CHAPTER 6 INTERPOLATION AND EXTRAPOLATION

SOLVED EXAMPLE PROBLEMS

for

NUMERICAL METHODS FOR SCIENTISTS AND ENGINEERS With Pseudocodes

By Zekeriya ALTAÇ

October 2024

EXAMPLE 6.1: Generating Lagrange polynomials for a given dataset

Find the Lagrange basis polynomials–$L_0(x)$, $L_1(x)$, $L_2(x)$, and $L_3(x)$–using the nonuniform spaced dataset given below. Plot and inspect the distributions of the polynomials in [2,7].

k	0	1	2	3
x_k	2	3	6	7

SOLUTION:

The given data set consists of four data pairs; thus, the degree of the Lagrange polynomials to be constructed can be maximum three; i.e., $n = 4 - 1 = 3$. Employing Eq. (6.8) to the given dataset, we obtain

$$\begin{aligned}
L_0(x) &= \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} = \frac{(x-3)(x-6)(x-7)}{-20}, \\
L_1(x) &= \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} = \frac{(x-2)(x-6)(x-7)}{12}, \\
L_2(x) &= \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} = \frac{(x-2)(x-3)(x-7)}{-12}, \\
L_3(x) &= \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} = \frac{(x-2)(x-3)(x-6)}{20}
\end{aligned}$$

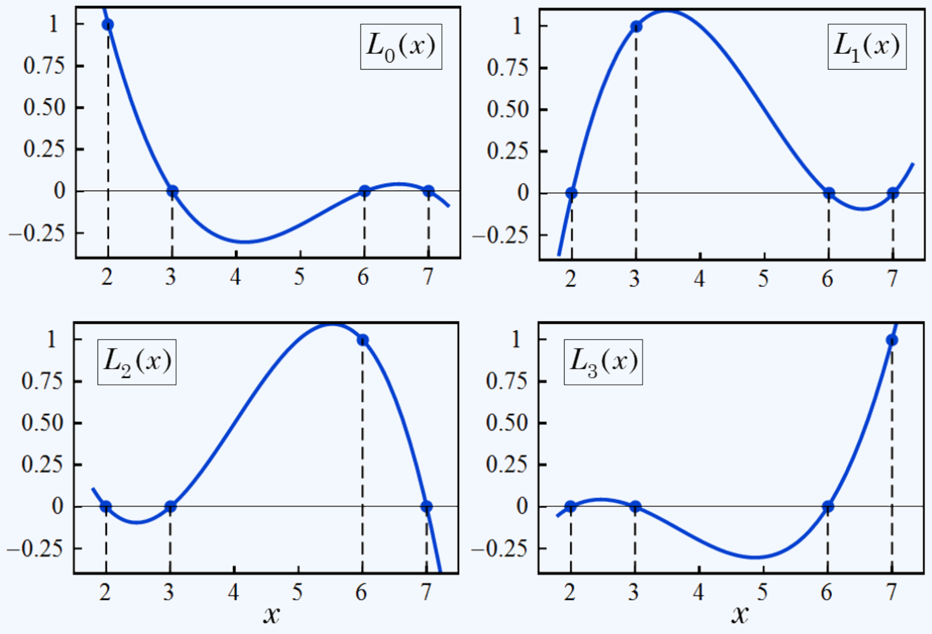


Figure 6.1: Lagrange polynomials for $L_0(x)$, $L_1(x)$, $L_2(x)$, and $L_3(x)$ on [2,7].

Discussion: The basis polynomials are unique for a given set of interpolation points. No two sets of Lagrange basis polynomials will be the same for different sets of points. Each Lagrange basis polynomial, $L_k(x)$, is designed so that $L_i(x_j) = 1$ if $i = j$ and $L_i(x_j) = 0$ if $i \neq j$. Thus, for a given set of $n+1$ data points, a Lagrange basis polynomial for n points will be zero and 1 for only one point.

