Homework 5

Blake Hamilton

April 5, 2024

Question 1

(a) The following are the first and second Barycentric Lagrange formulas, respectively:

$$p(x) = \Psi_n(x) \sum_{j=0}^{n} \frac{w_j}{x - x_j} f(x_j)$$

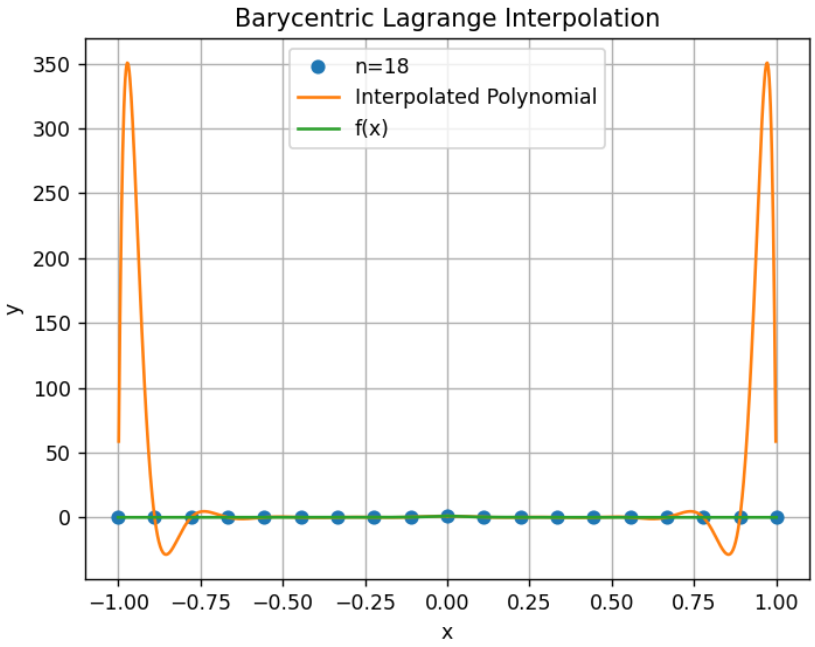
$$p(x) = \frac{\sum_{j=0}^{n} \frac{w_j}{x - x_j} f(x_j)}{\sum_{j=0}^{n} \frac{w_j}{x - x_j}}, \quad x \neq x_j$$

We choose to use the second Barycentric Lagrange formula because the Ψ_n terms cancel out and because we have weights on the numerator and denominator, they will scale together. It is faster to compute, as it is $O(n)$, and it is more stable. We have the following pseudocode:

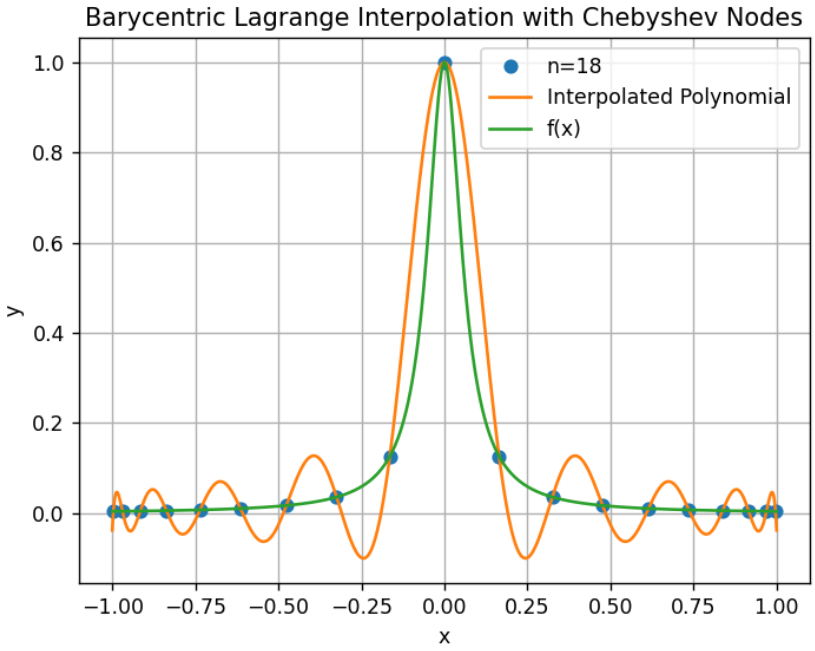
Algorithm 1 Barycentric Lagrange Interpolation

