# 1 Practice

**Solution 1.1.** For Lagrange interpolation, we first create polynomials of the form

$$L_{n,k}(x) = \frac{(x-x_0)\dots(x-x_{k-1})(x-x_{k+1})\dots(x-x_n)}{(x_k-x_0)\dots(x_k-x_{k-1})(x_k-x_{k+1})\dots(x_k-x_n)}$$

using the interpolation points. Then using the function values at the interpolation points, we form the interpolating polynomial

$$\sum_{k=0}^{n} f(x_k)L_{n,k}(x).$$

For Newton interpolation, we first create polynomials of the form

$$N_{n,k}(x) = (x-x_0)\dots(x-x_{k-1})$$

using the interpolation points. Then using the function values at the interpolation points, we form the divided differences

$$f[x_0,\dots,x_k] = \frac{f[x_1,\dots,x_k]-f[x_0,\dots,x_{k-1}]}{x_k-x_0}$$

which we use as coefficients for the interpolating polynomial

$$\sum_{k=0}^{n} f[x_0,\dots,x_k]N_{n,k}(x).$$

**Solution 1.2.** First we form the Lagrange interpolation polynomials

$$L_{3,0} = \frac{(x-1.2)(x-3.5)(x-7)}{(0-1.2)(0-3.5)(0-7)} \qquad L_{3,1} = \frac{(x-0)(x-3.5)(x-7)}{(1.2-0)(1.2-3.5)(1.2-7)}$$

$$L_{3,2} = \frac{(x-0)(x-1.2)(x-7)}{(3.5-0)(3.5-1.2)(3.5-7)} \qquad L_{3,3} = \frac{(x-0)(x-1.2)(x-3.5)}{(7-0)(7-1.2)(7-3.5)}.$$

The combining all of these yields

$$3L_{3,0}+6L_{3,1}+1L_{3,2}+5L_{3,3}$$

$$= 0.27246580791237035x^3 - 2.615993023896216x^2 + 5.246840865281645x + 2.9999999999999996$$

Next we form the Newton interpolation polynomials

$$N_0 = 1 \qquad N_1 = (x-0) \qquad N_2 = (x-0)(x-1.2) \qquad N_3 = (x-0)(x-1.2)(x-3.5)$$

and the divided differences

$$f[0] = f(0) = 3 \qquad f[0,1.2] = \frac{f(1.2)-f(0)}{1.2-0} = \frac{6-3}{1.2} = 2.5$$

$$f[1.2, 3.5] = \frac{f(3.5) - f(1.2)}{3.5 - 1.2} = \frac{1-6}{2.3} = \frac{-5}{2.3}$$

$$f[0, 1.2, 3.5] = \frac{f[1.2, 3.5] - f[0, 1.2]}{3.5 - 0} = \frac{\frac{-5}{2.3} - 2.5}{3.5} \approx -1.33540372670807$$

$$f[3.5, 7] = \frac{f(7) - f(3.5)}{7 - 3.5} = \frac{4}{3.5}$$

$$f[1.2, 3.5, 7] = \frac{f[3.5, 7] - f[1.2, 3.5]}{7 - 1.2} = \frac{\frac{4}{3.5} + \frac{5}{2.3}}{5.8}$$

$$f[0, 1.2, 3.5, 7] = \frac{f[1.2, 3.5, 7] - f[0, 1.2, 3.5]}{7 - 0} \approx 0.272465807912370.$$

Combining all these yields

$$f[0]N_0 + f[0, 1.2]N_1 + f[0, 1.2, 3.5]N_2 + f[0, 1.2, 3.5, 7]N_3$$

$$= 0.27246580791237035x^3 - 2.6159930238962152x^2 + 5.246840865281645x + 3$$

Observe the slight difference in the results due to roundoff errors. Theoretically, the polynomial should be the same because there is a unique polynomial of degree 3 which passes through these four given points and it does not matter how we construct this polynomial. In practice, we may encounter differences due to roundoff errors.

