CS 524 Task 1: Multidimensional Linear Interpolation Data Lookups

Description

This task serves as a warm-up exercise to help connect the lecture discussions on software quality assurance to something concrete. It involves defining and accessing data that would be typical in any application. It has no specific context; rather, it is a general-purpose data structure for performing lookups on three kinds numerical tables. All are a variation on linear interpolation, which Lecture 2 covers.

A lookup takes one or more independent variables as the inputs and returns the single corresponding dependent variable as the output.

Setup

Download the appropriate version of Gnuplot for your platform at https://sourceforge.net/projects/gnuplot/

Class `Simulation` generates the Gnuplot scripts that visualize the output of all three lookups over the complete range of independent variables. Type `load 'filename'` in Gnuplot to execute each. This class also includes methods to test individual independent values. Start with them to verify your solutions one piece at a time.

Specifications

Use the skeleton code provided. You may add anything you want, but do not change the method signatures. There is only one implementation for the linear interpolation, which belongs in `A_Lookup`. It works as follows:

> Given the value `independentVariable` to look up, determine its position as a percentage between the values before it in `independentVariableOpen` and after it in `independentVariableClose`. They come from the data table lookups defined below, but they may also be passed by hand for initial testing. Make sure this code works before attempting the rest of the task. Everything else depends on it.
>
> The range of this independent block is determined by:
>
> ```
> independentVariableClose - independentVariableOpen
> ```
>
> The percentage is determined by shifting the independent variable down onto the range and scaling it:
>
> ```
> (independentVariable - independentVariableOpen) / rangeIndependent
> ```
>
> This value must always lie between 0 and 1 inclusively. If not, then you provided something incorrect for the independent components.
>
> Similarly, the range of the dependent block is determined by:
>
> ```
> dependentVariableClose – dependentVariableOpen
> ```
>
> Scaling the percentage onto the dependent block and shifting up produces the dependent variable:
>
> ```
> (rangeDependent * ratioIndependent) + dependentVariableOpen
> ```
>
> As a reality check, setting `independentVariable` to `independentVariableOpen` should produce `dependentVariableOpen`. Likewise, `independentVariableClose` produces `dependentVariableClose`. Halfway between `independentVariableOpen` and `independentVariableClose` is halfway between `dependentVariableOpen` and `dependentVariableClose`.

The lookup subclasses reference this interpolation method. Their responsibility is to determine the five arguments and manipulate the corresponding dependent variable(s) to produce a single result. Specifically, one-dimensional interpolation calls `interpolate` once, two-dimensional three times, and three-dimensional seven times.

Think carefully about what you are trying to accomplish. Draw pictures. The math and code are simple, but getting them into the right form can be confusing. Make sure you understand what is supposed to be happening. Keep track of your thoughts and actions in a log because we will discuss not only what happened with this task, but also the process of developing and especially testing it.

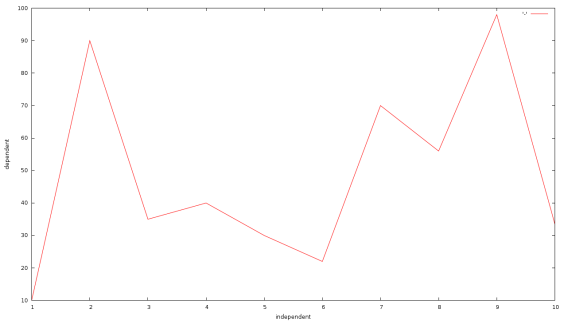
Part I: One-Dimensional Lookup

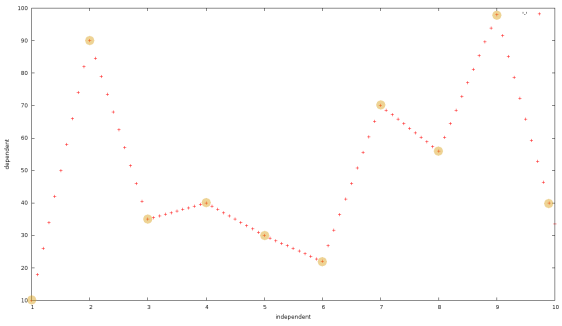
Class `Lookup1D` loads data file `data1d.csv`. The first line is the number of rows in the table. The format is `independentVariable,dependentVariable`. (Use the actual files, not these examples, just in case something changes later.)

```
10
1,10
2,90
3,35
4,40
5,30
6,22
7,70
8,56
9,98
10,33.5
```

The open/close block that contains the independent variable defines the open/close block of the dependent variable. Call `interpolate` with these arguments.

The Gnuplot output is generated by varying the independent variable from 1 to 10 by 0.1. In the left one, Gnuplot is connecting the interpolated data points to make lines. In the right one, only the points are shown. A smaller step size would result in more points closer together. Linear interpolation in theory provides infinitely small resolution from a limited number of defined points (the orange dots). In practice, the `double` datatype in Java has finite limitations, but the resolution is normally more than adequate. We have a later task that investigates accuracy and precision.





