Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

New Recursive Divided-Difference Interpolation Method

A.N. Samijonov
Waseda University
N.S. Mamatov
n.mamatov@tiiame.uz
Tashkent Institute of Irrigation and Agricultural Mechanization Engineers
B.N. Samijonov
Sejong University
N.A. Niyozmatova
Tashkent University of Information Technology

Research Article

Keywords: polynomial interpolation, divided differences, recursive algorithm, coefficient, determination, Newton interpolation, monomial basis

Posted Date: October 20th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11118244/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

New Recursive Divided-Difference Interpolation Method

A. N. Samijonov[1], N. S. Mamatov[2,3*], B. N. Samijonov[4], N. A. Niyozmatova[3,2]

[1]Graduate School of Information, Production and Systems, Waseda University, 2-7 Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka 808-0135, Japan.

[2]Department of Digital Technologies and Artificial Intelligence, Tashkent Institute of Irrigation and Agricultural Mechanization Engineers National Research University, 39 Kari Niyazov Avenue, Tashkent 100000, Uzbekistan.

[3]Department of Software Engineering, Tashkent University of Information Technologies named after Muhammad al-Khwarizmi, 108 Amir Temur Avenue, Tashkent 100200, Uzbekistan.

[4]Department of Computer Science and Engineering, Sejong University, 209 Neungdong-ro, Gwangjin-gu, Seoul 05006, South Korea.

*Corresponding author(s). E-mail(s): n.mamatov@tiiame.uz;
Contributing authors: abu@fuji.waseda.jp; samijonov.bn@gmail.com;
n.niyozmatova@tiiame.uz;

Abstract

This article is devoted to the direct determination of the coefficients of an interpolation polynomial in the monomial basis, proposing a new recursive method based on divided differences. The proposed method does not require an explicit expansion of the Newton interpolation polynomial into the monomial basis, nor does it require constructing and solving a Vandermonde system of linear equations. The method is based on the sequential recursive reconstruction of the divided-differences table. When a column of divided differences yields a constant value, this value is determined as the corresponding coefficient of the polynomial. The contributions of the coefficients found so far are then subtracted from the interpolation values, and the same process is repeated for the remaining

lower-degree polynomial. As a result, all coefficients of the polynomial are determined sequentially, starting from the highest-degree term down to the constant term. The mathematical foundations of the proposed algorithm are rigorously proven through lemmas and theorems. In particular, the necessary and sufficient condition for the common value obtained in a column of recursive divided differences to be equal to the corresponding coefficient of the polynomial is established. The numerical example provided confirms the correctness of the proposed algorithm and demonstrates that the polynomial coefficients can be recursively determined with high accuracy based on the interpolation data. The proposed approach offers a new computational interpretation of divided differences and forms an alternative recursive computational scheme for determining polynomial coefficients.

Keywords: polynomial interpolation, divided differences, recursive algorithm, coefficient determination, Newton interpolation, monomial basis

1 Introduction

Polynomial interpolation is one of the fundamental directions of numerical analysis and computational mathematics. This problem consists of constructing a polynomial that passes through each of a given set of mutually distinct node points and their corresponding function values. Due to its theoretical significance and broad practical applicability, polynomial interpolation has become an important and indispensable tool in numerous fields of modern computational science, including approximation theory, numerical differentiation and integration, numerical solution of differential equations, spectral methods, computer-aided geometric design, computer graphics, signal processing, data assimilation, numerical uncertainty estimation, and machine learning [1, 6, 7, 15, 24, 26, 27, 30].

The rapid development of scientific computing and numerical modeling has further increased the importance of interpolation. Modern numerical algorithms require repeated reconstruction, evaluation, differentiation, and integration of interpolation polynomials, as well as their transformation between different representations. This highlights the necessity of paying close attention not only to improving interpolation accuracy, but also to reducing computational complexity, enhancing numerical stability, and developing efficient recursive computational procedures [1, 6, 15, 18, 24, 26, 27, 30, 31].

Polynomial interpolation can be expressed in several equivalent forms. Among the most well-known methods are Lagrange, Newton, Hermite, barycentric, spline, and rational interpolation [1, 3, 6, 8, 14, 15, 25]. Each representation possesses its own computational advantages depending on the specific characteristics of the numerical problem being solved.

The Lagrange interpolation formula provides an explicit analytical form of the interpolation polynomial and occupies an important place in approximation theory. However, in this method, when a new interpolation node is added, the entire set of basis polynomials must be reconstructed. For this reason, the Lagrange method

is computationally inefficient in adaptive interpolation problems where interpolation nodes are added sequentially [1, 6]. The Newton interpolation formula, in contrast, constructs the interpolation polynomial recursively through divided differences. The recursive nature of Newton interpolation makes it possible to add new interpolation nodes without the need to recompute the previously determined coefficients. This has made it one of the most widely used interpolation methods in practical numerical computations [1, 6, 23]. This property of the Newton divided-differences method has turned it into a core computational mechanism in numerical analysis, and it has been implemented in numerous scientific computing software libraries.

Suppose the following interpolation nodes are given

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n),$$

where $x_i \neq x_j, i \neq j$.

