolds with uncertainty. This content provides the necessary background for understanding the full description of the actual unfold techniques described in Chapter 2

1.1 Spectral unfold

A spectral unfold method is a method for using dose data from multiple dose detectors with different radiation filters to obtain an estimate of the radiation spectrum. These radiation filters consist of layers of different materials which reduce different portions of the incoming spectrum, thereby making each detector sensitive to different regions of the spectrum. A spectrometer is a device consisting of multiple dose detectors used to obtain the dose measurements to use in the unfold.

The legacy method for performing spectrum unfolds on Saturn is the Yogi method [1]. The Yogi method is a non-parametric method that requires trial spectrum that is very close to the true spectrum, perturbs this spectrum in an iterated manner with a penalty on the logarithmic curvature of the spectrum. In contrast for the AXIOM project, we use a parametric model originally developed for the bremsstrahlung radiation source Pithon [2]. For the AXIOM-Unfold code, we have implemented both the parametric and Yogi unfold methods, where the Yogi method was generalized to use more powerful global optimization methods in order to use the same optimization framework for both methods. We have observed poor performance of the Yogi method and therefore plan to deprecate it in the near future.

In the remaining sections in this chapter, we discuss the nomenclature and mathematical relationship between the spectrum and dose measurements of relevance to the spectral unfold process, as well as the procedure used for sampling doses from experimental measurements. These sections will provide the context for the later chapters discussing the parametric and generalized Yogi spectrum unfold methods, as well as the numerical optimization methods used. The later chapters discuss the implementation including usage instructions, followed by several worked examples followed by a conclusion.

1.2 Photon spectrum

The photon flux spectrum $\Phi(r, E, \mathbf{\Omega})$ is defined as the photon number flux per energy E, where r is the radial distance from the source and $\mathbf{\Omega}$ is the solid angle with respect to the source axis. The flux spectrum can be separated into a product of two terms:

$$\Phi(r, E, \mathbf{\Omega})\phi(r, \mathbf{\Omega})\, S(r, E, \mathbf{\Omega}), \tag{1.1}$$

where $\phi(r, \mathbf{\Omega})$ is the total photon flux or fluence, $S(r, E, \mathbf{\Omega})$ is the normalized photon spectrum, and the normalization condition for $S(r, E, \mathbf{\Omega})$ requires

$$\int_0^\infty \mathrm{d}E\; S(r, E, \mathbf{\Omega}) = 1. \tag{1.2}$$

To use the unfold techniques described in this document, we require the very significant assumption $S(r, E, \mathbf{\Omega})$ does not vary significantly with position $(r, \mathbf{\Omega})$ for the region in which the spectrometer, dose detectors, and device under test are fielded. This spectrum uniformity assumption is required in order to ensure that: (i) the spectrometer itself is experiencing a relatively uniform field for all detectors, and (ii) that the normalized spectrum unfolded from data obtained at the spectrometer remains an accurate representation at the other dose detectors and device under test.

The uniformity assumption allows the photon flux spectrum to be split into a purely positional dependent fluence and purely energy dependent normalized spectrum via

$$\Phi(r, E, \mathbf{\Omega}) \approx \phi(r, \mathbf{\Omega})\, S(E). \tag{1.3}$$

The validity of this assumption is difficult to assess in general, and we adopt this approach simply because there is presently no reasonable way to avoid it. Nevertheless, future work in assessing this assumption and quantifying the uncertainty in propagates into the unfold process is warranted.

While parametric unfold techniques use the the continuous form of $S(E)$ directly, Yogi unfolds require a binned approximation obtained by averaging over defined energy bins. Here we define M energy bins with edges $\{E_m\}_{m=0}^{M}$, which are monotonically increasing. The energy bin widths $\boldsymbol{\Delta E}$ and binned spectrum averages $\overline{\boldsymbol{S}}$ are given (for $m = 1, 2, \ldots, M$) by

$$\Delta E_m = E_m - E_{m-1} \tag{1.4}$$

$$\overline{S}_m = \frac{1}{\Delta E_m} \int_{E_{m-1}}^{E_m} \mathrm{d}E\, S(E), \tag{1.5}$$

respectively. For the binned spectrum, the normalization condition of the spectrum reduces to

$$\sum_{m=1}^{M} \Delta E_m \overline{S}_m = 1. \tag{1.6}$$

1.3 Calculated dose

For a given photon flux distribution, the doses $\boldsymbol{D}$ for the detectors are related by

$$D_\ell = \int_0^\infty dE\; R_\ell(E, \mu)\, \Phi(r, E, \mathbf{\Omega}) \approx \phi(r, \mathbf{\Omega}) \int_0^\infty dE\; R_\ell(E, \mu)\, S(E), \tag{1.7}$$

where the uniformity assumption (Equation 1.3) was applied in the second expression. Here, the response functions $R_\ell(E, \mu)$ are defined as the fractional dose per energy per fluence, $\ell = 1, 2, \ldots, L$ is the index for a particular detector in a spectrometer, and μ is a direction cosine for the angle between the surface normal of the detector and the line of sight from the detector to the source.

For all present applications, we assume that the spectrometer is fielded so that all detectors are fielded normal to the source which removes the μ dependency, resulting in $R_\ell(E, \mu) \approx R_\ell(E)$. Under this assumption, the functions $R_\ell(E)$ are independent of how the spectrometer is fielded. These response functions are obtained with high numerical precision using a radiation transport code and stored for use in unfolds. While general μ dependence would render the unfold process less tractable, a small angle approximation $R_\ell(E, \mu) \approx \mu R_\ell(E)$ could be used in some situations so long as μ is known.

In order to make the fluence more tractable, we also assume that the r and Ω dependence is separable, i.e.

$$\phi(r, \mathbf{\Omega}) \approx \phi_0 f(r) g(\mathbf{\Omega}). \tag{1.8}$$

Similar to what was noted in the justification for the spectral uniformity assumption (Equation 1.3), we make this assumption out of necessity to make the unfold tractable, and further investigation is warranted. Here $\phi_0 = \phi(r_0, \mathbf{\Omega}_0)$ is the photon fluence incident on the spectrometer, where $(r_0, \mathbf{\Omega}_0)$ are the coordinates to the spectrometer face centroid, $f(r)$ is the radial falloff function, and $g(\mathbf{\Omega})$ is a function characterizing the angular variation.

The falloff function is a monotonically decreasing function which is normalized to the spectrometer face via $f(r_0) = 1$. Many models for $f(r)$ are possible depending on the radiation source, including planar

$f(r) = 1$ and far field $f(r) = (r_0/r)^2$. In some cases, more complicated expressions for $f(r)$ can be obtained by simulating the radiation source.

The angular dependence term is assumed to be $g(\mathbf{\Omega}) = 1$ for all present applications, which is a further assumption of source uniformity; in this case, a uniform fluence with respect to $\mathbf{\Omega}$. Other assumptions may include a separable form $g(\mathbf{\Omega}) \approx g_0(\omega) g_1(\eta)$ where the polar $g_0(\omega)$ and azimuthal $g_1(\eta)$ dependencies are decoupled. In any case, $g(\mathbf{\Omega})$ is required to be normalized to the spectrometer face, i.e. $g(\mathbf{\Omega}_0) = 1$.

For convenience, the net effect of the radial and angular photon flux dependence is included into a modified version $R^*_\ell(R)$ of the response functions

$$R^*_\ell(E) = f(r_\ell) g(\mathbf{\Omega}_\ell) R_\ell(E, \mu_\ell), \tag{1.9}$$

where $(r_\ell, \mathbf{\Omega}_\ell)$ are the coordinates of the ℓ^{th} detector, and μ_ℓ is the direction cosine of the ℓ^{th} detector face. For a spectrometer fielded with all detectors normal to the line of sight, the modified response function is $R^*_\ell(E) = f(r) R_\ell(E|1)$, where $R_\ell(E|1)$ are the response functions for normal incidence. Further assuming an r-squared falloff model results in $R^*_\ell(E) = (r_0/r_\ell)^2 R_\ell(E|1)$.

Using the modified response functions, Equation 2.24 can be rewritten in the following form.

$$D_\ell = \phi_0 \int_0^\infty dE \; R^*_\ell(E) \, S(E) \tag{1.10}$$

For the the binned spectrum this reduces to a matrix equation

$$\boldsymbol{D} = \phi_0 \, \boldsymbol{\Delta R}^* \, \overline{\boldsymbol{S}}, \tag{1.11}$$

where $\boldsymbol{\Delta R}^*$ is a matrix of integrated response functions with elements given by

$$\Delta R^*_{\ell,m} = \int_{E_{m-1}}^{E_m} dE \; R^*_\ell(E). \tag{1.12}$$

1.4 Average photon energy

The average photon energy is obtained as the first moment of the normalized spectrum.

$$\overline{E} = \int dE \, E \, S(E) \tag{1.13}$$

For the binned spectrum, this can be approximated using the binned average photon energies $\overline{E}_m = \frac{1}{2}(E_{m-1} + E_m)$.

$$\overline{E} = \sum_{m=1}^{M} \int_{E_{m-1}}^{E_m} dE \, E \, S(E) \approx \sum_{m=1}^{M} \overline{E}_m \int_{E_{m-1}}^{E_m} dE \, S(E) = \sum_{m=1}^{M} \Delta E_m \overline{E}_m \overline{S}_m \tag{1.14}$$

Note that the above expressions give the average energy per photon. Another useful quantity is the average energy weighted by its contribution to the total integrated energy. This is accomplished by taking the first moment of the energy distribution $E \, S(E)$ and normalizing since $\int dE \, E \, S(E) = \overline{E} \neq 1$.

$$\overline{E}^\dagger = \frac{1}{\overline{E}} \int dE \, E^2 \, S(E) \tag{1.15}$$

The binned version of $\overline{E}^\dagger$ is approximated by

$$\overline{E}^\dagger = \frac{1}{\overline{E}} \sum_{m=1}^{M} \int_{E_{m-1}}^{E_m} dE \, E^2 \, S(E) \approx \frac{1}{\overline{E}} \sum_{m=1}^{M} \overline{E^2}_m \int_{E_{m-1}}^{E_m} dE \, S(E) = \frac{1}{\overline{E}} \sum_{m=1}^{M} \Delta E_m \overline{E^2}_m \overline{S}_m, \tag{1.16}$$

where the average squared energy is given by $\overline{E^2}_m = \frac{1}{2}\left(E^2_{m-1} + E^2_m\right)$. For consistency of notation in the remainder of this manual, we will discuss only the average energy per photon $\overline{E}$, and the user wishing to extract $\overline{E}^\dagger$ data from code output can readily do so from code output

1.5 Sampling dose uncertainty

The next chapter describes the parametric and Yogi unfold methods which numerically invert these relationships (1.10–1.11) between dose and spectrum using dose measurements. In practice, each dose may be quantified from multiple measurements, each with uncertainty estimates. To model this uncertainty, we extend our notation for the doses D_ℓ^n, where $n = 1, 2, \ldots, N_\ell$ and $N_\ell \geq 1$ is the number of measurements for each detector. Likewise, the uncertainty for each measurement is given by the estimated standard deviation $\sigma[D_\ell^n]$. Below, we develop our procedures for performing unfolds on a single representative sample or a dual sampling approach used to propagate the uncertainty for a large number of unfolds with different dose samples.