procedure BARYCENTRICLAGRANGEINTERPOLATION($x_{\text{data}}, y_{\text{data}}, x$)
    $n \leftarrow$ length of x_{data}
    $p \leftarrow 0$
    for $j \leftarrow 0$ **to** n **do**
        $w \leftarrow$ compute barycentric weight(x_{data}, j)
        $p \leftarrow p + \dfrac{w \times y_{\text{data}}[j]}{x - x_{\text{data}}[j]}$
    end for
    return p
end procedure
procedure COMPUTEBARYCENTRICWEIGHT(x_{data}, j)
    $w \leftarrow 1$
    for $k \leftarrow 0$ **to** n **do**
        if $j \neq k$ **then**
            $w \leftarrow w \times \dfrac{1}{x_{\text{data}}[j] - x_{\text{data}}[k]}$
        end if
    end for
    return w
end procedure

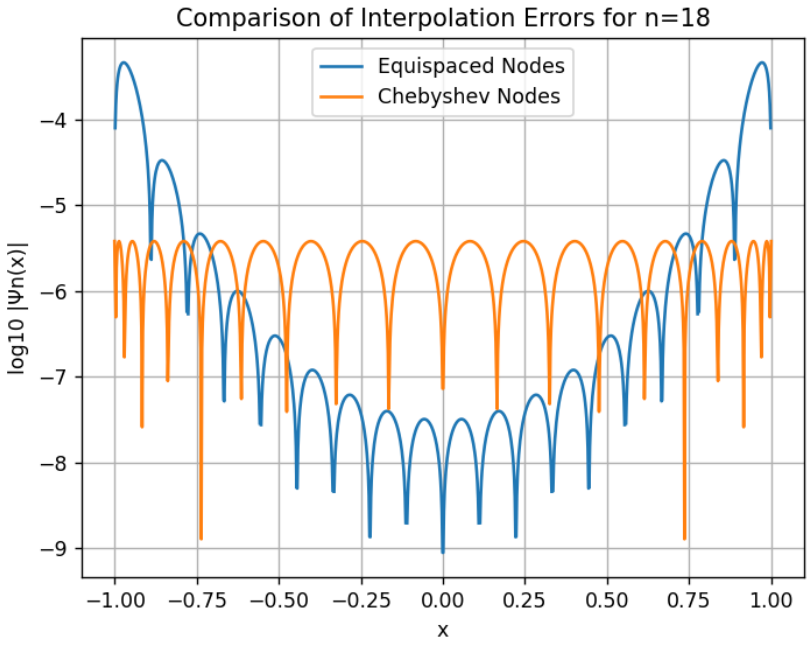
(b) Code in GitHub. We can see at $n = 18$, it behaves poorly at the endpoints. As we increase n, the polynomial starts to fit better for a while, then starts to fit very poorly at the endpoints.



(c) Code in GitHub. We can see that at $n = 18$ again, the polynomial approximates the function much better, especially near the ends. We can confrim that using Chebyshev points works much better for this model and makes the error more consistent across the x range.



(d) The Runge phenomenon describes the way that error decreases in teh middle of a model more than it does near the endpoints as n increases. We can see this below in the graph because as the equispace model has significantly worse error near the endpoints than the middle, the Chebyshev model keeps a consistent error maximum across the whole range. Chebyshev does this by weighting the data near the endpoints more than in the middle, so the model puts more of its weight to the endpoints, helping counteract the RUnge effect.