The classical Newton interpolation polynomial is expressed as follows:

$$P_m(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \ldots + a_m \prod_{i=0}^{m-1} (x - x_i),$$

where $a_i = f[x_0, x_1, \ldots, x_i]$ are the coefficients, obtained from the divided-differences table [1, 6, 23]. This representation is convenient from a computational point of view, since each additional interpolation node affects only one new coefficient, while the previously computed coefficients remain unchanged.

Although the Newton form is highly efficient for constructing interpolation polynomials, many numerical algorithms require the polynomial to be expressed in the standard monomial basis:

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0. \tag{1.1}$$

The monomial form naturally arises in numerous engineering problems, such as symbolic computation, polynomial differentiation, indefinite and definite integration, polynomial optimization, solving nonlinear equations, polynomial algebra, finite element methods, and computer algebra systems [7, 27, 30, 32]. For this reason, converting the Newton form into the monomial form is an important computational problem, and its efficiency directly affects the performance of many numerical algorithms.

1.1 Existing methods for computing polynomial coefficients

Although interpolation theory has been extensively studied for over two centuries, the problem of efficiently computing polynomial coefficients in the monomial basis remains a relevant research topic to this day. Existing methods can be broadly classified into four groups: Newton divided-differences methods, Vandermonde system-based approaches, basis-conversion algorithms, and barycentric or rational interpolation methods.

The Newton interpolation method based on divided differences remains one of the most successful recursive interpolation algorithms due to its simple implementation,

step-by-step construction, and moderate computational cost [6, 22, 23]. The interpolation polynomial is constructed recursively by successively adding interpolation nodes, while the divided-differences table provides an efficient mechanism for determining the corresponding Newton coefficients. For this reason, Newton interpolation continues to be used as a standard tool in numerical analysis textbooks and scientific software libraries [1, 6].

Despite these advantages, however, the coefficients obtained through the Newton algorithm belong to the Newton polynomial basis rather than the standard monomial basis. In practical scientific computations, however, the coefficients of the polynomial in form Eq. (1.1) are frequently required. This necessitates an additional computational step to convert the Newton form into the monomial form. The traditional conversion process is purely algebraic in nature: each Newton basis polynomial is successively expanded, after which the coefficients corresponding to the same powers of x are collected [1, 6]. Although this approach is mathematically simple, it requires repeated polynomial multiplications and rearrangement of coefficients. As the degree of the polynomial increases, both the number of arithmetic operations and the implementation complexity grow considerably. Another widely used approach consists of directly determining the monomial coefficients by solving a Vandermonde system of linear equations [4, 18], namely:

$$Va = y,$$

where V is the Vandermonde matrix, a is the vector of polynomial coefficients, and y is the interpolation data.

Numerous algorithms have been proposed for solving Vandermonde systems more efficiently than general linear systems [9, 16, 17, 33]. These include specialized elimination procedures, recursive algorithms, Björck–Pereyra factorization, methods based on shift structure, and fast structured-matrix methods. The numerical properties of Vandermonde matrices have been extensively studied for several decades, and it has now been established that, for many node distributions, their condition number grows exponentially with increasing polynomial degree [4, 17, 18]. As a result, computing coefficients by directly solving Vandermonde systems can become numerically unstable for high-degree interpolation problems.

In order to overcome these problems, considerable attention has been devoted to alternative interpolation representations. Among these, barycentric interpolation has become one of the most influential developments in modern interpolation theory, owing to its high numerical stability and computational efficiency [3, 5]. The barycentric form avoids the direct construction of the interpolation polynomial and provides highly accurate evaluation even at high interpolation degrees. Subsequently, a number of improvements and extensions have been proposed, including Floater–Hormann rational interpolation, adaptive barycentric formulas, and stable recursive evaluation schemes [2, 13, 19, 28].

Although barycentric algorithms significantly improve interpolation evaluation, they are not designed to determine the coefficients of the monomial form, since their primary purpose is to compute the polynomial value at arbitrary points with high accuracy.

Another active research direction is devoted to developing basis-conversion algorithms between different polynomial representations. Numerous studies have examined conversions between Newton, Bernstein, Chebyshev, Legendre, and monomial bases [11, 12, 21, 29]. Such algorithms are of considerable importance in computer-aided geometric design, approximation theory, spectral methods, and the computation of orthogonal polynomials. Nevertheless, most existing basis-conversion methods rely on explicit conversion matrices, recursive coefficient-transformation formulas, or repeated polynomial operations, which increases computational complexity for high-degree polynomials.

Recent advances in numerical analysis have also focused on improving computational efficiency through recursive algorithms, structured matrices, hierarchical polynomial representations, and numerically stable coefficient-computation methods [10, 16, 17, 20, 21, 29]. Despite these advances, most existing methods still rely on explicit basis conversion or the direct solution of a linear system.

According to the existing literature, a recursive divided-differences algorithm that determines the monomial basis coefficients through successive reduction of the interpolation data — without explicitly expanding the Newton polynomial and without constructing or solving a Vandermonde system — has not yet been proposed. This constitutes the main scientific problem addressed in this work.

1.2 Research gap and problem statement

The analysis presented above shows that existing interpolation methods are primarily directed toward three goals: constructing interpolation polynomials, improving numerical stability, and simplifying and accelerating the computation of polynomial values. Newton interpolation provides an efficient recursive form based on divided differences, while barycentric formulations improve numerical stability in evaluating the interpolation polynomial. Vandermonde-based approaches determine the monomial coefficients directly through linear systems, whereas basis-conversion algorithms transform the polynomial from one representation into another.

