Homework 8

1. Write a routine (in MATLAB or C++) to form the Newton divided differences and evaluate the interpolant and its derivatives. Your function should take as input the interpolation nodes \( \{ x_k \}_{k=1}^n \), the corresponding function values \( \{ f_k \}_{k=1}^n \), and the evaluation points \( \{ z_j \}_{j=1}^m \); and it should return \( \{ p(z_j) \}_{j=1}^m \) and \( \{ p'(z_j) \}_{j=1}^m \). Make sure your function works on low-order polynomials – the interpolation should be exact for polynomials \( z^k \) for \( k = 0 \) to \( n - 1 \). Also make sure that the code works on non-uniformly spaced interpolation points. Your write-up must include a description of how you tested your code in order to get credit.

It may be helpful to recall that the Newton form of the interpolant can be evaluated at points \( z \) by the MATLAB loop

\[
p = 0*z;
\text{for } k = \text{length(a)}:-1:1
\quad p = p.*(z-x(k))+a(k);
\text{end}
\]

where \( a_k = f[x_1, \ldots, x_k] \) are the Newton divided differences and the \( x_k \) are the interpolation nodes. You can compute the derivative by adding a little to this loop based on differentiating the algorithm with respect to \( z \). Your algorithm to compute the derivative should, like the algorithm to evaluate \( p \), take \( O(n) \) time.

2. Interpolate the function

\[
f(t) = \frac{1}{1 + 25t^2};
\]

using \( n = 4, 8, 12, 16 \) uniformly spaced points on \([-1, 1] \), and plot the interpolant. Also try interpolating with meshes of \( N + 1 = 4, 8, 12, 16 \) Chebyshev points \( x_0, \ldots, x_N \) given by

\[
x_i = \cos \left( \frac{2(N - i) + 1}{2N + 2} \pi \right).
\]

Comment on the difference between interpolation with the uniformly spaced nodes and the Chebyshev nodes.

You may use your routine from the previous problem, or the MATLAB polyfit routine if you didn’t get the first problem.
3. Generate a uniform grid of points $x_k$ for $k = 0$ to $N = 100$ on the interval $[0, 2\pi]$, e.g. using the MATLAB command

$$
x = \text{linspace}(0, 2*\text{pi}, N+1)';
\text{x} = \text{x}(2: \text{end});
$$

and let $f(x) = \exp(\cos(x))$, i.e.

$$
f = \exp(\cos(x));
$$

You can compute the FFT of $f$ in MATLAB with the command

$$
F = \text{fft}(f);
$$

According to the normalization used by MATLAB,

$$
F = Zf \quad \text{and} \quad f = \frac{1}{N}Z^*F
$$

where $Z_{jk} = \exp(-2\pi ij k/N)$, for $j$ and $k$ going from 0 to $N - 1$. The matrix $Z$ may be formed explicitly in MATLAB either using

$$
Z = \exp(-2i\pi /N * (0:N-1)'*(0:N-1));
$$

or, using the periodicity of the complex exponential, as

$$
\text{K} = [0:N/2, -N/2+1:-1];
Z = \exp(-2i\pi /N * \text{K}^{' }*\text{K});
$$

We could also use

$$
Z = \text{fft(eye(N))};
$$

(a) Verify that $\|F - Zf\|_2$ and $\|f - Z^*F/N\|_2$ are both on the order of machine roundoff in this case.

(b) Make a semilogarithmic plot of the magnitude of the components of $F$ versus the index. Comment on the Fourier components in the middle.
(c) Using the definition

\[ K = [0:N/2, -N/2+1:-1]; \]

we have the \textit{spectral differentiation formula}

\[ df = \text{ifft}(1i*K' .* F); \]

Plot the maximum difference between \( df \) and the exact derivative of \( f \) on the mesh point for even values of \( N \) from 2 through 40, using a semilogarithmic scale.