# xpecgen Documentation

***Release 1.3.0***

**dih5**

**Jun 18, 2021**

# Contents

For general instructions on the software check the Github repository.

The API is documented below.

# CHAPTER 1

# xpecgen

The module xpecgen.xpecgen contains the API most of the programmers will need.

xpecgen.py: A module to calculate x-ray spectra generated in tungsten anodes.

**class** xpecgen.xpecgen.**Spectrum**

Set of 2D points and discrete components representing a spectrum.

A Spectrum can be multiplied by a scalar (int, float...) to increase its counts in such a factor. Two spectra can be added if they share their x axes and their discrete component positions.

**Note: When two spectrum are added it is not checked it that addition makes sense. It is the user's responsibility to** do so.

**x**

x coordinates (energy) describing the continuum part of the spectrum.

**Type** numpy.ndarray

**y**

y coordinates (pdf) describing the continuum part of the spectrum.

**Type** numpy.ndarray

**discrete**

discrete components of the spectrum, each of the form [x, num, rel_x] where:

- x is the mean position of the peak.
- num is the number of particles in the peak.
- rel_x is a characteristic distance where it should extend. The exact meaning depends on the windows function.

**Type** List[List[float]]

**attenuate**(*depth=1*, *mu=<function Spectrum.<lambda>>*)

Attenuate the spectrum as if it passed thorough a given depth of material with attenuation described by a given attenuation coefficient. Consistent units should be used.

**Parameters**

- **depth** – The amount of material (typically in cm).
- **mu** – The energy-dependent absorption coefficient (typically in cm^-1).

**clone**()
Return a new Spectrum object cloning itself

**Returns** The new Spectrum.

**Return type** *Spectrum*

**export_csv**(*route='a.csv'*, *peak_shape=<function triangle>*, *transpose=False*)
Export the data to a csv file (comma-separated values).

**Parameters**

- **route** (*str*) – The route where the file will be saved.
- **peak_shape** – The window function used to plot the peaks. See *triangle* for an example.
- **transpose** (*bool*) – True to write in two columns, False in two rows.

**export_xlsx**(*route='a.xlsx'*, *peak_shape=<function triangle>*, *markers=False*)
Export the data to a xlsx file (Excel format).

**Parameters**

- **route** (*str*) – The route where the file will be saved.
- **peak_shape** – The window function used to plot the peaks. See *triangle* for an example.
- **markers** (*bool*) – Whether to use markers or a continuous line in the plot in the file.

**get_continuous_function**()
Get a function representing the continuous part of the spectrum.

**Returns** An interpolation function representing the continuous part of the spectrum.

**get_norm**(*weight=None*)
Return the norm of the spectrum using a weighting function.

**Parameters** **weight** – A function used as a weight to calculate the norm. Typical examples are:

- weight(E)=1 [Photon number]
- weight(E)=E [Energy]
- weight(E)=fluence2Dose(E) [Dose]

**Returns** The calculated norm.

**Return type** (float)

**get_plot**(*place*, *show_mesh=True*, *prepare_format=True*, *peak_shape=<function triangle>*)
Prepare a plot of the data in the given place

**Parameters**

- **place** – The class whose method plot is called to produce the plot (e.g., matplotlib.pyplot).
- **show_mesh** (*bool*) – Whether to plot the points over the continuous line as circles.

- **prepare_format** (*bool*) – Whether to include ticks and labels in the plot.
- **peak_shape** – The window function used to plot the peaks. See *triangle* for an example.

**get_points**(*peak_shape=<function triangle>*, *num_discrete=10*)
Returns two lists of coordinates x y representing the whole spectrum, both the continuous and discrete components. The mesh is chosen by extending x to include details of the discrete peaks.

**Parameters**

- **peak_shape** – The window function used to calculate the peaks. See *triangle* for an example.
- **num_discrete** – Number of points that are added to mesh in each peak.

**Returns**

tuple containing:

x2 (List[float]): The list of x coordinates (energy) in the whole spectrum.

y2 (List[float]): The list of y coordinates (density) in the whole spectrum.

**Return type** (tuple)

**hvl**(*value=0.5*, *weight=<function Spectrum.<lambda>>*, *mu=<function Spectrum.<lambda>>*, *energy_min=0*)
Calculate a generalized half-value-layer.

