Exercise set 6 – MATH-250 Advanced Numerical Analysis I

The exercise sheet is divided into two sections: quiz and exercises. The quiz will be discussed in the beginning of the lecture on Thursday, April 3. The exercises marked with (⋆) 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 4 at 10h15.**

Quiz

Let T_n, $n = 0, 1, 2, \dots$ denote the Chebyshev polynomials.

(a) For $m \geq n$ it holds $T_n(x)T_m(x) = T_{m+n}(x) + T_{m-n}(x)$.

☐ True ■ False

(b) The $(n+m)$-th derivative of $T_n(x)T_m(x)$ at $x = 0$ for $m, n \geq 1$ is

☐ 0 ■ $2^{n+m-2}(n+m)!$

☐ $2^{n+m}(n+m)!$ ☐ $(-1)^{n+m}(n+m)!$

(c) The Chebyshev interpolant of a nonnegative function is nonnegative.

☐ True ■ False

Solution.

(a) We know that $T_n(x) = \cos(n \arccos(x)) = \cos(ny)$ if we define $y = \arccos(x)$. We now apply the trigonometric addition theorem for the cosine and see

$$\cos(my)\cos(ny) = \frac{1}{2}(\cos((m+n)y) + \cos((m-n)y)), \tag{1}$$

meaning that the claim is false.

(b) We begin with (1) and write it as Chebyshev polynomials

$$T_m(x)T_n(x) = \frac{1}{2}(T_{m+n}(x) + T_{m-n}(x)).$$

The right-hand side thus consists of two polynomials, T_{m+n} of degree $m+n$, and T_{m-n} of degree strictly less than $m+n$. Therefore, the $(m+n)$-th derivative of T_{m-n} vanishes and we only need to compute the $(m+n)$-th derivative of T_{m+n}. We denote a_{m+n} the leading coefficient of T_{m+n} — if we can compute a_{m+n}, then the $m+n$-th derivative of T_{m+n} will be $a_{m+n} \cdot (m+n)!$ because no other monomials will be left.

We now show that $a_k = 2a_{k-1}$ for $k > 1$. On the one hand, we can express T_k as

$$T_k(x) = a_k x^k + p_{k-1}(x), \tag{2}$$

where p_{k-1} is a polynomial of degree $k-1$, and on the other hand we know that T_k is

$$T_k(x) = 2x \cdot T_{k-1}(x) - T_{k-2}(x). \tag{3}$$

In (3) it is evident that T_{k-2} does not influence the coefficient a_k from (2) because its degree is $k-2$. We thus see that $a_k = 2a_{k-1}$, where the recurrence terminates with $a_1 = 1$, and hence $a_k = 2^{k-1}$. Lastly, we need to account for the factor 1/2 from (1) to obtain $2^{m+n-2}(m+n)!$ and conclude the proof.

(c) We consider the function $f(x) = \exp(-3x)$ on the interval $(-1, 1)$ and use 5 interpolation points.

```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt

xx = np.linspace(-1, 1)
cheb = np.polynomial.chebyshev.chebpts1(5)
f = lambda x: np.exp(-3 * x)
fhat = sp.interpolate.lagrange(cheb, f(cheb))

plt.plot(xx, f(xx), xx, fhat(xx))
plt.show()
```

Exercises

Consider $n+1$ points $x_0, x_1, \ldots, x_n$. Suppose the interpolant of some data $y_0, y_1, \ldots, y_n$ at these points is $p_n(x) = \sum\limits_{i=0}^{n} a_i x^i$. One method to determine the coefficients $a_0, a_1, \ldots, a_n$ is to solve the linear system

$$V_n \mathbf{a}_n = \mathbf{y} \tag{4}$$

where V_n is the Vandermonde matrix defined by

$$V_n = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \tag{5}$$

and

$$\mathbf{a}_n = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

To facilitate the other exercises, write a function `interpolate_data` which takes as input an array $\mathbf{y}$ of $n+1$ values $y_0, y_1, \ldots, y_n$ and an array $\mathbf{x}$ of $n+1$ points $x_1, x_2, \ldots, x_n$; creates

V_n with the Python function `numpy.vander/np.vander`; and solves the linear system (4) with `numpy.linalg.solve/np.linalg.solve`; and finally returns the coefficients $\mathbf{a}_n$ of the corresponding interpolating polynomial p_n.

Write a second function, `interpolate_function`, which takes as input a function f and does the same as `interpolate_data` on $y_0 = f(x_0), y_1 = f(x_1), \dots, y_n = f(x_n)$.

Solution. Available in the Jupyter notebook `serie06-sol.ipynb` on Moodle.