For a single representative sample $D_\ell(0)$, we use a weighted mean as shown below.

$$D_\ell(0) = \sum_n W_n\, D_\ell^n \tag{1.17}$$

Here we use the standard approach to weight the measurements by the inverse variance $\sigma^{-2}[D_\ell^{n_\ell}]$, which gives the normalized weights as

$$W_n = \frac{\sigma^{-2}[D_\ell^n]}{\sum\limits_n \sigma^{-2}[D_\ell^n]}. \tag{1.18}$$

For propagating the uncertainty through to the unfolded spectrum, we perform multiple unfolds on doses which sample the underlying uncertainty in the dose measurements. To do this we use a dual sampling approach to produce samples $D_\ell(s)$ for $s = 1, 2, \ldots, S$ where S is the total number of samples. Dual samples are obtained by first making a random choice of the discrete distribution of measured doses D_ℓ^n, and then sampling from its associated uncertainty distribution. For each sample, we randomly choose a measurement n' using probabilities W_n. Next, we obtain dose sample s by sampling from a normal distribution with mean $D_\ell^{n'}$ and variance $\sigma^2[\hat{D}_\ell^{n'}]$, that is by sampling:

$$D_\ell(s) \sim \mathcal{N}\left[D_\ell^{n'}, \sigma^2[D_\ell^{n'}]\right], \text{where } n' \text{ is sampled with probabilities } W_n \tag{1.19}$$

By performing a spectral unfold for each dose sample, we can effectively propagate the influence of dose measurement uncertainty into the spectrum estimate.

2 Spectrum unfolds

This chapter presents the two implemented unfold methodologies, where the parametric spectrum is the prevalent method and the Yogi is the legacy method. Both methods are described in detail, as well as a brief discussion of the numerical optimization methods used and how to do time-resolved unfolds.

2.1 Parametric spectrum unfold

The parametric unfold method [2] uses a physics-based model of the spectrum with a small number of fit parameters. This model $S(E, \boldsymbol{\theta})$ is a function of energy and parameterized by a vector of fit parameters $\boldsymbol{\theta}$ and has the following form:

$$S(E, \boldsymbol{\theta}) = \frac{S_0}{E}\, S_B(E, a_1, a_2, n, E_{max})\, S_A(E, b, E_{cut})\, S_K(E, \alpha, \kappa), \tag{2.1}$$

where the vector of fit parameters is $\boldsymbol{\theta} = [a_1, a_2, \alpha, b, E_{cut}, E_{max}, \kappa]$ and $S_0(\boldsymbol{\theta})$ is a normalization constant which is determined by enforcing that the photon spectrum integrates to unity (see Equation 1.2).

$$\frac{1}{S_0} = \int_0^\infty \frac{dE}{E}\, S_B(E, a_1, a_2, n, E_{max})\, S_A(E, b, E_{cut})\, S_K(E, \alpha, \kappa) \tag{2.2}$$

The three factors in the parametric photon spectrum are described below.

The S_B factor in Equation 2.1 models Bremsstrahlung production based on the classical logarithmic cross section for electron energy E_{max} with correction coefficients a_1 and a_2 and a power law exponent n to account for the distribution of initial and straggled electron energies, which is generally assumed to be $n = 2$.

$$S_B(E, a_1, a_2, n, E_{max}) = \xi_{max}^n (1 + a_1 \xi_{max} + a_2 \xi_{max}^2), \text{ where } \xi_{max} = 1 - E/E_{max} \tag{2.3}$$

The S_A factor models photon absorption in the converter and vacuum window based on photo-absorption cross section with characteristic (e-folding) cutoff energy E_{cut} and a first-order correction coefficient b.

$$S_A(E, b, E_{cut}) = \exp\big[-\xi_{cut}^{1.6}(\xi_{cut} + b)\big], \text{ where } \xi_{cut} = E_{cut}/E \tag{2.4}$$

Finally, the S_K factor models the K_α and K_β lines for a tantalum converter with energies $E_\alpha = 56.9$ keV and $E_\beta = 65.2$ keV, and the absorption above the K-edge at $E_K = 67.4$ keV.

$$S_K(E, \alpha, \kappa) = \Big\{1 + \alpha\big[\delta(E - E_\alpha) + \beta\delta(E - E_\beta)\big]\Big\} \exp\left[-\kappa \left(\frac{E_K}{E}\right)^{2.6} \Theta(E - E_K)\right] \tag{2.5}$$

Here α is the line intensity coefficient, $\beta = 0.25$ is the branching ratio, and κ is a fit parameter related to the converter thickness.

Several fitting constraints are enforced for the parametric spectrum model in order to keep the fits out of non-physical regimes. First, the parameters α, b, and κ are required to be non-negative.

$$\alpha \geq 0 \tag{2.6}$$
$$b \geq 0 \tag{2.7}$$
$$\kappa \geq 0 \tag{2.8}$$

Second, the polynomial in the S_B term (Equation 2.3) is required to be positive in order to prevent the spectrum from having negative values. This results in two additional constraints, shown below.

$$1 + a_1 + a_2 > 0 \tag{2.9}$$

$$4a_2 - a_1^2 > 0 \text{ when } a_1\, a_2 < 0 \tag{2.10}$$

The parametric photon spectrum is obtained by minimizing the objective function

$$G(\boldsymbol{\theta}) = G_0 \sum_{\ell=1}^{L} \left(\frac{\hat{D}_\ell - D_\ell}{\sigma\left[D_\ell\right]} \right)^2, \tag{2.11}$$

to obtain the optimal parameters $\boldsymbol{\theta}$, where $\sigma\left[D_\ell\right]$ are the estimated uncertainties (standard deviations) of the measured doses D_ℓ. The normalization constant G_0 is defined as

$$G_0 = \left[\sum_{\ell=1}^{L} \left(\sigma\left[D_\ell\right]\right)^{-2} \right]^{-1}. \tag{2.12}$$

Using this normalization, the value of the objective function is the weighted average squared difference between the fitted and measured doses. This minimization is performed using global nonlinear optimization methods as described in Section 2.3.

When computing the dose integrals in Equation 2.11, the fluence and the parametric spectrum normalization constant are lumped into a single parameter $\phi_0 S_0$. Removing this parameter from the calculated doses, yields

$$\hat{D}_\ell^*(\boldsymbol{\theta}) = \frac{\hat{D}_\ell}{\phi_0\, S_0} = \int_0^\infty \frac{dE}{E} S_B(E, a_1, a_2, n, E_{max})\, S_A(E, b, E_{cut})\, S_K(E, \alpha, \kappa) R_\ell^*(E). \tag{2.13}$$

Then $\phi_0\, S_0$ is calculated as the value which minimizes Equation 2.11 for fixed $\boldsymbol{\theta}$, resulting in

$$\phi_0\, S_0 = \sum_{\ell=1}^{L} D_\ell \Bigg/ \sum_{\ell=1}^{L} \hat{D}_\ell^*. \tag{2.14}$$

The actual doses are then obtained as $\hat{D}_\ell = \phi_0\, S_0\, \hat{D}_\ell^*$.

After a global optimum $\boldsymbol{\theta}$ is obtained, S_0 is calculated directly from Equation 2.2, which allows the normalized photon spectrum to be fully specified. The fluence ϕ_0 may also be estimated from Equation 2.14. If the fluence ϕ' is desired from a dosimeter placed in an alternate location, this can be obtained directly from Equation 2.24.

$$\phi' = D' \Bigg/ \int_0^\infty dE\; S(E) R'(E, \mu') \tag{2.15}$$

Note that we are using the assumption that the normalized spectrum is identical at the spectrometer and the alternate location. Here, D' is the measured dose, and $R'(E, \mu')$ is the response function for the dosimeter.

2.2 Yogi unfold method

The Yogi method [1] was previously used for all spectral unfolds on Saturn. This method was known to be extraordinarily sensitive to the trial spectrum, and without a very good guess, the method would fail to find a physically reasonable spectrum. Moreover, the trial spectrum used for the bremsstrahlung diode was oversimplified and would fail spectacularly for off-normal shots where a pulsed power issue prevented the diode from performing as intended; for these type of shots, it is impossible to obtain a close enough trial spectrum.

The original Yogi method was an iterative method implemented in an IDL script, which produced an unfolded spectrum that generally did not vary far from the trial spectrum. We generalized it to use global optimizers in order to use the same optimization framework for both the parametric and Yogi unfolds, but this approach showed only minor improvement. The key difference between the Yogi and parametric unfolds is that the Yogi does not assume a functional form for the spectrum, but rather uses a binned spectrum model and uses the binned spectrum $\overline{S}$ as the fit parameters determined by optimization.

The objective function for the generalized Yogi method is shown below

$$G(\overline{\boldsymbol{S}}) = G_0 \sum_{\ell=1}^{L} \left(\frac{\hat{D}_\ell - D_\ell}{\sigma\left[D_\ell\right]} \right)^2 + \Lambda \sum_{m=1}^{M-1} C_m^2, \tag{2.16}$$

where G_0 is defined in Equation 2.12, $\boldsymbol{C}$ are the local second derivatives of the logarithmic spectrum with respect to the logarithmic energy

$$C_m = \frac{\log_{10} \Delta S_{m-1} - 2\log_{10} \Delta S_m + \log_{10} \Delta S_{m+1}}{[\log_{10} E_m - \log_{10} E_{m-1}]^2}, \tag{2.17}$$

and Λ is a (non-negative) regularization penalty.

The generalized Yogi spectrum is obtained by minimizing the dose error penalized by the logarithmic curvature.

$$G(\overline{\boldsymbol{S}}) = G_0 \sum_{\ell=1}^{L} \left(\frac{\hat{D}_\ell(\overline{\boldsymbol{S}}) - D_\ell}{\sigma\left[D_\ell\right]} \right)^2 + \Lambda\, \mathcal{C}_2^2(\overline{\boldsymbol{S}}) \tag{2.18}$$

Here G_0 is defined in Equation 2.12 and Λ is a positive scalar penalty. This minimization is performed using global nonlinear optimization methods as described in Section 2.3.

In a similar manner to Equation 2.19, a reduced form of the calculated dose is defined as

$$\hat{\boldsymbol{D}}^* = \boldsymbol{\Delta R}^* \, \overline{\boldsymbol{S}}^*, \tag{2.19}$$

where $\overline{\boldsymbol{S}}^* = \overline{\boldsymbol{S}}/S_0$ is an initial (un-normalized) guess. The value of $\phi_0 S_0$ is then obtained from

$$\phi_0 S_0 = \sum_{\ell=1}^{L} D_\ell \Bigg/ \sum_{\ell=1}^{L} \hat{D}_\ell^*, \tag{2.20}$$

from which the doses are calculated as $\hat{D}_\ell = \phi_0 S_0 \, \hat{D}_\ell^*$.

For the global optimum $\overline{\boldsymbol{S}}^*$, the spectrum is normalized using 1.6 to obtain S_0 as

$$\frac{1}{S_0} = \sum_{m-1}^{M} (E_m - E_{m-1}) \overline{S}_m^*. \tag{2.21}$$

As in the parametric case, ϕ_0 is calculated using Equation 2.20 and the fluence at an alternate location is computed from the binned version of Equation 2.15

$$\phi' = D' \Bigg/ \sum_{m-1}^{M} (E_m - E_{m-1}) \overline{S}_m \Delta R'_m(\mu'), \tag{2.22}$$

where

$$\Delta R'_m(\mu') = \int_{E_{m-1}}^{E_m} dE \; R'(E, \mu'). \tag{2.23}$$

Again, this is performed under the assumption that the spectrum is identical at the DAS and this alternate location.

2.3 Nonlinear optimization

Nonlinear optimization is used to minimize objective functions in Equations 2.11 and 2.18 using standard python packages from scipy.optimize. This optimization is performed in two steps consisting of a global optimizer followed by a final local optimizer that achieves a higher precision result based on the output of the global optimizer. We use simulated annealing as the global optimizer with a Nelder-Mead local optimizer, and this is generally the best approach due to the high number of dimensions. In either case, this is followed by a final application of the Nelder-Mead optimizer to high precision. Other options exist and are available to the user, including a brute-force global optimizer with various sampling approaches for the evaluation points. However, the default method is generally a good approach and recommended for most applications.

