Properties of interpolation algorithms

Jesse Wei

June 12, 2022

1 Newton interpolation

Before 2000, Newton's divided differences method was considered the best polynomial interpolation method. Given data points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, this method creates a polynomial of the form

$$p(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots + f[x_0, \ldots, x_n](x - x_0) \ldots (x - x_{n-1}) \quad (1.1)$$

where the $[y_0, \ldots, y_n]$ coefficients use the notation for Newton's tableau of divided differences, which is calculated recursively. Direct evaluation of $p(x)$ requires $\mathcal{O}(n^2)$ additions and multiplications. The number of multiplications follows the pattern $0+1+2+\cdots+n = \frac{n(n+1)}{2} = \mathcal{O}(n^2)$. With nesting, however, evaluation at a point x is an $\mathcal{O}(n)$ operation. The addition of another point requires only 1 more subtraction and 1 more multiplication, as shown in (1.2), and this generalizes to the addition of n points.

$$p(x) = a_0 + (x - x_0)(a_1 + (x - x_1)(a_2 + \cdots + (x - x_{n-1})a_n) \cdots) \quad (1.2)$$

Given one more point x_{n+1}, recalculation of $p(x)$ requires the $\mathcal{O}(n)$ calculation in (1.3).

$$\{y(x_{n+1}), f[x_n, x_{n+1}], f[x_{n-1}, x_n, x_{n+1}], \cdots, f[x_0, x_1, \cdots, x_{n+1}]\} \quad (1.3)$$

As linear time complexity is acceptable, this method remained the standard until 2000, when the Lagrange method was improved.

2 Lagrange interpolation

Lagrange interpolation was originally considered to be purely for theoretical purposes and not suitable for actual practice because a Lagrange polynomial $p(x)$ normally requires $\mathcal{O}(n^2)$ additions and multiplications to evaluate at a point x, and the addition of a new data point $(x, f(x))$ requires $p(x)$ to be recalculated from scratch. The barycentric formula (2.1), however, allows for an $\mathcal{O}(n)$ evaluation of $p(x)$ for some x once w_j, which are independent of x, are known (an $\mathcal{O}(n^2)$ operation). Also, adding another point x_{n+1}, f_{n+1} requires only an $\mathcal{O}(n)$ updating of the weights w_j. In these ways, then, the Lagrange method is on par with Newton's method.

$$p(x) = \frac{\sum_{j=0}^{n} \frac{w_j}{x - x_j} f_j}{\sum_{j=0}^{n} \frac{w_j}{x - x_j}} \quad (2.1)$$

Where the improved Lagrange interpolation method exceeds Newton's interpolation method is that the weights w_j are also independent of the data f_j, so once the weights are known, any function can be interpolated in $\mathcal{O}(n)$. On the other hand, Newton's interpolation method requires a new divided different tableau to be calculated for each new function. Lastly, the Lagrange interpolation formula does not depend on the order of the data inputs, whereas Newton's method does [BT04].

3 Point distributions

Both methods experience stability issues that are inherent to interpolation in general. Notably, there is the problem of node selection, with the most obvious method being uniformly spaced nodes on the interval $[-1, 1]$. According to [BT04], uniform spacing (with $h = 2/n$) yields

$$w_j = (-1)^j \binom{n}{j} \tag{3.1}$$

For large n, the Runge phenomenon occurs because the w_j vary by exponentially large factors due to the $\binom{n}{j}$ term. The error is largest at the endpoints of the interval, but even data near the center of the interval may experience error.

As such, a better approach is to use interpolation points that are clustered at the endpoints of the interval, with density proportional to $(1 - x^2)^{-1/2}$.

