Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 9020541, 15 pages
https://doi.org/10.1155/2020/9020541

Research Article

A New Approach to Newton-Type Polynomial Interpolation with Parameters

Le Zou,[1,2,3] **Liangtu Song**,[2,3] **Xiaofeng Wang**,[1] **Thomas Weise**,[4] **Yanping Chen**,[1] **and Chen Zhang**[1]

[1]*School of Artificial Intelligence and Big Data, Hefei University, Hefei 230601, Anhui, China*
[2]*Hefei Institutes of Physical Science, Chinese Academy of Sciences, Hefei 230031, China*
[3]*University of Science and Technology of China, Hefei 230027, China*
[4]*Institute of Applied Optimization, School of Artificial Intelligence and Big Data, Hefei University, Hefei 230601, Anhui, China*

Correspondence should be addressed to Xiaofeng Wang; xfwang@hfuu.edu.cn

Received 9 June 2020; Revised 28 September 2020; Accepted 10 October 2020; Published 17 November 2020

Academic Editor: Massimiliano Ferrara

Newton's interpolation is a classical polynomial interpolation approach and plays a significant role in numerical analysis and image processing. The interpolation function of most classical approaches is unique to the given data. In this paper, univariate and bivariate parameterized Newton-type polynomial interpolation methods are introduced. In order to express the divided differences tables neatly, the multiplicity of the points can be adjusted by introducing new parameters. Our new polynomial interpolation can be constructed only based on divided differences with one or multiple parameters which satisfy the interpolation conditions. We discuss the interpolation algorithm, theorem, dual interpolation, and information matrix algorithm. Since the proposed novel interpolation functions are parametric, they are not unique to the interpolation data. Therefore, its value in the interpolant region can be adjusted under unaltered interpolant data through the parameter values. Our parameterized Newton-type polynomial interpolating functions have a simple and explicit mathematical representation, and the proposed algorithms are simple and easy to calculate. Various numerical examples are given to demonstrate the efficiency of our method.

1. Introduction

The interpolation problem has been the subject of many classic studies in approximation theory [1–5]. In the last several years, many researchers have been focusing on this subject and have obtained interesting results. The existing approaches can be divided into polynomial and rational interpolation methods, both of which are applicable to numerical approximation [1, 2], image interpolation [3–6], and arc structuring and surface modeling [7–10]. Newton's polynomial interpolation and Thiele's interpolating continued fractions can be incorporated to generate various interpolation schemes based on rectangular grids. As one can see, the (partial) inverse differences and the (partial) divided differences play a great role on the study of the polynomial and rational interpolation. Many scholars have constructed a variety of blending rational interpolation. Vonza [11] developed rational interpolation-based associated continued fractions. Varsamis and Karampetakis [12] presented recursive algorithms for the Newton polynomial interpolation of a given bivariate function. Tan et al. constructed Newton–Thiele [13] and Thiele–Newton blending rational interpolation [1]. However, blending rational interpolants strongly depend on the existence of so-called blending differences. By using the method of divide and conquer, Tan and Tang [14] proposed composite interpolation over triangular subgrids. By dividing the original set of support points into some blocks, Tan and Zhao proposed the block-based Thiele-like blending rational interpolation [15] and the block-based Newton-type blending rational interpolation [16]. Tang and Liang [17] developed the bivariate blending Thiele–Werner osculatory rational interpolation. Based on Thiele interpolating continued fraction, polynomial interpolation, and barycentric rational interpolation,

many types of blending rational interpolation were studied. In recent years, Dyn and Floater [18] studied multivariate polynomial interpolation based on lower subsets. Cuyt and Salazar Celis [19] developed a generalized multivariate data fitting model to solve a variety of scientific computing problems, such as graphics, filtering, metamodeling, computational finance, and more, which captured linear as well as nonlinear phenomena. Interpolation was also applied to graphic image morphing and image processing [1, 3–6, 20–23]. He et al. [3, 20] studied Thiele–Newton rational interpolation in the polar coordinates, applied it to image super-resolution, and obtained better performance. Karim and Saaban [21] proposed a novel rational bicubic ball function with one parameter by using tensor product approach and used it for image upscaling. Zhan et al. [22] developed a nonlocal and local image interpolation model based on nonlocal bounded variation regularization and local total variation and obtained better performance. He et al. [24] proposed an image inpainting algorithm by using continued fractions rational interpolation. In order to obtain better repaired results, Thiele's rational interpolation was combined with Newton–Thiele rational interpolation to repair damaged images. In [25], the Newton interpolation method was adopted in capacitance gradient and the calibration of cantilever stiffness. Based on polynomial approximation of local surface, Rahman et al. [5] presented a novel local patch descriptor which is invariant to the changes in viewpoint, scale, orientation, and illumination.

The mathematical description and the construction method of the curve and surfaces are important problems in computer-aided geometric design [7–9, 26, 27]. There are many ways to handle this problem [1, 7–9, 28, 29], including Bézier and NURBS-based approaches as well as the polynomial spline method. These methods are used, for instance, for shaping the outer hull of aircraft and a ship [28]. However, the disadvantage of polynomial interpolation lies in its global property, that is, it is difficult to control the local constraint of a given interpolation point under the condition that the values of the interpolant points are fixed [29, 30]. To construct the polynomial spline, the derivative values of the interpolating data are required additionally to the function values. Unfortunately, in many practical fields, it is difficult to obtain these derivative values.

In recent years, many scholars have studied parametric spline interpolation. Zhang et al. [29, 30] proposed new types of weighted blending spline interpolation. By selecting appropriate parameters and different coefficients, the value of the spline interpolation function can be modified at any point in the interpolant interval, under the condition that the values of the interpolant points are fixed, so that the geometric surfaces can be adjusted. The drawback is that the computation is complicated. The bivariate rational interpolation with parameters, based only on the function values, has been studied in [29]. Based on function values and partial derivatives of the function being interpolated, Duan et al. [31] proposed a new bivariate rational interpolation, which had a simple and explicit rational mathematical representation with parameters. Based on function values, Duan et al. [32] constructed a rational cubic spline with parameters. This spline is C1-continuous in the interpolating interval. By selecting suitable parameters, it can also become C2-continuous. Reddy et al. [33] presented univariate and bivariate rational cubic fractal interpolation functions with shape parameters, which can be used for surface modeling when the partial derivatives of the given function are irregular. In [34], the authors presented a weighted bivariate rational bicubic spline interpolation based on function values, which is C1-continuous for any positive parameters. The shape of the interpolating surface can be modified by changing the parameters under the condition that the values of the interpolating nodes are fixed, which is convenient in engineering and useful in computer-aided geometric design [28–34]. Tang and Zou [35] studied and presented general interpolant frames with many classical interpolation formulas. Univariate and bivariate Thiele-like rational interpolation continued fractions with parameters were studied in [36]. There, the interpolation function can deal with interpolation problems where inverse differences do not exist. Through the selection of proper parameter values, the value of the proposed interpolation function can be changed at any point in the interpolant region under the condition that the values of the interpolating nodes are fixed. However, how to select appropriate parameters is a hard problem. It is also difficult to define such an interpolant function. Polynomials have many advantages, such as a simple structure, easy differentiation, integration, and having derivatives of any order. The polynomial function is a good tool for approximating smooth function. In computer-aided geometric design, there is still a great demand for complicated models and the integration of design and fabrication, but there are still two unresolved problems [29, 30]:

(1) How can one construct a suitable polynomial interpolation function with an explicit mathematical representation, simple calculation, which is also convenient to use and to study theoretically?

(2) For given data, how can one remodel the curves or surfaces shape to meet the actual requirements in the computer-aided geometric design?

Given $n+1$ points $(x_0, x_1, \ldots, x_n \in R)$ and $n+1$ values $(y_0, y_1, \ldots, y_n \in R)$, in the traditional interpolation method, the classical interpolating polynomial $P_n(x)$ of degree less or equal to n is unique to the given interpolation points, so it cannot be an answer to the above questions. At present, Newton's polynomial interpolation is in the center of research on polynomial interpolation methods. While it has shown good interpolation performance, its disadvantage is that it is unique to the given interpolation points [28]. To overcome the above shortcomings, we propose a novel Newton interpolation polynomial only based on (partial) divided differences by introducing one or multiple parameters, which can be seen as a new approach to interpolation method of Newton polynomials. Similar to the interpolation functions in [28–32], the proposed method

allows for modifying the curve or surface shape under the condition that the values of the interpolant points are fixed. At the same time, the proposed method has many advantages, such as a simple and explicit mathematical representation and better performance. The interpolant function only depends on the values of the function being interpolated and the (partial) divided differences, so the computation is simple.

The rest of the paper is organized as follows. First, we develop one univariate parameterized Newton-type polynomial interpolation and give the interpolation algorithms and theorems in Section 2. Then, we present one kind of bivariate parameterized Newton-type polynomial interpolation and give the interpolation algorithms, theorems, and the dual interpolation polynomial in Section 3. In Section 4, to illustrate the effectiveness of the proposed algorithms, numerical examples are given.

2. Parameterized Univariate Newton-Type Polynomial Interpolation

We now develop the univariate Newton-type polynomial interpolation problems in one variable. Suppose function $y = f(x)$ and we are given the support points set $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$ on interval $[a, b]$. We obtain the following classic Newton polynomial representation [1]:

$$P_n(x) = f(x_0) + f[x_0, x_1](x - x_0) + \cdots + f[x_0, x_1, \ldots, x_n](x - x_0)(x - x_1), \ldots, (x - x_{n-1}), \tag{1}$$

where $f(x_0), f[x_0, x_1], \ldots, f[x_0, x_1, \ldots, x_n]$ represent the divided differences of $f(x)$.

