Comparison of the Numerical Stability of Barycentric and Standard Lagrange Polynomial Interpolation

Mason Mault
(Dated: July 8, 2025)

Lagrange Polynomial Interpolation

For $n+1$ nodes, the problem of finding a polynomial $p(x) \in P_n$ interpolating a function $f(x)$ has a unique solution. Written in Lagrange form, this solution is

$$p(x) = \sum_{j=0}^{n} f_j \ell_j(x) \tag{1}$$

where $\ell_j(x)$ is the j^{th} Lagrange polynomial, the unique polynomial taking the value 1 at $x = x_j$ and 0 at all other nodes $x = x_k$

$$\ell_j(x) = \frac{\prod_{k=0,k\neq j}^{n}(x - x_k)}{\prod_{k=0,k\neq j}^{n}(x_j - x_k)} \tag{2}$$

Closely related is the node polynomial $\ell(x)$, taking the value 0 at all nodes

$$\ell(x) = \prod_{k=0}^{n}(x - x_k) \tag{3}$$

Trefethen lists three computational shortcomings arising from the Lagrange form of $p(x)$:

1) Each evaluation of $p(x)$ requires $\mathcal{O}(n^2)$ additions and multiplications.
2) Adding a new data pair (x_{n+1}, f_{n+1}) requires a new computation from scratch.
3) The computation is numerically unstable.

Improving Lagrange's Formula

An alternative method of polynomial interpolation is Newton's approach. The Newton table of divided differences is computed using a known recursive formula, then $p(x)$ is evaluated with Newton's interpolation formula. The table requires $\mathcal{O}(n^2)$ subtractions and divisions. However, $p(x)$ can be evaluated in $\mathcal{O}(n)$ with the use of nested multiplication, less floating point operations (flops) than directly using eq 1. But, we can do better. Notice that in both Newton and Lagrange's approach, the quantities computed in $\mathcal{O}(n^2)$ depend on the data f_j. If these $\mathcal{O}(n^2)$ quantities can be decoupled from the data, they can be precomputed for a fixed set of nodes and reused to interpolate multiple functions. This is the role of barycentric interpolation.

Barycentric Interpolation Formula

Define the barycentric weights w_j as (derivation worked in chapter 5 notes)

$$w_j = \frac{1}{\prod_{k\neq j}(x_j - x_k)} = \frac{1}{\ell'(x_j)} \tag{4}$$

With this, eq 2 becomes

$$\ell_j(x) = \ell(x)\frac{w_j}{x - x_j} \tag{5}$$

And eq 1 becomes the first form of the barycentric interpolation formula

$$p(x) = \ell(x) \sum_{j=0}^{n} \frac{w_j}{x - x_j} f_j \tag{6}$$

However, this formula has been shown by Higham to be conditionally unstable. Higham also showed that the following second form is stable

$$p(x) = \frac{\sum_{j=0}^{n} \frac{w_j}{x - x_j} f_j}{\sum_{j=0}^{n} \frac{w_j}{x - x_j}} \tag{7}$$

This formula is entirely equivalent to eq 1, but offers several computational advantages.

1) The weights w_j are computed independently of x and f in $\mathcal{O}(n^2)$ flops. These weights may be reused to interpolate any function on the same grid. For special grids, these weights have simple analytic formulas.
2) Adding a new data pair (x_{n+1}, f_{n+1}) requires $\mathcal{O}(n)$ flops, without recomputing from scratch.
3) Node ordering has no effect on barycentric interpolation, unlike Newton's approach. This sensitivity to ordering can lead to numerical instability for large n.
4) The weights appear identically in the numerator and denominator, so interpolation is independent of scaling. Every interpolation method yields exact results at nodes, but this cancellation of common factors makes barycentric interpolation a robust numerical process for evaluation between nodes.

Barycentric Interpolation at Chebyshev Points

Using equispaced nodes, the weights for barycentric interpolation vary exponentially. These equispaced nodes lead to an ill-conditioned problem, and small changes in the data can lead to huge changes in the interpolant. To avoid this, results from approximation theory tell us the right approach is to use point sets (like Chebyshev) clustered at the endpoints of the interval with an asymptotic density proportional to $(1 - x^2)^{-1/2}$ as $n \to \infty$ (if the nodes do not converge to this distribution, the condition numbers grow exponentially as $n \to \infty$). This asymptotic density places the weights on a comparable scale. Chebyshev points of the first kind lead to the following weights

$$w_j = (-1)^j \cdot \begin{cases} \frac{1}{2}, & \text{if } j = 0 \text{ or } j = n \\ 1, & \text{otherwise} \end{cases} \tag{8}$$

In the case of barycentric interpolation through Chebyshev points, the process is numerically stable and efficient, with analytical weights and only $\mathcal{O}(n)$ flops for evaluation. Using these weights, eq 7 becomes

$$p(x) = \frac{\sum_{j=0}^{n} \frac{(-1)^j}{x - x_j} f_j}{\sum_{j=0}^{n} \frac{(-1)^j}{x - x_j}} \tag{9}$$

with terms $j = 0$ and $j = n$ being multiplied by $1/2$.