2.4 Time-resolved spectral unfolds

A time-resolved spectral unfold capability is obtained by performing multiple unfolds using time binned dose rate data. The dose rate equivalent of the Equation 2.24 is shown below

$$\dot{D}_\ell(t) \approx \dot{\phi}(r, \mathbf{\Omega}, t) \int_0^\infty dE\; R_\ell(E, \mu)\, S(E, t), \tag{2.24}$$

where now the dose rate $\dot{D}$, fluence rate $\dot{\phi}$, and instantaneous spectrum S are now functions of time.

We introduce time bins

$$t_q = t_0 + \sum_{q'=1}^{q} \Delta t_q, \tag{2.25}$$

where t_0 is a reference time at the beginning of the radiation pulse and Δt are the sizes of the bins. The variable $q = 1, 2, \ldots, Q$ indexes the time value at the tail end of each bin, and Q is the total number of time bins.

A time-binned version of Equation 2.24 is obtained by integrating over each time bin and approximating $S(E, t) \approx \overline{S}_q(E)$ over the time bin.

$$\Delta D_{\ell,q} \approx \Delta\phi_q(r, \mathbf{\Omega}) \int_0^\infty dE\; R_\ell(E, \mu)\, \overline{S}_q(E), \tag{2.26}$$

Note that this is identical to Equation 2.24 except that the variables (D_ℓ, ϕ, S) are replaced by ($\Delta D_{\ell,q}$, $\Delta\phi_q$, $\overline{S}_q$). Thus, we use identical unfold procedures for this data to obtain the normalized spectrum $\overline{S}_q$.

The binned fluence estimate resulting from the unfold for each time bin is given by $\Delta\phi_{0,q}$, where the fluence is given by

$$\phi_0 = \sum_{q=1}^{Q} \Delta\phi_{0,q}. \tag{2.27}$$

An estimate of the binned fluence rate is obtained from $\dot{\phi}_{0,q} \approx \Delta\phi_{0,q} / \Delta t_q$.

3 Software implementation

This chapter discusses the python implementation of the AXIOM-Unfold, including description of the input and output. The AXIOM-Unfold code is written in python3 and requires the packages listed in Table 3.1. The version numbers listed here are those recommended at the time this document was published, although we tend to update this list frequently according to the latest versions readily available in the Anaconda package manager.

Table 3.1: Required python packages for AXIOM-Unfold at the time of writing.

package	version
python	3.9.2
numpy	1.20.2
scipy	1.6.2
pyyaml	5.4.1
pandas	1.2.4
mpi4py	3.0.3
matplotlib	3.3.4
openpyxl	3.0.7

3.1 Running AXIOM-Unfold

At the current state, the code is run from command line, but we will be looking to develop a GUI in the future. The *runAXIOM-Unfold.py* script within the repo is the main executable used to perform an analysis. For serial execution on a OSX or Linux terminal, this is performed using

```
python <path to executable>/runAXIOM-Unfold.py <input>.yaml
```

while for MPI operation, this is performed using

```
mpiexec -np <number of processes> python -m mpi4py
 <path to executable>/runAXIOM-Unfold.py <input>.yaml
```

Running in a Windows environment will be similar, but requires a backward (\) rather than a forward (/) slash within file paths. Note that the MPI command line instructions above actually consist of a single line and are only broken into two lines for readability.

3.2 Input deck

The basic structure of the yaml input deck is shown in Figure 3.1. Specific examples of this format are shown in Chapter 4, and the input parameters for each input block (indicated within brackets) are shown in the following sections. Note that the structure of the *locations of interest* parameters are handled differently, as there are an arbitrary number of locations of interest, each indicated by a different label (indicated in angle brackets), and each will have an independent set of parameters.

During execution, progress towards completion will be saved in the log file *AXIOM-Unfold.log*, and total execution time will be displayed after finishing successfully. An *output* folder will be created, and all output files will be saved there. The output consists of pickled pandas DataFrames (and text equivalents), as well as plots (both .png and .pdf format) as well as a python script to regenerate each plot. The various input parameter options affecting output are documented in Section 3.2.7, while the output files for several example analyses is discussed in Chapter 4.

```
shot information:
  [shot information parameters]

spectrometer data:
  [spectrometer data parameters]

falloff:
  [falloff parameters]

locations of interest:
  <location of interest label 1>:
    [parameters for location of interest 1]
  <location of interest label 2>:
    [parameters for location of interest 2]
  ...

unfold:
  [unfold parameters]

numerics:
  [numerics parameters]

output:
  [output parameters]

 uncertainty quantification:
   [uncertainty quantification parameters]
```

Figure 3.1: Overall structure of input deck.

The input parameter data types consist of built-in python types (such as str, int, etc.) or more complicated types derived from the built-in ones. These derived data types consist of restricted versions of the built-ins, or combinations of them. An example of a more restricted type is u_int, which accepts only unsigned integer values; here the key difference is not in how the code stores the data (the code stores the value as a standard int), but in the error handling for the input parser which will raise the appropriate exception when negative integers are input for the u_int type. An example of a derived type formed as a combination of the built-in types is the str_dict which is a string-valued is a python dictionary[1].

The full set of data types for the input parameters is listed in Table 3.2. For the dict types, the key values are typically checked for consistency in the parser error handling. For the array-valued parameters, the value can be any type that converts to a numpy array such as a list[2]. Alternately, any data type except the list and dict types can be input as a str which evaluates to the appropriate data type using built-in python or numpy (imported as np) methods. For example, a u_real parameter can be specified as np.log10(1000) which is equivalent to 3.0. All code input uses the same units for dose, energy, and length. Derived quantities in the code output includes the units for many variables, but all others are consistent with those listed in Table 3.3 below.

Table 3.2: Input parameter data types.

data type	description
str	Built-in string data type
bool	Built-in Boolean data type which takes *True* or *False* values
int	Built-in integer data type
float	Built-in float data type
real	Can be either int or float data type
dict	Built-in dictionary data type
u_int	Unsigned version of int
u_real	Unsigned version of real
str_list	String-valued list
str_dict	String-valued dict
bool_dict	Boolean-valued dict
real_dict	Real-valued dict
u_real_array	Unsigned real-valued numpy array
dose_data	A double-nested list of unsigned real numbers
spec_data	A triple-nested list of unsigned real numbers
str_nested_list	A doubly-nested list of strings

[1]The python dict consists of key-value pairs and is indicated by curly braces, i.e. $\{key_1 : value_1, key_2 : value_2, \ldots\}$
[2]Python lists are indicated by square brackets, i.e. $[value_1, value_2, \ldots]$

Table 3.3: Units used in the code.

quantity	units
dose D	Gy
energy E	MeV
length r	cm
normalized spectrum S	1/MeV
fluence ϕ	#/cm^2

3.2.1 Shot information

The shot information category includes parameters to describe a specific shot for a radiation source. These parameters are used only for the titles in the plots and do not affect the analysis and are listed in Table 3.4 and described in the itemized list below.

Table 3.4: Shot information input parameters.

parameter	type	default
machine name	str	*no default*
shot number	u_int	*no default*

machine name Name of the radiation source (e.g. Saturn, Pithon, etc.).

shot number Shot number for the given radiation source.

3.2.2 Spectrometer data

The spectrometer data category includes the dose data for the shot to be used in the spectral unfold, which are either inputted directly or read from a TLD report. These parameters are listed in Table 3.5 and described below. The primary purpose of this category is to specify the spectrometer used and the measured dose data used to perform the unfolds, where the spectrometer detectors must be consistent with the dose data.

Table 3.5: Spectrometer data input parameters.

parameter	type	default
spectrometers: selection	str	spectrometers
spectrometers: file path	str	*see description*
spectrometers: spectrometers and fielding distances	u_real_dict	*see description*
spectrometers: detectors and fielding distances	u_real_dict	*see description*
measurements: source	str	*no default*
measurements: doses	spec_data	*see description*
measurements: uncertainties	spec_data	*see description*
measurements: TLD reports: file names	str_nested_list	*see description*
measurements: TLD reports: file path	str	*see description*
measurements: TLD reports: skiprows	u_int	21
measurements: TLD reports: redundancy	u_int	4
measurements: TLD reports: usecols	str_list	['Sequence', 'Dose', '%UncTot']
measurements: TLD reports: droprows	str_list	['BKG', 'REF']

spectrometers: selection Method of selection for the spectrometer, with valid options *spectrometers* and *detectors*. In the first option, the user specifies a list of one or more complete spectrometers, and in the second option specifies a list of detectors which can be assembled from different spectrometers.

spectrometers: file path File path for spectrometer and detector responses. This defaults to the *response_functions* folder in the AXIOM-Unfold repo, but this option allows the use of user defined spectrometers.

spectrometers: spectrometers and fielding distances Dictionary of spectrometers and fielding distances, where the spectrometer response functions must exist in the folder specified by *spectrometers: file path*.

spectrometers: detectors and fielding distances Dictionary of spectrometer/detector values and fielding distances. This option calls out individual detector response functions for spectrometers specified in *spectrometers: file path*.

measurements: source Source for the spectrometer data, where the valid options are *inline* and *TLD reports*. In the first option, *measurements: doses/uncertainties* must be specified. In the second option, *measurements: TLD reports: file names* must be specified.

measurements: doses A triple-nested list of measured doses, where the first dimension indexes time bin for time-resolved problems, the second dimension indexes the detector of the spectrometer, and the third dimension indexes the replicated dose measurements made for the same time bin and detector.

measurements: uncertainties Measured dose uncertainties (standard deviations) corresponding to *measurements: doses*, where the two structures must have the same size and shape

measurements: TLD reports: file names A double-nested list of TLD reports (EXCEL spreadsheets) file names for reading dose data, where the first dimension indexes the time bin and the second indexes the detector of the spectrometer. These files must exist in the directory specified in *measurements: TLD reports: file path*.

measurements: TLD reports: file path File path for TLD reports. This defaults to the *TLD_reports* folder in the AXIOM-Unfold repo.

measurements: TLD reports: skiprows Number of rows to skip in the header of the TLD reports.

measurements: TLD reports: redundancy Number of measurements per detector for reading the TLD reports. The reader will group the doses/uncertainties into groups of this size which are assumed to be replicated measurements of the same detector

measurements: TLD reports: usecols Column names to read from the TLD reports for the dose measurement number, dose value, and percent uncertainty.

measurements: TLD reports: droprows Rows with these values in the first column are dropped from the TLD reports.

3.2.3 Falloff

The falloff category includes parameters to specify the model for the radial falloff function $f(r)$ (see Section 1.3). These parameters are listed in Table 3.6 and described below. The variable r used in several of the parameters is the distance from the source in cm.

Table 3.6: Falloff input parameters

parameter	type	default
model	str	r-order
order	real	2
function	str	r**(-2)
file name	str	*see description*
file path	str	*see description*

model Name of the dose falloff model, where the options are a *tabular* file, *r-order* model, or a specified *function*. This defaults to *r-order*.

order Exponent in a *r-order* falloff model, which is equivilent to specifying the *order* in a falloff function of the form r^{-order}.

function Python evaluatable function of r for the *function* falloff model.

file name Filename for a *tabular* falloff model. This has no default and must be specified if the *tabular* is chosen.

file path File path where the *tabular* falloff file specified in *falloff: file name* exists.

3.2.4 Locations of interest

The optional locations of interest block consists of a list of labeled locations of interest, which is of the form shown in Figure 3.2, where the fluence is estimated either from a supplied dosimeter measurement using the dosimeter response function or by correcting for the fielding distance using the falloff function. In either case, the unfolded spectrum is used for the calculation. Each of the parameter sub-blocks has the input parameters as listed in Table 3.7 and as described below.

```
locations of interest:
  label 1:
    [parameters for locations of interest 1]
  label 2:
    [parameters for locations of interest 2]
  ...
```

Figure 3.2: Location of interest input block.