If the interpolation polynomial function exists for the given $n+1$ interpolation data points, then classical interpolating polynomial by $P_n(x)$ of degree less or equal to n is unique. Zhao and Tan [16] developed the block-based Newton-type blending rational interpolation. For a given set of interpolant data, they divided the interpolation into many subsets (blocks); based on different block point data, many types of polynomial or rational interpolation can be constructed. The block-based divided difference is calculated based on the block interpolation points first. Then, the block-based Newton-type blending rational interpolation over all blocks is constructed. This requires a lot of calculation, which is inconvenient for interpolation application. Hence, we construct a novel Newton interpolation polynomial with a single parameter based on one virtual point. The point is adjusted strategically, and the univariate Newton-type polynomial interpolation with one parameter is constructed.

By introducing a new parameter λ, an arbitrary point of the original points $(x_k, y_k)(k = 0, 1, \ldots, n)$ is regarded as a virtual double point, while the multiplicity of the other points remains the same.

Let

$$y_i^0 = y_i, \quad i = 0, 1, \ldots, n. \tag{2}$$

When $(j = 1, \ldots, k+1)$, for $(i = j, j+1, \ldots, n)$,

$$y_i^j = \frac{y_i^{j-1} - y_{j-1}^{j-1}}{x_i - x_{j-1}}. \tag{3}$$

For $(i = k+1, k+2, \ldots, n)$,

$$z_i^{k+1} = \frac{y_i^{k+1} - \lambda}{x_i - x_k}. \tag{4}$$

When $(j = k+2, k+3, \ldots, n)$, for $(i = j, j+1, \ldots, n)$,

$$z_i^j = \frac{z_i^{j-1} - z_{j-1}^{j-1}}{x_i - x_{j-1}}. \tag{5}$$

Based on equations (2)–(5), the novel divided differences are given as shown in Table 1.

Compared with the classical Newton polynomial interpolation, it is easy to see that the method in this paper is simple and convenient and is the same as that of the classical Newton polynomial interpolation.

The Newton-type polynomial interpolation with a parameter λ can be constructed:

$$\begin{aligned} P_n^{(0)}(x) = {} & c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots \\ & + c_k(x - x_0)(x - x_1)\cdots(x - x_{k-1}) \\ & + c_{k+1}^0(x - x_0)(x - x_1)\cdots(x - x_{k-1})(x - x_k) \\ & + c_{k+1}(x - x_0)(x - x_1)\cdots(x - x_{k-1})(x - x_k)^2 \\ & + c_{k+2}(x - x_0)(x - x_1)\cdots(x - x_{k-1}) \\ & \cdot (x - x_k)^2(x - x_{k+1}) \\ & + \cdots + c_n(x - x_0)(x - x_1)\cdots(x - x_{k-1}) \\ & \cdot (x - x_k)^2(x - x_{k+1})\cdots(x - x_{n-1}), \end{aligned} \tag{6}$$

where

$$c_i = \begin{cases} y_i^i, & i = 0, 1, \ldots, k, \\ z_i^i, & i = k+1, k+2, \ldots, n, \end{cases} \qquad c_{k+1}^0 = \lambda. \tag{7}$$

Theorem 1. *Given the interpolation data* $(\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\})$,

$$P_n^{(0)}(x_i) = y_i, \quad i = 0, 1, \ldots, n. \tag{8}$$

Proof. Suppose $(0 \le i \le k)$.

Equation (6) is the classical Newton polynomial interpolation, so $P_n^{(0)}(x_i) = y_i$, $(i = 0, 1, \ldots, k)$. For $i = k+1$,

Table 1: Divided differences table.

Nodes	0 order divided differences	1 order divided differences		k order divided differences	$k+1$ order divided differences	$k+2$ order divided differences		$n+1$ order divided differences
x_0	y_0^0							
x_1	y_1^0	y_1^1						
$\vdots$	$\vdots$	$\vdots$	$\ddots$					
x_k	y_k^0	y_k^1	$\cdots$	y_k^k				
x_k	y_k^0	y_k^1	$\cdots$	y_k^k	λ			
x_{k+1}	y_{k+1}^0	y_{k+1}^1	$\cdots$	y_{k+1}^k	y_{k+1}^{k+1}	z_{k+1}^{k+1}		
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	
x_n	y_n^0	y_n^1	$\cdots$	y_n^k	y_n^{k+1}	z_n^{k+1}	$\cdots$	z_n^n

$$
\begin{aligned}
P_n^{(0)}(x_i) &= c_0 + c_1(x_i - x_0) + c_2(x_i - x_0)(x_i - x_1) + \cdots + c_k(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1}) \\
&\quad + c_{k+1}^0(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k) + c_{k+1}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k)^2 \\
&= c_0 + c_1(x_i - x_0) + c_2(x_i - x_0)(x_i - x_1) + \cdots + c_k(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1}) \\
&\quad + \lambda(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k) + \frac{y_i^{k+1} - \lambda}{x_i - x_k}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k)^2 \\
&= c_0 + c_1(x_i - x_0) + c_2(x_i - x_0)(x_i - x_1) + \cdots + c_k(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1}) \\
&\quad + y_i^{k+1}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k) \\
&= \cdots = y_{k+1}^0 = y_{k+1}.
\end{aligned}
$$

If $n \geq i \geq k+2$,

$$
\begin{aligned}
P_n^{(0)}(x_i) &= c_0 + c_1(x_i - x_0) + c_2(x_i - x_0)(x_i - x_1) + \cdots + c_k(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1}) \\
&\quad + c_{k+1}^0(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k) + c_{k+1}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k)^2 + \\
&\quad \cdot c_{k+2}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k)^2(x_i - x_{k+1}) + \cdots + \\
&\quad \cdot c_i(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k)^2(x_i - x_{k-1})\cdots(x_i - x_{i-1}) \\
&= c_0 + c_1(x_i - x_0) + c_2(x_i - x_0)(x_i - x_1) + \cdots + c_k(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1}) \\
&\quad + c_{k+1}^0(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k) + c_{k+1}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k)^2 + \\
&\quad \cdot c_{k+2}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k)^2(x_i - x_{k+1}) + \cdots + \\
&\quad \cdot \frac{z_i^{i-1} - z_{i-1}^{i-1}}{x_i - x_{i-1}}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k)^2(x_i - x_{k+1})\cdots(x_i - x_{i-1}) \\
&= c_0 + c_1(x_i - x_0) + c_2(x_i - x_0)(x_i - x_1) + \cdots + c_k(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1}) \\
&\quad + c_{k+1}^0(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k) + c_{k+1}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k)^2 + \\
&\quad \cdot c_{k+2}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k)^2(x_i - x_{k+1}) + \cdots + \\
&\quad \cdot z_i^{i-1}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k)^2(x_i - x_{k+1})\cdots(x_i - x_{i-2})
\end{aligned}
$$

$$
\begin{aligned}
&= \cdots = c_0 + c_1(x_i - x_0) + c_2(x_i - x_0)(x_i - x_1) + \cdots + c_k(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1}) \\
&\quad + c_{k+1}^0(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k) + z_i^{k+1}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k)^2 \\
&= c_0 + c_1(x_i - x_0) + c_2(x_i - x_0)(x_i - x_1) + \cdots + c_k(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1}) \\
&\quad + \lambda(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k) + \frac{y_i^{k+1} - \lambda}{x_i - x_k}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k)^2 \\
&= c_0 + c_1(x_i - x_0) + c_2(x_i - x_0)(x_i - x_1) + \cdots + c_k(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1}) \\
&\quad + y_i^{k+1}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{k-1})(x_i - x_k) \\
&= \cdots = y_{k+1}^0 = y_{k+1}.
\end{aligned}
\tag{9}
$$

So, we have

$$
P_n^{(0)}(x_i) = y_i, \quad i = 0, 1, \ldots, n. \tag{10}
$$

Therefore, the results follow.

Similarly, we can also consider the Newton-type polynomial interpolation with two parameters with the proposed algorithm. It also can be divided into two cases. One case is that an arbitrary point from the original points is treated as a virtual treble point; another case is that any two points in the original points are regarded as two virtual double points. The Newton interpolation polynomial with more parameters can be constructed similarly. □

3. Bivariate Parameterized Newton-Type Polynomial Interpolation

Now, we generalize some applications of the previous results to the multivariate Newton-type polynomial interpolation.

Suppose $(\prod_{m,n} \subset D \subset R^2)$ is a set of two dimensional points on the rectangular region D, $f(x, y)$ is a bivariate real function defined on the rectangular region D, and let

$$
f(x_i, y_j) = f_{i,j}, \quad i = 0, 1, \ldots, m;\ j = 0, 1, \ldots, n. \tag{11}
$$

The bivariate Newton polynomial interpolation is as follows:

$$
\begin{aligned}
N(x, y) &= a_{0,0} + a_{0,1}(y - y_0) + a_{0,2}(y - y_0)(y - y_1) + \cdots + a_{0,m}(y - y_0)\cdots(y - y_{m-1}) \\
&\quad + \left(a_{1,0} + a_{1,1}(y - y_0) + a_{1,2}(y - y_0)(y - y_1) + \cdots + a_{1,m}(y - y_0)\cdots(y - y_{m-1})\right)(x - x_0) \\
&\quad + \cdots + \left(a_{n,0} + a_{n,1}(y - y_0) + a_{n,2}(y - y_0)(y - y_1) + \cdots + a_{n,m}(y - y_0)\cdots(y - y_{m-1})\right)(x - x_0)\cdots(x - x_{n-1}),
\end{aligned}
\tag{12}
$$

and $a_{i,j}$ $(i = 0, 1, \ldots, m;\ j = 0, 1, \ldots, n)$ represent the bivariate partial divided differences.

