Problem 1: Newton interpolation (40 points)

(a) Write a Matlab program which, given \( n+1 \) data points \((x_j, f_j)\), computes the coefficients of the unique degree-\( n \) polynomial passing through those points, using Newton interpolation. The top-level function should take the form

\[
c = \text{NewtonInterpolate}(x, f)
\]

Here \( x \) and \( f \) will be vectors of the same length \( n+1 \), containing \( x_0, \ldots, x_n \) and \( f_0, \ldots, f_n \) respectively. Your function will return \( c \), a length-\((n+1)\) vector containing the polynomial’s coefficients.

(b) We need to evaluate the polynomial from (a). An efficient way to do this is via Horner’s scheme, which organizes the polynomial calculation so as to minimize the number of arithmetic operations required. If \( p(x) = c_n x^n + \cdots + c_1 x + c_0 \) (the case of the basis of monomials), Horner’s scheme computes \( p(x) \) as

\[
p(x) = (((((c_n x) + c_{n-1}) x + c_{n-2}) x + \cdots c_1) x) + c_0,
\]

requiring just \( n \) multiplications and \( n \) additions for each \( x \).

Write a function to evaluate polynomials using Horner’s scheme for the basis of Newton’s polynomials, given the coefficient \( c \) as output by your program from part (a) and a vector of input points \( x \). It should return a vector containing the values of the polynomial at each point in \( x \). Your code should be vectorized, meaning that it works with all the points of \( x \) at once instead of looping over them individually.

(c) Write a function which updates interpolating polynomials. It should take the coefficients of an existing degree-\( n \) interpolating polynomial, the points \( x_0, \ldots, x_n \), and a new point \((x_{n+1}, f_{n+1})\), and return the vector of coefficients of the degree-\((n+1)\) polynomial passing through all the existing points as well as \((x_{n+1}, f_{n+1})\). If \( x_{n+1} \) is equal to one of \( x_0, \ldots, x_n \), an error should be thrown.

(d) Test out your program from (a) in the following ways: For \( n = 5, 10, 20 \), let \( x_0, \ldots, x_n \) be equally spaced points in the interval \([-2, 4]\). For each \( n \), compute the polynomial interpolant for the functions \( f(x) = e^{-x^2} \) and \( f(x) = \sin(x) \). For each function, make a plot of the original function and the polynomial interpolant (using part (b) to evaluate the polynomial) for each \( n \) value (include legends!).
(e) Next, apply your program from (a) to Runge’s function, \( f(x) = 1/(1 + x^2) \), at 15 equally spaced points in the interval \([-3, 3]\). Plot \( f(x) \) and the polynomial interpolant \( p(x) \). You should see that \( p(x) \) matches \( f(x) \) in some areas within \([-3, 3]\) better than others (the behavior in the bad parts is Runge’s phenomenon). To improve the fit, choose appropriate additional \( x \) values in the range \([-3, 3]\) and add them to the polynomial interpolation using your code from part (c).

Make a plot of \( f \), the original interpolant \( p \), and your improved interpolant (including a legend).

Problem 2: Piecewise linear interpolation (30 points)

In this problem, you will prove an \( L^2 \) error estimate for piecewise linear polynomial interpolation. Let \( f(x) \) be a twice differentiable function on \([a, b]\), split uniformly into intervals \([x_j, x_{j+1}]\) of length \( \Delta = (b-a)/n \) for \( j = 0, \ldots, n-1 \). Let \( S \) be the piecewise linear polynomial interpolating \( f \) on each interval. A key question is how the interval size \( \Delta \) affects the approximation of \( f \). Here you will prove the following \( L^2 \) error estimate:

\[
\sqrt{\int_a^b |f(x) - S(x)|^2 \, dx} \leq \Delta^2 \sqrt{\int_a^b |f''(x)|^2 \, dx}. \tag{*}
\]

The integrals above are measuring the “size” of \( f(x) - S(x) \) and \( f''(x) \) respectively. So, what this estimate says is that (for fixed \( f \)) the \( L^2 \) error is \( O(\Delta^2) \).

(a) Define \( e(x) = f(x) - S(x) \). Since \( e(x_j) = 0, j = 0, \ldots, n \), Rolle’s Theorem assures the existence of \( \xi_j \in (x_j, x_{j+1}) \), such that \( e'((\xi_j) = 0, \) for \( j = 0, \ldots, n-1 \). Using the Fundamental Theorem of Calculus (FTC) prove that

\[ e'(x) = \int_{\xi_j}^{x} f''(s) \, ds \quad \text{for} \quad x \in (x_j, x_{j+1}) \]

and that \( |e'(x)| \leq \int_{x_j}^{x_{j+1}} |f''(s)| \, ds \) (also for \( x \in (x_j, x_{j+1}) \)).