Table 3.7: Locations of interest input parameters.

parameter	type	default
specification	str	*no default*
time integrated	bool	True
dosimeter data:dosimeter	str	*see description*
dosimeter data:measurements:dose	dose_data	*see description*
dosimeter data:measurements:uncertainty	dose_data	*see description*
fielding distance	u_real	*see description*

specification How the location of interest is defined, with options 'dosimeter data' and 'fielding distance'.In the first case, the fluence is estimated based on a fielded dosimeter with the dose data provided from *dosimeter data:measurements:dose* and *dosimeter data:measurements:uncertainty*. In the second case, the fluence is estimated based on the spectrum unfold and corrected by the fielding distance using the falloff function.

time integrated Specify a time-integrated fluence estimate for this location of interest.

dosimeter data:dosimeter Dosimeter response function file in response_functions/dosimeters folder in the AXIOM-Unfold repo. Required for the *dosimeter data* specification option.

dosimeter data:measurements:dose Double-nested list of measured doses at the location of interest, where the first dimension indexes the time bin and the second dimension indexes the replicated dose mesurements made for the same time bin and detector. Under the *time integrated* option, the time bin dimension can only have a single entry. Required for the *dosimeter data* specification option.

dosimeter data:measurements:uncertainty Measured dose uncertainties (standard deviations) corresponding to *dosimeter data:measurements:dose*, where the two structures must have the same size and shape

fielding distance The fielding distance for the fluence estimate under the *fielding distance* specification option.

3.2.5 Unfold

The unfold category consists of the parameters needed to specify both how the unfolds are performed. While both parametric and Yogi unfold parameters are described below and listed in Table 3.8 and described below, the Yogi unfold method will be deprecated in a future version and not intended for widespread use.

Table 3.8: Unfold input parameters.

parameter	type	default
method	str	parametric
energy bounds	u_real_array	*no default*
dose weighting	str	variance
parametric: function	str	*see description*
parametric: kwargs	dict	dict()
parametric: variable parameters	bool_dict	*see description*
parametric: initial parameters	real_dict	*see description*
parametric: lower parameter bounds	real_dict	*see description*
parametric: upper parameter bounds	real_dict	*see description*
parametric: parameter log scale	bool_dict	*see description*
parametric: parameter units	str_dict	*see description*
parametric: parameter LaTeX	str_dict	*see description*
Yogi: initial spectrum: filenames	str_list	*see description*
Yogi: initial spectrum: bound multiplier	u_real	10
Yogi: initial spectrum: threshold	u_real	1e-18
Yogi: energy bins: number	u_int	32
Yogi: energy bins: log scale	bool	True
Yogi: binned spectrum: log scale	bool	True
Yogi: regularization: log scale	bool	True
Yogi: regularization: log base	u_real	10
Yogi: regularization: curvature: penalty	u_real	1e-06
Yogi: regularization: curvature: spectrum threshhold	u_real	0.0001
Yogi: regularization: curvature: norm	u_real	2
Yogi: regularization: curvature: suppress positive	bool	True

method Specification of the unfold method, with options are *parametric* and *Yogi*.

energy bounds Upper and lower energy bounds for performing the unfold, which should be chosen to be bound the domain of the spectrum as well as be within the domain for all response functions used.

dose weighting Weighting of the objective function, where the options are *variance* and *uncertainty*. For the second option, the weights are the inverse of the expanded uncertainty rather than the inverse variance.

parametric: function The parametric spectrum model function name. This function must exist in Unfold/Unfold/param_spectra.py and is a required argument for the *parametric* model

parametric: kwargs Keyword arguments to pass into the parametric spectrum model function.

parametric: variable parameters Dictionary to enable/disable the fit parameters of the parametric spectrum model. The default is obtained from the parametric function defined in Unfold/Unfold/param_spectra.py.

parametric: initial parameters Dictionary of initial values for the fit parameters for the parametric spectrum model. The defaults are obtained from the parametric function defined in Unfold/Unfold /param_spectra.py.

parametric: lower parameter bounds Dictionary of lower bounds for the fit parameters for the parametric spectrum model. The defaults are obtained from the parametric function defined in Unfold/Unfold/param_spectra.py.

parametric: upper parameter bounds Dictionary of upper bounds for the fit parameters for the parametric spectrum model. The defaults are obtained from the parametric function defined in Unfold/Unfold/param_spectra.py.

parametric: parameter log scale Dictionary to enable/disable logarithmic scaling for the fit parameters of the parametric spectrum model. The defaults are obtained from the parametric function defined in Unfold/Unfold/param_spectra.py.

parametric: parameter units Dictionary of the units for the fit parameters of the parametric spectrum model. The defaults are obtained from the parametric function defined in Unfold/Unfold/param_spectra.py.

parametric: parameter LaTeX Dictionary of the LaTeX labels for the fit parameters of the parametric spectrum model. The defaults are obtained from the parametric function defined in Unfold/Unfold/param_spectra.py.

Yogi: initial spectrum: filenames Initial spectrum filename for the Yogi spectrum model. Must be specified for Yogi unfolds.

Yogi: initial spectrum: bound multiplier The bounds on the spectrum as a fraction of the spectrum. The upper bound is the multiplier times the spectrum while the lower bound is 1/multiplier times the spectrum.

Yogi: initial spectrum: threshold Lower threshold of the initial spectrum as a fraction of the peak.

Yogi: energy bins: number Number of energy bins to used in the binned spectrum for Yogi unfolds.

Yogi: energy bins: log scale Switch to enable/disable log scaling of the Yogi energy bins.

Yogi: binned spectrum: log scale Switch to enable/disable log scaling of the Yogi binned spectrum

Yogi: regularization: log scale Switch to enable/disable log scaling in the binned Yogi spectrum for the regularization term.

Yogi: regularization: log base Logarithm base for the regularization term.

Yogi: regularization: curvature: penalty Penalty constant for the spectrum bin curvature in the Yogi regularization term.

Yogi: regularization: curvature: spectrum threshhold Spectrum threshold as a fraction of the peak value. The curvature is not calculated for parts of the spectrum lower than this threshold in order to avoid pathological behavior for the regularization term.

Yogi: regularization: curvature: norm The vector norm used to evaluate the regularization term from the local curvature estimates.

Yogi: regularization: curvature: suppress positive Switch to enable/disable suppression of solutions with positive curvature.

3.2.6 Numerics

The numerics category consists of the parameters which control the numerical interpolation, integration, and optimization. These numerical algorithms use standard methods from the python *scipy* package, where most keyword arguments for these methods are exposed to the user. These parameters are listed in Table 3.9 and described below.

Table 3.9: Numerics input parameters.

parameter	type	default
interpolator: kwargs	dict	{'kind': 'linear'}
integrator: method	str	quad
integrator: kwargs	dict	*see description*
global optimizer: enable	bool	True
global optimizer: method	str	basinhopping
global optimizer: kwargs	dict	{'niter': 50}
local minimizer: kwargs	dict	*see description*
final minimizer: enable	bool	True
final minimizer: kwargs	dict	*see description*
enforce constraints	bool	True
random number generator: seed	u_int	*see description*
expanded uncertainty: two-sided interval percentage	u_real	95
expanded uncertainty: number of samples	u_int	1000
expanded uncertainty: sampling method	str	LHS

interpolator: kwargs Keyword arguments passed to the *scipy.interpolate.interp1d* method for all iterpolations.

integrator: method Method in the *scipy.integrate* package used to perform numerical quadrature.

integrator: kwargs Keyword arguments passed to *scipy.integrate* method. Default value is {'epsrel': 1.e-4, 'limit': 10, 'full_output': 1}.

global optimizer: enable Switch to enable/disable the global optimizer.

global optimizer: method Method in the *scipy.optimize* package to perform global optimizations.

global optimizer: kwargs Keyword arguments passed to the *scipy.optimize* method.

local minimizer: kwargs Keyword arguments passed to the *scipy.optimize.minimize* function used for local minimization, where the default value is {'method': 'Nelder-Mead', 'tol': 1.e-3, 'options': {'maxiter': 50}}.

final minimizer: enable Switch to enable/disable final minimizer for the unfold.

final minimizer: kwargs Keyword arguments passed to the *scipy.optimize.minimize* function used for final minimization, where the default is {'method': 'Nelder-Mead', 'tol': 1.e-4, 'options': {'maxiter': 500}}

enforce constraints Switch to prevent/allow unfolds which violate fit parameter or spectral bin constraints.

random number generator: seed Specify a random number seed to achieve repeatability in the analysis. By default, this is initialized randomly from the system clock.

expanded uncertainty: two-sided interval percentage Percentage used for the calculation of two-sided uncertainty intervals.

expanded uncertainty: number of samples Number of samples used to compute the expanded uncertainty in the dose.

expanded uncertainty: sampling method Sampling method used to compute the expanded uncertainty in the dose, where the options are *LHS* and *random*.

3.2.7 Output

The output category consists of the parameters which control the output that the various forms of output that the code produces. These are shown in Table 3.10 and described below.

Table 3.10: Output input parameters.

parameter	type	default
plotting: enable	bool	True
logging: enable	bool	True
opt files: purge	bool	True
folder name	str	output
dose: log scale	bool	True
dose: bar width	u_real	0.8
spectrum: range	u_real	1000000
spectrum: log scale	bool	True
energy: log scale	bool	True
energy: points	u_real_array	*see description*
energy: delta sigma	u_real	0.0005

plotting: enable Switch to enable/disable plots in output.

logging: enable Swith to enable/disable runtime logging.

opt files: purge Switch to enable/disable purging of optimization results files.

folder name The name of the output folder.

dose: log scale Switch to enable/disable log scaling for dose plots.

dose: bar width Relative bar width for bar plots.

spectrum: range Range for spectrum plots (ratio between maximum and minimum).

spectrum: log scale Swtich enable/disable log scale spectrum on plots.

energy: log scale Switch enable/disable log scale for energy on plots.

energy: points The energy values used in spectrum plots.

energy: delta sigma Standard deviation (in energy) of the normal representation of the delta functions used for plotting the parametric unfolds.

3.2.8 Uncertainty quantification

The uncertainty quantification category controls the parameters used to enable running in uncertainty quantification mode, and controlling the dose sampling. These are shown in Table 3.11 and described below.

enable Switch to enable uncertainty quantification.

number of samples Number of samples used for uncertainty quantification.

sampling method Sampling method for uncertainty quantification, where the options are *LHS* and *random*.

Table 3.11: Uncertainty quantification input parameters.

parameter	type	default
enable	bool	False
number of samples	u_int	100
sampling method	str	LHS

3.3 Output files

The code produces a large amount of output files, where the number of files depends on the the number of time bins to be unfolded as well as whether uncertainty quantification is enabled. These files consists of both data files are plot files. The data files always consist in pairs with the same root name, but with extensions .pkl and .txt. The .pkl files are pickled versions of a *pandas* DataFrame, which is loaded using the *pandas.read_pickle()* method, while the .txt format is text version of the same data. The plot files always consist in triples with the same root name, but with extensions .png, .pdf, and .py. Here the .png and .pdf are common image formats, while the .py is a python script that can be run in its folder to reproduce the two image files. The purpose of the .py file is a to provide an easy way to modify the plots or produce subsequent plots as desired; these also demonstrate how to read the .pkl files using python.