Despite these important achievements, determining the polynomial coefficients in the monomial basis remains computationally challenging. Existing approaches generally follow one of two strategies. In the first, the interpolation polynomial is initially constructed in the Newton basis, after which each Newton basis polynomial is expanded in powers of x. In the second, the coefficient vector is determined by solving a Vandermonde system of linear equations. Although these approaches are mathematically equivalent, in both cases the computational complexity increases as the interpolation degree grows.

Importantly, both strategies separate the construction of the interpolation polynomial from the computation of its coefficients. The divided-differences table is used only to generate the Newton coefficients, while determining the monomial coefficients requires an additional, independent computational step. As a result, the recursive information already present in the divided-differences table is not fully utilized.

From an algorithmic point of view, this situation gives rise to the following fundamental question.

Can the coefficients of the polynomial in the monomial basis be computed directly and recursively from divided differences, without expanding the Newton polynomial and without solving a Vandermonde system?

To the best of our knowledge, no recursive algorithm exists in the literature that solves this problem through successive reduction of the interpolation data. Current divided-differences algorithms terminate once the Newton coefficients have been determined, while the determination of the monomial coefficients is treated as a separate algebraic transformation or an independent linear algebra problem.

This work aims to fill this gap by proposing a recursive reduction procedure that extends the divided-differences algorithm beyond the computation of Newton coefficients. In the proposed approach, instead of converting the Newton polynomial into the monomial basis, the interpolation data are iteratively transformed after each higher-degree coefficient is determined. At each iteration, the contribution of the highest-degree monomial term computed so far is subtracted from the interpolation values, thereby reducing the degree of the remaining polynomial by one. The divided-differences table is then reconstructed for the reduced interpolation data, which makes it possible to recursively determine the next coefficient.

As a result, the entire coefficient vector

$$(a_0, a_1, \ldots, a_n),$$

is determined not by explicitly expanding the polynomial or solving an auxiliary linear system, but through the successive reduction of the interpolation table.

This recursive approach fundamentally changes the interpretation of divided-differences computation. Divided differences now serve not merely as the coefficients of the Newton interpolation polynomial, but also as intermediate computational quantities in the step-by-step determination of the polynomial's coefficients in the standard monomial representation. In this way, the proposed approach establishes a direct computational link between recursive divided differences and the coefficients of the polynomial in the monomial basis. This link constitutes the central idea of the theoretical approach developed in this article and is progressively elaborated in the sections that follow.

1.3 Main results

The main objective of this work is to develop a recursive computational approach that directly determines the coefficients of the polynomial in the monomial basis from the interpolation data, without explicitly expanding the Newton interpolation polynomial and without constructing or solving a Vandermonde system of linear equations.

To achieve this goal, a new recursive algorithm based on divided differences is proposed. Unlike the classical Newton interpolation method, where divided differences are used solely to construct the Newton interpolation polynomial, the proposed method extends the application of divided differences to an iterative process of coefficient reconstruction. After each coefficient is determined, the algorithm recursively transforms the interpolation data, so that the degree of the polynomial is reduced by one at each iteration.

The main results of this article can be summarized as follows.

A new recursive method based on divided differences has been proposed for the iterative computation of polynomial coefficients. The proposed method preserves the recursive nature of the classical Newton algorithm, while making it possible to successively reduce the interpolation data.

A recursive coefficient-reduction strategy has been developed, based on first determining the highest-degree coefficient and then subtracting its contribution from the interpolation values. This approach transforms the original interpolation problem into a sequence of interpolation problems of progressively decreasing degree.

A cyclic algorithm is proposed for computing the coefficients of polynomial Eq. (1.1), without expanding the Newton basis polynomials into the monomial basis.

The proposed algorithm avoids constructing and solving a Vandermonde linear system. As a result, the determination of the polynomial coefficients is carried out solely through recursive operations performed on divided differences.

The mathematical validity of the proposed recursive algorithm has been proven through a sequence of lemmas and theorems. The theoretical analysis shows that each recursive reduction step preserves the interpolation property and correctly determines the next polynomial coefficient.

The computational complexity of the proposed algorithm has been analyzed and compared with traditional coefficient-computation methods based on the explicit expansion of the Newton polynomial and on solving a Vandermonde system.

Numerical experiments have confirmed the correctness of the proposed method and demonstrated its computational efficiency for interpolation polynomials of various degrees.

The proposed recursive reduction algorithm provides an alternative computational interpretation of divided differences. Instead of being regarded merely as the coefficients of the Newton interpolation polynomial, divided differences are employed as intermediate recursive quantities for reconstructing the coefficients of the polynomial in the standard monomial basis. This perspective establishes a direct recursive relationship between divided differences and the computation of polynomial coefficients, which constitutes the main theoretical result of this work.

1.4 Structure of the article

The remainder of the article is organized as follows. Section 2 introduces the notation, basic definitions, and the construction of recursive divided differences used throughout the article. Section 3 presents the mathematical foundations of the proposed method for computing polynomial coefficients and develops the algorithm for the recursive determination of these coefficients. Section 4 provides numerical examples confirming the correctness and practical efficiency of the proposed algorithm. Finally, section 5 presents the main conclusions and outlines promising directions for future research.

