Assigment II

Pablo Agustín Ortega-Kral

October 25, 2024

Problem 1. Divided difference interpolation

Subproblem 1.a. *Implement a procedure that interpolates f(x) based on a divided difference approach.*

```python
def divided_diff(X: np.array, Y: np.array) -> np.array:
    # Helper array for storing the divided difference table
    N = len(X)
    divided_diff = np.zeros((N, N))
    divided_diff[:, 0] = Y
    # Build the divided difference table, each column is the kth divided difference. We will
     have N-1 columns.
    for i in range(1, N):
        for j in range(N-i):
            divided_diff[j,i] = (divided_diff[j, i-1] - divided_diff[j+1, i-1])/(X[j] - X[i+
    j])
    return divided_diff

def get_interpolated_value(value: float, X: np.array, divided_diff: np.array) -> float:
    N = len(X)
    interpolated_value = 0
    for n in range(N):
        coeff = divided_diff[0, n]
        term = 1
        # Build newton's polynomial coeff_k(x-x_0)(x-x_1)...(x-x_k-1)
        for i in range(n):
            term *= (value - X[i])
        interpolated_value += coeff * term
    return interpolated_value
```

Listing 1: Divided Difference Implementation

In the implemented approach, I first construct the divided difference table for the provided data points. Effectively, this computes $N-1$ divided differences, where each column will be the kth difference. Once the table has been constructed the `get_interpolated_value` uses the precomputed table to build the interpolating polynomial in Newton's form. The coefficients are in indexed from the table, and we construct the difference by taking the datapoint at the step with the desired x value $(\bar{x} - x_0) \dots (\bar{x} - x_{k-1})$.

Subproblem 1.b. *Use your procedure to interpolate* $\cos \pi x$ *at* $x = \frac{3}{10}$*, based on known values of* $(x, \cos \pi x)$ *at the following* x *locations:* $0, \frac{1}{8}, \frac{1}{4}, \frac{3}{8}, \frac{1}{2}$

Using the above code we obtain that $f(\frac{3}{10}) = 0.5878567543147465$. By reading the coefficients in the divided difference table, we know that the interpolating polynomial found was

$$\begin{aligned} &1.00 - 0.61(x - 0.00) \\ &\quad - 4.50(x - 0.00)(x - 0.12) \\ &\quad + 2.82(x - 0.00)(x - 0.12)(x - 0.25) \\ &\quad + 2.80(x - 0.00)(x - 0.12)(x - 0.25)(x - 0.38) \end{aligned}$$

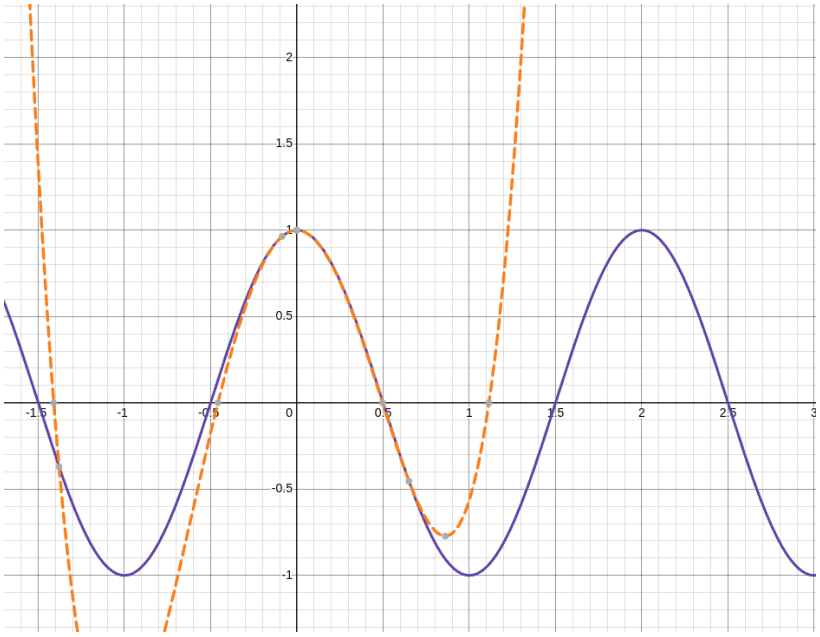


Figure 1: Interpolated polynomial obtained by divided differences. The orange dotted line is the polynomial obtained using the points specified in Q1, while the purple is the function being interpolated $\cos \pi x$

Subproblem 1.c. *Consider the function* $f(x) = \frac{2}{1+9x^2}$. *Use divided differences with the points* $x_i = I\frac{2}{n} - 1$ *and* $i = 0, \ldots, n$. *Interpolate for* 0.07 *with* $n = 2, n = 4, n = 40$. *What is the real value?*

We run the interpolation code using the specified points and report the results in Table 1. Note that we seemingly get close to the real value when increasing the degree; however, as explored in the following section, this does not necessarily indicate a better estimate of the over all polynomial; quite the opposite, it could be a sign of overfitting to the range with the specified points.

