1. *[3-term recursion and orthogonal polynomials, 2+2pt]* We consider the polynomial recursion \( l_0(x) = 1, \ l_1(x) = 1 - x, \) and
\[
 l_{k+1}(x) = \frac{2k+1-x}{k+1}l_k(x) - \frac{k}{k+1}l_{k-1}(x) \quad \text{for } k = 1, 2, \ldots
\]
(a) Derive the polynomials for \( l_2(x) \) and \( l_3(x) \) from the recursion.\(^1\) Verify\(^2\) that
\[
 \int_0^\infty \exp(-t)l_i(t)l_j(t) \, dt = 0 \quad \text{for } 0 \leq i < j \leq 3.
\]
Since this orthogonality relation holds for all \( i \neq j \) (you do not need to show that), the polynomials \( l_i() \) are orthogonal on \([0, \infty)\) with weight function \( \omega(t) = \exp(-t) \).
(b) Write a function \( \text{my}_1(k, x) \), which returns \( l_k(x) \). The function should also allow vector inputs \( x = (x_1, \ldots, x_n) \) and return \( (l_k(x_1), \ldots, l_k(x_n)) \). Your function should not derive the polynomials analytically, but just compute their value at the points \( x \) using the recursion. Hand in a code listing, and plot graphs of the first several polynomials \( l_i \) (these polynomials are called Laguerre polynomials).

2. *[Interpolation with trigonometric functions, 4pt]* For \( N \geq 1 \), we define the set of complex trigonometric polynomials of degree \( \leq N - 1 \) as
\[
 T_{N-1} := \left\{ \sum_{j=0}^{N-1} c_je^{ijt}, c_j \in \mathbb{C} \right\},
\]
where \( i \) denotes the imaginary unit.\(^3\) The corresponding (complex) interpolation problem is: Given pairwise distinct nodes \( t_0, \ldots, t_{N-1} \in [0, 2\pi) \) and corresponding nodal values \( f_0, \ldots, f_{N-1} \in \mathbb{C} \), find a trigonometric polynomial \( p \in T_{N-1} \) such that \( p(t_i) = f_i \), for \( i = 0, \ldots, N-1 \).
(a) Show that there exists exactly one \( p \in T_{N-1} \), which solves this interpolation problem.

---

\(^1\)Note that in this recursion the polynomials are not normalized as in the problems we discussed in class, where the leading coefficient was assumed to be 1. Such a (or another) normalization can easily be achieved by multiplication of \( L_k(x) \) with an appropriate scaling factor.

\(^2\)Feel free to look up the values for the indefinite integrals \( \int_0^\infty \exp(-t)t^k \, dx \) \((k = 0, 1, 2, 3)\)—I use Wolfram Alpha for looking up things like that: [http://www.wolframalpha.com/](http://www.wolframalpha.com/).

\(^3\)Note that \( e^{jt} = \cos(jt) + i\sin(jt) \).
(b) Define $W_N := e^{-\frac{2\pi i}{N}}$. Show that
\[
\frac{1}{N} \sum_{j=0}^{N-1} (W_N^{l-k})^j = \delta_{kl} := \begin{cases} 1 & \text{if } k = l, \\ 0 & \text{if } k \neq l, \end{cases} \quad \text{for } 0 \leq l, k \leq N - 1.
\]

(c) Let us now choose the equidistant nodes $t_k := \frac{2\pi k}{N}$ for $k = 0, \ldots, N - 1$. Show that the trigonometric polynomial that satisfies $p(t_i) = f_i$ for $i = 0, \ldots, N - 1$ has the coefficients
\[
c_j = \frac{1}{N} \sum_{k=0}^{N-1} W_N^{jk} f_k.
\]

(d) For equidistant nodes, the linear map from $\mathbb{C}^N \to \mathbb{C}^N$ defined by $(f_0, \ldots, f_{N-1}) \mapsto (c_0, \ldots, c_{N-1})$ is called the discrete Fourier transformation. How many (complex) floating point operations are required to compute this map?

3. **[Chebyshev polynomials, 1+1+1+1+1+2+2+pt]** The recurrence relation for Chebyshev polynomials $T_k$ is $T_0(x) = 1$, $T_1(x) = x$ and $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$. Show from this relation that:

(a) For even $k$, $T_k$ is symmetric, i.e., $T_k(-x) = T_k(x)$. For odd $k$, $T_k$ satisfies $T_k(-x) = -T_k(x)$.

(b) By showing that $T_k'(x) = \cos(k \arccos(x))$ satisfies this 3-term recurrence relation, argue that $T(x) = T'(x)$. Note that this implies that $|T_k(x)| \leq 1$ for all $x$.

(c) By showing that
\[
T''_k(x) = 1/2 \left( (x + \sqrt{x^2 - 1})^k + (x - \sqrt{x^2 - 1})^k \right)
\]
satisfies the same 3-term recurrence relation, argue that $T''(x) = T(x)$. Note that differently from $T'(x)$, the form $T''(x)$ is defined for all $x \in \mathbb{R}$.

(d) Show that the zeros of $T_k(x)$ are given by
\[
x_j = \cos \left( \frac{(2j + 1)\pi}{2k} \right) \quad \text{for } j = 0, \ldots, k - 1.
\]

These $x_j$ are called Chebyshev nodes (corresponding to the $k$-th Chebyshev polynomial).