This article does not consider the analysis of computational complexity or its comparative evaluation against existing methods. These aspects will be studied in detail in future research devoted to the computational efficiency and numerical stability of the proposed algorithm.

2 Preliminaries and notation

This section presents the notation and mathematical definitions required throughout the article. Divided differences are defined recursively in a form consistent with the proposed algorithm. Based on this definition, the divided-differences table is then constructed. This table forms the computational foundation for the theoretical results and the proposed algorithm presented in the subsequent sections.

2.1 Polynomial interpolation

Suppose that pairwise distinct interpolation nodes $x_0, x_1, \ldots, x_n$ are given, satisfying

$$x_i \neq x_j, i \neq j \tag{2.1}$$

and

$$y_i = f(x_i), i = \overline{0,n}. \tag{2.2}$$

Then there exists a unique polynomial

$$P_n(x) \in \Pi_n,$$

of degree not exceeding n, satisfying the condition

$$P_n(x_i) = y_i, i = \overline{0,n}, \tag{2.3}$$

where $\Pi_n = \{P(x) : \deg(P) \leq n\}$ denotes the space of all polynomials of degree not exceeding n.

Throughout this article, the polynomial is expressed in the monomial basis by means of Eq. (1.1).

Unlike the classical Newton interpolation algorithm, the objective of this work is to directly determine the coefficients $a_0, a_1, \ldots, a_n$ without expanding the Newton basis polynomials and without solving a Vandermonde linear system.

2.2 Recursive divided differences

The zeroth-order divided differences are taken as follows:

$$\Delta_i^{(0)} = y_i = f(x_i), \quad (i = \overline{0,n};) \tag{2.4}$$

Furthermore,

$$\Delta x_i^{(j)} = x_i - x_j, i, j = \overline{1,n}, i \neq j \tag{2.5}$$

The first-order divided differences corresponding to the nodes (x_i, x_0) are computed as follows:

$$\Delta_i^{(1)} = \frac{\Delta_i^{(0)} - \Delta_0^{(0)}}{\Delta x_i^{(0)}} = \frac{y_i - y_0}{x_i - x_0}, \quad (i = \overline{1,n}) \tag{2.6}$$

Similarly to the first-order divided differences, the second-order divided differences corresponding to the nodes (x_i, x_1) are computed as follows:

$$\Delta_{i-1}^{(2)} = \frac{\Delta_i^{(1)} - \Delta_1^{(1)}}{\Delta x_i^{(1)}}, \quad (i = \overline{2,n}) \tag{2.7}$$

The third-order divided differences corresponding to the nodes (x_i, x_2) are computed as follows:

$$\Delta_{i-2}^{(3)} = \frac{\Delta_{i-1}^{(2)} - \Delta_1^{(2)}}{\Delta x_i^{(2)}}, \quad (i = \overline{3,n}) \tag{2.8}$$

In general, the k-th order divided differences are computed recursively from the $k-1$-th order divided differences as follows:

$$\Delta_{i-k+1}^{(k)} = \frac{\Delta_{i-k+2}^{(k-1)} - \Delta_1^{(k-1)}}{\Delta x_i^{(k-1)}}, \quad (i = \overline{k,n}) \tag{2.9}$$

Equation (2.9) forms the basis of the recursive construction adopted throughout this article.

2.3 Divided-differences table

For convenience, the recursively computed divided differences are arranged in the form of a triangular table; see Table 1.

Table 1 Recursive divided-differences table.

N	0	1	2	3	$\cdots$	$k-1$	k	$\cdots$	n
x	x_0	x_1	x_2	x_3	$\cdots$	x_{k-1}	x_k	$\cdots$	x_n
$f(x)$	y_0	y_1	y_2	y_3	$\cdots$	y_{k-1}	y_k	$\cdots$	y_n
$\Delta^{(0)}$	$\Delta_0^{(0)}$	$\Delta_1^{(0)}$	$\Delta_2^{(0)}$	$\Delta_3^{(0)}$	$\cdots$	$\Delta_{k-1}^{(0)}$	$\Delta_k^{(0)}$	$\cdots$	$\Delta_{n+1}^{(0)}$
$\Delta^{(1)}$	0	$\Delta_1^{(1)}$	$\Delta_2^{(1)}$	$\Delta_3^{(1)}$	$\cdots$	$\Delta_{k-1}^{(1)}$	$\Delta_k^{(1)}$	$\cdots$	$\Delta_n^{(1)}$
$\Delta^{(2)}$	0	0	$\Delta_1^{(2)}$	$\Delta_2^{(2)}$	$\cdots$	$\Delta_{k-1}^{(2)}$	$\Delta_k^{(2)}$	$\cdots$	$\Delta_{n-1}^{(3)}$
$\Delta^{(3)}$	0	0	0	$\Delta_1^{(3)}$	$\cdots$	$\Delta_{k-1}^{(3)}$	$\Delta_k^{(3)}$	$\cdots$	$\Delta_{n-2}^{(3)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
$\Delta^{(k)}$	0	0	0	0	$\cdots$	0	$\Delta_1^{(k)}$	$\cdots$	$\Delta_{n-k+1}^{(k)}$

Unlike the classical Newton divided-differences table, where only the first element of each column is used to construct the interpolation polynomial, in this work all elements of each column are retained, since each of them participates in the recursive determination of the polynomial's coefficients in the monomial basis.

