Lab exercises on polynomial interpolation

Course on Mechanical Engineering, AY 2015-16

Prof. S. De Marchi

Padova, May 2, 2016

1 Useful Matlab commands: `polyfit` and `polyval`

- Of the command `polyfit` we take into account only the case

  `p = polyfit(x,y,n)`

  In detail: `p=polyfit(x,y,n)`, returns in the vector `p`, the coefficients of the polynomial of degree $n \le$ `length(x)` that **approximates**, in the least-square sense the data in `y`.

- The command `y=polyval(p,x)`, allows to evaluate at the vector `x` a polynomial whose coefficients are stored in the vector `p` (returned by `polyfit`), that is

$$y = p_1 x^n + p_2 x^{n-1} + \ldots + p_{n+1}\,.$$

2 Interpolating polynomial in Lagrange form

Given $n+1$ couples $\{x_i, y_i\}, \quad i = 1, \ldots, n+1$, the interpolating polynomial of degree n in Lagrange form is

$$p_n(x) = \sum_{i=1}^{n+1} l_i(x) y_i \tag{1}$$

where l_i is an `elementary Lagrange polynomial` of degree n defined as

$$l_i(x) = \prod_{i=1, i\neq j}^{n+1} \frac{x - x_j}{x_i - x_j}\,.$$

Let us observe that (2) can be seen as the `scalar product` between the vectors $\mathbf{y} = (y_1, \ldots, y_{n+1})^T$ and $\mathbf{l} = (l_1(x), \ldots, l_{n+1}(x))^T$.

As just observed, in the **evaluation** of $p_n(x)$ on a set of *target points*, $\bar{x}$, which are in general different from the interpolation points $\{x_i\}$ and in a bigger number (think when you need to plot the p_n or its error estimation), it will be necessary having a function that allows to evaluate the i-th elementary Lagrange polynomial, l_i at the vector $\bar{x}$. To this aim, by means of the command `repmat`, we can use the following function

```
function l = lagrange(i,x,xbar)
%--------------------------------
% INPUTS
% i=index of the polynomial
% x=vector of interpolation points
% xbar= vector of target points
%      (column vector!!)
%
% OUTPUT
% l=vector of the ith elementary
%   Lagrange polynomial at xbar
%--------------------------------
n = length(x); m = length(xbar);

l = prod(repmat(xbar,1,n-1)-repmat(x([1:i-1,i+1:n]),m,1),2)/...
prod(x(i)-x([1:i-1,i+1:n]));
```

We have used the command `repmat` which makes copies of a matrix. As an example. Take the matrix `[1, 2; 3, 4]` and make a 2×2 copies of it, as follows

```
>> repmat([1,2;3,4],2,2)

ans =

     1     2     1     2
     3     4     3     4
     1     2     1     2
     3     4     3     4
```

NOTICE. Once we have constructed, by using the above function `lagrange.m`, the $n+1$ column vectors **l**, we collect them in a matrix, say L, and with the product **p=L*y** we then have the value of the interpolating polynomial `p` at all the target points.

2.1 Chebyshev and Chebyshev-Lobatto points

The Chebyshev points are the zeros of the *Chebyshev polynomial of the first kind*, they belong to the interior of interval $[-1, 1]$ and are so defined:

$$x_i^{(C)} = \cos\left(\frac{(2i-1)\pi}{2n}\right), \quad i = 1, \ldots n\,.$$

The ones of *Chebsyshev-Lobatto* consider also the extremals of the interval [-1,1] and are defined as follows:

$$x_i^{(CL)} = \cos\left(\frac{(i-1)\pi}{(n-1)}\right), \quad i = 1, \ldots n\,.$$

If the interval is not $[-1,1]$, say generally $[a,b]$, then by means of the linear transformation $g(x) = Sx + W$ we can map the points to the general interval $[a,b]$. For example, the Chebyshev points mapped on [a,b] are

$$\tilde{x}_i^{(C)} = \frac{a+b}{2} + \frac{b-a}{2} x_i^{(C)}$$

where $x_i^{(C)}$ are the Chebyshev points in $[-1,1]$. Similarly for the Chebyshev-Lobatto ones.

⋄⋄

Exercises

1. Construct the interpolating polynomial in Lagrange form, of degrees $n = 5, \ldots, 10$ of the *Runge function*

   $$g(x) = \frac{1}{1+25\,x^2}, \quad x \in [-1,1]$$

   on equispaced points. Make the plots of the function and its interpolating polynomials.

2. Construct the interpolating polynomial in Lagrange form of degree $n = 5, \ldots, 11$, of the Runge function

   $$g(x) = \frac{1}{1+25x^2}, \quad x \in [-1,1]$$

   on Chebyshev and Chebyshev-Lobatto points.

3. Take the `error function`,

   $$\texttt{erf}(x) := \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt\,,$$

   (in Matlab/Octave the built-in function is `erf`), on the set of equispaced points `x=(-5.0:1:5.0)'`. Find the coefficients of the approximating polynomial by `polyfit` with degrees that vary from 4 to 10. Then, use `polyval`, on a finer set of points, for the evalution of the polynomial. Why the fitting does not work?

4. Consider the function $f(x) = x + \mathrm{e}^x + \dfrac{20}{1+x^2} - 5$ restricted to the interval $[-2, 2]$.

   (a) Determine the interpolating polynomial of degree 5 in *Newton form* on the equispaced points $x_k = -2 + kh,\ k = 0, ..., 5$.

   (b) Compute the interpolation error at $x^* \in (-2, 2)$.

   (c) Repeat the calculations by using Chebyshev points.

5. Take the function

$$f(x) = \log(2 + x)\ , \quad x \in [-1, 1]\ .$$

Let p_n be the interpolating polynomial of degree $\leq n$ built using the Chebyshev points

$$x_k = \cos\left(\frac{2k+1}{2(n+1)}\,\pi\right),\ k = 0, 1, \ldots, n$$

In this case, it is known that the interpolation error can be bound as follows

$$\|f - p_n\|_\infty \leq \frac{\|f^{(n+1)}\|_\infty}{(n+1)!} 2^{-n}\ . \tag{2}$$

   (a) In the case $n = 4$, find an upper bound of the error using formula (2).

   (b) In the case the interpolating polynomial can be written in `Taylor form`

$$t_n(x) = f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}x^2 + \cdots + \frac{f^{(n)}(0)}{n!}x^n\ , \tag{3}$$

   the error at the generic point x can be expressible as

$$f(x) - t_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}x^{n+1}\ , \quad -1 < \xi < 1\,.$$

   Find a bound of the error

$$\|f - t_4\|_\infty = \max_{-1\leq x\leq 1} |f(x) - t_4(x)|\,,$$

   and compare the result with the case of Chebyshev points.

   (c) *free*: Plot in the same graph, $f(x)$, $p_4(x)$ and $t_4(x)$.

Time: **2 hours**.