Theorem 2 (see [2]). *Suppose $(\prod_{m,n} \subset D \subset R^2)$ is a set of two dimensional points on rectangular region D and $f(x, y)$ is a real function defined on rectangular region D; then,*

$$
\begin{aligned}
&N(x_i, y_j) = f_{i,j}, \quad \forall (x_i, y_j) \in \prod_{m,n}, \\
&\qquad i = 0, 1, \ldots, m;\ j = 0, 1, \ldots, n.
\end{aligned}
\tag{13}
$$

3.1. Bivariate Newton-Type Polynomial Interpolation with a Single Parameter. By introducing a new parameter λ, we use any point of the original points $(x_k, y_l, f_{k,l})$ $(k = 0, 1, \ldots, m; l = 0, 1, \ldots, n)$ as a virtual double point, while the multiplicity of the other points remains 1. The Newton-type polynomial interpolation based on parameter λ can be constructed using Algorithm 1.

Algorithm 1

Step 1: initialization.

$$
f_{i,j}^{(0,0)} = f(x_i, y_j), \quad i = 0, 1, \ldots, m;\ j = 0, 1, \ldots, n. \tag{14}
$$

Step 2: for $(j = 0, 1, \ldots, n; p = 1, 2, \ldots, m; i = p, p+1, \ldots, m)$,

$$
f_{i,j}^{(p,0)} = \frac{f_{i,j}^{(p-1,0)} - f_{p-1,j}^{(p-1,0)}}{x_i - x_{p-1}}. \tag{15}
$$

Step 3: For $(i = 0, 1, \ldots, k-1, k+1, \ldots, m; q = 1, 2, \ldots, n; j = q, q+1, \ldots, n)$,

$$
f_{i,j}^{(i,q)} = \frac{f_{i,j}^{(i,q-1)} - f_{i,q-1}^{(i,q-1)}}{y_j - y_{q-1}}. \tag{16}
$$

Step 4: by introducing parameter λ into the formula $f_{k,j}^{(k,0)}$ $(j = 0, 1, \ldots, n)$ and using formulas (2)–(5), the final results are

$$\left(a_{k,0}, a_{k,1}, \ldots, a_{k,l}, a_{k,l+1}^{0}, a_{k,l+1}, a_{k,l+2}, a_{k,l+3}, \ldots, a_{k,n}\right)^{T}. \tag{17}$$

Step 5: using the data in formulas (16) and (17), the Newton-type polynomial interpolation with a single parameter with regard to y can be constructed:

$$A_i(y) = \begin{cases} f_{i,0}^{(i,0)} + f_{i,1}^{(i,1)}(y - y_0) + f_{i,2}^{(i,2)}(y - y_0)(y - y_1) + \cdots + f_{i,n}^{(i,n)}(y - y_0)(y - y_1)\cdots(y - y_{n-1}), \\ \qquad i = 0, 1, \cdots, k-1, k+1, \cdots, m, \\ a_{k,0} + a_{k,1}(y - y_0) + \cdots + a_{k,l}(y - y_0)\cdots(y - y_{l-1}) + a_{k,l+1}^{0}(y - y_0)\cdots(y - y_{l-1})(y - y_l) \\ +a_{k,l+1}(y - y_0)\cdots(y - y_{l-1})(y - y_l)^2 + a_{k,l+2}(y - y_0)\cdots(y - y_{l-1})(y - y_l)^2(y - y_{l+1}) + \cdots + \\ a_{k,n}(y - y_0)\cdots(y - y_{l-1})(y - y_l)^2(y - y_{l+1})\cdots(y - y_{n-1}), \quad i = k. \end{cases} \tag{18}$$

Step 6: let

$$\begin{aligned} N_{m,n}^{0}(x, y) = {} & A_0(y) + A_1(y)(x - x_0) \\ & + A_2(y)(x - x_0)(x - x_1) + \cdots \\ & + A_m(y)(x - x_0)(x - x_1)\cdots(x - x_{m-1}). \end{aligned} \tag{19}$$

Then, $N_{m,n}^{0}(x, y)$ is a bivariate Newton-type polynomial interpolation with a single parameter.

Theorem 3. *Given the interpolation points $f_{i,j}^{(0,0)} = f(x_i, y_j)$ $(i = 0, 1, \ldots, m;\ j = 0, 1, \ldots, n)$, the bivariate Newton-type polynomial interpolation with a single parameter $N_{m,n}^{0}(x, y)$ satisfies*

$$N_{m,n}^{0}(x_i, y_j) = f_{i,j}, \quad \forall (x_i, y_j) \in \Pi_{m,n}, \quad i = 0, 1, \ldots, m;\ j = 0, 1, \ldots, n. \tag{20}$$

Proof. For an arbitrary point $(x_i, y_j, f_{i,j})$, if $(i = 0, 1, \ldots, k-1, k+1, \ldots, m)$, it holds that

$$\begin{aligned} A_i(y_j) = {} & f_{i,0}^{(i,0)} + f_{i,1}^{(i,1)}(y_j - y_0) + f_{i,2}^{(i,2)}(y_j - y_0)(y_j - y_1) \\ & + \cdots + f_{i,j}^{(i,j)}(y_j - y_0)(y_j - y_1) \\ & \cdots(y_j - y_{j-1}) = \cdots = f_{i,j}^{(i,0)}. \end{aligned} \tag{21}$$

If $(i = k)$,

$$\begin{aligned} A_i(y) = {} & a_{k,0} + a_{k,1}(y - y_0) + \cdots + a_{k,l}(y - y_0)\cdots(y - y_{l-1}) + a_{k,l+1}^{0}(y - y_0)\cdots(y - y_{l-1})(y - y_l) \\ & + a_{k,l+1}(y - y_0)\cdots(y - y_{l-1})(y - y_l)^2 + a_{k,l+2}(y - y_0)\cdots(y - y_{l-1})(y - y_l)^2(y - y_{l+1}) + \cdots + \\ & \cdot a_{k,n}(y - y_0)\cdots(y - y_{l-1})(y - y_l)^2(y - y_{l+1})\cdots(y - y_{n-1}). \end{aligned} \tag{22}$$

Regardless of $(0 \le j < l, l = j, n \ge j > l)$, from Theorem 1, we can derive

$$\begin{aligned} A_i(y_j) = {} & a_{k,0} + a_{k,1}(y_j - y_0) + \cdots + a_{k,l}(y_j - y_0)\cdots(y_j - y_{l-1}) + a_{k,l+1}^{0}(y_j - y_0)\cdots(y_j - y_{l-1})(y_j - y_l) \\ & + a_{k,l+1}(y_j - y_0)\cdots(y_j - y_{l-1})(y_j - y_l)^2 + a_{k,l+2}(y_j - y_0)\cdots(y_j - y_{l-1})(y_j - y_l)^2(y_j - y_{l+1}) \\ & + \cdots + a_{k,j}(y_j - y_0)\cdots(y_j - y_{l-1})(y_j - y_l)^2(y_j - y_{l+1})\cdots(y_j - y_{j-1}) = \cdots = f_{i,j}^{(i,0)}. \end{aligned} \tag{23}$$

From Theorem 4, $(\forall (x_i, y_j) \in \Pi_{m,n})$,

$$\begin{aligned} N_{m,n}^{0}(x_i, y_j) = {} & A_0(y_t) + A_1(y_j)(x_i - x_0) + A_2(y_j)(x_i - x_0)(x_i - x_1) + \cdots + A_s(y_j)(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1}) \\ = {} & f_{0,j}^{(0,0)} + f_{1,j}^{(1,0)}(x_i - x_0) + f_{2,j}^{(2,0)}(x_i - x_0)(x_i - x_1) + \cdots + f_{i,j}^{(i,0)}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1}) = \cdots = f_{i,j}^{(0,0)} = f_{i,j}. \end{aligned} \tag{24}$$

Then, we have proved Theorem 3.

Similarity, we can also consider the Newton-type polynomial interpolation with two parameters. There are three cases: one case is that a point was viewed as a virtual treble point; another case is that two points were viewed as virtual double points in the same column; the third case is taking two points as virtual double points in different columns. The bivariate Newton-type polynomial interpolation with three or more parameters can be derived in a similar fashion. □

3.2. Dual Bivariate Newton-Type Polynomial Interpolation with a Single Parameter. From Algorithm 1, according to the partial divided difference of x first and then concerning the partial divided difference of y, the novel bivariate Newton-type polynomial interpolation was constructed. In fact, we can also construct it by first regarding the partial divided difference of y and then the partial divided difference of x. By introducing a new parameter θ, any point in the original interpolant points $(x_k, y_l, f_{k,l})(k = 0, 1, \ldots, m; l = 0, 1, \ldots, n)$ is regarded as a virtual double point. Then, the Newton-type polynomial interpolation with parameter θ can be obtained using Algorithm 2.