EXAMPLE 6.2: Constructing Lagrange interpolating polynomial and error assessment

A dataset derived from $y(x) = -2x^2+(x^4+9)/5$ is tabulated below. Find the Lagrange basis polynomials $L_0(x)$, $L_1(x)$, $L_2(x)$, and $L_3(x)$ for the given dataset. (a) Using the Lagrange interpolation polynomial that passes through all data points, evaluate $y(3)$ and $y(4.7)$. (b) Investigate the absolute error of the interpolating polynomial in the interval [1,5].

x_k	1	2	4	5
y_k	0	-3	21	76.8

SOLUTION:

This dataset consists of four data pair, and the degree of the Lagrange basis polynomials can be maximum 3; i.e., $n = 4 - 1 = 3$. Employing Eq. (6.8) to the given dataset, we obtain

$$L_0(x) = \frac{(x-2)(x-4)(x-5)}{(1-2)(1-4)(1-5)}, \qquad L_1(x) = \frac{(x-1)(x-4)(x-5)}{(2-1)(2-4)(2-5)},$$
$$L_2(x) = \frac{(x-1)(x-2)(x-5)}{(4-1)(4-2)(4-5)}, \qquad L_3(x) = \frac{(x-1)(x-2)(x-4)}{(5-1)(5-2)(5-4)},$$

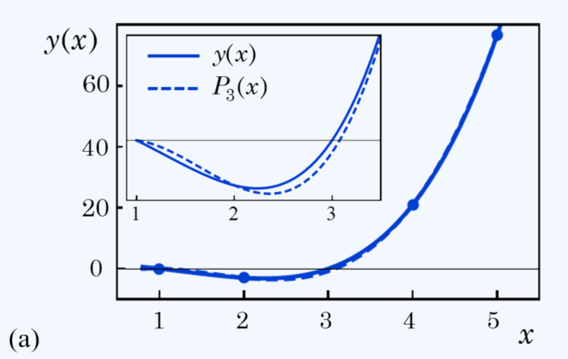
Using Eq. (6.4), the third degree Lagrange interpolation polynomial is constructed as follows:

$$y(x) \cong P_3(x) = (0)L_0(x) + (-3)L_1(x) + (21)L_2(x) + (76.8)L_3(x)$$

Expanding and simplifying Lagrange polynomials and interpolation function, we obtain

$$y(x) \cong P_3(x) = 2.4x^3 - 11.8x^2 + 15.6x - 6.2$$

Using this interpolation polynomial, we find $y(3) \cong p_3(3)$= -0.8 and $y(4.7) \cong p_3(4.7)$= 55.6332, while the true values are y(0)=0 and y(4.7)=55.2136.



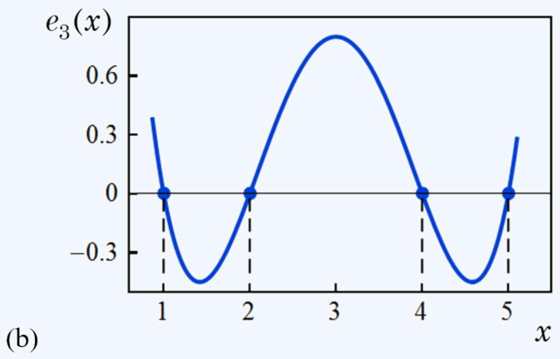


Figure 6.2: (a) The agreement of the true distribution with the Lagrange interpolation polynomial constructed, (b) the true absolute error distribution due to the interpolating polynomial.

Discussion: In **Figure 6.2**, the true function and the interpolating polynomial are depicted in the interval $1 \leqslant x \leqslant 5$. The error due to interpolation with Lagrange polynomials is observed to vary with the independent variable x throughout the interval. Also notice that the true function and the interpolating polynomial depict overall very good agreement (**Figure 6.2a**). As shown in **Figure 6.2b**, the true absolute error, $e_3(x)$, for the data points are zero due to imposing Eq. (6.2) condition. Furthermore, the errors near the data points are small, but $|e_3(x)|$ increase towards the middle of the sub-intervals.

EXAMPLE 6.3: Constructing Newton's interpolating polynomial

Use the following dataset to construct an interpolating polynomial which uses all data.

x_k	-1	0	0.5	1.5	3
y_k	3	1	2.625	7.375	7

SOLUTION:

The table of divided differences is given in **Table 6.1**. Notice that the 3rd degree divided differences (5th column) is "-1" and the last column (i.e., 4th degree divided difference) is zero. This means that the dataset is truly represented by a 3rd degree polynomial.

