MATH 587: Numerical Analysis 1
Task 1: Due 27 September 2021
Shereen Elaidi (260727874)

0. The Lagrange Interpolation

The Lagrange interpolating polynomial is expressed as a linear combination of the following basis polynomials:

$$\ell_j(x) := \prod_{i=1, i\neq j}^{N} \frac{x - x_i}{x_j - x_i}.$$

The weights in the linear combination of the $\ell_j(x)$'s correspond to the actual values of the function we are trying to interpolate, which we'll denote $f(x_j)$. Therefore, the Lagrange interpolating polynomial $p(x)$ is given by:

$$p(x) = \sum_{i=1}^{N} f(x_i)\ell_i(x).$$

In this task we implement this interpolation in Python and test it on three functions:

$$u(x) = \sin(x),\ v(x) = \text{ Heaviside}(x),\ \text{and } w(x) = \frac{1}{10x^2 + 1}.$$

1. Plots for Equidistant Spacing

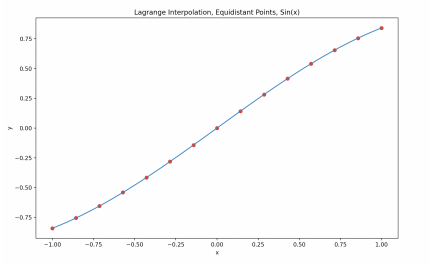

Figure 1: $f(x) = \sin(x)$.

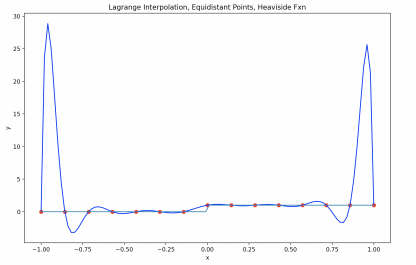

Figure 2: Heaviside function.

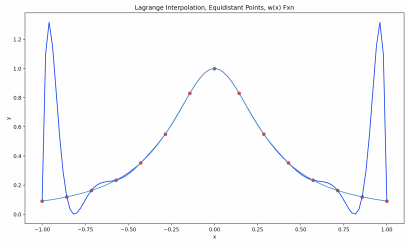

Figure 3: $f(x) = \frac{1}{10x^2+1}$.

The sharp blue colour corresponds to the plot of the interpolating polynomial. The other blue colour corresponds to the original function. We notice that in Fig. 1, the Lagrange fit is really good in $[-1, 1]$ compared to the fit in Fig. 2 and Fig. 3, especially at near the endpoints of $[-1, 1]$. This behaviour seems to be irrelevant of the smoothness of the function we are trying to interpolate: the interpolation by the endpoints is very poor for both a discontinuous function (the Heaviside function) and a very smooth function ($w(x) = \frac{1}{10x^2+1}$).

2. Plots for Chebychev Spacing

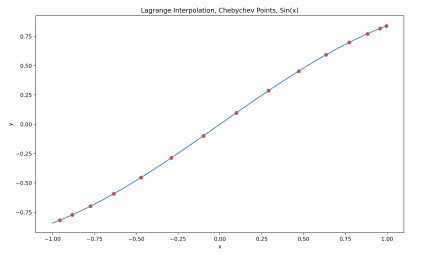

Figure 4: $f(x) = \sin(x)$.

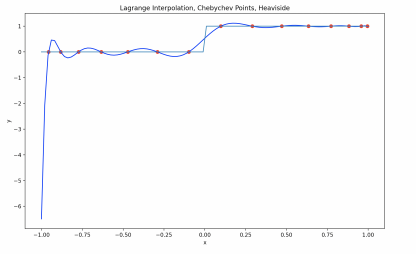

Figure 5: Heaviside function.

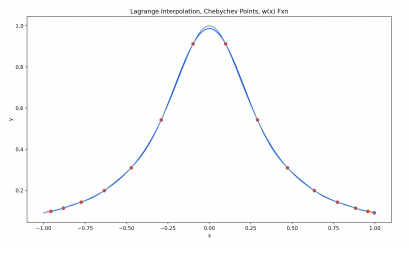

Figure 6: $f(x) = \frac{1}{10x^2+1}$.

For Chebychev points, we need to choose the roots of the 15th (since $N = 15$) Chebychev polynomial. From lecture, we know those are the following points:

$$\xi_j = \cos\left(\frac{(2j+1)\pi}{2N+2}\right),\ j = 0, ..., N. \tag{1}$$

Otherwise, the code is exactly the same. We immediately see that the interpolation at the end points for the Heaviside function (Fig. 5) and $w(x)$ (Fig. 6) performs much better than the equidistant point spacing in Fig. 2 and Fig. 3.

3. Appendix

The code for this assignment is in two files: the implementation of the Lagrange interpolation polynomial for a general case can be found in the file `lagrange.py` and the code to run our specific functions and specific point spacings can be found in the file `task1.py` (uploaded on myCourses). Just in case, both files are also on GitHub: `https://github.com/ShereenElaidi/math-578/tree/master/task-1`.