Approximation-Degree-Based Interpolation: A New Interpolation Method

Authors: Lian Shiyou

Date: 2019-12-29T00:00:00+00:00

Abstract

This paper introduces the measure of approximate-degree and the concept of approximate-degree function between numerical values, thus developing a new interpolation method ——approximation-degree-based interpolation, i.e., AD interpolation. One-dimensional AD interpolation is done directly by using correlative interpolation formulas; n(n>1)-dimensional AD interpolation is firstly separated into n parallel one-dimensional AD interpolation computations to do respectively, and then got results are synthesized by Sum-Times-Difference formula into a value as the result value of the n-dimensional interpolation. If the parallel processing is used, the efficiency of n-dimensional AD interpolation is almost the same as that of the one-dimensional AD interpolation. Thus it starts a feasible and convenient approach and provides an effective method for high-dimensional interpolations. Furthermore, if AD interpolation is introduced into machine learning, a new instance-based learning method is expected to be realized.

Full Text

Approximation-Degree-Based Interpolation: A New Interpolation Method

Shiyou Lian
Xi' an Shiyou University, Xi' an, China
sylian@xsyu.edu.cn

Abstract

This paper introduces the measure of approximation-degree and the concept of approximation-degree functions between numerical values, thereby developing a novel interpolation method—approximation-degree-based interpolation (AD interpolation). One-dimensional AD interpolation is performed directly using

corresponding interpolation formulas. For n-dimensional cases where n > 1, the computation is first separated into n parallel one-dimensional AD interpolation computations, and the resulting values are then synthesized into a final result using the Sum-Times-Difference formula. When parallel processing is employed, the efficiency of n-dimensional AD interpolation becomes nearly equivalent to that of one-dimensional AD interpolation. This establishes a feasible and convenient approach that provides an effective method for high-dimensional interpolation problems. Furthermore, introducing AD interpolation into machine learning is expected to enable a new instance-based learning paradigm.

Keywords: Approximation-Degree; High-Dimensional Interpolation; First Separating Then Synthesizing; Sum-Times-Difference; Parallel Processing; Instance-Based Learning

1 Introduction

With the rapid development of artificial intelligence and machine learning, traditional numerical computation methods such as interpolation, data fitting, and regression analysis have regained interest and become active research fields again. Although interpolation has been extensively studied, results for high-dimensional interpolation remain relatively scarce, and existing high-dimensional methods often involve heavy computational loads that are unsuitable for local interpolation. High-dimensional interpolation has important applications in artificial intelligence and machine learning; for instance, instance-based learning frequently involves high-dimensional interpolation. Therefore, high-dimensional interpolation—particularly methods suitable for local interpolation—remains an important subject worthy of serious investigation.

Inspired by the approximate evaluation method of flexible linguistic functions in reference [1], this paper introduces the measure of approximation-degree between numerical values to study the approximate evaluation of numerical functions, and then explores interpolation approaches and methods based on approximation-degree, especially for high-dimensional scenarios.

2 Approximation Axiom, Approximation-Degree, and Approximation-Degree Transmission

Approximation Axiom

Let $\mathbb{R}$ be the real number field, $\mathbb{R}^n$ be the n-dimensional real vector space, $U = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n] \subset \mathbb{R}^n$ (where $n \geq 1$), $V = [c, d] \subset \mathbb{R}$, and $y = f(x)$ (with $x = (x_1, x_2, \ldots, x_n)$) be a continuous and non-chaotic function from U to V. If $y_0 = f(x_0)$ (where $x_0 \in U$) is known, then when $x \in U$ is approximate to x_0, $y = f(x) \in V$ is also approximate to y_0.

Definition 2-1

Let $\mathbb{R}$ be the real number field, $x_0 \in [a,b] \subset \mathbb{R}$, and $[\alpha_0, \beta_0] \subset [a,b]$ be a neighborhood of x_0, which is called the approximation domain of x_0. For any $x \in [a,b]$, we say x is approximate to x_0 if and only if $x \in [\alpha_0, \beta_0]$.

Definition 2-2

Let $\mathbb{R}^n$ be the n-dimensional real vector space, U be an n-dimensional subspace of $\mathbb{R}^n$, $x_0 = (x_1^0, x_2^0, \dots, x_n^0) \in U$, and $C \subset U$ be a "circular" region with center x_0, which is called an approximation domain of x_0. For any $x = (x_1, x_2, \dots, x_n) \in U$, we say x is approximate to x_0 if and only if $x \in C$.

Definition 2-3

Let $\mathbb{R}^n$ be the n-dimensional real vector space, $U = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n] \subset \mathbb{R}^n$, and $x_0 = (x_1^0, x_2^0, \dots, x_n^0) \in U$. For any $x = (x_1, x_2, \dots, x_n) \in U$, we say x is strictly approximate to x_0 if and only if the components $x_1, x_2, \dots, x_n$ of x are approximate to the components $x_1^0, x_2^0, \dots, x_n^0$ of x_0, respectively—that is, $x_i \in [\alpha_i, \beta_i] \subset [a_i, b_i]$ (where $[\alpha_i, \beta_i]$ is the approximation domain of x_i^0, for $i = 1, 2, \dots, n$). The "square" region $[\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \cdots \times [\alpha_n, \beta_n] \subset U$ is called the strict approximation domain of x_0.

In contrast to the strict approximation and strict approximation domain in Definition 2-3, we refer to the approximation and approximation domain defined in Definition 2-2 as ordinary approximation and ordinary approximation domain. The ordinary approximation operates at the level of points (vectors), while strict approximation operates at the level of point coordinates (vector components). The relationship between the ordinary and strict approximation domains of the same point x_0 is illustrated in Figure [Figure 2: see original paper]-1, which visually demonstrates the relationship between ordinary and strict approximation.

We know that approximate evaluation of a function aims to obtain the approximate value of $f(x')$ given a pair of corresponding values (x_0, y_0) of function $y = f(x)$ (where $x = (x_1, x_2, \dots, x_n)$) and a point x' approximate to x_0. By the Approximation Axiom, when x' is approximate to x_0, $f(x') = y$ is approximate to $f(x_0) = y_0$. However, approximation is a flexible concept (i.e., a flexible linguistic value [1]). To precisely characterize these two approximation relations and their interrelationship, we introduce the measure of approximation-degree.

Definition 2-4

Let $\mathbb{R}$ be the real number field, $x_0 \in [a,b] \subset \mathbb{R}$, and $[\alpha_0, \beta_0] \subset [a,b]$ be the approximation domain of x_0. Define

$$A_{x_0}(x)=\begin{cases}\frac{x_0-\alpha_0}{x_0-\alpha_0}, & x\in[\alpha_0,x_0]\\ \frac{\beta_0-x_0}{x_0-\beta_0}, & x\in[x_0,\beta_0]\end{cases} \tag{2-1}$$

as the approximation-degree of x to x_0. Here $x_0-\alpha_0=r_l$ is called the left approximation radius of x_0, and $\beta_0-x_0=r_r$ is called the right approximation radius of x_0. We call the functional relation defined by Formula (2-1) the approximation-degree function of x_0.

The range of the approximation-degree function $A_{x_0}(x)$ is $[0,1]$, and it is reversible. In fact, it is easy to obtain

$$A_{x_0}(x)^{-1}=\begin{cases}d_x(x_0-\alpha_0)+\alpha_0, & d_x\in[0,1]\\ d_x(x_0-\beta_0)+\beta_0, & d_x\in[0,1]\end{cases} \tag{2-2}$$

where $d_x\in[0,1]$ is the approximation-degree of x to x_0.

Definition 2-5

Let $\mathbb{R}^n$ be the n-dimensional real vector space, U be an n-dimensional subspace of $\mathbb{R}^n$, $x_0=(x_1^0,x_2^0,\dots,x_n^0)\in U$, and $C\subset U$ be an approximation domain of x_0. Define

$$A_{x_0}(x)=1-\frac{\|x_0-x\|}{r},\quad x=(x_1,x_2,\dots,x_n)\in C \tag{2-3}$$

as the approximation-degree of x to x_0, where r is the radius of C, called the approximation radius of x_0.