**Solution 1.3.** The error formula has three parts

$$\left(\frac{1}{(n+1)!}\right)\left(f^{(n+1)}(\xi)\right)\left((x - x_0)\dots(x - x_n)\right).$$

The leftmost term is a scaling factor to account for the fact that we are performing a degree $n$ interpolation (using $n+1$ points). The middle term accounts for the part of the error from the function we are approximating. If the function we are approximating behaves wildly (in the sense of having a large derivative), then we would expect the error to be larger. The final term accounts for the fact that we are demanding accuracy at the interpolation points, but not necessarily elsewhere.

This formula cannot be applied directly in this case because we have no information about the function $f(x)$.

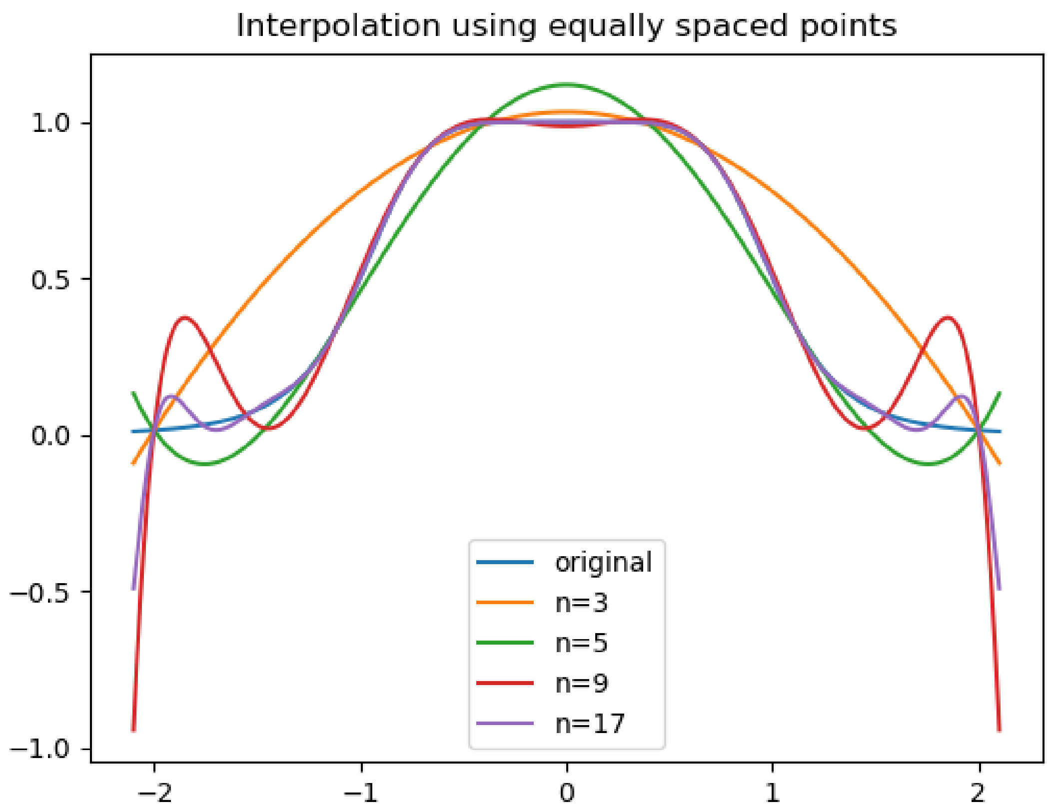
**Solution 1.4.** This polynomial has its zeroes at the interpolation points. The error in between interpolation points can be relatively small, but this term will explode quickly near the endpoints of the interpolation interval as well as outside of the interpolation interval. This means that we should not necessarily trust the output of the interpolations above if we need high accuracy near the endpoints of our interpolation interval.

Chebyshev points are used to minimize the error from this term by making the values of the polynomial as small as possible on the interpolation interval. This is accomplished by selecting more points closer to the end of the interpolation interval and fewer points in the middle (relative to using equally spaced points).