Algorithm 2

Step 1: initialization.

$$f_{i,j}^{(0,0)} = f\left(x_i, y_j\right), \quad i = 0, 1, \ldots, m;\ j = 0, 1, \ldots, n. \tag{25}$$

Step 2: if $(i = 0, 1, \ldots, m; q = 1, 2, \ldots, n; j = q, q+1, \ldots, n)$, let

$$f_{i,j}^{(0,q)} = \frac{f_{i,j}^{(0,q-1)} - f_{i,q-1}^{(0,q-1)}}{y_j - y_{q-1}}. \tag{26}$$

Step 3: for $(j = 0, 1, \ldots, l-1, l+1, \ldots, n;\ p = 1, 2, \ldots, m; i = p, p+1, \ldots, m)$,

$$f_{i,j}^{(p,j)} = \frac{f_{i,j}^{(p-1,j)} - f_{p-1,j}^{(p-1,j)}}{x_i - x_{p-1}}. \tag{27}$$

Step 4: by introducing a parameter θ into the $f_{i,l}^{(0,l)} (i = 0, 1, \ldots, m)$ and calculating them using formulas (2)–(5), the final results are

$$a_{0,l}, a_{1,l}, \ldots, a_{k,l}, a_{k+1,l}^{0}, a_{k+1,l}, a_{k+2,l}, \ldots, a_{m,l}. \tag{28}$$

Step 5: by using the data in formulas (27) and (28), the univariate Newton-type polynomial interpolation with a single parameter in regard to x can be constructed as follows:

$$A_j(x) = \begin{cases} f_{0,j}^{(0,j)} + f_{1,j}^{(1,j)}(x - x_0) + f_{2,j}^{(2,j)}(x - x_0)(x - x_1) + \cdots + f_{m,j}^{(m,j)}(x - x_0)(x - x_1)\cdots(x - x_{m-1}), \\ \qquad\qquad j = 0, 1, \ldots, l-1, l+1, \ldots, n, \\ a_{0,l} + a_{1,l}(x - x_0) + \cdots + a_{k,l}(x - x_0)\cdots(x - x_{l-1}) + a_{k+1,l}^{0}(x - x_0)\cdots(x - x_{l-1})(x - x_l) \\ +a_{k+1,l}(x - x_0)\cdots(x - x_{l-1})(x - x_l)^2 + a_{k+2,l}(x - x_0)\cdots(x - x_{l-1})(x - x_l)^2(x - x_{l+1}) + \cdots + \\ a_{m,l}(x - x_0)\cdots(x - x_{l-1})(x - x_l)^2(x - x_{l+1})\cdots(x - x_{m-1}), \quad j = l. \end{cases} \tag{29}$$

Step 6: let

$$\begin{aligned} N_{m,n}^{2}(x, y) = {} & A_0(x) + A_1(x)(y - y_0) \\ & + A_2(x)(y - y_0)(y - y_1) + \cdots \\ & + A_m(x)(y - y_0)(y - y_1)\cdots(y - y_{m-1}). \end{aligned} \tag{30}$$

Then, $N_{m,n}^{2}(x, y)$ is a dual bivariate Newton-type polynomial interpolation with a single parameter.

We call $N_{m,n}^{2}(x, y)$ as the dual interpolation of $N_{m,n}^{0}(x, y)$. As a matter of fact, the bivariate Newton-type polynomial interpolation with a single parameter can also be calculated by the information matrix method. This algorithm borrows elementary transformation thought of linear algebra, so it is easy to implement. We can calculate the partial divided differences in Algorithms 1 and 2 in a manner of the information matrix method. Here, we only give the information matrix method of the dual bivariate Newton-type polynomial interpolation with a single parameter. Algorithm 2 can also be described by Algorithm 3 based on information matrix as follows.

Algorithm 3

Step 1: initialization. Let

$$f_{i,j}^{(0,0)} = f\left(x_i, y_j\right), \quad (i = 0, 1, \ldots, m;\ j = 0, 1, \ldots, n). \tag{31}$$

One can define the following initial information matrix:

$$M_1 = \begin{bmatrix} f_{0,0}^{(0,0)} & f_{1,0}^{(0,0)} & \cdots & f_{m,0}^{(0,0)} \\ f_{0,1}^{(0,0)} & f_{1,1}^{(0,0)} & \cdots & f_{m,1}^{(0,0)} \\ \vdots & \vdots & \cdots & \vdots \\ f_{0,n}^{(0,0)} & f_{1,n}^{(0,0)} & \cdots & f_{m,n}^{(0,0)} \end{bmatrix}. \tag{32}$$

Step 2: for $(i = 0, 1, \cdots, \cdots, m; q = 1, 2, \cdots, n; j = q, q+1, \cdots, n)$, let

$$f_{i,j}^{(0,q)} = \frac{f_{i,j}^{(0,q-1)} - f_{i,q-1}^{(0,q-1)}}{y_j - y_{q-1}}. \tag{33}$$

The above recursive process aims to change the initial information matrix M_1 into the matrix M_2 by using row transformation:

$$M_2 = \begin{bmatrix} f_{0,0}^{(0,0)} & f_{1,0}^{(0,0)} & \cdots & f_{m,0}^{(0,0)} \\ f_{0,1}^{(0,1)} & f_{1,1}^{(0,1)} & \cdots & f_{m,1}^{(0,1)} \\ \vdots & \vdots & \cdots & \vdots \\ f_{0,n}^{(0,n)} & f_{1,n}^{(0,n)} & \cdots & f_{m,n}^{(0,n)} \end{bmatrix}. \tag{34}$$

Step 3: for $(j = 0, 1, \cdots, l-1, l+1, \cdots, n; p = 1, 2, \cdots, m; i = p, p+1, \cdots, m)$, let

$$f_{i,j}^{(p,j)} = \frac{f_{i,j}^{(p-1,j)} - f_{p-1,j}^{(p-1,j)}}{x_i - x_{p-1}}. \tag{35}$$

The above recursive process aims to change the $(0, 1, \ldots, l-1, l+1, \ldots, m)$ columns in the initial information matrix M_2 into the matrix M_3 by using column transformation:

$$M_3 = \begin{bmatrix} f_{0,0}^{(0,0)} & f_{1,0}^{(1,0)} & \cdots & f_{m,0}^{(m,0)} \\ f_{0,1}^{(0,1)} & f_{1,1}^{(1,1)} & \cdots & f_{m,1}^{(m,1)} \\ \vdots & \vdots & \cdots & \vdots \\ f_{0,l-1}^{(0,l-1)} & f_{1,l-1}^{(1,l-1)} & \cdots & f_{m,l-1}^{(m,l-1)} \\ f_{0,l}^{(0,l)} & f_{1,l}^{(0,l)} & \cdots & f_{m,l}^{(0,l)} \\ f_{0,l+1}^{(0,l+1)} & f_{1,l+1}^{(1,l+1)} & \cdots & f_{m,l+1}^{(m,l+1)} \\ \vdots & \vdots & \cdots & \vdots \\ f_{0,n}^{(0,n)} & f_{1,n}^{(1,n)} & \cdots & f_{m,n}^{(m,n)} \end{bmatrix}. \tag{36}$$

By introducing a parameter θ into the $f_{i,l}^{(0,l)} (i = 0, 1, \ldots, m)$ in the information matrix M_3 and calculating them using formulas (2)–(5), the final results are

$$a_{0,l}, a_{1,l}, \ldots, a_{k,l}, a_{k+1,l}^{0}, a_{k+1,l}, a_{k+2,l}, \ldots, a_{m,l}. \tag{37}$$

Step 4: by using the data in formulas (36) and (37), the univariate Newton-type polynomial interpolation with a single parameter in regard to x can be constructed as follows:

$$A_j(x) = \begin{cases} f_{0,j}^{(0,j)} + f_{1,j}^{(1,j)}(x - x_0) + f_{2,j}^{(2,j)}(x - x_0)(x - x_1) + \cdots + f_{m,j}^{(m,j)}(x - x_0)(x - x_1)\cdots(x - x_{m-1}), \\ \qquad j = 0, 1, \cdots, l-1, l+1, \cdots, n, \\ a_{0,l} + a_{1,l}(x - x_0) + \cdots + a_{k,l}(x - x_0)\cdots(x - x_{l-1}) + a_{k+1,l}^{0}(x - x_0)\cdots(x - x_{l-1})(x - x_l) \\ +a_{k+1,l}(x - x_0)\cdots(x - x_{l-1})(x - x_l)^2 + a_{k+2,l}(x - x_0)\cdots(x - x_{l-1})(x - x_l)^2(x - x_{l+1}) + \cdots + \\ a_{m,l}(x - x_0)\cdots(x - x_{l-1})(x - x_l)^2(x - x_{l+1})\cdots(x - x_{m-1}), \quad j = l. \end{cases} \tag{38}$$

Step 5: let

$$\begin{aligned} N_{m,n}^{2}(x, y) = {} & A_0(x) + A_1(x)(y - y_0) \\ & + A_2(x)(y - y_0)(y - y_1) + \cdots \\ & + A_m(x)(y - y_0)(y - y_1)\cdots(y - y_{m-1}). \end{aligned} \tag{39}$$

Then, $N_{m,n}^{2}(x, y)$ is the dual bivariate Newton-type polynomial interpolation with a single parameter.