Problem 1. (Python) Consider $n+1$ points $x_0, x_1, \dots, x_n \in [-1, 1]$, the functions

$$f^{(1)}(x) = \frac{1}{1+9x^2}, \quad f^{(2)}(x) = \sin(x),$$

and the interpolating polynomials $p_n^{(1)}$ and $p_n^{(2)}$ which interpolate $f^{(1)}$ and $f^{(2)}$, respectively, at the points $x_0, x_1, \dots, x_n$. Suppose $p_n^{(j)}(x) = \sum_{i=0}^{n} a_i^{(j)} x^i$, $j = 1, 2$.

(a) Use the Python function `numpy.vander/np.vander` to get the Vandermonde matrix V_n. For $n = 2, 3, \dots, 40$, plot the condition number of the Vandermonde matrix $\kappa(V_n)$ uniformly distributed interpolation nodes and Chebyshev nodes on $[-1, 1]$. As will be seen later in the course, the condition number measures the sensitivity of a linear system to roundoff error. Large condition numbers usually mean that the accuracy of the computed solution is low.

Hint: The condition number can be computed with `numpy.linalg.cond/np.linalg.cond`.

(b) For $n = 10, 20, 30, 40$ compute the coefficients $\mathbf{a}_n^{(j)}$ of the interpolants of $f^{(j)}$, $j = 1, 2$ for uniformly distributed interpolation nodes and Chebyshev nodes. Use these coefficients to plot the evaluation of $p_n^{(j)}(x)$ at 500 evenly spaced values x. Compare them to $f^{(j)}$ for $j = 1, 2$. Explain what you observe.

Hint: You can evaluate a polynomial from its coefficients with `numpy.polyval/np.polyval`.

(c) Approximate the error

$$\max_{x \in [-1,1]} |f^{(j)}(x) - p_n^{(j)}(x)|$$

by replacing the maximum in $[-1, 1]$ with the maximum at 500 evenly spaced points in $[-1, 1]$. and plot it against $n = 2, 3, \dots, 40$ for $j = 1, 2$.

Solution. Available in the Jupyter notebook `serie06-sol.ipynb` on Moodle.

Problem 2. (Python) In this exercise we will study the stability of the Lagrange interpolation polynomial on $n+1$ uniformly distributed nodes and on Gauss-Legendre nodes. Gauss-Legendre nodes are defined to be the zeros of the Legendre polynomials q_n, which can be otained with the Python function `scipy.special.roots_legendre/sp.special.roots_legendre`. Consider the function

$$f(x) = \sin(x) + x, \quad x \in [0, 10]$$

which we will interpolate on the nodes $x_0, x_1, \dots, x_n$. Further define $y_i = f(x_i)$ for $i = 0, 1, \dots, n$.

(a) For $n = 1, 2, \ldots, 15$, numerically compute the Lebesgue constant Λ_n for uniformly distributed nodes and plot the result. Based on the results obtained, formulate a conjecture of the asymptotic behavior of the Lebesgue constant, e.g., $O(\log^c n), O(n^c), O(c^n)$ for some constant c.

(b) Plot the function f and the interpolation polynomials for $n = 4$ and $n = 15$ for uniformly distributed nodes.

(c) For $i = 0, 1, \ldots, n$ let ε_i be independent uniformly distributed random variables in $[-0.1, 0.1]$. For each i perturb $\tilde{y}_i = y_i + \varepsilon_i$. Repeat (b) with the new data $\tilde{y}_0, \tilde{y}_1, \ldots, \tilde{y}_n$. The function `numpy.random.uniform/np.random.uniform` in Python will be useful.

(d) Repeat (a)-(c) with Gauss-Legendre nodes.

Solution. Available in the Jupyter notebook `serie06-sol.ipynb` on Moodle.

Problem 3. Consider the interpolation of the function $f(x) = x^{-3}$ on $[3, 4]$ using 4 Chebyshev nodes. Denote the interpolation polynomial $p_3(x)$.

(a) Write down the numerical values of the 4 nodes at which p_3 interpolates f.

(b) Find an upper bound for the error $|f(x) - p_3(x)|$ which is valid for any x in the interval $[3, 4]$.

(c) How many digits of accuracy will you have when p_3 is used to approximate $f(x)$?

(d) Calculate $p_3(x)$ numerically in Python and plot the graph of the error and the upper bound of the error as a function of x on a semi-logarithmic scale. Compare the interpolating polynomial obtained using the Chebyshev nodes with the one using the equispaced nodes over the interval $[3, 4]$.

Solution.