This method calculates the depth of a material needed for a certain dosimetric magnitude to decrease in a given factor.

**Parameters**

- **value** (*float*) – The factor the desired magnitude is decreased. Must be in [0, 1].
- **weight** – A function used as a weight to calculate the norm. Typical examples are:
  - weight(E)=1 [Photon number]
  - weight(E)=E [Energy]
  - weight(E)=fluence2Dose(E) [Dose]
- **mu** – The energy absorption coefficient as a function of energy.
- **energy_min** (*float*) – A low-energy cutoff to use in the calculation.

**Returns** The generalized hvl in cm.

**Return type** (float)

**set_norm**(*value=1*, *weight=None*)
Set the norm of the spectrum using a weighting function.

**Parameters**

- **value** (*float*) – The norm of the modified spectrum in the given convention.
- **weight** – A function used as a weight to calculate the norm. Typical examples are:
  - weight(E)=1 [Photon number]
  - weight(E)=E [Energy]
  - weight(E)=fluence2Dose(E) [Dose]

**show_plot**(*show_mesh=True*, *block=True*)
Prepare the plot of the data and show it in matplotlib window.

**Parameters**

- **show_mesh** (*bool*) – Whether to plot the points over the continuous line as circles.
- **block** (*bool*) – Whether the plot is blocking or non blocking.

xpecgen.xpecgen.**add_char_radiation**(*s*, *method='fraction_above_poly'*)
Adds characteristic radiation to a calculated bremsstrahlung spectrum, assuming it is a tungsten-generated spectrum

If a discrete component already exists in the spectrum, it is replaced.

**Parameters**

- **s** (*Spectrum*) – The spectrum whose discrete component is recalculated.
- **method** (*str*) – The method to use to calculate the discrete component. Available methods include:
  - 'fraction_above_linear': Use a linear relation between bremsstrahlung above the K-edge and peaks.
  - 'fraction_above_poly': Use polynomial fits between bremsstrahlung above the K-edge and peaks.

xpecgen.xpecgen.**calculate_spectrum**(*e_0*, *theta*, *e_min*, *num_e*, *phi=0.0*, *epsrel=0.2*, *monitor=<function console_monitor>*, *z=74*)
Calculates the x-ray spectrum for given parameters. Characteristic peaks are also calculated by add_char_radiation, which is called with the default parameters.

**Parameters**

- **e_0** (*float*) – Electron kinetic energy in keV
- **theta** (*float*) – X-ray emission angle in degrees, the normal being at 90º
- **e_min** (*float*) – Minimum kinetic energy to calculate in the spectrum in keV
- **num_e** (*int*) – Number of points to calculate in the spectrum
- **phi** (*float*) – X-ray emission elevation angle in degrees.
- **epsrel** (*float*) – The tolerance parameter used in numeric integration.
- **monitor** – A function to be called after each iteration with arguments finished_count, total_count. See for example *console_monitor*.
- **z** (*int*) – Atomic number of the material.

**Returns** The calculated spectrum

**Return type** *Spectrum*

xpecgen.xpecgen.**calculate_spectrum_mesh**(*e_0*, *theta*, *mesh*, *phi=0.0*, *epsrel=0.2*, *monitor=<function console_monitor>*, *z=74*)
Calculates the x-ray spectrum for given parameters. Characteristic peaks are also calculated by add_char_radiation, which is called with the default parameters.

**Parameters**

- **e_0** (*float*) – Electron kinetic energy in keV
- **theta** (*float*) – X-ray emission angle in degrees, the normal being at 90º

- **mesh** (*list of float or ndarray*) – The photon energies where the integral will be evaluated
- **phi** (*float*) – X-ray emission elevation angle in degrees.
- **epsrel** (*float*) – The tolerance parameter used in numeric integration.
- **monitor** – A function to be called after each iteration with arguments finished_count, total_count. See for example *console_monitor*.
- **z** (*int*) – Atomic number of the material.

**Returns** The calculated spectrum

**Return type** *Spectrum*

xpecgen.xpecgen.**console_monitor**(*a*, *b*)

Simple monitor function which can be used with *calculate_spectrum*.