Table 3.12: Output files

file	format	tagged	description
computed_doses	pkl/txt	no	Computed doses for all detectors and time bins.
dose_samples	pkl/txt	no	The sampled doses for all detectors and time bins.
dose_statistics	pkl/txt	no	Various statistics for measured, computed, and sampled doses.
dose_statistics	py/png/pdf	yes	Plot comparing various dose statistics.
optimization_objectives	pkl/txt	yes	Objective values for the initial, global, and final optimizations.
spectrum	py/png/pdf	yes	Plot of the unfolded spectrum (non-UQ only).
spectrum_uncertainty	py/png/pdf	yes	Plot of the unfolded spectrum uncertainty (UQ only).
spectrum_statistics	pkl/txt	no	Various statistics on the unfolded spectra (UQ only).
spectrum_parameter_statistics	pkl/txt	no	Various statistics of spectrum parameters (UQ only).
spectrum_parameters	pkl/txt	yes	Spectrum parameters obtained for all unfolds.
unfolded_spectra	pkl/txt	no	Tabular values for the unfolded spectra.

4 Example spectrum unfolds

In this chapter, we demonstrate how to use the AXIOM-Unfold code with several examples which span the common use cases relevant to most analysts; more complex We first show the basic example consisting of a single parametric unfold using shot data for the standard DAS spectrometer and the parametric Bremsstrahlung model, followed by a Yogi version of same shot. Next, we show an unfold with uncertainty of a Pithon shot using the standard DAS array spectrometer fielded at the Pithon facility. Finally, we show a time dependent example which was the first shot to field the AXIOM spectrometer which was fielded at the Idaho State University TriMeV facility.

4.1 Basic unfold

The input deck for the basic unfold example is shown in Figure 4.1, which has data taken from Saturn shot 4446. Here the *shot information* block is required for every analysis, and contains information that is mainly used for labels in the standard plots produced by AXIOM-Unfold. The remaining input blocks specify information more critical to the example.

The *spectrometer data* block defines the spectrometer (or spectrometers) used and the source of the dose data used to perform the unfold (or unfolds). Here, the standard DAS is indicated as *STDAS*, corresponds to a file, *response_functions/STDAS/STDAS.yaml*, which defines the dosimeters in the proper order that this spectrometer consists of. Notice that the fielding distance is specified by a python expression (here *np* is the python numpy package) instead of a numerical value, which is evaluated when the input is parsed and replaced with $\sqrt{160^2 + 20^2}$ cm; this is a convenient way to introduce a 160 cm vertical distance with a 20 cm horizontal offset into the code input without performing an offline calculation. The dose measurements are obtained from a single TLD report, where this file name also must exist in the repository in the *TLD_reports* folder. Note that the TLD report file names is a double-nested list, where the outer list allows data for various time periods, and each inner list allows for data for multiple spectrometers.

The *falloff* section indicates how falloff is estimated in the region containing the test device, as well as the spectrometers and dosimeters at locations of interest. Here, we indicate a tabular model from an ACCEPT simulation of Saturn, where this file is found in the *falloff_functions* in the repository.

The *unfold* section defines the type of unfold performed; for the parametric method, there is only a single option implemented at this time: the *L3_Brems_model*. Here we specify appropriate energy bounds to define the domain for the integrals used to define the relationship between dose and spectrum, which must encompass the effective range of the source spectrum for the technique to be valid.

The (optional) *locations of interest* indicate locations where the fluence is desired. Here a single location, indicated with label *spectrometer*, is defined with an identical fielding distance to the spectrometer which will yield the associated fluence value. Since dose information is not included, this is estimated using the spectrometer dose data and the falloff model.

The *output* block includes only a single specification, where we again use a python expression to yield an array of points over which the spectrum values will be output. Similarly, the *numerics* block specifies only the random number generator seed.

When running the AXIOM-Unfold application, a single log file is produced which is always named *AXIOM-Unfold.log*. The log file for this example is shown in Figure 4.2 below. The analyst is encouraged to look a the log file while the code runs to observe progress, and the total execution time will always be displayed at the end whenever the code finishes successfully. Note that the files saved to the *output* folder are indicated by their file names as they are saved.

```yaml
shot information:
  machine name: Saturn
  shot number: 4446

spectrometer data:
  spectrometers:
    selection: spectrometers
    spectrometers and fielding distances:
      STDAS: np.sqrt(160**2 + 20**2)
  measurements:
    source: TLD reports
    TLD reports:
      file names: [[529393_SATURN-4446-DAS.xlsx]]

falloff:
  model: tabular
  file name: Saturn-ACCEPT_falloff_tabular.dat

unfold:
  method: parametric
  parametric:
    function: L3_Brems_model
  energy bounds: [0.002, 5]

locations of interest:
  spectrometer:
    specification: fielding distance
    fielding distance: np.sqrt(160**2 + 20**2)

output:
  energy:
    points: 10 ** np.linspace(-3, 1, 2001)

numerics:
  random number generator:
    seed: 975412709
```

Figure 4.1: Input YAML file for the basic example.

```
2021-08-16 14:53:24,206  AXIOM-Unfold version 0.7.0
2021-08-16 14:53:24,206  running in MPI mode [size=1]
2021-08-16 14:53:24,206  parsed input parameters from Saturn-4446.yaml
2021-08-16 14:53:24,212  --------------------------------------------------------------------------
2021-08-16 14:53:24,212  creating dose samples...
2021-08-16 14:53:24,214  saved output/dose_samples.pkl
2021-08-16 14:53:24,221  saved output/dose_samples.txt
2021-08-16 14:53:25,982  saved output/dose_statistics.pkl
2021-08-16 14:53:25,988  saved output/dose_statistics.txt
2021-08-16 14:53:25,988  --------------------------------------------------------------------------
2021-08-16 14:53:26,025  initialized parametric spectrum unfold method
2021-08-16 14:53:26,025  performing unfolds...
2021-08-16 14:54:59,839  performed 1/1 unfolds
2021-08-16 14:54:59,839  --------------------------------------------------------------------------
2021-08-16 14:54:59,839  saving output...
2021-08-16 14:54:59,871  saved output/optimization_objectives.pkl
2021-08-16 14:54:59,873  saved output/optimization_objectives.txt
2021-08-16 14:54:59,878  saved output/computed_doses.pkl
2021-08-16 14:54:59,882  saved output/computed_doses.txt
2021-08-16 14:54:59,892  saved output/dose_statistics.pkl
2021-08-16 14:54:59,896  saved output/dose_statistics.txt
2021-08-16 14:54:59,905  saved output/spectrum_parameters.pkl
2021-08-16 14:54:59,909  saved output/spectrum_parameters.txt
2021-08-16 14:54:59,914  saved output/unfolded_spectra.pkl
2021-08-16 14:54:59,951  saved output/unfolded_spectra.txt
2021-08-16 14:54:59,954  saved output/dose_statistics.py
2021-08-16 14:54:59,962  saved output/dose_statistics.png
2021-08-16 14:54:59,962  saved output/dose_statistics.pdf
2021-08-16 14:54:59,965  saved output/spectrum.py
2021-08-16 14:54:59,972  saved output/spectrum.png
2021-08-16 14:54:59,973  saved output/spectrum.pdf
2021-08-16 14:54:59,973  --------------------------------------------------------------------------
2021-08-16 14:54:59,973  total execution time: 96.12s
```

Figure 4.2: Log file for the basic example.

A successful run of the AXIOM-Unfold code will also produce a number of data files and plots, which is always placed in the *output* folder. The data files will always occur in a pickled pandas DataFrame format with extension pkl; these can be opened using python via the read_pickle method in pandas. For those wishing to avoid using python, these data are also available in a human readable text format with extension txt. For all created plots, there are three files with the same name but with three different extensions: py, png, and pdf. The png and pdf files are the plot images themselves, while the python file is a script that can be run to reproduce these plots; this script can be modified by the analyst and rerun in order to modify the png and pdf files.

The basic example produces a total of six data files and three plots. The *dose_samples* and *computed_doses* are most useful when the code is run in uncertainty quantification mode, and will not be discussed for this example, but the remaining four are shown and discussed below.

The *dose_statistics* are shown in Table 4.1 and consists of the basic statistics for the measured, sample, and computed doses. Note that here (and elsewhere) the ellipses indicate rows or columns omitted from display in order make them fit on the page. The measured dose statistics repeat the input doses provided to the code for the analysis; we recommend that the analyst scan through these values to ensure that the code is parsing the input data appropriately. Here, the $\mp$ expanded uncertainties refer to the absolute difference between lower and upper 95% confidence intervals and the median value respectively, and the average expanded uncertainty is the average of the two.

The sampled dose value is simply the single representative dose values (equivalent to the measured mean) used to perform the single unfold. Likewise, the computed values are the those produced from the single unfold. Differences between the computed and measured values indicate that the unfolds don't fit the model well.

Table 4.1: Dose statistics data file for the basic example.

	type	spectrometers				
	device	STDAS				
	detector	1	2	3	...	13
source	statistic					
measured	mean	10.627	8.77225	6.3325	...	0.395475
	median	10.54	8.6985	6.3315	...	0.39895
	standard deviation	0.67085177	0.40716523	0.29568768	...	0.02083987
	- expanded uncertainty	1.0436045	0.81558186	0.5847553	...	0.037933793
	+ expanded uncertainty	1.4462284	0.80264777	0.5625598	...	0.039836615
	average expanded uncertainty	1.2449164	0.8091148	0.5736575	...	0.038885206
sampled	mean	10.627	8.77225	6.3325	...	0.395475
	median	10.627	8.77225	6.3325	...	0.395475
computed	mean	10.292773	9.074024	6.232047	...	0.38985312
	median	10.292773	9.074024	6.232047	...	0.38985312

The unfolded *spectrum_parameters* are shown in Table 4.2 and consist of all of the model parameters used in the unfold, the scale constant S_0, the average energy $\overline{E}$ for the spectrum (see Section 1.4), and the particle fluence at each location of interest.

Table 4.2: Spectrum parameters data file for the basic example.

type	model param					particle spectrum	spectrometer
param	a1	...	Emax	kappa	scale constant	average energy	fluence
units	unitless	...	MeV	unitless	1/MeV	MeV	#/cm2
sample							
1	0.0	...	1.5585746	4.7470305e-08	0.40322933	0.19839445	4.215327e+12

Next, the *unfolded_spectra* consists of a tabular representation of the normalized spectrum as a function of energy as shown in Table 4.3 Finally, the *optimization_objectives* in Table 4.4 are the values of Equation 2.11 obtained at various stages of the optimization, namely for the initial guess, after the global optimization, and after the final optimization. The objective values are in units of Gray square, and the square root of these values gives an indication of the average error in the discrepancy between the measured and computed doses.

Table 4.3: Unfolded spectra data file for the basic example.

sample E [MeV]	1
0.0010000	0.0
0.0010046	0.0
0.0010093	0.0
...	...
0.0990832	3.2676024
0.0995405	3.253962
0.1000000	3.2403283
0.1004616	3.2267017
0.1009253	3.2130833
...	...
9.9083194	0.0
9.9540542	0.0
10.0000000	0.0

Table 4.4: Optimization objectives data file for the basic example.

sample	initial	global	final
1	0.02437941	0.00096165953	0.0009181244

For this basic example, there are only two plots. The *spectrum* plot is simply a plot of the normalized spectrum as shown in Figure 4.3. The *dose_statistics* is a plot of the 95% confidence intervals for the measured doses compared to the computed doses, and is shown in Figure 4.4, which shows good agreement within the measurement uncertainties.

