Introduction to Numerical Mathematics for Data Scientists

Joseph Marziale February 15, 2024

Contents

1 Lecture on 15 February 2024

1.1 Lagrange polynomial interpolation

Suppose we have a function $f(x)$ that passes through the points $(x_1, y_1),\ (x_2, y_2),\ (x_n, y_n)$. The goal of polynomial interpolation of a function f is to obtain some

$$f(x) \approx y_1 l_1(x) + y_2 l_2(x) + \ldots + y_n l_n(x) \tag{1}$$

that also passes through $(x_1, y_1),\ (x_2, y_2),\ (x_n, y_n)$. For this to be true, the basis functions $l_i(x)$ must obey the condition

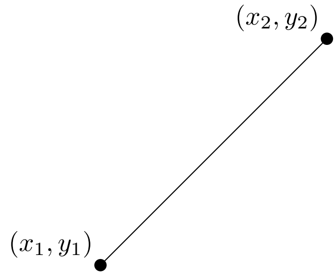
$$\boxed{l_i(x_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}} \qquad \textit{condition } 1 \tag{2}$$

such that $l_1(x_1) = 1,\ l_1(x_2) = 0,\ l_3(x_1) = 0,\ l_3(x_3) = 1$, etc. *If* the basis functions $l_i(x)$ also obey the condition

$$\boxed{\sum_i l_i(x) = 1}, \qquad \textit{condition } 2 \tag{3}$$

then these are called Lagrange polynomials.

To see how $l_i(x)$ are developed, let's look at a simple linear interpolation of a function passing through $(x_1, y_1),\ (x_2, y_2)$.



Derived from our generic algebraic formula for a linear equation $f(x) = mx + b$, the formula for this linear interpolant is

$$f(x) = y_1 + \left(\frac{y_2 - y_1}{x_2 - x_2}\right)(x - x_1) \tag{4}$$

$$= \frac{y_1(x_2 - x_1)}{x_2 - x_1} + \frac{y_2(x - x_1)}{x_2 - x_1} - \frac{y_1(x - x_1)}{x_2 - x_1} \tag{5}$$

$$= y_1 \frac{x_2 - x}{x_2 - x_1} + y_2 \frac{x - x_1}{x_2 - x_1} \tag{6}$$

$$= y_1 \underbrace{\frac{x - x_2}{x_1 - x_2}}_{l_1(x)} + y_2 \underbrace{\frac{x - x_1}{x_2 - x_1}}_{l_2(x)}. \tag{7}$$

To see whether or not this function is a Lagrange interpolator, we will see whether or not the basis functions $l_i(x)$ obey the 2 conditions listed earlier:

- Condition 1

$$l_1(x_1) = \frac{x_1 - x_2}{x_1 - x_2} = 1 \quad \checkmark \tag{8}$$

$$l_1(x_2) = \frac{x_2 - x_2}{x_1 - x_2} = 0 \quad \checkmark \tag{9}$$

$$l_2(x_1) = \frac{x_2 - x_1}{x_2 - x_1} = 1 \quad \checkmark \tag{10}$$

$$l_2(x_2) = \frac{x_2 - x_1}{x_2 - x_1} = 0 \quad \checkmark \tag{11}$$

- Condition 2

$$\sum_i l_i(x) = \frac{x - x_2}{x_1 - x_2} + \frac{x - x_1}{x_2 - x_1} = \frac{x_2 - x + x - x_1}{x_2 - x_1} = \frac{x_2 - x_1}{x_2 - x_1} = 1 \quad \checkmark \tag{12}$$

Therefore we assert that $l_i(x)$ are Lagrange polynomials.
In general, Lagrange interpolations of $f(x)$ take on the form

$$\boxed{f(x) \approx \sum_i y_i l_i(x), \qquad l_i(x) = \prod_{j=1, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}}. \tag{13}$$