2.4 Computational principle

The proposed method is based on examining each column of the recursively constructed divided-differences table.

The process begins with the first-order divided differences, and the elements of the columns are computed successively according to Eq. (2.9). If the elements of the current column are not identical, the divided differences of the next order are computed. The recursive process continues until a column is reached that satisfies the condition

$$\Delta_1^{(k)} = \Delta_2^{(k)} = \ldots = \Delta_{n-k+1}^{(k)} \tag{2.10}$$

The formation of a row consisting of identical values constitutes the main computational step of the proposed algorithm. Unlike the classical Newton interpolation method, where only the first divided difference of each column is used to construct the interpolation polynomial, the proposed approach makes use of all elements of this recursively obtained column.

The mathematical essence of condition Eq. (2.10) and its relationship to the coefficients of the interpolation polynomial will be theoretically justified in the next section.

2.5 General description of the algorithm for recursive determination of coefficients

The recursive algorithm developed in this article is based on sequentially determining the coefficients of the polynomial, starting from the highest-degree coefficient and proceeding down to the constant term. After each coefficient is determined, the divided-differences table is reconstructed in accordance with the proposed recursive procedure, and the same computational process is repeated for the remaining coefficients. This process continues until all coefficients of the interpolation polynomial have been fully determined.

3 Mathematical foundations of the proposed method

This section develops the theoretical foundation of the proposed recursive algorithm. First, the representation of the interpolation polynomial in terms of the recursive divided differences introduced in section 2 is presented. This representation subsequently serves as the theoretical basis for the main theorem on determining the polynomial coefficients and for justifying the proposed recursive algorithm.

3.1 Form of the polynomial

Lemma 3.1

$$P_m(x) = \sum_{i=1}^{m} \left(\Delta_1^{(i)} \prod_{j=0}^{i-1} \Delta x^{(j)} \right) + \Delta_0^{(0)} \tag{3.1}$$

Proof The proof is carried out by the method of mathematical induction.

When $x = x_0$, since $x - x_0$ appears in each term of the product, $\sum_{i=1}^{m} \left(\Delta_1^{(i)} \prod_{j=0}^{i-1} \Delta x_i^{(j)} \right) = 0$ holds. Therefore, $P_m(x_0) = \Delta_0^{(0)} = y_0$.

Let $x = x_1$. Then

$$P_1(x_1) = \Delta_1^{(1)}(x_1 - x_0) + \Delta_0^0 = \frac{y_1 - y_0}{x_1 - x_0}(x_1 - x_0) + \Delta_0^0 = y_1$$

Similarly, for $x = x_2$

$$\begin{aligned} P_2(x_2) &= \Delta_1^{(1)}(x_2 - x_0) + \Delta_1^{(2)}(x_2 - x_0)(x_2 - x_1) + \Delta_0^0 = \\ &= \Delta_1^{(1)}(x_2 - x_0) + \frac{\Delta_2^{(1)} - \Delta_1^{(1)}}{x_2 - x_1}(x_2 - x_0)(x_2 - x_1) + \Delta_0^0 = \\ &= \Delta_1^{(1)}(x_2 - x_0) + \left(\Delta_2^{(1)} - \Delta_1^{(1)} \right)(x_2 - x_0) + \Delta_0^0 = \\ &= \Delta_2^{(1)}(x_2 - x_0) + \Delta_0^0 = \frac{y_2 - y_0}{x_2 - x_0}(x_2 - x_0) + \Delta_0^0 = y_2 \end{aligned}$$

Now suppose that $P_k(x_k) = y_k$.

It remains to verify the interpolation property at the next node $x = x_{k+1}$. According to Eq. (3.1),

$$P_k(x_{k+1}) = \sum_{i=1}^{k} \left(\Delta_1^{(i)} \prod_{j=0}^{i-1} \Delta x_i^{(j)} \right) + \Delta_0^{(0)} = y_k + \Delta_0^{(0)} + \Delta_1^{k+1} \prod_{j=0}^{k} \Delta x_{k+1}^{(j)}$$

Using the recursive definition of divided differences,

$$\Delta_1^{(k+1)} = \frac{\Delta_2^{(k)} - \Delta_1^{(k)}}{\Delta x_{k+1}^k}$$

we obtain the following:

$$\begin{aligned} \Delta_1^{k+1} \prod_{j=0}^{k} \Delta x_{k+1}^{(j)} &= \frac{\Delta_2^{(k)} - \Delta_1^{(k)}}{\Delta x_{k+1}^k} \cdot \prod_{j=0}^{k} \Delta x_{k+1}^{(j)} = \left(\Delta_2^{(k)} - \Delta_1^{(k)} \right) \prod_{j=0}^{k} \Delta x_{k+1}^{(j)} = \\ &= \Delta_2^{(k)} \prod_{j=0}^{k} \Delta x_{k+1}^{(j)} - y_k = y_{k+1} - y_0 - y_k. \end{aligned}$$

As a result,

$$P_{k+1}(x_{k+1}) = \sum_{i=1}^{m} \left(\Delta_1^{(i)} \prod_{j=0}^{i-1} \Delta x_i^{(j)} \right) + \Delta_0^{(0)} = y_k + \Delta_0^{(0)} + y_{k+1} - y_0 - y_k = y_{k+1}$$

Thus, $P_m(x_i) = y_i, i = \overline{0, m}$, which proves that polynomial Eq. (3.1) interpolates at all the given nodes. Hence, the assertion of the lemma is proved. □

3.2 Determination of the polynomial coefficients

In Lemma 3.1, the representation of the interpolation polynomial in terms of recursively generated divided differences was presented. This representation serves as the initial theoretical basis for determining the coefficients of the polynomial in the monomial basis.