Question 2

(a) Since we have $p(-1) = y_0, p(1) = y_1, p'(1) = z_1$, we set each term defined in the problem to 1 and the rest to 0. Since we know the system is second order, we can solve each system of three terms of form $ax^2 + bx + c = y$ (with derivative equation $y' = 2ax + b$). For each $p(r)$ given, we plug in our r tot he equation to find the a, b, and c terms:

$$L_0(-1) = 1 = a_0 - b_0 + c_0$$
$$L_0(1) = 0 = a_0 + b_0 + c_0$$
$$L_0'(1) = 0 = 2a_0 + b_0$$
$$L_1(-1) = 0 = a_1 - b_1 + c_1$$
$$L_1(1) = 1 = a_1 + b_1 + c_1$$
$$L_1'(1) = 0 = 2a_1 + b_1$$
$$L_2(-1) = 0 = a_2 - b_2 + c_2$$
$$L_2(1) = 0 = a_2 + b_2 + c_2$$
$$L_2'(1) = 1 = 2a_2 + b_2$$

Solving each system yields:

$$L_0 = \frac{x^2-2x+1}{4}$$

$L_1 = \frac{-x^2+2x+3}{4}$

$L_0 = \frac{x^2-1}{2}$

(b) Using the Lagrange basis we found in part (a), we can find the polynomial $p(x) = y_0L_0 + y_1L_1 + y_2L_2$:

$$\begin{aligned} p(x) &= y_0\frac{x^2-2x+1}{4} + y_1\frac{-x^2+2x+3}{4} + y_2\frac{x^2-1}{2} \\ &= \frac{y_0 - y_1 + 2z_1}{4}x^2 + \frac{y_1 - y_0}{2}x + \frac{y_0 + 3y_1 - 2z_1}{4} \end{aligned}$$

(c) Given the same conditions described in the previous problems adn inspiration from Hermite interpolation, the Newton basis of this equation would be of the form $p(x) = c_0 + c_1(x+1) + c_2(x+1)(x-1)^2$. The $(x-1)$ term is squared because as we need $p(1) = p'(1) = 1$, we need a double root at 1 (as inspired by Hermite). So, we have the following equations:

$$\begin{aligned} p(x) &= c_0 + c_1 + c_2 + (c_1 - c_2)x - c_2x^2 + c_2x^3 \\ p'(x) &= c_1 - c_2 - 2c_2x + 3c_2x^2 \end{aligned}$$

we now plug in each of our given conditions:

$$\begin{aligned} p(-1) = y_0 &= c_0 + c_1 + c_2 - (c_1 - c_2) - c_2 - c_2 \\ &= c_0 \\ p(1) = y_1 &= c_0 + c_1 + c_2 + (c_1 - c_2) - c_2 + c_2 \\ &= c_0 + 2c_1 \\ p'(1) = z_1 &= c_1 - c_2 - 2c_2 + 3c_2 \\ &= c_1 \end{aligned}$$

Now, we can solve for each c to get $c_0 = y_0$, $c_1 = \frac{y_1-y_0}{2}$, and $c_2 = \frac{2z_1-y_1+y_0}{4}$. This would give the resulting polynomial:

