MATH 131: Numerical Methods for scientists and engineers – Discussion 5: Coding

The goals of this discussion section are:

- Get a deeper understanding of Lagrange interpolant and cubic splines.
  Use the codes you developed for Assignment 2.
- Get familiar with numerical differentiation.

1. **Pair-up with a classmate and** open the functions `Lagrange_poly` and `newtons_divided_differences` you developed during your homework assignment 2.
   (a) Using the Matlab functions `tic` and `toc`, compare the CPU time needed to execute the 2 functions to interpolate $f_1(x) = e^{-x^2}$, given `datx = [-3:dx:3]`, over `x=[-3:0.01:3]`, for `dx = 0.5, 0.2, 0.1, 0.05`.
   (b) Plot on the same graph the CPU time of each method with respect to number of nodes used in the interpolant. Comment.
   (c) Plot on the same graph the function, the interpolant obtained with `Lagrange_poly` and the one obtained with `newtons_divided_differences` for `datx = [-3:0.05:3]`. Comment on the result. What do you recommand to do to improve the result ?
2. If you didn't have time last week, download on Catcourses `cubic_spline.mlx`. Complete the `MATLAB` function, called `cubic_spline` that inputs a set of data points $(x, y) =$ `(datx, daty)`, x the numbers at which to interpolate, and outputs the cubic spline interpolant, S, evaluated at x using natural cubic spline interpolant:

   ```
   function S = cubic_spline(x,datx,daty)
   ```

   Use your textbook to complete the code. Use the code to interpolate $f_1(x) = e^{-x^2}$, given `datx = [-3:0.05:3]`, over `x=[-3:0.01:3]`. Compare with the previous exercise. What is the error bound when using a cubic spline interpolant ?
3. Using your function `Lagrange_poly`, create a Matlab function called `numerical_differentation` that takes as inputs a function f, a point x_0 and the number of points n needed to compute the $n+1$-point midpoint formula (see formula (4.2) p 174 in your textbook). Test for $f(x) = x^2 - 3x + 2$ at 0 for $n = 3, 5$. Comment on the result.