EXERCISE SET 7 – MATH-250 Advanced Numerical Analysis I

There is quiz on this exercise sheet. The exercises marked with ($\star$) are graded homework. The exercises marked with **(Python)** are implementation based and can be solved in the Jupyter notebooks which are available on Moodle/Noto. **The deadline for submitting your solutions to the homework is Friday, April 11 at 10h15.**

Exercises

Problem 1. (Python)

The degree n best approximation in the L^2 norm is the polynomial $p_n^* \in \mathbb{P}_n$ that minimizes the L^2 approximation error

$$\|p_n - f\|_2 = \sqrt{\int_{-1}^{1} (p_n(t) - f(t))^2 \, \mathrm{d}t} \tag{1}$$

among all degree n polynomials $p_n \in \mathbb{P}_n$. It can explictly be expressed as

$$p_n^* = \sum_{k=0}^{n} (\tilde{q}_k, f)_2 \tilde{q}_k, \tag{2}$$

where $\tilde{q}_k$ are the rescaled Legendre polynomials $\tilde{q}_k = q_k \sqrt{(2k+1)/2}, k = 0, 1, \ldots, n$, and $(u, v)_2 = \int_{-1}^{1} u(t)v(t) \, \mathrm{d}t$ denotes the L^2 inner product.

(a) Define a function `rescaled_legendre_polynomial(k)` which returns the k-th rescaled Legendre polynomial. You can use `np.polynomial.legendre.Legendre.basis` which takes as input a natural number k and returns the k-th Legendre polynomial q_k.

(b) Define a function `l_2_inner_product` which takes as input two functions u and v and approximates their L^2 inner product $(u, v)_2$ with a sufficiently accurate quadrature rule of your choice.

(c) Using the two previously defined functions and the expression (2), write a function `l_2_optimal_approximation` which takes as input a function f and a natural number n and returns the L^2 best approximation p_n^*.

(d) For the function $f(x) = 1/(1 + 25x^2)$ and the degrees $n = 1, 2, \ldots, 20$, compare the L^2-error (1) of the best approximation in the L^2 norm with the one for the Chebyshev interpolant of the same degree. Use an appropriate quadrature rule of your choice to approximate the integral.

Hint: Use `np.polynomial.chebyshev.Chebyshev.interpolate` to compute the Chebyshev interpolant.

Solution. Available in the Jupyter notebook `serie07-sol.ipynb` on Moodle.

Problem 2.

(a) For each of the following matrices, determine if an LU factorisation exists. If an LU factorisation exists, compute it.

(i) $\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$ (ii) $\begin{pmatrix} 4 & 2 \\ 3 & 4 \end{pmatrix}$ (iii) $\begin{pmatrix} 3 & 3 \\ 1 & 1.5 \end{pmatrix}$ (iv) $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$

(b) Compute the LU factorisation with pivoting for the matrix

$$A = \begin{pmatrix} 5 & 1 & 2 \\ 2 & \frac{2}{5} & 1 \\ -4 & 0 & 6 \end{pmatrix}$$

Compare it with the Python function `sp.linalg.lu`. (If you are not working in the notebooks we provided you will need to use `scipy.linalg.lu`.)

(c) Suppose that $PA = LU$ is the LU factorisation with pivoting of A. Find a simple formula for $\det(A)$ and $|\det(A)|$ in terms of L, U, and the permutation determining P.

(d) In your linear algebra course you have seen the following definition of the determinant of a matrix $A \in \mathbb{R}^{n\times n}$:

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)} \tag{3}$$

where S_n is the set of all permutations of length n.

Compare the computational complexity of computing the determinant with Equation (3) and computing the determinant via the LU factorisation.

Solution.

(a) (i) The LU factorization exists and equals $\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 0 & -3 \end{pmatrix}$.

(ii) The LU factorization exists and equals $\begin{pmatrix} 4 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{3}{4} & 1 \end{pmatrix} \begin{pmatrix} 4 & 2 \\ 0 & \frac{5}{2} \end{pmatrix}$

(iii) The LU factorization exists and equals $\begin{pmatrix} 3 & 3 \\ 1 & 1.5 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{1}{3} & 1 \end{pmatrix} \begin{pmatrix} 3 & 3 \\ 0 & \frac{1}{2} \end{pmatrix}$

(iv) No LU factorization exists. One can see that if such factorization would exist then

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \ell & 1 \end{pmatrix} \begin{pmatrix} u_1 & u_2 \\ 0 & u_3 \end{pmatrix}$$

which implies $u_1 = 0, u_2 = 1$. This is a contradiction because it would mean

$$1 = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}_{2,1} = \ell \cdot 0 + 1 \cdot 0 = 0$$

which is clearly a contradiction. The reason why the LU factorization does not exist is because the first leading principal supmatrix is not invertible.