When the interpolating polynomial is constructed using Eq. (6.19), we find

$$y(x) = 3 + (x+1)\left(-2 + (x-0)\left\{3.5 + (x-0.5)\left(-1+0\right)\right\}\right) = -x^3 + 3x^2 + 2x + 1$$

This polynomial gave the "true function". Note that the interpolating polynomial is third degree polynomial even though the Newton's polynomial was supposed to be 4th degree.

Table 6.1: Divided difference table for Example 6.3

x_i	$f[x_i]$	$f[x_i, x_{i+1}]$	$f[x_i, x_{i+1}, x_{i+2}]$	$f[x_i, \ldots, x_{i+3}]$	$f[x_i, \ldots, x_{i+4}]$
-1	3	-2.0	3.5	-1	0
0	1	3.25	1	-1	
0.5	2.625	4.75	-2		
1.5	7.375	-0.25			
3	7				

Discussion: Unlike this example, when dealing with experimental data, the true distribution is almost always unknown. Thus, the error estimate of the interpolating polynomial is assessed by Eq. (6.22).

EXAMPLE 6.4: Assessing the accuracy of Newton's interpolating polynomials

The dataset presented below was generated from $y(x) = (2x^2 + 1)\sin x$. Use the method of Newton's divided differences to develop 2nd, 3rd, and 4th degree interpolation polynomials and remark on the distribution of observed true absolute error across the data range.

x_i	0.61	0.995	1.18	1.8	2
y_i	1	2.5	3.5	7.28	8.2

SOLUTION:

The Newton's divided difference table is generated and presented in **Table 6.2**. The degree of the interpolating polynomial that passes through all points can be maximum, $n = 5 - 1 = 4$.

Table 6.2

x_i	$f[x_i]$	$f[x_i, x_{i+1}]$	$f[x_i, x_{i+1}, x_{i+2}]$	$f[x_i, \ldots, x_{i+3}]$	$f[x_i, \ldots, x_{i+4}]$
0.61	1	3.896104	2.647897	−1.503407	−0.839868
0.995	2.5	5.405405	0.858843	−2.670823	
1.18	3.5	6.096774	−1.825334		
1.8	7.28	4.600000			
2	8.2				

The Newton's interpolating polynomials can be obtained recursively as follows:

$$\begin{aligned}
p_2(x) &= 1 + 3.8961\,(x - 0.61) + 2.6479\,(x - 0.995)\,(x - 0.61)\,,\\
p_3(x) &= p_2(x) - 1.50341\,(x - 1.18)\,(x - 0.995)\,(x - 0.61)\,,\\
p_4(x) &= p_3(x) - 0.83987\,(x - 1.8)\,(x - 1.18)\,(x - 0.995)\,(x - 0.61)
\end{aligned}$$

Note that the $p_2(x)$ and $p_3(x)$ polynomials require less data points to construct 2nd and 3rd degree interpolating polynomials.

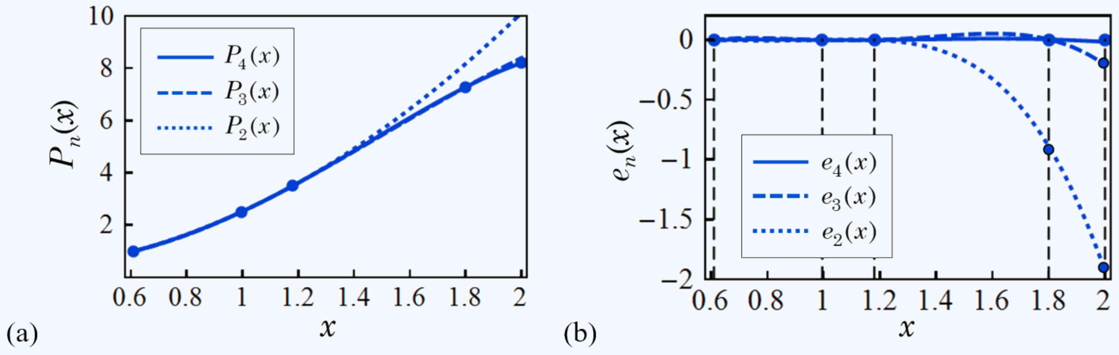


Figure 6.3: (a) Polynomial approximations and (b) true absolute error distribution.