We further consider the dependence relation between approximation-degrees $A_{y_0}(y)$ and $A_{x_0}(x)$ under the constraint of the functional relation $y=f(x)$. By the Approximation Axiom, the closer x is to x_0, the closer y is to y_0, and thus the closer $A_{y_0}(y)$ is to $A_{x_0}(x)$. Therefore, when the corresponding approximation domains are small, we can simply assume the two are equal, setting

$$A_{y_0}(y)=A_{x_0}(x)$$

This equation effectively transmits the approximation-degree of the independent variable x to x_0 to the corresponding function value y as the approximation-degree of y to y_0.

3 Approximate Evaluation Based on Approximation-Degree

3.1 Approximate Evaluation of Univariate Functions

Let $\mathbb{R}$ be the real number field, $U=[a,b]\subset\mathbb{R}$, $V=[c,d]\subset\mathbb{R}$, and $y=f(x)$ be a continuous and non-chaotic function from U to V. Given a pair of corresponding values (x_0,y_0) of function $y=f(x)$ and a point x' approximate to x_0, we seek the approximate value of $f(x')$.

Let the approximation domain of x_0 be $[a_1,b_1]\subset[a,b]$. According to the definition of approximation-degree functions, the approximation-degree function of x_0 is

$$A_{x_0}(x)=\begin{cases}\frac{x_0-a_1}{x_0-a_1}, & x\in[a_1,x_0]\\ \frac{x_0-b_1}{x_0-b_1}, & x\in[x_0,b_1]\end{cases} \tag{3-1}$$

Let the approximation domain of y_0 be $[c_1,d_1]\subset[c,d]$. The approximation-degree function of y_0 is

$$A_{y_0}(y)=\begin{cases}\frac{y_0-c_1}{y_0-c_1}, & y\in[c_1,y_0]\\ \frac{y_0-d_1}{y_0-d_1}, & y\in[y_0,d_1]\end{cases} \tag{3-2}$$

The inverse function of $A_{y_0}(y)$, which is not difficult to obtain, is

$$A_{y_0}(y)^{-1}=\begin{cases}d_y(y_0-d_1)+d_1, & d_y\in[0,1]\\ d_y(y_0-c_1)+c_1, & d_y\in[0,1]\end{cases} \tag{3-3}$$

where d_y is the approximation-degree of y to y_0.

The graphs of the approximation-degree functions $A_{x_0}(x)$ and $A_{y_0}(y)$ are shown in Figure [Figure 3: see original paper]-1 and Figure 3-2, respectively.

To find the approximate value, we first compute the approximation-degree $A_{x_0}(x')$, then set $A_{y_0}(y')=A_{x_0}(x')$ (transmitting the approximation-degree of x' to x_0 to y'). Next, we derive the required approximate value of y' from the approximation-degree $A_{y_0}(y')$ and the inverse function $A_{y_0}(y)^{-1}$.

It can be seen that the inverse function $A_{y_0}(y)^{-1}$ is a piecewise function with two parallel expressions. Substituting the approximation-degree $A_{y_0}(y')=d$ into $A_{y_0}(y)^{-1}$ yields two y-values (y_1 and y_2). In other words, from a single x' we obtain two y-values (as shown in Figure 3-3). The question then arises: which y-value should be chosen as the desired approximate value?

The appropriate y-value depends on the orientation of x' relative to x_0 and the trend of $f(x)$ near x_0 (i.e., whether it is increasing, decreasing, or constant).

When the derivative $f'(x_0)$ is unknown, we can consider whether there exists a point x^* on the x' side near x_0 whose corresponding function value $f(x^*) = y^*$ is known. If such a point x^* exists, we can estimate the trend of $f(x)$ between x_0 and x^* using the size relationship between y^* and y_0, and then determine which y-value to select.

Specifically, when $x^* < x' < x_0$, if $y^* < y_0$, this indicates that the general trend of $f(x)$ is increasing on the subinterval (x^*, x_0), so y_1 (the y-value less than y_0) should be chosen. Conversely, if $y^* > y_0$, this shows that the general trend is decreasing on (x^*, x_0), so y_2 (the y-value greater than y_0) should be selected. Similarly, when $x_0 < x' < x^*$, if $y_0 < y^*$, indicating an increasing trend on (x_0, x^*), then y_2 should be chosen; if $y_0 > y^*$, indicating a decreasing trend, then y_1 should be chosen. If $y^* = y_0$, suggesting that $f(x)$ is likely constant on the subintervals (x^*, x_0) or (x_0, x^*), then $y = y_0$ can be taken regardless of whether $x^* < x' < x_0$ or $x_0 < x' < x^*$.

3.2 Approximate Evaluation of Multivariate Functions

Let us consider a function of two variables as an example. Let $z = f(x, y)$ be a function from $[a_1, b_1] \times [a_2, b_2]$ to $[c, d]$. Suppose a pair of corresponding values $((x_0, y_0), z_0)$ of function $z = f(x, y)$ is known, and point (x', y') is approximate to point (x_0, y_0). When the expression of $f(x, y)$ is unknown or not used, we seek the approximate value of $f(x', y')$.

This problem is similar to the univariate case, so we extend the previous treatment to functions of two variables. Specifically, we can find the approximate value of the function at point (x', y') from the approximation-degree of point (x', y') to point (x_0, y_0). Let $z' = A_{x_0,y_0}(x', y')$, then substitute it into the inverse function $A_{z_0}(z)^{-1}$ of $A_{z_0}(z)$. Here, two z-values will also be obtained. Thus, we can consider whether there exists an adjacent point (x^*, y^*) of (x_0, y_0) in the direction determined by points (x_0, y_0) and (x', y') whose function value is known. If such a point (x^*, y^*) exists (as shown in Figure 3-4), we can determine the choice of z-value using $f(x^*, y^*)$ as a reference.

However, although this technique is theoretically feasible, it has a precondition: points (x^*, y^*), (x_0, y_0), and (x', y') must lie exactly on a straight line. Therefore, the method has limitations for multivariate functions. This raises the question: is there a better approach?

By the definition of strict approximation (see Definition 2-3), point (x', y') being approximate to (x_0, y_0) is equivalent to x' being approximate to x_0 and y' being approximate to y_0. Thus, we can find the approximate values z_x and z_y of functions $f(x, y_0)$ and $f(x_0, y)$ at x' and y', respectively. We can then consider whether there exist adjacent points of (x_0, y_0) in the x-direction and y-direction, namely (x^*, y_0) and (x_0, y^*), as shown in Figure 3-5, with known function values $f(x^*, y_0)$ and $f(x_0, y^*)$. If such points exist, the approximate value z_x of $f(x', y_0)$ can be obtained using $f(x^*, y_0)$, and the approximate value z_y of $f(x_0, y')$ can be obtained using $f(x_0, y^*)$, just as in the univariate case.

Specifically, we first compute the approximation-degrees $A_{x_0}(x')$ and $A_{y_0}(y')$, then set $A_{z_0}(z') = A_{x_0}(x')$ and $A_{z_0}(z') = A_{y_0}(y')$. Substituting these separately into the inverse function $A_{z_0}(z)^{-1}$ yields two pairs of candidate z-values. Using $f(x^*, y_0)$ and $f(x_0, y^*)$ as references, we choose z_x and z_y from their respective candidate values (as shown in Figure 3-5).

Having obtained the approximate values z_x and z_y, how can we further derive the required approximate value z? Can z_x and z_y be synthesized into a single value? For example, could we take the average or weighted sum of z_x and z_y as z? Numerical experiments show that while the average or weighted sum can serve as a synthesized value, the results are generally not ideal. However, fortunately, through further analysis based on the average value, we have discovered a better method for synthesizing z_x and z_y.