(b) Following Algorithm 4.13 we get

$$\tilde{A}^{(0)} = A, P_1 = I_3$$
$$A^{(1)} = \begin{pmatrix} 5 & 1 & 2 \\ -\frac{2}{5} & 0 & \frac{1}{5} \\ -\frac{4}{5} & \frac{4}{5} & \frac{38}{5} \end{pmatrix}$$
$$\tilde{A}^{(1)} = \begin{pmatrix} 5 & 1 & 2 \\ \frac{-4}{5} & \frac{4}{5} & \frac{38}{5} \\ \frac{2}{5} & 0 & \frac{1}{5} \end{pmatrix}, P_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$
$$A^{(2)} = \tilde{A}^{(2)}$$

Hence,

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{4}{5} & 1 & 0 \\ \frac{2}{5} & 0 & 1 \end{pmatrix}, U = \begin{pmatrix} 5 & 1 & 2 \\ 0 & \frac{4}{5} & \frac{38}{5} \\ 0 & 0 & \frac{1}{5} \end{pmatrix}, P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

and if we use `P, L, U = sp.linalg.lu(A)` we can see that it returns the same matrices.

(c) We have $\det(PA) = \det(P)\det(A) = \det(LU) = \det(L)\det(U) \Rightarrow \det(A) = \frac{\det(L)\det(U)}{\det(P)}$. Since P is a permutation matrix, $\det(P)$ equals the sign of the associated permutation, call it σ. Since L and U are triangular their determinants are equal to the product of their diagonal elements. Since, $\operatorname{sgn}(\sigma) = \pm 1$ we have

$$\det(A) = \frac{\prod_{i=1}^{n} \ell_{ii} u_{ii}}{\operatorname{sgn}(\sigma)} = \operatorname{sgn}(\sigma) \prod_{i=1}^{n} u_{ii}$$

and

$$|\det(A)| = \prod_{i=1}^{n} |u_{ii}|$$

(d) Computing the $\det(A)$ via the LU factorization requires $O(n^3)$ operations to get the factorization and $O(n)$ operations to carry out the multiplications. Hence, the total work done is $O(n^3)$.

Computing the determinant from its definition requires $O(n \cdot n!)$ operations, since we need to perform n multiplications $n!$ times and sum the resulting values.

Problem 3.

Let the linear system $Ax = b$ be given by defining

$$A = \begin{pmatrix} 10^{-4} & 1 \\ 1 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

Solve the system of linear equations $Ax = b$ via the LU factorisation

(a) in exact arithmetic,

(b) without pivoting in floating point arithmetic $\mathbb{F}(10, 3, -10, 10)$ (i.e. with 3 significant digits), and

(c) with partial pivoting in floating point arithmetic $\mathbb{F}(10, 3, -10, 10)$.

and compare the results.

Solution.

(a) Exact arithmetic:

$$\begin{array}{c} \\ \frac{1}{1\cdot 10^{-4}} \end{array} \left(\begin{array}{cc|c} 1\cdot 10^{-4} & 1 & 1 \\ 1 & 1 & 2 \end{array}\right)$$

$$\left(\begin{array}{cc|c} 1\cdot 10^{-4} & 1 & 1 \\ 0 & -9999 & -9998 \end{array}\right)$$

$$\Rightarrow x_2 = \frac{9998}{9999} = 1 - \frac{1}{9999}$$

$$x_1 = \frac{1}{1\cdot 10^{-4}}(1 - (1 - \frac{1}{9999})) = \frac{10000}{9999} = 1 + \frac{1}{9999}.$$

(b) Solution in floating point arithmetic without pivoting:

$$\begin{array}{c} \\ \frac{1}{1\cdot 10^{-4}} \end{array} \left(\begin{array}{cc|c} 1\cdot 10^{-4} & 1 & 1 \\ 1 & 1 & 2 \end{array}\right)$$

Performing the operations in floating point arithmetic gives us

$$a_{22} = 1 - 1.00 \cdot 10^4 \cdot 1 = -9.999 \cdot 10^3 \approx -1.00 \cdot 10^4$$
$$b_2 = 2 - 1.00 \cdot 10^4 \cdot 1 = -9.998 \cdot 10^3 \approx -1.00 \cdot 10^4$$

This gives us the system in the next step

$$\left(\begin{array}{cc|c} 1\cdot 10^{-4} & 1 & 1 \\ 0 & -1\cdot 10^4 & -1\cdot 10^4 \end{array}\right)$$

$$\Rightarrow x_2 = 1$$

$$x_1 = 1 \cdot 10^4(1 - 1) = 0,$$

whereby the result for x_1 deviates significantly from the exact result.

(c) Solution in floating point arithmetic, with pivoting. We swap lines 1 and 2 because $|a_{21}| > |a_{11}|$:

$$\begin{array}{c} \\ 1\cdot 10^{-4} \end{array} \left(\begin{array}{cc|c} 1 & 1 & 2 \\ 1\cdot 10^{-4} & 1 & 1 \end{array}\right)$$

Performing the operations in floating point arithmetic gives us

$$a_{22} = 1 - 1.00 \cdot 10^{-4} \cdot 1 = 9.999 \cdot 10^{-1} \approx 1.00 \cdot 10^0$$
$$b_2 = 1 - 1.00 \cdot 10^{-4} \cdot 2 = 9.998 \cdot 10^{-1} \approx 1.00 \cdot 10^0.$$

This gives us the system in the next step

$$\left(\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & 1 & 1 \end{array}\right)$$
$$\Rightarrow x_2 = 1$$
$$x_1 = 2 - 1 = 1,$$

which corresponds to the exact result rounded to 3 significant digits. Different algorithms can therefore lead to different results numerically, even if the results would have to be the same in theory.