

InterpolationTable

EcoSwarm software developed by:
Steve Railsback, Lang Railsback & Assoc. (LRA@Northcoast.com)
Steve Jackson, Jackson Scientific Computing (jackson3@humboldt1.com)

Overview and Conventions

The InterpolationTable class performs simple linear and logarithmic interpolation and can serve as a lookup table. The user provides a series of X-Y pairs that define a function (the “interpolation function”). The interpolationTable can then return a Y value for any input X value.

InterpolationTable uses the EcoSwarm class ZoneAllocator (which allows memory used by InterpolationTable to be dropped between replicate simulations). ZoneAllocator must be included in any model that uses InterpolationTable.

The following formats and conventions are used by InterpolationTable.

- InterpolationTable requires the X-Y points used to define the interpolation function to be added in order of ascending X value. This approach has the advantage of making it more likely that errors in the input to InterpolationTable are found. (However, the InterpolationTable’s **addX: Y:** method could easily be modified, using Swarm ListIndex’s **addAfter:** method, so points can be added in any order.)
- If asked to find a Y value for an X that is larger than the largest X in the interpolation function, the returned Y value is extrapolated from the highest two pairs in the interpolation function. Likewise, the Y value returned for an X that is lower than the first X in the interpolation function is extrapolated from the lowest two pairs in the function.
- If logarithmic interpolation is selected, all X and Y values provided to define the interpolation function are converted to their natural logarithms. The Y value for an input X is determined by taking the X input’s logarithm, doing the interpolation, then returning the antilog of the interpolated value. Negative or zero values cannot be used for X or Y when logarithmic interpolation is used.

The interpolation function is defined by calling the method **addX: Y:** for each point used to define the function. An InterpolationTable can be re-used with a new function by using the `reset` method.

Creating

+ **create: aZone**

Creates and returns an InterpolationTable object.

Setting

addX: (double) *anX* **Y:** (double) *aY*

Adds a pair of X-Y values to the interpolation function. This method is called repeatedly to add pairs to the function. Pairs must be added in order of increasing X values: if *anX* is less than or equal to the largest X value already in the function, an error is raised. Additional X-Y pairs can be added at any time.

reset

Removes all points from the interpolation function. This method is provided so an InterpolationTable can be cleared out and re-used with a new function. If logarithmic interpolation has been set on (see following method), it is turned back off.

setLogarithmicInterpolationOn

Causes the InterpolationTable to use logarithmic interpolation. The X, Y pairs in the interpolation function at the time this method is called are converted to their logarithms. Any new pairs are automatically converted to logarithms when added. When **getValueFor:** *anX* is called, *anX* is converted to a logarithm, a Y value interpolated, then the antilog of the interpolated Y value is returned.

If **setLogarithmicInterpolationOn** is executed after it has already been executed once, the message is ignored. An error will be raised if any zero or negative values are in the interpolation table when **setLogarithmicInterpolationOn** is executed, or if subsequently an attempt is made to add zero or negative values via **addX:** **Y:**, or to interpolate a zero or negative X value.

Using

- (double) **getValueFor:** (double) *anX*

Returns an interpolated value for *anX*. An error is raised if there are fewer than two data pairs in the interpolation table.

- (int) **getTableIndexFor:** (double) *anX*

Returns the interpolation table index for *anX*. This index is the position in the list of X, Y pairs (created by **addX:** **Y:**) of the higher of the two pairs used to interpolate the value of *anX*. The value can be between one (*anX* is lower than the second X in the table, so its value is interpolated or extrapolated from the pairs at positions 0 and 1) and one minus the number of pairs in the interpolation table. This method is used with **getValueWithTableIndex:** **withInterpFractionFor:**, as explained below.

An error is raised if there are fewer than two data pairs in the interpolation table. If logarithmic interpolation has been set on, then the natural log of *anX* is used to find the table index value; an error is raised if *anX* is zero or negative.

- (double) **getInterpFractionFor:** (double) *anX*

Returns the interpolation fraction for *anX*. This is the fraction that *anX* is between the interpolation table X values above and below *anX*. The value is normally between zero and one, but can be negative (when *anX* is lower than the first X in the table) or greater than one (when *anX* is higher than the last X in the table). This method is used with **getValueWithTableIndex: withInterpFractionFor:**, as explained below.

An error is raised if there are fewer than two data pairs in the interpolation table. If logarithmic interpolation has been set on, then the natural log of *anX* is used to find the table index value; an error is raised if *anX* is zero or negative.

- (double) **getValueWithTableIndex:** (int) *anIndex*
 withInterpFraction: (double) *aFraction*

Returns an interpolated value for the specified interpolation index and fraction. This method (and the two preceding methods used to calculate *anIndex* and *aFraction*) is useful when a model has many objects that each have interpolation tables that use the same X values. For example, each habitat cell in an ecological model may have a similar interpolation table for how primary production (Y) varies with day length (X); the tables for all cells have the same X values but different Y values. Each day all the habitat cells can have their primary production updated efficiently: use **getTableIndexFor:** and **getInterpFractionFor:** once for the new day length, then execute the **getValueWithTableIndex** method for each cell.

If logarithmic interpolation has been set on, then the antilog of the interpolated Y value is returned.

Document History

First version: SFRailsback, 4/8/02.

Revised and checked against code: SFRailsback, SJackson, 5/8/03.

Revised and checked against code: SFRailsback, SJackson, 6/30/04.