Lab exercises on polynomial interpolation

Course on Mechanical Engineering, AY 2017-18

Prof. S. De Marchi

Padova, April 24, 2018

1 Bernstein polynomials and Bézier curves

We recall that the Berstein polynomial approximating a function f in $[0,1]$ is

$$b_n(x) = \sum_{k=0}^{n} f\left(\frac{k}{n}\right) \beta_k^n(x) \tag{1}$$

where $\beta_k^n(x) = \binom{n}{k} x^k (1-x)^{n-k}$ indicates the k-th Bernstein basis polynomial of degree n.

Moreover, given the set $P = \{P_0, \ldots, P_n\}$ (called the *control polygon*) of $n+1$ bi-dimensional points $P_k = (x_k, y_k)$, the *Bézier curve* of degree n is

$$\mathcal{B}_n(t) = \sum_{k=0}^{n} P_k \beta_k^n(t) = \begin{pmatrix} \sum_{k=0}^{n} x_k \beta_k^n(t) \\ \sum_{k=0}^{n} y_k \beta_k^n(t) \end{pmatrix}, \quad t \in [0,1]. \tag{2}$$

1.1 Exercises

- Consider the function $f(x) = x(1-x), \quad x \in [0,1]$. Write a Matlab script that constructs the Berstein polynomial approximating f.

  To do this, we suggest to write a function `bern_poly.m` that given in input the vector of evaluation points `x`, the degree `n` and the index `k`, returns the vector, say `b`, consisting of the k-th Bernstein basis polynomial of degree n, evaluated at `x`.

- Consider the set `P=[ 0 0; 0.5 0.8; 1 1; 2 1; 2.5 0.8; 3.5 0]`. Construct the Bézier curve of degree 5 approximating the point set `P`. Moreover construct the curve

$$\mathcal{B}_{n,\mathbf{w}}(t) = \begin{pmatrix} \frac{\sum_{k=0}^{n} w_k x_k \beta_k^n(t)}{\sum_{k=0}^{n} w_k} \\ \\ \frac{\sum_{k=0}^{n} w_k y_k \beta_k^n(t)}{\sum_{k=0}^{n} w_k} \end{pmatrix}, \quad t \in [0,1]. \tag{3}$$

  where the weights `w=[0 1/2 1/2 1/2 500]` . What's about the vector `w1=[3 3 3 3 3 3]`?

2 Least-square approximation

As we have seen, consider the points set $X = \{(x_i, y_i),\, i = 0, \ldots, n\}$. Fix a $m \ll n$. The approximating polynomial

$$p_m(x) = \sum_{i=0}^{m} a_i x^i$$

is determined as the solution of *normal equations*

$$A^T A\mathbf{x} = A^T\mathbf{y} \tag{4}$$

where the rectangular $(n+1) \times (m+1)$ matrix A has components $A_{k,j} = (x_k^{n-j+1})$, $k = 1, \ldots, n+1,\ j = 1, \ldots, m+1$.

- Consider the point set `[x,y] = titanium` (which returns measurements of a certain property of titanium as a function of temperature). Approximate the titanium data with polynomials of degree 30 in the least-squares sense, that is by solving the normal equations (4). As evaluation points consider the set `t=linspace(min(x),max(x),1000)`. Compare the results by using the functions `polyfit` and `polyval`. Compute also the *relative errors*.
- Approximate the values with the cubic spline, given by command `s= spline(x,y,t)`. What do you see?

Time: **2 hours**.