Curve Module Documentation

1 Overview

This module provides utilities for working with exceedance curves and probability distributions. It includes functions for manipulating and processing bin edges and probabilities, as well as a class `ExceedanceCurve` for representing and working with exceedance curves.

2 Functions

2.1 `add_x_value_to_curve(x, curve_x, curve_y)`

Adds an x value to a curve, interpolated from the existing curve.

- Parameters:
  - `x`: The x value to add
  - `curve_x`: The existing x values of the curve (sorted non-decreasing)
  - `curve_y`: The existing y values of the curve
- Returns: Updated `curve_x` and `curve_y`

2.2 `to_exceedance_curve(bin_edges, probs)`

Converts bin edges and probabilities to an exceedance curve.

- Parameters:
  - `bin_edges`: The edges of the bins
  - `probs`: The probabilities for each bin
- Returns: An `ExceedanceCurve` object

2.3 process_bin_edges_and_probs(bin_edges, probs, range_fraction=0.05)

Processes bin edges and probabilities to handle zero-width bins.

- Parameters:
  - bin_edges: The edges of the bins
  - probs: The probabilities for each bin
  - range_fraction: Fraction of range to use for zero-width bins (default: 0.05)
- Returns: Processed bin edges and probabilities

2.4 process_bin_edges_for_graph(bin_edges, range_fraction=0.05)

Processes bin edges for graph display, handling zero-width bins.

- Parameters:
  - bin_edges: The edges of the bins
  - range_fraction: Fraction of range to use for zero-width bins (default: 0.05)
- Returns: Processed bin edges

2.5 __next_non_equal_index(ndarray, i)

Helper function to find the next index in an array with a different value.

- Parameters:
  - ndarray: The input array
  - i: The starting index
- Returns: The next index with a different value

3 Class: ExceedanceCurve

Represents an exceedance curve, where each point comprises a value v and a probability p. The probability p represents the chance that the random variable is greater than or equal to v.

3.1 Methods

3.1.1 __init__(self, probs, values)

Initializes the ExceedanceCurve.

- Parameters:
  - probs: Exceedance probabilities (must be sorted and decreasing)
  - values: Corresponding values (must be sorted and non-decreasing)

3.1.2 `add_value_point(self, value)`

Adds a point to the curve with the specified value.

- Parameters:
  - `value`: The value to add
- Returns: A new ExceedanceCurve with the added point

3.1.3 `get_value(self, prob)`

Gets the value corresponding to a given probability.

- Parameters:
  - `prob`: The probability
- Returns: The corresponding value

3.1.4 `get_probability_bins(self, include_last=False)`

Converts from exceedance probability to bins of constant probability density.

- Parameters:
  - `include_last`: Whether to include the last bin (default: False)
- Returns: Tuple of value bins and probabilities

3.1.5 `get_samples(self, uniforms)`

Returns values for given uniform random variables.

- Parameters:
  - `uniforms`: Array of uniform random variables
- Returns: Array of corresponding values

4 Usage Notes

1. The `ExceedanceCurve` class is central to this module and provides various methods for working with exceedance curves.
2. The `add_x_value_to_curve` function is useful for aligning curves and bins, but care should be taken with multiple identical x values.
3. The `process_bin_edges_and_probs` and `process_bin_edges_for_graph` functions are helpful for handling zero-width bins, which can be problematic in certain analyses or visualizations.

4. When creating an `ExceedanceCurve`, ensure that the probabilities are sorted and decreasing, and the values are sorted and non-decreasing.

5. The `get_samples` method of `ExceedanceCurve` can be used for Monte Carlo simulations or other sampling-based analyses.

6. This module relies heavily on numpy for efficient array operations. Users should be familiar with numpy array handling for optimal use.