(e) The Newton polynomial $\omega_{n+1}$ that is based on Chebyshev nodes (i.e., the roots of $T_{n+1}$) satisfies
\[
|\omega_{n+1}(x)| \leq \frac{1}{2^n} \quad \text{for all } x \in [-1, 1].
\]

*Hint: Think about what the difference is between the Newton polynomial with node points given by the roots of the Chebyshev polynomial $T_{n+1}$, and the Chebyshev polynomial $T_{n+1}$ itself.* Note that it can be shown that the Chebyshev nodes minimize the pointwise bound given by (1), which is referred to as the min-max property of the Chebyshev nodes and explains their usefulness for interpolation.
(f) Show that this implies the following error estimate for the interpolation error $E_f(x) := f(x) - p_n(x)$, where $p_n \in \mathcal{P}_n$ is a polynomial that interpolates a function $f \in C^{n+1}$ at the Chebyshev nodes $x_0, \ldots, x_n$:

$$|E_f(x)| \leq \frac{1}{2^n(n+1)!} \|f^{(n+1)}\|_{\infty},$$

(2)

where for continuous functions $g$, $\|g\|_{\infty} = \|g\|_{C^0([-1,1])} := \max_{-1 \leq t \leq 1} |g(t)|$ is the supremum norm on the interval $[-1, 1]$.

(g) Compute and plot the approximation of the function $f(x) = 1/(1 + 12x^2)$ with a polynomial of order $N = 14$ on the interval $[-1, 1]$. Choose $N + 1$ equidistant nodes (including the points $\pm 1$) and the Chebyshev nodes given by the roots of $T_{N+1}(x)$ as interpolation points, and compare the results.\(^4\) What do you observe?

(h) Repeat the Chebyshev point-based interpolation with at least two larger numbers $N$ of Chebyshev interpolation points, and approximate the maximal interpolation error.\(^5\) Plot the maximal error as a function of $N$ and compare with the estimate (2) (without evaluating (2) explicitly, just in terms of how it changes as you change $N$). Use a logarithmic scale for plotting the error (in MATLAB, you can use semilogy). Can you spot a trend for the error?

4. [Polynomial interpolation and error estimation, 1+1+1+2+2pt] Let us interpolate the function $f : [0, 1] \to \mathbb{R}$ defined by $f(t) = \exp(3t)$ using the nodes $t_i = i/2, i = 0, 1, 2$ by a quadratic polynomial $p_2 \in \mathcal{P}_2$.

(a) Use the monomial basis $1, t, t^2$ and compute (numerically) the coefficients $c_j \in \mathbb{R}$ such that $p_2(t) = \sum_{j=0}^2 c_j t^j$.

(b) Give an alternative form for $p_2$ using Lagrange interpolation polynomials.

(c) Give yet another alternative form of $p_2$ using the Newton polynomial basis $\omega_0(t) = 1$, and $\omega_j(t) = \prod_{i=0}^{j-1} (t - t_i)$ for $j = 1, 2$. Compute the coefficients of $p_2$ in this basis directly, or follow the lecture slides to compute the coefficients using divided differences.

(d) Compare the exact interpolation error $E_f(t) := f(t) - p_2(t)$ at $t = 3/4$ with the estimate

$$|E_f(t)| \leq \frac{\|\omega_{n+1}\|_{\infty}}{(n+1)!} \|f^{(n+1)}\|_{\infty},$$

where $f^{(n+1)}$ is the $(n+1)$st derivative of $f$, and $\|\cdot\|$ is the supremum norm for the interval $[0, 1]$.

(e) Find a (Hermite) polynomial $p_3 \in \mathcal{P}_3$ that interpolates $f$ in $t_0, t_1, t_2$ and additionally satisfies $p_3'(t_3) = f'(t_3)$, where $t_2 = t_3 = 1$. Give the polynomial $p_3$ using the Newton basis.\(^6\)

---

\(^4\)You can use MATLAB’s polynomial interpolation functions polyfit.

\(^5\)You can do that by choosing a very fine mesh with, say, 10,000 uniformly distributed points $s_i$ in $[-1,1]$, and compute the maximum of $|E_f(s_i)|$ for all $i$.

\(^6\)You should only have to add a row to the derivation you did when you computed $p_2$ using the Newton basis. Note that since $t_2 = t_3$, you can use that the divided difference $[t_2 t_3]f = f'(t_2)$. Moreover, note that $[t_3]f = f(t_3) = f(t_2)$. 

3
5. [Property of polynomial interpolation, 2pt] We consider Lagrange interpolation with distinct interpolation nodes \(t_0, \ldots, t_n\).

(a) Show that the \(n\)th divided difference \([t_0, \ldots, t_n]f\), i.e., the leading coefficient of the polynomial interpolant, satisfies:

\[
[t_0, \ldots, t_n]f = \sum_{k=0}^{n} \frac{f(t_k)}{\prod_{i \neq k}(t_i - t_k)}.
\]  

\(3\)

Hint: Differentiate the expressions of the interpolant in the monomial and the Newton basis \(n\)-times and compare.

(b) Show that, for any continuous function \(f(t)\) holds

\[(P_n(tf) - tP_n(f))(t) = (-1)^{n+1}[t_0, \ldots, t_n]f \omega_{n+1}(t),\]

where \(P_n(g) = P_n(g|t_0, \ldots, t_n)\) is the polynomial interpolant for a continuous function \(g\), and \(\omega_{n+1}(t) = \prod_{i=0}^{n}(t_i - t)\) is the \((n + 1)st\) Newton polynomial. Hint: Multiply for \(t \neq t_j, 1 \leq j \leq n\), the numerator and the denominator in (3) by \(L_k(t)(t_k - t)\), where \(L_i\) is the Lagrange polynomial for the node \(t_i\). Argue separately for \(t = t_j\).