Numerical Stability of Weights

The first form of barycentric interpolation can be sensitive to overflow and underflow when computing the weights. Consider interpolating the interval $[a, b]$ with length $4C$ (where C is the capacity of the interval). As $n \to \infty$ the magnitude of the weights will scale exponentially as C^{-n}, and issues can arise if n is large or C differs significantly from 1. Rescaling the weights uniformly by C^n typically keeps the result between 10^{-290} and 10^{290}, ensuring no digits of accuracy are lost. Using the second form of barycentric interpolation removes this problem altogether.

Division by Zero

While the second form of barycentric interpolation solves the first form's problem of overflow and underflow for weights, issues may still arise when x is close but not equal to x_j, as the term $w_j/(x - x_j)$ becomes huge. However, Henrici showed that any inaccuracy arising as $x \to x_j$ appears symmetrically in the numerator and denominator and cancels, resulting in a numerically stable formula. If $x = x_j$, eq 7 calls for division by zero. By definition, when $x - x_j = 0$ the interpolant should take the value $f(x_j)$. Luckily, a fix takes a few lines of code. Inside the barycentric loop, a variable tracks the indices where $x - x_j = 0$, and after the loop, those indices are set directly to $f(x_j)$. With this fix, the second form of barycentric interpolation is entirely reliable in practice.

Derivation of Barycentric Weights for Chebyshev Points

By manipulating Chebyshev polynomials T_n, the barycentric weights for Chebyshev points (of the second kind) become remarkably simple. First, we show that for $n+1$ Chebyshev points, the node polynomial eq 3 can be written as

$$\ell(x) = 2^{-n}(T_{n+1}(x) - T_{n-1}(x)) \tag{10}$$

Though as we will see, 2^{-n} is washed out by the symmetry of the weights in eq 7.

Theorem 4.1 in ATAP states for $n \geq 1$ and $0 \leq m \leq n$, the following Chebyshev polynomials take the same value on an $n+1$ point Chebyshev grid

$$T_m, T_{2n-m}, T_{2n+m}, T_{4n-m}, ... \tag{11}$$

choosing $m = n - 1$

$$T_m = T_{n-1} \tag{12}$$

$$T_{2n-m} = T_{2n-(n-1)} = T_{n+1} \tag{13}$$

But $T_m = T_{2n-m}$, so $T_{n-1}(x_j) = T_{n+1}(x_j)$ for all $j = 0, ..., n$ (that is, all j on the $n+1$ point grid, proving eq 10 takes the value 0 where required.

Next, we show

$$T'_{n+1}(x_j) - T'_{n-1}(x_j) = 2n(-1)^j, \qquad 1 \leq j \leq n-1 \tag{14}$$

and twice this for $j = 0$ and $j = n$.

By definition of Chebyshev polynomials, $T_n(x) = \cos(n\theta)$, where $x = \cos(\theta)$. By the chain rule

$$T'_n(x) = -n\sin(n\theta)\frac{d\theta}{dx} = \frac{n\sin(n\theta)}{sin(\theta)} \tag{15}$$

The difference becomes

$$T'_{n+1}(x) - T'_{n-1}(x) = \frac{1}{\sin(\theta)}\left[(n+1)\sin((n+1)\theta) - (n-1)\sin((n-1)\theta)\right] \tag{16}$$

$$= \frac{1}{\sin(\theta)}\left[n\left(\sin((n+1)\theta) + \sin((n-1)\theta)\right) + \left(\sin((n+1)\theta) - \sin((n-1)\theta)\right)\right] \tag{17}$$

$$= \frac{1}{\sin(\theta)}\left[n\left(\sin(n\theta+\theta) + \sin(n\theta-\theta)\right) + \left(\sin(n\theta+\theta) - \sin(n\theta-\theta)\right)\right] \tag{18}$$

Using trigonometric identities for $\sin(a) \pm \sin(b)$

$$= \frac{1}{\sin(\theta)} \left[2n \cos\left(\frac{(n\theta+\theta)+(n\theta-\theta)}{2} \right) \sin\left(\frac{(n\theta+\theta)-(n\theta-\theta)}{2} \right) \right. \tag{19}$$

$$\left. + 2\sin\left(\frac{(n\theta+\theta)+(n\theta-\theta)}{2} \right) \cos\left(\frac{(n\theta+\theta)-(n\theta-\theta)}{2} \right) \right] \tag{20}$$

$$= \frac{1}{\sin(\theta)} \left[2n\cos(n\theta)\sin(\theta) + 2\sin(n\theta)\cos(\theta) \right] \tag{21}$$

