Numerical analysis: Assignment 5

Niccolo Zuppichini

October 25, 2020

Exercise 1

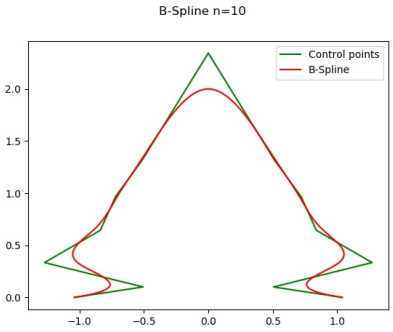
We want to interpolate n + 1 points P0 ... Pn given n + 7 knots points t_i. In order to do this we need do compute the control points D. The control points are calculated by solving the system $MD = P$ where M is a tridiagonal matrix containing the B-spline basis.
In order to compute the basis the recursive definition has been employed.

$$N_i^0(t) = \begin{cases} 1 & \text{for} t \in [t_i, t_{i+1}) \\ 0 & \text{else} \end{cases} \qquad \text{for} i = 0, ..., 2n + m \tag{1}$$

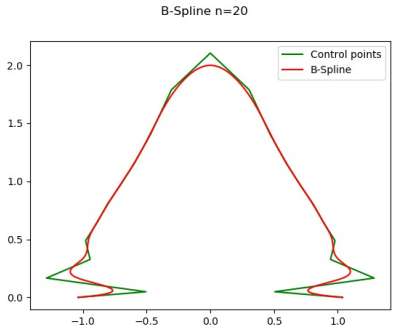
$$N_i^j = \frac{t - t_i}{t_{i+j} - t_i} N_i^{j-1} + \frac{t_{i+j+1} - t}{t_{i+j+1} - t_i} N_{i+1}^{j-i} \tag{2}$$

Then we populate the tridiagonal matrix M using those basis functions. We have a tridiagonal matrix because each basis N has local support.

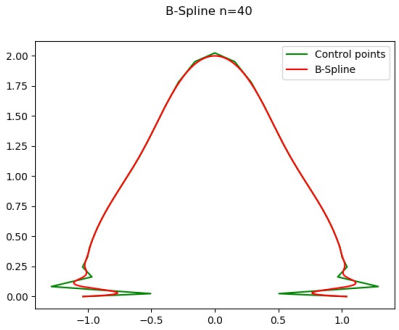
I was not able to compute the natural end conditions, therefore the first and last rows of the matrix are missing and so the matrix is not square. This means that I can't use the Thomas algorithm to solve it efficiently. However, I've implemented the tridiagonal solver (code in *code.py*) anyway, but since my matrix is missing 2 rows I haven't used it (after employing the natural conditions it can be used). I've solved the system using the least square method included in the scipy library, instead.
After evaluating the control points, I've De Boor's algorithm to evaluate the b-spline.



(a) n=10



(b) n=20



(c) n=40

Figure 1: Result from my python implementation. The red curve is my b-spline and the green polygon is the control polygon.

Figure **??** shows that the function interpolates the points correctly. Moreover, it is more stable than the interpolation polynomial computed in exercise 4 (which had stability problems at the boundaries).

Exercise 2

Unfortunately I ran out of time.