The proposed approach is based on the recursive construction presented in section 2. When condition Eq. (2.10) is satisfied, all elements of the corresponding column of divided differences take on a constant value. The following theorem shows that this invariant value uniquely determines the corresponding coefficient of the interpolation polynomial.

Theorem 3.2 *In order for* $a_k = \Delta_1^{(k)}$ *to hold, it is necessary and sufficient that* $\Delta_1^{(k)} = \Delta_j^{(k)}$, $j = \overline{1, n-k}$.

Proof Necessity. Suppose that $\Delta_1^{(k)} = \Delta_j^{(k)}$, $j = \overline{1, n-k}$.

For all given nodes, the following equality holds:

$$y_i = P_i(x) = a_i x_i^i + a_{i-1} x_i^{i-1} + \ldots + a_1 x_i + a_0, i = \overline{1, k}; \tag{3.2}$$

We write Eq. (3.2) in the form Eq. (3.3).

$$y_i - a_i x_i^i = a_{i-1} x_i^{i-1} + \ldots + a_1 x_i + a_0, i = \overline{1, k}; \tag{3.3}$$

If we let $P_{i-1}(x_i) = a_{i-1} x_i^{i-1} + \ldots + a_1 x_i + a_0$, then $y_i - a_i x_i^i = P_{(i-1)}(x)$ and, by Lemma 3.1, we obtain $y_i = \Delta_j^{(k)} x_i^i + P_{(i-1)}(x)$. From this, we obtain the following:

$$y_i - a_k x_i^k = y_i - \Delta_j^{(k)} x_i^k \tag{3.4}$$

holds. Since $\Delta_i^{(k)} = \Delta_j^{(k)}$, it follows that $a_i = \Delta_1^{(k)}$.

Sufficiency. Suppose that $\Delta_1^{(k)} \neq \Delta_j^{(k)}$, $j = \overline{1, n-k}$, then from equality Eq. (3.4) it follows that $a_i \neq \Delta_1^{(k)}$.

The theorem is proved. □

3.3 Corollaries and recursive determination of the coefficients

From the lemmas and theorems above, the following corollaries can be obtained.

Corollary 3.3 *If the highest-order nonzero divided difference is of order* m*, then interpolation polynomial Eq. (1.1) is a polynomial of degree* m.

Corollary 3.4 *The divided difference of the algebraic sum of two functions is equal to the algebraic sum of their corresponding divided differences.*

Corollary 3.5 *A constant factor can be taken outside a divided difference, that is, the divided-difference operator is linear with respect to scalar multiplication.*

Corollary 3.6 *A divided difference is a symmetric function of its interpolation nodes. Thus, the value of a divided difference remains unchanged under an arbitrary permutation of its* $x_0, x_1, \ldots, x_n$ *arguments.*

The proofs of these corollaries follow directly from the properties of recursively obtained divided differences, and therefore they are not presented.

Based on Lemma 3.1 and Theorem 3.2, the coefficients of polynomial Eq. (1.1) can be recursively reconstructed without expanding the interpolation polynomial into the monomial basis. The reconstruction process begins with the highest-degree coefficient and proceeds sequentially toward the constant term.

After $a_k = \Delta_1^{(k)}$ is determined, the zeroth-order divided differences are updated according to Eq. (3.5):

$$\Delta_i^{(0)} = y_i - \Delta_1^{(k)} x_i^k, \forall i, i \in \{1, 2, \ldots, n\}; \tag{3.5}$$

This substitution removes the contribution of the highest-degree term from the interpolation data, while the structure of the remaining polynomial is preserved. As a result, the updated data correspond to a polynomial of degree $k-1$.

Next, a new recursive divided-differences table is constructed from the modified zeroth-order row. By applying Theorem 3.2 to the new table, the next coefficient is obtained as follows:

$$a_{k-1} = \Delta_1^{(k-1)} \tag{3.6}$$

By repeating this same recursive process, the remaining

$$a_{k-1}, a_{k-2}, \ldots, a_0$$

coefficients are obtained sequentially.

Finally

$$a_1 = \Delta_1^{(1)} \tag{3.7}$$

while the constant coefficient is computed as follows:

$$a_0 = y_i - \Delta_1^{(1)} x_i, \forall i, i \in \{1, 2, \ldots, n\}; \tag{3.8}$$

Thus, all coefficients of polynomial Eq. (1.1) are recursively reconstructed by repeatedly updating the zeroth-order divided-differences row. The proposed strategy determines the coefficients directly in the monomial basis, without requiring the expansion of the Newton interpolation polynomial.