Discussion: In **Figure 6.3**, the interpolating polynomials, $p_2(x)$, $p_3(x)$ and $p_4(x)$ and associated true absolute errors in the interval [0.6, 2] are comparatively depicted. The preset condition, which Newton's divided difference polynomials should satisfy the dataset, restrains the magnitude of the error between two points. Since the magnitude of the successive divided differences in the first row decrease, the influence of the higher divided differences become relatively small, leading the predictions to converge quicker with the increasing degree of interpolation polynomial. Notice that $p_2(2)$ and $p_3(1.8)$ do not satisfy the original data points. That is because when constructing these polynomials, the data points were not incorporated to the interpolating polynomials. Therefore, as more data points are excluded when generating the interpolating polynomials, larger deviations at the right end of the interval are observed (**Figure 6.3b**), while $p_2(x)$ and $p_3(x)$ depict a good agreement on the left side of the interval.

EXAMPLE 6.5: Cubic Spline Interpolation with two end-conditions

Using the dataset given below, estimate $f(x)$ for $x = 0.2$, 0.6, 1, 1.4, and 1.8 using the cubic splines that employ (a) natural spline and (b) linear extrapolation end conditions. Compare your estimates with the true values, given by $y(x) = (25x^4 + x^2)/4$.

x_k	0	0.4	0.8	1.2	1.6	2
f_k	0	0.2	2.72	13.32	41.6	101

SOLUTION:

(a) For the natural spline end condition, we set $S_1 = S_6 = 0$. Then Eq. (6.62) takes the following form:

$$\begin{bmatrix} 1.6 & 0.4 & & \\ 0.4 & 1.6 & 0.4 & \\ & 0.4 & 1.6 & 0.4 \\ & & 0.4 & 1.6 \end{bmatrix} \begin{bmatrix} S_2 \\ S_3 \\ S_4 \\ S_5 \end{bmatrix} = \begin{bmatrix} 34.8 \\ 121.2 \\ 265.2 \\ 466.8 \end{bmatrix}$$

When the tridiagonal system of equations is solved with Thomas algorithm we obtain $S_1 = 0$, $S_2 = 8.66986$, $S_3 = 52.3206$, $S_4 = 85.0478$, $S_5 = 270.488$ and $S_6 = 0$. Using Eq. (6.61), the cubic splines (i.e, coefficients) corresponding each interval are determined as follows:

$$\begin{aligned}
y_1(x) &= 3.612442(x-0)^3 + 0\,(x-0)^2 - 0.077991(x-0) - 0, && [0,\ 0.4]\\
y_2(x) &= 18.18779(x-0.4)^3 + 4.334931(x-0.4)^2 + 1.655981(x-0.4) + 0.2, && [0.4,\ 0.8]\\
y_3(x) &= 13.63638(x-0.8)^3 + 26.16028\,(x-0.8)^2 + 13.85406(x-0.8) + 2.72, && [0.8,\ 1.2]\\
y_4(x) &= 77.26675(x-1.2)^3 + 42.52394\,(x-1.2)^2 + 41.32775(x-1.2) + 13.32, && [1.2,\ 1.6]\\
y_5(x) &= -112.7033(x-1.6)^3 + 135.244(x-1.6)^2 + 112.43493(x-1.6) + 41.6, && [1.6, 2]
\end{aligned}$$

(b) For linear extrapolation end condition requirement Eq. (6.60) leads to

$$\begin{bmatrix} 2.4 & 0 & & \\ 0.4 & 1.6 & 0.4 & \\ & 0.4 & 1.6 & 0.4 \\ & & 0 & 2.4 \end{bmatrix} \begin{bmatrix} S_2 \\ S_3 \\ S_4 \\ S_5 \end{bmatrix} = \begin{bmatrix} 34.8 \\ 121.2 \\ 265.2 \\ 466.8 \end{bmatrix}$$

