Numerical Computations - Ex 3, Oct 29, 2020

9. Impement B-spline basis functions.

   [Lecture from 20.10, or Dahmen+Reusken, Chap 9] Knots are $t_0 \leq t_1 \leq \ldots \leq t_n$.

$$\begin{aligned} N_{j,1}(x) &= \chi_{[t_j,t_{j+1})}(x) \\ N_{j,k}(x) &= \frac{x - t_j}{t_{j+k-1} - t_j} N_{j,k-1}(x) + \frac{t_{j+k} - x}{t_{j+k} - t_{j+1}} N_{j+1,k-1}(x) \end{aligned}$$

   for $j = 0, \ldots n-k-1$ and $k = 2, \ldots n-1$.

   (a) What is the complexity of a recursive evaluation in a given point x using this definition ? What is the complexity if multiple evaluation of the same term is avoided ? Searching the interval $[t_j, t_{j+1})$ containing x is an separate issue.

   (b) Plot the basis functions once for knots $t_j = j$, and then for using k multiple knots at both ends, i.e. $t_0 = \ldots = t_{k-1}$ and $t_{n-k+1} = \ldots = t_n$.

   (c) Plot $\sum_{j=0}^{n} N_{j,k}$

   (d) How do basis functions look like for multiple internal knots ?

10. Implement the de Boor algorithm (Dahmen+Reusken Alg 9.12) for evaluating spline functions $\sum c_j N_{j,k}(x)$. Familiarize yourself also with the scipy.interpolate.splev function and related functions.

    Consider the curve given by $\gamma : [0,1] \to \mathbb{R}^2 : t \mapsto ((t+1)\cos(4\pi t), (t+1)\sin(4\pi t))$.

    Choose control points $p_j = \gamma(j/n)$, and plot the spline functions

$$t \mapsto \sum_{j=0}^{n} p_j N_{j,k}(t)$$

    for $k = 1, 2, 3, 4$. Choose uniformly distributed knots, with k multiple knots at the ends.

    Note that choosing the control points at the curve does not lead to optimal approximation of the curve by the spline - curve.

11. Given a B-spline function of order k, with knots t_i and coefficients c_i. Implement a function to compute the derivative as a B-spline function of order $k-1$.

    Test your function for computing tangent vectors to the curve from Ex 10.

12. Given are values $f_j = f(x_j)$ for $x_j = j/n$, with $n = 2^p$. Compute the coefficients of the Discrete Fourier Transfrom (DFT)

$$a_k = \frac{1}{n} \sum_{j=0}^{n-1} e^{\frac{2\pi ijk}{n}} f_j$$

    using the FFT algorithm from scipy.

(a) How big can you chose n such that the run-time for the FFT is below 1 second on your computer ? How long would a straight forward evaluation of the DFT of the same size by a matrix-vector product take ? How much memory would be required ?

(b) How are the coefficients a_k and a_{n-k} related ?

(c) Test the functions $f : [0,1] \to \mathbb{R}$:

$$\begin{aligned}
f_1(x) &= \sin(40\pi x) \\
f_2(x) &= \chi_{[0.25,0.75)} \\
f_3(x) &= \min\{x, 1-x\} \\
f_4(x) &= e^{-100(x-0.5)^2} \\
f_5(x) &= e^{-4(x-0.5)^2} \\
f_6(x) &= e^{-100(x-0.5)^2}\sin(40\pi x)
\end{aligned}$$

How fast do coefficients fall for $i \in [0, n/2)$, i.e. find C and β such that $|a_i| \leq Ci^{-\beta}$, by visual inspection ? Try to explain your observations.