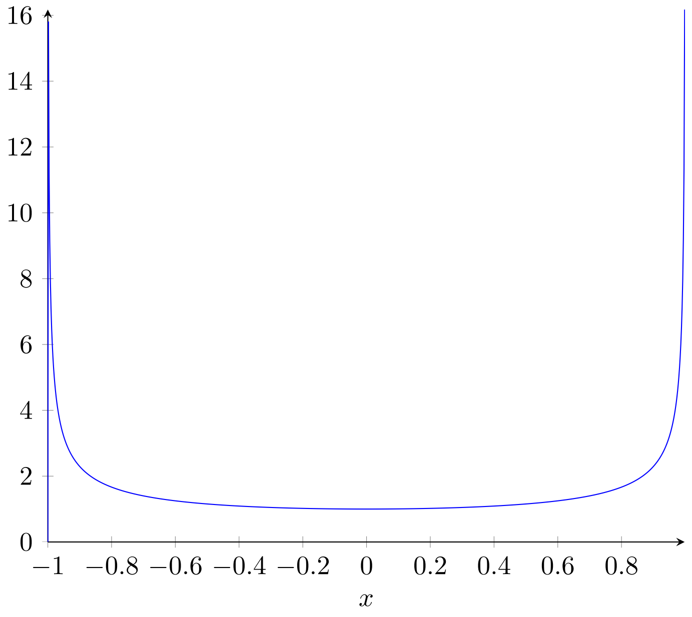


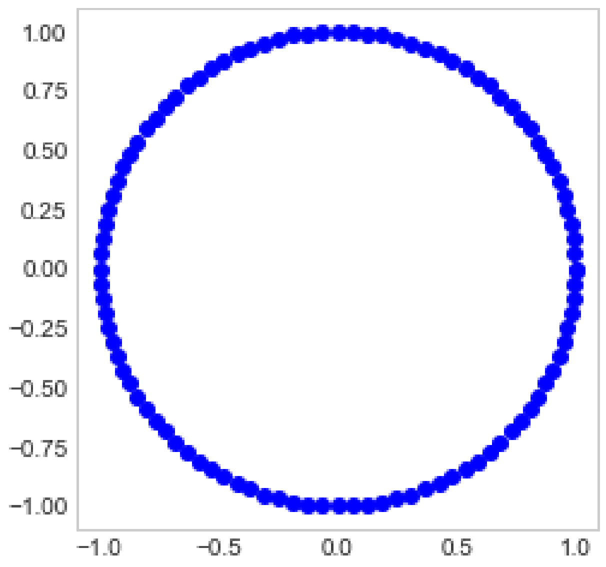
Figure 1: $(1 - x^2)^{-1/2}$

One such point distribution is the family of Chebyshev points, which are formed by projecting equally spaced points on the unit circle onto the interval $[-1, 1]$. From [BT04], the *Chebyshev points of the first kind* are generated by

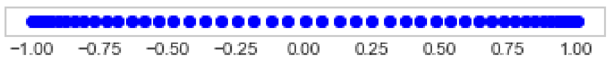
$$x_j = \cos \frac{(2j+1)\pi}{2n+2}, \quad j = 0, \dots, n \tag{3.2}$$

with weights

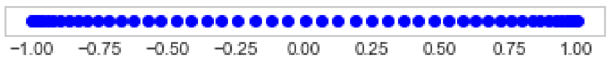
$$w_j = (-1)^j \sin \frac{(2j+1)\pi}{2n+2} \tag{3.3}$$



(a) Unit circle, $n = 100$

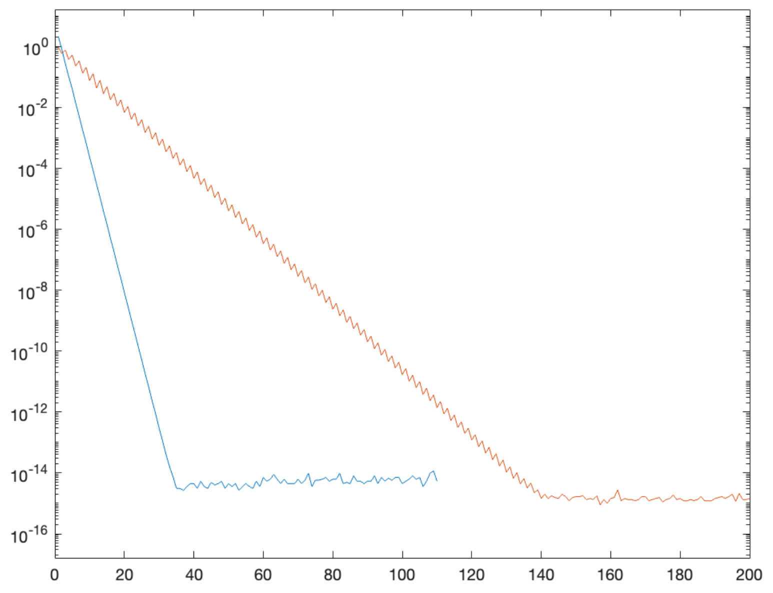


(b) Unit circle projection onto $[-1, 1]$



(c) Chebyshev points ($n = 50$) generated by (3.2)

Figure 2: `chebyshev.ipynb`



Replication of Figure 6.1 from [BT04]

The lower function represents errors for $e^x / \cos x$, and the upper function represents $(1 + 16x^2)^{-1}$. Like [BT04], we choose to stop plotting errors for the lower function at $n \approx 110$.

Plotting errors for the lower function up to $n = 200$ would simply require changing line 40 to

```
semilogy(n_array, errors_lower, '-', n_array, errors_upper, '-');
```

```
% Replicates Fig 6.1 on pg. 508 of [BT04]
% The barycentric interpolation/chebyshev points code is
% from pages 506-507 of [BT04]

% Code omitted in public version of document.
```

figure6_1.m

References

[BT04] Jean-Paul Berrut and Lloyd N. Trefethen. "Barycentric Lagrange Interpolation". In: *SIAM Review* 46.3 (2004), pp. 501–517. DOI: `https://doi.org/10.1137/S0036144502417715`.