3.4 Iterative algorithm for reconstructing the coefficients

Based on the theoretical results obtained in Lemma 3.1 and Theorem 3.2, the proposed iterative algorithm determines the coefficients of the interpolation polynomial sequentially. For this purpose, at each step the recursive divided-differences table is reconstructed, and the contribution of the determined highest-degree term to the interpolation values is removed. As a result, the degree of the remaining polynomial decreases by one, and this same recursive computational process is repeated until all coefficients have been fully determined.

The algorithm terminates after n recursive steps, each step determining exactly one coefficient of the interpolation polynomial. Unlike the classical Newton interpolation

Algorithm 3.1 Iterative reconstruction of the polynomial coefficients.

Require: interpolation nodes $\{(x_i, y_i)\}_{i=0}^{n}$, the recursively constructed divided-differences table.

Ensure: the polynomial coefficients $a_n, a_{n-1}, \ldots, a_0$.

1: The recursive divided-differences table is constructed according to Eq. (2.9).
2: Starting from the highest available order, the first divided-differences column satisfying condition Eq. (2.10) is identified.
3: The corresponding polynomial coefficient $a_k = \Delta_1^{(k)}$ is computed.
4: The zeroth-order divided differences $\Delta_i^{(0)} = y_i - \Delta_1^{(k)} x_i^k, \forall i, i \in \{1, 2, \ldots, n\}$ are updated.
5: A new recursive divided-differences table is constructed from the updated values.
6: Steps 2–5 are repeated until the linear coefficient $a_1 = \Delta_1^{(1)}$ is obtained.
7: The constant coefficient $a_0 = y_i - \Delta_1^{(1)} x_i, \forall i, i \in \{1, 2, \ldots, n\}$ is computed.
8: **return** $a_n, a_{n-1}, \ldots, a_0$.

process, the proposed algorithm computes the coefficients directly in the monomial basis, without expanding the Newton form of the interpolation polynomial. As a result, the recursive divided-differences table simultaneously serves both as the interpolation structure and as the mechanism for reconstructing the coefficients.

4 Numerical examples

These experiments serve three main purposes: first, to confirm the correctness of the theoretical results obtained in section 3; second, to show step by step the process of recursively determining the coefficients; third, to demonstrate that the coefficients of the interpolation polynomial can be determined directly, without explicitly expanding the Newton interpolation formula into the monomial basis.

Unless otherwise noted, all computations are performed using double-precision floating-point (real-number) arithmetic. The reconstructed coefficients are compared with the exact coefficients of the polynomial in order to verify the correctness of the proposed method.

The examples are arranged in order of increasing complexity. The first example shows the complete recursive reconstruction process of the coefficients for a low-degree polynomial. In the subsequent examples, higher-degree polynomials are considered, demonstrating the stability and computational efficiency of the proposed method.

Example 4.1 Consider the polynomial $y = 2x^3 + 3x^2 + 5x + 7$.

Let the interpolation nodes be $x = (0.5, 10, 17, 0.7, 13.1)$. The corresponding function values are given in Table 2.

The table of recursive divided differences is constructed in accordance with section 2. The resulting tables for the successive recursive steps are given in Table 3.

Table 2 Interpolation data for Example 4.1.

x_i	0.5	10	17	0.7	13.1
y_i	10.5	2357	10785	12.656	5083.512

(a)

Table 3 Successive recursive divided-differences tables for Example 4.1: (a) initial data; (b) after removing the contribution of $a_3 = 2$; (c) after removing the contribution of $a_2 = 3$; (d) after removing the contribution of $a_1 = 5$.

x_i	0.5	10	17	0.7	13.1
$\Delta^{(0)}$	10.5	2357	10785	12.656	5083.512
$\Delta^{(1)}$	0	247	653	10.78	402.62
$\Delta^{(2)}$	0	0	58	25.4	50.2
$\Delta^{(3)}$	0	0	0	2	2

(b)

$\Delta^{(0)}$	10.25	357	959	11.97	587.33
$\Delta^{(1)}$	0	36.5	57.5	8.6	45.8
$\Delta^{(2)}$	0	0	3	3	3

(c)

$\Delta^{(0)}$	9.5	57	92	10.5	72.5
$\Delta^{(1)}$	0	5	5	5	5

(d)

a_0	7	7	7	7	7

From the analysis of the table of recursive divided differences, it can be seen that condition Eq. (2.10) is satisfied for the first time for the third-order divided differences. According to Theorem 3.2, the constant value in this column determines the corresponding coefficient of the polynomial.

Next, the zeroth-order divided differences are updated according to Eq. (3.5). Based on the updated values, a new table of recursive divided differences is constructed, and the process of recursively reconstructing the coefficients is repeated in the same manner for the remaining coefficients of the polynomial.

5 Conclusions

In this article, a new recursive approach based on divided differences, designed for the direct determination of polynomial coefficients, has been proposed. Unlike the

classical Newton interpolation method, the proposed approach makes it possible to directly determine the coefficients of the interpolation polynomial in the monomial basis, without explicitly expanding the Newton interpolation formula and without resorting to constructing and solving a Vandermonde system of linear equations.

The theoretical foundations of the proposed method were developed through the recursive construction of divided differences. Lemma 3.1 presents the representation of the interpolation polynomial in terms of recursive divided differences, while Theorem 3.2 proves the necessary and sufficient condition for the common value in a column of recursively obtained divided differences to be equal to the corresponding coefficient of the polynomial. These results form the mathematical basis of the proposed strategy for the recursive reconstruction of the coefficients.

