Exercises Laboratorio di Calcolo: Practicing SciPy + SymPy

Exercise 1

Piecewise linear interpolation has approximation order $O(h^2)$ where h is the maximal distance between the interpolation sites. This means that the error between any smooth function and its interpolant (measured in any L_q-norm, $1 \leq q \leq \infty$) behaves asymptotically like $O(h^2)$. Check this behavior by approximating the function $\sin(x)$ on the interval [0, 10] and measuring the error in the inf-norm ($q = \infty$).

1. Compute a sequence of piecewise linear interpolants. Choose the interpolation sites uniformly over the interval [0, 10] such that the maximal distance $h = 10 / 2^L$, for $L = 0, ..., 9$. Use the built-in SciPy function `interpolate.interp1d`.
2. Visualize the computed interpolants.
3. Compute the inf-norm of the error between $\sin(x)$ and all interpolants. This can be approximately done by taking a dense sampling of the error (say $N = 1000$ samples).
4. Visualize the convergence of the error in inf-norm, and show numerically that it behaves like $O(h^2)$. A semi-log plot is very useful here.

Exercise 2

A quadrature rule provides an approximation of the definite integral of a function, formulated as a weighted sum of function values at specified points within the domain of integration.

An n-point Gaussian quadrature rule, named after Carl Friedrich Gauss, is a quadrature rule constructed to yield an exact result for polynomials of degree $2n - 1$ or less by a suitable choice of the nodes x_i and weights w_i for $i = 1, \ldots, n$. The most common domain of integration for such a rule is [-1, 1], so

$$\int_{-1}^{1} f(x)\,dx \approx \sum_{i=1}^{n} w_i f(x_i),$$

which is known as the Gauss-Legendre quadrature rule.

Compute numerically the integral of the polynomial $x^{11} + 3x^4$ on the interval [0, π] in the following three ways:

1. Gauss-Legendre quadrature based on the nodes and weights provided by the built-in function `np.polynomial.legendre.leggauss` from the NumPy module `polynomial.legendre`;
2. Gauss-Legendre quadrature based on the nodes and weights provided by the built-in function `special.roots_legendre` from the SciPy module `special`;
3. Adaptive quadrature using the built-in function `integrate.quad` from the SciPy module `integrate`.

Then:

1. Compute the numerical error of the three quadrature implementations. The exact value of the integral is $\pi^{12}/12 + \pi^{5} 3/5$.
2. Time the three quadrature implementations, and check which one is the fastest.

Remark: a change of variable is necessary in the Gauss-Legendre quadrature cases, to match the domain of integration!

Exercise 3

Consider the n x n matrix T_n and the n x 1 vector b_n with the following structure:

$$T_n = \begin{bmatrix} 1 & -3 & -5 & -7 & \cdots \\ 2 & 1 & -3 & -5 & \cdots \\ 3 & 2 & 1 & -3 & \cdots \\ 4 & 3 & 2 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \qquad b_n = \begin{bmatrix} n \\ n-1 \\ n-2 \\ n-3 \\ \vdots \end{bmatrix}.$$

Then, compute the solution x_n of the linear system

$$T_n \, x_n = b_n$$

in the following two ways:

1. Solve this system numerically using the SciPy module `linalg`.
2. Solve this system symbolically using the module `sympy`.

Compare the 2 solutions.