Prints in stdout 'a/b'.

**Parameters**

- **a** – An object representing the completed amount (e.g., a number representing a part...).
- **b** – An object representing the total amount (... of a number representing a total).

xpecgen.xpecgen.**custom_dblquad**(*func*, *a*, *b*, *c*, *d*, *args=()*, *epsabs=1.49e-08*, *epsrel=1.49e-08*, *maxp1=50*, *limit=2000*)

A wrapper around numpy's dblquad to restrict it to a rectangular region and to pass arguments to the 'inner' integral.

**Parameters**

- **func** – The integrand function f(y,x).
- **a** (*float*) – The lower bound of the second argument in the integrand function.
- **b** (*float*) – The upper bound of the second argument in the integrand function.
- **c** (*float*) – The lower bound of the first argument in the integrand function.
- **d** (*float*) – The upper bound of the first argument in the integrand function.
- **args** (*sequence, optional*) – extra arguments to pass to func.
- **epsabs** (*float, optional*) – Absolute tolerance passed directly to the inner 1-D quadrature integration.
- **epsrel** (*float, optional*) – Relative tolerance of the inner 1-D integrals. Default is 1.49e-8.
- **maxp1** (*float or int, optional*) – An upper bound on the number of Chebyshev moments.
- **limit** (*int, optional*) – Upper bound on the number of cycles (>=3) for use with a sinusoidal weighting and an infinite end-point.

**Returns**

tuple containing:

y (float): The resultant integral.

abserr (float): An estimate of the error.

**Return type** (tuple)

xpecgen.xpecgen.**get_cs**(*e_0=100*, *z=74*)
Returns a function representing the scaled bremsstrahlung cross_section.

**Parameters**

- **e_0** (*float*) – The electron kinetic energy, used to scale u=e_e/e_0.
- **z** (*int*) – Atomic number of the material.

**Returns** A function representing cross_section(e_g,u) in mb/keV, with e_g in keV.

xpecgen.xpecgen.**get_csda**(*z=74*)
Returns a function representing the CSDA range in tungsten.

**Parameters** **z** (*int*) – Atomic number of the material.

**Returns** The CSDA range in cm in tungsten as a function of the electron kinetic energy in keV.

xpecgen.xpecgen.**get_fluence**(*e_0=100.0*)
Returns a function representing the electron fluence with the distance in CSDA units.

**Parameters** **e_0** (*float*) – The kinetic energy whose CSDA range is used to scale the distances.

**Returns** A function representing fluence(x,u) with x in CSDA units.

xpecgen.xpecgen.**get_fluence_to_dose**()
Returns a function representing the weighting factor which converts fluence to dose.

**Returns** A function representing the weighting factor which converts fluence to dose in Gy * cm^2.

xpecgen.xpecgen.**get_mu**(*z=74*)
Returns a function representing an energy-dependent attenuation coefficient.

**Parameters** **z** (*int or str*) – The identifier of the material in the data folder, typically the atomic number.

**Returns** The attenuation coefficient mu(E) in cm^-1 as a function of the energy measured in keV.

xpecgen.xpecgen.**get_mu_csda**(*e_0*, *z=74*)
Returns a function representing the CSDA-scaled energy-dependent attenuation coefficient in tungsten.

**Parameters**

- **e_0** (*float*) – The electron initial kinetic energy.
- **z** (*int*) – Atomic number of the material.

**Returns** The attenuation coefficient mu(E) in CSDA units as a function of the energy measured in keV.

xpecgen.xpecgen.**get_source_function**(*fluence*, *cs*, *mu*, *theta*, *e_g*, *phi=0.0*)
Returns the attenuated source function (Eq. 2 in the paper) for the given parameters.

An E_0-dependent factor (the fraction found there) is excluded. However, the E_0 dependence is removed in integrate_source.

**Parameters**

- **fluence** – The function representing the fluence.
- **cs** – The function representing the bremsstrahlung cross-section.
- **mu** – The function representing the attenuation coefficient.
- **theta** (*float*) – The emission angle in degrees, the anode's normal being at 90º.
- **e_g** (*float*) – The emitted photon energy in keV.