Based on these theoretical results, an iterative algorithm has been developed that sequentially determines the polynomial coefficients through the recursive updating of the zeroth-order divided differences at each step. The proposed algorithm is based on the principle of determining one coefficient at each recursive step and progressively reducing the degree of the remaining polynomial.

The numerical examples presented confirmed the correctness of the proposed method and demonstrated that all coefficients of the polynomial can be reconstructed with high accuracy based on the interpolation data. The obtained results fully agree with the theoretical analysis.

The proposed recursive approach offers a new computational scheme for determining polynomial coefficients and can serve as a theoretical basis for further research in this direction. In particular, the computational complexity, numerical stability, and efficiency of the algorithm for various interpolation nodes, as well as its comparative analysis with existing methods, are planned to be studied within the scope of future work.

Statements and Declarations

Funding

Funding not applicable.

Competing Interests

The authors declare that they have no competing interests.

Data Availability Statement

All data generated or analysed during this study are included in this article.

References

[1] K. E. Atkinson, *An Introduction to Numerical Analysis*, John Wiley & Sons, New York, 2nd ed., 1989.

[2] J.-P. Berrut, *Rational functions for guaranteed and experimentally well-conditioned global interpolation*, Computers & Mathematics with Applications, 15 (1988), pp. 1–16.

[3] J.-P. Berrut and L. N. Trefethen, *Barycentric Lagrange interpolation*, SIAM Review, 46 (2004), pp. 501–517.

[4] Å. Björck and V. Pereyra, *Solution of Vandermonde systems of equations*, Mathematics of Computation, 24 (1970), pp. 893–903.

[5] L. Bos, S. De Marchi, K. Hormann, and G. Klein, *On the barycentric interpolation formula*, Applied Mathematics and Computation, 218 (2011), pp. 786–790.

[6] R. L. Burden, J. D. Faires, and A. M. Burden, *Numerical Analysis*, Cengage Learning, Boston, 10th ed., 2016.

[7] P. J. Davis, *Interpolation and Approximation*, Dover Publications, New York, 1975.

[8] C. de Boor, *A Practical Guide to Splines*, Springer, New York, revised ed., 2001.

[9] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[10] F. Dubeau, *On Hermite interpolation and divided differences*, Surveys in Mathematics and its Applications, 15 (2020), pp. 257–279.

[11] R. T. Farouki, *The Bernstein polynomial basis: A centennial retrospective*, Computer Aided Geometric Design, 29 (2012), pp. 379–419.

[12] R. T. Farouki and V. Rajan, *Algorithms for polynomials in Bernstein form*, Computer Aided Geometric Design, 5 (1988), pp. 1–26.

[13] M. S. Floater and K. Hormann, *Barycentric rational interpolation*, Numerical Algorithms, 45 (2007), pp. 227–239.

[14] M. S. Floater and K. Hormann, *Barycentric rational interpolation with no poles and high rates of approximation*, Numerische Mathematik, 107 (2007), pp. 315–331.

[15] W. Gautschi, *Numerical Analysis*, Birkhäuser, Boston, 2nd ed., 2012.

[16] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 4th ed., 2013.

[17] N. J. Higham, *Stability analysis of algorithms for solving Vandermonde systems*, SIAM Journal on Matrix Analysis and Applications, 11 (1990), pp. 23–41.

[18] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 2nd ed., 2002.

[19] G. Klein, *An extension of the Floater–Hormann family of barycentric rational interpolants*, Mathematics of Computation, 82 (2013), pp. 2273–2292.

[20] N. Mamatov, N. A. Niyozmatova, A. Samijonov, et al., *Algorithm for determining the coefficients of the Newton interpolation polynomial with separated differences*, IOP Conference Series: Materials Science and Engineering, 862 (2020), p. 042019.

[21] R. D. Neidinger, *Multivariate polynomial interpolation in Newton forms*, SIAM Review, 61 (2019), pp. 361–381.

[22] E. H. Neville, *Iterative interpolation*, Journal of the Indian Mathematical Society, 20 (1934), pp. 87–120.

[23] I. Newton, *Methodus Differentialis*, London, 1711.

[24] G. M. Phillips, *Interpolation and Approximation by Polynomials*, Springer, New York, 2003.

[25] M. J. D. Powell, *Approximation Theory and Methods*, Cambridge University Press, Cambridge, 1981.

[26] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical Mathematics*, Springer, Berlin, 2nd ed., 2007.

[27] T. Sauer, *Numerical Analysis*, Pearson, Boston, 3rd ed., 2018.

[28] C. Schneider and W. Werner, *Some new aspects of rational interpolation*, Mathematics of Computation, 47 (1986), pp. 285–299.

[29] Z. Shen and K. Serkh, *On polynomial interpolation in the monomial basis*, SIAM Journal on Numerical Analysis, 63 (2025), pp. 469–494.

[30] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer, New York, 3rd ed., 2002.

[31] L. N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, Philadelphia, extended ed., 2019.

[32] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[33] L. R. Turner, *Inverse of the Vandermonde matrix with applications*, Tech. Report Technical Note D-3547, NASA, 1966.