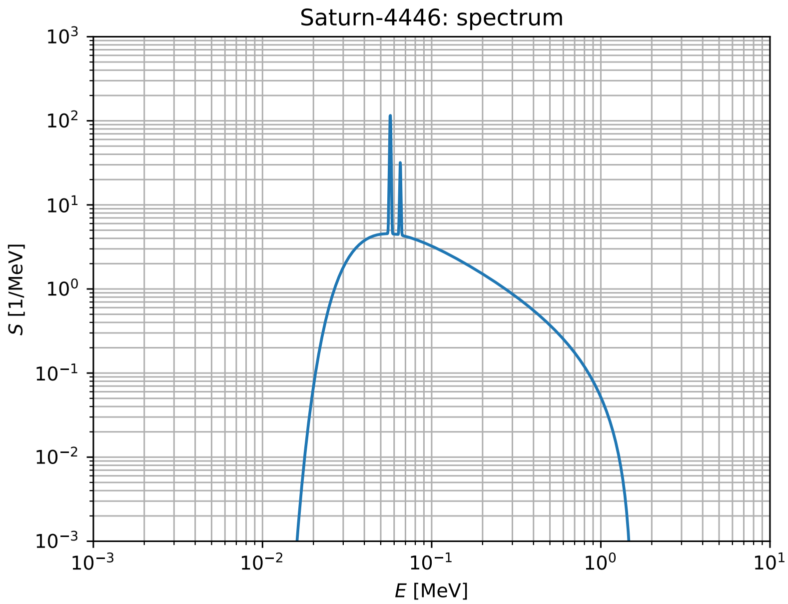

Figure 4.3: Spectrum plot for the basic example.

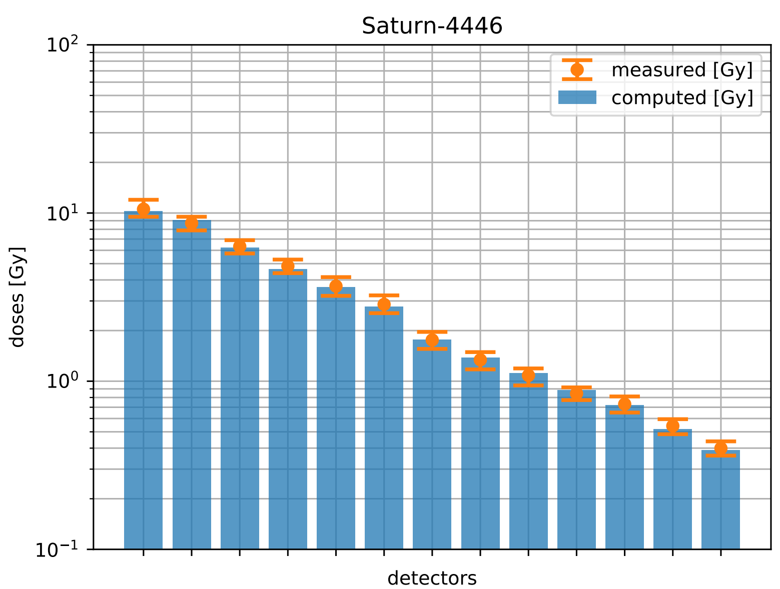

Figure 4.4: Dose statistics plot for the basic example.

4.2 Yogi unfold

The Yogi method is a legacy method which has been superseded by the parametric unfold. This example is included to document this capability until it is depreciated in the future, and uses the same Saturn shot as was shown for the basic example in Section 4.1. The input file for this example is shown in Figure 4.5, where the main difference is in the unfold block. Here, the Yogi method requires an initial guess for the spectrum which is a two-column text file; this can be from a parametric unfold if desired, and the format is identical to what is shown in Figure 4.3 with headings removed or commented out. The energy bounds must be chosen more carefully for the Yogi method, and these are narrower to avoid placing spectral bins outside the domain of the spectrum. We have also modified the basinhopping global optimizer by increasing the maximum number of iterations *niter* from the default of 50 to 1000, since the Yogi method is much computationally cheaper.

```yaml
shot information:
  machine name: Saturn
  shot number: 4446

spectrometer data:
  spectrometers:
    selection: spectrometers
    spectrometers and fielding distances:
      STDAS: np.sqrt(160**2 + 20**2)
  measurements:
    source: TLD reports
    TLD reports:
      file names: [[529393_SATURN-4446-DAS.xlsx]]

falloff:
  model: tabular
  file name: Saturn-ACCEPT_falloff_tabular.dat

unfold:
  method: Yogi
  Yogi:
    initial spectrum:
      filenames: [initial_spectrum.dat]
    energy bins:
      number: 32
    regularization:
      curvature:
        penalty: 1e-6
  energy bounds: [0.009, 2]

locations of interest:
  spectrometer:
    specification: fielding distance
    fielding distance: np.sqrt(160**2 + 20**2)

output:
  energy:
    points: 10 ** np.linspace(-3, 1, 2001)

numerics:
  global optimizer:
    method: basinhopping
    kwargs: {niter: 1000}
  random number generator:
    seed: 1427650775
```

Figure 4.5: Input YAML file for the Yogi example.

The data file and plot output contains all of the same files as the basic example, and the discussion below will focus on only the outputs that shown a substantial difference. First, the *spectrum_params* file shown in Table 4.5 has much fewer columns since it omits the spectrum parameters. Second, the binned spectrum is defined as piecewise constant, and therefore there are much fewer points included in the *unfolded_spectra* file in Table 4.6 and plotted in Figure 4.6.

Table 4.5: Spectrum parameters data file for the Yogi example.

type param units sample	particle spectrum average energy MeV	spectrometer fluence #/cm2
1	0.19815497	4.164362e+12

Table 4.6: Unfolded spectra data file for the Yogi example.

sample E [MeV]	1
0.0090000	0.0
0.0090000	8.3023866e-07
0.0107138	8.3023866e-07
0.0107138	0.0003292437
0.0127541	0.0003292437
0.0127541	0.015657237
...	...
0.0867719	5.1632915
0.0867719	4.371765
0.1032956	4.371765
0.1032956	3.5622478
0.1229659	3.5622478
0.1229659	2.897469
...	...
1.4113163	0.008199487
1.4113163	0.00023268048
1.6800692	0.00023268048
1.6800692	6.216485e-18
2.0000000	6.216485e-18
2.0000000	0.0

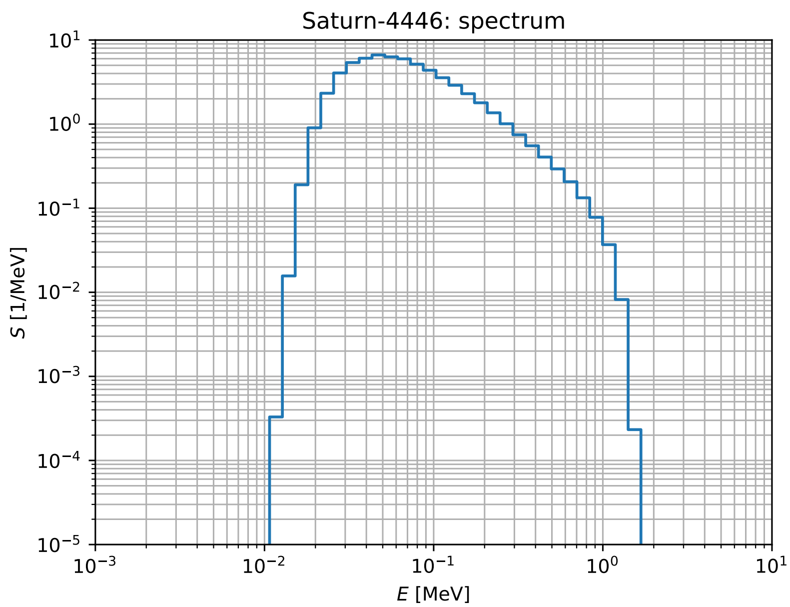

Figure 4.6: Spectrum plot for the Yogi example.

4.3 Unfold with uncertainty

The next example is an unfold of Pithon shot 7784, and primarily demonstrates a case where the uncertainty in the spectrum and fluence is estimated, but also demonstrates different options used for specification of the dose data and locations of interest as compared to the previous two examples. The input deck for this example is shown as Figure 4.7. Note that uncertainty quantification is enabled in the *uncertainty_quantification* block, where the total number of samples is also specified. Here, we are using the default of 100 samples, but the user can increase this for better fidelity if desired.

The spectrometer data measurements are also different here, as we specify inline data. For both doses and uncertainties the required input is a triply-nested list, where the inner list are the multiple measurements for a dose (a single measurement per dose for this example), the next level iterates over the various dosimeters in the spectrometer, and the outer list iterates over various time bins for time-resolved cases. Since this is modeled as a parallel source, we specify an r-order falloff model with exponent zero, which models the falloff as $\sim r^{-0} = 1$; the 1 cm fielding distance for both the spectrometer and location of interest then becomes irrelevant. The single location of interest consists of an aluminum-clad calcium fluoride TLD fielded at an alternate location, where the response function for the TLD exists in file *response_functions/dosimeters/TLD_35mil-CaF2_10mil-Al.dat*. The measurement dose and uncertainty data is similar to that for the spectrometer, except that it is only a doubly-nested list over multiple measurements (inner) and time bins (outer).

```yaml
shot information:
  machine name: Pithon
  shot number: 7784

spectrometer data:
  spectrometers:
    selection: spectrometers
    spectrometers and fielding distances:
      L3_Pithon: 1
  measurements:
    source: inline
    doses:
      [[[4.63], [4.925], [3.88], [1.87], [0.90], [0.34],
        [0.125], [0.02], [0.415], [0.62], [0.705], [0.945]]]
    uncertainties:
      [[[0.6945], [0.73875], [0.582], [0.2805], [0.135], [0.051],
        [0.01875], [0.003], [0.06225], [0.093], [0.10575], [0.14175]]]

falloff:
  model: r-order
  order: 0

unfold:
  method: parametric
  parametric:
    function: L3_Brems_model
    variable parameters:
      {a1: False, a2: False, alpha: True, b: False, Ecut: True, Emax: True, kappa: True}
    initial parameters:
      {a1: 0., a2: 0., alpha: 0.1, b: 0., Ecut: 0.02, Emax: 0.4, kappa: 0.1}
    lower parameter bounds:
      {a1: -1000., a2: -1000., alpha: 0., b: 0., Ecut: 0.005, Emax: 0.2, kappa: 0.}
    upper parameter bounds:
      {a1: 1000., a2: 1000., alpha: 0.1, b: 0.01, Ecut: 0.05, Emax: 0.6, kappa: 1.}
  energy bounds: [0.005, 0.8]

locations of interest:
  TLD:
    specification: dosimeter data
    dosimeter data:
      dosimeter: TLD_35mil-CaF2_10mil-Al
      measurements:
        dose: [[7.553]]
        uncertainty: [[1.133]]
  spectrometer:
    specification: fielding distance
    fielding distance: 1

numerics:
  random number generator:
    seed: 1255245163

uncertainty quantification:
  enable: True
  number of samples: 100
```


Figure 4.7: Input YAML file for the UQ example.

The *dose_samples* and *computed_doses* files were not discussed in the preceding two examples, as they contained only a single sample that was more easily obtained from *dose_statistics*. However, this example uses 100 dose samples resulting in a pool of 100 computed doses and these data files are where these are saved. These data are shown in Tables 4.7–4.8 below, and the statistics are summarized in *dose_statistics* shown in Table 4.9. Note that the TLD dosimeter is not involved in the unfold method, and therefore there are no computed values.