- **phi** (*float*) – The elevation angle in degrees, the anode's normal being at 12º.

**Returns** The attenuated source function s(u,x).

xpecgen.xpecgen.**integrate_source**(*fluence*, *cs*, *mu*, *theta*, *e_g*, *e_0*, *phi=0.0*, *x_min=0.0*, *x_max=0.6*, *epsrel=0.1*, *z=74*)

Find the integral of the attenuated source function.

An E_0-independent factor is excluded (i.e., the E_0 dependence on get_source_function is taken into account here).

**Parameters**

- **fluence** – The function representing the fluence.
- **cs** – The function representing the bremsstrahlung cross-section.
- **mu** – The function representing the attenuation coefficient.
- **theta** (*float*) – The emission angle in degrees, the anode's normal being at 90º.
- **e_g** – (float): The emitted photon energy in keV.
- **e_0** (*float*) – The electron initial kinetic energy.
- **phi** (*float*) – The elevation angle in degrees, the anode's normal being at 12º.
- **x_min** – The lower-bound of the integral in depth, scaled by the CSDA range.
- **x_max** – The upper-bound of the integral in depth, scaled by the CSDA range.
- **epsrel** – The relative tolerance of the integral.
- **z** (*int*) – Atomic number of the material.

**Returns** The value of the integral.

**Return type** float

xpecgen.xpecgen.**log_interp_1d**(*xx*, *yy*, *kind='linear'*)

Perform interpolation in log-log scale.

**Parameters**

- **xx** (*List[float]*) – x-coordinates of the points.
- **yy** (*List[float]*) – y-coordinates of the points.
- **kind** (*str or int, optional*) – The kind of interpolation in the log-log domain. This is passed to scipy.interpolate.interp1d.

**Returns** A function whose call method uses interpolation in log-log scale to find the value at a given point.

xpecgen.xpecgen.**triangle**(*x*, *loc=0*, *size=0.5*, *area=1*)

The triangle window function centered in loc, of given size and area, evaluated at a point.

**Parameters**

- **x** – The point where the function is evaluated.
- **loc** – The position of the peak.
- **size** – The total.
- **area** – The area below the function.

**Returns** The value of the function.

# CHAPTER 2

## xpecgenGUI

The module xpecgen.xpecgenGUI defines the tk-based GUI and probably is only interesting to developers aiming to extend the GUI.

xpecgenGUI.py: A GUI for the xpecgen module

**class** xpecgen.xpecgenGUI.**CreateToolTip**(*widget*, *text*, *color='#ffe14c'*)
A tooltip for a given widget.

**class** xpecgen.xpecgenGUI.**ParBox**(*master=None*, *textvariable=0*, *lblText=''*, *unitsTxt=''*, *helpTxt=''*, *row=0*, *read_only=False*)
A parameter entry with labels preceding and succeeding it and an optional tooltip

**class** xpecgen.xpecgenGUI.**XpecgenGUI**(*master=None*)
Tk-based GUI for the xpecgen package.

**attenuate**()
Attenuate the active spectrum according to the parameters in the GUI.

**calculate**()
Calculates a new spectrum using the parameters in the GUI.

**clean_history**()
Clean the spectra history.

**createWidgets**()
Create the widgets in the GUI

**enable_analyze_buttons**()
Enable widgets requiring a calculated spectrum to work.

**export**()
Export the selected spectrum in xlsx format, showing a file dialog to choose the route.

**history_poll**()
Polling method to update changes in spectrum list.

**initVariables**()
Create and initialize interface variables

**monitor_bar**(*a*, *b*)
Update the progress bar.

**Parameters**

- **a** (*int*) – The number of items already calculated.
- **b** (*int*) – The total number of items to calculate.

**normalize**()
Normalize the active spectrum according to the parameters in the GUI.

**update_param**()
Update parameters calculated from the active spectrum.

**update_plot**()
Update the canvas after plotting something. If matplotlib is not embedded, show it in an independent window.

**wait_for_calculation**()
Polling method to wait for the calculation thread to finish. Also updates monitor_bar.

xpecgen.xpecgenGUI.**main**()
Start an instance of the GUI.

CHAPTER 3

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## x

# Index

## P



## S



## T



## U



## W



## X



## Y