Lab 12: Interpolation

12.1 Introduction

This (mini)lab focuses on using interpolation to estimate values of functions. For some of the problems you will be presenting graphical representations of the interpolations as well as calculating estimates.

Each problem has a component on the Connect system - typically you will be entering the values of estimates. For some of the problems, you will also be required to generate a graph. These graphs should be saved as .png files. Each problem will specify names for the scripts and for the graphs - please be sure to use these names to assist with creating the lab document and with grading. The lab document is already *completely done* other than the need to remove comments from the file and graphics imports. You will be turning in your PDF and zip as usual.

12.2 Getting Started

This is an individual exercise - you may ask the TAs or instructors for assistance but you are not to discuss these problems with classmates. Once you have logged in to a computer, go to:

https://pundit.pratt.duke.edu/wiki/EGR_103/Fall_2021/Beginning_of_Lab

and follow those instructions. Then open a browser to

https://pundit.pratt.duke.edu/wiki/EGR_103/Fall_2021/Lab_12

12.3 Assignment

12.3.1 Chapra 18.9, p. 481

Script name: `chapra_18_009.py`
Graph name: `chapra_18_009_plot.png`

For your code, after the imports you will want to create two arrays for your data set. Also create an array of model T values that spans the same domain with 100 linearly spaced points – you will be using this for the plots. Have your code print out the values of the estimates of $o(27)$, then make a plot with four different items: the original data set as magenta circles, the linear interpolations as a solid red line, the polynomial interpolation as a dashed green line, and the cubic spline interpolation as a dotted blue line. Your graph should have a legend with entries for "Data," "Linear," "Polynomial," and "Cubic Spline," but does not need a title or axis labels. Note that the latter two interpolations for this data set end up being very similar such that the lines you plot will be almost exactly the same. The linear interpolation is generally close but has a different shape between the points.

12.3.2 Chapra 18.10, p. 481

Script name: `chapra_18_010.py`
Graph name: `chapra_18_010_plot.png`

For your code, after the imports you will want to create two arrays for your data set. Also create an array of model x values that spans the same domain with 100 linearly spaced points – you will be using this for the plots. Have your code print out the values of the estimates of $y(1.5)$, then make a plot with three different items: the original data set as magenta circles, the default cubic spline interpolation as a solid red line, and the cubic spline with clamped end conditions as a dotted blue line. Your graph should have a legend with entries for "Data," "Cubic Spline," and "Clamped," but does not need a title or axis labels. For this plot, notice that there are clear differences between the two interpolants in the domain between the first two data points, but that those differences gradually disappear. On the right side of the graph, the default cubic spline already seems to have a derivative of nearly zero so that last interval does not show much deviation between the two interpolations.

12.3.3 Chapra 18.14, p. 482

Script name: `chapra_18_014.py`

Two dimensional interpolation in Python is a little tricky because of the way the `interp2d` command wants the independent variable information. See the "Two Dimensional Interpolation" section on the Pundit page for Python:Interpolation for more details.

For this problem, you are going to be creating your own data set using a pre-determined formula. You will then compare the results of two different types of interpolation with those data sets to the actual value generated by the formula. The wording of the problem may be confusing, so here is a summary of what your code should do:

- Import what you need.
- 2D Interpolation wants the independent data as two arrays, one containing the unique x values and one containing the unique y values.
  - Create a linearly spaced array of nine values for x between -2 and 0.
  - Create a linearly spaced array of nine values for y between 0 and 3.
- 2D interpolation wants the dependent data as a matrix created by assuming the x value changes as the column changes and the y value changes as the row changes. The easiest way to create this is:
  - Use `np.meshgrid` to create the matrix versions of your x and y values. Be sure to give these different names from the arrays you made above, but note that the arrays you made above will be the arguments of the `np.meshgrid` function.
  - Define a function that calculates the temperature as a function of x and y.
  - Use that function with the *matrix* versions of x and y to create a matrix version of the temperatures.
- Create a two dimensional interpolation function with the *array* versions of x and y, the matrix version of the temperature, and the kwarg `kind='linear'`.
- Use this function to make your prediction.

You will not be required to make a graph for this problem, but please read the section on Sakai about using two dimensional interpolation to make graphs. The process is a bit more complicated than it needs to be simply because the function created by `interp2d` only accepts one dimensional arguments as inputs. That means you need to take a two dimensional set of independent values, "flatten" them to make the one dimensional, calculate the interpolations (which results in a one dimensional array), and then reshape that to a two dimensional array in order to make plotting a surface work.