Interpolation

Due at 4:00pm on Friday 15 November 2019

What you need to get

- **https://www.overleaf.com/read/vztpkwrpmtcn**: (optional) Overleaf template for Q1 and Q2
- `YOU_a4q3.ipynb`
- `YOU_a4q4.ipynb`

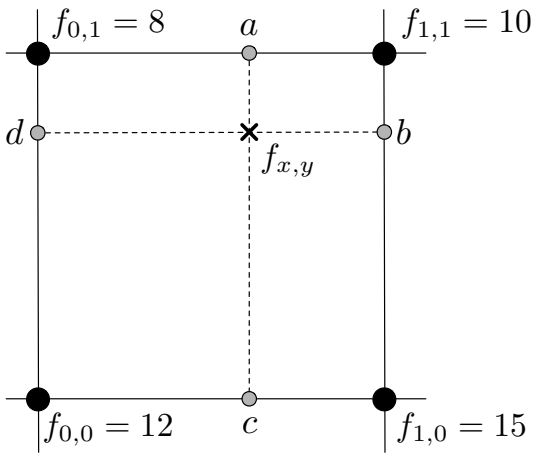
What to do

1. **[5 marks]** Consider the following alternative representation for a cubic spline, $S(x)$:

$$S(x) = \begin{cases} 20 + 25x + 18x^2 + a_1x^3 & -3 \le x < -1 \\ 26 + a_2x + a_3x^2 + a_4x^3 & -1 \le x < 0 \\ 26 + 19x + a_5x^2 + x^3 & 0 \le x \le 3. \end{cases}$$

   Is it possible to find values for $\{a_1, a_2, a_3, a_4, a_5\}$ so that $S(x)$ is a natural cubic spline? Justify your answer.

2. **[5 marks]** The figure shows four samples of a function f on a 2D domain. However, you would like to interpolate it at (x, y) to get a value $f_{x,y}$, where $0 < x < 1$ and $0 < y < 1$. There are three strategies for doing this:



   (a) Interpolate horizontally between $f_{0,1}$ and $f_{1,1}$ to get a, interpolate horizontally between $f_{0,0}$ and $f_{1,0}$ to get c, and then interpolate vertically between those two values to get the value $f_{x,y}$ at (x, y).

   (b) Interpolate vertically between $f_{0,0}$ and $f_{0,1}$ to get d, interpolate vertically between $f_{1,0}$ and $f_{1,1}$ to get b, and then interpolate horizontally between those two values to get the value $f_{x.y}$ at (x, y).

   (c) Take a weighted sum of the f-values, where the weight for each f-value is given by the area of the diagonally opposite rectangle. For example, the weight for $f_{0,1}$ is $(1 - x)y$.

   Prove that all three methods give the same value $f_{x,y}$. Show your work.

3. **[7 marks] Complete the Python function** `MySpline` that reads in a set of `x` and `y` values (each as arrays or lists), and outputs a piecewise-cubic function with natural boundary conditions. The function prototype is

```
cs = MySpline(x, y)
```

   The returned function can be called using `cs(2.4)` or `cs([2.4, 3.7])`, for example. You will have to find the parameter values for the as, bs, and cs so that the cubic spline can be evaluated

using

$$p_i(x) = a_i \frac{(x_{i+1} - x)^3}{6h_i} + a_{i+1} \frac{(x - x_i)^3}{6h_i} + b_k (x_{i+1} - x) + c_i (x - x_i) \ ,$$

where $h_i = x_{i+1} - x_i$ and $i = 1, 2, \ldots, n-1$. Note that all Python indexing starts at 0, so all the indices can be decremented by 1. However, some care is needed to handle $a_1, \ldots, a_n$. See the documentation for `MySpline` for some guidance on this issue.

The function `MySpline` is already set up to return a cubic spline function. But it's not a very interesting one, and it does not pass through the points (x_i, y_i).

The notebook has a small sample set of points to interpolate. Feel free to choose your own set of points. **Create a figure that plots the interpolation points overtop of the smooth cubic spline.**

4. **[8 marks]** Create a parametric curve representation of your nickname written in your handwriting. This representation should be based on parametric spline interpolation. Your nickname must have at least two curved segments. Figure 1 below shows an example using 5 coarsly-sampled segments.

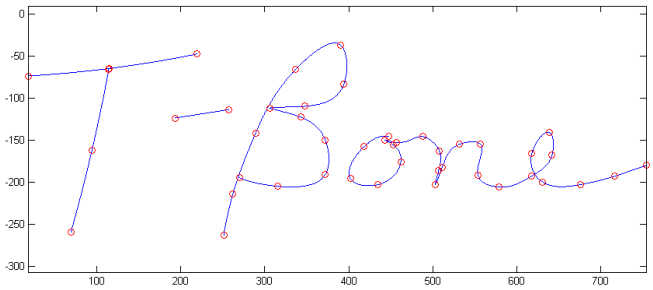


Figure 1: The nickname "T-Bone" written with 5 parametric cubic spline segments

Steps:

(a) Draw your nickname and select interpolation points. You can do this on graph paper by hand, or you can use `ginput`. Make sure the interpolation points are not too close together. Notice that the loops of the "B" in Figure 1 are made from only 6 points each. Your curved segments should be sampled sparingly.

(b) Add code to the notebook to initialize and store the interpolation points for each segment. For example, you could store the coordinates of the interpolation points for the first segment in the arrays `x1` and `y1`, and you would have one `x-y` pair of arrays for each segment. Your data arrays should be hardcoded into your script.

(c) Complete the function `ParametricSpline`; it takes the interpolation points of one segment as input, and outputs a pair of cubic spline functions (one for `x(t)` and one for `y(t)`), along with an array, `t`, that holds the parameter value at the interpolation points. See the function's documentation for more details. The parameter `t` must reflect the pseudo-arclength of the curve. You can use `scipy.interpolate.make_interp_spline` (and return the spline functions that it creates). Alternatively, if you feel confident in your `MySpline` function, you can use that instead.

(d) Add lines to the notebook to call `ParametricSpline` for each segment in your nickname.

(e) Add more lines to to the notebook to plot your nickname. The plot should graph the parametric spline segments using a refined selection of parameter values (eg. 1000 points per

segment). The plot should also display the original interpolation points, and the `x` and `y` axes should be rendered using the same scale ("`plt.axis('equal')`"). Figure 1 is an example of what your output should look like.

What to submit

You must submit a series of PDF documents to Crowdmark. For each coding question, export your jupyter notebook to PDF. When a proof or manual calculation is requested, you can typeset your solution (eg. using LaTeX or Word), or write your solution by hand. In either case, your solution should also be submitted as a PDF. **If you submit photos of handwritten work, it is your responsibility to ensure that they are legible.**