Let $z_1 = (z_x + z_y)/2$ be the average of z_x and z_y. As shown in Figure 3-6, z_1 can actually be viewed as an approximate value of $f(x, y)$ at the midpoint (denoted by (x_1, y_1)) between (x', y_0) and (x_0, y'). From the figure, we observe that $z_1 < z_0$, indicating a decreasing trend in function values from z_0 to z_1. Setting $z_0 - z_1 = c_1$ (where c_1 is the length of segment BC in Figure 3-7(a)), we have $z_1 = z_0 - c_1$. Based on the trend of function values from z_0 to z_1 (i.e., the slope of segment AB in Figure 3-7(a)), and considering that point (x_1, y_1) is exactly the midpoint of the segment joining (x', y') and (x_0, y_0)—that is, $(x', y') - (x_0, y_0) = 2 \cdot ((x_1, y_1) - (x_0, y_0))$—we infer that the approximate function value at point (x', y') can be $z_0 - 2c_1$ (as shown in Figure 3-7(a)). Thus, it follows that

$$z = z_0 - 2c_1 = z_0 - 2(z_0 - z_1) = z_0 - 2\left[z_0 - \frac{z_x + z_y}{2}\right] = z_x + z_y - z_0$$

Of course, z_1 may also be greater than z_0 or equal to z_0. If $z_1 > z_0$, the trend of function values from z_0 to z_1 is increasing (as shown in Figure 3-7(b)). Setting $z_0 - z_1 = c_2$ (where c_2 is the length of segment BC in Figure 3-7(b)), we have $z_1 = z_0 - c_2$. Based on the increasing trend from z_0 to z_1 (i.e., the slope of segment AB in Figure 3-7(b)), we infer that the function value at point (x', y') can be $z_0 - 2c_2$. Thus,

$$z = z_0 + 2c_2 = z_0 + 2(z_1 - z_0) = z_0 + 2\left[\frac{z_x + z_y}{2} - z_0\right] = z_x + z_y - z_0$$

The third case occurs when $z_1 = z_0$, indicating that function values remain unchanged from z_0 to z_1. Thus, we can take $z = z_0$. From $z_0 = z_1 = (z_x + z_y)/2$, it follows that $2z_0 = z_x + z_y$, and therefore

$$z = 2z_0 - z_0 = z_x + z_y - z_0$$

In summary, regardless of the relationship between the average of z_x and z_y and z_0, or how the function value varies from z_0 to z_1, the approximate function value at point (x', y') can always be taken as

$$z = z_x + z_y - z_0 \tag{3-4}$$

Based on this analysis, we obtain a method for finding the approximate value of a function at point (x', y') given adjacent points (x^*, y_0) and (x_0, y^*) of (x_0, y_0). First, find the approximate values z_x and z_y of $f(x', y_0)$ and $f(x_0, y')$ using $f(x_0, y_0) = z_0$, approximation-degrees $A_{x_0}(x')$ and $A_{y_0}(y')$, and the known values $f(x^*, y_0)$ and $f(x_0, y^*)$. Then synthesize z_x and z_y into a value z using Formula (3-4) as the approximate value of $f(x', y')$.

This approach effectively separates the approximate evaluation of a bivariate function into the approximate evaluation of two univariate functions, then synthesizes the two obtained approximate values into a single value as the approximation of the original bivariate function.

The "first separating then synthesizing" idea and method for bivariate functions can be extended to functions of three variables. That is, given a pair of corresponding values $((x_0, y_0, z_0), u_0)$ of a trivariate function $u = f(x, y, z)$ and a point (x', y', z') approximate to (x_0, y_0, z_0), if there exist adjacent points of (x_0, y_0, z_0) along the coordinate directions of (x', y', z')—namely (x^*, y_0, z_0), (x_0, y^*, z_0), and (x_0, y_0, z^*) (as shown in Figure 3-8)—with known function values $f(x^*, y_0, z_0)$, $f(x_0, y^*, z_0)$, and $f(x_0, y_0, z^*)$, then we can first find the approximate values u_x, u_y, and u_z of $f(x', y_0, z_0)$, $f(x_0, y', z_0)$, and $f(x_0, y_0, z')$ using $f(x_0, y_0, z_0) = u_0$, approximation-degrees $A_{x_0}(x')$, $A_{y_0}(y')$, and $A_{z_0}(z')$, and the known adjacent point values. Then we synthesize u_x, u_y, and u_z into a value u as the approximate value of $f(x', y', z')$.

From the synthesizing formula for bivariate functions, $z = z_x + z_y - z_0 = z_x + z_y - 1 \cdot z_0$, we infer that the synthesizing formula for trivariate functions should be

$$u = u_x + u_y + u_z - 2u_0 \tag{3-5}$$

In fact, when $(x', y', z') = (x_0, y_0, z_0)$, we have $u_x = u_y = u_z = u_0$, while on the other hand, $u_0 = u_0 + u_0 + u_0 - 2u_0$. This indicates that for the special point (x_0, y_0, z_0), Formula (3-5) is correct, indirectly verifying the rationality of the synthesizing formula (the three-dimensional AD interpolation example in Section 5.2 further confirms this).

Based on the above analysis, in general, given a pair of corresponding values (x_0, y_0) (where $x_0 = (x_1^0, x_2^0, \ldots, x_n^0)$) of an n-variate function $y = f(x)$ (with $x = (x_1, x_2, \ldots, x_n)$) and a point $x' = (x_1', x_2', \ldots, x_n')$ approximate to x_0, if there exists an adjacent point of x_0 on the side of each corresponding coordinate of x' in each coordinate direction of x_0—that is, points $(x_1^*, x_2^0, \ldots, x_n^0)$,

$(x_1^0, x_2^*, \dots, x_n^0)$, ..., and $(x_1^0, x_2^0, \dots, x_n^*)$—with known function values, then we can first find the approximate values $y_{x_1}, y_{x_2}, \dots, y_{x_n}$ of $f(x_1', x_2^0, \dots, x_n^0)$, $f(x_1^0, x_2', \dots, x_n^0)$, ..., and $f(x_1^0, x_2^0, \dots, x_n')$ using $f(x_0) = y_0$, approximation-degrees $A_{x_0}(x_1'), A_{x_0}(x_2'), \dots, A_{x_0}(x_n')$, and the known values at adjacent points. Then we synthesize $y_{x_1}, y_{x_2}, \dots, y_{x_n}$ into a value y as the approximate value of $f(x')$ using the formula

$$y = \sum_{i=1}^{n} y_{x_i} - (n-1)y_0 \tag{3-6}$$

This provides an effective approach and method for finding approximate values of multivariate functions. For convenience, we refer to Formula (3-6) as the Sum-Times-Difference formula.

4 Interpolation Based on Approximation-Degree

Let $y = f(x)$ be a function from $[a, b]$ to $[c, d]$, and let $\{(x_i, y_i)\}_{i=1}^n$ be a set of corresponding value pairs of $y = f(x)$. We need to construct an interpolating function $g(x)$ (when the expression of $f(x)$ is unknown or not used) such that $g(x_i) = f(x_i)$ for $i = 1, 2, \dots, n$, and for other $x \in [a, b]$, $g(x) \approx f(x)$. This is the standard interpolation problem.

The method for finding functional approximate values based on approximation-degree can also be used for interpolation. We discuss this problem in detail below.

Let $a = x_1 < x_2 < \cdots < x_n = b$. Then $\{x_i\}_{i=1}^n$ is a set of interpolation base points (or nodes). We define the approximation domain of x_1 as $[x_1, x_2]$, the approximation domains of x_i as $[x_{i-1}, x_{i+1}]$ for $i = 2, 3, \dots, n-1$, and the approximation domain of x_n as $[x_{n-1}, x_n]$. The approximation-degree function of base point x_1 is

$$A_{x_1}(x) = \frac{x_1 - x_2}{x_1 - x_2}, \quad x \in [x_1, x_2] \tag{4-1}$$

The approximation-degree functions of x_i are

$$A_{x_i}(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}}, & x \in [x_{i-1}, x_i] \\ \frac{x - x_{i+1}}{x_i - x_{i+1}}, & x \in [x_i, x_{i+1}] \end{cases} \tag{4-2}$$

and the approximation-degree function of x_n is

$$A_{x_n}(x) = \frac{x - x_{n-1}}{x_n - x_{n-1}}, \quad x \in [x_{n-1}, x_n] \tag{4-3}$$

The graphs of these functions are shown in Figure [Figure 4: see original paper]-1.