So a 3-point interpolation of a function $f(x)$ that passes through the points $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ will be formulated as

$$f(x) \approx y_1 l_1(x) + y_2 l_2(x) + y_3 l_3(x), \tag{14}$$

$$l_1(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} \tag{15}$$

$$l_2(x) = \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} \tag{16}$$

$$l_3(x) = \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}. \tag{17}$$

1.2 Runge phenomenon

It is intuitive to think that the accuracy of a polynomial interpolation increases as we increase the number of points on which we are interpolating. So for example, you would expect that the 3-point interpolation of $f(x)$ is more accurate than the 2-point interpolation of the same $f(x)$. However, we will show that if $x_1, x_2, \ldots, x_n$ are *uniformly spaced* (such that $x_2 - x_1 = x_3 - x_2 = \ldots = \textit{constant}$), then the accuracy is not guaranteed to improve with increased x_i. This is known as the Runge phenomenon, as the Runge function

$$R(x) = \frac{1}{1 + 25x^2}, \qquad x \in [-1, 1] \tag{18}$$

clearly demonstrates this behavior.

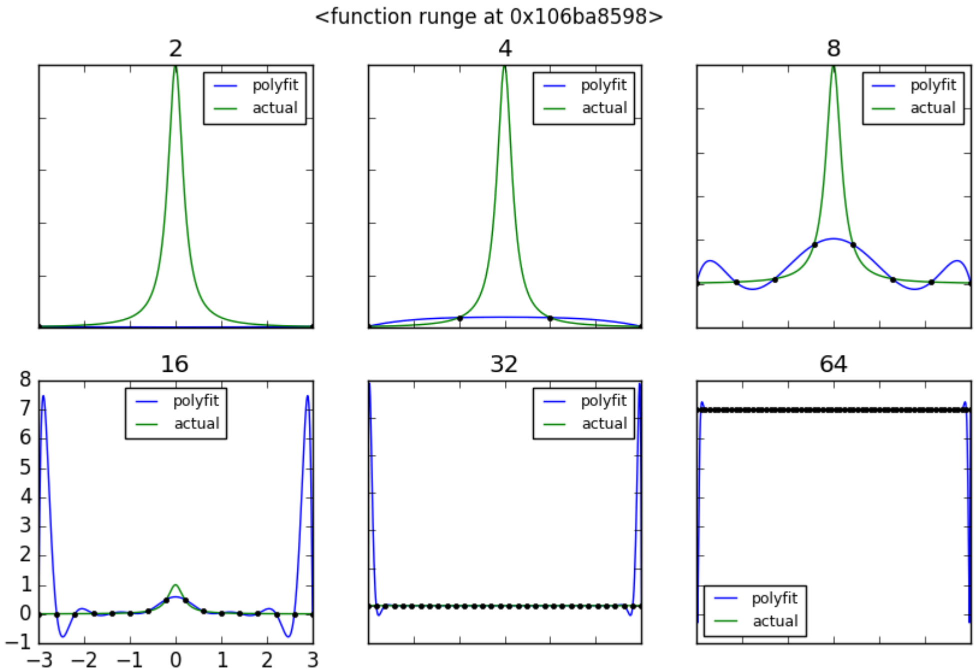


Figure 1: **Uniformly** spaced N-point polynomial interpolations (blue line) of the Runge function $R(x)$ (green line). $N = 2, 4, 8, 16, 32, 64$ (black dots). Plots are from: this link

You can see in Fig. 1, the increase of x_i leads to more catastrophic errors when x_i are uniformly spaced. The solution to this is to utilize a so-called Chebyshev point spacing, defined over $[-1, 1]$ as

$$x_i = \cos\left(\frac{2i-1}{2n}\pi\right). \tag{19}$$

Now let's look at our polynomial interpolation with Chebyshev spaced points.