Theorem 4. *Given the interpolation data $(x_i, y_j, f_{i,j})(i = 0, 1, \ldots, m; j = 0, 1, \ldots, n)$, the dual bivariate Newton-type polynomial interpolation with a single parameter $N_{m,n}^{2}(x, y)$ satisfies*

$$N_{m,n}^{2}(x_i, y_j) = f_{i,j}, \quad \forall (x_i, y_j) \in \Pi_{m,n}, i = 0, 1, \ldots, m; j = 0, 1, \ldots, n. \tag{40}$$

Proof. For an arbitrary point $(x_i, y_j, f_{i,j})$, if $(j = 0, 1, \ldots, l-1, l+1, \ldots, n,)$, we have

$$\begin{aligned} A_j(x_i) = {} & f_{i,0}^{(i,0)} + f_{i,1}^{(i,1)}(x_i - x_0) + f_{i,2}^{(i,2)}(x_i - x_0)(x_i - x_1) \\ & + \cdots + f_{i,j}^{(i,j)}(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1}) \\ = {} & \cdots = f_{i,j}^{(0,j)}. \end{aligned} \tag{41}$$

If $j = l$,

$$\begin{aligned} A_j(x) = {} & a_{0,l} + a_{1,l}(x - x_0) + \cdots + a_{k,l}(x - x_0)\cdots(x - x_{l-1}) + a^0_{k+1,l}(x - x_0)\cdots(x - x_{l-1})(x - x_l) \\ & + a_{k+1,l}(x - x_0)\cdots(x - x_{l-1})(x - x_l)^2 + a_{k+2,l}(x - x_0)\cdots(x - x_{l-1})(x - x_l)^2(x - x_{l+1}) \\ & + \cdots + a_{m,l}(x - x_0)\cdots(x - x_{l-1})(x - x_l)^2(x - x_{l+1})\cdots(x - x_{m-1}). \end{aligned} \tag{42}$$

Regardless of $(0 \le i < k, i = k, n \ge i > k)$, from Theorem 1, we have

$$\begin{aligned} A_j(x_i) = {} & a_{0,l} + a_{1,l}(x_i - x_0) + \cdots + a_{k,l}(x_i - x_0)\cdots(x_i - x_{l-1}) + a^0_{k+1,l}(x_i - x_0)\cdots(x_i - x_{l-1})(x_i - x_l) \\ & + a_{k+1,l}(x_i - x_0)\cdots(x_i - x_{l-1})(x_i - x_l)^2 + a_{k+2,l}(x_i - x_0)\cdots(x_i - x_{l-1})(x_i - x_l)^2(x_i - x_{l+1}) \\ & + \cdots + a_{i,l}(x_i - x_0)\cdots(x_i - x_{l-1})(x_i - x_l)^2(x_i - x_{l+1})\cdots(x_i - x_{i-1}) = \cdots = f^{(0,j)}_{i,j}. \end{aligned} \tag{43}$$

So, we have

$$\begin{aligned} N^2_{m,n}(x_i, y_j) &= A_0(x_i) + A_1(x_i)(y_j - y_0) + A_2(x_i)(y_j - y_0)(y_j - y_1) + \cdots + A_t(x_i)(y_j - y_0)(y_j - y_1)\cdots(y_j - y_{j-1}) \\ &= f^{(0,0)}_{i,0} + f^{(0,1)}_{i,1}(y_j - y_0) + f^{(0,2)}_{i,2}(y_j - y_0)(y_j - y_1) + \cdots + f^{(0,j)}_{i,j}(y_j - y_0)(y_j - y_1)\cdots(y_j - y_{j-1}) = \cdots = f^{(0,0)}_{s,t} \\ &= f_{s,t}. \end{aligned} \tag{44}$$

Then, we can obtain the result.

In addition, it can also construct other dual bivariate parameterized Newton-type polynomial interpolations, similar to the proposed algorithm in Section 3.2. Furthermore, according to the previous discussion, we can also construct parameterized Newton-type polynomial interpolation in both x and y directions. The formulas of every new parameterized Newton-type polynomial interpolation depend on the selected parameters. We can construct different parameterized Newton-type polynomial interpolation functions according to the application needs. At the same time, the value of the interpolant function can be changed at any point in the interpolant region under the condition that the values of the interpolant points are fixed by selecting appropriate parameters.

As a matter of fact, one can also construct a one-parameter family of interpolating polynomials:

$$P^{\lambda}(x) = \phi_n(x) + P_n(x), \tag{45}$$

with $P_n(x)$ being the classical interpolating polynomial and $\Phi_n(x) = \prod_k^n (x - x_k)$.

The proposed parameterized Newton-like interpolation polynomial is another approach to the one-parameter family of interpolating polynomials. In this paper, the proposed algorithms are only based on (partial) divided differences, which seems a little complicated. In fact, the calculation is the same as that of the classical Newton interpolation polynomial. We can also discuss the univariate case with more parameters and bivariate case with one or more parameters similarly. The algorithm given in this paper describes a new approach method for Newton interpolation polynomial.

The parameterized Newton-type polynomial interpolation defined in this paper is a local interpolation in each subinterval or subrectangle, which only depends on the function values. There are some parameters in each interpolating region, and when the parameters vary, the interpolant function will be changed under the condition that the values of the interpolant points are fixed. Therefore, the interpolation curves and surfaces will change with different parameters. Thus, the shape of the interpolant curves or surface can be adjusted for the given interpolation points. However, the question of how to select the optimal

Table 2: Interpolation data.

i	1	2	3	4
x_i	0	1	2	3
f_i	0	0	1	3

parameters of the proposed interpolant algorithms is still a challenging problem, which we will consider in our future work. □

4. Numerical Examples

In this section, we give some examples to demonstrate how the proposed algorithms are implemented and how the parameters can be adjusted to change the curve or surface shape.

Example 1. Let the interpolation data be as given in Table 2.

From the interpolation data in the Table 2, we get the Newton interpolation polynomial as

$$p(x) = \frac{1}{2}x(x-1). \tag{46}$$

Take $(2,1)$ as a virtual double point, and the multiplicity of the other points remains 1. The Newton-type polynomial interpolation with one parameter λ derived via Algorithm 1 is

$$p^{(0)}(x) = \frac{1}{2}x(x-1) + \lambda x(x-1)(x-2) - \lambda x(x-1)(x-2)^2. \tag{47}$$

It is easy to verify that

$$p(x_i) = p^{(0)}(x_i) = f_i, \quad (i = 0, 1, 2). \tag{48}$$

If the interpolant points are given, the shape of interpolating curves is fixed. This is because of the uniqueness of the interpolation function. The parameterized Newton-type polynomial interpolation methods defined in this paper are parametric. By changing the parameters, the interpolation function can be adjusted while observing the condition that the values of the interpolant points are fixed. The locality constraint in the interpolation curves is to make the function value at a specific point (x, y) equal to a given real number in actual design [29, 30]. The central point in the interpolating curve is more important than other points for shape control [29, 30], so we consider the central point value control problem.

In fact, if we take $M = 1.5$, based on the parameterized Newton-type polynomial interpolation, we can get $\lambda = -2$, and the function value of the central point is $p^{(0)}(1.5) = 1.5$. Figure 1(b) shows the plot of the interpolation $p^{(0)}(x)$. If we take $M = 0$, it is easy to prove $\lambda = (2/3)$, and Figure 1(c) shows the plot of the interpolation $p^{(0)}(x)$. In this case, the function value of central point is $p^{(0)}(1.5) = 0$. If we take $M = -1$, it is easy to prove $\lambda = (22/9)$, and Figure 1(d) shows the plot of the interpolation $p^{(0)}(x)$; the function value of central point is $p^{(0)}(1.5) = -1$.

By choosing different values of parameter λ, the function $p^{(0)}(x)$ invariably satisfies the given interpolation data. The value of the interpolation function $p^{(0)}(x)$ can be changed in the interpolant region by choosing appropriate parameter λ.

Example 2. Let the interpolation data be as given in Table 3.

We obtain the bivariate Newton polynomial interpolation by the method in [2] as

$$P(x, y) = 1 + y + x(1 + y). \tag{49}$$

Following Algorithm 1 in this paper, we increase the multiplicity of node $(0, 0, 1)$ to 2, the multiplicity of the other points remains 1, and then we construct the Hermite interpolation with first-order derivative in point $(0, 0, 1)$ by introducing parameter λ. We obtain the Newton-type polynomial interpolation with parameter λ with regard to y:

$$P^{(1)}(x, y) = 1 + \lambda y + (1 - \lambda)y^2 + x(1 + y). \tag{50}$$

Following Algorithm 2, we derive the Newton-type polynomial interpolation with parameter β with regard to x:

$$P^{(2)}(x, y) = 1 + \beta x + (1 - \beta)x^2 + y(1 + x). \tag{51}$$

Following Algorithms 1 and 2, we derive the Newton-type polynomial interpolation with parameters δ, η with regard to x and y:

$$P^{(3)}(x, y) = 1 + \delta x + (1 - \delta)x^2 + \eta y + (1 - \eta)y^2 + xy. \tag{52}$$

We can also obtain the bivariate Newton polynomial interpolation by adding times with parameter γ:

$$P^{(4)}(x, y) = 1 + x + y + xy + \gamma xy(x - 1)(y - 1). \tag{53}$$

It is easy to verify

$$P(x_i, y_j) = P^{(1)}(x_i, y_j) = P^{(2)}(x_i, y_j) = P^{(3)}(x_i, y_j) = P^{(4)}(x_i, y_j) = f_{i,j}, \quad (i, j = 0, 1). \tag{54}$$

It is easy to verify that the function $P^{(1)}(x, y)$ invariably satisfies the given interpolation data with the different values of parameter λ. Figure 2 shows the plot of the classical bivariate Newton polynomial interpolation $P(x, y)$.