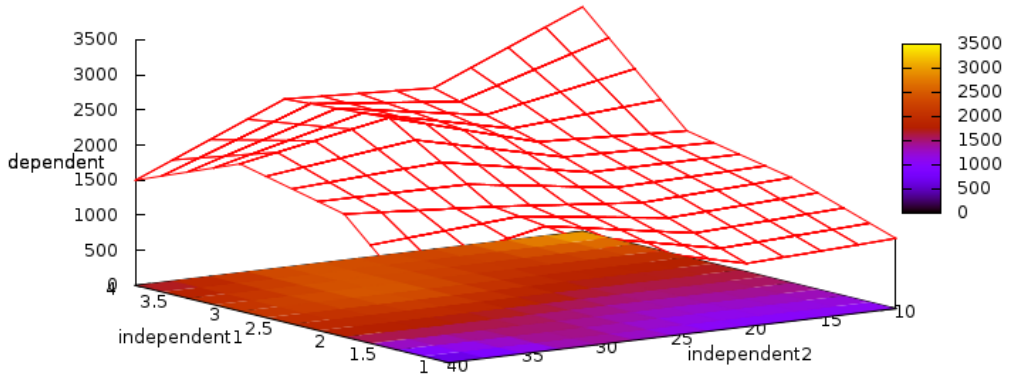
Part II: Two-Dimensional Lookup

Class `Lookup2D` loads data file `data2d.csv`. The first line is the number of rows of data in the table, and the second line is the number of columns (not including the headers). The first row of the table (line three of the file) defines independent variable 2. Ignore its first column; it is only a placeholder and not used in any way. The subsequent rows have independent variable 1 in the first column and the dependent variables corresponding to independent variable 2 in the subsequent columns.

```
4
4
0,10,20,30,40
1,1000,900,1410,350
2,1500,1400,1600,1750
3,1800,1900,2500,2100
4,3200,2300,2400,1500
```

The intersection of the vertical open/close block that contains independent variable 1 and the horizontal open/close block that contains independent variable 2 defines the square bounds where the dependent variable lies. Interpolate vertically between the open/close range of independent variable 1 at the open range of the independent variable 2. Then do the same at the close range. Finally, interpolate horizontally between these two values to get the dependent variable. (The opposite order, horizontally then vertically, produces the same result.)

The Gnuplot output is generated by varying the independent variables from 1 to 4 by 0.25 and 10 to 40 by 2.5:



You can disable the color map by omitting `set pm3d at b`

Part III: Three-Dimensional Lookup

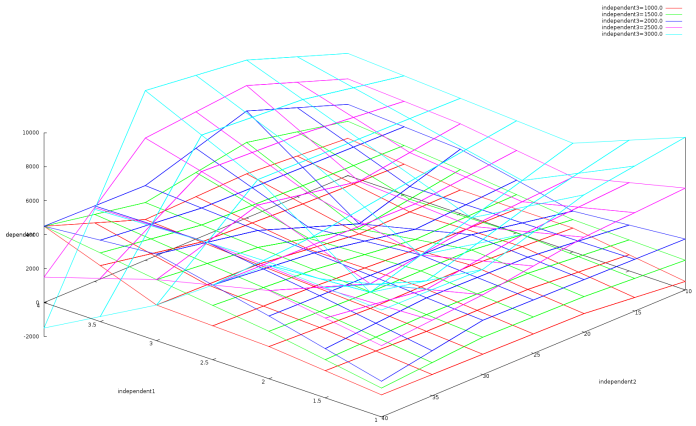
Class `Lookup3D` references multiple files. It is not possible to create a three-dimensional table in a text file without breaking it into separate layers somehow. To avoid a third type of file loader, this lookup uses a collection of two-dimensional files. In fact, they are the same file formats from Part II. To capture the third dimension, this lookup reads `data3d.csv`. The first row defines the number of layers, which are the two-dimensional tables. Each subsequent row defines the layer. The first column is independent variable 3, and the second is the filename of the table defining independent variables 1 and 2 and their dependent variable as defined in Part II. The approach conveniently reuses the solution from Part II.

```
3
1000,data3d1000.csv
2000,data3d2000.csv
3000,data3d3000.csv
```

If done strategically, the solution is actually quite small, but it does require managing a more complex data structure. The process is as follows:

Start with independent variable 3. Find the open/close range in `data3d.csv` and use the two-dimensional lookup on their files to get the two interpolated values on those planes with independent variables 1 and 2. As described above, each performs three interpolations, which accounts for six of the seven total. Then interpolate one last time between these values with independent variable 3.

The Gnuplot output is generated by varying the independent variables from 1000 to 3000 by 500, from 1 to 4 by 0.5, and 10 to 40 by 5:



It is difficult to make sense of this visualization. The `testLookup3D` method lets you play with individual points.

Miscellaneous

The location of the data files referenced in `Simulation` and `data3d.csv` depends on your environment. Change the paths as necessary.

You may assume that all data files are correctly formatted and contain valid data. Independent variables are always in increasing order.

Throw a `RuntimeException` if any independent variable is not defined in the table. We are doing interpolation only, not extrapolation.

Deliverables

Submit the following files (zipped is ok):

- Your Java code for everything except `Simulation`.
- A nicely formatted PDF document that captures the Gnuplot graphs for the tests generated by `Simulation`.
- Your development journal. The format is your choice, but make sure it captures your thoughts and actions throughout the software development process, which starts with reading this requirements/specifications document.

Do not submit the `.gnu` files.