The system solution yields $S_2 = 14.5 S_3 = 45.7$, $S_4 = 105.7$, and $S_5 = 194.5$. Next we use the linear extrapolation end conditions to determine $S_1 = 2S_2 - S_3 = -16.7$ and $S_6 = 2S_5 - S_4 = 283.3$. Using Eq. (6.61), the cubic splines (i.e, coefficients) corresponding each interval are determined as follows:

$$\begin{aligned}
y_1(x) &= 13(x-0)^3 - 8.350\,(x-0)^2 + 1.760(x-0) - 0, && [0,\ 0.4]\\
y_2(x) &= 13(x-0.4)^3 + 7.250(x-0.4)^2 + 1.320(x-0.4) + 0.2, && [0.4,\ 0.8]\\
y_3(x) &= 25(x-0.8)^3 + 22.85\,(x-0.8)^2 + 13.36(x-0.8) + 2.72, && [0.8,\ 1.2]\\
y_4(x) &= 37(x-1.2)^3 + 52.85\,(x-1.2)^2 + 43.64(x-1.2) + 13.32, && [1.2,\ 1.6]\\
y_5(x) &= 37(x-1.6)^3 + 97.25(x-1.6)^2 + 103.68(x-1.6) + 41.6, && [1.6, 2]
\end{aligned}$$

Th piece-wise cubic splines generated with both end conditions are used as interpolating polynomials. For example, the interpolating polynomial corresponding to $x = 0.2$ is the first polynomial, i.e., $y_1(x)$. Thus, the interpolated value is computed as $y(0.2) = y_1(0.2)$, $y(0.6) = y_2(0.6)$, $y(1.8) = y_5(1.8)$, and so on.

The results obtained with the interpolated values with both end conditions are comparatively presented in **Table 6.3**. The comparison of the results depicts linear extrapolation end condition case performing better.

Table 6.3: Comparison of the cubic spline estimates with natural and linear end conditions.

x	True Value	Natural Spline EC		Linear Extrapolation EC	
0.2	0.02	0.0133	33.5%	0.122	−510%
0.6	0.90	0.8501	5.54%	0.858	4.67%
1.0	6.50	6.6463	−2.25%	6.506	−0.09%
1.4	24.50	23.9046	2.43%	24.458	0.17%
1.8	66.42	68.5951	−3.27%	66.522	−0.15%

Discussion: These splines are piecewise cubic polynomials, and their behavior is influenced by the choice of end conditions. There are several other end conditions employed in cubic spline interpolations besides those covered in the textbook. The accuracy of the cubic splines near the end points depends on how suitable the end conditions are to the dataset.

The *natural spline* end conditions require zero second derivatives at the endpoints. In other words, when using this condition, the second derivative of the spline function is assumed to be zero or negligible at both the start and end points. This condition gives a spline that is "flatter" at the endpoints compared to other conditions, making it behave more like a straight line near the boundaries. This end condition may be useful when you do not desire the spline to exhibit excessive curvature beyond the data points. However, it is less suitable if the actual constraints are known, as it assumes no specific behavior at the endpoints.

The *linear extrapolation* end conditions extends the spline beyond the data points using a linear extrapolation. The line is determined by the last segment of the cubic spline (i.e., the segment connecting the last two data points) and its slope. This approach combines the smoothness of cubic splines with the simplicity of linear extrapolation, providing a method to closely estimate the curvature and the actual data distribution.

The *clamped (fixed) spline* end conditions, though not presented in the text, require the first derivatives (slopes) at the endpoints; i.e., $f'(x_1) = m_1$ and $f'(x_n) = m_n$, where m_1 and m_n are the specified slopes at x_1 and x_n, respectively. This end condition can better capture the distribution of the data if one has information about how the spline should behave at the boundaries. Thus, this alternative provides a control over the slope of the spline at the boundaries, which can be particularly useful if the slopes at the endpoints are roughly known.

The *not-a-knot spline* end conditions, also not presented in the text, assumes that the third derivative of the spline is continuous at the second and second-to-last points. In other words, the spline is constructed such that the third derivatives are continuous across the interior knots. This end condition ensures smoothness in the spline and avoids sudden changes in curvature at the knots. It can be advantageous in cases where a smooth transition through the data points is more important than controlling the specific behavior at the endpoints.

In this example, since we know the true function, we find $f''(x) = 75x^2 + 0.5$. Overall, better results are obtained with the linear extrapolation end condition. Choosing between these conditions depends on your specific needs regarding boundary behavior and smoothness requirements.