Similar to interpolant curves, we consider the local point control problem of the interpolation surface, that is, how to make the function value of the interpolation at a point (x, y) equal to a given real number S in the actual design. We cannot adjust the shape of the classical Newton interpolation polynomial $P^{(1)}(x, y)$. We can use the algorithm in this article to solve this problem. Compared with other points, the center point is the key area of the interpolation patch. The function value of the central point is $P(0.5, 0.5) = 2.25$. If we set $S = -1$, based on the parameterized Newton-type

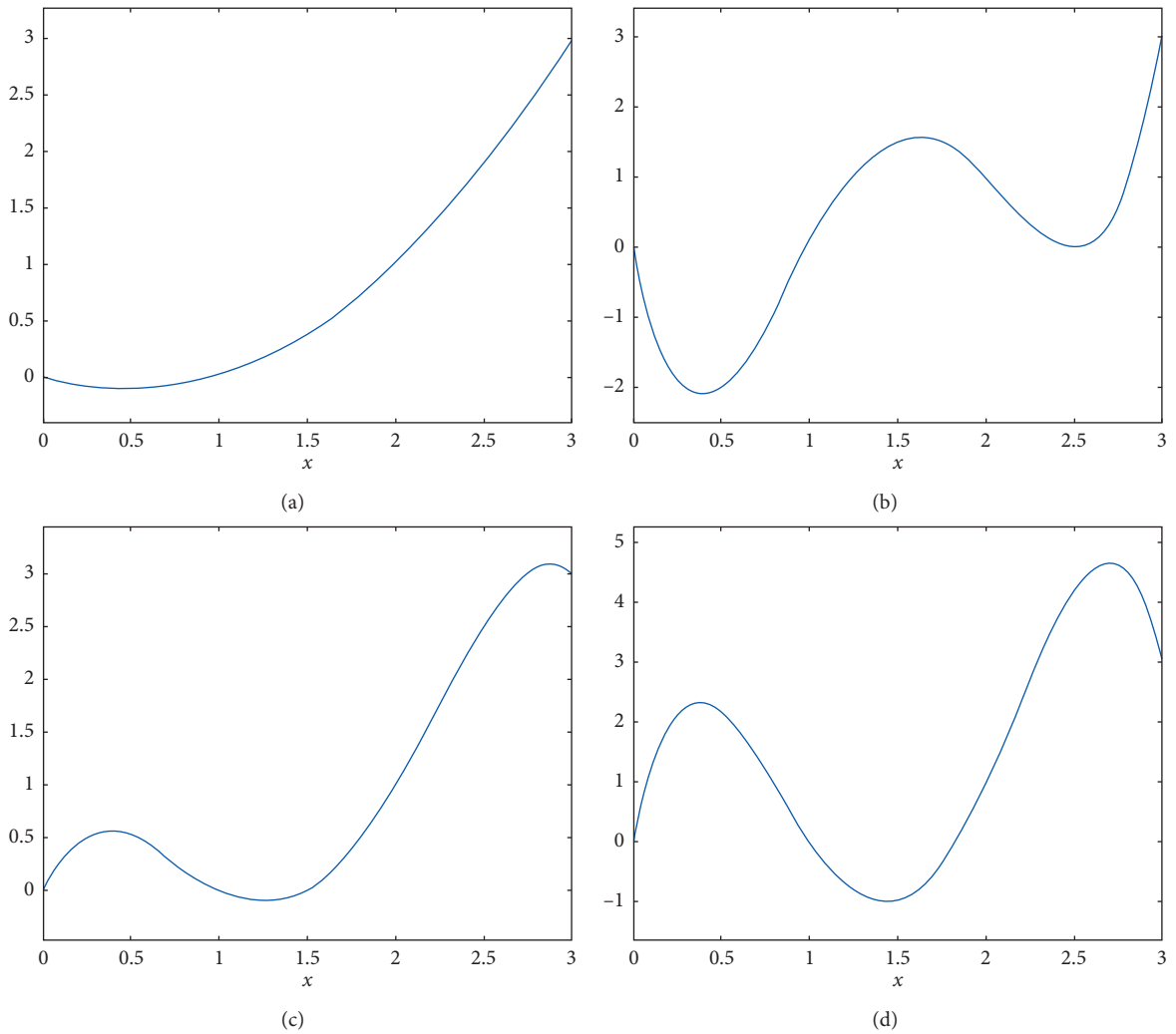

Figure 1: (a) The plot of $p(x)$. (b) The plot of $p^{(0)}(x)$ with $\lambda = -2$. (c) The plot of $p^{(0)}(x)$ with $\lambda = (2/3)$. (d) The plot of $p^{(0)}(x)$ with $\lambda = (22/9)$.

Table 3: Interpolation data.

(x_i, y_j)	$x_0 = 0$	$x_1 = 1$
$y_0 = 0$	1	2
$y_1 = 1$	2	4

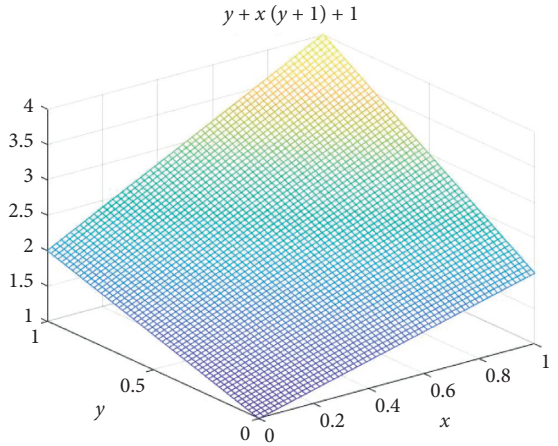

Figure 2: The plot of $P(x, y)$.

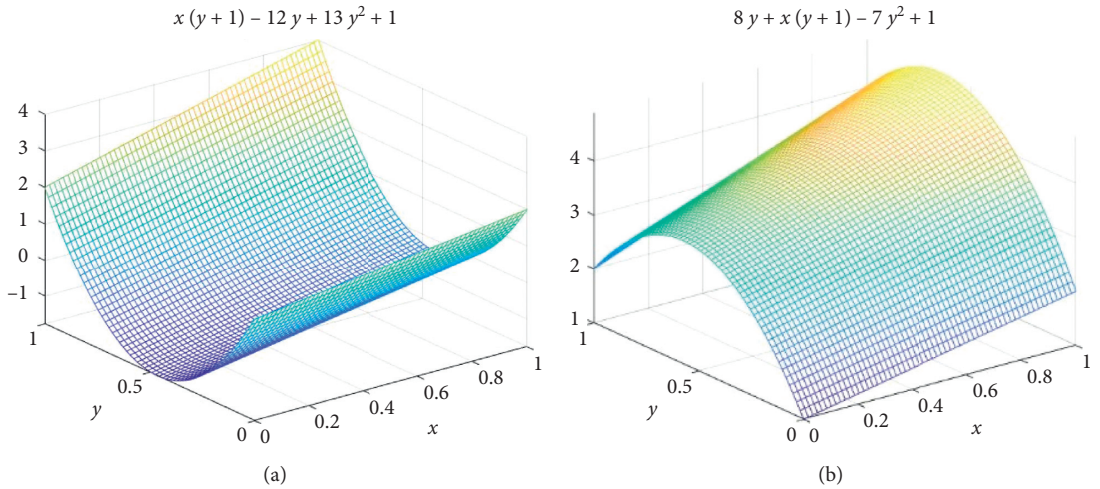

Figure 3: The plot of $P^{(1)}(x, y)$. (a) The plot of $P^{(1)}(x, y)$ with $\lambda = -12$. (b) The plot of $P^{(1)}(x, y)$ with $\lambda = 8$.

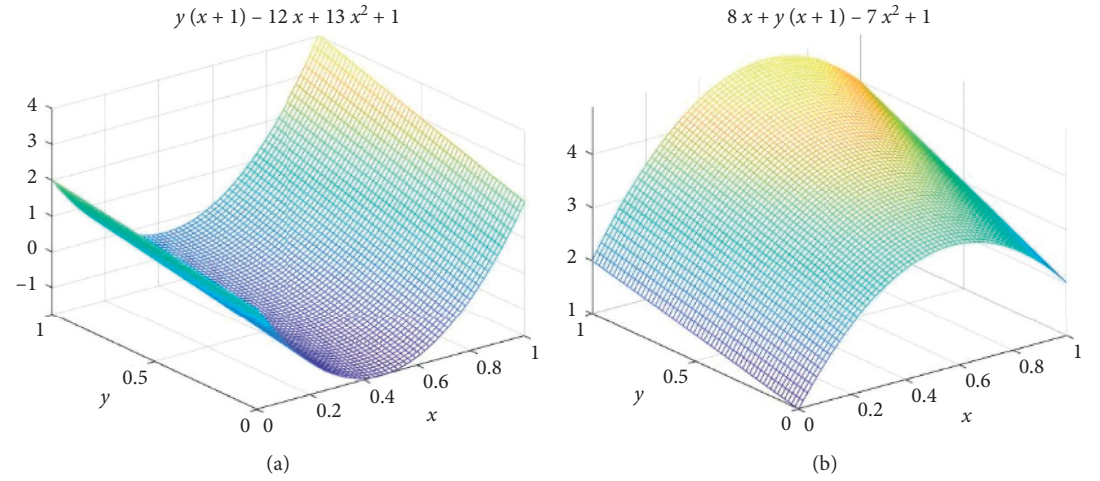

Figure 4: The plot of $P^{(2)}(x, y)$. (a) The plot of $P^{(2)}(x, y)$ with $\beta = -12$. (b) The plot of $P^{(2)}(x, y)$ with $\beta = 8$.