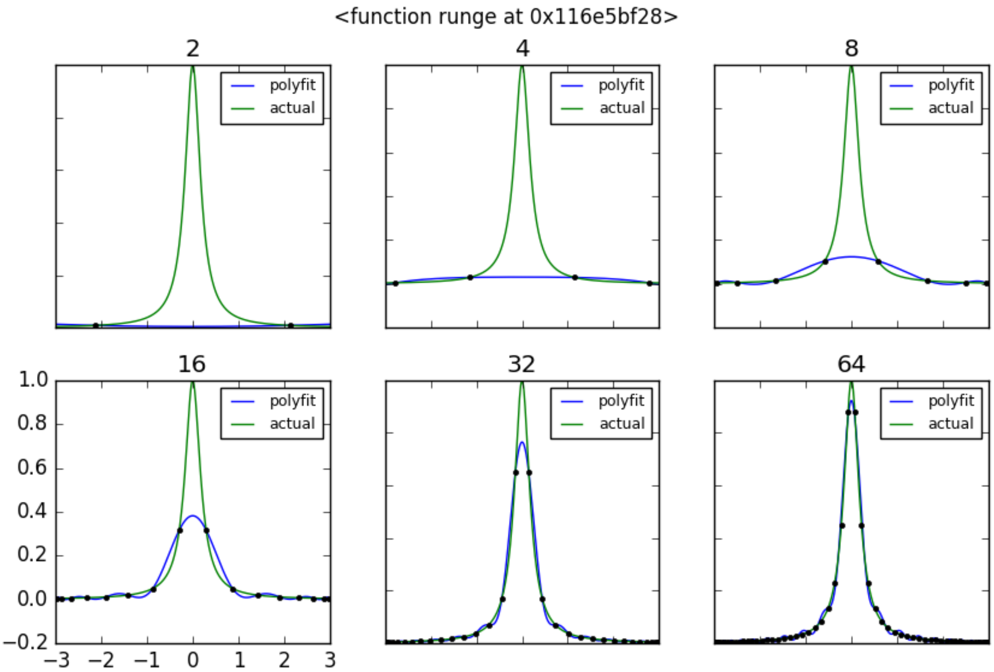


Figure 2: **Chebyshev** spaced N-point polynomial interpolations (blue line) of the Runge function $R(x)$ (green line). $N = 2, 4, 8, 16, 32, 64$ (black dots). Plots are from: this link

In Fig. 2, you can see that now the increase of the number of points x_i improves the accuracy of the interpolation instead of harming it.

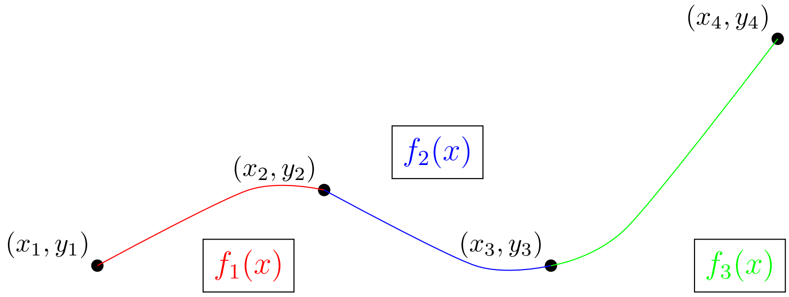
Note: if the function you are attempting to interpolate is not defined over the interval $[-1, 1]$, but it is defined over $[a, b]$ then you can do a linear mapping

$$x_i = a + \frac{b-a}{2}(\hat{x}_i - 1) \tag{20}$$

in which x_i is the point in $[a, b]$, and $\hat{x}_i$ is the Chebyshev point in $[-1, 1]$.

1.3 Piecewise interpolation

We have discussed Lagrange interpolation, but there are other methods of interpolation, one of which is named *piecewise*. Given n data points (which in this context we call *knots*), we can form $n - 1$ interpolation functions $f_1, \ldots, f_{n-1}$ called *splines*:



The interpolation is piecewise in that the function being used to interpolate changes depending on the interval. That is, if $x \in [x_1, x_2]$ we use $f_1(x)$; if $x \in [x_2, x_3]$ we use $f_2(x)$; etc.

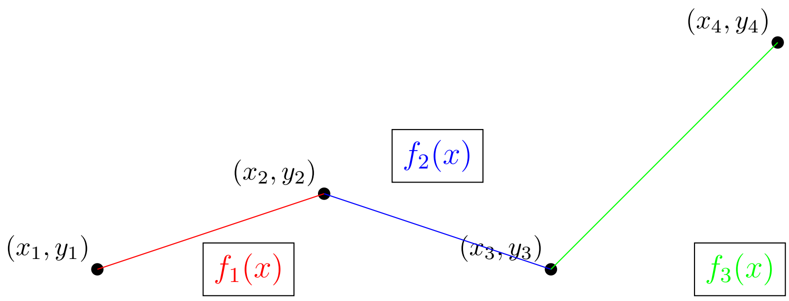
Linear splines are formulated as

$$f_i(x) = a_i + b_1(x - x_i), \tag{21}$$

in which

$$a_i = y_i, \qquad b_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}. \tag{22}$$

Below is an example of a linear spline.