Evaluating at $x = x_j$ is equivalent to setting $\theta = \theta_j = j\pi/n$, giving

$$T'_{n+1}(x_j) - T'_{n-1}(x_j) = \frac{1}{\sin(j\pi/n)} \left[2n\cos(j\pi)\sin(j\pi/n) + 2\sin(j\pi)\cos(j\pi/n) \right] \tag{22}$$

$$= \frac{1}{\sin(j\pi/n)} \left[2n\cos(j\pi)\sin(j\pi/n) \right] \tag{23}$$

$$= \frac{1}{\sin(j\pi/n)} \left[2n(-1)^j \sin(j\pi/n) \right] \tag{24}$$

Before continuing, notice if $j = 0$ or $j = n$, the expression is of indeterminant form $0/0$. However, if $1 \leq j \leq n-1$, it is safe to make the simplification

$$T'_{n+1}(x_j) - T'_{n-1}(x_j) = 2n(-1)^j \tag{25}$$

proving eq 14 for $1 \leq j \leq n-1$. Now, the cases $j = 0$ and $j = n$ are considered through a limiting argument of eq 16. When $j \to 0$, $\theta_j \to 0$, and eq 16 becomes

$$T'_{n+1}(x_0) - T'_{n-1}(x_0) = \lim_{\theta\to 0} \frac{(n+1)\sin((n+1)\theta) - (n-1)\sin((n-1)\theta)}{\sin(\theta)} \tag{26}$$

$$= \lim_{\theta\to 0} \frac{(n+1)^2\cos((n+1)\theta) - (n-1)^2\cos((n-1)\theta)}{\cos(\theta)} \tag{27}$$

$$= \frac{(n+1)^2\cos(0) - (n-1)^2\cos(0)}{\cos(0)} \tag{28}$$

$$= (n+1)^2 - (n-1)^2 = 4n \tag{29}$$

The same computation for $j \to n$, $\theta_j \to \pi$ gives

$$T'_{n+1}(x_n) - T'_{n-1}(x_n) = \lim_{\theta\to \pi} \frac{(n+1)^2\cos((n+1)\theta) - (n-1)^2\cos((n-1)\theta)}{\cos(\theta)} \tag{30}$$

$$= \frac{(n+1)^2\cos((n+1)\pi) - (n-1)^2\cos((n-1)\pi)}{\cos(\pi)} \tag{31}$$

$$= \frac{(n+1)^2(-1)^{n+1} - (n-1)^2(-1)^{n-1}}{-1} \tag{32}$$

$$= (-1)^{n+1}\frac{((n+1)^2 - (n-1)^2)}{-1} \tag{33}$$

$$= -(-1)^{n+1}((n+1)^2 - (n-1)^2) = (-1)^{n+2}4n \tag{34}$$

$$= (-1)^n 4n \tag{35}$$

Notice when $x_j = x_0$ and $x_j = x_n$, $T'_{n+1}(x_j) - T'_{n-1}(x_j)$ takes twice the value when compared with all other values of j. This is discussed shortly.

We are in a position to arrive at the weights. Using the node polynomial given in eq 10, eq 5 becomes

$$\ell_j(x) = \ell(x)\frac{w_j}{x - x_j} = 2^{-n} w_j \frac{(T_{n+1}(x) - T_{n-1}(x))}{x - x_j} \tag{36}$$

By definition of the weights, and by differentiating eq 10, eq 4 becomes

$$w_j = \frac{1}{\ell'(x_j)} = \frac{2^n}{T'_{n+1}(x_j) - T'_{n-1}(x_j)} \tag{37}$$

Because $w_j \propto 1/(T_{n+1}(x_j) - T_{n-1}(x_j))$, we are in a position to understand the difference in weights for $j = 0$,n and all others j's.

$$T_{n+1}(x_0) - T_{n-1}(x_0) = 4n \tag{38}$$

$$T_{n+1}(x_j) - T_{n-1}(x_j) = 2n(-1)^j \qquad 1 \le j \le n-1 \tag{39}$$

$$T_{n+1}(x_n) - T_{n-1}(x_n) = 4n(-1)^n \tag{40}$$

$$\tag{41}$$

Because of the extraneous factor of 2, the weights w_0 and w_n are multiplied by 1/2 so all weights have the same absolute magnitude. We have proven the denominator of eq 37 satisfies eq 14, giving the final form of the weights as

$$w_j = \frac{2^n}{2n(-1)^j} = \frac{2^{n-1}}{n}(-1)^j \qquad 1 \le j \le n-1 \tag{42}$$

and half this value for $j = 0$ and $j = n$. Compared with eq 9, these weights seem to have extraneous factors of n. However, any factors independent of j will cancel because the weights appear symmetrically in the numerator and denominator, and the sum is with respect to j. Thus, the weights above lead directly to eq 9.