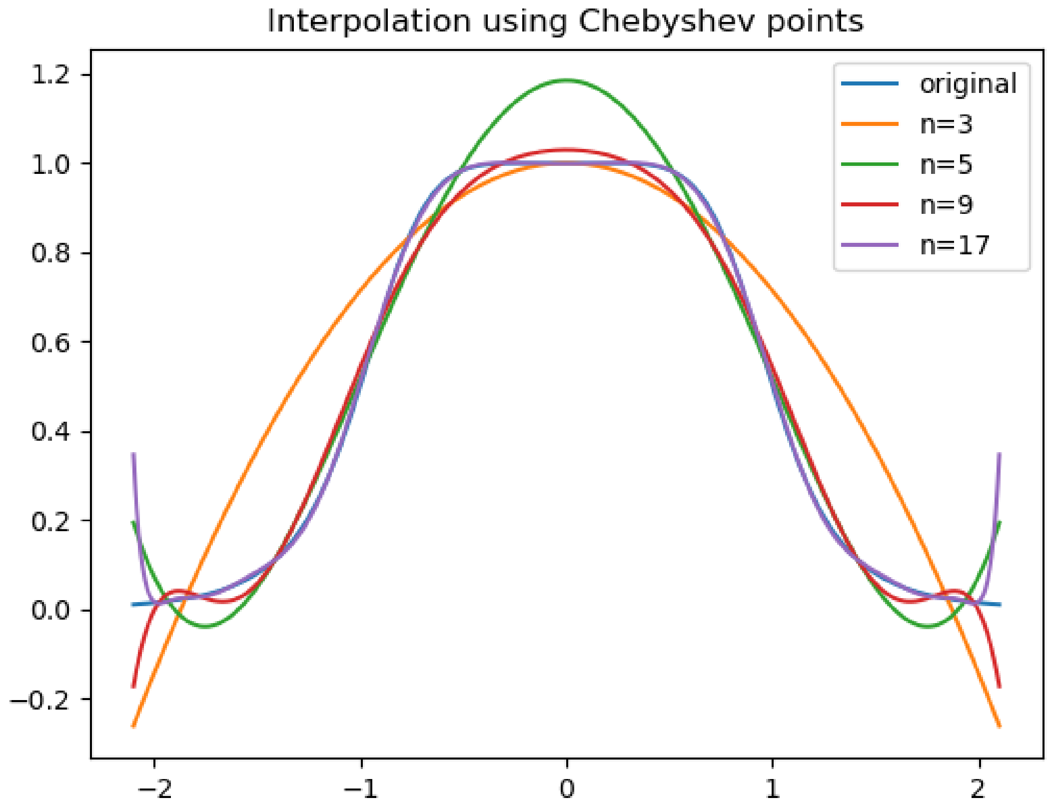
## 2 Interpolation

**Solution 2.1.** See "sol3.py" on Canvas for my sample code. I wrote a better divided differences function which makes use of previous computations to save time and also incorporates the first derivative if desired. (This is used in Hermite interpolation.) I chose to output a function which computes values for the interpolating polynomial.

**Solution 2.2.** In each case, I ran my code and plotted the interpolation polynomials against the original polynomial.




Increasing the degree gives a better approximation near the middle but leads to more oscillation near the endpoints. This is due to the term $(x - x_0) \ldots (x - x_n)$ which is much larger near the endpoints of the interval than it is near the center.




Chebyshev approximation gives slightly worse behavior in the middle of the region, but has better performance near the endpoints compared to equally spaced points. This is due to the choice of interpolation points $x_0, \ldots, x_n$ which decreases the effect of the error term $(x - x_0) \ldots (x - x_n)$ near the endpoints at the cost of some slightly increased error near the middle.