Table 4.7: Dose samples data file for the UQ example.

type device detector sample	spectrometers L3_Pithon 0.25mm-Al	 1mm-Al	 4mm-Al	 ...	 0.15mm-Au	dosimeters TLD TLD_35mil-CaF2_10mil-Al
1	4.282391	5.4439855	4.5155525	...	0.751361	6.084156
2	4.7913427	4.843667	3.8198287	...	1.0846595	9.164395
3	4.821981	3.7891414	3.7852182	...	0.98220414	7.02631
...	...	...	...	...	...	...
98	5.861426	5.0524893	4.5120606	...	0.8616586	7.9967585
99	4.8507395	5.466274	3.0549974	...	0.88964015	6.7781696
100	2.7964225	5.1399226	4.7794037	...	1.1151402	5.008461

Table 4.8: Computed doses data file for the UQ example.

type device detector sample	spectrometers L3_Pithon 0.25mm-Al	 1mm-Al	 4mm-Al	 ...	 0.15mm-Au
1	5.427091	4.5067897	3.1228569	...	0.64247835
2	5.202408	4.132148	2.8252127	...	0.6383787
3	4.8538337	4.2432356	3.1927183	...	0.674767
...	...	...	...	...	...
98	6.126521	4.9607644	3.4258254	...	0.6800188
99	5.362213	4.4533515	3.1324892	...	0.70900583
100	4.327503	3.8816762	3.0681412	...	0.7629541

Table 4.9: The dose statistics data file for the UQ example.

	type	spectrometers				dosimeters
	device	L3_Pithon				TLD
	detector	0.25mm-Al	1mm-Al	4mm-Al	...	TLD_35mil-CaF2_10mil-Al
source	statistic					
measured	mean	4.63	4.925	3.88	...	7.553
	median	4.63	4.925	3.88	...	7.553
	standard deviation	0.6945	0.73875	0.582	...	1.133
	- expanded uncertainty	1.361195	1.4479234	1.140699	...	2.2206392
	+ expanded uncertainty	1.361195	1.4479234	1.140699	...	2.2206392
	average expanded uncertainty	1.361195	1.4479234	1.140699	...	2.2206392
sampled	mean	4.6124964	4.9364524	3.911593	...	7.6119046
	median	4.67678	4.91425	3.9022696	...	7.598536
	standard deviation	0.69430405	0.7372591	0.57637423	...	1.1649007
	- expanded uncertainty	1.3790689	1.3261184	1.097359	...	2.1407733
	+ expanded uncertainty	1.2238338	1.4298403	0.97161543	...	2.3823109
	average expanded uncertainty	1.3014513	1.3779794	1.0344871	...	2.261542
computed	mean	5.139593	4.351306	3.1809802	...	nan
	median	5.1476827	4.3761024	3.1530538	...	nan
	standard deviation	0.53882337	0.34004727	0.18514818	...	nan
	- expanded uncertainty	1.0580193	0.67660916	0.2885031	...	nan
	+ expanded uncertainty	0.99793535	0.65358096	0.3978835	...	nan
	average expanded uncertainty	1.0279773	0.66509503	0.3431933	...	nan

For the UQ example, the *spectrum_parameters* file shown in Table 4.10 differs only by including 100 samples compared to one for the preceding examples. This example also contains the *spectrum_parameter_statistics* file shown in Table 4.11. The unfolded spectrum output is contained in the *unfolded_spectra* file shown in Table 4.12 which now contains multiple columns for each sample, as well as *spectrum_statistics* shown in Table 4.13.

Table 4.10: The spectrum parameters data file for the UQ example.

type param units sample	model param a1 unitless		 kappa unitless	 scale constant 1/MeV	particle spectrum average energy MeV	TLD fluence #/cm2	spectrometer fluence #/cm2
1	0.0	...	0.86703926	1.113127	0.074427724	1.712956e+12	1.462514e+12
2	0.0	...	1.0442561e-06	0.67615694	0.07371929	2.558856e+12	1.404816e+12
3	0.0	...	0.9999788	1.0480231	0.073603086	2.172514e+12	1.418041e+12
...	...	...	...	...	...	...	...
98	0.0	...	0.84730166	0.7639722	0.068561055	2.166290e+12	1.596954e+12
99	0.0	...	2.2950624e-07	0.9397112	0.07601346	2.001387e+12	1.516149e+12
100	0.0	...	2.4847478e-07	0.8508779	0.07952352	1.741232e+12	1.415954e+12

Table 4.11: The spectrum parameter statistics data file for the UQ example.

type param units statistic	model param a1 unitless		 kappa unitless	 scale constant 1/MeV	particle spectrum average energy MeV	TLD fluence #/cm2	spectrometer fluence #/cm2
mean	0.0	...	0.3391066	0.8518334	0.07435519	2.314882e+12	1.481286e+12
median	0.0	...	0.3391066	0.8518334	0.07435519	2.314882e+12	1.481286e+12
standard deviation	0.0	...	0.44173917	0.17737405	0.00306615	3.718400e+11	8.292893e+10
- expanded uncertainty	0.0	...	0.00013558388	0.3214006	0.0051309867	6.062193e+11	1.377162e+11
+ expanded uncertainty	0.0	...	0.9998641	0.3548126	0.0053365882	7.242147e+11	1.763782e+11
average expanded uncertainty	0.0	...	0.49999982	0.3381066	0.0052337875	6.652170e+11	1.570472e+11

Table 4.12: The unfolded spectra data file for the UQ example.

sample E [MeV]	1	2	3	...	98	99	100
0.0050000	0.0	0.0	0.0	...	0.0	0.0	0.0
0.0050127	3.930297e-32	4.4655566e-16	1.9e-43	...	1.411862e-19	8.353448e-29	0.0
0.0050254	6.533659e-32	5.8025463e-16	3.76e-43	...	1.9359335e-19	1.3189659e-28	0.0
...	...	...	...	...	...	...	...
0.0629254	10.434192	7.10426	9.376736	...	7.6396136	8.962477	7.7950315
0.0630853	10.402785	7.083024	9.351369	...	7.6161976	8.935043	7.7774105
0.0632456	10.371387	7.0649385	9.328393	...	7.5955005	8.907631	7.7665944
0.0634062	10.339996	7.0564456	9.31279	...	7.583078	8.880258	7.7766976
0.0635674	10.308615	7.0741034	9.317361	...	7.593209	8.852964	7.843996
...	...	...	...	...	...	...	...
0.7959501	0.0	0.0	0.0	...	0.0	0.0	0.0
0.7979725	0.0	0.0	0.0	...	0.0	0.0	0.0
0.8000000	0.0	0.0	0.0	...	0.0	0.0	0.0

Table 4.13: The spectrum statistics data file for the UQ example.

statistic E [MeV]	mean	median	...	+ expanded uncertainty	average expanded uncertainty
0.0050000	0.0	0.0	...	0.0	0.0
0.0050127	5.828384e-09	3.042645e-31	...	3.9893185e-15	1.9946593e-15
0.0050254	6.5777814e-09	4.981038e-31	...	5.0890506e-15	2.5445253e-15
...	...	...	...	...	...
0.0629254	8.087256	8.096174	...	2.3569489	2.5825696
0.0630853	8.064914	8.076272	...	2.345299	2.5724757
0.0632456	8.04635	8.060047	...	2.3299925	2.5573223
0.0634062	8.039365	8.047675	...	2.3108835	2.5266535
0.0635674	8.064004	8.047102	...	2.2800925	2.4535992
...	...	...	...	...	...
0.7959501	0.0	0.0	...	0.0	0.0
0.7979725	0.0	0.0	...	0.0	0.0
0.8000000	0.0	0.0	...	0.0	0.0

The plot output for the UQ example is similar to the previous cases, except they plot the spectrum and doses with uncertainty. The *spectrum* plot is replaced with a *spectrum_uncertainty* shown in Figure 4.8, where the line is the median unfolded spectrum, and the shaded region represents 95% confidence. Likewise, the *dose_statisics* plot in FIgure 4.9 now shows computed doses as error bars instead of a bar plot.

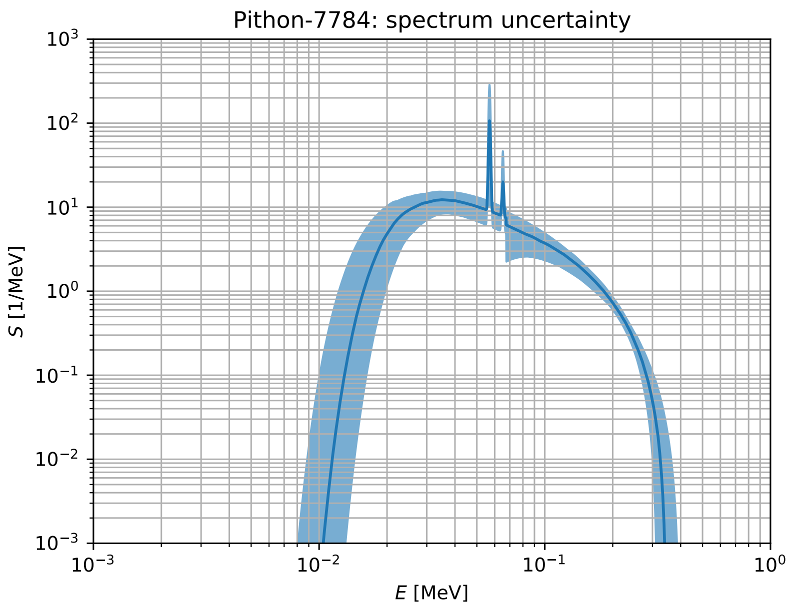

Figure 4.8: Spectrum uncertainty plot for the UQ example.

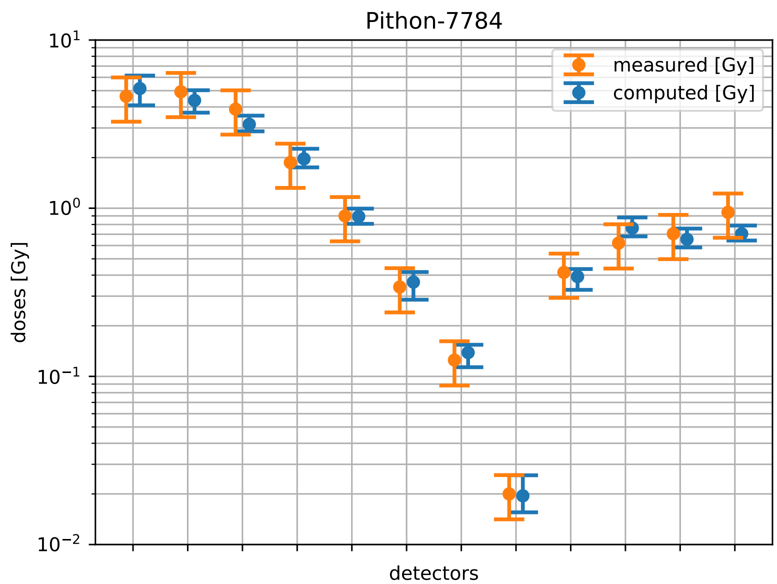

Figure 4.9: Dose statistics plot for the UQ example.