A linear spline is continuous in C^0, which means that the 0th derivative (i.e., the function itself) is itself continuous, but the 1st derivative is discontinuous. In particular, the derivative spikes discontinuously at x_i, y_i. In general, a function is C^n-continuous if its first n derivatives are continuous.

We have just noted that a linear spline (degree 1) is C^0 continuous. Similarly, a cubic spline (degree 3), defined as

$$f_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad x \in [x_i, x_{i+1}] \tag{23}$$

is C^2 continuous. But how do we obtain these coefficients a, b, c, d? To do this we compare the number of unknowns vs. the number of equations, noting that to solve a linear system we need $\#unknowns = \#equations$:

- Unknowns
  - We have 4 coefficients a_i, b_i, c_i, d_i for each of the $i = 1, \ldots, n - 1$ splines. Therefore, we have $4(n - 1)$ *uknowns*.

- Equations
  - Each spline must return the value y at the left point:
$$f_i(x_i) = y_i, \quad i = 1, \dots, n-1 \tag{24}$$
$$\implies a_i = y_i \ \ (n-1 \ equations) \tag{25}$$
  - The last spline must return the value y at the right point:
$$f_{n-1}(x_n) = y_n \quad (1 \ equation) \tag{26}$$
  - Splines must be C^0 at the knots:
$$f_i(x_{i+1}) = f_{i+1}(x_{i+1}) \quad i = 2, \dots, n-1 \tag{27}$$
$$\implies f_i(x_{i+1}) = a_i + b_i(x_{i+1}-x_i) + c_i(x_{i+1}-x_i)^2 + d_i(x_{i+1}-x_i)^3 = a_{i+1} = f_{i+1}(x_{i+1}) \tag{28}$$
$$(n-2 \ equations)$$
  - Splines must be C^1 at the knots:
$$f_i'(x_{i+1}) = f_{i+1}'(x_{i+1}) \quad i = 2, \dots, n-1 \tag{29}$$
$$\implies f_i'(x_{i+1}) = b_i + 2c_i(x_{i+1}-x_i) + 3d_i(x_{i+1}-x_i)^2 = b_{i+1} = f_{i+1}'(x_{i+1}) \tag{30}$$
$$(n-2 \ equations)$$
  - Splines must be C^2 at the knots:
$$f_i''(x_{i+1}) = f_{i+1}''(x_{i+1}) \quad i = 2, \dots, n-1 \tag{31}$$
$$\implies f_i''(x_{i+1}) = 2c_i + 6d_i(x_{i+1}-x_i) = 2c_{i+1} = f_{i+1}''(x_{i+1}) \tag{32}$$
$$(n-2 \ equations)$$

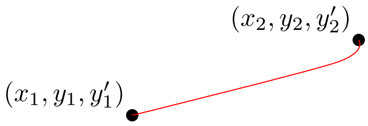
Comparing the number of uknowns to the number of equations so far, we have
$$unknowns: \quad 4(n-1) \tag{33}$$
$$equations: \quad (n-1) + 1 + (n-2) + (n-2) + (n-2) = 4n - 6 = 4(n-1) - 2 \tag{34}$$
so we need to impose 2 more constraints/equations onto the system in order to solve for the unknowns. These will be boundary conditions on the leftmost and rightmost splines. Some options are

- Natural boundary conditions
$$\begin{cases} f_1''(x_1) = 0 \\ f_{n-1}''(x_n) = 0 \end{cases} \tag{35}$$
- Clamped boundary conditions
$$\begin{cases} f_1'(x_1) = constant \\ f_{n-1}'(x_n) = constant. \end{cases} \tag{36}$$

Adding two additional equations results in a $4(n-1) \times 4(n-1)$ linear system with $a_i, b_i, c_i, d_i, \quad i \in [1, n-1]$ as unknowns.

1.4 Hermite interpolation

Hermite interpolation is an extension of piecewise interpolation, using both the function values and their derivatives.