# 3 Numerical Differentiation

**Solution 3.1.** Since we have two points, this leads to a linear polynomial approximation. The line passing through $(x_0, f(x_0))$ and $(x_1, f(x_1))$ is given by the equation

$$P_1(x) = \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0) + f(x_0) = \frac{f(x_0 + h) - f(x_0)}{h}(x - x_0) + f(x_0).$$

According to the formula from class, the error of this interpolation is

$$f(x) - P_1(x) = \frac{f''(\xi(x))}{2!}(x - x_0)(x - x_1).$$

**Solution 3.2.** The derivative of $P_1$ is

$$P_1'(x) = \frac{f(x_0 + h) - f(x_0)}{h}.$$

Taking a derivative of the interpolation error, we find

$$f'(x) - P_1'(x) = \frac{f'''(\xi(x))\xi'(x)}{2}(x - x_0)(x - x_1) + \frac{f''(\xi(x))}{2}(x - x_0 + x - x_1)$$

and plugging in $x_0$ gives

$$|f'(x_0) - P_1'(x_0)| = \left|\frac{f''(\xi(x))}{2}(x_0 - x_1)\right| = \left|\frac{f''(\xi(x))}{2}\right| h$$

which is exactly what we found on the first homework. This result tells us that the limit approximation to the derivative is already obtained by using essentially the simplest nontrivial interpolating polynomial.

**Solution 3.3.** In the previous problem, we used the degree 1 interpolation $P_1(x)$. In this problem, we will use the degree $n$ interpolation polynomial $P_n(x)$. The error formula is

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)\dots(x - x_n).$$

Taking a derivative yields

$$f'(x) - P_n'(x) = \frac{f^{(n+2)}(\xi(x))\xi'(x)}{(n+1)!}(x - x_0)\dots(x - x_n)$$

$$+\frac{f^{(n+1)}(\xi(x))}{(n+1)!}((x-x_1)\dots(x-x_n)+(x-x_0)(x-x_2)\dots(x-x_n)+\dots+(x-x_0)\dots(x-x_{n-1})).$$

When we plug in any interpolation point $x_j$, the first term becomes zero, and we obtain

$$f'(x_j) - P_n'(x_j) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x_j - x_0)\dots(x_j - x_{j-1})(x_j - x_{j+1})\dots(x - x_n)$$

which does not depend on $\xi'(x)$ anywhere. Thus we may use this to obtain a bound by replacing $f^{(n+1)}(\xi(x))$ by the maximum possible value of the $n+1$ derivative.

**Solution 3.4.** First we verify this equation if $m = 1$.

$$L_{n,k}^{(1)}(x) = \frac{d}{dx}\left(\frac{(x - x_0)\dots(x - x_{k-1})(x - x_{k+1})\dots(x - x_{n-1})}{(x_k - x_0)\dots(x_k - x_{k-1})(x_k - x_{k+1})\dots(x_k - x_{n-1})} * \frac{x - x_n}{x_k - x_n}\right)$$

$$= \frac{d}{dx}(L_{n-1,k}(x) * \frac{x - x_n}{x_k - x_n}) = \frac{1}{x_k - x_n}L_{n-1,k}^{(0)}(x) + \frac{x - x_n}{x_k - x_n}L_{n-1,k}^{(1)}(x).$$

Next, we prove by induction

$$L_{n,k}^{(m+1)}(x) = \frac{d}{dx}(L_{n,k}^{(m)}) = \frac{d}{dx}\left(\frac{m}{x_k - x_n}L_{n-1,k}^{(m-1)}(x) + \frac{x - x_n}{x_k - x_n}L_{n-1,k}^{(m)}(x)\right)$$

$$= \frac{m}{x_k - x_n}L_{n-1,k}^{(m)}(x)+\frac{1}{x_k - x_n}L_{n-1,k}^{(m)}(x)+\frac{x - x_n}{x_k - x_n}L_{n-1,k}^{(m+1)}(x) = \frac{m+1}{x_k - x_n}L_{n-1,k}^{(m)}(x)+\frac{x - x_n}{x_k - x_n}L_{n-1,k}^{(m+1)}(x).$$