EXAMPLE 6.6: Root finding of Discrete functions

Estimate the root of the function given by the discrete data set below.

x_k	1	1.25	1.5	2
f_k	0.3863	0.0581	−0.2952	−1.0541

SOLUTION:

The data set rearranged by interchanging the abscissas and the ordinates as follows:

f_k	0.3863	0.0581	−0.2952	−1.0541
x_k	1	1.25	1.5	2

Then the problem is reduced to finding an interpolation for $f = 0$. The Lagrange interpolating polynomial that uses all data points is a third degree polynomial which may be expressed as

$$x \cong P_3(f) = (1.0)L_0(f) + (1.25)L_1(f) + (1.5)L_2(f) + (2.0)L_3(f)$$

As the Lagrange polynomials for the data set are generated and the interpolating polynomial is evaluated for $f = 0$, we get

$$\begin{aligned} x = P_3(0) &= (1.0)(-0.056116) + (1.25)(0.93209) + (1.5)(0.129476) \\ &+ (2.0)(-0.00545) = 1.29231 \end{aligned}$$

Discussion: We may also investigate the effect of the lower order interpolating polynomials on the estimates. To do that we need three data point for a quadratic interpolation polynomial whereas we require four data points in the given set. When f_1, f_2, f_3 and f_0, f_1, f_2 data sets are used, the estimates for the root are found as 1.29186 and 1.29247, respectively. On the other hand, a simple linear interpolation gives 1.29111. All of the estimates yield the values with the same two-decimal places.

EXAMPLE 6.7: Application of Double Lagrange Interpolation

Given below is a 2D discrete data for a two-variable function $z = f(x,y)$. Use double Lagrange interpolation to compute $f(0.48, 0.33)$ and $f(0.08, 0.57)$. Note: The true solutions are given as $f(0.48, 0.33) = 4.04757$ and $f(0.08, 0.57) = 0.359635$.

	y		
x	0.2	0.4	0.6
0	0.0256	0.4096	2.0736
0.1	0	0.0168	0.2999
0.3	0.547	0.0731	0.0011
0.5	8.3521	3.8416	1.4641

SOLUTION:

In this example, we have $M-1=3$ and $N-1=2$. We construct the Lagrange polynomials for x's ($L_0(x)$, $L_1(x)$, $L_2(x)$ and $L_4(x)$) and y's ($L_0(y)$, $L_1(y)$, $L_2(y)$) as follows:

$$\begin{aligned}
L_0(x) &= \frac{(x-0.1)(x-0.3)(x-0.5)}{(0-0.1)(0-0.3)(0-0.5)},\\
L_1(x) &= \frac{(x-0)(x-0.3)(x-0.5)}{(0.1-0)(0.1-0.3)(0.1-0.5)},\\
L_2(x) &= \frac{(x-0)(x-0.1)(x-0.5)}{(0.3-0)(0.3-0.1)(0.3-0.5)},\\
L_3(x) &= \frac{(x-0)(x-0.1)(x-0.3)}{(0.5-0)(0.5-0.1)(0.5-0.3)},\\
L_0(y) &= \frac{(y-0.4)(y-0.6)}{(0.2-0.4)(0.2-0.6)},\\
L_1(y) &= \frac{(y-0.2)(y-0.6)}{(0.4-0.2)(0.4-0.6)},\\
L_2(y) &= \frac{(y-0.2)(y-0.4)}{(0.6-0.2)(0.6-0.4)}
\end{aligned}$$

Function values are sub-scripted as $f(x_n, y_m) = f_{nm}$; for example, $f_{1,1} = 0.0256$, $f_{2,1} = 0$, $f_{3,1} = 0.547$, etc. Lastly the interpolating polynomial using Eq. (6.64) becomes

$$f(x,y) \cong P_{m,n}(x,y) = \sum_{n=0}^{2} L_n(y) \left(\sum_{m=0}^{3} L_m(x) f_{m,n} \right)$$

Using this interpolating polynomial, we get $f(0.48, 0.33) = 4.32456$ (absolute error=$-$%27.7) and $f(0.08, 0.57) = 0.45244$ (absolute error=$-$%9.28).

Discussion: The bi-Lagrange interpolation is a simple extension of linear interpolation to two dimensions. Its accuracy and effectiveness depend mostly on the relationship of the underlying function and the resolution of the grid. The accuracy of bilinear or biLagranfe interpolation improves with a refined grid. A higher resolution grid provides points closer to the interpolation point, reducing interpolation error. While bilinear or bicubic interpolation is often used in practice due to its simplicity, more advanced methods like bicubic interpolation or spline-based methods can provide better accuracy for distributions with rapid changes.