Notes: - The approximation domain of a number x_i is determined not only by x_i itself but also by its two neighbors, x_{i-1} and x_{i+1}. Therefore, the two approximation radii $r_l = x_i - x_{i-1}$ and $r_r = x_{i+1} - x_i$ defined by Equation (4-2) are not necessarily equal. For distinction, we call the approximation domain determined by a number alone the absolute approximation domain, and that determined by its two neighbors the relative approximation domain. Unless otherwise stated, subsequent approximation domains refer to relative approximation domains. - When $x \in [x_{i-1}, \frac{x_{i-1}+x_i}{2})$, we have $A_{x_i}(x) < 0.5$ and $A_{x_{i-1}}(x) > 0.5$, indicating that x is closer to x_{i-1}. Similarly, when $x \in (\frac{x_i+x_{i+1}}{2}, x_{i+1}]$, we have $A_{x_i}(x) < 0.5$ and $A_{x_{i+1}}(x) > 0.5$, showing that x is closer to x_{i+1}. Only when $x \in [\frac{x_i+x_{i-1}}{2}, \frac{x_i+x_{i+1}}{2}]$ do we have $A_{x_i}(x) \geq 0.5$. If we stipulate that a point x is practically approximate to base point x_i if and only if its approximation-degree $A_{x_i}(x) \geq 0.5$, then the interval $[\frac{x_i+x_{i-1}}{2}, \frac{x_i+x_{i+1}}{2}]$ is the practical domain of the approximation-degree function $A_{x_i}(x)$, which is also the practical approximation domain of x_i (while $[x_{i-1}, x_{i+1}]$ is the conceptual domain of $A_{x_i}(x)$ and the conceptual approximation domain of x_i). In the sense of practical approximation, we have the following conclusion:

Proposition 4-1. A point x is practically approximate to base point x_i (i.e., $A_{x_i}(x) \geq 0.5$) if and only if x lies in the practical approximation domain ($[\frac{x_i+x_{i-1}}{2}, \frac{x_i+x_{i+1}}{2}]$) of x_i.

We define the approximation-degree functions of y_i $(i = 1, 2, \dots, n)$ using the same principle and approach.

First, we note that $y_1, y_2, \dots, y_n$ corresponding to $x_1, x_2, \dots, x_n$ in order do not necessarily satisfy $y_1 < y_2 < \cdots < y_n$. In fact, the size relationship between y_i and y_{i-1} in the sample data may be $y_i < y_{i-1}$, $y_i = y_{i-1}$, or $y_i > y_{i-1}$; similarly, the relationship between y_i and y_{i+1} may be $y_i < y_{i+1}$, $y_i = y_{i+1}$, or $y_i > y_{i+1}$. We also observe that when x lies between x_{i-1} and x_i, the corresponding y certainly lies between y_{i-1} and y_i; and when x lies between x_i and x_{i+1}, the corresponding y certainly lies between y_i and y_{i+1}. Thus, we only need to consider four (semi) approximation-degree functions of y_i corresponding to the four cases: $y_{i-1} < y_i$, $y_i < y_{i-1}$, $y_i < y_{i+1}$, and $y_{i+1} < y_i$.

1. Suppose $y_{i-1} < y_i$. We define the first sub-expression of the approximation-degree function of y_i in this case as

$$A_{y_i}(y) = \frac{y - y_{i-1}}{y_i - y_{i-1}}, \quad y \in [y_{i-1}, y_i] \tag{4-4}$$

Its graph is shown in Figure 4-2(a).

2. Suppose $y_i < y_{i-1}$. We define the second sub-expression of the approximation-degree function of y_i in this case as

$$A_{y_i}(y) = \frac{y_{i-1} - y}{y_{i-1} - y_i}, \quad y \in [y_i, y_{i-1}] \tag{4-5}$$

Its graph is shown in Figure 4-2(b).

3. Suppose $y_i < y_{i+1}$. We define the second sub-expression of the approximation-degree function of y_i in this case as

$$A_{y_i}(y) = \frac{y_{i+1} - y}{y_{i+1} - y_i}, \quad y \in [y_i, y_{i+1}] \tag{4-6}$$

Its graph is shown in Figure 4-2(c).

4. Suppose $y_{i+1} < y_i$. We define the first sub-expression of the approximation-degree function of y_i in this case as

$$A_{y_i}(y) = \frac{y - y_{i+1}}{y_i - y_{i+1}}, \quad y \in [y_{i+1}, y_i] \tag{4-7}$$

Its graph is shown in Figure 4-2(d).

Clearly, in the four expressions above, (4-4) = (4-5) and (4-6) = (4-7). That is, regardless of whether $y_{i-1} < y_i$ or $y_i < y_{i-1}$, when y lies between y_{i-1} and y_i, the expression of the approximation-degree function of y_i is always

$$A_{y_i}(y) = \frac{y - y_{i-1}}{y_i - y_{i-1}}$$

and regardless of whether $y_{i+1} < y_i$ or $y_i < y_{i+1}$, when y lies between y_i and y_{i+1}, the expression is always

$$A_{y_i}(y) = \frac{y - y_{i+1}}{y_i - y_{i+1}}$$

Thus, the four functional expressions can be reduced to two expressions.

Furthermore, the coefficients of the first-degree terms in these two expressions are precisely the slopes of the corresponding straight lines, and their signs determine whether the function is increasing or decreasing. Consequently, whether $y = f(x)$ is increasing or decreasing on subintervals $[\frac{x_i + x_{i-1}}{2}, x_i]$ and $[x_i, \frac{x_i + x_{i+1}}{2}]$ is determined by the signs of differences $y_i - y_{i-1}$ and $y_i - y_{i+1}$. This means that numerical calculation replaces logical judgment for the trend of $f(x)$ on these subintervals.

The inverse expressions of these two functional expressions are respectively

$$A_{y_i}(y)^{-1} = d_y(y_i - y_{i-1}) + y_{i-1} \tag{4-8}$$

$$A_{y_i}(y)^{-1} = d_y(y_i - y_{i+1}) + y_{i+1} \tag{4-9}$$

where $d_y = A_{y_i}(y) \in [0,1]$ is the approximation-degree of y to y_i.

We then transmit the approximation-degree of x (where $x \in [\frac{x_i+x_{i-1}}{2}, \frac{x_i+x_{i+1}}{2}]$) to x_i to the corresponding y, setting $d_y = d_x = A_{x_i}(x)$. Considering that on subinterval $[\frac{x_i+x_{i-1}}{2}, x_i]$, the interpolated function $y = f(x)$ may be increasing, decreasing, or constant. When $y = f(x)$ is increasing, certainly $y_{i-1} < y_i$, so the corresponding $y \in [\frac{y_i+y_{i-1}}{2}, y_i]$; when decreasing, certainly $y_i < y_{i-1}$, so $y \in [y_i, \frac{y_i+y_{i-1}}{2}]$; and when constant, $y_i = y_{i-1}$, so $y = y_i \in [\frac{y_i+y_{i-1}}{2}, y_i]$ as well as $y \in [y_i, \frac{y_i+y_{i-1}}{2}]$. Thus, when $x \in [\frac{x_i+x_{i-1}}{2}, x_i]$, the corresponding y_i and y_{i-1} are adjacent, and here $A_{x_i}(x) = \frac{x-x_{i-1}}{x_i-x_{i-1}}$. The expression of the corresponding inverse function $A_{y_i}(y)^{-1}$ in Equation (4-8) becomes

$$A_{y_i}(y)^{-1} = \frac{x - x_{i-1}}{x_i - x_{i-1}}(y_i - y_{i-1}) + y_{i-1} = \frac{y_i - y_{i-1}}{x_i - x_{i-1}}x + \frac{x_i y_{i-1} - x_{i-1} y_i}{x_i - x_{i-1}}$$

namely

$$y = \frac{y_i - y_{i-1}}{x_i - x_{i-1}}x + \frac{x_i y_{i-1} - x_{i-1} y_i}{x_i - x_{i-1}}, \quad x \in \left[\frac{x_i + x_{i-1}}{2}, x_i\right] \tag{4-10}$$