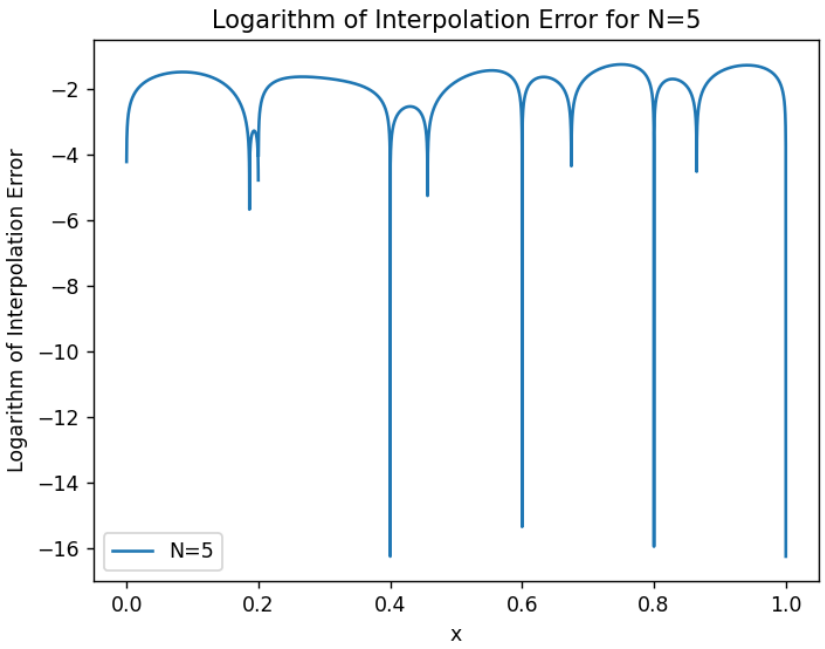
$$p(x) = \frac{2z_1 - y_1 + y_0}{4}x^2 + \frac{y_1 - y_0}{2}x + [\frac{2z_1 - y_1 + y_0}{4} + \frac{y_1 - y_0}{2} + y_0]$$

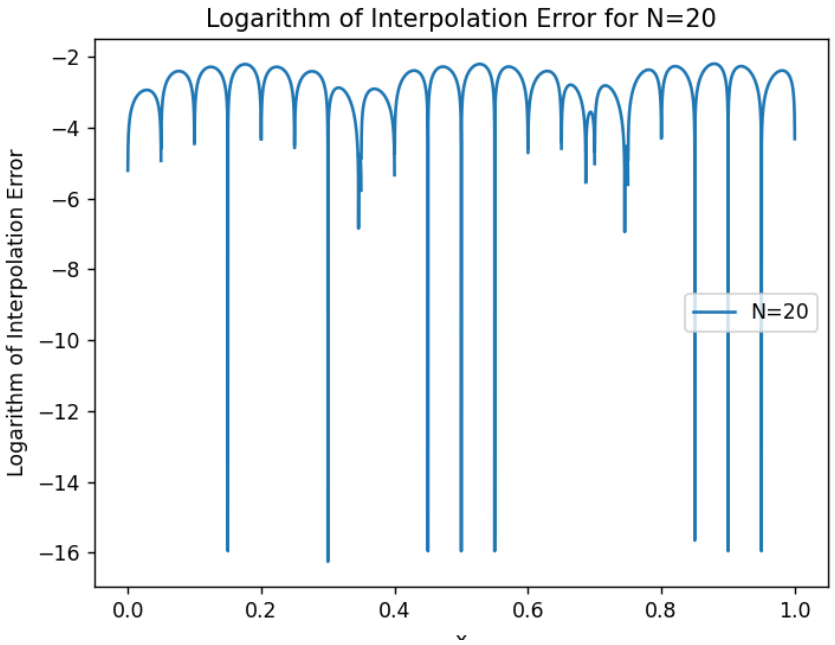
Question 3

(a) Natural splines are made for general curve fitting tasks requiring smoothness, while periodic splines are specifically designed for interpolating periodic data or signals that repeat. Natural splines require that the second derivatives are zero at the endpoints. Periodic splines require that the first and second derivatives at the endpoints are equal to each other at the endpoints.

(b) If we were looking at the derivation of the coefficients for a natural cubic spline in class, we could modify it to work for periodic cubic splines by changing them way we create the linear system for the endpoints, as we need to set the first and second derivatives equal to each other at these points. This will only affect the two points, and result in the sytem matrix having a non zero entry in the top right and bottom left positions.