Let

$$t(x) = \frac{x - x_i}{x_{i+1} - x_i} = \frac{x - x_i}{h_i}, \quad t \in [0, 1]. \tag{37}$$

Also let

$$p_i(t) = (2t^3 - 2t^2 + 1)y_i + (t^3 - 2t^2 + t)h_i y_i' + (-2t^3 + 3t^2)y_{i+1} + (t^3 - t^2)h_i y_{i+1}' \tag{38}$$

which obey the conditions

$$p_i(1) = p_{i+1}(0), \tag{39}$$

$$p_i'(1) = p_{i+1}(0). \tag{40}$$

Then the interpolation function and its derivative is

$$f_i(x) = p_i\Big(\frac{x - x_i}{h_i}\Big), \tag{41}$$

$$f_i'(x) = \frac{1}{h_i} p_i'\Big(\frac{x - x_i}{h_i}\Big). \tag{42}$$

1.5 Radial basis interpolation

Radial basis interpolation is used for scattered and possibly overlapping data sets. Here we do interpolation based on so-called *radial basis kernels*, which are functions that depend only on the radial distance between points. Let

$$\phi(r) = \phi(||\mathbf{x} - \mathbf{y}||) \tag{43}$$

be a radial basis kernel. Then, the interpolant is

$$s(\mathbf{x}) = \sum_i^n \omega_i \phi(||\mathbf{x} - \mathbf{x}_i||) \tag{44}$$

in which $\mathbf{x}_i$, $i \in 1, n$ are the n data points and ω_i are called the weights.

1.5.1 Weights

To obtain the weights, we enforce the condition

$$s(\mathbf{x}_i) = f_i \tag{45}$$

such that

$$\omega_1\phi(||\mathbf{x}_1 - \mathbf{x}_1||) + \omega_2\phi(||\mathbf{x}_1 - \mathbf{x}_2||) + \ldots + \omega_n\phi(||\mathbf{x}_1 - \mathbf{x}_n||) = f_1, \tag{46}$$

$$\omega_2\phi(||\mathbf{x}_2 - \mathbf{x}_1||) + \omega_2\phi(||\mathbf{x}_2 - \mathbf{x}_2||) + \ldots + \omega_n\phi(||\mathbf{x}_2 - \mathbf{x}_n||) = f_2, \tag{47}$$

$$\vdots$$

etc. Here, we will have n equations solving for the n unknowns ω_i, $i \in [1, n]$, since the quantities $||\mathbf{x}_i - \mathbf{x}_j||$ are known, and f_i are also known. Also, note that $||\mathbf{x}_i - \mathbf{x}_j|| = ||\mathbf{x}_j - \mathbf{x}_i|| \implies \phi(r_{ij}) = \phi(r_{ji})$, meaning that the system matrix will be symmetric. That is, we will have $\mathbf{\Phi}\boldsymbol{\omega} = \mathbf{f}$, $\mathbf{\Phi}$ symmetric.

1.5.2 Kernels

Some example kernels (among others) are

- Gaussian
$$\phi(r) = e^{(-\epsilon r)^2}, \quad \epsilon \sim 1/h \tag{48}$$
- Multiquadratic
$$\phi(r) = (1 + (\epsilon r)^2)^{1/2} \tag{49}$$
- Inverse multiquadratic
$$\phi(r) = (1 + (\epsilon r)^2)^{-1/2}. \tag{50}$$

Note, radial basis functions do not describe flat data very well, only scattered data. To fix the lack of accuracy we can add corrective polynomials such that

$$s(\mathbf{x}) = \sum_i^n \omega_i\phi(||\mathbf{x} - \mathbf{x}_i||) + p(\mathbf{x}). \tag{51}$$

The polynomials consist of

$$p(\mathbf{x}) = a_{00} + a_{10}x + a_{20}x^2 + a_{01}y + a_{11}xy + a_{02}y^2 \tag{52}$$

with the constraint

$$p(\omega_i) = 0. \tag{53}$$

Then we can assemble the linear system

$$\begin{bmatrix} \mathbf{\Phi} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{\omega} \\ \mathbf{a} \end{Bmatrix} = \begin{Bmatrix} \mathbf{f} \\ \mathbf{0} \end{Bmatrix}. \tag{54}$$

to solve for the weights ω_i as well as the coefficients a_{00}, a_{10}, etc.