Lab exercises

Prof. S. De Marchi

`Padova, 18th April 2017`

1 The commands: `polyfit` and `polyval`

- Of `polyfit` we consider only the case `p = polyfit(x,y,n)`.

  In detail: `c=polyfit(x,y,n)`, returns in the vector `c`, the coefficients of the polynomial of degree $n \leq$ `length`(`x`) that **approximates**, in the least-square sense the values in `y` sampled at the vector `x`.

  The coefficients are stored in the vector `c` so that

$$p = c_1 x^n + c_2 x^{n-1} + \ldots + c_{n+1}\,. \tag{1}$$

- Using the command `p=polyval(c,x)` we then evaluate at `x` the polynomial in (2).

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

$$p_n(x) = \sum_{i=1}^{n+1} l_i(x) y_i \tag{2}$$

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

$$l_i(x) = \frac{(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_{n+1})}{(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_{n+1})} = \prod_{i=1, i\neq j}^{n+1} \frac{x-x_j}{x_i-x_j}\,.$$

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.

3 Exercises

1. Figure 1 shows the plot of the "equation of time" given by a sundial (*meridiana* in Italian) that *looks like* a polynomial of degree 4 or 5.

   Reading the legend, the roots are located at the 4 days 15 April, 13 June, 1 September and 25 December.

   We want to find this polynomial by considering the interval $[1, 365]$ (the days of the year) and the points sets $X_1 = \{(1,4),(90,5),(135,-4),(275,-10),(365,8)\}$ and $X_2 = \{(1,4),(90,5),(135,-4),(214,6),(275,-10),(365,8)\}$

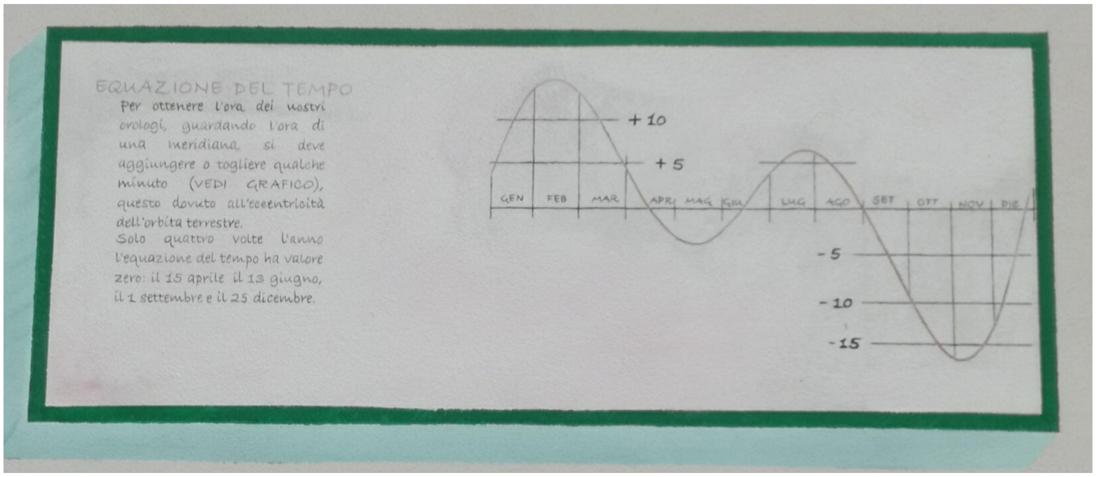

Figure 1: Equation of annual time done by a sundial. This photo can be seen at "Corte Civrana", Cona (Ve).

Using the function `p=polyfit(x,y,deg)` (where `x` and `y` are the vectors of the abscissas and ordinates of the set) and `p=polyval(p,xx)` find an approximation of the "equation of the time". Which set, between X_1 and X_2 does better represent the graph of the function?

Using the function `roots(p)`, find the roots of the approximating polynomial `p` and take their integer part.

What does it happen on taking $\tilde{X}_2 = \{(1, 3.5), (90, 5), (135, -4), (214, 5), (275, -9.8), (365, 10)\}$?

Comment the results.

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

$$g(x) = \frac{1}{1 + x^2}, \quad x \in [-5, 5]$$

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

Time: **2 hours**.