Similarly, on subinterval $[x_i, \frac{x_i+x_{i+1}}{2}]$, the function $y = f(x)$ may be increasing, decreasing, or constant. When increasing, certainly $y_i < y_{i+1}$, so $y \in [y_i, \frac{y_i+y_{i+1}}{2}]$; when decreasing, certainly $y_{i+1} < y_i$, so $y \in [\frac{y_i+y_{i+1}}{2}, y_i]$; and when constant, $y_i = y_{i+1}$, so $y = y_i \in [\frac{y_i+y_{i+1}}{2}, y_i]$ as well as $y \in [y_i, \frac{y_i+y_{i+1}}{2}]$. Thus, when $x \in [x_i, \frac{x_i+x_{i+1}}{2}]$, the corresponding y_i and y_{i+1} are adjacent, and here $A_{x_i}(x) = \frac{x_{i+1}-x}{x_{i+1}-x_i}$. The expression of the corresponding inverse function $A_{y_i}(y)^{-1}$ in Equation (4-9) becomes

$$A_{y_i}(y)^{-1} = \frac{x_{i+1} - x}{x_{i+1} - x_i}(y_i - y_{i+1}) + y_{i+1} = \frac{y_i - y_{i+1}}{x_i - x_{i+1}}x + \frac{x_i y_{i+1} - x_{i+1} y_i}{x_i - x_{i+1}}$$

namely

$$y = \frac{y_i - y_{i+1}}{x_i - x_{i+1}}x + \frac{x_i y_{i+1} - x_{i+1} y_i}{x_i - x_{i+1}}, \quad x \in \left[x_i, \frac{x_i + x_{i+1}}{2}\right] \tag{4-11}$$

Thus, with these two expressions, we can directly obtain the corresponding $y \in [\frac{y_i+y_{i-1}}{2}, y_i]$ or $[y_i, \frac{y_i+y_{i-1}}{2}]$ from $x \in [\frac{x_i+x_{i-1}}{2}, x_i]$, and directly obtain $y \in$

$[y_i, \frac{y_i+y_{i+1}}{2}]$ or $[\frac{y_i+y_{i+1}}{2}, y_i]$ from $x \in [x_i, \frac{x_i+x_{i+1}}{2}]$, without needing to compute approximation-degrees and evaluate inverse functions.

Equations (4-10) and (4-11) are in fact two interpolation formulas. Using these formulas, we can solve the interpolation problem described above. For an evaluation point $x \in [a, b]$, we only need to first find the approximation domain in which x lies (equivalent to finding the nearest base point x_i where $i \in \{1, 2, \dots, n\}$), then construct and apply the corresponding interpolation formula based on the position of x relative to x_i.

In this way, we derive an interpolation method from approximate evaluation of functions based on approximation-degree—approximation-degree-based interpolation, or AD interpolation for short. Specifically, AD interpolation proceeds as follows: take base points (or nodes) $a = x_1, x_2, \dots, x_n = b$ as reference points, partition the interval $[a, b] = [x_1, x_n]$ into $2n-2$ subintervals as shown in Figure 4-3—$[x_1, \frac{x_1+x_2}{2}]$, $[\frac{x_1+x_2}{2}, x_2]$, $[x_2, \frac{x_2+x_3}{2}]$, ..., $[\frac{x_{n-1}+x_n}{2}, x_n]$—as interpolation intervals. Then for an evaluation point $x \in [\frac{x_i+x_{i-1}}{2}, x_i]$, interpolate using Formula (4-10); for $x \in [x_i, \frac{x_i+x_{i+1}}{2}]$, interpolate using Formula (4-11). Since each specific interpolation formula incorporates the trend of the interpolated function $y = f(x)$ on the corresponding subintervals, the obtained approximate value does not deviate significantly from the expected value, and the case of obtaining two y-values from a single x does not occur.

Example 4.1 Use AD interpolation to interpolate the functions $y = \sin x$ and $y = \cos x$.

We take sampled data points (x_i, y_i) of $y = \sin x$ as follows: - x_i: $0 \times \pi, 0.1 \times \pi, 0.2 \times \pi, 0.3 \times \pi, \dots, 1.9 \times \pi, 2 \times \pi$ - y_i: $\sin(x_i)$

Evaluation points x are: $0 \times \pi, 0.02 \times \pi, 0.04 \times \pi, 0.06 \times \pi, \dots, 1.98 \times \pi, 2 \times \pi$.

Using AD interpolation (programmed in MATLAB), the obtained y-values are:

```
-0.2809  -0.3400  -0.3963  -0.4526  -0.5089  -0.5653  -0.6146  -0.6593
-0.7040  -0.7487  -0.7934  -0.8277  -0.8564  -0.8851  -0.9138  -0.9424
-0.9580  -0.9679  -0.9778  -0.9876  -0.9975  -0.9926  -0.9827  -0.9728
-0.9629  -0.9530  -0.9281  -0.8994  -0.8707  -0.8420  -0.8133  -0.7710
-0.7263  -0.6816  -0.6369  -0.5923  -0.5371  -0.4808  -0.4245  -0.3681
-0.3118  -0.2497  -0.1873  -0.1249  -0.0624   0.0000
```

The interpolation effect is shown in Figure 4-4(a). For $y = \cos x$, we interpolate using the same method and data; the effect is shown in Figure 4-4(b).

Accuracy analysis: - $\sin x$: maximum error 0.0121, minimum error 0, average error 0.0052 - $\cos x$: maximum error 0.0122, minimum error 0, average error 0.0052

From the interpolation formulas, it can be seen that one-dimensional AD interpolation is essentially a local linear interpolation, achieving the same result as conventional piecewise linear interpolation through a different approach.

5 Multidimensional and High-Dimensional Interpolations Based on Approximation-Degree

5.1 Two-Dimensional AD Interpolation

Suppose data points of a bivariate function $z = f(x, y)$, denoted as $((x_i, y_j), z_{ij})$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$, are known, and the base points (x_i, y_j) are regularly distributed, satisfying $a_x = x_1 < x_2 < \cdots < x_n = b_x$ and $a_y = y_1 < y_2 < \cdots < y_m = b_y$. These points form an $m \times n$ matrix and a Cartesian grid on the x-y plane (as shown in Figure [Figure 5: see original paper]-1). We consider the corresponding interpolation problem.

For this type of two-dimensional interpolation (which is also standard two-dimensional interpolation), we first adopt ideas and techniques similar to one-dimensional AD interpolation from Section 4: take base points $(x_1, y_1), \ldots, (x_n, y_m)$ as reference points, and partition the region $U = [a_x, b_x] \times [a_y, b_y] = [x_1, x_n] \times [y_1, y_m]$ into $(2n-2) \times (2m-2)$ subregions as shown in Figure 5-1, serving as interpolation regions. Then for an evaluation point $(x, y) \in U$, we locate the subregion it lies in (equivalent to finding the nearest base point (x_i, y_j) and determining the position of (x, y) relative to (x_i, y_j)), and perform interpolation.

For interpolation on these two-dimensional subregions, we have not derived corresponding interpolation formulas. Since the base points are regularly distributed—meaning each base point has adjacent base point(s) along its coordinate directions—we employ the "first separating then synthesizing" technique described in Section 3.2 to obtain the interpolated value at point (x, y). Specifically, we separate the interpolation computation of evaluation point (x, y) with its nearest base point (x_i, y_j) into interpolation computations of x with x_i and y with y_j, obtaining approximate values z_x and z_y, respectively. Then we synthesize z_x and z_y into a value z as the interpolated value at (x, y).

The two interpolation computations resulting from separation are already one-dimensional interpolations, so the previous two interpolation formulas can be applied. However, the corresponding formulas are with respect to x and y separately, totaling four formulas grouped as two pairs:

$$z = \frac{z_{ij} - z_{i-1,j}}{x_i - x_{i-1}} x + \frac{x_i z_{i-1,j} - x_{i-1} z_{ij}}{x_i - x_{i-1}}, \quad x \in \left[\frac{x_i + x_{i-1}}{2}, x_i\right] \tag{5-1}$$

$$z = \frac{z_{ij} - z_{i+1,j}}{x_i - x_{i+1}} x + \frac{x_i z_{i+1,j} - x_{i+1} z_{ij}}{x_i - x_{i+1}}, \quad x \in \left[x_i, \frac{x_i + x_{i+1}}{2}\right] \tag{5-2}$$

$$z = \frac{z_{ij} - z_{i,j-1}}{y_j - y_{j-1}} y + \frac{y_j z_{i,j-1} - y_{j-1} z_{ij}}{y_j - y_{j-1}}, \quad y \in \left[\frac{y_j + y_{j-1}}{2}, y_j\right] \tag{5-3}$$

$$z = \frac{z_{ij} - z_{i,j+1}}{y_j - y_{j+1}} y + \frac{y_j z_{i,j+1} - y_{j+1} z_{ij}}{y_j - y_{j+1}}, \quad y \in \left[y_j, \frac{y_j + y_{j+1}}{2}\right] \tag{5-4}$$

where $z_{ij} = f(x_i, y_j)$.