Table 1: Estimated values of $f(0.07)$ for different number of points n used in interpolation. Note that the real value $f(0.07) = 1.915525$

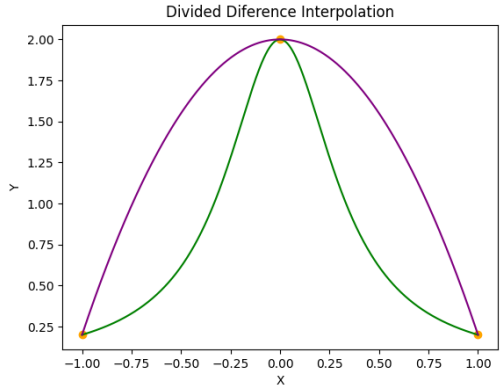
n	Estimated value
2	1.991180
4	1.966875
40	1.915525

Subproblem 1.d. *Estimate the maximum interpolation error. Estimate En for n = 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, and 40, for the function*

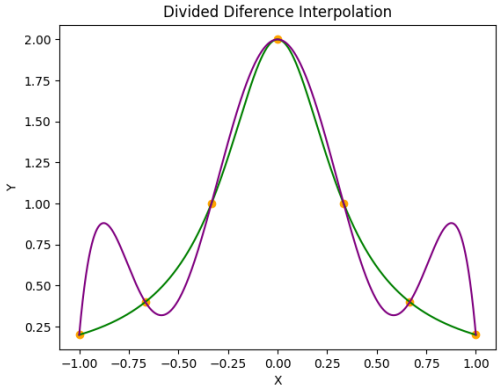
As we increase the degree so too does the maximum error over the specified interval. This is because even though we increase the coverage of the function, we are also increasing the degree of the polynomial, allowing for *overfitting* to the provided datapoints; notice that the function used to generate datapoints means there will not be a point at $x = 1$, so there will always be an error value for an unknown point in the range. Figure 2 illustrates the overfitting problems, notice that even though the polynomial does a better job at passing through the provided data points, it is not a better approximation for the underlying function.

Table 2: Interpolation error for different degrees of the interpolating polynomial

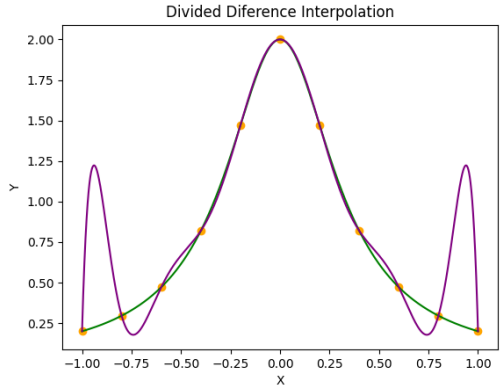
N	Maximum interpolation error
2	0.9350889356804338
4	0.5963194206875789
6	0.6302534939552873
8	0.7681902288008251
10	0.999667273412413
12	1.3515537274515963
14	1.8739582441846299
16	2.6451962355587417
18	3.7840847227275005
20	5.469626491159951
40	290.8879768335694



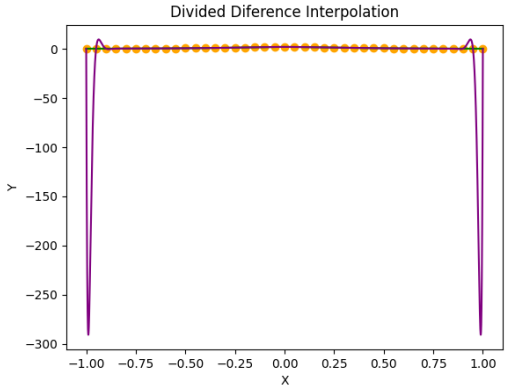
(a) $n = 2$



(b) $n = 6$



(c) $n = 10$



(d) $n = 40$

Figure 2: Interpolating polynomials for $f(x)$ and different number of points.