4.4 Time-dependent unfold

The final example is an initial trial of the AXIOM spectrometer fielded at the TriMeV facility, which is included primarily to demonstrate how to run time-dependent spectral unfolds. The key difference for the input deck shown in Figure 4.10 is that it contains dose (and uncertainty) data for five different time bins, where only the first and last are shown. This example also demonstrates how to disable bad measured doses which are specified as *np.nan*, which causes the unfold routine to ignore these measurements in the calculations. The corresponding uncertainties are also not used in the unfold routine, but here are indicated as *np.inf* for consistency only; however, the results would be completely unchanged if these uncertainties for the disabled doses were specified as any floating point value.

```yaml
shot information:
  machine name: TriMeV
  shot number: 364

spectrometer data:
  spectrometers:
    selection: spectrometers
    spectrometers and fielding distances:
      AXIOM-Unpotted_SARS: 1
  measurements:
    source: inline
    doses:
      [[[0.270288], [0.284805], [0.262331], [0.384524], [0.145324], [np.nan], [0.027777], [0.0628975],
        [0.296722], [0.241925], [0.27508], [0.189945], [0.144541], [0.111232], [0.0986409], [0.354737],
        [0.370704], [0.377755], [0.280731], [0.123694], [np.nan], [np.nan], [0.0743794], [0.0684051]],
       ...
       [[1.40783], [1.33561], [1.30936], [1.2018], [0.758377], [np.nan], [0.40325], [0.355617],
        [1.31043], [1.26119], [1.16904], [0.888885], [0.636213], [0.487754], [0.437299], [1.45046],
        [1.40512], [1.40396], [1.11342], [0.49781], [np.nan], [np.nan], [0.392662], [0.455283]]]
    uncertainties:
      [[[0.05], [0.05], [0.05], [0.05], [0.05], [np.inf], [0.05], [0.05],
        [0.05], [0.05], [0.05], [0.05], [0.05], [0.05], [0.05], [0.05],
        [0.05], [0.05], [0.05], [0.05], [np.inf], [np.inf], [0.05], [0.05]],
       ...
       [[0.05], [0.05], [0.05], [0.05], [0.05], [np.inf], [0.05], [0.05],
        [0.05], [0.05], [0.05], [0.05], [0.05], [0.05], [0.05], [0.05],
        [0.05], [0.05], [0.05], [0.05], [np.inf], [np.inf], [0.05], [0.05]]]

falloff:
  model: r-order
  order: 0

unfold:
  method: parametric
  parametric:
    function: L3_Brems_model
    variable parameters:
      {a1: False, a2: False, alpha: True, b: False, Ecut: True, Emax: True, kappa: False}
    initial parameters:
      {a1: 0., a2: 0., alpha: 0., b: 0., Ecut: 0.02, Emax: 1.8, kappa: 0.5}
    lower parameter bounds:
      {a1: -10., a2: -10., alpha: 0., b: 0., Ecut: 0.01, Emax: 0.2, kappa: 0.0}
    upper parameter bounds:
      {a1: 10., a2: 10., alpha: 0.5, b: 0.05, Ecut: 0.1, Emax: 5.0, kappa: 3.0}
  energy bounds: [0.005, 5.0]

locations of interest:
  time integrated:
    specification: fielding distance
    time integrated: True
    fielding distance: 1
  time resolved:
    specification: fielding distance
    time integrated: False
    fielding distance: 1

numerics:
  random number generator:
    seed: 3673100370
```


Figure 4.10: Input YAML file for the time-dependent example.

The key difference in the output files is the presence of a time bin index in the output. For the *dose_statistics* shown in Table 4.14, the time bin index occurs in the row, and the *dose_samples* and *computed_doses* (not shown) are similar. Specifically, for the *dose_statistics* we show only detectors 1, 2, 6, and 24 to demonstrate that measured and sampled doses for the disabled detector (detector 6) are consistent with the inputted values. Also shown is the *spectrum_parameters* file in Table 4.15. The *unfold_spectra* data shown in Table 4.16 includes the time bin in the header.

Table 4.14: The dose statistics data file for the time-dependent example.

		type	spectrometers					
		device	AXIOM-Unpotted_SARS					
		detector	0.32mm-Al	1.00mm-Al	...	1.40mm-Ta	...	10.02mm-Cu
time bin	source	statistic						
1	measured	mean	0.270288	0.284805	...	nan	...	0.0684051
		median	0.270288	0.284805	...	nan	...	0.0684051
		standard deviation	0.05	0.05	...	inf	...	0.05
		- expanded uncertainty	0.0979982	0.0979982	...	inf	...	0.0979982
		+ expanded uncertainty	0.0979982	0.0979982	...	inf	...	0.0979982
		average expanded uncertainty	0.0979982	0.0979982	...	inf	...	0.0979982
	sampled	mean	0.270288	0.284805	...	nan	...	0.0684051
		median	0.270288	0.284805	...	nan	...	0.0684051
	computed	mean	0.3122916	0.31338188	...	0.13472141	...	0.12954324
		median	0.3122916	0.31338188	...	0.13472141	...	0.12954324
2	measured	mean	0.813174	0.776741	...	nan	...	0.179588
		median	0.813174	0.776741	...	nan	...	0.179588
		standard deviation	0.05	0.05	...	inf	...	0.05
		- expanded uncertainty	0.0979982	0.0979982	...	inf	...	0.0979982
		+ expanded uncertainty	0.0979982	0.0979982	...	inf	...	0.0979982
		average expanded uncertainty	0.0979982	0.0979982	...	inf	...	0.0979982
	sampled	mean	0.813174	0.776741	...	nan	...	0.179588
		median	0.813174	0.776741	...	nan	...	0.179588
	computed	mean	0.88534725	0.82824147	...	0.36002287	...	0.34012756
		median	0.88534725	0.82824147	...	0.36002287	...	0.34012756
...	...	...	...	...	...	...	...	...
10	measured	mean	1.40783	1.33561	...	nan	...	0.455283
		median	1.40783	1.33561	...	nan	...	0.455283
		standard deviation	0.05	0.05	...	inf	...	0.05
		- expanded uncertainty	0.0979982	0.0979982	...	inf	...	0.0979982
		+ expanded uncertainty	0.0979982	0.0979982	...	inf	...	0.0979982
		average expanded uncertainty	0.0979982	0.0979982	...	inf	...	0.0979982
	sampled	mean	1.40783	1.33561	...	nan	...	0.455283
		median	1.40783	1.33561	...	nan	...	0.455283
	computed	mean	1.393732	1.3708354	...	0.6386324	...	0.60033774
		median	1.393732	1.3708354	...	0.6386324	...	0.60033774

Table 4.15: The spectrum parameters data file for the time-dependent example.

time bin	type param units sample	model param a1 unitless		 kappa unitless	 scale constant 1/MeV	particle spectrum average energy MeV	time integrated fluence #/cm2	time resolved fluence #/cm2
1	1	0.0	...	0.5	0.34528777	0.21059673	nan	1.948222e+11
2	1	0.0	...	0.5	0.33823428	0.26145804	nan	3.802834e+11
3	1	0.0	...	0.5	0.3851994	0.24757245	nan	9.241549e+11
4	1	0.0	...	0.5	0.34912616	0.25250405	nan	1.455820e+12
5	1	0.0	...	0.5	0.36247858	0.2544544	nan	1.791050e+12
6	1	0.0	...	0.5	0.36857328	0.2397391	nan	1.941392e+12
7	1	0.0	...	0.5	0.2973199	0.23431304	nan	1.627355e+12
8	1	0.0	...	0.5	0.23325348	0.24545863	nan	1.185879e+12
9	1	0.0	...	0.5	0.15691192	0.27399656	nan	8.082517e+11
10	1	0.0	...	0.5	0.16989478	0.29767245	nan	6.273867e+11
int	1	nan	...	nan	nan	nan	1.093640e+13	nan

Table 4.16: Unfolded spectra data file for the time-dependent example.

time bin sample E [MeV]	1 1	2 1	... ...	10 1
0.0050000	0.0	0.0	...	0.0
0.0050173	0.0	5.997484e-26	...	1.7602944e-19
0.0050347	0.0	1.042984e-25	...	2.6632514e-19
0.0050521	0.0	1.8047804e-25	...	4.014375e-19
...	...	...	...	...
0.1570254	1.630195	1.7693832	...	0.95452225
0.1575687	1.6254048	1.7634435	...	0.9515407
0.1581139	1.6206086	1.7575115	...	0.9485631
0.1586609	1.6158063	1.7515877	...	0.9455894
0.1592099	1.6109984	1.745672	...	0.94261974
...	...	...	...	...
4.9655802	0.0	0.0	...	1.6204704e-06
4.9827604	0.0	0.0	...	4.048906e-07
5.0000000	0.0	0.0	...	0.0

The plots for time-dependent unfolds are unchanged, except that a separate plot is produced for each time bin. The *spectrum* plot and *dose_statistics* plots for the sixth time bin are shown in Figures 4.11–4.12. Note that for the *dose_statistics* plot, there are no measured doses displayed for the disabled doses (detectors 6, 21, and 22).

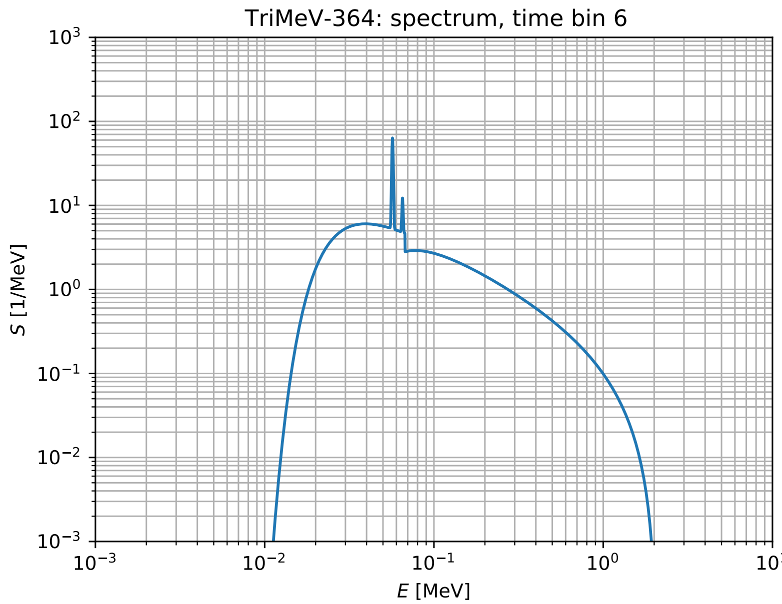

Figure 4.11: Spectrum plot for sixth time bin of the time-dependent example.

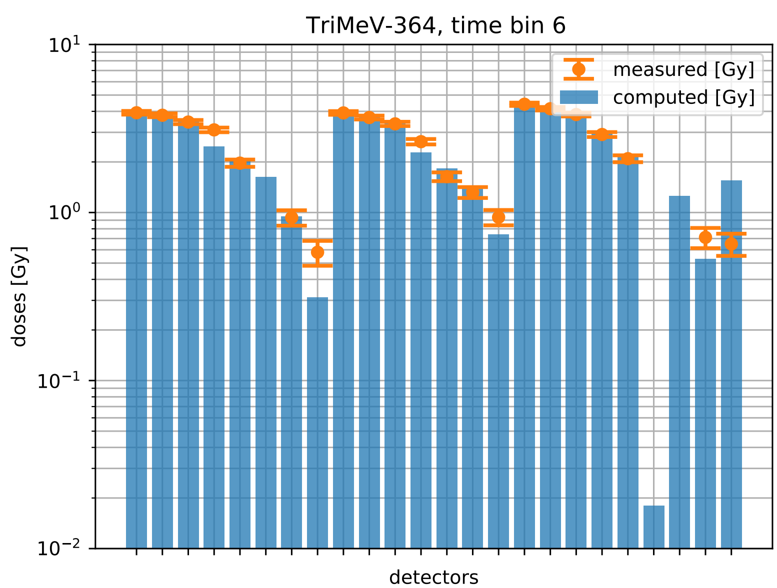

Figure 4.12: Dose statistics plot for the sixth time bin of the time-dependent example.

5 Conclusion

The AXIOM-Unfold application is a modern python implementation of the parametric spectral unfold method, with options for both uncertainty quantification and time-resolved unfolds. This code was developed along with the AXIOM project to develop time-resolved spectral unfolds on Saturn, but can also be used on other radiation sources like Pithon, and remains under active development.

References

[1] S. Johnson and S. Gorbics, *Health Phys.*, vol. 31, p. 859, 1981.

[2] J. C. Riordan and N. Qi, "Parametric spectral unfold for differential absorption spectrometer." Hardened Electronics and Radiation Technologies (HEART) Conference, Tampa, FL, March 21–25, 2005.

Distribution

Email - Internal

Name	Org.	Email Address
Technical Library	9536	libref@sandia.gov