For an evaluation point $(x, y) \in U$, we only need to select one formula with respect to x and one with respect to y. The specific pair of formulas used should be determined by the subintervals in which x and y respectively lie. For example, if $x \in [\frac{x_i+x_{i-1}}{2}, x_i]$ and $y \in [y_j, \frac{y_j+y_{j+1}}{2}]$, then Formulas (5-1) and (5-4) should be used.

After obtaining z_x and z_y using the selected formulas, they must be synthesized into a value z. From Formula (3-7) in Section 3.2, the formula for synthesizing z_x and z_y is

$$z = z_x + z_y - z_{ij} \tag{5-5}$$

In this way, we have extended AD interpolation to two-dimensional cases. From Formula (5-5), it can be seen that this two-dimensional AD interpolation using "first separating then synthesizing" differs from conventional bilinear interpolation, but its effect is comparable.

Example 5.1 Use AD interpolation for the function $z = -x^2 - y^2$.

We take sampled data points $((x_i, y_j), z_{ij})$ as follows: - x_i: $-20, -18, -16, \dots, -2, 0, 2, \dots, 16, 18, 20$
- y_j: $-20, -18, -16, \dots, -2, 0, 2, \dots, 16, 18, 20$

Evaluation points (x, y) are: - x: $-20, -20, 20, -19.5, -17.8, -18, -15.3, -12, -10.2, -10, -10, 0, 0, 10, 10, 5.6, 4$
- y: $-20, 20, -20, -19.5, -17.8, -5, -15.5, 2.5, -10.2, 10, -20, 0, -20, -20, -10, -15.3, -3.8, -13.4, -2.8, -1.9, -$

Using AD interpolation (programmed in MATLAB), the obtained z-values are:

```
-800.0000  -800.0000  -800.0000  -762.0000  -634.4000  -350.0000  -476.0000
-151.0000  -208.8000  -200.0000  -500.0000     0.0000  -400.0000  -500.0000
-200.0000  -267.0000   -37.8000  -192.8000   -12.4000    -9.6000   -45.6000
-106.8000   -73.0000  -276.0000  -400.0000  -500.0000  -500.0000  -444.2000
-667.0000  -573.2000  -470.8000  -362.8000  -164.4000  -280.0000  -214.2000
-317.0000  -195.2000   -60.2000   -72.4000  -118.4000  -198.8000  -191.8000
-215.8000  -653.8000  -492.4000
```

The interpolation effect is shown in Figure 5-2.

Accuracy analysis: maximum error 1.8300, minimum error 0, average error 0.9222.

Example 5.2 Use AD interpolation for the function $z = \frac{\sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}$.

We take sampled data points $((x_i, y_j), z_{ij})$ as follows: - x_i: $-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5$
- y_j: $-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5$

Evaluation points (x, y) are: $x = -5, -4.9, -4.8, -4.7, -4.6, -4.5, \dots, 4.5, 4.6, 4.7, 4.8, 4.9, 5$; y similarly.

Using AD interpolation (programmed in MATLAB, the corresponding z-values are omitted), the interpolation effect is shown in Figure 5-3.

Accuracy analysis: maximum error 0.0625, minimum error 0, average error 0.0216.

Example 5.3 Figure 5-4 shows another effect drawing of AD interpolation, with sampled data (omitted) taken from the classic peaks function in MATLAB.

Accuracy analysis: maximum error 0.3177, minimum error 0, average error 0.0224.

5.2 Three-Dimensional AD Interpolation

Suppose data points of a trivariate function $u = f(x, y, z)$, denoted as $((x_i, y_j, z_k), u_{ijk})$ for $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$, and $k = 1, 2, \ldots, l$, are known, and the base points (x_i, y_j, z_k) are regularly distributed, satisfying $a_x = x_1 < x_2 < \cdots < x_n = b_x$, $a_y = y_1 < y_2 < \cdots < y_m = b_y$, and $a_z = z_1 < z_2 < \cdots < z_l = b_z$. These points form an $m \times n \times l$ "three-dimensional matrix" (as shown in Figure 5-5) and a Cartesian grid in x-y-z space (as shown in Figure 5-6). We consider the corresponding interpolation problem.

For three-dimensional interpolation, we again use a technique similar to two-dimensional interpolation. Specifically, take base points $(x_1, y_1, z_1), \ldots, (x_n, y_m, z_l)$ as reference points, and partition the region $U = [a_x, b_x] \times [a_y, b_y] \times [a_z, b_z]$ into $(2n-2) \times (2m-2) \times (2l-2)$ subregions as shown in Figure 5-6, serving as interpolation regions. Then separate the interpolation computation of evaluation point (x, y, z) with its nearest base point (x_i, y_j, z_k) into three one-dimensional interpolation computations: x with x_i, y with y_j, and z with z_k, obtaining approximate values u_x, u_y, and u_z, respectively. Finally, synthesize u_x, u_y, and u_z into a value u as the interpolated value at (x, y, z).

While two-dimensional interpolation had 4 formulas separated as 2 pairs, three-dimensional interpolation has 6 formulas separated as 3 pairs:

$$u = \frac{u_{ijk} - u_{i-1,jk}}{x_i - x_{i-1}} x + \frac{x_i u_{i-1,jk} - x_{i-1} u_{ijk}}{x_i - x_{i-1}}, \quad x \in \left[\frac{x_i + x_{i-1}}{2}, x_i \right] \tag{5-6}$$

$$u = \frac{u_{ijk} - u_{i+1,jk}}{x_i - x_{i+1}} x + \frac{x_i u_{i+1,jk} - x_{i+1} u_{ijk}}{x_i - x_{i+1}}, \quad x \in \left[x_i, \frac{x_i + x_{i+1}}{2} \right] \tag{5-7}$$

$$u = \frac{u_{ijk} - u_{i,j-1,k}}{y_j - y_{j-1}} y + \frac{y_j u_{i,j-1,k} - y_{j-1} u_{ijk}}{y_j - y_{j-1}}, \quad y \in \left[\frac{y_j + y_{j-1}}{2}, y_j \right] \tag{5-8}$$

$$u = \frac{u_{ijk} - u_{i,j+1,k}}{y_j - y_{j+1}} y + \frac{y_j u_{i,j+1,k} - y_{j+1} u_{ijk}}{y_j - y_{j+1}}, \quad y \in \left[y_j, \frac{y_j + y_{j+1}}{2} \right] \tag{5-9}$$

$$u = \frac{u_{ijk} - u_{ij,k-1}}{z_k - z_{k-1}} z + \frac{z_k u_{ij,k-1} - z_{k-1} u_{ijk}}{z_k - z_{k-1}}, \quad z \in \left[\frac{z_k + z_{k-1}}{2}, z_k\right] \tag{5-10}$$

$$u = \frac{u_{ijk} - u_{ij,k+1}}{z_k - z_{k+1}} z + \frac{z_k u_{ij,k+1} - z_{k+1} u_{ijk}}{z_k - z_{k+1}}, \quad z \in \left[z_k, \frac{z_k + z_{k+1}}{2}\right] \tag{5-11}$$

where $u_{ijk} = f(x_i, y_j, z_k)$.

Similar to two-dimensional AD interpolation, for an evaluation point $(x, y, z) \in U$, we only need to select one formula with respect to x, one with respect to y, and one with respect to z. The specific three formulas used should be determined by the subintervals in which x, y, and z respectively lie. For example, if $x \in [\frac{x_i+x_{i-1}}{2}, x_i]$, $y \in [y_j, \frac{y_j+y_{j+1}}{2}]$, and $z \in [\frac{z_k+z_{k-1}}{2}, z_k]$, then Formulas (5-6), (5-9), and (5-10) should be used. By Formula (3-12) in Section 3.2, the formula synthesizing u_x, u_y, and u_z is

$$u = u_x + u_y + u_z - 2u_{ijk} \tag{5-12}$$

Example 5.4 Use AD interpolation for the trivariate function $u = -x^2 - y^2 - z^2$.