(a) Using $x_k = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{(2k+1)\pi}{2n+2}\right) = \frac{7}{2} + \frac{1}{2} \cos\left(\frac{(2k+1)\pi}{2n+2}\right), k = 0, 1, \cdots, n$ we get

$$\begin{aligned} x_0 &= 3.96 \\ x_1 &= 3.69 \\ x_2 &= 3.31 \\ x_3 &= 3.04 \end{aligned}$$

(b) We know by Theorem 3.6

$$\|f - p_3\|_\infty \leq \frac{1}{2^3(3+1)!} \frac{1}{2^4} \|f^{(4)}\|_\infty = \frac{1}{24 \cdot 2^7} \|f^{(4)}\|_\infty.$$

By direct differentiation $f^{(4)}(x) = 360x^{-7}$ which takes it maximum at $x = 3$. Hence,

$$\|f - p_3\|_\infty \leq \frac{360}{24 \cdot 2^7 \cdot 3^7} = 5.35 \times 10^{-5}$$

(c) You will have approximately $-\log_{10}(5.35 \times 10^{-5}) \approx 4$ digits of precision.

(d) Available in the Jupyter notebook `serie06-sol.ipynb` on Moodle.

Problem 4. In this exercise T_n denotes the n^{th} Chebyshev polynomial in $[-1, 1]$.

(a) Show that T_n is even if n is even and T_n is odd if n is odd.

(b) T_n is only defined in $[-1, 1]$, but using the three-term recurrence relation one can extend its definition outside $[-1, 1]$. Show that for $|x| \geq 1$ we have

$$T_n(x) = \begin{cases} \cosh(n \operatorname{arccosh}(x)), & x \geq 1; \\ (-1)^n \cosh(n \operatorname{arccosh}(-x)), & x \leq -1. \end{cases}$$

Solution.

(a) This will follow by induction. Clearly $T_0(x) = 1$ is even and $T_1(x) = x$ is odd. Now suppose the results holds for all $k \leq n$ where n is odd.

By the three term recurrence relation we have

$$T_{n+1}(-x) = -2xT_n(-x) - T_{n-1}(-x) = 2xT_n(x) - T_{n-1}(x) = T_{n+1}(x).$$

Hence, T_{n+1} is even. Similarly we have

$$T_{n+2}(-x) = -2xT_{n+1}(-x) - T_n(-x) = -(2xT_{n+1}(x) - T_n(x)) = -T_{n+1}(x).$$

Hence, T_{n+2} is odd. Thus, by induction the result is proven.

(b) Consider the function

$$t_n(x) = \frac{1}{2}((x - \sqrt{x^2 - 1})^n + (x + \sqrt{x^2 - 1})^n).$$

It is easy to see that $t_0(x) = 1$ and $t_1(x) = x$. Now we can note that $t_n(x)$ satisfies the three term recurrence relation

$$\begin{aligned}
&2xt_{n-1}(x) - t_{n-2}(x) \\
&= x((x - \sqrt{x^2 - 1})^{n-1} + (x - \sqrt{x^2 - 1})^{n-1}) - (x - \sqrt{x^2 - 1})^{n-2} - (x + \sqrt{x^2 - 1})^{n-2} \\
&= \frac{1}{2}(2x^2 - 2x\sqrt{x^2 - 1} - 1)(x - \sqrt{x^2 - 1})^{n-2} + \frac{1}{2}(2x^2 + 2x\sqrt{x^2 - 1} - 1)(x + \sqrt{x^2 - 1})^{n-2} \\
&= \frac{1}{2}(x - \sqrt{x^2 - 1})^2(x - \sqrt{x^2 - 1})^{n-2} + \frac{1}{2}(x + \sqrt{x^2 - 1})^2(x + \sqrt{x^2 - 1})^{n-2} \\
&= t_n(x).
\end{aligned}$$

Hence, $t_n(x) = T_n(x) \quad \forall n \in \mathbb{N}_0$. Now note

$$\begin{aligned}
1^n &= 1 \\
(x^2 - x^2 + 1)^n &= 1 \\
((x + \sqrt{x^2 - 1})(x - \sqrt{x^2 - 1}))^n &= 1 \\
(x + \sqrt{x^2 - 1})^{-n} &= (x - \sqrt{x^2 - 1})^n.
\end{aligned}$$

Hence,

$$T_n(x) = \frac{1}{2}(x + \sqrt{x^2 - 1})^n + \frac{1}{2}(x + \sqrt{x^2 - 1})^{-n}$$

and note $\operatorname{arcosh}(x) = \ln(x + \sqrt{x + \sqrt{x^2 - 1}})$ for $x \geq 1$. Thus,

$$T_n(x) = \frac{1}{2}\exp(n \operatorname{arcosh}(x)) + \frac{1}{2}\exp(-n \operatorname{arcosh}(x)), \quad x \geq 1$$

and by $\cosh(x) = \frac{1}{2}\exp(x) + \frac{1}{2}\exp(-x)$ we get

$$T_n(x) = \cosh(n \operatorname{arcosh}(x)), \quad x \geq 1.$$

Now, when $x \leq 1$ we use the fact that T_n is odd/even whenever n is odd/even to get

$$T_n(x) = (-1)^n \cosh(n \operatorname{arcosh}(-x)), \quad x \leq 1.$$