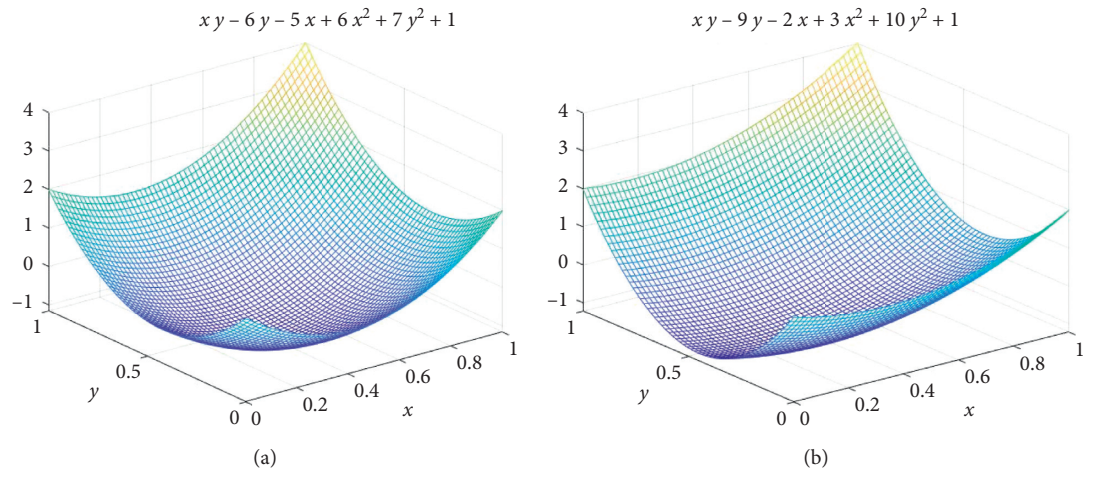

Figure 5: Continued.

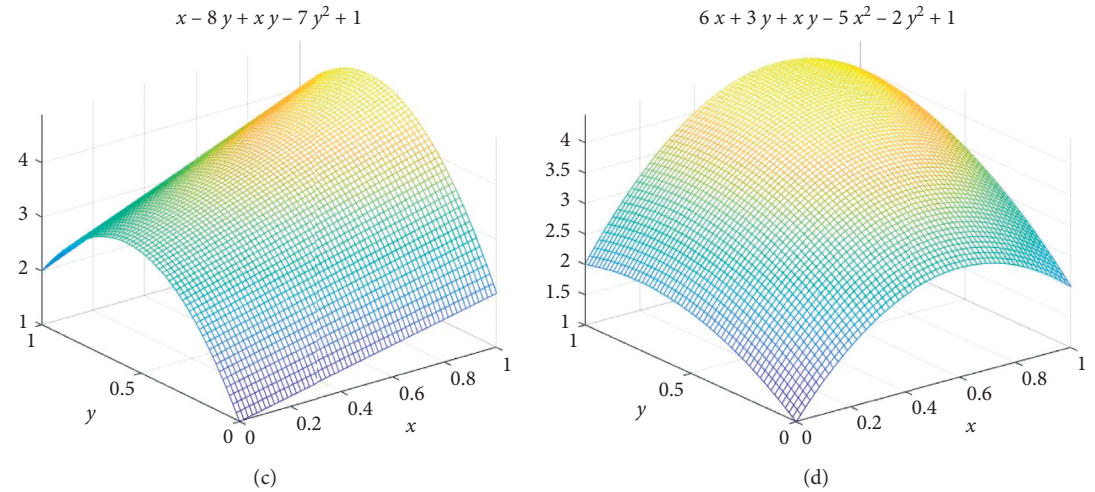

Figure 5: The plot of $P^{(3)}(x, y)$. (a) The plot of $P^{(3)}(x, y)$ with $\delta = -5, \eta = -6$. (b) The plot of $P^{(3)}(x, y)$ with $\delta = -9, \eta = -2$. (c) The plot of $P^{(3)}(x, y)$ with $\delta = 8, \eta = 1$. (d) The plot of $P^{(3)}(x, y)$ with $\delta = 3, \eta = 6$.

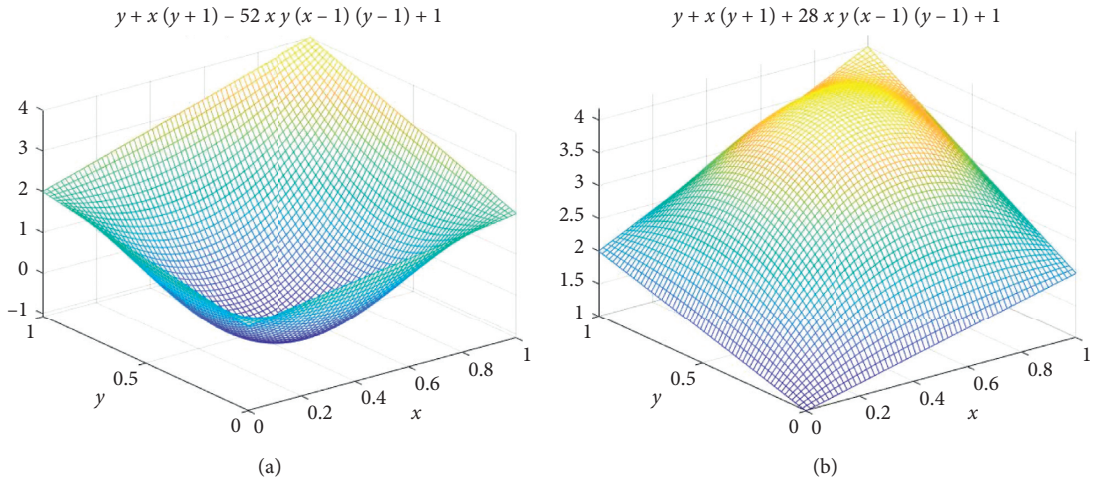

Figure 6: The plot of $P^{(4)}(x, y)$. (a) The plot of $P^{(4)}(x, y)$ with $\gamma = -52$. (b) The plot of $P^{(4)}(x, y)$ with $\gamma = 28$.

polynomial interpolation, we can get $\lambda = -12$, and the shape of the surface will "drop." In this case, the function value of the central point is $P^{(1)}(0.5, 0.5) = -1$. If we set $S = 4$, it is easy to prove $\lambda = 8$, and the shape of the surface "rises." In this case, the function value of central point is $P^{(1)}(0.5, 0.5) = 4$. Figures 3(a) and 3(b) show the plot of the parameterized Newton-type polynomial interpolation $P^{(1)}(x, y)$ with $\lambda = -12$ and $\lambda = 8$, respectively. Similarly, we can set $S = -1$ and $S = 4$ for $P^{(i)}(0.5, 0.5) = -1$, $(i = 2, 3, 4)$ and $P^{(i)}(0.5, 0.5) = 4$, $(i = 2, 3, 4)$; we can discuss polynomial interpolation $P^{(2)}(x, y)$, $P^{(3)}(x, y)$, and $P^{(4)}(x, y)$. Figures 4(a) and 4(b) show the plot of the parameterized Newton-type polynomial interpolation $P^{(2)}(x, y)$ with $\beta = -12$ and $\beta = 8$, respectively. Figures 5(a)–5(d) show the plot of the parameterized Newton-type polynomial interpolation $P^{(3)}(x, y)$ with $\delta = -5, \eta = -6$ and $\delta = -9, \eta = -2$ and $\delta = 8, \eta = 1$ and $\delta = 3, \eta = 6$, respectively. In fact, we can choose other combinations of parameters δ, η that satisfy certain conditions, so that $P^{(3)}(0.5, 0.5) = -1$ and $P^{(3)}(0.5, 0.5) = 4$. Figures 6(a) and 6(b) show the plot of the parameterized Newton-type polynomial interpolation $P^{(4)}(x, y)$ with $\gamma = 28$ and $\gamma = -52$, respectively.

For the given interpolation points, we get four Newton polynomial interpolations with one or two parameters. The polynomial $P^{(4)}(x, y)$ has a higher degree than the proposed method in this paper. In terms of the central point value control problem, the proposed method in this paper has a large adjustment space as shown in Figures 3–5. Meanwhile, we can adjust the shape of surface in one or in two directions or add parameters to adjust in both directions according to actual needs. The interpolation function can be adjusted under the condition that the values of the interpolant points are fixed. This is achieved by adjusting the parameters of our Newton-type polynomial interpolation methods. We can also modify the new Newton-type polynomial interpolation by parameter λ, so our methods give a new choice for the

application and a new method for studying the polynomial interpolation theory. According to the other algorithms in this paper, we can also construct many other novel Newton-type polynomial interpolations with more parameters similarly.

5. Conclusions and Future Work

We presented a new approach to interpolation method of Newton polynomials. We constructed several kinds of univariate and bivariate parameterized Newton-type polynomial interpolation. We discussed the interpolation theorem, the algorithms, the dual bivariate parameterized Newton-type polynomial interpolation, and algorithm based on information matrix. Our methods are easy to be used and theoretically sound. The value of the Newton-type polynomial interpolant function can be adjusted in the interpolant region by choosing appropriate parameter values, while observing the condition that the values of the interpolant points are fixed. According to the actual geometric design needs, the shape of the interpolation curves or surfaces can be adjusted, which is convenient in engineering and useful in computer-aided geometric design [28, 30]. The proposed method has advantages, such as a simple and explicit mathematical representation and ease of computation. But how to select the optimal parameter for application is a difficult problem. Our future work will concentrate on the following aspects:

(1) How to select the optimal parameter values and suitably adjust the shape of the interpolant curves or surface according to the actual design requirements.