(c) All the graphs are available on GitHub, but I chose $n = 5, n = 20$ to show below. We can see that as we increase n, the interpolation is more often overlapped with the actual function, so the spline is more accurate s n increases. Although the worst point in the error is about the same between both the models, the $n = 20$ model is overall more accurate.





Question 4

(a) We want to find the linear polynomial $p(x) = a_1x + a_0$ that best fits this data in the least squares sense. The optimization problem is formulated as:

$$\min_{a_0,a_1} \sum_{i=1}^{n} (y_i - (a_1x_i + a_0))^2$$

We know that $\min ||Ma - y||^2 = \min a^T M^T Ma - 2y^T M^T a + y^T y$ and it is of full rank, so $M^T Ma = M^T y$. We have matrix $M = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}$. To find the coefficients a_0 and a_1, we set up the system of normal equations $Ga = b$, where G is the matrix of coefficients $G = M^T M$ and b is the vector of constants. For a linear polynomial, $Ga = b$ is given by:

$$\begin{bmatrix} \sum_{j=0}^{n} 1 & \sum_{j=0}^{n} x_j \\ \sum_{j=0}^{n} x_j & \sum_{j=0}^{n} x_j^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum_{j=0}^{n} y_j \\ \sum_{j=0}^{n} x_j y_j \end{bmatrix}$$

$$\begin{bmatrix} 4 & 6 \\ 6 & 14 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 13 \\ 22 \end{bmatrix}$$

We have the following normal equations, knowing that the minimizer happens when the gradient is equal to zero, which we know is when: $M^TMa = M^Ty$:

$$\begin{bmatrix}1 & 1 & 1 & 1\\ 0 & 1 & 2 & 3\end{bmatrix}\begin{bmatrix}1 & 0\\ 1 & 1\\ 1 & 2\\ 1 & 3\end{bmatrix}\begin{bmatrix}a_0\\ a_1\end{bmatrix} = \begin{bmatrix}1 & 1 & 1 & 1\\ 0 & 1 & 2 & 3\end{bmatrix}\begin{bmatrix}1\\ 4\\ 2\\ 6\end{bmatrix}$$

$$\begin{bmatrix}a_0\\ a_1\end{bmatrix} = G^{-1}M^Ty$$

$$\begin{bmatrix}a_0\\ a_1\end{bmatrix} = \begin{bmatrix}\frac{7}{10} & \frac{-3}{10}\\ \frac{-3}{10} & \frac{1}{5}\end{bmatrix}\begin{bmatrix}1 & 1 & 1 & 1\\ 0 & 1 & 2 & 3\end{bmatrix}\begin{bmatrix}1\\ 4\\ 2\\ 6\end{bmatrix} = \begin{bmatrix}1.3\\ 1.3\end{bmatrix}$$

(b) As the $||\cdot||^2$ norm is squaring each element, we know that $q_w(a_0, a_1)$ can be written as $||D(Ma-y)||$ because it does the same (squaring the difference of each element by the approximation, multiplied by the weight). To minimize $q_w(a_0, a_1) = \sum_{i=0}^{3} w_i(p(x_i) - y_i)^2$, we want to multiply the diagonal matrix D by both sides of our original $Ax = b$ formatted equation. We know from before that $\min ||Ma-y||^2 = \min a^TM^TMa - 2y^TM^Ta + y^Ty$, we need to distribute the D for $||D(Ma - y)||$: $\min ||D(Ma - y)||^2 = \min a^TDM^TMa - 2Dy^TM^Ta + y^TDy$. As the matrix is still full rank, we have the normal equations $M^TDMa = M^TDy$ with $D = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 2 & 0 & 0\\ 0 & 0 & 3 & 0\\ 0 & 0 & 0 & \sqrt{6}\end{bmatrix}$:

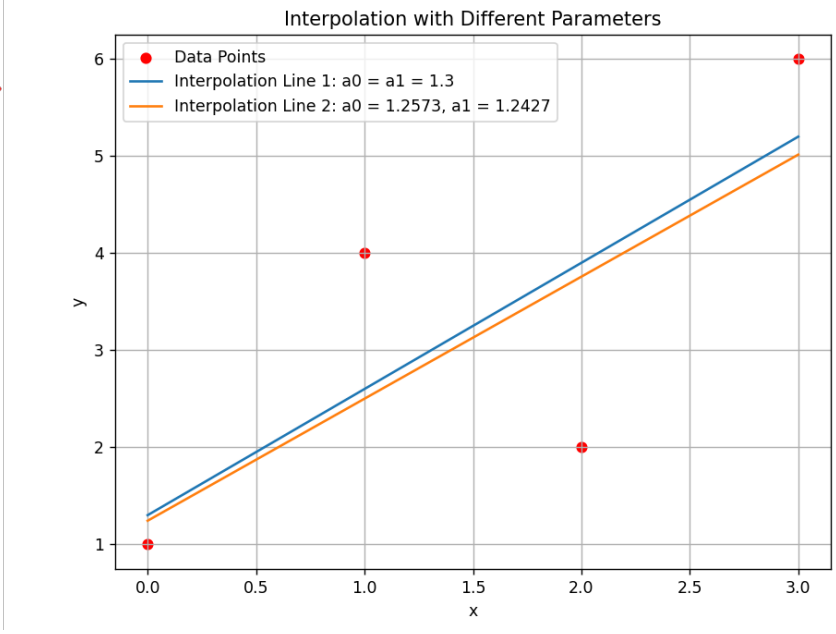
$$\begin{bmatrix}1 & 1 & 1 & 1\\ 0 & 1 & 2 & 3\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 2 & 0 & 0\\ 0 & 0 & 3 & 0\\ 0 & 0 & 0 & \sqrt{6}\end{bmatrix}\begin{bmatrix}1 & 0\\ 1 & 1\\ 1 & 2\\ 1 & 3\end{bmatrix}\begin{bmatrix}a_0\\ a_1\end{bmatrix} = \begin{bmatrix}1 & 1 & 1 & 1\\ 0 & 1 & 2 & 3\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 2 & 0 & 0\\ 0 & 0 & 3 & 0\\ 0 & 0 & 0 & \sqrt{6}\end{bmatrix}\begin{bmatrix}1\\ 4\\ 2\\ 6\end{bmatrix}$$

(c) We now solve the above system to find the coefficients a_0 and a_1. We know $a = G^{*-1}M^TDy$ for $G^* = M^TDM$. We first find G^{*-1} :

$$G^{*-1} = (M^TDM)^{-1} = (\begin{bmatrix}1 & 1 & 1 & 1\\ 0 & 1 & 2 & 3\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 2 & 0 & 0\\ 0 & 0 & 3 & 0\\ 0 & 0 & 0 & \sqrt{6}\end{bmatrix}\begin{bmatrix}1 & 0\\ 1 & 1\\ 1 & 2\\ 1 & 3\end{bmatrix})^{-1} = \begin{bmatrix}0.5225 & -0.2225\\ -0.2225 & 0.1225\end{bmatrix}$$

We can now solve for a:

$$\begin{bmatrix}a_0\\ a_1\end{bmatrix} = G^{*-1}M^TDy = \begin{bmatrix}0.5225 & -0.2225\\ -0.2225 & 0.1225\end{bmatrix}\begin{bmatrix}1 & 1 & 1 & 1\\ 0 & 1 & 2 & 3\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 2 & 0 & 0\\ 0 & 0 & 3 & 0\\ 0 & 0 & 0 & \sqrt{6}\end{bmatrix}\begin{bmatrix}1\\ 4\\ 2\\ 6\end{bmatrix} = \begin{bmatrix}1.2573\\ 1.2427\end{bmatrix}$$



Intuitively, the higher weighted points bring the line closer to them. As the third point is the "heaviest", we can see that it brings the weighted line down. This is as expected, as the heaviest point is far below the original interpolated line.