(b) Using the Cauchy-Schwarz inequality

\[
\left| \int_{\alpha}^{\beta} u(s)v(s) \, ds \right| \leq \sqrt{\int_{\alpha}^{\beta} u(s)^2 \, ds} \sqrt{\int_{\alpha}^{\beta} v(s)^2 \, ds}
\]

show that

\[ |e'(x)| \leq \Delta^\frac{1}{2} \left( \int_{x_j}^{x_{j+1}} f''(s)^2 \, ds \right)^{\frac{1}{2}} \quad \text{for} \quad x \in (x_j, x_{j+1}). \]

Why can you apply this inequality safely?

(c) Again applying the FTC, arrive at

\[ |e(x)| \leq \Delta^\frac{3}{2} \left( \int_{x_j}^{x_{j+1}} f''(s)^2 \, ds \right)^{\frac{1}{2}} \quad \text{for} \quad x \in [x_j, x_{j+1}]. \]
(d) Square, integrate, and sum from $0$ to $n-1$ to finally obtain $(\ast)$.

(e) How does this error compare to the $L^\infty$ one given in the Lecture Notes?

(f) In this question you will try to predict the $\Delta^2$ order in the bounds. For convenience you can measure the error in the $L^\infty$ way and use Matlab built-in functions. Choose $[a, b] = [-5, 5]$, $f$ to be the Runge’s function, and equally spaced points. Make a semi-log plot of the error for $n$ ranging from 10 to 100. Make sure to include a legend. Plot $\Delta^2$ also in the semi-log scale. Present also a table with the values of the error and its square roots for $\Delta = 5, 2.5, 1.25, \ldots, 0.078125$. What do you observe?

(g) OPTIONAL: Calculate a similar type of bound for the difference between the derivative of $f$ and the derivative of $S$. Why is this error well defined? Repeat the calculations in (f) accordingly. Can you give an informal argument to compare the order obtained here with the one in $(\ast)$?

Problem 3: Hermite type interpolation (15 points)

(a) In this first question you will see that the Hermite interpolating polynomial on one node coincides with the Taylor polynomial. For this purpose determine an interpolating polynomial $p \in P_n$ such $p^{(k)}(x_0) = f^{(k)}(x_0)$, $k = 0, \ldots, n$ and then check that $p(x) = \sum_{j=0}^{n} \frac{f^{(j)}(x_0)}{j!} (x - x_0)^j$.

(b) Not all types of derivative interpolation are well defined on a set of distinct points. Check that trying to interpolate in $P_3$ using

$$f(-1) = 1, \quad f'(-1) = 1, \quad f'(1) = 2, \quad f(2) = 1.$$ 

Problem 4: Trigonometric interpolation (15 points)

Consider the discrete transform Fourier matrix of order 3:

$$F = \begin{bmatrix}
1 & 1 & 1 \\
1 & w & w^2 \\
1 & w^2 & w^4
\end{bmatrix}.$$ 

This matrix differs from the one in the Lecture Notes/Class in the order of the columns. Instead of $\ell = -1, 0, 1$ here we follow the order $\ell = 0, -1, 1$ (In $5 \times 5$ case, it would be $\ell = 0, -1, 1, -2, 2$ instead of $\ell = -2, -1, 0, 1, 2$.)

(a) Write all components of $F$ in terms of $1$, $w$, and $w^2$. Mark $1 = w^0$, $w = w^1$, and $w^2$ in the complex unit circle.

(b) Show that $FF = 3I$. Identify $F^{-1}$.
(c) Show that $FC = DF$ where

$$C = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

and $D$ is a diagonal matrix of diagonal elements 1, $w$, and $w^2$. What are the eigenvalues of $C$?

(d) Write the circulant matrix

$$H = \begin{bmatrix} h_0 & h_2 & h_1 \\ h_1 & h_0 & h_2 \\ h_2 & h_1 & h_0 \end{bmatrix}$$

in function of the powers of $C$ given by $I = C^0$, $C$, and $C^2$.

(e) Multiply this expression of $H$ at the left by $F$ and at the right by $F^{-1}$. Conclude that $F$ also diagonalizes $H$. What are the eigenvalues of $H$?