We take sampled data points $((x_i, y_j, z_k), u_{ijk})$ as follows: - x_i: $-3, -2.5, -2.0, \dots, -0.5, 0, 0.5, \dots, 2.0, 2.5, 3$
- y_j: $-4, -3.5, -3.0, \dots, -0.5, 0, 0.5, \dots, 3.0, 3.5, 4$ - z_k: $-4, -3.5, -3.0, \dots, -0.5, 0, 0.5, \dots, 3.0, 3.5, 4$
- u_{ijk}: $u = -x_i^2 - y_j^2 - z_k^2$

Evaluation points (x, y, z) are: - x: $-3, -2.75, -2.50, \dots, -0.75, 0, 0.25, \dots, 2.50, 2.75, 3$
- y: $-4, -3.80, -3.60, \dots, -0.80, 0, 0.20, \dots, 3.60, 3.80, 4$ - z: $-4, -3.75, -3.50, \dots, -0.75, 0, 0.25, \dots, 3.50, 3.75, 4$

Performing AD interpolation (the corresponding u-values are omitted), the interpolation effect (slice chart) is shown in Figure 5-7.

Accuracy analysis: maximum error 0.1850, minimum error 0, average error 0.0993.

5.3 N-Dimensional AD Interpolation

Generalizing the two-dimensional and three-dimensional AD interpolations yields a general n-dimensional AD interpolation method.

Suppose data points of an n-variate function $y = f(x_1, x_2, \dots, x_n)$, denoted as $((x_{1i}, x_{2j}, \dots, x_{nk}), y_{ij\dots k})$ for $i = 1, 2, \dots, r$, $j = 1, 2, \dots, s$, and $k = 1, 2, \dots, t$, are known, and the base points $(x_{1i}, x_{2j}, \dots, x_{nk})$ are regularly distributed. The procedure for n-dimensional AD interpolation is:

1. Partition the n-dimensional interpolation space $U = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n] = [x_{11}, x_{1r}] \times [x_{21}, x_{2s}] \times \cdots \times [x_{n1}, x_{nt}]$ according to the approximation domains of all base points $x_{i,j,\dots,k} = (x_{1i}, x_{2j}, \dots, x_{nk})$.

2. For evaluation point $x = (x_1, x_2, \dots, x_n) \in U$, locate the approximation domain it lies in (equivalent to finding its nearest base point $x_{i,j,\dots,k}$).
3. Separate the n-dimensional interpolation computation of point x with base point $x_{i,j,\dots,k}$ into n one-dimensional interpolation computations: x_1 with x_{1i}, x_2 with x_{2j}, ..., and x_n with x_{nk}, performing each using the corresponding one-dimensional interpolation formulas.
4. Synthesize the n obtained approximate values $y_{x_1}, y_{x_2}, \dots, y_{x_n}$ into a single value y as the interpolated value at point x.

N-dimensional AD interpolation has n pairs of interpolation formulas:

$$y = \frac{y_{ij\dots k} - y_{i-1,j\dots k}}{x_{1i} - x_{1,i-1}} x_1 + \frac{x_{1i} y_{i-1,j\dots k} - x_{1,i-1} y_{ij\dots k}}{x_{1i} - x_{1,i-1}}, \quad x_1 \in \left[\frac{x_{1i} + x_{1,i-1}}{2}, x_{1i}\right] \tag{5-13}$$

$$y = \frac{y_{ij\dots k} - y_{i+1,j\dots k}}{x_{1i} - x_{1,i+1}} x_1 + \frac{x_{1i} y_{i+1,j\dots k} - x_{1,i+1} y_{ij\dots k}}{x_{1i} - x_{1,i+1}}, \quad x_1 \in \left[x_{1i}, \frac{x_{1i} + x_{1,i+1}}{2}\right] \tag{5-14}$$

$$y = \frac{y_{ij\dots k} - y_{i,j-1,\dots k}}{x_{2j} - x_{2,j-1}} x_2 + \frac{x_{2j} y_{i,j-1,\dots k} - x_{2,j-1} y_{ij\dots k}}{x_{2j} - x_{2,j-1}}, \quad x_2 \in \left[\frac{x_{2j} + x_{2,j-1}}{2}, x_{2j}\right] \tag{5-15}$$

$$y = \frac{y_{ij\dots k} - y_{i,j+1,\dots k}}{x_{2j} - x_{2,j+1}} x_2 + \frac{x_{2j} y_{i,j+1,\dots k} - x_{2,j+1} y_{ij\dots k}}{x_{2j} - x_{2,j+1}}, \quad x_2 \in \left[x_{2j}, \frac{x_{2j} + x_{2,j+1}}{2}\right] \tag{5-16}$$

$$\vdots$$

$$y = \frac{y_{ij\dots k} - y_{ij\dots k-1}}{x_{nk} - x_{n,k-1}} x_n + \frac{x_{nk} y_{ij\dots k-1} - x_{n,k-1} y_{ij\dots k}}{x_{nk} - x_{n,k-1}}, \quad x_n \in \left[\frac{x_{nk} + x_{n,k-1}}{2}, x_{nk}\right] \tag{5-17}$$

$$y = \frac{y_{ij\dots k} - y_{ij\dots k+1}}{x_{nk} - x_{n,k+1}} x_n + \frac{x_{nk} y_{ij\dots k+1} - x_{n,k+1} y_{ij\dots k}}{x_{nk} - x_{n,k+1}}, \quad x_n \in \left[x_{nk}, \frac{x_{nk} + x_{n,k+1}}{2}\right] \tag{5-18}$$

where $y_{ij\dots k} = f(x_{1i}, x_{2j}, \dots, x_{nk})$.

The synthesizing formula for the final interpolated value is

$$y = \sum_{i=1}^{n} y_{x_i} - (n-1)y_0 \tag{5-19}$$

Here y_{x_i} are the approximate values obtained by one-dimensional AD interpolation computations of x_i $(i = 1, 2, \ldots, n)$. Formula (5-19) is the Sum-Times-Difference formula in AD interpolation.

It can be seen that in theory, the dimensionality n of n-dimensional AD interpolation has no upper limit. Using the "first separating then synthesizing" technique and the Sum-Times-Difference formula, we can realize high-dimensional and even super-high-dimensional AD interpolations with more than 3 dimensions. In fact, the author has implemented AD interpolation for more than 10 dimensions, and the dimension can be easily increased (due to space limitations, relevant content will be introduced in another article).

If we substitute the interpolation formulas into the Sum-Times-Difference formula, it is not difficult to see that the Sum-Times-Difference formula is actually a linear combination of all coordinate components of point $(x_1, x_2, \ldots, x_n)$. For example, substituting the two-dimensional interpolation formulas

$$z = \frac{z_{ij} - z_{i-1,j}}{x_i - x_{i-1}} x + \frac{x_i z_{i-1,j} - x_{i-1} z_{ij}}{x_i - x_{i-1}}$$

and

$$z = \frac{z_{ij} - z_{i,j-1}}{y_j - y_{j-1}} y + \frac{y_j z_{i,j-1} - y_{j-1} z_{ij}}{y_j - y_{j-1}}$$

into the corresponding Sum-Times-Difference formula $z = z_x + z_y - z_{ij}$ yields

$$z = \frac{z_{ij} - z_{i-1,j}}{x_i - x_{i-1}} x + \frac{z_{ij} - z_{i,j-1}}{y_j - y_{j-1}} y + \left(\frac{x_i z_{i-1,j} - x_{i-1} z_{ij}}{x_i - x_{i-1}} + \frac{y_j z_{i,j-1} - y_{j-1} z_{ij}}{y_j - y_{j-1}} - z_{ij} \right)$$

Therefore, the Sum-Times-Difference formula can also be regarded as the interpolation formula for multidimensional AD interpolation.

Although AD interpolation is essentially a local linear interpolation, it is derived based on approximate evaluation of functions using approximation-degree. The base points involved in interpolation are related to the position of evaluation point $x \in U$ relative to its nearest base point $x_{i,j,\ldots,k}$. It can be seen that n-dimensional AD interpolation involves a total of $1 + n$ base points, and when dimensionality increases by 1, only 1 additional base point is added (see Figures 3-7, 3-8, 5-1, and 5-5). Since point x is only approximate to base point

$x_{i,j,\ldots,k}$, the n approximate values $y_{x_1}, y_{x_2}, \ldots, y_{x_n}$ obtained from separated one-dimensional interpolations are most affected by $x_{i,j,\ldots,k}$, and the final interpolated value y is also most affected by $x_{i,j,\ldots,k}$. In other words, AD interpolation is dominated by the nearest base point of the evaluation point.