(2) Investigating how the presented parameterized Newton-type polynomial interpolation method can be applied in other pixel-level image processing steps, such as super-resolution reconstruction, image upscaling, image rotation, removal of salt and pepper noise, image metamorphosis, and image inpainting.

(3) How to construct multivariate composite interpolation and blending rational interpolation over lacunary, triangular, pyramid-type, and other irregularity grids based on the parameterized Newton-type polynomial interpolation, Thiele continued fraction interpolation, and other classical interpolation methods. The proposed method can also be generalized to Hermite interpolation and osculatory rational interpolation.

In summary, the parameterized Newton-type polynomial interpolation algorithm has many advantages, such as a simple and explicit mathematical representation and ease of computation; the value of the Newton-type polynomial interpolant function can be adjusted in the interpolant region by choosing appropriate parameter values; according to the actual geometric design needs, the shape of the interpolation curves or surfaces can be adjusted.

In addition, with the help of the Samlson generalized inverse, it is easy to generalize the parameterized Newton-type polynomial interpolation method to vector-valued cases or matrix-valued cases [1, 15, 16].

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (nos. 61672204, 61673359, and 61806068), the Natural Science Foundation of Anhui Province (nos. 1908085MF184 and 1908085QF285), in part by the Key Scientific Research Foundation of Education Department of Anhui Province (nos. KJ2018A0555 and KJ2019A0833), and in part by the Key Technologies R&D Program of Anhui Province (no. 1804a09020058).

References

[1] R. H. Wang and G. Q. Zhu, *Approximation of Rational Interpolation and its Application*, Science Publishers, Beijing, China, 2004, in Chinese.

[2] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, NY, USA, 2nd edition, 2002.

[3] L. He, J. Tan, Z. Su, X. Luo, and C. Xie, "Super-resolution by polar Newton-Thiele's rational kernel in centralized sparsity paradigm," *Signal Processing: Image Communication*, vol. 31, pp. 86–99, 2015.

[4] T. Bai, J. Tan, M. Hu, and Y. Wang, "A novel algorithm for removal of salt and pepper noise using Continued Fractions interpolation," *Signal Processing*, vol. 102, pp. 247–255, 2014.

[5] M. Rahman, M. J. Khan, M. A. Asghar et al., "Image local features description through polynomial approximation," *IEEE Access*, vol. 7, pp. 183692–183705, 2019.

[6] D. N. H. Thanh, V. B. S. Prasath, L. M. Hieu, and S. Dvoenko, "An adaptive method for image restoration based on high-order total variation and inverse gradient," *Signal, Image and Video Processing*, vol. 14, no. 6, pp. 1189–1197, 2020.

[7] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*, Academic Press, New York, NY, USA, 1988.

[8] C. de Boor, *A Practical Guide to Splines*, Springer-Verlag, New York, NY, USA, Revised edition, 2001.

[9] W.-B. Zhong, X.-C. Luo, W.-L. Chang et al., "Toolpath interpolation and smoothing for computer numerical control machining of freeform surfaces: a review," *International Journal of Automation and Computing*, vol. 17, no. 1, pp. 1–16, 2020.

[10] S. Vats, B. B. Sagar, K. Singh, A. Ahmadian, and B. A. Pansera, "Performance evaluation of an independent time optimized infrastructure for big data analytics that maintains symmetry," *Symmetry*, vol. 12, no. 8, p. 1274, 2020.

[11] S. M. Vonza, "Newton-Tile-Type interpolational formula in the form of two-dimensional continued fraction with non-equivalent variables," *Mathematical Methods and Physicomechanical Fields*, vol. 47, pp. 67–72, 2004.

[12] D. N. Varsamis and N. P. Karampetakis, "On the Newton bivariate polynomial interpolation with applications," *Multidimensional Systems and Signal Processing*, vol. 25, no. 1, pp. 179–209, 2014.

[13] J. Tan and Y. Fang, "Newton-Thiele's rational interpolants," *Numerical Algorithms*, vol. 24, no. 1/2, pp. 141–157, 2000.
[14] J. Tan and S. Tang, "Composite schemes for multivariate blending rational interpolation," *Journal of Computational and Applied Mathematics*, vol. 144, no. 1-2, pp. 263–275, 2002.
[15] Q.-J. Zhao and J. Tan, "Block-based Thiele-like blending rational interpolation," *Journal of Computational and Applied Mathematics*, vol. 195, no. 1-2, pp. 312–325, 2006.
[16] Q. J. Zhao and J. Q. Tan, "Block-based Newton-type blending rational interpolation," *Journal of Computational Mathematics*, vol. 24, pp. 515–526, 2006.
[17] S. Tang and Y. Liang, "Bivariate blending Thiele-Werner's osculatory rational interpolation," *Numerical Mathematics: A Journal of Chinese Universities (English Series)*, vol. 3, pp. 271–288, 2007.
[18] N. Dyn and M. S. Floater, "Multivariate polynomial interpolation on lower sets," *Journal of Approximation Theory*, vol. 177, pp. 34–42, 2014.
[19] A. Cuyt and O. Salazar Celis, "Multivariate data fitting with error control," *BIT Numerical Mathematics*, vol. 59, no. 1, pp. 35–55, 2019.
[20] L. He, J. Tan, Y. Xing et al., "Super-resolution reconstruction based on Continued Fractions interpolation kernel in the polar coordinates," *Journal of Electronic Imaging*, vol. 27, Article ID 043035, 2018.
[21] S. A. Karim and A. Saaban, "Shape preserving interpolation using rational cubic ball function and its application in image interpolation," *Mathematical Problems in Engineering*, vol. 2017, Article ID 7459218, 13 pages, 2017.
[22] Y. Zhan, S. J. Li, and M. Li, "Local and nonlocal regularization to image interpolation," *Mathematical Problems in Engineering*, vol. 2014, Article ID 230348, 9 pages, 2014.
[23] R. Behl, A. J. Alsolami, B. A. Pansera, W. M. Al-Hamdan, M. Salimi, and M. Ferrara, "A new optimal family of schröder's method for multiple zeros," *Mathematics*, vol. 7, no. 11, p. 1076, 2019.
[24] L. He, Y. Xing, K. Xia et al., "An adaptive image inpainting method based on Continued Fractions interpolation," *Discrete Dynamics in Nature and Society*, vol. 2018, Article ID 9801361, 17 pages, 2018.
[25] Y. Zheng, M. Zhao, P. Sun et al., "Optimization of electrostatic force system based on Newton interpolation method," *Journal of Sensors*, vol. 2018, Article ID 7801597, 8 pages, 2018.
[26] H. L. Minh, T. T. H. Hanh, and D. N. H. Thanh, "Monotone finite-difference schemes with second order approximation based on regularization approach for the dirichlet boundary problem of the gamma equation," *IEEE Access*, vol. 8, pp. 45119–45132, 2020.
[27] A. Pourmoslemi, M. Ferrara, B. A. Pansera, and M. Salimi, "Probabilistic norms on the homeomorphisms of a group," *Soft Computing*, vol. 24, no. 10, pp. 7021–7028, 2020.
[28] Q. Duan, L. Wang, and E. H. Twizell, "A new bivariate rational interpolation based on function values," *Information Sciences*, vol. 166, no. 1-4, pp. 181–191, 2004.
[29] Y. Zhang, Q. Duan, F. Bao, and C. Zhang, "A rational interpolation surface model and visualization constraint," *Scientia Sinica Mathematica*, vol. 44, no. 7, pp. 729–740, 2014.
[30] Y. F. Zhang, F. X. Bao, M. Zhang et al., "A weighted bivariate blending rational interpolation function and visualization control," *Journal Computational Analysis and Applications*, vol. 14, no. 7, pp. 1303–1320, 2012.
[31] Q. Duan, Y. Zhang, and E. H. Twizell, "A bivariate rational interpolation and the properties," *Applied Mathematics and Computation*, vol. 179, no. 1, pp. 190–199, 2006.
[32] Q. Duan, L. Wang, and E. H. Twizell, "A new C2 rational interpolation based on function values and constrained control of the interpolant curves," *Applied Mathematics and Computation*, vol. 161, no. 1, pp. 311–322, 2005.
[33] K. M. Reddy and A. K. B. Chand, "Constrained univariate and bivariate rational fractal interpolation," *International Journal for Computational Methods in Engineering Science and Mechanics*, vol. 20, no. 5, pp. 404–422, 2019.
[34] W.-X. Huang and G.-J. Wang, "A weighted bivariate blending rational interpolation based on function values," *Applied Mathematics and Computation*, vol. 217, no. 9, pp. 4644–4653, 2011.
[35] L. Zou and S. Tang, "A new approach to general interpolation formulae for bivariate interpolation," *Abstract and Applied Analysis*, vol. 2014, Article ID 421635, 11 pages, 2014.
[36] L. Zou, L. Song, X. Wang et al., "Bivariate Thiele-Like rational interpolation Continued Fractions with parameters based on virtual points," *Mathematics*, vol. 8, no. 71, 2020.