6 Performance Analysis

6.1 Accuracy

From the one-dimensional interpolation formulas and the Sum-Times-Difference formula for multidimensional interpolation, it is evident that AD interpolation is essentially a piecewise (block) linear interpolation. Therefore, the accuracy of AD interpolation should be at the level of linear interpolation. Data experiments show that the accuracy of 1-D, 2-D, and 3-D AD interpolations is about the same as traditional linear interpolation, especially for quadratic functions (such as in previous examples), where the accuracy is exactly identical.

Additionally, we find that for an interpolated function $y = f(x_1, x_2, \ldots, x_n)$ that is monotonic on each interpolation subinterval, the theoretical maximum error of AD interpolation is

$$E = \left| f\left(\frac{x_{1,i+1} + x_{2,i+1}}{2}, \ldots, \frac{x_{n,i+1} + x_{n,i+1}}{2}\right) - f\left(\frac{x_{1,i+1} + x_{2,i+1}}{2}, \ldots, \frac{x_{n,i+1} + x_{n,i+1}}{2}\right)\right|$$

where $f(\mathbf{x})$ and $\hat{f}(\mathbf{x})$ are respectively the interpolated value and desired value at point $x = (x_1, x_2, \ldots, x_n)$.

From the principle of AD interpolation, it is clear that accuracy depends on two factors: the size of the approximation domain of the relevant base point and the approximation-degree of the evaluation point x to its nearest base point. Obviously, the narrower the approximation domain (including both relative and absolute approximation domains), the higher the interpolation accuracy. When an approximation domain is fixed, the higher the approximation-degree of point x to the corresponding base point, the higher the accuracy of the interpolated value.

Thus, for regularly distributed base points, as long as there are sufficient data points (i.e., sufficient samples), any required accuracy can be achieved using AD interpolation.

6.2 Efficiency

From a theoretical perspective, the main difference between AD interpolation and traditional linear interpolation lies in multidimensional cases. In multi-dimensional AD interpolation, using the "first separating then synthesizing" technique, an n-dimensional AD interpolation computation is separated into n one-dimensional AD interpolation computations performed respectively, and

the results are synthesized into a single value using the Sum-Times-Difference formula.

Thus, one n-dimensional AD interpolation computation becomes n one-dimensional AD interpolation computations plus one synthesizing computation. From the Sum-Times-Difference formula, all multidimensional AD interpolations have only two layers of computation: one-dimensional AD interpolation computations and synthesizing computation (as shown in Figure [Figure 6: see original paper]-1(a)). When dimensionality increases by 1, only one additional one-dimensional AD interpolation computation and one addition operation are added (as shown in Figure 6-1(b)). That is, the two-layer computing structure of AD interpolation remains unchanged, but the number of computation terms in the lower layer increases with data dimensionality. The number of formula computations (including interpolation formulas and Sum-Times-Difference formula) in n-dimensional ($n > 1$) AD interpolation is $n + 1$.

Starting from two dimensions, the numbers of formula computations in AD interpolation are in order: 3, 4, 5, ⋯, forming an arithmetic sequence with difference 1, with general term $n+1$. Denoting the number of formula computations in n-dimensional ($n > 1$) AD interpolation as N_n, the recurrence formula is $N_{n+1} = N_n + 1$. This shows that the computational load of multidimensional AD interpolation is much smaller than that of traditional multidimensional linear interpolation.

Furthermore, since the n one-dimensional AD interpolation computations obtained from separation are independent of each other, parallel computation can be employed. Thus, the efficiency of n-dimensional AD interpolation is almost equal to that of one-dimensional AD interpolation. In summary, AD interpolation exhibits high efficiency, with higher dimensions yielding greater efficiency advantages.

7 Summary and Prospect

Inspired by the approximate evaluation method of flexible linguistic functions in reference [1], this paper introduces the measure of approximation-degree and the concept of approximation-degree functions between numerical values. Based on the "approximation-degree transmission axiom"and through computing approximation-degrees, evaluating inverse functions, and selecting approximate values, we realize the approximate evaluation of numerical functions. Especially for multivariate functions, in the sense of "strict approximation," we employ the "first separating then synthesizing" technique and derive the "Sum-Times-Difference" formula to achieve more skillful approximate evaluation. On this basis, we further derive a set of interpolation formulas for regularly distributed base points, developing an interpolation method based on approximation-degree —approximation-degree-based interpolation (AD interpolation). We then extend one-dimensional AD interpolation to high-dimensional cases.

Testing AD interpolation on actual functions yields satisfactory results: one-

dimensional AD interpolation achieves the same effect as conventional piecewise linear interpolation through a different approach, while two-dimensional and three-dimensional AD interpolations are comparable to conventional bilinear and trilinear interpolations. High-dimensional AD interpolations with more than three dimensions have also passed data tests.

AD interpolation is essentially a linear interpolation method with accuracy similar to traditional (piecewise) linear interpolation, but its principle differs. One-dimensional AD interpolation uses corresponding interpolation formulas directly; n-dimensional ($n > 1$) AD interpolation first separates into n one-dimensional AD interpolation computations (which can be processed in parallel) and then synthesizes the results using the Sum-Times-Difference formula. AD interpolation is dominated by the nearest base point and assisted by its adjacent base point(s). n-dimensional AD interpolation involves $1 + n$ base points, meaning that when dimensionality increases by 1, only one additional correlated base point is added. The n-dimensional AD interpolation has two computation layers: parallel one-dimensional AD interpolation computations followed by synthesizing computation. When dimensionality increases by 1, only one additional one-dimensional AD interpolation computation and one addition operation are added. Compared with traditional linear interpolation, AD interpolation' s computational load is much smaller, and with parallel processing, its efficiency is almost the same as one-dimensional AD interpolation. Moreover, the dimensionality of multidimensional AD interpolation is theoretically unlimited (limited only by computational resources).

In summary, AD interpolation offers novel ideas, unique methods, simple computation, adjustable accuracy, high efficiency, and unlimited dimensionality. This opens a new path for data fitting and function approximation, especially providing a feasible and effective method for high-dimensional and super-high-dimensional interpolations.

This paper expounds the basic principles of AD interpolation. Future work can include: - Applying AD interpolation to practical high-dimensional and super-high-dimensional problems - AD interpolation with irregularly distributed base points - AD interpolation for vector-valued functions - AD interpolation with derivatives (partial and directional) - Using AD interpolation to realize a function approximator

The approximator using AD interpolation is always interpretable and avoids the "dilemma between precision and interpretability" [7] encountered by function approximators based on fuzzy technology. Additionally, introducing AD interpolation into machine learning can realize a new instance-based learning method —AD interpolation learning. The author has conducted preliminary research on these topics, which will be discussed separately due to space constraints.

References

[1] Shiyou Lian, 2016. *Principles of Imprecise-Information Processing: A New*

Theoretical and Technological System, Springer Nature.

[2] David Kincaid and Ward Cheney, 2002. *Numerical Analysis: Mathematics of Scientific Computing* (Third Edition), Thomson Learning.

[3] Tom M. Mitchell, 1997. *Machine Learning*, McGraw-Hill Companies, Inc.

[4] Ethem Alpaydin, 2014. *Introduction to Machine Learning* (Third Edition), Massachusetts Institute of Technology, MIT Press.

[5] Shiyou Lian, 2016. Correspondence between Flexible Sets and Flexible Linguistic Functions, in: Shiyou Lian, *Principles of Imprecise-Information Processing: A New Theoretical and Technological System*, Springer Nature, pp. 205–228.

[6] Shiyou Lian, 2016. Approximate Evaluation of Flexible Linguistic Functions, in: Shiyou Lian, *Principles of Imprecise-Information Processing: A New Theoretical and Technological System*, Springer Nature, pp. 393–417.

[7] Jyh-Shing Roger Jang, Chuen-Tsai Sun, Eiji Mizutani, 1997. *Neuro-Fuzzy and Soft Computing* (Prentice Hall, Upper Saddle River, NJ), pp. 342–245, 382–385.

Note: Figure translations are in progress. See original paper for figures.